FUJITSU

FUJITSU Software
Symfoware Server V12.7.0

SQLリファレンス

Windows/Solaris/Linux

J2UL-1764-12Z0(00)
2023年2月

まえがき

本書の目的

本書は、Symfoware Server SQLリファレンスです。本書は、データベースの作成およびデータ操作のためのプログラムの開発に使用するSQL文の文法について説明しています。

本書の読者

本書は、Symfoware/RDBのデータベースを作成する方およびSymfoware/RDBのデータベースを処理するアプリケーションを開発する方に読んでいただくように書かれています。本書を読むには、以下に示す技術および知識が必要です。

- Symfoware/RDBの機能およびデータベースの概要をある程度理解している
- Symfoware/RDBを適用する業務についてある程度知識を持っている
- C言語またはCOBOLでアプリケーションを開発することができる
- S **Solarisの場合**
  Oracle Solarisを使用できる
- L **Linuxの場合**
  Linuxシステムを使用できる
- W **Windowsの場合**
  Windows(R)システムを使用できる

本書の構成

本書の構成と内容は以下のとおりです。

第1章 文法の表記方法と文/要素一覧

第2章以降のSQL文法の表記方法を説明しています。また、SQL文および要素の一覧表を記載しています。

第2章 共通要素の文法規則

SQL文に指定する要素のうち、共通要素について説明しています。

第3章 基本的なSQL文

データベース定義文、データ操作文など基本的なSQL文の文法について説明しています。

第4章 動的SQL文

アプリケーションの実行時にSQL文を動的に指定できる動的SQL文の文法を説明しています。

第5章 ストアドプロシジャ

ストアドプロシジャの文法を説明しています。

第6章 埋込みSQL

コンパイラ(C言語およびCOBOL)でアプリケーションを開発する場合のSQL文の文法を説明しています。

第7章 SQL拡張インタフェース

SQL拡張インタフェースの関数の文法を説明しています。

付録

付録A SQLSTATE値

処理結果としてアプリケーションに通知されるSQLSTATEについて説明しています。

付録B キーワード一覧

トークン中のキーワードの一覧を示しています。

付録C 定量制限

Symfoware/RDBの定量制限の一覧を示しています。

付録D SQL文の省略値に関する注意事項

Symfoware/RDBのSQL文の一般規則に関するチューニング、SQL文で使用する名前に関する注意事項および格納構造定義を行わない場合の格納構造について説明しています。

付録E SQL文の使用範囲

Symfoware/RDBがコンパイル・実行時にサポートするSQL文の使用範囲について説明しています。

付録F 定義文の実行時間に関する注意事項

定義文の実行時間に関する注意事項について説明しています。

付録G SQL規格に対するSymfoware Serverの準拠性

SQL規格のコア機能とオプション機能に対するSymfoware Serverの準拠性について説明しています。

本書の読み方

本書は、Symfoware/RDBを使用してデータベースを実際に作成するとき、またはアプリケーションを実際に開発するときに、リファレンスとして使用することを目的として書かれています。

初めてSymfoware/RDBを使用する場合、またはSQL文をあまりよく知らない場合には、“アプリケーション開発ガイド(共通編)”および“RDB運用ガイド(データベース定義編)”を読んで、データベースの概要、SQL文の機能と指定の仕方およびアプリケーションの開発方法を理解してから本書をお読みください。

本書に記載するアプリケーションおよびSQL文の記述の例は、特にことわらない限りC言語での記述を使用しています。COBOLを使用する場合に、Cでの記述と特に異なる点については、“第6章 埋込みSQL”で説明しています。

SQLについてはすでに理解していて、実際にデータベースを作成する場合、アプリケーションを開発する場合、またはSQLの詳細な文法を確認する場合には、SQL文を使用するためのリファレンスとしてご利用ください。

関連マニュアル

本書に関連するマニュアルは以下のとおりです。

- NetCOBOL使用手引書
- COBOL文法書

輸出管理規制について

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

出版年月および版数

2023年　2月	第12版
2021年 10月	第11版
2021年　6月	第10版
2020年 10月	第9版
2019年　2月	第8版
2016年　9月	第7版
2015年 12月	第6版
2015年 10月	第5版
2014年 10月	第4版
2014年　8月	第3版
2013年 11月	第2版
2013年　9月	初版

著作権

目　次

第1章 文法の表記方法と文/要素一覧

本章では、共通要素といくつかに分類したSQL文およびSQL拡張インタフェースの関数の、表記方法を説明します。

記述形式は、構文図を用いてわかりやすく説明します。記述形式だけでは表現できない場合には、一般規則で補足します。また、機能の詳細なども一般規則に示します。

各SQL文は、共通要素やほかのSQL文の要素の組合せで構成しています。あるSQL文がほかの要素を参照している場合には、記述形式の直後に参照する要素の項番を示しています。また、文/要素一覧表では、共通要素、SQL文の簡単な機能および本書での参照先をまとめてあります。

1.1 文法の表記方法

SQL文の文法の表記方法およびSQL拡張インタフェースの関数の表記方法について説明します。

1.1.1 SQL文の表記方法

SQL文の文法規則の説明形式は、原則として次の項目に分けて説明してあります。

- 機能
- 記述形式
- 参照項番
- 権限
- 一般規則
- 使用例

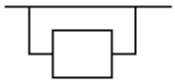
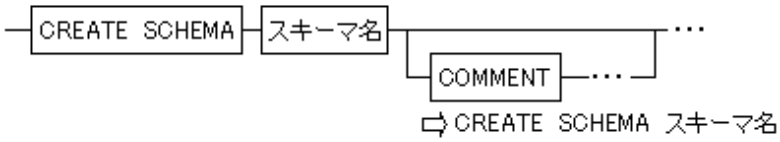
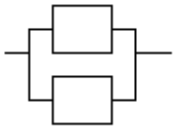
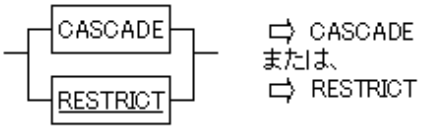
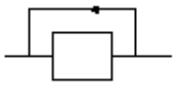
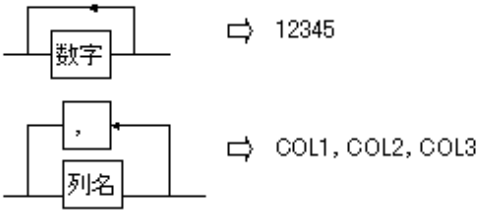
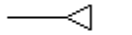
機能

要素が持つ機能を説明します。

記述形式

文を構成する各要素の並べ方を示します。

記述形式で使用している記号の意味を以下に示します。

記号	意　味
	線の下に分岐している部分は、省略可能な要素を示しています。 例: 
	上下に並列に分岐した部分は、そのうちの1つを選択する要素を示しています。 また、省略値となる要素には下線を付加しています。 例: 
	線の上に延びている矢印付きの線は、その要素が繰り返されることを示しています。 例: 
	構文が下段に続くことを示しています。
	構文が完結することを示しています。

構文要素の構成または構文の構成

第2章の“構文要素の構成”、第2章以外の“構文の構成”は、“記述形式”に対する補助的な説明です。説明している用語は、複数の構文要素をまとめて表現するSQL文の用語です。説明している用語は“一般規則”や“メッセージ”で使われています。

例

CREATE SCHEMA文の“構文の構成”の例を以下に示します。二重線上に記述した“注釈定義”と“スキーマ要素”が記述形式の構文図では表現されない用語です。

参照項番

文を構成する要素がほかの項で説明されている場合、その項番を示します。

権限

SQL文の実行に必要な権限を説明しています。

一般規則

要素の文法、注意事項や要素間の意味の相互関係を説明しています。

使用例

各要素の具体的な記述例を示しています。ただし、単純な規則しか持たない文に対しては、例を省略していることがあります。

使用例に用いるデータベースを以下に示します。なお、表に含まれるデータは架空のものです。

図1.1 データベースの構成

a) 在庫表

製品番号	製 品 名	在庫数量	倉庫番号
110	テレビ	85	2
111	テレビ	90	2
123	冷蔵庫	60	1
124	冷蔵庫	75	1
140	CDプレーヤー	120	2
212	テレビ	0	2
215	ビデオ	5	2
226	冷蔵庫	8	1
227	冷蔵庫	15	1
240	CDプレーヤー	25	2
243	CDプレーヤー	14	2
351	CD	2500	2

b) 発注表

取引先	取引製品	仕入価格	発注数量
61	123	48000	60
61	124	64000	40
61	140	8000	80
61	215	240000	10
61	240	80000	20
62	110	37500	120
62	226	112500	20
62	351	375	800
63	111	57400	80
63	212	205000	30
63	215	246000	10
71	140	7800	50
71	351	390	600
72	140	7000	70
72	215	210000	10
72	226	105000	20
72	243	84000	10
72	351	350	1000
74	110	39000	120
74	111	54000	120
74	226	117000	20
74	227	140400	10
74	351	390	700

c) 会社表

会社番号	会社名	電話番号	住所
61	アダム電気	111－777－4444	東京都 小田区 新蒲田 7－8－9
62	アイデア商事	222－888－5555	神奈川県 小浜市 旭区 1－2－3
63	大月産業	333－999－6666	埼玉県 浦和町 大崎 1－1－1
71	川川電気	444－111－7777	大阪府 堺田市 赤坂台 4－5－6
72	龍巻産業	555－222－8888	大阪府 灘町 東淀川 2－3－7
74	第第商事	666－333－9999	京都府 中市 4－16－16

1.1.2 関数の表記方法

SQL拡張インタフェースの関数の説明形式は、原則として次の項目に分けて説明してあります。

- 機能
- 記述形式
- 一般規則
- 異常時の対処

機能

関数の機能を説明します。

記述形式

関数の記述形式を示します。

一般規則

関数の引数の文法、注意事項を説明しています。

異常時の対処

関数の復帰コードと対処方法を説明します。

1.2 SQLと関数の一覧

SQLの文/要素一覧

各節、各項で説明している内容の検索が便利なようにSQLの文/要素の一覧を“表1.1 文/要素の一覧”に示します。簡単な機能とそれを説明している本文の項番号を知ることができます。

“表1.1 文/要素の一覧”の見方は次のとおりです。

文/要素:

各項目名が記述されています。

機能:

文/要素の機能を簡単に説明してあります。

項番:

対応する文/要素の説明がある本書中の項番号です。

“表1.1 文/要素の一覧”は以下の項目から構成されています。

- 共通要素の文法規則
- データベース定義文 (注)
- データベース操作文 (注)
- スキーマ定義文
- スキーマ操作文
- 格納構造定義文
- 格納構造操作文
- データ操作文
- トランザクション管理文
- コネクション管理文
- セション管理文
- 資源操作文
- 利用者制御文
- アクセス制御文
- システム制御文
- 動的SQL文
- ストアドプロシジャ
- 埋込みSQL

注) Symfoware Server クライアント機能では、データベース定義文およびデータベース操作文は利用できません。

表1.1 文/要素の一覧

項目	文／要素		機能	項番
共通要素の文法規則	文字		SQL文で使用可能な文字を規定します。	2.1.1 文字
	定数		NULLでない値を指定します。	2.1.2 定数
	トークン		SQL文を構成する識別子、キーワード、各種定数などの最小単位の要素を指定します。	2.1.3 トークン
	名前		種々の名前を指定します。	2.1.4 名前
	データ型		データの型を定義します。	2.2 データ型
	値指定と相手指定		値および変数を指定します。	2.3 値指定と相手指定
	列指定		名前で指定された列を参照します。	2.4 列指定
	関数		さまざまなデータ型の関数を指定します。	2.5 関数
		集合関数指定	引数に関数を適用することによって導出される値を指定します。	2.5.1 集合関数指定
		数値関数	数値型の値をとる関数を指定します。	2.5.2 数値関数
		データ列値関数	文字列値の値をとる関数を指定します。	2.5.3 データ列値関数
		日時値関数	日時型の値となる関数を指定します。	2.5.4 日時値関数
		ファンクションルーチン指定	ファンクションルーチンを指定します。	2.5.5 ファンクションルーチン指定
		XMLQUERY関数	XQuery式を実行し、XMLデータを返却する関数を指定します。	2.5.6 XMLQUERY関数
	CASE式		条件付けられた値を指定します。	2.6 CASE式
	CAST指定		データ変換を指定します。	2.7 CAST指定
	順序		順序を指定します。	2.8 順序
	ROWNUM		ROWNUMには、FROM句の結果の表に対して、WHERE句の探索条件を適用した結果の行に、1から振った順番号が入ります。	2.9 ROWNUM
	行値構成子		順序付けられた列の並びを指定します。	2.10 行値構成子
	値式		値指定、列指定、集合関数指定、またはこれらの演算結果による、値を指定します。	2.11 値式
		数値式	数値を指定します。	2.11.1 数値式
		データ列値式	文字列値を指定します。	2.11.2 データ列値式
		日時値式	日時値を指定します。	2.11.3 日時値式
		時間隔値式	時間隔値を指定します。	2.11.4 時間隔値式
	述語		“真”、“偽”または“不定”の真偽値の評価ができる条件を指定します。	2.12 述語
		比較述語	2つの値の比較を指定します。	2.12.1 比較述語
		BETWEEN述語	範囲比較を指定します。	2.12.2 BETWEEN述語
		IN述語	ある限定された値の集合について比較を指定します。	2.12.3 IN述語

項目	文／要素		機能	項番
		LIKE述語	文字型データについて、指定された文字パターンの照合比較を指定します。	2.12.4 LIKE述語
		NULL述語	NULL値との比較を指定します。	2.12.5 NULL述語
		限定述語	限定された値の集合との比較を指定します。	2.12.6 限定述語
		EXISTS述語	空集合との比較を指定します。	2.12.7 EXISTS述語
		XMLEXISTS述語	XQuery式に指定した検索式に一致するXMLデータであるかを評価します。	2.12.8 XMLEXISTS述語
	探索条件		ブール演算子を適用した結果によって、“真”、“偽”または“不定”となる条件を指定します。	2.13 探索条件
	副問合せ		表式の結果から導出される値の集合を指定します。	2.14 副問合せ
	ASSIST指定		SQL文を実行する際のアクセスプランを固定化する時に指定します。	2.15 ASSIST指定
データベース定義文	CREATE DATABASE文(データベース定義)		データベースを定義します。	3.9 CREATE DATABASE文(データベース定義)
	CREATE DBSPACE文(データベーススペース定義)		データベーススペースを定義します。	3.10 CREATE DBSPACE文(データベーススペース定義)
データベース操作文	DROP DATABASE文(データベース削除文)		データベースを削除します。	3.31 DROP DATABASE文(データベース削除文)
	DROP DBSPACE文(データベーススペース削除文)		データベーススペースを削除します。	3.32 DROP DBSPACE文(データベーススペース削除文)
スキーマ定義文	CREATE SCHEMA文(スキーマ定義)		スキーマを定義します。	3.19 CREATE SCHEMA文(スキーマ定義)
	CREATE TABLE文(表定義)		表を定義します。	3.22 CREATE TABLE文(表定義)
	CREATE VIEW文(ビュー定義)		ビュー表を定義します。	3.25 CREATE VIEW文(ビュー定義)
	CREATE PROCEDURE文(プロシジャルーチン定義)		プロシジャルーチンを定義します。	3.17 CREATE PROCEDURE文(プロシジャルーチン定義)
	CREATE FUNCTION文(ファンクションルーチン定義)		ファンクションルーチンを定義します。	3.15 CREATE FUNCTION文(ファンクションルーチン定義)

項目	文／要素	機能	項番
	CREATE INDEX文 (インデックス定義)	インデックスを定義します。	3.16 CREATE INDEX文(インデックス定義)
	CREATE TRIGGER文 (トリガ定義)	トリガを定義します。	3.23 CREATE TRIGGER文(トリガ定義)
	CREATE SEQUENCE文 (順序定義)	順序を定義します。	3.21 CREATE SEQUENCE文(順序定義)
スキーマ操作文	DROP SCHEMA文 (スキーマ削除文)	スキーマを削除します。	3.39 DROP SCHEMA文(スキーマ削除文)
	DROP TABLE文 (表削除文)	表を削除します。	3.42 DROP TABLE文(表削除文)
	ALTER TABLE文 (表定義変更文)	表とその定義を変更します。	3.2 ALTER TABLE文(表定義変更文)
	DROP VIEW文 (ビュー削除文)	ビュー表を削除します。	3.45 DROP VIEW文(ビュー削除文)
	DROP PROCEDURE文 (プロシジャルーチン削除文)	プロシジャルーチンを削除します。	3.37 DROP PROCEDURE文(プロシジャルーチン削除文)
	DROP INDEX文 (インデックス削除文)	インデックスを削除します。	3.36 DROP INDEX文(インデックス削除文)
	DROP TRIGGER文 (トリガ削除文)	トリガを削除します。	3.43 DROP TRIGGER文(トリガ削除文)
	SWAP TABLE文 (表交換文)	指定された2つの表について、表名を交換します。	3.65 SWAP TABLE文(表交換文)
	DROP SEQUENCE文 (順序削除文)	順序を削除します。	3.41 DROP SEQUENCE文(順序削除文)
	DROP FUNCTION文 (ファンクションルーチン削除文)	ファンクションルーチンを削除します。	3.35 DROP FUNCTION文(ファンクションルーチン削除文)
格納構造定義文	CREATE DSO文 (表のDSO定義文)	表のDSOを定義します。	3.14 CREATE DSO文(表のDSO定義文)
	CREATE DSO文 (インデックスのDSO定義文)	インデックスのDSOを定義します。	3.13 CREATE DSO文(インデックスのDSO定義文)
	CREATE DSI文 (表のDSI定義文)	表のDSO定義に従って、表とデータベーススペースとの割付けを定義します。	3.12 CREATE DSI文(表のDSI定義文)

項目	文／要素		機能	項番
	CREATE DSI文 (インデックスのDSI定義文)		インデックスのDSO定義に従って、インデックスとデータベーススペースとの割付けを定義します。	3.11 CREATE DSI文(インデックスのDSI定義文)
	CREATE SCOPE文 (スコープ定義文)		データ操作の範囲(スコープ)を定義します。	3.20 CREATE SCOPE文(スコープ定義文)
格納構造操作文	DROP DSO文 (DSO削除文)		DSOを削除します。	3.34 DROP DSO文(DSO削除文)
	DROP DSI文 (DSI削除文)		DSIを削除します。	3.33 DROP DSI文(DSI削除文)
	ALTER DSI文 (DSI変更文)		分割値を変更または容量を拡張します。	3.1 ALTER DSI文(DSI変更文)
	DROP SCOPE文 (スコープ削除文)		スコープを削除します。	3.40 DROP SCOPE文(スコープ削除文)
	APPLY SCOPE文 (スコープ適用文)		スコープを表の利用者に適用します。	3.4 APPLY SCOPE文(スコープ適用文)
	RELEASE SCOPE文 (スコープ解除文)		適用されたスコープを解除します。	3.51 RELEASE SCOPE文(スコープ解除文)
	PRINT STATISTICS文 (最適化情報出力文)		最適化情報を出力します。	3.50 PRINT STATISTICS文
	SET STATISTICS文 (最適化情報設定文)		データベースに最適化情報を設定します。	3.60 SET STATISTICS文
データ操作文	非カーソル系	Single row SELECT文 (単一行SELECT文)	表の指定された行から値を取り出します。	3.64 Single row SELECT文(単一行SELECT文)
		DELETE文:探索	表から探索条件を満たす行を削除します。	3.29 DELETE文:探索
		INSERT文	表に新しい行を挿入します。	3.48 INSERT文
		UPDATE文:探索	探索条件を満たす行の列を更新します。	3.67 UPDATE文:探索
	カーソル系	DECLARE CURSOR (カーソル宣言)	カーソルを定義します。	3.26 DECLARE CURSOR(カーソル宣言)
		OPEN文	カーソルを開きます。	3.49 OPEN文
		CLOSE文	カーソルを閉じます。	3.6 CLOSE文
		FETCH文	カーソルを表の次の行に位置づけ、その行から値を取り出します。	3.46 FETCH文
		DELETE文:位置づけ	カーソルによって位置づけられた1行を削除します。	3.28 DELETE文:位置づけ
		UPDATE文:位置づけ	カーソルによって位置づけられた1行を更新します。	3.66 UPDATE文:位置づけ

項目	文／要素		機能	項番
		DECLARE TABLE (表宣言)	修飾なしの表名を定義します。	3.27 DECLARE TABLE(表宣言)
トランザクション管理文	SET TRANSACTION文		トランザクションモードの切替えを行います。	3.62 SET TRANSACTION文
	COMMIT文		現行のデータベースの変更をすべて有効にしてトランザクションを終了させます。	3.7 COMMIT文
	ROLLBACK文		現行のデータベースの変更をすべて無効にしてトランザクションを終了させます。	3.54 ROLLBACK文
コネクション管理文	CONNECT文		コネクションを接続します。	3.8 CONNECT文
	SET CONNECTION文		現コネクションを変更します。	3.56 SET CONNECTION文
	DISCONNECT文		コネクションを切断します。	3.30 DISCONNECT文
セション管理文	SET CATALOG文		動的SQLの被準備文の対象となるデータベース名を設定します。	3.55 SET CATALOG文
	SET SCHEMA文		動的SQLの被準備文の省略したスキーマ名を設定します。	3.58 SET SCHEMA文
	SET SESSION AUTHORIZATION文		現行セションの対象となる利用者を変更します。	3.59 SET SESSION AUTHORIZATION文
資源操作文	RELEASE TABLE文		一時表を解放します。	3.52 RELEASE TABLE文
利用者制御文	CREATE USER文 (利用者定義文)		利用者を定義します。	3.24 CREATE USER文(利用者定義文)
	DROP USER文 (利用者削除文)		利用者を削除します。	3.44 DROP USER文(利用者削除文)
	ALTER USER文 (利用者変更文)		利用者の属性を変更します。	3.3 ALTER USER文(利用者変更文)
	SET USER PASSWORD文		現行セションの利用者のパスワードを変更します。	3.63 SET USER PASSWORD文
アクセス制御文	CREATE ROLE文 (ロール定義文)		ロールを定義します。	3.18 CREATE ROLE文(ロール定義文)
	DROP ROLE文 (ロール削除文)		ロールを削除します。	3.38 DROP ROLE文(ロール削除文)
	GRANT文		権限を定義します。	3.47 GRANT文
	REVOKE文		権限を削除します。	3.53 REVOKE文
	SET ROLE文		現行SQLセションの対象となる利用者を変更します。	3.57 SET ROLE文
システム制御文	SET SYSTEM PARAMETER文		セキュリティパラメタを設定します。	3.61 SET SYSTEM PARAMETER文

項目	文／要素	機能	項番
動的SQL文	ALLOCATE DESCRIPTOR文	SQL記述子域を割り当てます。	4.3 ALLOCATE DESCRIPTOR文
	DEALLOCATE DESCRIPTOR文	SQL記述子域を解放します。	4.5 DEALLOCATE DESCRIPTOR文
	DESCRIPTOR取得文	SQL記述子域から情報を取得します。	4.14 GET DESCRIPTOR文 (DESCRIPTOR取得文)
	DESCRIPTOR設定文	SQL記述子域に情報を設定します。	4.17 SET DESCRIPTOR文 (DESCRIPTOR設定文)
	PREPARE文	動的に実行するSQL文を準備します。	4.16 PREPARE文
	DEALLOCATE PREPARE文	PREPARE文によって準備されている被準備文を解放します。	4.6 DEALLOCATE PREPARE文
	DESCRIBE文	被準備文の動的パラメタ指定または選択リストについての情報を取り出します。	4.10 DESCRIBE文
	EXECUTE文	被準備文を実行します。	4.11 EXECUTE文
	EXECUTE IMMEDIATE文	SQL文を動的に実行します。	4.12 EXECUTE IMMEDIATE文
	DECLARE CURSOR (動的カーソル宣言)	動的カーソルを定義します。	4.7 DECLARE CURSOR(動的カーソル宣言)
	OPEN文(動的OPEN文)	動的カーソルを開きます。	4.15 OPEN文(動的OPEN文)
	FETCH文 (動的FETCH文)	動的カーソルを表の次の行に位置づけ、その行から値を取り出します。	4.13 FETCH文(動的FETCH文)
	CLOSE文 (動的CLOSE文)	動的カーソルを閉じます。	4.4 CLOSE文(動的CLOSE文)
	DELETE文(動的DELETE文: 位置づけ)	動的カーソル宣言で宣言されたカーソルによって位置づけられた1行を削除します。	4.9 DELETE文(動的DELETE文:位置づけ)
	UPDATE文(動的UPDATE文: 位置づけ)	動的カーソル宣言で宣言されたカーソルによって位置づけられた1行を更新します。	4.19 UPDATE文(動的UPDATE文:位置づけ)
	DELETE文(準備可能動的DELETE文: 位置づけ)	動的カーソル宣言で宣言されたカーソルによって位置づけられた1行を削除します。	4.8 DELETE文(準備可能動的DELETE文:位置づけ)
	UPDATE文(準備可能動的UPDATE文: 位置づけ)	動的カーソル宣言で宣言されたカーソルによって位置づけられた1行を更新します。	4.18 UPDATE文(準備可能動的UPDATE文:位置づけ)
ストアドプロシジャ	CALL文	プロシジャを呼び出します。	3.5 CALL文
	SQL変数宣言	複合文中で使用するSQL変数を宣言します。	5.1 ストアドプロシジャの概要

項目	文／要素	機能	項番
	条件宣言	特定のSQLSTATE値の条件名を宣言します。	5.1 ストアドプロシジャの概要
	ハンドラ宣言	複合文中で使用するハンドラを宣言します。	5.1 ストアドプロシジャの概要
	SET文(代入文)	値を変数に設定します。	5.8 SET文(代入文)
	複合文	複数のSQL文をグループ化します。	5.1 ストアドプロシジャの概要
	IF文	条件付き実行制御を行います。	5.3 IF文
	LOOP文	文の実行を繰り返します。	5.5 LOOP文
	LEAVE文	LOOP文や複合文から脱出します。	5.4 LEAVE文
	WHILE文	条件が真の間、実行文を繰り返します。	5.11 WHILE文
	REPEAT文	条件が真になるまで文の実行を繰り返します。	5.6 REPEAT文
	GOTO文	文ラベルに無条件に分岐します。	5.2 GOTO文
	分岐先文	分岐先である文ラベルです。	5.1 ストアドプロシジャの概要
	WHENEVER文	ルーチン内のSQL手続き文実行時に、例外事象が発生した場合にとる動作を指定します。	5.10 WHENEVER文
	SIGNAL文	例外条件を送信します。	5.9 SIGNAL文
	RESIGNAL文	例外条件を再送します。	5.7 RESIGNAL文
埋込みSQL	SQL埋込みホストプログラム	埋込みSQL適用業務プログラムを指定します。	6.5 SQL埋込みホストプログラム
	Embedded SQL文 (埋込みSQL文)	ホストプログラムにSQL文を埋め込みます。	6.1 Embedded SQL文(埋込みSQL文)
	INCLUDE文	別ファイルのSQL文や対象となる言語テキストを展開します。	6.2 INCLUDE文
	SQL埋込みCプログラム	ホストプログラムとしてC言語を使用するために必要な各種の定義をします。	6.3 SQL埋込みCプログラム
	SQL埋込みCOBOLプログラム	ホストプログラムとしてCOBOLを使用するために必要な各種の定義をします。	6.4 SQL埋込みCOBOLプログラム
	WHENEVER文 (埋込み例外宣言)	SQL文が例外条件を発生したときに、SQL埋込みホストプログラムがとるべき動作を指定します。	6.6 WHENEVER文(埋込み例外宣言)

関数の一覧

SQL拡張インタフェースで説明している内容の検索が便利なように関数の一覧を“表1.2 関数の一覧”に示します。簡単な機能とそれを説明している本文の項番号を知ることができます。

“表1.2 関数の一覧”の見方は次のとおりです。

関数:

各関数名が記述されています。

機能:

関数の機能を簡単に説明してあります。

項番:

関数の説明がある本書中の項番号です。

表1.2 関数の一覧

項目	関数	機能	項番
SQL拡張インタフェース	SQLThrALLocID	プロセス内にセションを作成します。	7.1.1 SQLThrAllocID
	SQLThrEndID	セションとスレッドの関係付けを終了します。	7.1.2 SQLThrEndID
	SQLThrFreeID	セションを破棄します。	7.1.3 SQLThrFreeID
	SQLThrStartID	セションとスレッドの関係付けを行います。	7.1.4 SQLThrStartID
	SQLSignalMSG	プログラムの処理で異常を検出して、Symfoware/RDBに異常復帰します。	7.2.1 SQLSignalMSG
	SQLDynSetCallback	コールバック関数を動的に登録します。このコールバック関数は、利用者が作成します。	7.3.1 SQLDynSetCallback
	SQLGetCallback	コールバック関数の登録状況を取得します。	7.3.2 SQLGetCallback
	SQLSetCallback	コールバック関数を登録します。	7.3.3 SQLSetCallback

第2章 共通要素の文法規則

SQL文を構成する要素を、“共通要素”と呼びます。共通要素は、大きく分けると、以下のように分類することができます。

- 名前の付け方に関する規則
- データ型(属性とも呼びます)に関する規則
- 名前またはデータ型の記述方法に関する規則
- 数値および文字列値の指定方法に関する規則
- 表の中のデータを関数を使って処理するための規則
- データ検索の対象となる表の中から、特定の行を指定する条件を記述するための規則

2.1 構文要素

SQL文を構成する基本的な要素は、次の4項目です。

- 文字
- 定数
- トークン
- 名前

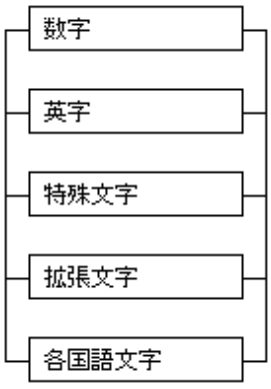
これらの要素が組み合わされることによって、SQL文の構成要素となります。

2.1.1 文字

機能

SQL文で使用可能な文字を規定します。

記述形式



一般規則

Symfoware/RDBの文字コード系がEUCコードの場合

- 各国語文字を除く各文字は、EUCコード(Extended Unix Code)のコードセット0に対応します。これは、ASCIIコードと同じ1バイトコードです。
- ASCIIの文字に対応しない(視覚的な図形によって表せない)値は、文字として扱いませんが、定数やホスト変数に設定する値としては使用することができます。ただし、改行コードを定数に含むことはできません。
- 外字として、各OSで提供される標準の外字と利用者定義文字が使用できます。

Symfoware/RDBの文字コード系がシフトJISコードの場合

- 各国語文字を除く各文字は、シフトJISコードのコードセット0に対応します。これは、ASCIIコードと同じ1バイトコードです。
- ASCIIの文字に対応しない(視覚的な図形によって表せない)値は、文字として扱いませんが、定数やホスト変数に設定する値としては使用することができます。ただし、改行コードを定数に含むことはできません。
- 外字として、各OSで提供される標準の外字と利用者定義文字が使用できます。

Symfoware/RDBの文字コード系がUNICODEの場合

- 各国語文字を除く各文字は、UNICODEのコードセット0に対応します。これは、ASCIIコードと同じ1バイトコードです。
- ASCIIの文字に対応しない(視覚的な図形によって表せない)値は、文字として扱いませんが、定数やホスト変数に設定する値としては使用することができます。ただし、改行コードを定数に含むことはできません。
- 外字として、各OSで提供される標準の外字と利用者定義文字が使用できます。

数字および英字

- 数字および英字に指定できる文字を以下に示します。

要素	指定できる文字
数字	0 ～ 9
英大文字	A ～ Z
英小文字	a ～ z

特殊文字

- 特殊文字として指定できる文字を以下に示します。

,	(	)	.	:	;
=	*	+	-	/	?
<	>	%	_	'	"

拡張文字

- 拡張文字として指定できる文字を以下に示します。

@	¥	#

各国語文字

Symfoware/RDBの文字コード系がEUCコードの場合

- 各国語文字は、日本語の文字を表します。
- 各国語文字のコードは、EUCコードでのJIS漢字／非漢字の範囲(コードセット1である2バイト)およびJEF拡張漢字／非漢字の範囲(コードセット3である3バイト)です。
- 各国語文字に、半角カタカナは含みません。

Symfoware/RDBの文字コード系がシフトJISコードの場合

- 各国語文字は、日本語の文字を表します。
- 各国語文字のコードは、シフトJISコードでのJIS漢字／非漢字を含む範囲(2バイト)です。
- 各国語文字に、半角カタカナは含みません。

Symfoware/RDBの文字コード系がUNICODEの場合

- 各国語文字は、日本語の文字を表します。

- 各国語文字のコードは、UNICODEでのJIS漢字／非漢字を含む範囲です。
  UTF-8形式では2バイト、3バイトまたは4バイト、UCS-2形式では2バイトまたは4バイトです。UCS-2形式では、UNICODEの補助文字(1～16面の4バイト文字)はUCS-2の2文字として扱われます。

- 補助文字のコード変換はUnicode 4.1で変換します。Unicode 4.1の補助文字のコード変換は、文字列型ではUTF-8の4バイト、各国語文字列型ではUCS-2の2文字として扱われます。ただし、以下のいずれかに該当する場合、Unicode 2.0でコード変換を行います。

  1)Symfoware Server 9.1.0以前のバージョンレベル

  2)Unicode 4.1に対応していない以下の製品をインストールしている場合

  <Windows版>
  ・SystemWalker/CharsetMGR-M V5.1L10以降
  ・Interstage Charset Manager V8.2以前

  <Linux版>
  ・Interstage Charset Manager V8.2以前

  <Solaris 32ビット版>
  ・Interstage Charset Manager V8

  Unicode 2.0とUnicode 4.1の補助文字のコード変換はサロゲートペア領域の扱いが異なります。Unicode 4.1はUNICODEの補助文字を、UTF-8に変換すると、4バイト文字に変換します。Unicode 2.0はUTF8として3バイトまでしか扱えません。そのため、UNICODEの補助文字をUTF-8に変換すると、3バイト文字のペア(3バイト×2)に変換されます。

  サーバとクライアントで、Unicode 4.1とUnicode 2.0のコード変換環境が混在する場合、クライアント用の動作環境ファイルのCHARACTER_TRANSLATEの指定により、Unicode 2.0でコード変換を行う場合があります。UNICODEの補助文字をUnicode 4.1で扱いたい場合、クライアント用の動作環境ファイルのCHARACTER_TRANSLATEにUnicode 4.1のコード変換環境を指定してください。

外字

― 外字を使用する場合は、富士通製のコード変換プログラム(Interstage Charset Manager)をインストールしてください。

参照

外字定義の詳細は、“Interstage Charset Manager 使用手引書 標準コード変換機能編”を参照してください。

使用例

例1

英大文字

```
UPDATE
```

例2

英大文字と特殊文字

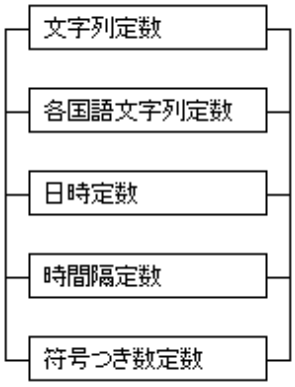
```
COUNT(*)
```

2.1.2 定数

機能

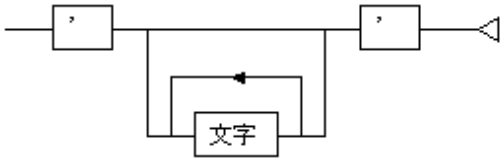
NULLでない値を指定します。

記述形式



文字列定数

記述形式



参照項番

ー 文字 → “2.1.1 文字”

一般規則

文字型を取り扱うコードとしてEUCコードを使用した場合

- 文字列定数に指定できる文字のコードは、EUCコードのコードセット0(1バイト)、EUCコードのコードセット1(2バイト)、EUCコードのコードセット2(2バイト)およびEUCコードのコードセット3(3バイト)です。
- 文字列定数に各国語文字を指定することができます。
- 文字列定数に引用符(')を指定するときは、引用符(')を2つ連続して記述します。

注意

文字列定数に各国語文字を指定する場合、EUCコードの2バイトコードおよび3バイトコードはそのままデータとして扱われ、比較の対象となったり、データベースへの格納の対象となります。このため文字列データに空白を含む場合、文字の空白と各国語文字の空白が異なるコードであることなどに注意が必要です。

文字型を取り扱う文字コード系としてシフトJISコードを使用した場合

- 文字列定数に指定できる文字コード系は、シフトJISコードのコードセット0(1バイト)、シフトJISコードのコードセット1(2バイト)、シフトJISコードのコードセット2(2バイト)およびシフトJISコードのコードセット3(2バイト)です。
- 文字列定数に各国語文字を指定することができます。
- 文字列定数に引用符(')を指定するときは、引用符(')を2つ連続して記述します。

注意

文字列定数に各国語文字を指定する場合、シフトJISコードの2バイトコードはそのままデータとして扱われ、比較の対象となったり、データベースへの格納の対象となります。このため文字列データに空白を含む場合、文字の空白と各国語文字の空白が異なるコードであることなどに注意が必要です。

文字型を取り扱う文字コード系としてUNICODEを使用した場合

- 文字列定数にUNICODEの文字を指定することができます。
- 文字列定数に各国語文字を指定することができます。
- 文字列定数に引用符(')を指定するときは、引用符(')を2つ連続して記述します。

使用例(文字列定数)

例1

```
'TOKYO' 'T123'
```

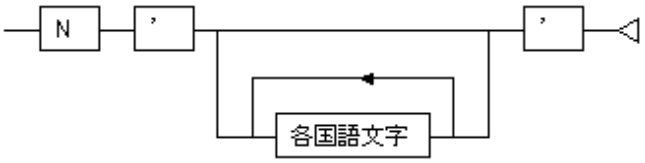
例2

```
'JAPAN''TOKYO''' →結果は、JAPAN'TOKYO'になります。
```

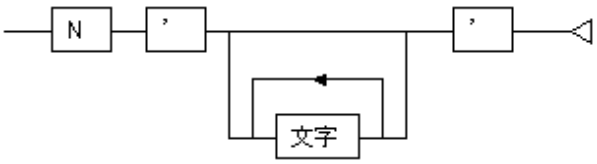
各国語文字列定数

記述形式

各国語文字列定数を取り扱う文字コード系としてEUCコードまたはシフトJISコードを使用した場合



各国語文字列定数を取り扱う文字コード系としてUNICODEを使用した場合



参照項番

- 各国語文字 → “2.1.1 文字”
- 文字 → “2.1.1 文字”

一般規則

- 各国語文字列定数は、各国語文字のデータを指定します。データ型は各国語文字列型です。

各国語文字列定数を取り扱う文字コード系としてEUCコードを使用する場合

- 指定できる各国語文字の文字コード系は、EUCコードのコードセット1(2バイト)およびEUCコードのコードセット3(3バイト)です。

注意

各国語文字は、EUCコードの2バイトコードおよび3バイトコードはCOBOL_EUCに変換され、2バイトコードのままデータとして扱われ、比較の対象となったり、データベースへの格納の対象となります。

各国語文字列定数を取り扱う文字コード系としてシフトJISコードを使用する場合

- 指定できる各国語文字の文字コード系は、シフトJISコードのコードセット1(2バイト)、コードセット2(2バイト)、またはコードセット3(2バイト)です。

各国語文字列定数を取り扱う文字コード系としてUNICODEを使用する場合

- 指定できる各国語文字の文字コード系は、UCS-2の文字です。補助文字(1～16面の4バイト文字)はUCS-2の2文字として格納されます。
- 補助文字のコード変換はUnicode 4.1で変換します。Unicode 4.1の補助文字のコード変換は、文字列型ではUTF-8の4バイト、各国語文字列型ではUCS-2の2文字として扱われます。ただし、以下のいずれかに該当する場合、Unicode 2.0でコード変換を行います。

  1)Symfoware Server 9.1.0以前のバージョンレベル

  2)Unicode 4.1に対応していない以下の製品をインストールしている場合

  <Windows版>
  ・SystemWalker/CharsetMGR-M V5.1L10以降
  ・Interstage Charset Manager V8.2以前

  <Linux版>
  ・Interstage Charset Manager V8.2以前

  <Solaris 32ビット版>
  ・Interstage Charset Manager V8

  Unicode 2.0とUnicode 4.1の補助文字のコード変換はサロゲートペア領域の扱いが異なります。Unicode 4.1はUNICODEの補助文字を、UTF-8に変換すると、4バイト文字に変換します。Unicode 2.0はUTF8として3バイトまでしか扱えません。そのため、UNICODEの補助文字をUTF-8に変換すると、3バイト文字のペア(3バイト×2)に変換されます。

  サーバとクライアントで、Unicode 4.1とUnicode 2.0のコード変換環境が混在する場合、クライアント用の動作環境ファイルのCHARACTER_TRANSLATEの指定により、Unicode 2.0でコード変換を行う場合があります。UNICODEの補助文字をUnicode 4.1で扱いたい場合、クライアント用の動作環境ファイルのCHARACTER_TRANSLATEにUnicode 4.1のコード変換環境を指定してください。

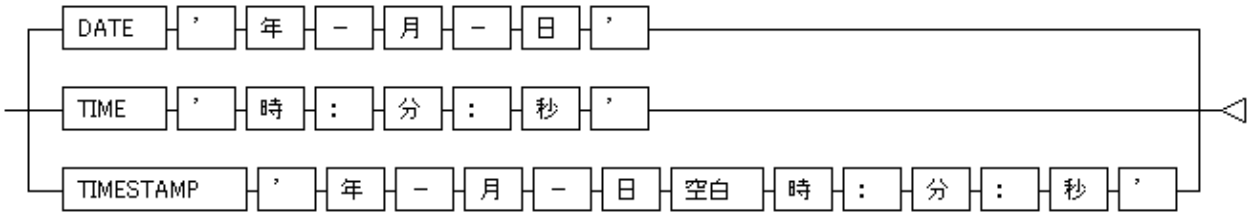
使用例(各国語文字列定数)

例

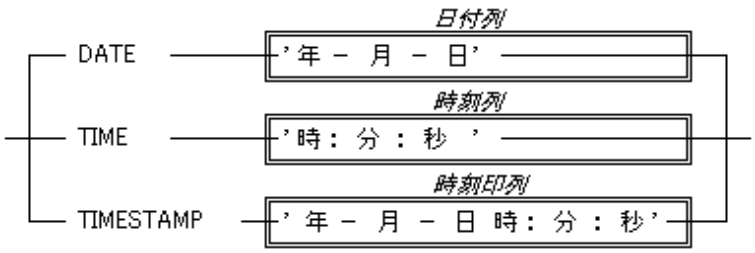
```
N'東京'
```

日時定数

記述形式



構文要素の構成



一般規則

日時定数

- 日時定数の年、月、日、時、分、秒は、日時値を指定します。日時値は、暦上存在する自然な日時値であることが必要です。
- 日時定数中で、年は最大4桁、月～秒は最大2桁の数字で指定します。
- 日付定数、時刻定数、時刻印定数を引用符で囲んだ文字列を日付列、時刻列、時刻印列といいます。

DATE(日付定数)

- DATEで始まる定数を、“日付定数”と呼びます。
- 日付定数は、年から日までの10文字の日付を格納することができます。年から日のそれぞれを日時フィールドYEAR、MONTH、DAYといいます。日付定数の日時フィールドの有効値を以下に示します。

表2.1 日付定数の日時フィールドの有効値

日時フィールド	意味	有効値
YEAR	年	1 ～ 9999
MONTH	月	1 ～ 12
DAY	日	1 ～ 31

TIME(時刻定数)

- TIMEで始まる定数を、“時刻定数”と呼びます。
- 時刻定数は、時から秒までの8文字の時刻を格納することができます。時から秒のそれぞれを日時フィールドHOUR、MINUTE、SECONDといいます。時刻定数の日時フィールドの有効値を以下に示します。

表2.2 時刻定数の日時フィールドの有効値

日時フィールド	意味	有効値
HOUR	時	0 ～ 23
MINUTE	分	0 ～ 59
SECOND	秒	0 ～ 59

TIMESTAMP(時刻印定数)

- TIMESTAMPで始まる定数を、“時刻印定数”と呼びます。
- 時刻印定数は、年から秒までの19文字の時刻印を格納することができます。年から秒のそれぞれを日時フィールドYEAR、MONTH、DAY、HOUR、MINUTE、SECONDといいます。時刻印定数の日時フィールドの有効値を以下に示します。

表2.3 時刻印定数の日時フィールドの有効値

日時フィールド	意味	有効値
YEAR	年	1 ～ 9999
MONTH	月	1 ～ 12

日時フィールド	意味	有効値
DAY	日	1 ～ 31
HOUR	時	0 ～ 23
MINUTE	分	0 ～ 59
SECOND	秒	0 ～ 59

使用例(日時定数)

例1

日付定数

```
DATE '2007-04-10'
```

例2

時刻定数

```
TIME '12:10:40'
```

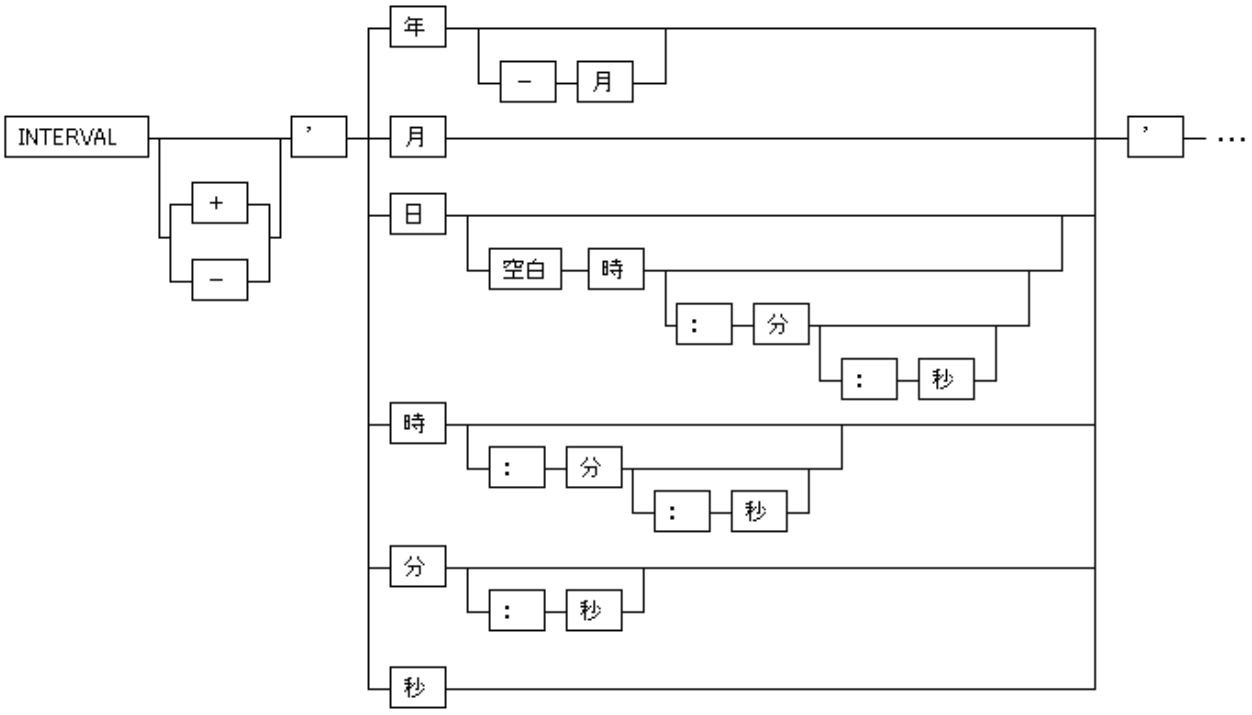
例3

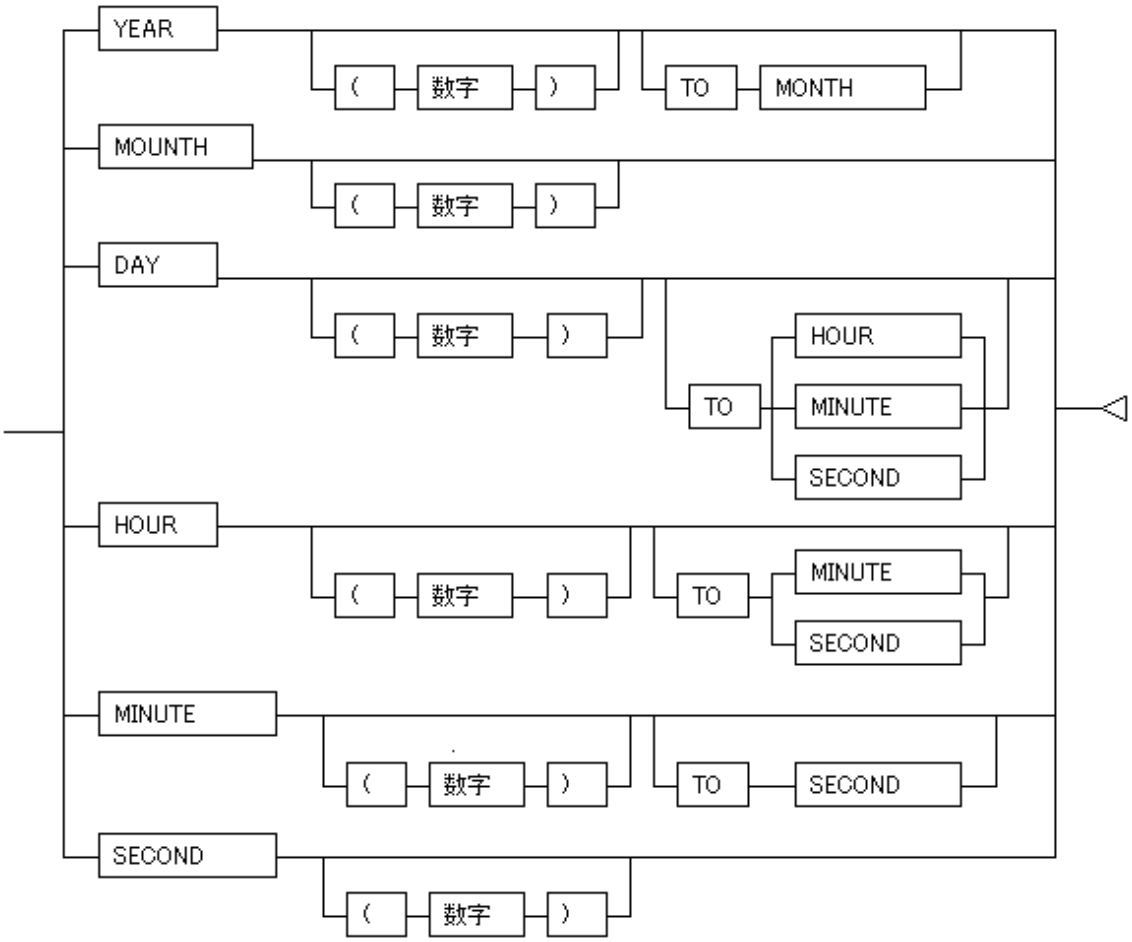
時刻印定数

```
TIMESTAMP '2007-04-10 12:10:40'
```

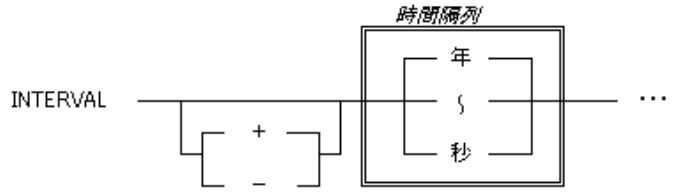
時間隔定数

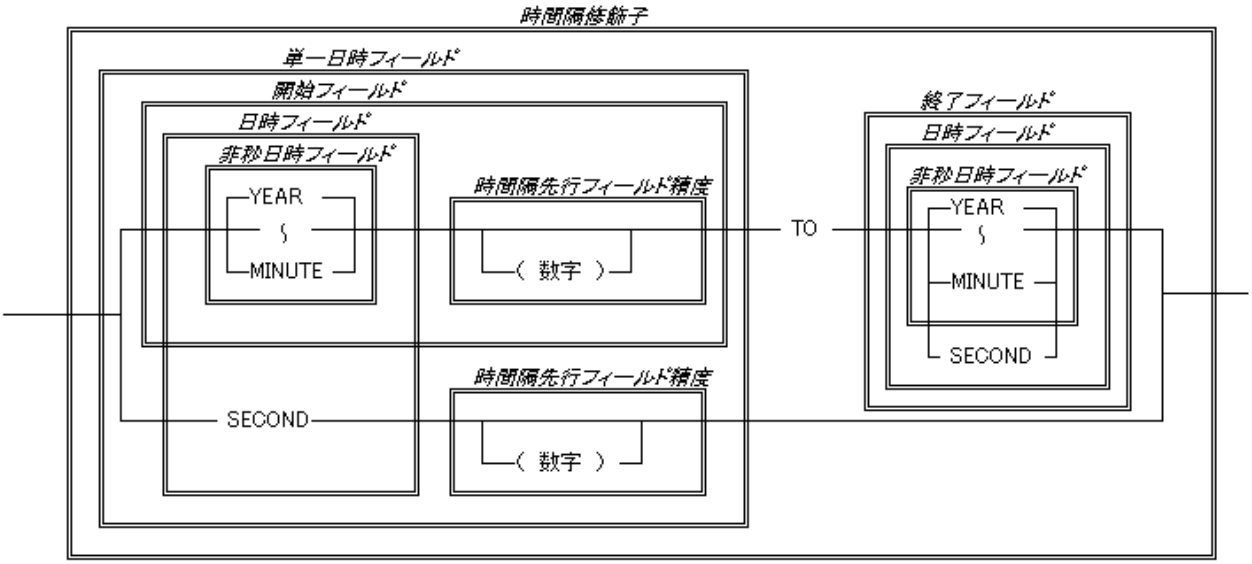
記述形式




YEAR
(
数字
)
TO
MONTH
MOUNTH
(
数字
)
DAY
(
数字
)
TO
HOUR
MINUTE
SECOND
HOUR
(
数字
)
TO
MINUTE
SECOND
MINUTE
(
数字
)
TO
SECOND
SECOND
(
数字
)

構文要素の構成


時間隔列
INTERVAL
+
-
年
秒


時間隔修飾子
単一日時フィールド
開始フィールド
日時フィールド
非秒日時フィールド
YEAR
MINUTE
時間隔先行フィールド精度
(数字)
SECOND
時間隔先行フィールド精度
(数字)
TO
終了フィールド
日時フィールド
非秒日時フィールド
YEAR
MINUTE
SECOND

参照項番

— 数字 → “2.1.1 文字”

一般規則

時間隔定数

- 時間隔定数の年、月、日、時、分、秒は、日時値を指定します。日時値は、開始フィールドおよび単一日時フィールドを除き、暦上存在する自然な日時値であることが必要です。
- 時間隔修飾子は、時間隔の内容を示す修飾子です。
- 単一日時フィールドは、日時を示す1つの要素で、年、月、日、時、分、秒があります。
- 日時フィールドには順位があります。その順位は、最上位のものから最下位のものの順に、YEAR、MONTH、DAY、HOUR、MINUTE、SECONDです。
- 非秒日時フィールドは、単一日時フィールドのうち、秒を除いた残りのことです。
- 時間隔先行フィールド精度は、1以上9以下の符号なし整数であることが必要です。
- 時間隔定数は、年月クラスと日時クラスに分類されます。各クラスにまたがるような日時フィールドを持つような指定はできません。
- 開始フィールドは終了フィールドより上位であることが必要です。
- 年月クラスは、開始フィールドおよび終了フィールドにより指定された年および月からなる連続する日時フィールドを含みます。または、単一日時フィールドにより指定される年または月の単一日時フィールドを含みます。時間隔定数の年月クラスを以下に示します。

表2.4 時間隔定数の年月クラス

開始フィールド	終了フィールド
YEAR	MONTH

表2.5 時間隔定数の年月クラス

単一日時フィールド
YEAR
MONTH

- 日時クラスは、開始フィールドおよび終了フィールドにより指定された日、時、分および秒からなる連続する任意の日時フィールドを含みます。または、単一日時フィールドにより指定される日、時、分および秒の単一日時フィールドを含みます。時間隔定数の日時クラスを以下に示します。

表2.6 時間隔定数の日時クラス

開始フィールド	終了フィールド
DAY	HOUR MINUTE SECOND
HOUR	MINUTE SECOND
MINUTE	SECOND

表2.7 時間隔定数の日時クラス

単一日時フィールド
DAY
HOUR
MINUTE

単一日時フィールド
SECOND

- 時間隔定数の最初の日時フィールドはn桁の整数です。nは時間隔先行フィールド精度で指定された符号なし整数です。時間隔先行フィールド精度を省略した場合は、以下のようになります。時間隔定数の先行フィールド精度の省略値を以下に示します。

表2.8 時間隔定数の先行フィールド精度の省略値

開始フィールド	時間隔定数の先行フィールド精度の省略値
YEAR	2
MONTH	2
DAY	2
HOUR	2
MINUTE	2
SECOND	2

- 時間隔定数の2番目以降の日時フィールドに許される値は以下のように制約されています。時間隔定数の日時フィールドの有効値を以下に示します。

表2.9 時間隔定数の日時フィールドの有効値

日時フィールド	有効値
MONTH	0 ～ 11
HOUR	0 ～ 23
MINUTE	0 ～ 59
SECOND	0 ～ 59

使用例(時間隔定数)

例1

年-月型

```
INTERVAL '1-6' YEAR TO MONTH
```

例2

日-秒型

```
INTERVAL '10 12:10:40' DAY TO SECOND
```

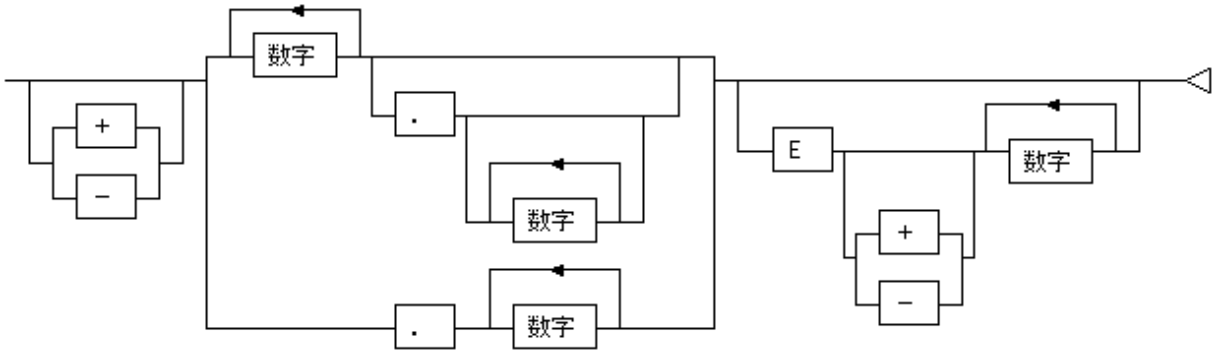
例3

時間隔先行フィールド精度を指定

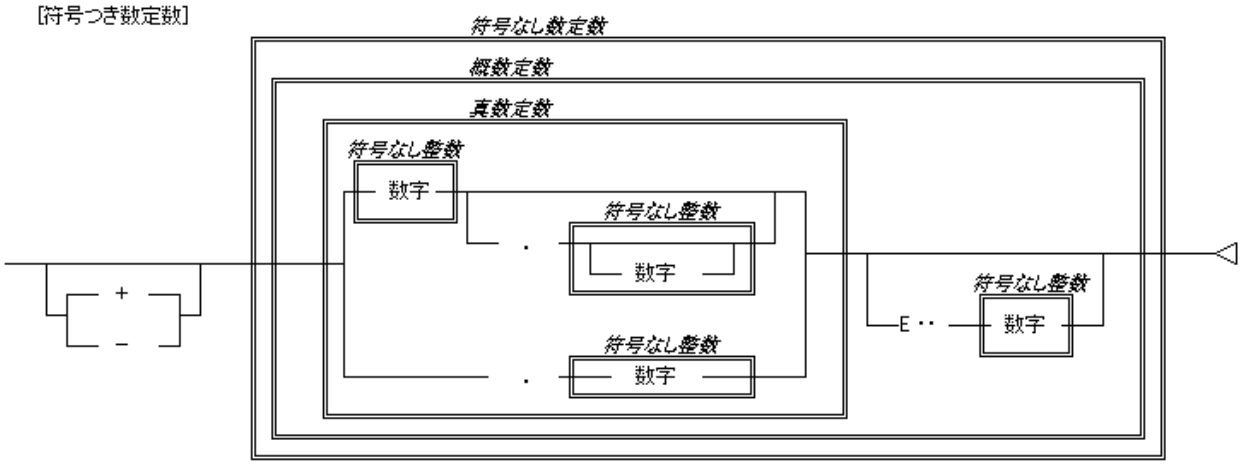
```
INTERVAL '100:29' MINUTE(3) TO SECOND
```

符号つき数定数

記述形式



構文要素の構成



参照項番

― 数字 → “2.1.1 文字”

一般規則

符号なし数定数

- 符号なし数定数は、真数定数と概数定数があります。

符号なし整数

- 符号なし整数は、符号の指定を行わない整数です。

真数定数

- 真数定数は、真数のデータを指定します。データ型は真数型です。
- 真数定数は、指定する値によって、扱われるデータ型が変わります。データ型について以下に示します。

要素指定	桁数 (注1)	指定される値	データ型
小数点なし	1 ～ 5	-32768 ～ +32767	SMALLINT
	5 ～ 10	-2147483648 ～ -32769 + 32768 ～ +2147483647	INTEGER
	10 ～ 18	-999999999999999999 ～ -2147483649 +2147483648 ～ +999999999999999999	DEC(精度,0) (注2)

要素指定	桁数 (注1)	指定される値	データ型
	19以上	-----	構文エラー
小数点あり	1 ～ 18	-999999999999999999. ～ +999999999999999999.	DEC(精度,位取り) (注2)
	19以上	-----	構文エラー

注1) 桁数に小数点は含みません。

注2) 精度は数字の数、位取りは小数点以下の数字の数のことです。

概数定数

- 概数定数は、概数のデータを指定します。データ型は概数型です。
- 概数定数は、DOUBLE PRECISION型で扱われます。

使用例(符号つき数定数)

例1

真数定数

```
+15.5
```

例2

概数定数

```
-20.4E5
```

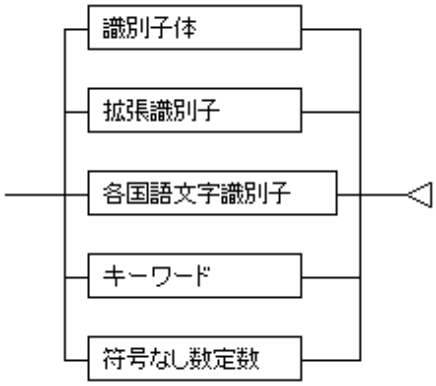
2.1.3 トークン

機能

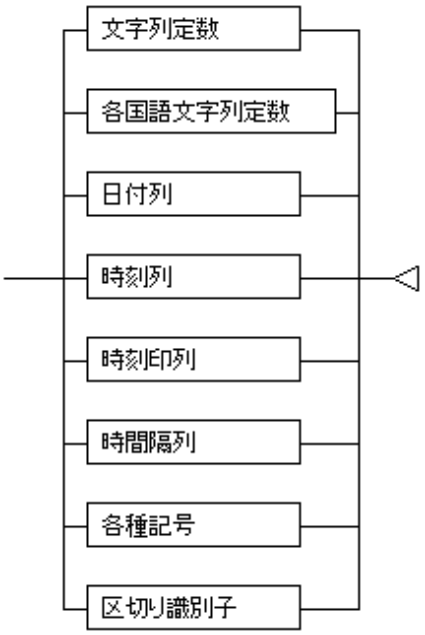
SQL文を構成する最小単位を指定します。

記述形式

非区切りトークン

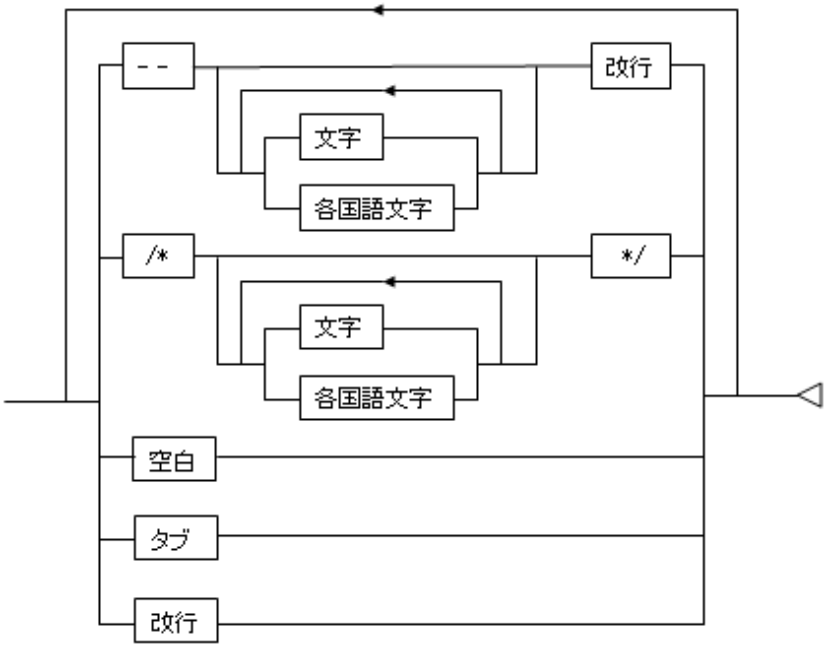


区切りトークン

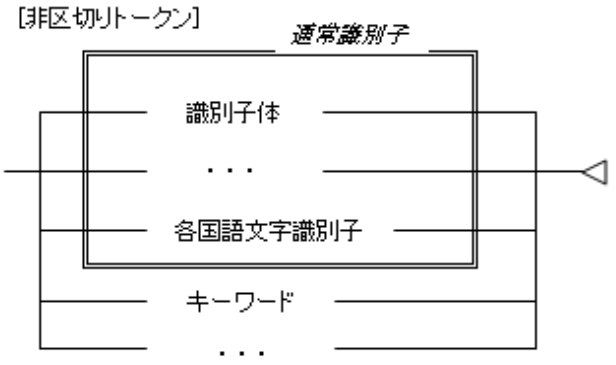


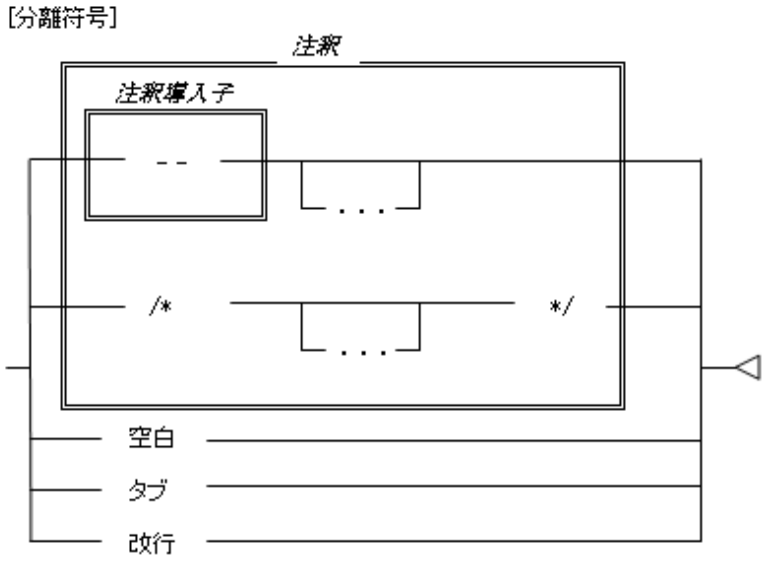
識別子

分離符号



構文要素の構成





参照項番

- 英字 → “2.1.1 文字”
- 数字 → “2.1.1 文字”
- 文字 → “2.1.1 文字”
- 各国語文字 → “2.1.1 文字”
- 拡張文字 → “2.1.1 文字”
- 符号なし数定数 → “2.1.2 定数”
- 各国語文字列定数 → “2.1.2 定数”
- 文字列定数 → “2.1.2 定数”
- 日付列 → “2.1.2 定数”
- 時刻列 → “2.1.2 定数”
- 時刻印列 → “2.1.2 定数”
- 時間隔列 → “2.1.2 定数”

一般規則

識別子体

一　英字で始まる下線または英数字で指定します。

拡張識別子

― 拡張文字または英字で始まる下線、拡張文字または英数字で指定します。

注意

Symfoware Serverのマニュアルでは、識別子体と拡張識別子を合わせて、“先頭が英字で始まる英数字”と説明する場合があります。Symfoware Serverのマニュアルで“先頭が英字で始まる英数字”と説明している部分は、特別な説明がない限り、“英字で始まる下線または英数字”または、“拡張文字または英字で始まる下線、拡張文字または英数字”という意味になります。

各国語文字識別子

各国語文字識別子を構成する文字数は18文字以内です。ただし、認可識別子や共用バッファ識別子など、識別子の種類によって識別子に指定可能な長さは異なります。各識別子の制限値については、“付録C 定量制限”を参照してください。

Symfoware/RDBの文字コード系がEUCコードの場合

- 各国語文字識別子には、各国語文字を指定します。
- 各国語文字識別子に、各国語文字の空白およびJEF拡張漢字／非漢字および83JIS改定文字を含むことはできません。
- 各国語文字識別子に、半角カタカナを含むことはできません。
- EUCコードの各国語文字は2バイトコードです。
- 各国語文字識別子に指定できるのは、シフトJISコードで表現可能な文字の範囲です。

Symfoware/RDBの文字コード系がシフトJISコードの場合

- 各国語文字識別子には、各国語文字を指定します。
- 各国語文字識別子に、各国語文字の空白を含むことはできません。
- 各国語文字識別子に、半角カタカナを含むことはできません。
- シフトJISコードの各国語文字は2バイトコードです。

Symfoware/RDBの文字コード系がUNICODEの場合

- 各国語文字識別子に指定できる各国語文字のコードは、UNICODEです。
- 各国語文字識別子に、各国語文字の空白を含むことはできません。
- 各国語文字識別子に、半角カタカナを含むことはできません。
- UNICODEの各国語文字は2バイトコードと3バイトコードがあります。
- 各国語文字識別子に指定できるのは、シフトJISコードで表現可能な文字の範囲です。
- 各国語文字識別子に、UNICODEの補助文字(1～16面の4バイト文字)を含むことはできません。
- UNICODEの各国語文字はUTF-8形式では2バイトまたは3バイト、UCS-2形式では2バイトです。

キーワード

― キーワードは、予約語です。識別子として指定することはできません。キーワードを識別子として指定したい場合は、区切り識別子で指定する必要があります。キーワードの種類については、“付録Bキーワード一覧”を参照してください。

識別子

― 識別子を構成する文字数は36文字以内です。識別子が各国語文字識別子である場合は、各国語文字の文字数は18文字以内です。ただし、認可識別子や共用バッファ識別子など、識別子の種類によって識別子に指定可能な長さは異なります。各識別子の制限値については、“付録C 定量制限”を参照してください。

― 識別子が各国語文字識別子の場合については、“各国語文字識別子”を参照してください。

通常識別子

- 通常識別子に英小文字を指定した場合、対応する英大文字に変換されます。ただし、各国語文字識別子は英小文字と英大文字を等価に扱いません。
- 通常識別子としてキーワードを使用することはできません。

区切り識別子

- 通常識別子またはキーワードを二重引用符(")で囲って指定します。
- 英小文字と英大文字を等価に扱いません。

各種記号

- 区切りトークンに指定できる各種記号を以下に示します。

||	,	(	)	.	:
=	*	+	-	/	?
<	>	<>	<=	>=	!<
!>	!=				

分離符号

- すべてのトークンのうしろに、“/*”で始まる注釈以外の分離符号を指定することができます。非区切りトークンのうしろには区切りトークンまたは分離符号を指定することが必要です。ただし、キーワード(非区切りトークン)の直後に各国語文字列定数(区切りトークン)を指定することはできません。この場合は、分離符号を指定することが必要です。
- 空白、タブおよび改行はEUCコードのコードセット0(ASCIIコードに同じ)、シフトJISコードのコードセット0(ASCIIコードに同じ)またはUNICODEのコードセット(ASCIIコードに同じ)に対応する空白文字、タブ文字および改行文字です。
- “--”で始まる注釈は、SQL文の任意の場所に指定できます。文字列定数中以外で注釈導入子を指定した位置から、その行の終わりまでが注釈になります。1行すべてが注釈でもかまいません。
- “/*”で始まる注釈は、カーソル宣言、単一行SELECT文、INSERT文の問合せ指定、導出表、または副問合せのキーワード“SELECT”の直後に指定して、ASSIST指定のために使用できます。“/*”を指定した位置から“*/”の位置までが注釈になります。“/*”と“*/”で囲まれた注釈は、複数行に渡って指定することも可能です。
- “/*”と“*/”で囲まれた注釈中に注釈導入子“--”を指定すると、注釈導入子が優先されます。
- “/*”と“*/”で囲まれた注釈中にSQL終了子(END-EXECまたは;)を指定すると、SQL終了子が優先されます。
- “/*”の後ろに“ASSIST”とある場合のみ、ASSIST指定として有効になります。ただし、有効なASSIST要素が1つもない場合は、ASSIST指定として有効になりません。

使用例

例1

非区切りトークン

```
COLUMN_NAME   ・・・ 識別子(列名)
UPDATE        ・・・ キーワード
15            ・・・ 数定数
```

例2

区切りトークン

```
'TOKYO'              ・・・ 文字列定数
N'東京'              ・・・ 各国語文字列定数
'2007-04-10'         ・・・ 日付列
```

```
'12:10:40'                ・・・ 時刻列
'2007-04-10 12:10:40'     ・・・ 時刻印列
+                         ・・・ 演算子
"SCHEMA"                  ・・・ 区切り識別子(キーワードを識別子とし
                                 て使用)
```

例3

注釈

```
EXEC SQL
    DECLARE 在庫表カーソル CURSOR FOR
    SELECT 製品番号,製品名 -- 在庫表を検索
    FROM 在庫管理.在庫表              (1)
END-EXEC
```

(1) 注釈

例4

注釈

```
EXEC SQL
    DECLARE 在庫表カーソル CURSOR FOR
    SELECT          /* 在庫表を検索 */     製品番号,製品名
    FROM 在庫管理.在庫表       (1)
END-EXEC
```

(1) 注釈

例5

ASSIST指定

```
EXEC SQL
    DECLARE 在庫表カーソル CURSOR FOR
    SELECT        /* ASSIST FIRST_ROWS */     製品番号,製品名
    FROM 在庫管理.在庫表       (1)
    ORDER BY 製品番号
END-EXEC
```

(1) ASSIST指定

2.1.4 名前

機能

種々の名前を指定します。

参照項番

- 識別子 → “2.1.3 トークン”
- 単純値指定 → “2.3 値指定と相手指定”

一般規則

- 以下の名前は識別子で記述します。
  - ― スキーマ名
  - ― 列名

- 相関名
- カーソル名
- SQL文識別子
- インデックス名
- データベース名
- データベーススペース名
- DSO名
- DSI名
- SQL変数名
- パラメタ名
- 認可識別子
- 文ラベル
- 条件名
- スコープ名
- ロール名

・ 以下の名前はスキーマ名で修飾した識別子で記述します。スキーマ名は省略できる場合があります。スキーマ名修飾の詳細は“D.2 名前に関する注意事項”を参照してください。

- 表名
- ルーチン名
- 順序名
- トリガ名

・ 以下の名前は単純値指定で記述します。

- 記述子名
- コネクション名
- SQLサーバ名

・ 以下の名前は単純値指定で記述します。ただし定数は指定できません。

- 拡張カーソル名
- 拡張SQL文識別子

使用例

例

カーソル名、列名、表名および相関名の使用例です。

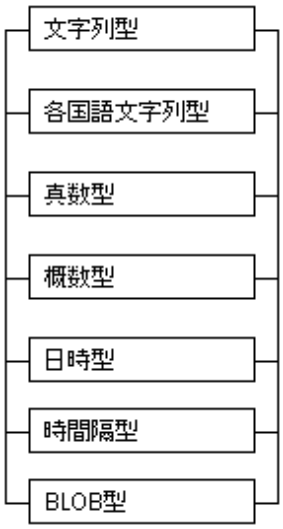
```
DECLARE 検索 CURSOR FOR          ・・・ カーソル名(検索)
    SELECT 製品番号              ・・・ 列名(製品番号)
        FROM ZAIKO.在庫表 T1     ・・・ 表名(在庫表)、相関名(T1)
        WHERE T1.在庫数量 > 100・・・ 相関名(T1)、列名(在庫数量)
```

2.2 データ型

機能

データの型を定義します。データ型は、表の列のデータの型、ストアドプロシジャ、ファンクションルーチンで扱うデータの型に指定します。

記述形式



一般規則

・ 指定できるデータ型を以下に示します。

表2.10 列のデータ型

型種類	データ型指定形式	指定の意味
文字列型	CHARACTER(n) CHAR(n)	固定長で長さn文字の文字列 (n)を省略すると1文字になります。 n:1～32000
	CHARACTER VARYING(n) CHAR VARYING(n) VARCHAR(n)	可変長で長さ最大n文字の文字列 (n)を省略すると1文字になります。 n:1～32000
各国語文字列型	NATIONAL CHARACTER(n) NATIONAL CHAR(n) NCHAR(n)	固定長で長さn文字の日本語文字列 (n)を省略すると1文字になります。 n:1～16000
	NATIONAL CHARACTER VARYING(n) NATIONAL CHAR VARYING(n) NCHAR VARYING(n)	可変長で長さ最大n文字までの日本語文字列 (n)を省略すると1文字になります。 n:1～16000
真数型	NUMERIC(p,q)	桁数p、小数点以下q桁のゾーン形式10進数 pを省略すると18、qを省略すると0になります。 p:1～18 q:0～p

型種類	データ型指定形式	指定の意味
	DECIMAL(p,q) DEC(p,q)	桁数p、小数点以下q桁のパック形式10進数 pを省略すると18、qを省略すると0になります。 p:1～18 q:0～p
	INTEGER INT	-2^{31}～2^{31}-1の整数
	SMALLINT	-2^{15}～2^{15}-1の整数
概数型	FLOAT(p)	仮数部が-2^{p}～2^{p}の概数 p:1～52 p=1～23の場合はREALの扱い p=24～52の場合はDOUBLE PRECISIONの扱い
	REAL	4バイトの浮動小数点数
	DOUBLE PRECISION	8バイトの浮動小数点数
日時型	DATE	年から日までの10文字の日付を格納
	TIME	時から秒までの8文字の時刻を格納
	TIMESTAMP	年から秒までの19文字の時刻印を格納
時間隔型	INTERVAL 開始日時フィールド TO 終了日時フィールド	各フィールドの指定により、年～月、日～時刻の時間隔を格納 (詳細は“表2.11 時間隔型の指定と意味”を参照)
BLOB	BINARY LARGE OBJECT(n単位) BLOB(n単位)	バイナリ属性のデータを格納 単位にはK、MまたはGを指定します。 単位は省略できません。 単位がKの場合) n :1 ～2097152 単位がMの場合) n :1 ～2048 単位がGの場合) n :1 または2

n:文字数
p:精度
q:位取り

表2.11 時間隔型の指定と意味

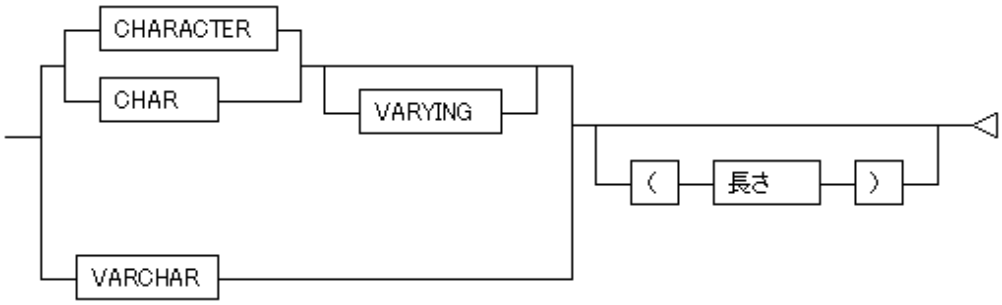
型種類	開始日時フィールド	終了日時フィールド	指定の意味
年・月型	YEAR(p)	―	桁数pの年を表す時間隔
		MONTH	桁数pの年と月を表す時間隔
	MONTH(p)	―	桁数pの月を表す時間隔
日・時型	DAY(p)	SECOND	桁数pの日と時間から秒を表す時間隔
		MINUTE	桁数pの日と時間から分を表す時間隔
		HOUR	桁数pの日と時間を表す時間隔
		―	桁数pの日を表す時間隔
	HOUR(p)	SECOND	桁数pの時間と分から秒を表す時間隔
		MINUTE	桁数pの時間と分を表す時間隔
		―	桁数pの時間を表す時間隔
	MINUTE(p)	SECOND	桁数pの分と秒を表す時間隔

型種類	開始日時フィールド	終了日時フィールド	指定の意味
		－	桁数pの分を表す時間隔
	SECOND(p)	－	桁数pの秒を表す時間隔

p : 開始日時フィールドの精度(1～9の整数を指定)

文字列型

記述形式



一般規則

－ CHARACTERは、文字列型のデータ型です。文字列型の長さは次のようになります。

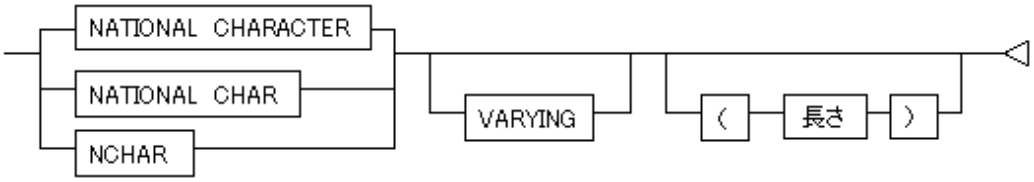
- VARYINGが指定されていないとき、文字列の長さは固定長であり、長さで指定した値となります。
- VARYINGが指定されているとき、文字列の長さは可変長となります。このとき、下限は0、上限は長さで指定した値となります。

－ CHARACTERとCHARは同じ意味です。

－ CHARACTER VARYING、CHAR VARYINGおよびVARCHARは同じ意味です。

各国語文字列型

記述形式



一般規則

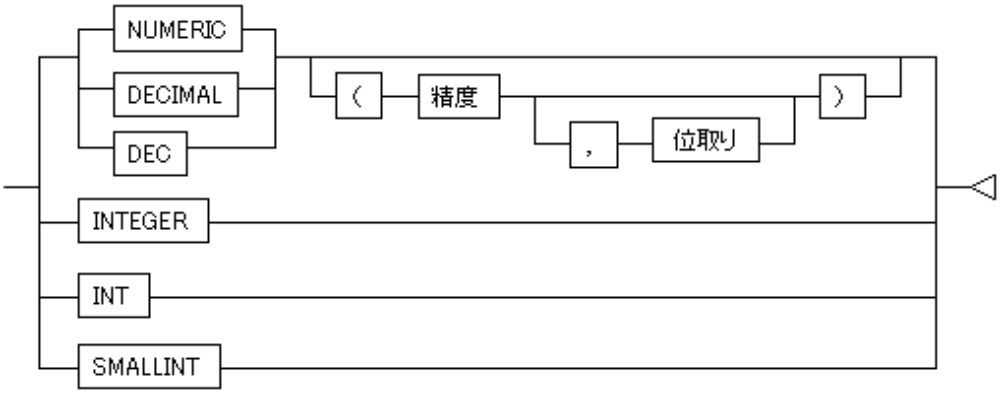
－ NATIONAL CHARACTERは、各国語文字列型のデータ型です。各国語文字列型の長さは次のようになります。

- VARYINGが指定されていないとき、各国語文字列の長さは固定長であり、長さで指定した値となります。
- VARYINGが指定されているとき、各国語文字列の長さは可変長となります。このとき、下限は0、上限は長さで指定した値となります。

－ NATIONAL CHARACTER、NATIONAL CHARおよびNCHARは同じ意味です。

真数型

記述形式

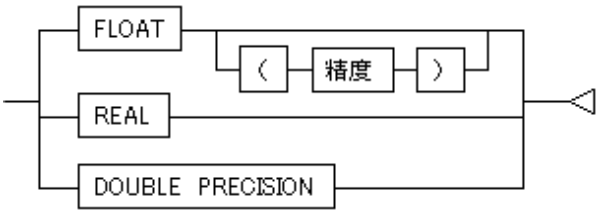


一般規則

- 真数型は、符号をもつデータ型です。
- DECIMALとDECは同じ意味です。
- INTEGERとINTは同じ意味です。
- NUMERICおよびDECIMALは、10進の精度と位取りを持ちます。
- プロシジャルーチン定義にNUMERICまたはDECIMALを指定する場合、精度を指定する必要があります。
- INTEGERおよびSMALLINTは、2進の精度を持ちます。INTEGERとSMALLINTの精度は31と15です。

概数型

記述形式

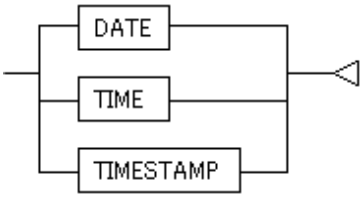


一般規則

- 概数型は、符号をもつデータ型です。
- REALおよびDOUBLE PRECISIONは、2進の精度を持ちます。REALとDOUBLE PRECISIONの精度は23と52です。FLOATは、指定した精度によってREALまたはDOUBLE PRECISIONが指定されたものとみなします。精度が1～23のときはREALに、24～52のときにはDOUBLE PRECISIONに対応します。

日時型

記述形式

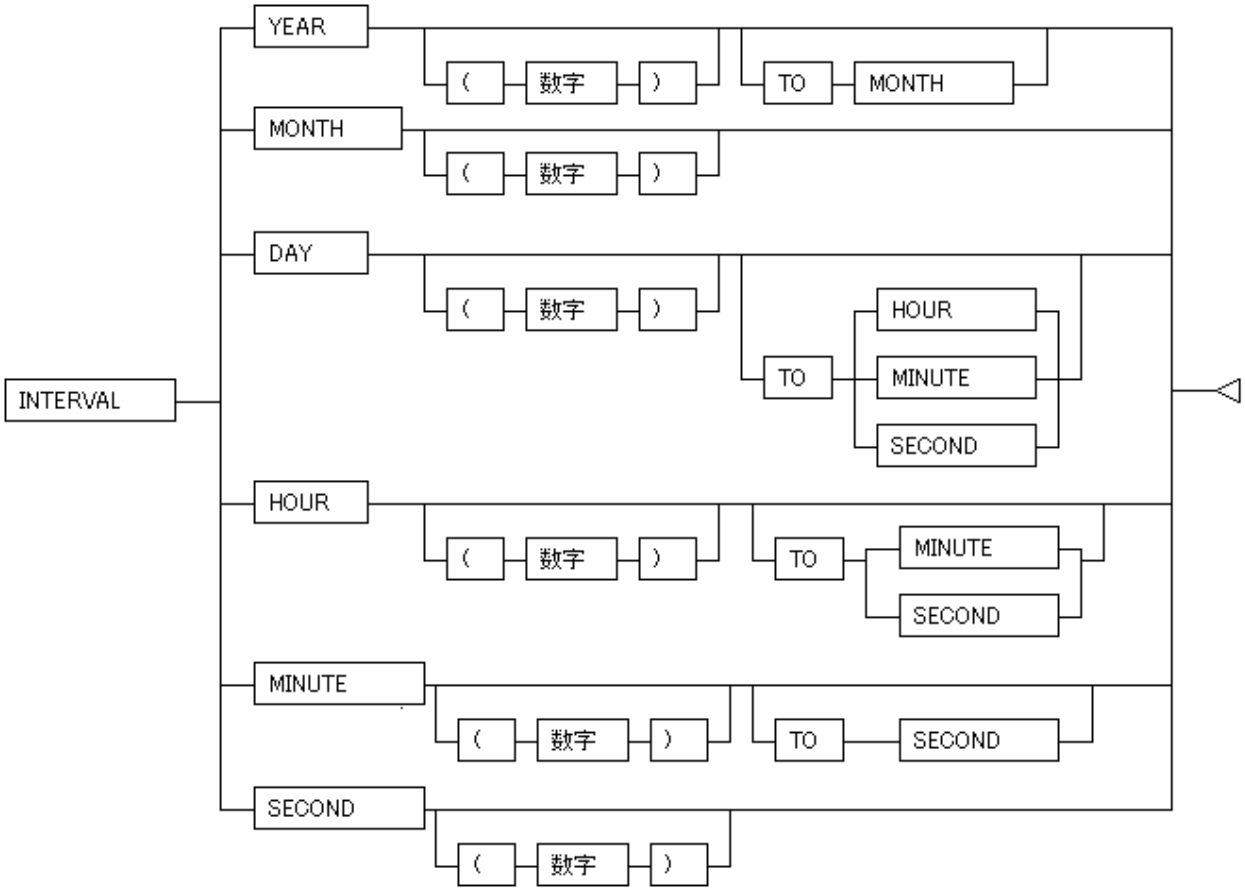


一般規則

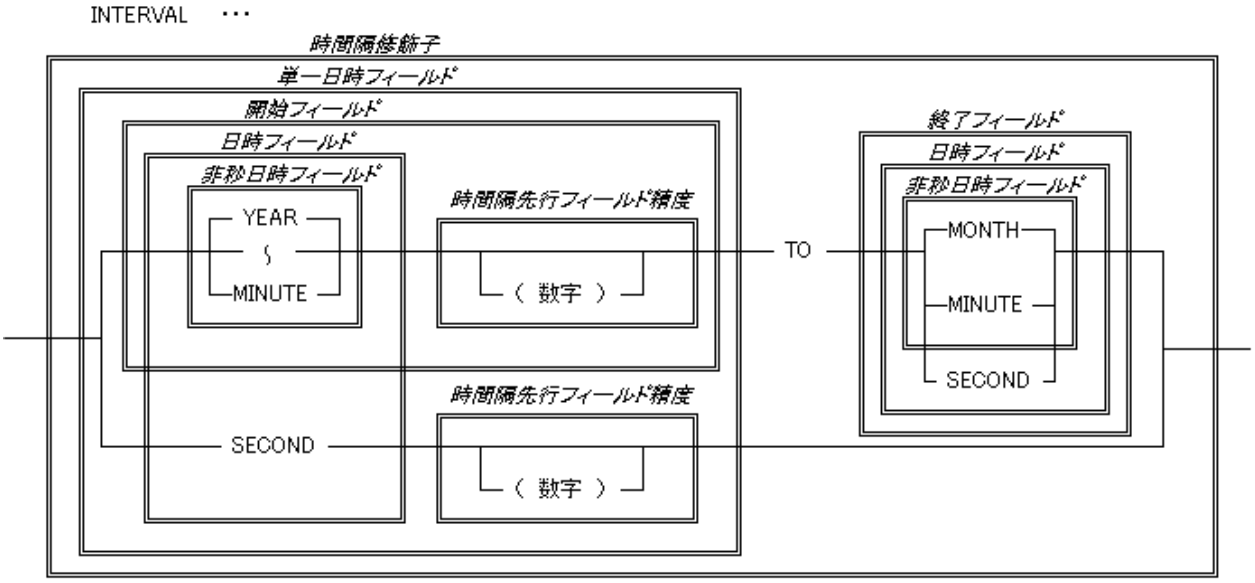
- DATE、TIMEおよびTIMESTAMPは、日時型のデータ型です。
- DATEは、日付の情報を格納するためのデータ型です。
- TIMEは、時間の情報を格納するためのデータ型です。
- TIMESTAMPは、日付と時間の両方の情報を格納するためのデータ型です。

時間隔型

記述形式



構文要素の構成



参照項番

― 数字 → “2.1.1 文字”

一般規則

― 時間隔型は、時間隔をもつデータ型です。

― 日時フィールドには順位があります。その順位は、最上位のものから最下位のものの順に、YEAR、MONTH、DAY、HOUR、MINUTE、SECONDです。

― 時間隔先行フィールド精度は、1以上9以下の符号なし整数であることが必要です。

― 時間隔型は、年月クラスと日時クラスに分類されます。各クラスにまたがるような日時フィールドを持つ指定はできません。

― 開始フィールドは終了フィールドより上位であることが必要です。

― 非秒日時フィールドは、単一日時フィールドのうち、秒を除いた残りのことです。

― 年月クラスは、開始フィールドおよび終了フィールドにより指定された年および月からなる連続する日時フィールドを含みます。または、単一日時フィールドにより指定される年または月の単一日時フィールドを含みます。時間隔型の年月クラスを以下に示します。

表2.12 時間隔型の年月クラス

開始フィールド	終了フィールド
YEAR	MONTH

表2.13 時間隔型の年月クラス

単一日時フィールド
YEAR
MONTH

― 日時クラスは、開始フィールドおよび終了フィールドにより指定された日、時、分および秒からなる連続する任意の日時フィールドを含みます。または、単一日時フィールドにより指定される日、時、分および秒の単一日時フィールドを含みます。時間隔型の日時クラスを以下に示します。

表2.14 時間隔型の日時クラス

開始フィールド	終了フィールド
DAY	HOUR MINUTE SECOND
HOUR	MINUTE SECOND
MINUTE	SECOND

表2.15 時間隔型の日時クラス

単一日時フィールド
DAY
HOUR
MINUTE
SECOND

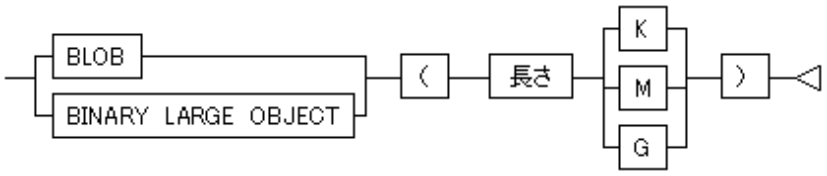
- 時間隔型の最初の日時フィールドはn桁の整数です。nは時間隔先行フィールド精度で指定された符号なし整数です。時間隔先行フィールド精度を省略した場合は、以下のようになります。時間隔先行フィールドの省略値を以下に示します。

表2.16 時間隔先行フィールドの省略値

開始フィールド	時間隔先行フィールドの省略値
YEAR	2
MONTH	2
DAY	2
HOUR	2
MINUTE	2
SECOND	2

BLOB型

記述形式



一般規則

- BLOB型は、バイナリ属性のデータを格納することができます。格納する長さは符号なし整数と単位記号の組合せで指定します。単位記号はK(キロバイト)、M(メガバイト)、G(ギガバイト)が指定できます。1キロバイトは1024バイトを示します。
- BLOBとBINARY LARGE OBJECTは同じ意味です。
- 長さに2047メガバイトより大きな値を指定した場合は、2047メガバイトとして扱います。

使用例

例

会社表の定義でのデータ型の使用例です。

```
CREATE TABLE 会社表(会社番号 SMALLINT NOT NULL, ・・・ 真数型
             会社名 NCHAR(10),                  ・・・ 各国語文字列型
             電話番号 CHAR(14),                 ・・・ 文字列型
             住所 NCHAR(20),                    ・・・ 各国語文字列型
             創立 DATE,                         ・・・ 日時型
             地図 BLOB(20K))                     ・・・ BLOB型
```

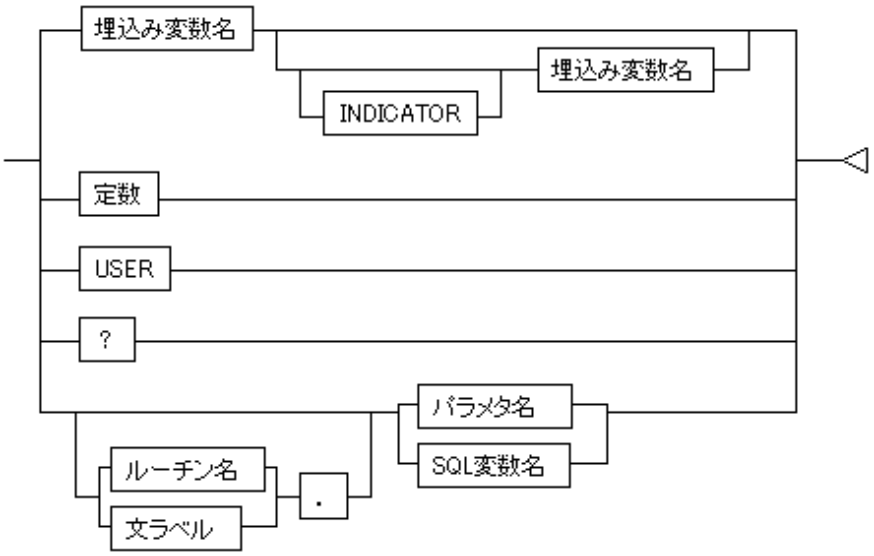
2.3 値指定と相手指定

機能

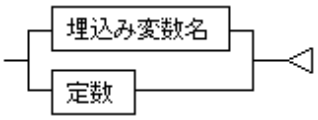
値および変数を指定します。

記述形式

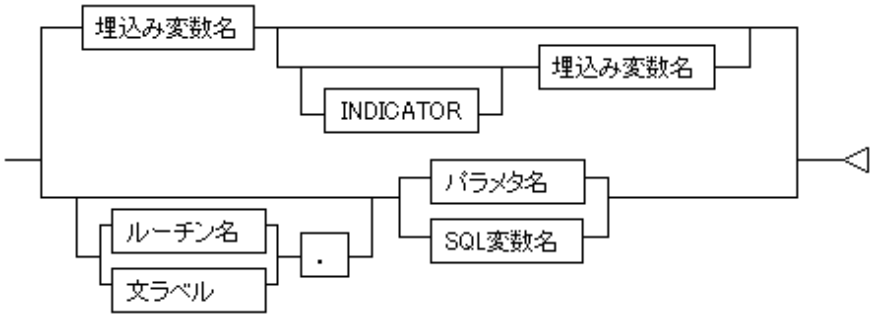
値指定



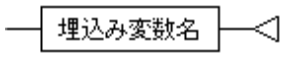
単純値指定



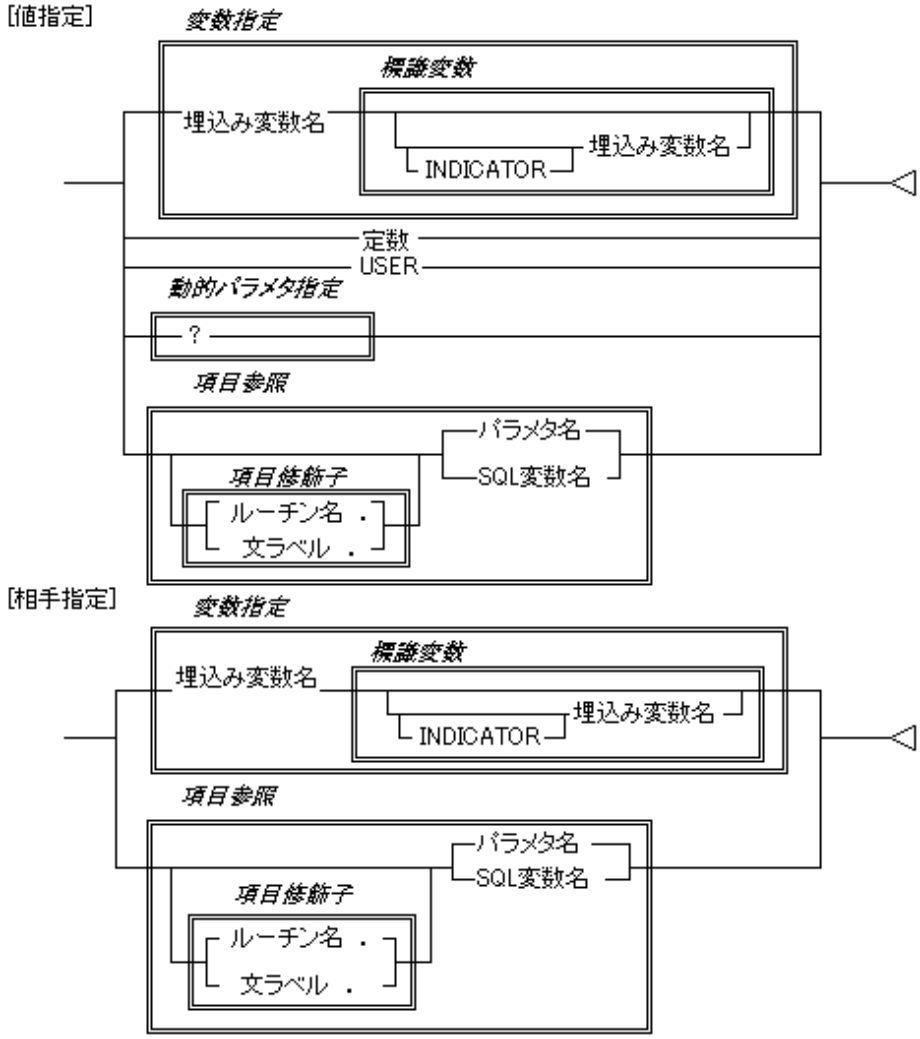
相手指定



単純相手指定



構文要素の構成



参照項番

- 定数 → “2.1.2 定数”
- 埋込み変数名 → “6.5 SQL埋込みホストプログラム”
- 日本語文字列 → “2.1.3 トークン”

一般規則

- 相手指定または単純相手指定は、値を代入される変数を指定します。

変数指定

- 変数指定または埋込み変数名は、埋込みSQL文中に含まれることが必要です。
- 変数指定は、ホスト変数またはホスト変数と標識変数を識別します。標識変数は、INDICATORで識別します。
- 変数指定に標識変数を指定する場合、標識変数のデータ型は、SMALLINTに対応するデータ型とします。

- 変数指定が標識変数を含み、その値が負ならば、変数指定の値はNULLとなります。そうでなければ、変数指定の埋込み変数に設定された値が、変数指定の値となります。

USER

- USERを指定した場合、データ型は文字列型となります。値はCONNECT文で指定した認可識別子、またはSET SESSION AUTHORIZATION文で指定した認可識別子となります。

動的パラメタ指定

- 動的パラメタ指定は、動的SQL文にのみ含まれることができます。
- 動的パラメタ指定は、動的SQL文によって用いられるパラメタを識別します。
- 動的パラメタ指定は、選択リストに指定してはいけません。
- 動的パラメタ指定は、集合関数の引数として指定してはいけません。

項目参照

- 項目参照は、プロシジャルーチン定義中のSQL手続き文にのみ指定が可能です。
- 項目参照が項目修飾子を含む場合、単純値指定および単純相手指定のパラメタ名およびSQL変数名は、プロシジャルーチン定義中のSQL手続き文にのみ指定が可能です。

パラメタ名

- パラメタ名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

SQL変数名

- SQL変数名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

ルーチン名

- ルーチン名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

文ラベル

- 文ラベルには、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

使用例

例1

定数

検索条件として、列の値を定数で指定する場合に使用します。

```
WHERE C1 = 10
```

例2

相手指定

SQL埋込みホストプログラムの変数に、検索結果を渡すときに指定します。

```
FETCH CSR1 INTO :TGR
```

例3

動的パラメタ指定

準備可能文の動的SELECT文に動的パラメタ指定を指定します。

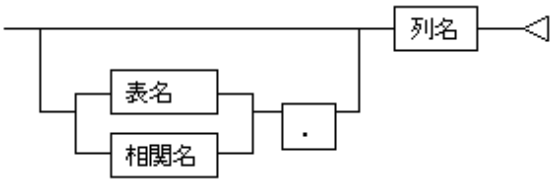
```
SELECT COL1,COL2,COL3 FROM TBL WHERE COL4 = ?
```

2.4 列指定

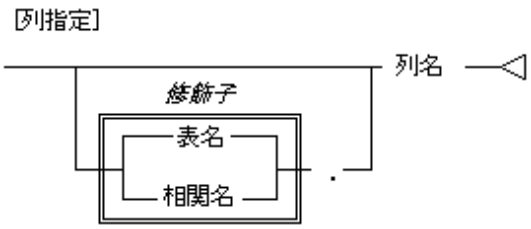
機能

名前で指定された列を参照します。

記述形式



構文要素の構成



参照項番

- 日本語文字列 → “2.1.3 トークン”

一般規則

- 列名を修飾する表名および相関名は、“修飾子”と呼びます。
- 列指定が修飾子を含むならば、列指定は、修飾子に等しい表名、または相関名の有効範囲の中にあることが必要です。
- 列指定が修飾子を含まないならば、有効範囲の中の列指定の列を含む表が表名、または相関名で指定されていることが必要です。
- 表式中に含まれる列指定で指定される修飾子の有効範囲が、表式を含むSQL文または表式ならば、その修飾子に関連する表への“外への参照”といいます。
- 修飾子を省略した場合、同一のFROM句内に指定された複数の表の間に同じ列名があってはいけません。

表名

- 列が含まれる表の名前を指定します。
- 表名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

相関名

- 列が含まれる表の別名を指定します。
- 相関名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

列名

- 列名は、列の名前です。
- 列名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

使用例

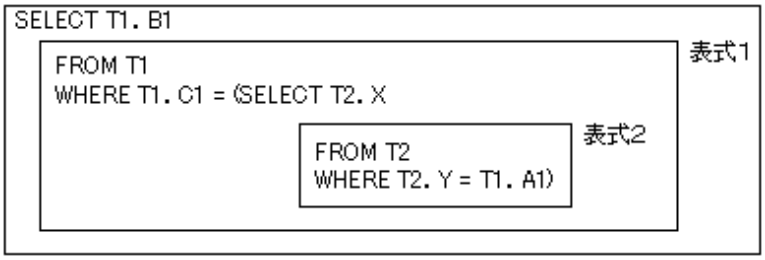
例1

表T1と表T2の両方に同じ列名Aが存在する場合の検索例を示します。この場合は表名、列名を指定します。

```
SELECT T1.A
    FROM T1,T2
    WHERE T1.A = T2.A
```

例2

表式2で指定したT1.A1は、表式2を含む表式1で指定された表の列であるためT1.A1は外への参照といいます。



2.5 関数

機能

関数は、データ型の値を導出します。

関数の種類について、以下に示します。

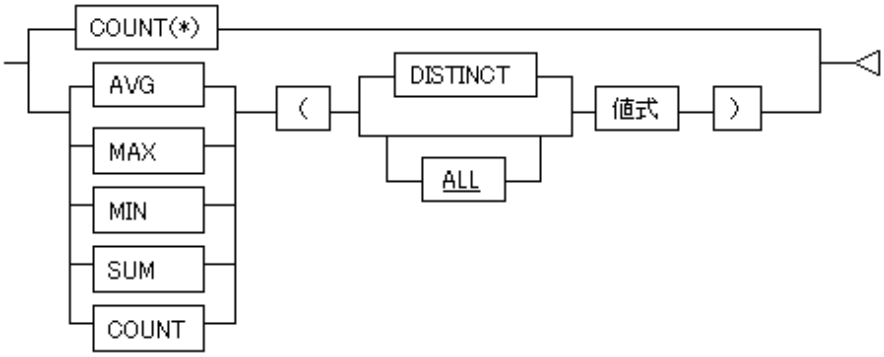
関数	機能
集合関数指定	集合関数指定は、値の合計、平均などの集計を行う関数です。
数値関数	数値関数は、数値型の値となる関数です。
データ列値関数	データ列値関数は、文字列型の値となる関数です。
日時値関数	日時値関数は、日時型の値となる関数です。
ファンクションルーチン指定	ファンクションルーチンを指定します。
XMLQUERY関数	XQuery式を実行し、XMLデータを返却する関数です。

2.5.1 集合関数指定

機能

集合関数指定は、値の合計、平均などの集計を行う関数です。

記述形式



参照項番

- 値式 → “2.11 値式”

一般規則

COUNT(*)

- COUNT(*)を指定した場合、集合関数の引数となる値は、表の全行となります。

AVG

- AVGを指定した集合関数の結果は、引数の平均となります。

MAXまたはMIN

- MAX、またはMINを指定した集合関数の結果は、引数の最大値または最小値となります。
- MAX、またはMINを指定した集合関数の引数のデータ型が文字列、または各国語文字列の場合は、そのデータ型を取り扱う文字コード系(EUCコード、シフトJISコードまたはUNICODE)に基づいて表現したときの数値で比較した結果となります。

SUM

- SUMを指定した集合関数の結果は、引数の合計となります。

COUNT

- COUNTを指定した場合、集合関数の結果は、引数となる値の個数となります。引数となる値が空集合の場合、0となります。

DISTINCT

- DISTINCTを指定した場合、集合関数の引数となる値は、値式の結果がNULL値の場合、それを取り除き、重複する値がある場合は重複を取り除き1つにした結果の集合となります。

ALL

- ALLを指定した場合、集合関数の引数となる値は、値式の結果がNULL値の場合、それを取り除いた結果の集合となります。

値式

- 引数の値式には、列指定が1つ以上指定されていることが必要です。
- 集合関数の引数が外への参照である場合、値式に演算子を指定することはできません。外への参照は、“2.4 列指定”を参照してください。
- AVG、MAX、MIN、またはSUMを指定した集合関数の引数となる値が空集合の場合、結果はNULLとなります。
- 引数の値式に、集合関数、動的パラメタ指定、順序を指定することはできません。

その他の一般規則

- BLOB型は、ALLを指定したCOUNT以外の集合関数に指定することはできません。
- 集合関数の実行結果のデータ型は引数のデータ型により異なります。以下に集合関数の引数のデータ型と結果のデータ型を示します。引数の値式が演算子を含んでいる場合には、値式のデータ変換規則に従って変換した結果が引数のデータ型となります。値式は“2.11 値式”を参照してください。

表2.17 集合関数の引数のデータ型と結果のデータ型

関数	引数データ型	結果データ型
AVG	INTEGER	DECIMAL(12,2) (注)
	SMALLINT	DECIMAL(7,2) (注)
	DECIMAL(p,q)	DECIMAL(m,n) m: MIN(18,p+2) n: MIN(18-(p-q),q+2)
	NUMERIC(p,q)	
	REAL	REAL
	DOUBLE PRECISION	DOUBLE PRECISION
	INTERVAL	INTERVAL
MAX	INTEGER	INTEGER
	SMALLINT	SMALLINT
	DECIMAL(p,q)	DECIMAL(p,q)
	NUMERIC(p,q)	NUMERIC(p,q)
	REAL	REAL
	DOUBLE PRECISION	DOUBLE PRECISION
	CHARACTER	CHARACTER
	VARCHAR	VARCHAR
	NCHAR	NCHAR
	NCHAR VARYING	NCHAR VARYING
	DATE	DATE
	TIME	TIME
	TIMESTAMP	TIMESTAMP
	INTERVAL	INTERVAL
MIN	INTEGER	INTEGER
	SMALLINT	SMALLINT
	DECIMAL(p,q)	DECIMAL(p,q)
	NUMERIC(p,q)	NUMERIC(p,q)
	REAL	REAL
	DOUBLE PRECISION	DOUBLE PRECISION
	CHARACTER	CHARACTER
	VARCHAR	VARCHAR
	NCHAR	NCHAR
	NCHAR VARYING	NCHAR VARYING
	DATE	DATE
	TIME	TIME

関数	引数データ型	結果データ型
	TIMESTAMP	TIMESTAMP
	INTERVAL	INTERVAL
SUM	INTEGER	INTEGER (注)
	SMALLINT	
	DECIMAL(p,q)	DECIMAL(m,n) m: MIN(18,p+7), n: q
	NUMERIC(p,q)	
	REAL	REAL
	DOUBLE PRECISION	DOUBLE PRECISION
	INTERVAL 時間隔先行フィールド精度: p	INTERVAL 時間隔先行フィールド精度: MIN(9,p+7)
COUNT	-	INTEGER

p: 精度

q: 位取り

- : すべてのデータ型

注) 中間精度は、INTEGERです。

使用例

表T1に対して集合関数を使用して検索した例を示します。

表T1

C1	C2	C3	C4
A	10	X	——
B	20	Y	90
D	——	Z	40
B	40	X	70
C	20	Y	40

—— : NULL値を表します。

例1

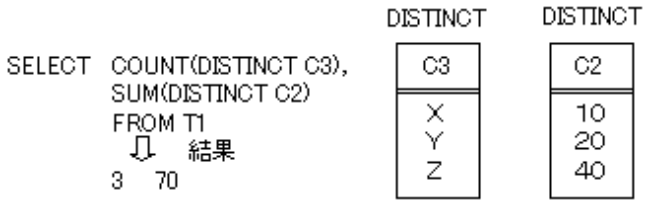
COUNT(*)を指定した例です。

```
SELECT COUNT(*) FROM T1
        ↓ 結果
        5
```

この例では、表T1の行数を求めています。関数COUNT(*)では、NULL値を含んだ行もカウントするので、この結果は5となります。

例2

DISTINCTを指定した例です。

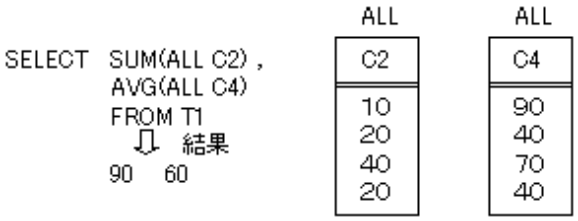


COUNTでは、NULL値を除き、また同じ値の列を1個にします。そのため、結果は3となります。

SUMでも、引数を同様に扱うので結果は70となります。

例3

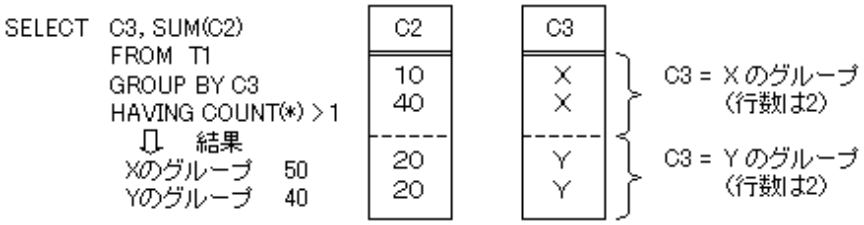
ALLを指定した例です。



この例では、SUMとAVGを使用しています。NULL値を除いた値に対して、それぞれの関数が適用されます。結果として、それぞれ、90、60となります。

例4

GROUP BY句とHAVING句を併用した例です。



この例では、まずGROUP BY句によってC3の値でグループ分けして、HAVING句により行数が2以上のグループを選び出します。結果として、C3がXでありSUM結果が50の行とC3がYでありSUM結果が40の行の2行となります。

2.5.2 数値関数

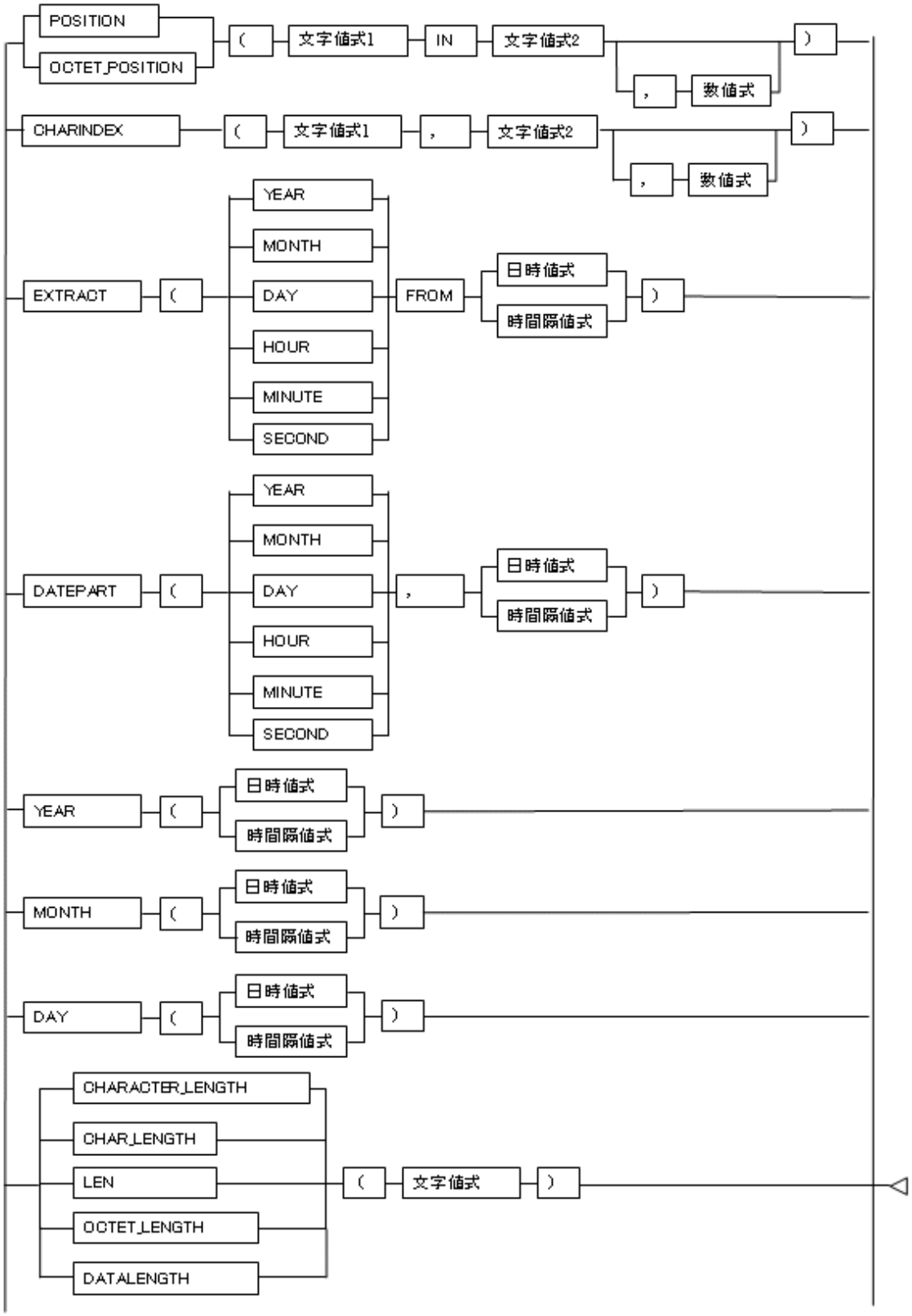
機能

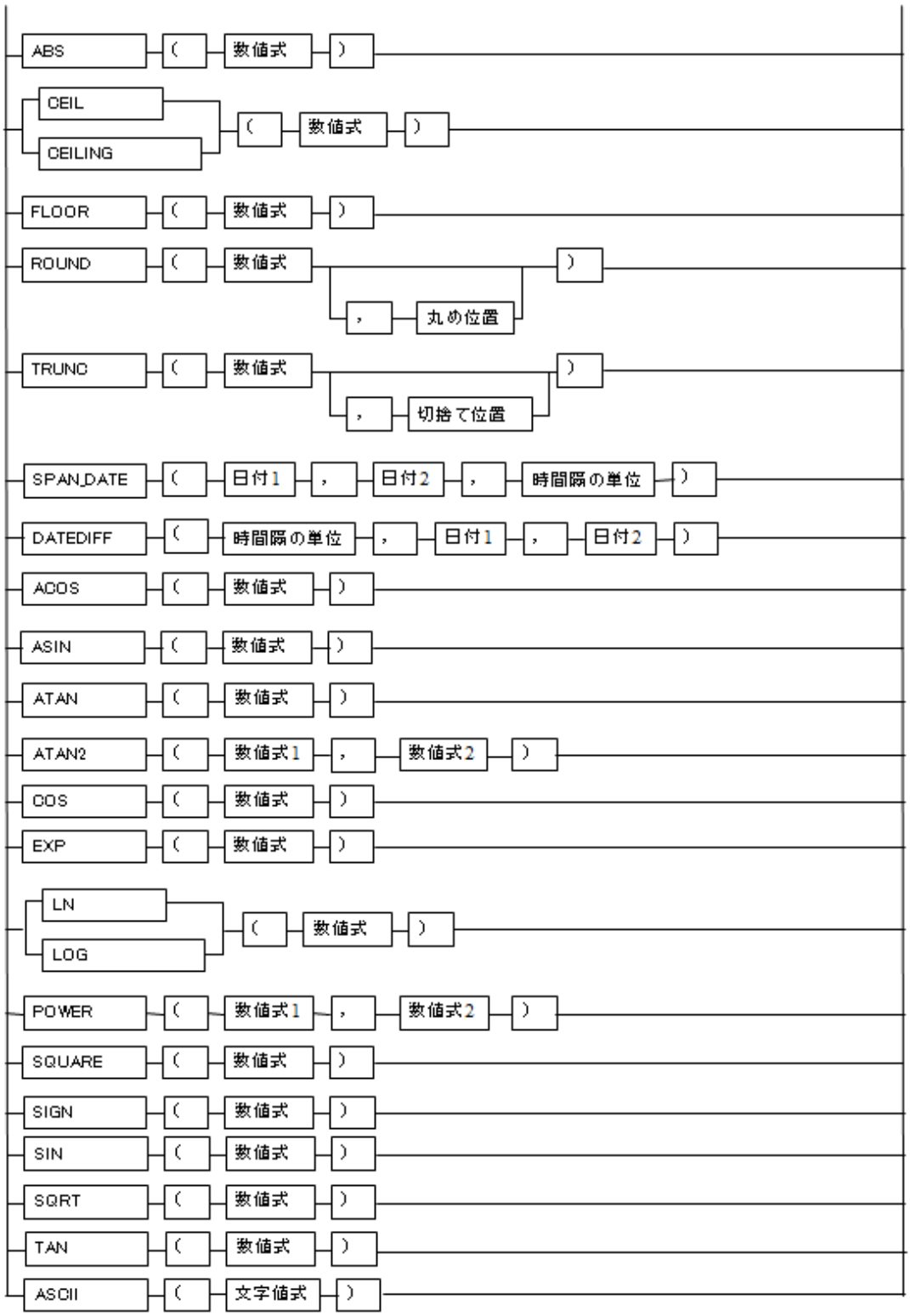
数値関数は、数値型の値となる関数です。

関数	機能
POSITION式	POSITION式は、POSITIONまたはOCTET_POSITIONを指定して、1番目の文字値式の文字列を持つ2番目の文字値式中の文字の開始位置を求めます。
EXTRACT式	EXTRACT式は、抜き出し元の日時値式および時間隔値式から、抜き出しフィールドで指定された日時フィールドに相当する部分を抽出します。
LENGTH式	LENGTH式は、データ列値式の文字数またはバイト数を求めます。
ABS式	ABS式は、指定した数値式の絶対値を求めます。
CEIL式	CEIL式は、指定した数値式以上の最小整数値を求めます。
FLOOR式	FLOOR式は、指定した数値式以下の最大整数値を求めます。
ROUND式	ROUND式は、数値式を丸め位置で四捨五入します。

関数	機能
TRUNC式	TRUNC式は、数値式を切捨て位置で切り捨てます。
SPAN_DATE関数	2つの日付間の時間隔を、指定した時間隔の単位で返却します。
ACOS関数	ACOS関数は、数値式のアークコサイン（逆余弦）を求めます。
ASIN関数	ASIN関数は、数値式のアークサイン（逆正弦）を求めます。
ATAN関数	ATAN関数は、数値式のアークタンジェント（逆正接）を求めます。
ATAN2関数	ATAN2関数は、数値式1、数値式2のアークタンジェント（逆正接）を求めます。
COS関数	COS関数は、数値式のコサイン（余弦）を求めます。
EXP関数	EXP関数は、数値式の指数関数値を求めます。
LN関数	LN関数は、数値式の自然対数を求めます。
POWER関数	POWER関数は、数値式の指定された累乗値を求めます。
SQUARE関数	SQUARE関数は、数値式の指定された2乗の値を求めます。
SIGN関数	SIGN関数は、数値式の符号の標識を返却します。
SIN関数	SIN関数は、数値式のサイン（正弦）を求めます。
SQRT関数	SQRT関数は、数値式の平方根を求めます。
TAN関数	TAN関数は、数値式のタンジェント（正接）を求めます。
ASCII関数	ASCII関数は、文字値式の左端の1バイトのASCIIコードを返却します。

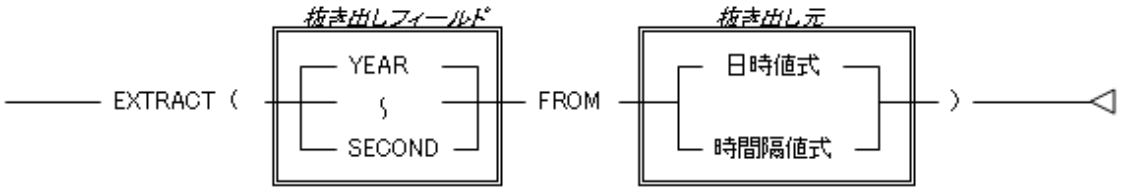
記述形式


POSITION
OCTET_POSITION
(
文字値式1
IN
文字値式2
,
数値式
)
CHARINDEX
(
文字値式1
,
文字値式2
,
数値式
)
EXTRACT
(
YEAR
MONTH
DAY
HOUR
MINUTE
SECOND
FROM
日時値式
時間隔値式
)
DATEPART
(
YEAR
MONTH
DAY
HOUR
MINUTE
SECOND
,
日時値式
時間隔値式
)
YEAR
(
日時値式
時間隔値式
)
MONTH
(
日時値式
時間隔値式
)
DAY
(
日時値式
時間隔値式
)
CHARACTER_LENGTH
CHAR_LENGTH
LEN
OCTET_LENGTH
DATALENGTH
(
文字値式
)


ABS
(
数値式
)
CEIL
CEILING
(
数値式
)
FLOOR
(
数値式
)
ROUND
(
数値式
,
丸め位置
)
TRUNC
(
数値式
,
切捨て位置
)
SPAN_DATE
(
日付1
,
日付2
,
時間隔の単位
)
DATEDIFF
(
時間隔の単位
,
日付1
,
日付2
)
ACOS
(
数値式
)
ASIN
(
数値式
)
ATAN
(
数値式
)
ATAN2
(
数値式1
,
数値式2
)
COS
(
数値式
)
EXP
(
数値式
)
LN
LOG
(
数値式
)
POWER
(
数値式1
,
数値式2
)
SQUARE
(
数値式
)
SIGN
(
数値式
)
SIN
(
数値式
)
SQRT
(
数値式
)
TAN
(
数値式
)
ASCII
(
文字値式
)

構文要素の構成

[EXTRACT式]



参照項番

- 文字値式 → “2.11.2 データ列値式”
- 日時値式 → “2.11.3 日時値式”
- 時間隔値式 → “2.11.4 時間隔値式”
- 数値式 → “2.11.1 数値式”

一般規則

POSITION式

- POSITION式は、文字値式1の文字列を持つ文字値式2中の文字の開始位置を求めます。
- 文字値式2が文字値式1の文字列を含まないならば、結果は0になります。
- 文字値式1の文字列長が0ならば、POSITION式の結果は1となります。
- 数値式には文字値式2の文字列中から文字値式1の文字列の検索を開始する位置を指定します。数値式を省略した場合、または負の値が指定された場合は文字値式2の先頭から検索が行われます。また数値式は整数でなければなりません。
- 数値式がNULLならば、POSITION式の結果はNULLになります。
- POSITIONを指定した場合、数値式には文字値式2の文字列中から文字値式1の文字列の検索を開始する位置を、文字値式2の文字列の先頭からの文字数で指定します。また式の結果は文字数で返されます。
- OCTET_POSITIONを指定した場合、数値式には文字値式2の文字列中から文字値式1の文字列の検索を開始する位置を、文字値式2の文字列の先頭からのバイト数で指定します。また式の結果はバイト数で返されます。
- CHARINDEXは、POSITIONと同じ意味です。
- データ型がNCHAR、NCHAR VARYINGの文字値式2に対してPOSITION、CHARINDEXを使用する場合、文字値式2に含まれるUNICODEの補助文字(1～16面の4バイト文字)を2文字として扱います。動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、1文字として扱うことが可能となります。

> **参照**
>
> 動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

- POSITION式の文字値式1または文字値式2に動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。SQL記述子域のTYPEにVARCHARやNCHAR VARYINGの情報が設定されます。SQL記述子域のLENGTHにVARCHARやNCHAR VARYINGの最大文字数が設定されます。

表2.18 動的パラメタ指定が指定された場合のDESCRIBE情報

文字値式1または文字値式2のデータ型	DESCRIBE情報
CHAR(n)	VARCHAR(Nvc)

文字値式1または文字値式2のデータ型	DESCRIBE情報
VARCHAR(n)	VARCHAR(Nvc)
NCHAR(n)	NCHAR VARYING(Nvn)
NCHAR VARYING(n)	NCHAR VARYING(Nvn)

Nvc:VARCHARのデータの最大長
Nvn:NCHAR VARYINGのデータの最大長

- 文字値式1と文字値式2の両方が動的パラメタ指定の場合はエラーになります。
- 文字値式1または文字値式2がNULLならば、結果もNULLになります。POSITION式の結果のデータ型を以下に示します。

表2.19 POSITION式の結果のデータ型

文字値式1または文字値式2のデータ型	結果のデータ型
CHAR(n)	SMALLINT
VARCHAR(n)	SMALLINT
NCHAR(n)	SMALLINT
NCHAR VARYING(n)	SMALLINT
その他	エラー

- POSITION式の数値式に動的パラメタ指定が指定された場合のDESCRIBE情報はINTEGER型になります。

例

POSITION式

```
POSITION('F' IN 'OFFICE')  →結果は2になります。
POSITION('F' IN 'OFFICE', 3 )  →結果は3になります。
POSITION('C' IN 'JAVA')    →結果は0になります。
POSITION('' IN 'ABC')      →結果は1になります。
POSITION('A' IN 'あいうAB')  →結果は4になります。
OCTET_POSITION('A' IN 'あいうAB')  →結果は7になります。(データベースの文字コード系がEUCコードまたはシフトJISコードの場合)
OCTET_POSITION('A' IN 'あいうAB')  →結果は10になります。(データベースの文字コード系がUNICODEの場合)
CHARINDEX('F','OFFICE') →結果は2になります。
```

EXTRACT式

- EXTRACT式は、抜き出し元の日時値式および時間隔値式から、抜き出しフィールドで指定された日時フィールドに相当する部分を抽出します。
- DATEPARTは、EXTRACTと同じ意味です。
- YEARは、以下の形式でEXTRACTを実行します。

```
EXTRACT(YEAR FROM 抜き出し元の日時値式および時間隔値式)
```

- MONTHは、以下の形式でEXTRACTを実行します。

```
EXTRACT(MONTH FROM 抜き出し元の日時値式および時間隔値式)
```

- DAYは、以下の形式でEXTRACTを実行します。

```
EXTRACT(DAY FROM 抜き出し元の日時値式および時間隔値式)
```

- EXTRACT式の結果のデータ型は、INTEGER型になります。

－ 抜き出し元がNULLならば、EXTRACT式の結果はNULLになります。

－ 抜き出し元が時間隔型で負の値ならば、EXTRACT式の結果は負数になります。

－ EXTRACT式に動的パラメタ指定が指定された場合は、エラーになります。

例

EXTRACT式

```
EXTRACT(YEAR FROM DATE'2007-08-23') →結果は2007になります。
DATEPART(YEAR,DATE'2007-08-23') →結果は2007になります。
YEAR( DATE'2007-08-23') →結果は2007になります。
MONTH( DATE'2007-08-23') →結果は8になります。
DAY( DATE'2007-08-23') →結果は23になります。
```

LENGTH式

－ CHARACTER LENGTH式は、データ列値式の文字数を求めます。ただし、データ列値式に列指定を指定した場合、CHARACTER LENGTH式の結果は列の定義長に等しくなります。

－ OCTET LENGTH式は、データ列値式のバイト数を求めます。

－ CHARACTER_LENGTHとCHAR_LENGTHとLENは同じ意味です。
ただし、LENは指定されたデータ列値式の末尾の空白を除いた文字数を返します。

－ OCTET_LENGTHとDATALENGTHは同じ意味です。

－ データ型がNCHAR、NCHAR VARYINGのデータ列値式に対してCHARACTER_LENGTH、CHAR_LENGTH、LENを使用する場合、データ列値式に含まれるUNICODEの補助文字(1～16面の4バイト文字)を2文字として扱います。動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、1文字として扱うことが可能となります。

参照

動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

－ LENGTH式に動的パラメタ指定が指定された場合は、エラーになります。

－ データ列値式がNULLならば、結果もNULLになります。LENGTH式の結果のデータ型を以下に示します。

表2.20 LENGTH式の結果のデータ型

文字値式のデータ型	結果のデータ型
CHAR(n)	SMALLINT
VARCHAR(n)	SMALLINT
NCHAR(n)	SMALLINT
NCHAR VARYING(n)	SMALLINT
その他	エラー

例

LENGTH式

```
CHAR_LENGTH('ABあいう') →結果は5になります。
OCTET_LENGTH('ABあいう') →結果は8になります。(データベースの文字コード系がEUCコードまたはシフトJISコードの場合)
OCTET_LENGTH('ABあいう') →結果は11になります。(データベースの文字コード系がUNICODEの場合)
LEN(N'ＡＢあいう　') →結果は5になります。
LEN('ABあいう　') →結果は5になります。
DATALENGTH('ABあいう') →結果は8になります。(データベースの文字コード系がEUCコードまたはシフトJISコードの場合
```

ABS式

- ABS式は、指定した数値式の絶対値を求めます。
- 数値式がNULLの場合は、ABS式の結果はNULLとなります。
- ABS式の結果のデータ型は、数値式のデータ型と同じです。
- 数値式のデータ型がSMALLINTまたはINTEGERで、負の最大値を指定した場合、結果となる正の値はSMALLINTまたはINTEGERで表現することができません。このような指定をした場合は、エラーとなりますので注意が必要です。以下の場合がエラーとなります。

数値式のデータ型	指定した値
SMALLINT	-32768
INTEGER	-2147483648

- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、INTEGER型になります。

例

ABS式

```
ABS(123) →結果は123になります。
ABS(123.45) →結果は123.45になります。
ABS(-123.45) →結果は123.45になります。
ABS(-123.45E6) →結果は123.45E6になります。
```

CEIL式

- CEIL式は、指定した数値式以上の最小整数値を求めます。
- CEILINGはCEILと同じ意味です。
- 数値式がNULLの場合は、CEIL式の結果はNULLとなります。
- 数値式の値が負数の場合、0に近い側の整数値が結果となります。
- CEIL式に動的パラメタ指定が指定された場合はエラーになります。
- CEIL式の結果のデータ型は次のようになります。

数値式のデータ型	CEIL式の結果のデータ型
SMALLINT	SMALLINT
INTEGER	INTEGER
DECIMAL(p,q)	DECIMAL(p1,q) p1=MIN(18,p+1)
NUMERIC(p,q)	NUMERIC(p1,q) p1=MIN(18,p+1)
FLOAT(p)	FLOAT(p)
REAL	REAL
DOUBLE PRECISION	DOUBLE PRECISION

例

CEIL式

```
CEIL(123) →結果は123になります。
CEIL(123.45) →結果は124.00になります。
CEIL(-123.45) →結果は-123.00になります。
CEIL(-1.2345E2) →結果は-1.2300E2になります。
CEIL(0.5) →結果は1.0になります。
```

```
CEIL(-0.5) →結果は0になります。
CEILING(123) →結果は123になります。
```

FLOOR式

- FLOOR式は、指定した数値式以下の最大整数値を求めます。
- 数値式がNULLの場合は、FLOOR式の結果はNULLとなります。
- 数値式の値が負数の場合、0より遠い側の整数値が結果となります。
- FLOOR式に動的パラメタ指定が指定された場合はエラーになります。
- FLOOR式の結果のデータ型は次のようになります。

数値式のデータ型	FLOOR式の結果のデータ型
SMALLINT	SMALLINT
INTEGER	INTEGER
DECIMAL(p,q)	DECIMAL(p1,q) p1=MIN(18,p+1)
NUMERIC(p,q)	NUMERIC(p1,q) p1=MIN(18,p+1)
FLOAT(p)	FLOAT(p)
REAL	REAL
DOUBLE PRECISION	DOUBLE PRECISION

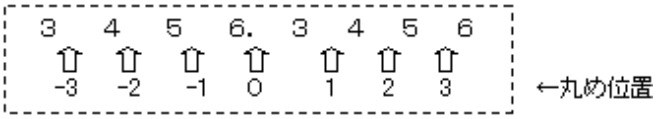
例

FLOOR式

```
FLOOR(123) →結果は123になります。
FLOOR(123.45) →結果は123.00になります。
FLOOR(-123.45) →結果は-124.00になります。
FLOOR(-1.2345E2) →結果は-1.2400E2になります。
FLOOR(0.5) →結果は0.0になります。
FLOOR(-0.5) →結果は-1.0になります。
```

ROUND式

- ROUND式は、数値式を丸め位置で四捨五入します。
- 丸め位置は数値式です。丸め位置には、小数点位置を起点とした有効桁(丸め位置)を指定します。また、-18から+18の範囲の整数値に変換できる値を指定します。丸め位置は、INTEGER型に変換されます。



- 丸め位置を省略した場合は、0を指定したと見なします。
- 数値式または丸め位置がNULLの場合、ROUND式の結果はNULLとなります。
- 数値式のデータ型がSMALLINTまたはINTEGERの場合、四捨五入した結果がSMALLINTまたはINTEGERで表現できない値になった場合はエラーとなります。
- 数値式の値が負の数の場合は、数値式の値を絶対値にしてROUND式で求めた値の符号を反転させた結果と同一です。

```
ROUND(-n) = -ROUND(n) (nは任意の正の数)
```

- ROUND式の結果のデータ型は次のようになります。

数値式のデータ型	ROUND式の結果のデータ型
SMALLINT	SMALLINT
INTEGER	INTEGER
DECIMAL(p,q)	DECIMAL(p1,q1) p1=MIN(18,p+1) q1=q
NUMERIC(p,q)	NUMERIC(p1,q1) p1=MIN(18,p+1) q1=q
FLOAT(p)	FLOAT(p)
REAL	REAL
DOUBLE PRECISION	DOUBLE PRECISION

ー ROUND式の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.21 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
数値式	INTEGER
丸め位置	INTEGER

例

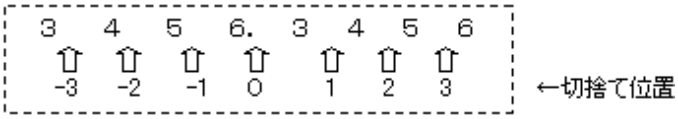
ROUND式

```
ROUND(3456.3456) →結果は3456.0000になります。
ROUND(3456.3456,2) →結果は3456.3500になります。
ROUND(3456.3546,-3) →結果は3000.0000になります。
ROUND(3.456E-1,2) →結果は3.500E-1になります。
ROUND(0.5) →結果は1.0になります。
ROUND(-0.5) →結果は-1.0になります。
ROUND(0.4) →結果は0.0になります。
ROUND(-0.4) →結果は0.0になります。
```

TRUNC式

ー TRUNC式は、数値式を切捨て位置で切り捨てます。

ー 切捨て位置は数値式です。切捨て位置は、小数点位置を起点とした有効桁を指定します。また、-18から+18の範囲の整数値に変換できる値を指定します。切捨て位置は、INTEGER型に変換されます。



ー 引数に数値式以外を指定すると、エラーとなります。

ー 切捨て位置を省略した場合は、0を指定したと見なします。

ー 数値式または切捨て位置がNULLの場合、TRUNC式の結果はNULLとなります。

ー 数値式の値が負数の場合は、数値式の値を絶対値にしてTRUNC式で求めた値の符号を反転させた結果と同一です。

```
TRUNC(-n) = -TRUNC(n) (nは任意の正の数)
```

ー TRUNC式の結果のデータ型は数値式のデータ型と同一になります。

ー TRUNC式の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.22 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
数値式	INTEGER
切捨て位置	INTEGER

例

TRUNC式

```
TRUNC(3456.3456) →結果は3456.0000になります。
TRUNC(3456.3456,2) →結果は3456.3400になります。
TRUNC(3456.3546,-3) →結果は3000.0000になります。
TRUNC(3.456E-1,2) →結果は3.400E-1になります。
TRUNC(0.4) →結果は0.0になります。
TRUNC(-0.4) →結果は0.0になります。
TRUNC(0.5) →結果は0.0になります。
TRUNC(-0.5) →結果は0.0になります。
```

SPAN_DATE関数

- SPAN_DATE関数は、2つの日付間の時間隔を、指定した時間隔の単位で返却します。
- DATEDIFFは、SPAN_DATEと同じ意味です。
- SPAN_DATE関数は、日付2-日付1の結果を返却します。日付1が日付2よりも新しい場合は、実行結果は負数となります。
- 日付1、日付2は日時値式です。データ型はDATE型に代入可能でなければなりません。
- 日付1または日付2がNULL値である場合、SPAN_DATE関数の結果はNULLとなります。
- SPAN_DATE関数の結果のデータ型はINTEGER型です。
- 時間隔の単位は、文字列型の値指定です。日付間の時間隔を求める単位を指定します。時間隔の単位に指定できる値とSPAN_DATE関数の結果は、以下のようになります。

時間隔の単位	SPAN_DATE関数の結果
YEAR	日付間の年の変わる回数を返却します。
MONTH	日付間の月の変わる回数を返却します。
DAY	日付間の日の変わる回数を返却します。

- SPAN_DATE関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.23 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
日付1	DATE
日付2	DATE
時間隔の単位	CHAR(5)

例

SPAN_DATE関数

```
SPAN_DATE(DATE'2006-08-01',DATE'2007-09-20','YEAR')→1になります。
SPAN_DATE(CURRENT_DATE,DATE'2007-11-10','MONTH')→3になります。
        現在の日付を2007-08-22とします。
SPAN_DATE(DATE'2007-09-10',DATE'2007-08-22','DAY')→-19になります。
DATEDIFF(YEAR,DATE'2006-08-01',DATE'2007-09-20')→1になります。
```

ACOS関数

- ACOS関数は、数値式のアークコサイン(逆余弦)を求めます。

- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- ACOS関数の結果は0～π（ラジアン単位）の範囲です(π:円周率)。
- 数値式の有効な範囲は-1～1です。この範囲外の数値式を指定するとエラーとなります。
- 数値式がNULLの場合は、ACOS関数の結果はNULLとなります。
- ACOS関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

ACOS関数

```
ACOS(-1.0) →結果は3.141593E+00になります。
```

ASIN関数

- ASIN関数は、数値式のアークサイン（逆正弦）を求めます。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- ASIN関数の結果は-π/2～π/2(ラジアン単位）の範囲です(π:円周率)。
- 数値式の有効な範囲は-1～1です。この範囲外の数値式を指定するとエラーとなります。
- 数値式がNULLの場合は、ASIN関数の結果はNULLとなります。
- ASIN関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

ASIN関数

```
ASIN(1.0) →結果は1.570796E+00になります。
```

ATAN関数

- ATAN関数は、数値式のアークタンジェント（逆正接）を求めます。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- ATAN関数の結果は-π/2～π/2(ラジアン単位）の範囲です(π:円周率)。
- 数値式がNULLの場合は、ATAN関数の結果はNULLとなります。
- ATAN関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

ATAN関数

```
ATAN(1.0) →結果は7.853982E-01になります。
```

ATAN2関数

- ATAN2関数は、数値式1を数値式2で除算した値のアークタンジェント（逆正接）を求めます。

  ATAN2関数とATAN関数の関係を以下に示します。

数値式1の値	数値式2の値	ATAN2関数の演算結果
n1 >= 0	n2 > 0	ATAN(n1/n2)
n1 >= 0	n2 < 0	ATAN(n1/n2) + π
n1 < 0	n2 > 0	ATAN(n1/n2)
n1 < 0	n2 < 0	ATAN(n1/n2) - π
n1 > 0	n2 = 0	π/2
n1 < 0	n2 = 0	−π/2
n1 = 0	n2 = 0	エラーが発生します

- 数値式1と数値式2には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- ATAN2関数の結果は-π～π(ラジアン単位)の範囲です(π:円周率)。
- 数値式1と数値式2がNULLの場合、またはどちらかがNULLの場合はATAN2関数の結果はNULLとなります。
- ATAN2関数の結果のデータ型はDOUBLE PRECISION型です。
- ATAN2関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.24 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
数値式1	DOUBLE PRECISION
数値式2	DOUBLE PRECISION

例

ATAN2関数

```
ATAN2(1.0,2.0) →結果は4.636476E-01になります。
```

COS関数

- COS関数は、数値式(ラジアン単位)のコサイン(余弦)を求めます。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- 数値式の有効な範囲は、-3.53E+15より大きく、+3.53E+15未満です。この範囲外の数値式を指定するとエラーとなります。
- COS関数の結果は-1～1の範囲です。
- 数値式がNULLの場合は、COS関数の結果はNULLとなります。
- COS関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

COS関数

```
COS(3.141592654) →結果は-1.000000E+00になります。
```

EXP関数

- EXP関数は、数値式の指数関数値を求めます。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。

- 演算結果がDOUBLE PRECISION型の最大値を超えた場合、エラーが発生します。
- 数値式がNULLの場合は、EXP関数の結果はNULLとなります。
- EXP関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

EXP関数

```
EXP(10) →結果は2.202647E+04になります。
```

LN関数

- LN関数は、数値式の自然対数を求めます。
- LOGはLNと同じ意味です。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- 数値式の有効な範囲は正の値です。この範囲外の数値式を指定するとエラーとなります。
- 演算結果がDOUBLE PRECISION型の最小値より小さい場合、または、最大値を超えた場合、エラーが発生します。
- 数値式がNULLの場合は、LN関数の結果はNULLとなります。
- LN関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

LN関数

```
LN(15) →結果は2.708050E+00になります。
LOG(15) →結果は2.708050E+00になります。
```

SQUARE関数

- SQUARE関数は、数値式を2乗した値を求めます。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- 演算結果がDOUBLE PRECISION型の最小値より小さい場合、または、最大値を超えた場合、エラーが発生します。
- 数値式がNULLの場合、SQUARE関数の結果はNULLとなります。
- SQUARE関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

SQUARE関数

```
SQUARE(2) →結果は4.000000E+00になります。
```

POWER関数

- POWER関数は、数値式1を数値式2の値分累乗した値を求めます。
- 数値式1と数値式2には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- 演算結果がDOUBLE PRECISION型の最小値より小さい場合、または、最大値を超えた場合、エラーが発生します。
- 数値式1が負の値の場合、数値式2は整数でなければなりません。
- 数値式1が0の場合、数値式2は0または正の値でなければなりません。

- 数値式1と数値式2がともにNULLの場合、またはどちらかがNULLの場合は、POWER関数の結果はNULLとなります。
- POWER関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.25 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
数値式1	DOUBLE PRECISION
数値式2	DOUBLE PRECISION

- POWER関数の結果のデータ型はDOUBLE PRECISION型です。

例

POWER関数

```
POWER(2,3) →結果は8.000000E+00になります。
```

SIGN関数

- SIGN関数は、数値式の符号の標識を返却します。
- 数値式が負の値の場合は-1が返されます。0の場合は0が返されます。正の値の場合は1が返されます。
- 数値式がNULLの場合は、SIGN関数の結果はNULLとなります。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、INTEGER型になります。
- SIGN関数の結果のデータ型を以下に示します。

数値式のデータ型	SIGN関数の結果のデータ型
SMALLINT	SMALLINT
INTEGER	INTEGER
DECIMAL(p,q)	DECIMAL(1,0)
NUMERIC(p,q)	NUMERIC(1,0)
FLOAT(p)	FLOAT(p)
REAL	REAL
DOUBLE PRECISION	DOUBLE PRECISION

例

SIGN関数

```
SIGN(-15) →結果は-1になります。
```

SIN関数

- SIN関数は、数値式(ラジアン単位)のサイン(正弦)を求めます。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- 数値式の有効な範囲は、-3.53E+15より大きく、+3.53E+15未満です。この範囲外の数値式を指定するとエラーとなります。
- SIN関数の結果は-1～1の範囲です。
- 数値式がNULLの場合は、SIN関数の結果はNULLとなります。
- SIN関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

SIN関数

```
SIN(3.141592654) →結果は-4.102069E-10になります。
```

SQRT関数

- SQRT関数は、数値式の平方根を求めます。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- 数値式は0以上でなければなりません。
- 数値式がNULLの場合は、SQRT関数の結果はNULLとなります。
- SQRT関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

SQRT関数

```
SQRT(9) →結果は3.000000E+00になります。
```

TAN関数

- TAN関数は、数値式(ラジアン単位)のタンジェント(正接)を求めます。
- 数値式には真数型・概数型が指定可能です。データ型はDOUBLE PRECISION型に代入可能でなければなりません。
- 数値式の有効な範囲は、-3.53E+15より大きく、+3.53E+15未満です。この範囲外の数値式を指定するとエラーとなります。
- 演算結果がDOUBLE PRECISION型の最小値より小さい場合、または、最大値を超えた場合、エラーが発生します。
- 数値式がNULLの場合は、TAN関数の結果はNULLとなります。
- TAN関数の結果のデータ型はDOUBLE PRECISION型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、DOUBLE PRECISION型になります。

例

TAN関数

```
TAN(3.141592654) →結果は4.102069E-10になります。
```

ASCII関数

- ASCII関数は、文字値式の左端の1バイトのASCIIコードを返却します。
- 文字値式がNULLの場合は、ASCII関数の結果はNULLとなります。
- ASCII関数の結果のデータ型はINTEGER型です。
- ASCII関数に動的パラメタ指定が指定された場合は、エラーになります。

例

ASCII関数

```
ASCII('A')  →結果は65になります。
```

備考.データベースの文字コード系がシフトJISコードの場合の例です。データベースの文字コード系により結果は異なることがあります。

注意

文字値式にマルチバイト文字を指定した場合、マルチバイト文字の先頭の1バイトの文字コードが返却されるため、文字値式へのマルチバイト文字の指定は推奨しません。

使用例

例1

POSITION式(N1が'アダム電気'の場合。なお、N1のデータ型は各国語文字列型とします。)

```
SELECT POSITION(N'電気' IN N1) INTO :POS1 FROM S1.T1
→結果は4になります。
```

例2

POSITION式(C1が'ADAM電気(株)'の場合。なお、C1のデータ型は文字列型とします。ADAMは、1バイトの英数字です。)

```
SELECT POSITION('(株)' IN C1) INTO :POS1 FROM S1.T1
→結果は7になります。
```

例3

EXTRACT式(C1がDATE型で、'2007-04-17'の場合)

```
SELECT EXTRACT(DAY FROM C1) INTO :DAY1 FROM S1.T1
→結果は17になります。
```

例4

LENGTH式(N1が'アダム電気'の場合。なお、N1のデータ型は各国語文字列型とします。)

```
SELECT CHAR_LENGTH(N1) INTO :LEN1 FROM S1.T1
→結果は5になります。
```

例5

LENGTH式(C1が'ADAM電気(株)'の場合。なお、C1のデータ型は文字列型とします。ADAMは1バイトの英数字です。)

```
SELECT CHAR_LENGTH(C1) INTO :LEN1 FROM S1.T1
→結果は9になります。
```

例6

ABS式(最低気温が-38.45の場合の絶対値を求めます。)

```
SELECT 地区コード , ABS(最低気温) FROM 気象情報管理表
→結果は38.45 になります。
```

例7

CEIL式(最高気温が27.255で最低気温が-21.235をその値以上の最小整数値にします。)

```
SELECT 地区名, CEIL(最高気温), CEIL(最低気温) FROM 気象情報管理表
→結果は、最高気温が28.000で最低気温が-21.000になります。
```

例8

FLOOR式(最高気温が27.255で、最低気温が-21.235をその値以下の最大整数値にします。)

```
SELECT 地区名, FLOOR(最高気温), FLOOR(最低気温) FROM 気象情報管理表
→結果は、最高気温が27.000で最低気温が-22.000になります。
```

例9

ROUND式(最高気温が27.255を小数第2位で四捨五入します。)

```
SELECT 地区名, ROUND(最高気温, 1) FROM 気象情報管理表
→結果は、最高気温が27.300になります。
```

例10

TRUNC式(最高気温が27.255を小数第2位で切捨てます。)

```
SELECT 地区名, TRUNC(最高気温, 1) FROM 気象情報管理表
→結果は、最高気温が27.200になります。
```

例11

SPAN_DATE関数(発注日“2007-04-01”から現在の日付“2007-04-10”までの経過日数を求めます。)

```
SELECT 発注番号,SPAN_DATE(CURRENT_DATE,発注日,'DAY') AS 経過日数
    FROM 発注表
    WHERE 納品日 IS NULL
    ORDER BY 経過日数
→結果は、経過日数が9になります。
```

例12

算術関数(C1の値が1.0である場合)

```
SELECT ACOS( C1 ) FROM T  →0.000000E+00が返されます。
SELECT ASIN( C1 ) FROM T  →1.570796E+00が返されます。
SELECT ATAN( C1 ) FROM T  →7.853982E-01が返されます。
SELECT COS( C1 ) FROM T   →5.403023E-01が返されます。
SELECT SIN( C1 ) FROM T   →8.414710E-01が返されます。
SELECT TAN( C1 ) FROM T   →1.557408E+00が返されます。
SELECT LN( C1 ) FROM T    →0.000000E+00が返されます。
SELECT EXP( C1 ) FROM T   →2.718282E+00が返されます。
SELECT SIGN( C1 ) FROM T  →+1.が返されます。
SELECT SQRT( C1 ) FROM T  →1.000000E+00が返されます。
SELECT SQUARE( C1 ) FROM T   →1.000000E+00が返されます。
```

算術関数(C1の値が2、C2の値が3である場合)

```
SELECT POWER( C1, C2 ) FROM T   →8.000000E+00が返されます。
```

例13

ASCII関数('A'で始まる名前の従業員の数を求めます。)

```
SELECT COUNT(*) FROM 従業員表 WHERE ASCII(SUBSTRING(名前 FROM 1 FOR 1)) = 65
→'A'で始まる名前が5件の場合、結果は5になります。
```

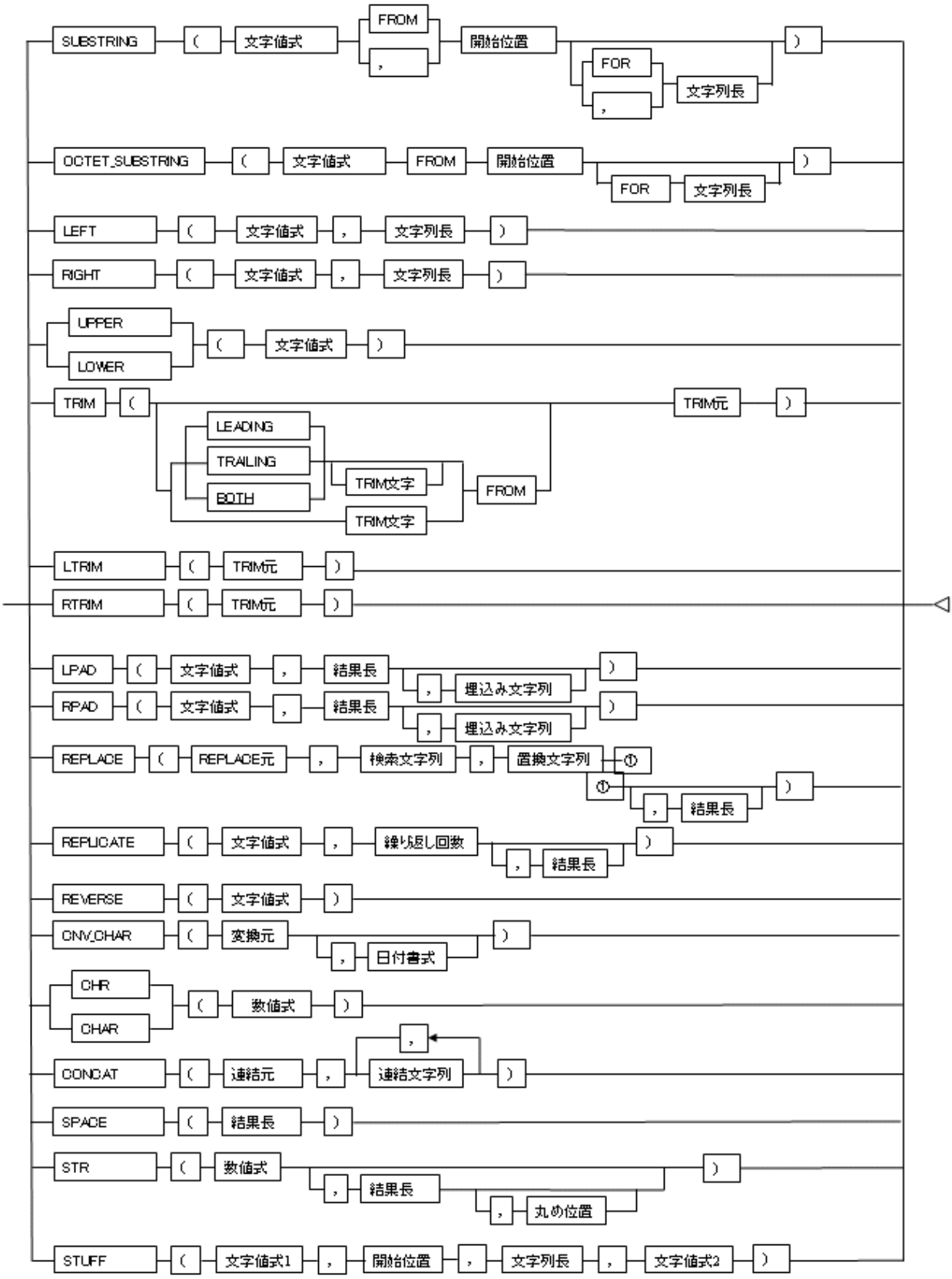
2.5.3 データ列値関数

機能

データ列値関数は、文字列型の値となる関数です。

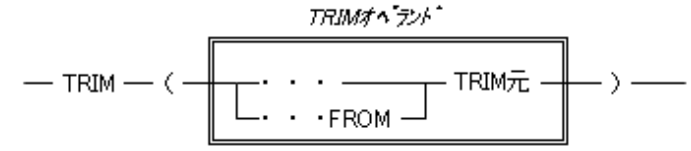
関数	機能
文字部分列関数	文字部分列関数は、SUBSTRINGまたはOCTET_SUBSTRINGを指定して、文字値式の開始位置から、文字列長分の部分列を求めます。
大文字小文字変換	大文字小文字変換は、UPPERまたはLOWERを指定して、文字を大文字または小文字に変換します。
TRIM関数	TRIM関数は、指定された文字を取り除きます。
LTRIM関数	LTRIM関数は、TRIM元の前端から連続する空白を削除します。
RTRIM関数	RTRIM関数は、TRIM元の後端から連続する空白を削除します。
LPAD式	LPAD式は、文字値式の左側に、全体の文字列数が結果長になるまで、埋込み文字列をサイクリックに埋め込んで、結果長数分の文字列を返します。
RPAD式	RPAD式は、文字値式の右側に、全体の文字列数が結果長になるまで、埋込み文字列をサイクリックに埋め込んで、結果長数分の文字列を返します。
REPLACE式	REPLACE式はREPLACE元の文字列中に含まれる検索文字列のすべてを置換文字列に置き換えた結果を返却します。
REPLICATE式	REPLICATE式は、指定された結果文字数の範囲内で、指定された回数だけ、値式を繰り返します。
REVERSE式	REVERSE式は、値式を逆に並び替えたものを返します。
CNV_CHAR関数	日時値式のデータを指定された書式に従って文字型に変換します。
CHR関数	CHR関数は、数値式を文字コードとして該当するASCII文字を返却します。
CONCAT関数	CONCAT関数は、連結元に連結文字列を結合して返却します。
SPACE関数	SPACE関数は、指定された結果長の空白文字列を返却します。
STR関数	STR関数は、指定された数値式のデータを文字型に変換します。
STUFF関数	STUFF関数は、文字値式1に対して開始位置から文字列長分の文字を削除し、文字値式2を挿入して、部分的に入れ替えた文字列を返却します。

記述形式


SUBSTRING
(
文字値式
FROM
,
開始位置
FOR
,
文字列長
)
OCTET_SUBSTRING
(
文字値式
FROM
開始位置
FOR
文字列長
)
LEFT
(
文字値式
,
文字列長
)
RIGHT
(
文字値式
,
文字列長
)
UPPER
LOWER
(
文字値式
)
TRIM
(
LEADING
TRAILING
BOTH
TRIM文字
TRIM文字
FROM
TRIM元
)
LTRIM
(
TRIM元
)
RTRIM
(
TRIM元
)
LPAD
(
文字値式
,
結果長
,
埋込み文字列
)
RPAD
(
文字値式
,
結果長
,
埋込み文字列
)
REPLACE
(
REPLACE元
,
検索文字列
,
置換文字列
①
①
,
結果長
)
REPLICATE
(
文字値式
,
繰り返し回数
,
結果長
)
REVERSE
(
文字値式
)
CNV_CHAR
(
変換元
,
日付書式
)
CHR
CHAR
(
数値式
)
,
CONCAT
(
連結元
,
連結文字列
)
SPACE
(
結果長
)
STR
(
数値式
,
結果長
,
丸め位置
)
STUFF
(
文字値式1
,
開始位置
,
文字列長
,
文字値式2
)

構文要素の構成

[TRIM関数]



[LTRIM関数]

[RTRIM関数]

参照項番

- 文字値式 → “2.11.2 データ列値式”
- 日付書式 → “2.5.4 日時値関数”
- 数値式 → “2.11.1 数値式”

一般規則

文字部分列関数(SUBSTRING、OCTET_SUBSTRING、LEFTまたはRIGHT)

- 文字部分列関数は、SUBSTRINGまたはOCTET_SUBSTRINGを指定して、文字値式の開始位置から文字列長分の部分列を求めます。
- LEFTは文字値式の先頭から文字列長分の部分列を求めます。
- RIGHTは文字値式の終端から文字列長分の部分列を求めます。
- 開始位置と文字列長は数値式で指定します。開始位置と文字列長はINTEGER型に変換されます。
- SUBSTRINGを指定した場合、開始位置と文字列長はそれぞれ文字数を指定します。
- OCTET_SUBSTRINGを指定した場合、開始位置と文字列長はそれぞれバイト数を指定します。
- LEFTまたはRIGHTを指定した場合、文字列長は文字数を指定します。
- SUBSTRINGまたはOCTET_SUBSTRINGを指定して、文字列長が省略された場合は、文字値式の最後の文字までの部分列を求めます。LEFTまたはRIGHTを指定した場合、文字列長は省略できません。
- 開始位置と文字列長の和が文字値式の長さより大きければ、文字部分列関数の結果は開始位置から文字値式の最後の文字までの部分列を求めます。
- 開始位置が文字値式の長さより大きければ、文字部分列関数の結果は長さ0の文字列になります。
- 開始位置に負の値が指定された場合、負の位置からの部分列が返却されます。
- 文字列長が負の値ならば、エラーになります。
- 開始位置または文字列長がNULLならば、結果もNULLになります。

- データ型がCHAR、VARCHAR、NCHAR、NVARCHARの文字値式に対してOCTET_SUBSTRINGを使用する場合、開始位置や文字列長の値によって、文字値式に含まれるマルチバイト文字が断片化することがあります。データ型がCHAR、VARCHARの場合、このようなマルチバイト文字の断片はブランクに置き換えられて返却されます。データ型がNCHAR、NVARCHARの場合、このようなマルチバイト文字の断片は無視されます。

- データ型がNCHAR、NCHAR VARYINGの文字値式に対してSUBSTRING、LEFT、RIGHTを使用する場合、開始位置や文字列長の値によって、文字値式に含まれるUNICODEの補助文字(1～16面の4バイト文字)が断片化することがあります。断片は無視されます。動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、1文字として扱うことが可能となり、断片化しなくなります。

> **参照**
>
> 動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

- 文字部分列関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。SQL記述子域のTYPEにINTEGERの情報が設定されます。SQL記述子域のLENGTHにINTEGERのデータ長が設定されます。

表2.26 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
文字値式	エラー
開始位置	INTEGER
文字列長	INTEGER

- 文字値式がNULLならば、結果もNULLになります。文字部分列関数の結果のデータ型を以下に示します。

表2.27 文字部分列関数の結果のデータ型

値式のデータ型	結果のデータ型
CHAR(n)	VARCHAR(MIN(Nvc,n))
VARCHAR(n)	VARCHAR(n)
NCHAR(n)	NCHAR VARYING(MIN(Nvn,n))
NCHAR VARYING(n)	NCHAR VARYING(n)
その他	エラー

Nvc: VARCHARのデータの最大長

Nvn: NCHAR VARYINGのデータの最大長

例

文字部分列関数

```
SUBSTRING('Symfoware' FROM 6 FOR 3)  →結果は 'war'になります。
SUBSTRING('Symfoware' FROM 6)        →結果は 'ware'になります。
SUBSTRING('Symfoware' FROM 10)       →結果は ''になります。
SUBSTRING('Symfoware' FROM 6 FOR -1) →結果は エラーになります。
SUBSTRING('Symfoware' FROM -1 FOR 5) →結果は 'Sym'になります。
SUBSTRING('あいうAB' FROM 3 FOR 2)   →結果は 'うA'になります。
OCTET_SUBSTRING('あいうAB' FROM 3 FOR 2)→結果は 'い'になります。(データベースの文字コード系がEUCコードまたはシフトJISコードの場合)
OCTET_SUBSTRING('あいうAB' FROM 4 FOR 3)→結果は 'い'になります。(データベースの文字コード系がUNICODEの場合)
OCTET_SUBSTRING('あいうAB' FROM 2 FOR 4)→結果は ' い 'になります。(データベースの文字コード系がEUCコードまたはシフトJISコードの場合)(注)
```

```
LEFT('Symfoware', 3)          →結果は 'Sym'になります。
RIGHT('Symfoware', 3)         →結果は 'are'になります。
```

注) この例では、“い”の前後はマルチバイト文字の断片のためブランクに置き換えられます。

大文字小文字変換(UPPERまたはLOWER)

- 大文字小文字変換は、指定された文字を大文字または小文字に変換します。
- 大文字小文字変換には、UPPERとLOWERがあります。
- UPPERが指定されると、大文字小文字変換の結果は、小文字を大文字に置き換えた値になります。
- LOWERが指定されると、大文字小文字変換の結果は、大文字を小文字に置き換えた値になります。
- 大文字小文字変換に動的パラメタ指定が指定された場合は、エラーになります。
- 文字値式がNULLならば、結果もNULLになります。
- 大文字小文字変換の結果のデータ型を以下に示します。

表2.28 大文字小文字変換の結果のデータ型

値式のデータ型	結果のデータ型
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
その他	エラー

例

大文字小文字変換

```
UPPER('Symfoware') →結果は 'SYMFOWARE'になります。
LOWER('Symfoware') →結果は 'symfoware'になります。
```

TRIM関数(TRIM)

- TRIM関数は、指定された文字を取り除きます。
- LEADING、TRAILINGおよびBOTHをTRIM指定と呼びます。
- TRIM指定が省略された場合、BOTHが指定されたものとみなされます。
- TRIM文字は、文字値式で指定します。TRIM文字が省略された場合、空白が指定されたものとみなされます。
- TRIM文字の長さが1でなければ、エラーになります。
- TRIM元は、文字値式で指定します。
- LEADINGが指定されると、TRIM関数の結果は、TRIM元の前端からTRIM文字を取り除いた値になります。同一文字が前端にある場合は、すべて取られます。
- TRAILINGが指定されると、TRIM関数の結果は、TRIM元の後端からTRIM文字を取り除いた値になります。同一文字が後端にある場合は、すべて取られます。
- BOTHが指定されると、TRIM関数の結果は、TRIM元の両端からTRIM文字を取り除いた値になります。同一文字が両端にある場合は、すべて取られます。
- データ型がNCHAR、NCHAR VARYINGのTRIM文字にUNICODEの補助文字(1～16面の4バイト文字)を使用したい場合は、動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定してください。

> **参照**
>
> 動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

- TRIM関数に動的パラメタ指定が指定された場合は、エラーになります。

－ TRIM文字またはTRIM元がNULLならば、結果もNULLになります。

－ TRIMオペランドは、TRIM関数で指定する引数で、TRIM文字とTRIM元を指定します。

－ TRIM関数の結果のデータ型を以下に示します。

表2.29 TRIM関数の結果のデータ型

TRIM元のデータ型	結果のデータ型
CHAR(n)	VARCHAR(MIN(Nvc,n))
VARCHAR(n)	VARCHAR(n)
NCHAR(n)	NCHAR VARYING(MIN(Nvn,n))
NCHAR VARYING(n)	NCHAR VARYING(n)
その他	エラー

Nvc: VARCHARのデータの最大長

Nvn: NCHAR VARYINGのデータの最大長

例

TRIM関数

```
TRIM(' rdb ') →結果は 'rdb'になります。
TRIM(N' ＲＤＢ ') →結果はN'ＲＤＢ'になります。
TRIM('a' FROM 'aaaBBaBBaaa') →結果は'BBaBB'になります。
TRIM(BOTH 'a' FROM 'aaaBBaBBaaa') →結果は'BBaBB'になります。
TRIM(LEADING '0' FROM '00011232100') →結果は'11232100'になります。
TRIM(TRAILING '0' FROM '00011232100') →結果は'000112321'になります。
```

LTRIM関数

－ LTRIM関数は、TRIM元の前端から連続する空白を削除します。

－ TRIM元は、文字値式で指定します。

－ LTRIM関数は、以下の形式でTRIM関数を実行します。

```
TRIM( LEADING FROM 文字値式 )
```

－ LTRIM関数に動的パラメタ指定が指定された場合は、エラーになります。

－ 文字値式がNULLならば、結果もNULLになります。

－ TRIMオペランドは、LTRIM関数で指定する引数で、TRIM元を指定します。

－ LTRIM関数の結果のデータ型を以下に示します。

表2.30 LTRIM関数の結果のデータ型

文字値式のデータ型	結果のデータ型
CHAR(n)	VARCHAR(MIN(Nvc,n))
VARCHAR(n)	VARCHAR(n)
NCHAR(n)	NCHAR VARYING(MIN(Nvn,n))
NCHAR VARYING(n)	NCHAR VARYING(n)
その他	エラー

例

LTRIM関数

```
LTRIM('  ABCDEF  ') →結果は'ABCDEF  'になります。
```

RTRIM関数

- RTRIM関数は、TRIM元の後端から連続する空白を削除します。
- TRIM元は、文字値式で指定します。
- RTRIM関数は、以下の形式でTRIM関数を実行します。

```
TRIM( TRAILING FROM 文字値式 )
```

- RTRIM関数に動的パラメタ指定が指定された場合は、エラーになります。
- 文字値式がNULLならば、結果もNULLになります。
- TRIMオペランドは、RTRIM関数で指定する引数で、TRIM元を指定します。
- RTRIM関数の結果のデータ型を以下に示します。

表2.31 RTRIM関数の結果のデータ型

文字値式のデータ型	結果のデータ型
CHAR(n)	VARCHAR(MIN(Nvc,n))
VARCHAR(n)	VARCHAR(n)
NCHAR(n)	NCHAR VARYING(MIN(Nvn,n))
NCHAR VARYING(n)	NCHAR VARYING(n)
その他	エラー

例

RTRIM関数

```
RTRIM('  ABCDEF  ') →結果は'  ABCDEF'になります。
```

LPAD式

- LPAD式は、文字値式の左側に、全体の文字列数が結果長になるまで、埋込み文字列をサイクリックに埋め込んで、結果長数分の文字列を返します。
- 埋込み文字列は、文字値式で指定します。
- 結果長は定数で指定します。整数でなければなりません。指定できる値は、文字値式のデータ型の最大長以下でなければなりません。文字値式のデータ型がCHAR、VARCHARの場合はバイト長で指定します。NCHAR、NCHAR VARYINGの場合は文字数で指定します。結果長は、INTEGER型に変換されます。
- 結果長に0または負数を指定するとエラーとなります。
- LPAD式に動的パラメタ指定が指定された場合は、エラーになります。
- 埋込み文字列を省略すると空白を指定したものと見なします。
- 文字値式がNULLあるいは長さ0の文字列である場合は、LPAD式の結果はNULLとなります。
- 埋込み文字列がNULLあるいは長さ0の文字列である場合は、LPAD式の結果はNULLとなります。
- 文字値式の文字列数が結果長より大きい場合、文字値式の先頭から結果長数分の文字列を返却します。
- 文字値式のデータ型がCHAR、VARCHARのとき、1文字が2バイト以上で構成される文字を埋込み文字列に指定すると、結果長の指定によっては、文字の埋め込み後に空きができることがあります。そのような場合は、文字の代わりに1バイトの空白が埋め込まれます。
- データ型がNCHAR、NCHAR VARYINGの文字値式や埋込み文字列に対してLPAD式を使用する場合、文字値式や埋込み文字列に含まれるUNICODEの補助文字(1～16面の4バイト文字)を2文字として扱います。動作環境

ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、1文字として扱うことが可能となります。

参照

動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

― LPAD式の引数の組み合わせおよび結果のデータ型は次のようになります。

文字値式のデータ型	埋込み文字列のデータ型	結果長	LPAD式の結果のデータ型
CHAR	CHAR または VARCHAR	n: バイト数 INTEGERに変換。	CHAR(n)
VARCHAR	CHAR または VARCHAR	n: バイト数 INTEGERに変換。	VARCHAR(n)
NCHAR	NCHAR または NCHAR VARYING	n: 文字数 INTEGERに変換。	NCHAR(n)
NCHAR VARYING	NCHAR または NCHAR VARYING	n: 文字数 INTEGERに変換。	NCHARVARYING(n)

― 動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”が“1”の場合、LPAD式の引数の組み合わせおよび結果のデータ型は次のようになります。

文字値式のデータ型	埋込み文字列のデータ型	結果長	LPAD式の結果のデータ型
NCHAR	NCHAR または NCHAR VARYING	n: 文字数 INTEGERに変換。	NCHARVARYING(n*2)
NCHAR VARYING	NCHAR または NCHAR VARYING	n: 文字数 INTEGERに変換。	NCHARVARYING(n*2)

例

LPAD式

```
LPAD('ABCDE',10,'xyz') →結果は 'xyzxyABCDE'になります。
LPAD(N'ＡＢＣ',5,N'ｘｙｚ') →結果は N'ｘｙＡＢＣ'になります。
LPAD('ABC',5) →結果は '  ABC'になります。
LPAD('ABCDEFG',5,'xyz') →結果は 'ABCDE'になります。
LPAD(N'ＡＢＣ',5,'xyz') →結果は エラーになります。
    文字値式と埋込み文字とのデータ型の組合せが誤っています。
LPAD('ＡＢＣ',9,'Ｘ') →結果は '  ＸＡＢＣ'になります。
    Ｘの前に半角ブランクが設定されます。
```

RPAD式

― RPAD式は、文字値式の右側に、全体の文字列数が結果長になるまで、埋込み文字列をサイクリックに埋め込んで、結果長数分の文字列を返します。

― 埋込み文字列は文字値式です。

- 結果長は定数で指定します。整数でなければなりません。指定できる値は、文字値式のデータ型の最大長以下でなければなりません。文字値式のデータ型がCHAR、VARCHARの場合はバイト長で指定します。NCHAR、NCHAR VARYINGの場合は文字数で指定します。結果長は、INTEGER型に変換されます。
- 結果長に0または負数を指定するとエラーとなります。
- 埋込み文字列を省略すると空白を指定したものとみなされます。
- RPAD式に動的パラメタ指定が指定された場合は、エラーになります。
- 文字値式がNULLあるいは長さ0の文字列である場合は、RPAD式の結果はNULLとなります。
- 埋込み文字列がNULLあるいは長さ0の文字列である場合は、RPAD式の結果はNULLとなります。
- 文字値式の文字列数が結果長より大きい場合、文字値式の先頭から結果長数分の文字列を返却します。
- 文字値式のデータ型がCHAR、VARCHARのとき、1文字が2バイト以上で構成される文字を埋込み文字列に指定すると、結果長の指定によっては、文字を埋め込むスペースが足りなくなることがあります。そのような場合は、文字の代わりに半角ブランクが埋め込まれます。
- データ型がNCHAR、NCHAR VARYINGの文字値式や埋込み文字列に対してRPAD式を使用する場合、文字値式や埋込み文字列に含まれるUNICODEの補助文字(1～16面の4バイト文字)を2文字として扱います。動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、1文字として扱うことが可能となります。

参照

動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

- RPAD式の引数の組み合わせおよび結果のデータ型は次のようになります。

文字値式のデータ型	埋込み文字列のデータ型	結果長	RPAD式の結果のデータ型
CHAR	CHAR または VARCHAR	n: バイト数 INTEGERに変換	CHAR(n)
VARCHAR	CHAR または VARCHAR	n: バイト数 INTEGERに変換	VARCHAR(n)
NCHAR	NCHAR または NCHAR VARYING	n: 文字数 INTEGERに変換	NCHAR(n)
NCHAR VARYING	NCHAR または NCHAR VARYING	n: 文字数 INTEGERに変換	NCHAR VARYING(n)

- 動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”が“1”の場合、RPAD式の引数の組み合わせおよび結果のデータ型は次のようになります。

文字値式のデータ型	埋込み文字列のデータ型	結果長	RPAD式の結果のデータ型
NCHAR	NCHAR または NCHAR VARYING	n: 文字数 INTEGERに変換	NCHAR VARYING(n*2)

文字値式のデータ型	埋込み文字列のデータ型	結果長	RPAD式の結果のデータ型
NCHAR VARYING	NCHAR または NCHAR VARYING	n: 文字数 INTEGERに変換	NCHAR VARYING(n*2)

例

RPAD式

```
RPAD('ABCDE',10,'xyz') →結果は'ABCDExyzxy'になります。
RPAD(N'ＡＢＣ',5,N'ｘｙｚ') →結果はN'ＡＢＣｘｙ'になります。
RPAD('ABC',5) →結果は'ABC  'になります。
RPAD('ABCDEFG',5,'xyz') →結果は'ABCDE'になります。
RPAD(N'ＡＢＣ',-1,N'ｘ') →結果はエラーになります。
    結果長に負数が指定されています。
RPAD('ＡＢＣ',9,'Ｘ') →結果は'ＡＢＣＸ　'になります。
    Ｘの後ろに半角ブランクが設定されています。
```

REPLACE式

- REPLACE式は、REPLACE元の文字列中に含まれる検索文字列のすべてを置換文字列に置き換えた結果を返却します。
- REPLACE元、検索文字列、置換文字列は文字値式です。
- REPLACE式に動的パラメタ指定が指定された場合は、エラーになります。
- REPLACE元がNULLあるいは長さ0の文字列である場合は、REPLACE式の結果はNULLとなります。
- 検索文字列がNULLあるいは長さ0の文字列である場合は、REPLACE元の文字列をそのまま返却します。
- 置換文字列がNULLあるいは長さ0の文字列である場合、REPLACE元から検索文字列が取り除かれます。
- 結果長は定数で指定します。整数でなければなりません。また、指定できる値は、REPLACE元のデータ型の最大長以下でなければなりません。REPLACE元のデータ型がCHAR、VARCHARの場合はバイト長で指定します。NCHAR、NCHAR VARYINGの場合は文字数で指定します。結果長は、INTEGER型に変換されます。省略した場合は、REPLACE元の文字列長を指定したものとみなされます。
- 置き換えた結果の長さが、結果長を超えた場合は、エラーとなります。
- データ型がNCHAR、NCHAR VARYINGの文字値式や埋込み文字列に対してREPLACE式を使用する場合、文字値式や埋込み文字列に含まれるUNICODEの補助文字(1～16面の4バイト文字)を2文字として扱います。動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、1文字として扱うことが可能となります。

> **参照**
>
> 動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

- REPLACE式の引数の組み合わせおよび結果のデータ型は次のようになります。

REPLACE元のデータ型	検索文字列のデータ型	置換文字列のデータ型	結果長の指定	REPLACE式の結果のデータ型
CHAR	CHAR または VARCHAR	CHAR または VARCHAR	あり	CHAR(n) n: 結果長
			なし	CHAR(REPLACE元の文字列長)

REPLACE元のデータ型	検索文字列のデータ型	置換文字列のデータ型	結果長の指定	REPLACE式の結果のデータ型
VARCHAR	CHAR または VARCHAR	CHAR または VARCHAR	あり	VARCHAR(n) n: 結果長
			なし	VARCHAR(REPLACE元の文字列長)
NCHAR	NCHAR または NCHAR VARYING	NCHAR または NCHAR VARYING	あり	NCHAR(n) n: 結果長
			なし	NCHAR(REPLACE元の文字列長)
NCHAR VARYING	NCHAR または NCHAR VARYING	NCHAR または NCHAR VARYING	あり	NCHAR VARYING(n) n: 結果長
			なし	NCHAR VARYING(REPLACE元の文字列長)

― 動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”が“1”の場合、REPLACE式の引数の組み合わせおよび結果のデータ型は次のようになります。

REPLACE元のデータ型	検索文字列のデータ型	置換文字列のデータ型	結果長の指定	REPLACE式の結果のデータ型
NCHAR	NCHAR または NCHAR VARYING	NCHAR または NCHAR VARYING	あり	NCHAR VARYING(n*2) n: 結果長
			なし	NCHAR VARYING(REPLACE元の文字列長*2)
NCHAR VARYING	NCHAR または NCHAR VARYING	NCHAR または NCHAR VARYING	あり	NCHAR VARYING(n*2) n: 結果長
			なし	NCHAR VARYING(REPLACE元の文字列長*2)

例

REPLACE式

```
REPLACE('ABxyCDxyzE','xy','#',8) →結果は'AB#CD#zE'になります。
REPLACE(N'ＡＢｘｘＤ',N'ｘ',N'ＸＹＺ',10) →結果はN'ＡＢＸＹＺＸＹＺＤ'
                                          になります。
REPLACE('AxyxyBxyC','xy','',5) →結果は'ABC  'になります。
REPLACE('ＡＢＣｘ',N'ｘ','Ｄ',10) →結果はエラーになります。
        文字列のデータ型の組合せが誤っています。
REPLACE('ABCABCABC','A','ZZZ') →結果はエラーになります。
        置換え後の長さが元の長さを超えています。
```

REPLICATE式

― REPLICATE式は、指定された結果文字数の範囲内で、指定された回数だけ、文字値式を繰り返します。

― 繰り返し回数は数値式です。INTEGER型に変換されます。

― 繰り返し回数に負数を指定した場合、REPLICATE式の結果はNULLとなります。

― 結果長は定数で指定します。整数でなければなりません。文字値式のデータ型がCHAR、VARCHARの場合はバイト長で指定します。NCHAR、NCHAR VARYINGの場合は文字数で指定します。指定できる値は、文字値式の長さと繰り返し回数を掛け合わせた値か、それ以上の値を指定します。それより小さな値を指定した場合はエラーとなります。

― 結果長は、文字値式のデータ型の最大長以下でなければなりません。結果長は、INTEGER型に変換されます。

- 結果長を省略した場合は、文字値式の長さと繰り返し回数を掛け合わせた値を指定したものとみなされます。繰り返し回数を定数以外で与えた場合は、文字値式のデータ型の最大長となります。
- 結果長が、文字値式の長さに満たない場合はエラーとなります。
- 文字値式がNULLであるか、繰り返し回数がNULLである場合は、REPLICATE式の結果はNULLとなります。
- データ型がNCHAR、NCHAR VARYINGの文字値式に対してREPLICATE式を使用する場合、文字値式に含まれるUNICODEの補助文字(1～16面の4バイト文字)を2文字として扱います。動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、1文字として扱うことが可能となります。

> **参照**
>
> 動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

- REPLICATE式に動的パラメタ指定が指定された場合は、エラーになります。
- REPLICATE式の引数の組合せおよび結果のデータ型は次のようになります。

文字式のデータ型	繰り返し回数	結果長の指定	結果長	REPLICATE式の結果のデータ型
CHAR	n1: INTEGERに変換	あり	n2: INTEGERに変換	CHAR(n2)
		なし	－	CHAR(文字値式の長さ*n1) (注)
VARCHAR	n1: INTEGERに変換	あり	n2: INTEGERに変換	VARCHAR(n2)
		なし	－	VARCHAR(文字値式の長さ*n1) (注)
NCHAR	n1: INTEGERに変換	あり	n2: INTEGERに変換	NCHAR(n2)
		なし	－	NCHAR(文字値式の長さ*n1) (注)
NCHAR VARYING	n1: INTEGERに変換	あり	n2: INTEGERに変換	NCHAR VARYING(n2)
		なし	－	NCHAR VARYING(文字値式の長さ*n1) (注)

注) 繰り返し回数が定数でない場合は最大長になります。

- 動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”が“1”の場合、REPLICATE式の引数の組合せおよび結果のデータ型は次のようになります。

文字式のデータ型	繰り返し回数	結果長の指定	結果長	REPLICATE式の結果のデータ型
NCHAR	n1: INTEGERに変換	あり	n2: INTEGERに変換	NCHAR VARYING(n2*2)

文字式のデータ型	繰り返し回数	結果長の指定	結果長	REPLICATE式の結果のデータ型
		なし	－	NCHAR VARYING(文字値式の長さ*n1*2)(注)
NCHAR VARYING	n1: INTEGERに変換	あり	n2: INTEGERに変換	NCHAR VARYING(n2*2)
		なし	－	NCHAR VARYING(文字値式の長さ*n1*2)(注)

注) 繰り返し回数が定数でない場合は最大長になります。

例

REPLICATE式

```
REPLICATE('Xyz',2) →結果は'XyzXyz'になります。
REPLICATE(N'ＡＢＣ',3,10) →結果はN'ＡＢＣＡＢＣＡＢＣ　'になります。
REPLICATE('A A',3,7) →結果はエラーになります。
          結果長が、文字式の長さ×繰り返し回数より小さい値が指定されています。
REPLICATE(N'ｂ',5,1) →結果はエラーになります。
          結果長が、文字式の長さより小さい値が指定されています。
REPLICATE('ABC',20000,60000) →結果はエラーになります。
          結果長が、文字式の最大文字数より大きい値が指定されています。
REPLICATE('ZZZZ',-3,12) →結果はNULLになります。
```

REVERSE式

- REVERSE式は、文字値式を逆に並び替えたものを返します。
- 文字列内のデータには、各国語文字が混在していても構いません。文字として認識して処理を行います。
- REVERSE式の結果のデータ型は、文字値式のデータ型と同じです。
- 文字値式がNULLである場合は、REVERSE式の結果はNULLとなります。
- REVERSE式に動的パラメタ指定が指定された場合はエラーになります。
- データ型がNCHAR、NCHAR VARYINGの文字値式に対してREVERS式を使用する場合、文字値式にUNICODEの補助文字(1～16面の4バイト文字)が含まれるとエラーになります。動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、補助文字を使えるようになります。

参照

動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

例

REVERSE式

```
REVERSE(N'ＡＢＣ') →結果はN'ＣＢＡ'になります。
REVERSE('ABCDE  ') →結果は'  EDCBA'になります。
REVERSE('vWxＹz') →結果は'ｚＹxWv'になります。
REVERSE(TRIM(TRAILING FROM 'XYZ  ')) →結果は'ZYX'になります。
```

CNV_CHAR関数

- CNV_CHAR関数は、変換元の日付を日付書式に従って文字型に変換します。日付書式については、“日付書式”を参照してください。
- 書式要素の「H」および「Q」は「YYYJ」または「YYYY」のいずれかと組み合わせて使用する必要があります。「YYYJ」と「YYYY」の両方を同時に指定している場合、「H」または「Q」を指定するとエラーとなります。
- 変換元は日時値式です。DATE型に代入可能でなければなりません。
- 日付書式は値指定です。変換後の書式を表す文字列を指定します。
- 日付書式を省略すると、デフォルトの日付書式が選択されたとみなされます。デフォルトの日付書式は、'YYYY-MM-DD'です。
- 変換元がNULLである場合は、CNV_CHAR関数の結果はNULLとなります。
- CNV_CHAR関数の結果のデータ型はVARCHAR型です。
- 日時書式がNULLである場合は、エラーとなります。
- CNV_CHAR関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.32 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
変換元	DATE
日付書式	CHAR(10)

例

CNV_CHAR関数

```
CNV_CHAR(DATE'2007-04-14','DAY MONTH DD,YYYY')
→結果は'Saturday April    14,2007'になります。
CNV_CHAR(DATE'2007-04-14','DYMONDDYYYY')
→結果は'SATAPR142007'になります。
CNV_CHAR(DATE'2007-04-14','YYYYMMDD')
→結果は'20070414'になります。
CNV_CHAR(DATE'2007-03-14','YYYJ-MM-DD')
→結果は'2006-03-14'になります。
CNV_CHAR(DATE'2007-04-14') →結果は'2007-04-14'になります。
CNV_CHAR(DATE'2007-04-14','DY') →結果は'SAT'になります。
```

CHR関数

- CHR関数は、数値式を文字コードとして該当するASCII文字を返却します。
- CHARはCHRと同じ意味です。
- 数値式に指定できる数値は0～127の範囲です。数値式がNULL、または範囲外の値が指定された場合はNULLが返却されます。
- 数値式に指定できるデータ型はSMALLINT型、またはINTEGER型です。
- 数値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、INTEGER型になります。
- CHR関数の結果のデータ型はCHAR(1)です。

例

CHR関数

```
CHR(65) →結果は'A'になります。
CHAR(65) →結果は'A'になります。
```

CONCAT関数

- CONCAT関数は、連結元に連結文字列を結合して返却します。連結元および連結文字列は文字値式です。

- 連結元および連結文字列に指定できるデータ型は文字列型または各国語文字列型で、かつ比較可能であることが必要です。比較可能なデータ型は“表2.61 比較可能なデータ型”を参照してください。
- 連結元または連結文字列がNULLならば、結果もNULLになります。
- CONCAT関数の引数の組み合わせおよび結果のデータ型は次のようになります。

連結元のデータ型	連結文字列のデータ型	CONCAT関数の結果のデータ型
CHAR(n1)	CHAR(n2～nx)	CHAR(n1+n2…+nx) Nc<n1+n2…+nxのときはエラー
	VARCHAR(n2～nx)	VARCHAR(n) n: MIN(Nvc,n1+n2…+nx)
VARCHAR(n1)	CHAR(n2～nx)または VARCHAR(n2～nx)	VARCHAR(n) n: MIN(Nvc,n1+n2…+nx)
NCHAR(n1)	NCHAR(n2～nx)	NCHAR(n1+n2…+nx) Nn<n1+n2…+nxのときはエラー
	NCHAR VARYING(n2～nx)	NCHAR VARYING (n) n: MIN(Nvn,n1+n2…+nx)
NCHAR VARYING(n1)	NCHAR(n2～nx)または NCHAR VARYING(n2～nx)	NCHAR VARYING (n) n: MIN(Nvn,n1+n2…+nx)

Nc: CHARのデータの最大長

Nn: NCHARのデータの最大長

Nvc: VARCHARのデータの最大長

Nvn: NCHAR VARYINGのデータの最大長

- 結果がVARCHARまたはNCHAR VARYINGのときで、結合した長さが最大長を超えた場合は、最大長以降の文字は切り捨てられます。ただし、切り捨てられる文字が空白でない場合は、エラーになります。
- 引数に動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。なお、連結元と1番目の連結文字列の両方が動的パラメタ指定の場合は、エラーになります。SQL記述子域のTYPEにVARCHARやNCHAR VARYINGの情報が設定されます。SQL記述子域のLENGTHにVARCHARやNCHAR VARYINGの最大文字数が設定されます。

表2.33 動的パラメタ指定が指定された場合のDESCRIBE情報

連結元(または連結文字列)のデータ型	DESCRIBE情報
CHAR(n)	VARCHAR(Nvc)
VARCHAR(n)	VARCHAR(Nvc)
NCHAR(n)	NCHAR VARYING(Nvn)
NCHAR VARYING(n)	NCHAR VARYING(Nvn)

Nvc: VARCHARのデータの最大長

Nvn: NCHAR VARYINGのデータの最大長

例

CONCAT関数

```
CONCAT('ABCDE','xyz') →結果は'ABCDExyz'になります。
CONCAT(N'ＡＢＣＤＥ',N'ｘｙｚ') →結果はN'ＡＢＣＤＥｘｙｚ'になります。
CONCAT('ABCDE','xyz','ABCDE') →結果は'ABCDExyzABCDE'になります。
CONCAT('ABCDE', N'ｘｙｚ') →結果はエラーになります。
                文字列のデータ型の組合せが誤っています。
```

SPACE関数

- SPACE関数は、指定された結果長の空白文字列を返却します。
- 結果長は数値式です。INTEGER型に変換されます。
- 結果長は、CHARのデータの最大長以下でなければなりません。
- 結果長に負の値を指定した場合、または結果長がNULLの場合、結果はNULLとなります。
- 動的パラメタ指定が指定された場合は、エラーになります。
- SPACE関数の結果のデータ型はCHAR型です。

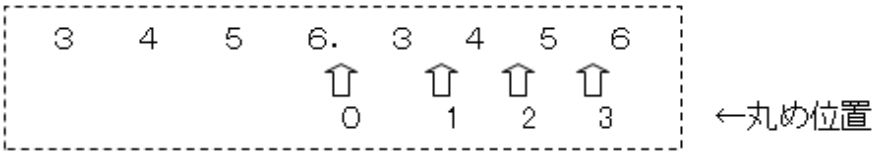
例

SPACE関数

```
SPACE(3) →結果は'   'になります。
```

STR関数

- STR関数は、指定された数値式のデータを文字型に変換します。
- 結果長は定数で指定します。整数でなければなりません。文字数で指定します。指定できる値はCHARのデータの最大長以下です。また、INTEGER型に変換されます。
- 結果長には小数点、符号、数字、空白文字を含みます。
- 変換結果の文字数が結果長に満たない場合は、変換結果の左側に全体の文字数が結果長になるまで、空白をサイクリックに埋め込んで、結果長数分の文字列を返します。
- 変換結果の文字数が結果長を超えた場合は、エラーとなります。
- 数値式が負の値の場合は、数値式の値を絶対値にしてSTR関数で求めた値の符号を反転させた結果と同一です。
- 丸め位置が指定された場合、数値式を丸め位置で四捨五入します。
- 丸め位置は数値式です。丸め位置には、小数点位置を起点とした有効桁(丸め位置)を指定します。また、0から18の範囲の整数値に変換できる値を指定します。丸め位置は、INTEGER型に変換されます。



- 結果長が省略された場合、10が指定されたものとみなします。
- 丸め位置が省略された場合、0が指定されたものとみなします。
- 丸め位置が負の値ならば、エラーになります。
- 数値式がNULLならば、結果もNULLになります。
- STR関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。SQL記述子域のTYPEにINTEGERの情報が設定されます。SQL記述子域のLENGTHにINTEGERのデータ長が設定されます。

表2.34 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
数値式	INTEGER

オペランド	DESCRIBE情報
結果長	エラー
丸め位置	エラー

- STR関数の結果のデータ型はVARCHAR型です。

例

STR関数

```
STR(12345.678) →結果は'      12346' になります。
STR(12345.678,7) →結果は'   12346' になります。
STR(12345.678,7,1) →結果は'12345.7' になります。
STR(-12345.678,8,1) →結果は'-12345.7' になります。
```

STUFF関数

- STUFF関数は、文字値式1に対して開始位置から文字列長分の文字を削除し、文字値式2を挿入して、部分的に入れ替えた文字列を返却します。
- 開始位置と文字列長は数値式で指定します。整数でなければなりません。開始位置と文字列長はINTEGER型に変換されます。
- 開始位置と文字列長はそれぞれ文字数を指定します。
- 開始位置が文字値式1の長さより大きければ、STUFF関数の結果は文字値式1と文字値式2を連結した文字列になります。
- 開始位置が0または負の値ならば、エラーになります。
- 文字列長に負の値を指定した場合、文字値式1の先頭から開始位置までの部分列に文字値式2を連結し、その後ろに文字値式1の開始位置から文字列長分戻った位置から最後の文字までの部分列を連結した文字列になります。
- データ型がNCHAR、NCHAR VARYINGの文字値式1に対してSTUFF関数を使用する場合、文字値式1に含まれるUNICODEの補助文字(1～16面の4バイト文字)を2文字として扱います。動作環境ファイルの実行パラメタ“SURROGATE_PAIR_NUMBER”に“1”を指定することで、1文字として扱うことが可能となります。

> **参照**
>
> 動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

- STUFF関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。SQL記述子域のTYPEにINTEGERの情報が設定されます。SQL記述子域のLENGTHにINTEGERのデータ長が設定されます。

表2.35 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
文字値式1	エラー
数値式	INTEGER
文字列長	INTEGER
文字値式2	エラー

- 文字値式1のデータ型と、文字値式2のデータ型は文字列型または各国語文字列型で、かつ比較可能であることが必要です。比較可能なデータ型は“表2.61 比較可能なデータ型”を参照してください。
- 文字値式1、開始位置、文字列長または文字値式2がNULLならば、結果もNULLになります。STUFF関数の結果のデータ型を以下に示します。

文字値式1のデータ型	文字値式2のデータ型	STUFF関数の結果のデータ型
CHAR(n1)または VARCHAR(n1)	CHAR(n1) または VARCHAR(n1)	VARCHAR(MIN(Nvc,n1+n1+n2))
NCHAR(n1)または NCHAR VARYING(n1)	NCHAR(n2)または NCHAR VARYING(n2)	NCHAR VARYING(MIN(Nvn,n1+n1+n2))

Nvc: VARCHARのデータの最大長

Nvn: NCHAR VARYINGのデータの最大長

例

STUFF関数

```
STUFF('abcdef',2,3,'ijklmn') →結果は'aijklmnef'になります。
```

使用例

例1

文字部分列関数(N1が'CDプレーヤー'の場合。なお、N1のデータ型は各国語文字列型とします。)

```
SELECT SUBSTRING(N1 FROM 3 FOR 5) INTO :SUB1 FROM S1.T1
→結果は'プレーヤー'になります。
```

例2

文字部分列関数(C1が'CDラジカセ'の場合。なお、C1のデータ型は文字列型とします。CDは1バイトの英数字です。)

```
SELECT SUBSTRING(C1 FROM 3 FOR 4) INTO :SUB1 FROM S1.T1
→結果は'ラジカセ'になります。
```

例3

大文字小文字変換(C1が'aBcDe'の場合)

```
SELECT UPPER(C1) INTO :UPP1 FROM S1.T1
→結果は'ABCDE'になります。
```

例4

TRIM関数(N1が'アダム電気□□□'の場合。なお、N1のデータ型は各国語文字列型とします。)

なお、□は空白を示します。

```
SELECT TRIM(BOTH N'□' FROM N1) INTO :MOJI1 FROM S1.T1
→結果は'アダム電気'になります。
```

例5

TRIM関数(C1が'**CDラジカセ****'の場合。なお、C1のデータ型は文字列型とします。CDは1バイトの英数字です。)

```
SELECT TRIM(BOTH '*' FROM C1) INTO :MOJI1 FROM S1.T1
→結果は'CDラジカセ'になります。
```

例6

LTRIM/RTRIM関数(C1が' CDラジカセ 'の場合。なお、C1のデータ型は文字列型とします。CDは1バイトの英数字です。)

```
SELECT LTRIM(C1) INTO :MOJI1 FROM S1.T1
→結果は'CDラジカセ    'になります。
SELECT RTRIM(C1) INTO :MOJI1 FROM S1.T1
→結果は'  CDラジカセ'になります。
```

例7

LPAD式(支店名“札幌”の左側に“*”を埋め込みます。全体の長さは、5文字とします。)

```
SELECT 支店コード, LPAD(支店名, 5, N'＊') FROM 支店コード表
→結果は'＊＊＊札幌'になります。
```

例8

RPAD式(支店名“札幌”の右側に、“*”を埋め込みます。全体の長さは、5文字とします。)

```
SELECT 支店コード, RPAD(支店名, 5, N'＊') FROM 支店コード表
→結果は'札幌＊＊＊'になります。
```

例9

REPLACE式(電話番号“011-999-9999”の文字列中の“-”(ハイフン)を“ ”(半角ブランク)に置き換えます。)

```
SELECT 支店コード, REPLACE(電話番号, '-', ' ') FROM 支店コード表
→結果は'011 999 9999'になります。
```

例10

REPLICATE式(支店名“札幌”を2回繰り返します。ただし、支店名の全角ブランクは、繰り返し対象としません。)

```
SELECT 支店コード, REPLICATE(TRIM(TRAILING N'□' FROM 支店名), 2)
    FROM 支店コード表
→結果は'札幌札幌'になります。
```

例11

REVERSE式(支店名“札幌”を逆に並び替えます。)

```
SELECT 支店コード, REVERSE(支店名) FROM 支店コード表
→結果は'幌札'になります。
```

例12

CNV_CHAR関数(気象情報管理表の最高気温日付“2007-08-14”をYYYY.MM.DD DYの形式で出力します。)

```
SELECT 地区コード , CNV_CHAR(最高気温日付, 'YYYY.MM.DD DY')
    FROM 気象情報管理表
→結果は'2007.08.14 TUE'になります。
```

例13

CHR関数('A'で始まる名前の従業員の数を求めます。)

```
SELECT COUNT(*) FROM 従業員表 WHERE SUBSTRING(名前 FROM 1 FOR 1) = CHR(65)
→'A'で始まる名前が5件の場合、結果は5になります。
```

例14

CONCAT関数(仕入価格“20000”の先頭に¥を結合させて出力します。)

```
SELECT 取引製品,CONCAT('¥',CAST(仕入価格 AS CHARACTER(5)))
    FROM 在庫管理.発注表 WHERE 取引先 = 61
→結果は'¥20000'になります。
```

例15

SPACE関数(支店コード0001と支店名”札幌”の間にスペース2つ挟んで連結します。)

```
SELECT 支店コード|| SPACE(2) || 支店名 FROM 支店コード表
→結果は'0001　札幌'になります
```

例16

STR関数(最高気温が27.255を小数第2位で四捨五入します。)

```
SELECT地区名, STR(最高気温,4,1) FROM 気象情報管理表
→結果は、最高気温が27.3になります。
```

例17

STUFF関数(C3に格納された商品名'XX-DVDレコーダー'を'XX-Blu-rayレコーダー'と出力します。)

```
SELECT STUFF(C3,4,3,'Blu-ray') FROM 商品表
→結果は' XX-Blu-rayレコーダー'になります。
```

2.5.4 日時値関数

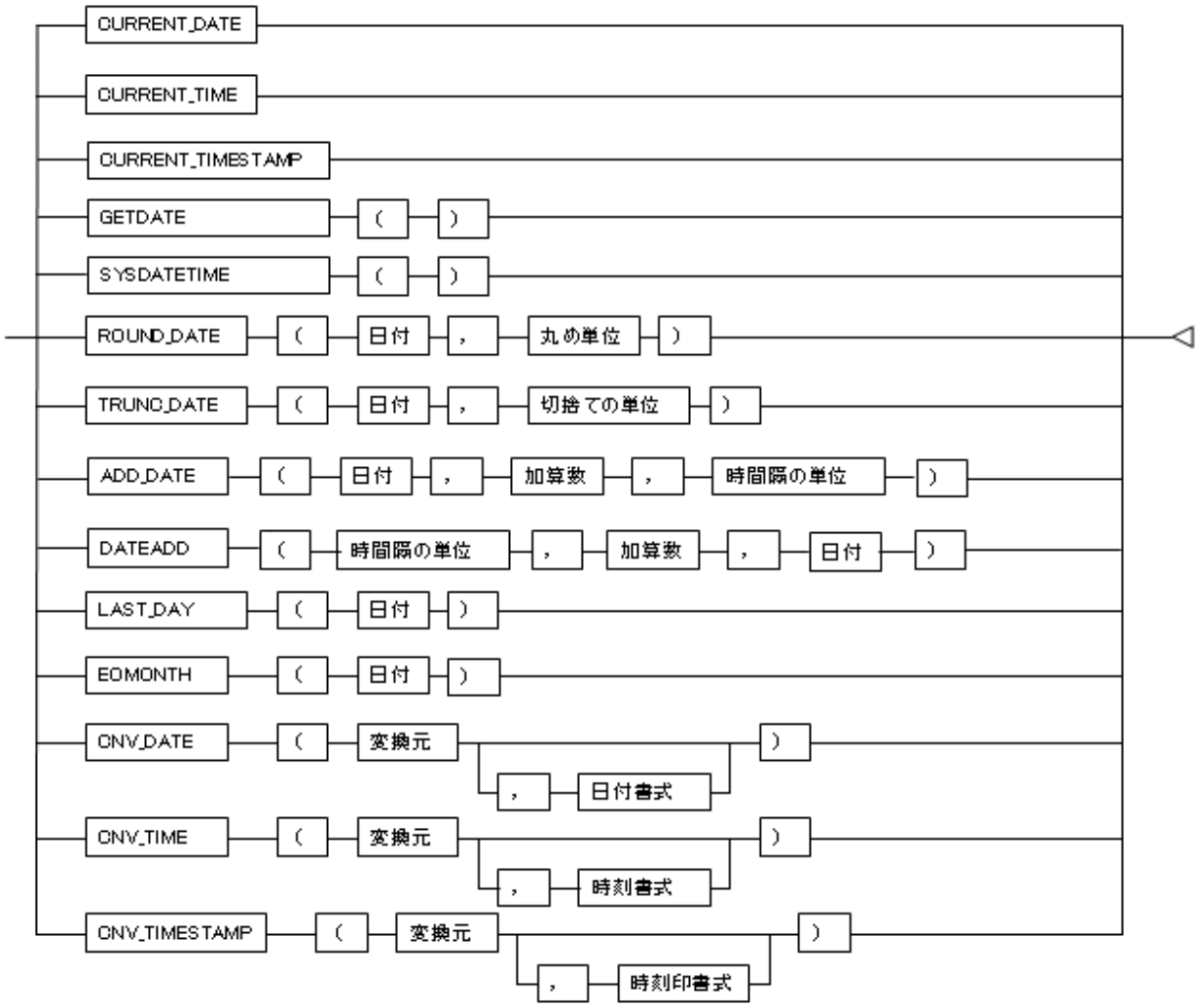
機能

日時値関数は、日時型の値となる関数です。

関数	機能
CURRENT DATE値関数	現在の日付を返却します。
CURRENT TIME値関数	現在の時刻を返却します。
CURRENT TIMESTAMP値関数	現在の時刻印を返却します。
ROUND_DATE関数	日付を指定した日時フィールドで丸めます。
TRUNC_DATE関数	日付を指定した日時フィールドで切捨てます。
ADD_DATE関数	DATE型に、数値属性で表現した時間隔を加算した結果の日時を返却します。
LAST_DAY関数	当該月の最終日付を返却します。
CNV_DATE関数	指定された日付書式で記述された文字型データをDATE型に変換します。
CNV_TIME関数	指定された時刻書式で記述された文字型データをTIME型に変換します。

関数	機能
CNV_TIMESTAMP関数	指定された時刻印書式で記述された文字型データを TIMESTAMP型に変換します。

記述形式



一般規則

CURRENT DATE値関数

- CURRENT_DATEは、現在の日付を返します。
- CURRENT DATE値関数のデータ型は、DATE型です。

例

CURRENT_DATE

```
CURRENT_DATE
→結果はDATE'2007-04-10'になります。
  現在日付が西暦2007年4月10日の場合。
```

CURRENT TIME値関数

- CURRENT_TIMEは、現在の時刻を返します。

－ CURRENT TIME値関数のデータ型は、TIME型です。

例

CURRENT_TIME

```
CURRENT_TIME
→結果はTIME'15:58:57'になります。
  現在時刻が15時58分57秒の場合。
```

CURRENT TIMESTAMP値関数

－ CURRENT_TIMESTAMPは、現在の時刻印を返します。

－ GETDATEおよびSYSDATETIMEは、CURRENT_TIMESTAMPと同じ意味です。

－ CURRENT TIMESTAMP値関数のデータ型は、TIMESTAMP型です。

注意

CURRENT_TIMESTAMP値関数を使用する環境がWindowsシステムの場合、以下の可能性があるため、システム時刻の変更タイミングに留意してください。

－ 同一の現在時刻が複数回存在する

－ 未来の値が存在する

例

CURRENT_TIMESTAMP

```
CURRENT_TIMESTAMP
→結果はTIMESTAMP'2007-04-10 15:58:59'になります。
  現在日付が西暦2007年4月10日でかつ現在時刻が15時
  58分59秒の場合。
GETDATE()
→結果はTIMESTAMP'2007-04-10 15:58:59'になります。
  現在日付が西暦2007年4月10日でかつ現在時刻が15時
  58分59秒の場合。
SYSDATETIME()
→結果はTIMESTAMP'2007-04-10 15:58:59'になります。
  現在日付が西暦2007年4月10日でかつ現在時刻が15時
  58分59秒の場合。
```

ROUND_DATE関数

－ ROUND_DATE関数は、日付を丸め単位で丸めます。

－ 日付は日時値式です。データ型はDATE型に代入可能でなければなりません。

－ 日付がNULLである場合は、ROUND_DATE関数の結果はNULLとなります。

－ ROUND_DATE関数の結果のデータ型はDATE型です。

－ 丸め単位は文字列型の値指定です。丸めを行う単位を指定します。丸め単位に指定できる値と、ROUND_DATE関数の結果は以下のようになります。

丸め単位	意味	引数の日付	結果
YEAR	年で丸め	1月1日 ～ 6月30日 7月1日 ～ 12月31日	当年の1月1日 翌年の1月1日
JYEAR	年度で丸め	4月1日 ～ 9月30日 10月1日 ～ 3月31日	当年度の4月1日 翌年度の4月1日

丸め単位	意味	引数の日付	結果
MONTH	月で丸め	1日 ～ 15日 16日 ～ 31日	当月の1日 翌月の1日
HALF	半期で丸め	1月1日 ～ 3月31日 4月1日 ～ 9月30日 10月1日 ～ 12月31日	当年の1月1日 当年の7月1日 翌年の1月1日
JHALF	年度の半期で丸め	4月1日 ～ 6月30日 7月1日 ～ 12月31日 翌年1月1日 ～ 3月31日	当年の4月1日 当年の10月1日 翌年の4月1日
QUARTER	四半期で丸め	1月1日 ～ 2月15日 2月16日 ～ 5月15日 5月16日 ～ 8月15日 8月16日 ～ 11月15日 11月16日 ～ 12月31日	当年の1月1日 当年の4月1日 当年の7月1日 当年の10月1日 翌年の1月1日

－ ROUND_DATE関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.36 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
日付	DATE
丸め単位	CHAR(7)

例

ROUND_DATE関数

```
ROUND_DATE(DATE'2007-08-15','MONTH') →結果はDATE'2007-08-01'になります。
ROUND_DATE(DATE'2007-08-23','QUARTER') →結果はDATE'2007-10-01'になります。
```

TRUNC_DATE関数

－ TRUNC_DATE関数は、日付を切捨ての単位で切り捨てます。

－ 日付は日時値式です。データ型はDATE型に代入可能でなければなりません。

－ 日付がNULLである場合は、TRUNC_DATE関数の結果はNULLとなります。

－ TRUNC_DATE関数の結果のデータ型はDATE型です。

－ 切捨ての単位は文字列型の値指定です。切捨てを行う単位を指定します。切捨ての単位に指定できる値と、TRUNC_DATE関数の結果は以下のようになります。

切捨ての単位	意味	引数の日付	結果
YEAR	年で切捨て	1月1日 ～ 12月31日	当年の1月1日
JYEAR	年度で切捨て	4月1日 ～ 3月31日	当年度の4月1日
MONTH	月で切捨て	1日 ～ 31日	当月の1日
HALF	半期で切捨て	1月1日 ～ 6月30日 7月1日 ～ 12月31日	当年の1月1日 当年の7月1日
JHALF	年度の半期で切捨て	4月1日 ～ 9月30日 10月1日 ～ 12月31日 翌年1月1日 ～3月31日	当年の4月1日 当年の10月1日 当年の10月1日

切捨ての単位	意味	引数の日付	結果
QUARTER	四半期で切捨て	1月1日 ～ 3月31日 4月1日 ～ 6月30日 7月1日 ～ 9月30日 10月1日 ～ 12月31日	当年の1月1日 当年の4月1日 当年の7月1日 当年の10月1日

- TRUNC_DATE関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.37 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
日付	DATE
切捨ての単位	CHAR(7)

例

TRUNC_DATE関数

```
TRUNC_DATE(DATE'2007-08-22','YEAR') →結果はDATE'2007-01-01'になります。
TRUNC_DATE(DATE'2007-02-15','JHALF') →結果はDATE'2006-10-01'になります。
```

ADD_DATE関数

- ADD_DATE関数は、日付に指定した加算数を指定した時間隔の単位として日付に加算した結果の日時を返却します。
- DATEADDは、ADD_DATEと同じ意味です。
- 日付のデータ型はDATE型に代入可能でなければなりません。
- 加算数は数値式です。INTEGER型に変換されます。
- ADD_DATE関数の結果のデータ型は、DATE型です。
- 日付または加算数がNULL値である場合、ADD_DATE関数の結果はNULLとなります。
- 加算数が負の値の場合、日付から加算数の絶対値を時間隔の単位として減算します。減算により、1年より小さくなった場合はエラーとなります。
- 月数の加算により、暦上存在しない日付になった場合は、日フィールドの値をその月の最終日に補正します。
- 時間隔の単位は文字列型の値指定です。指定できる文字列と意味を以下に示します。

時間隔の単位	意味
YEAR	加算数を年数として加算します。
MONTH	加算数を月数として加算します。
DAY	加算数を日数として加算します。

- ADD_DATE関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.38 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
日付	DATE
加算数	INTEGER
時間隔の単位	CHAR(5)

例

ADD_DATE関数

```
ADD_DATE(DATE'2006-08-28',6,'MONTH') →結果はDATE'2007-02-28'になります。
ADD_DATE(DATE'2007-08-31',1,'MONTH') →結果はDATE'2007-09-30'になります。
```

```
ADD_DATE(DATE'2007-08-28',-2,'DAY') →結果はDATE'2007-08-26'になります。
DATEADD(MONTH,6,DATE'2006-08-28') →結果はDATE'2007-02-28'になります。
```

LAST_DAY関数

- LAST_DAY関数は、日付で指定した月の最終日付を返します。
- EOMONTHは、LAST_DAYと同じ意味です。
- 日付は日時値式です。DATE型に代入可能でなければなりません。
- LAST_DAY関数の結果のデータ型はDATE型です。
- 日付がNULLの場合、LAST_DAY関数の実行結果はNULLとなります。
- LAST_DAY関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.39 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
日付	DATE

例

LAST_DAY関数

```
LAST_DAY(DATE'2007-06-01') →結果はDATE'2007-06-30'になります。
EOMONTH(DATE'2007-06-01') →結果はDATE'2007-06-30'になります。
```

CNV_DATE関数

- CNV_DATE関数は、変換元の文字値式を日付書式に従ってDATE型に変換します。
- 変換元は文字値式です。データ型は文字列型で、その書式は日付書式と一致していなければなりません。月や日など、先行する0は省略できません。
- 日付書式は値指定です。変換元の書式を表す文字列を指定します。
- 日付書式を省略すると、デフォルトの日付書式が選択されたと見なします。デフォルトの日付書式は、“YYYY-MM-DD”です。
- 変換元がNULLである場合は、CNV_DATE関数の結果はNULLとなります。
- 日付書式がNULLである場合は、CNV_DATE関数の結果はNULLとなります。
- CNV_DATE関数の結果のデータ型はDATE型です。
- 日付書式に月や日を表す文字列を指定しなかった場合、CNV_DATE関数は書式要素の組合せに応じて、ある期間の最終日を返します。詳細は“CNV_DATE関数に指定可能な書式要素の組合せ”の備考を参照してください。
- 日付書式には、書式要素の組合せを任意の順序で指定できます。また書式要素は任意のセパレータで区切って指定することができます。ただし、同じ書式要素を2度以上指定した場合やセパレータを2文字以上連続して記述することはできません。詳細は、“日付書式”、“CNV_DATE関数に指定可能な書式要素の組合せ”、“セパレータ”を参照してください。
- CNV_DATE関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.40 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
変換元	CHAR(10)
日付書式	CHAR(10)

例

CNV_DATE関数

```
CNV_DATE('Saturday APR 14,2007','DAY MON DD,YYYY')
→結果はDATE'2007-04-14'になります。
```

```
CNV_DATE('20070414','YYYYMMDD')
→結果はDATE'2007-04-14'になります。
CNV_DATE('2007-03-10','YYYJ-MM-DD')
→結果はDATE'2008-03-10'になります。
CNV_DATE('2007-04-14')
→結果はDATE'2007-04-14' になります。
CNV_DATE('2007','YYYY')
→結果はDATE'2007-12-31' になります。
CNV_DATE('Mondey 2007.04.23','DAY YYYY.MM.DD')
→結果はエラーになります。正しくはMondayです。
CNV_DATE('MONJuly232007','DYMONTHDDYYYY')
→結果はDATE'2007-07-23'になります。
```

日付書式

DATE型は通常、「YYYY-MM-DD」のシステム標準書式で扱います。しかし、システム標準書式以外の書式でDATE型を扱いたいような場合があります。このような場合に「日付書式」を利用します。

日付書式は、日付を表現する際の文字列の書式です。この日付書式は、DATE型と文字型との間の変換関数の入力パラメタとして指定します。

書式要素を以下に示します。

書式要素	意味
YYYY	年(西暦: 1～9999) 年の開始日は1月1日 データの表現は4文字固定(0001～9999)
YYYJ	年度(西暦: 1～9999) 年度の開始日は4月1日 データの表現は4文字固定(0001～9999)
MM	月(1～12) データの表現は2文字固定(01～12)
MONTH	月の名称 (January,February,March,April,May,June,July,August,September, October,November,December) データの表現は9文字固定。不足分には空白を埋める
MON	省略形の月の名前 (JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DE C) データの表現は3文字固定
DD	日(1～31) データの表現は2文字固定(01～31)
DAY	曜日 (Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday) データの表現は9文字固定。不足分には空白を埋める
DY	省略形の曜日(SUN,MON,TUE,WED,THU,FRI,SAT) データの表現は3文字固定
H	半期(1～2)データの表現は1文字固定 「YYYJ」と共に記述した場合 1: 4月1日 ～ 9月30日(上期) 2: 10月1日 ～ 3月31日(下期) 「YYYY」と共に記述した場合 1: 1月1日 ～ 6月30日(上期) 2: 7月1日 ～ 12月31日(下期)
Q	四半期(1～4) データの表現は1文字固定 「YYYJ」と共に記述した場合 1: 4月1日 ～ 6月30日 (第1四半期)

書式要素	意味
	2: 7月1日 ～ 9月30日 (第2四半期) 3: 10月1日 ～ 12月31日 (第3四半期) 4: 1月1日 ～ 3月31日 (第4四半期) 「YYYY」と共に記述した場合 1: 1月1日 ～ 3月31日 (第1四半期) 2: 4月1日 ～ 6月30日 (第2四半期) 3: 7月1日 ～ 9月30日 (第3四半期) 4: 10月1日 ～ 12月31日 (第4四半期)

CNV_DATE関数に指定可能な書式要素の組合せ

CNV_DATE関数には、指定可能な書式要素の組合せがあります。

指定可能な書式要素の組合せを以下に示します。

書式要素の組合せ	備考
YYYY, MM , DD ,[DAYまたはDY]	DAYまたはDYは省略可能です。
YYYJ, MM , DD ,[DAYまたはDY]	DAYまたはDYは省略可能です。
YYYY, MONTH, DD ,[DAYまたはDY]	DAYまたはDYは省略可能です。
YYYJ, MONTH, DD ,[DAYまたはDY]	DAYまたはDYは省略可能です。
YYYY, MON, DD ,[DAYまたはDY]	DAYまたはDYは省略可能です。
YYYJ, MON, DD ,[DAYまたはDY]	DAYまたはDYは省略可能です。
YYYY, MM	その月の最終日を返します。
YYYJ, MM	その年度月の最終日を返します。
YYYY, MONTH	その月の最終日を返します。
YYYJ, MONTH	その年度月の最終日を返します。
YYYY, MON	その月の最終日を返します。
YYYJ, MON	その年度月の最終日を返します。
YYYY, H	半期の最終日を返します。(6月30日か12月31日)
YYYJ, H	年度半期の最終日を返します。(3月31日か9月30日)
YYYY, Q	各四半期の最終日を返します。
YYYJ, Q	各四半期(年度)の最終日を返します。
YYYY	当年の最終日(12月31日)を返します。
YYYJ	当年度の最終日(3月31日)を返します。

セパレータ

セパレータには、以下の文字が指定できます。

セパレータ	文字
半角ブランク	
ハイフン	－
スラッシュ	／
ピリオド	.
カンマ	,
コロン	:

注意事項

日付書式には、上記の書式要素を任意の順序で指定することができます。また書式要素を任意のセパレータで区切って指定することもできます。ただし、以下のような指定をした場合、エラーとなります。

- 同じ書式要素を2度以上指定する。

```
CNV_DATE('2007-04-10','YYYY.MM.MM') →エラー
```

- セパレータを2文字以上連続して記述する。

```
CNV_DATE('2007-04-10','YYYY//MM///DD') →エラー
```

- セパレータを日付書式の先頭、末尾に記述する。

```
CNV_DATE('2007-04-10','YYYY/MM/DD/') →エラー
```

- 書式要素にH、またはQを指定した場合、YYYYとYYYJを同時に指定することはできません。どちらか一方を指定してください。

```
CNV_DATE('2007-04-10','YYYY-YYYJ-H-Q') →エラー
```

- CNV_DATE関数に、書式要素MONTH、MON、DAYおよびDYを指定する場合、変換元の文字列は、以下の文字列と完全に一致していなければいけません。大文字、小文字の違いでもエラーとなりますので注意が必要です。

```
MONTH:January,February,March,April,May,June,July,August,September,October,November, December
MON:JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
DAY:Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
DY:SUN,MON,TUE,WED,THU,FRI,SAT
```

- 書式要素DAYおよびDYを指定した場合、指定した日付と曜日が正しいかどうかの妥当性チェックは行いません。誤っていても曜日は無視して処理されます。

CNV_TIME関数

- CNV_TIME関数は、変換元の文字値式を時刻書式に従ってTIME型に変換します。
- 変換元は文字値式です。データ型は文字列型で、その書式は時刻書式と一致していなければなりません。
- 時刻書式は値指定です。変換元の書式を表す文字列を指定します。
- 時刻書式を省略すると、デフォルトの時刻書式が選択されたと見なします。デフォルトの時刻書式は、“hh24:mm:ss”です。
- 変換元がNULLである場合は、CNV_TIME関数の結果はNULLとなります。
- 時刻書式がNULLである場合は、CNV_TIME関数の結果はNULLとなります。
- CNV_TIME関数の結果のデータ型はTIME型です。
- 時刻書式に秒を表す文字列を指定しなかった場合、CNV_TIME関数は秒の値として0秒を返します。時刻書式に分と秒を表す文字列を指定しなかった場合、CNV_TIME関数は分と秒の値として0分0秒を返します。詳細は“CNV_TIME関数に指定可能な書式要素の組合せ”の備考を参照してください。

- 時刻書式には、書式要素の組合せを任意の順序で指定できます。また書式要素は任意のセパレータで区切って指定することができます。ただし、同じ書式要素を2度以上指定した場合やセパレータを2文字以上連続して記述することはできません。詳細は、“時刻書式”、“CNV_TIME関数に指定可能な書式要素の組合せ”、“セパレータ”を参照してください。
- CNV_TIME関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.41 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
変換元	CHAR(8)
時刻書式	CHAR(10)

例

CNV_TIME関数

```
CNV_TIME('141220','hh24mmss')
→結果はTIME'14:12:20'になります。
CNV_TIME('02-12-20-PM','hh-mm-ss-PM')
→結果は'14:12:20'になります。
CNV_TIME('14:12:20')
→結果はTIME'14:12:20'になります。
CNV_TIME('14','hh24')
→結果はTIME'14:00:00'になります。
```

時刻書式

TIME型は通常、「hh:mm:ss」のシステム標準書式で扱います。しかし、システム標準書式以外の書式でTIME型を扱いたいような場合があります。このような場合に「時刻書式」を利用します。

時刻書式は、時刻を表現する際の文字列の書式です。この時刻書式は、TIME型と文字型との間の変換関数の入力パラメタとして指定します。

書式要素を以下に示します。

書式要素	意味
hh	時間(1～12) 時間の開始は12
hh24	時間(0～23) 時間の開始は0
mm	分(0～59) 分の開始は0
ss	秒(0～59) 秒の開始は0
AMまたはPM	12時間表記の文字列を24時間の表記の時刻に変換します。

CNV_TIME関数に指定可能な書式要素の組合せ

CNV_TIME関数には、指定可能な書式要素の組合せがあります。

CNV_TIME関数に指定可能な書式要素の組合せを以下に示します。

書式要素の組合せ	備考
hh, mm, ss, AMまたはPM	12時間表記での文字列を24時間表記での時刻に変換します。
hh24,mm,ss	
hh, mm, AMまたはPM	開始秒(0秒)を返します。 12時間表記での文字列を24時間表記での時刻に変換します。
hh, AMまたはPM	開始分秒(0分0秒)を返します。

書式要素の組合せ	備考
	12時間表記での文字列を24時間表記での時刻に変換します。
hh24	開始分秒(0分0秒)を返します。

セパレータ

セパレータに指定可能な文字はCNV_DATE関数と共通です。詳細はCNV_DATE関数の“セパレータ”を参照してください。

注意事項

時刻書式には、上記の書式要素を任意の順序で指定することができます。また書式要素を任意のセパレータで区切って指定することもできます。ただし、以下のような指定をした場合、エラーとなります。

- 同じ書式要素を2度以上指定する。

```
CNV_TIME('14:12:20','hh24:mm:mm') →エラー
```

- セパレータを2文字以上連続して記述する。

```
CNV_TIME('14:12:20','hh24::mm::ss') →エラー
```

- セパレータを時刻書式の先頭、末尾に記述する。

```
CNV_TIME('14:12:20','hh24:mm:ss:') →エラー
```

- 書式要素にhhを指定した場合、AMとPMを同時に指定することはできません。どちらか一方を指定してください。

```
CNV_TIME('14:12:20','hh:mm:ss:AM:PM') →エラー
```

CNV_TIMESTAMP関数

- CNV_TIMESTAMP関数は、変換元の文字値式を時刻印書式に従ってTIMESTAMP型に変換します。
- 変換元は文字値式です。データ型は文字列型で、その書式は時刻印書式と一致していなければなりません。
- 時刻印書式は値指定です。変換元の書式を表す文字列を指定します。
- 時刻印書式を省略すると、デフォルトの時刻印書式が選択されたと見なします。デフォルトの時刻印書式は、“YYYY-MM-DD hh24:mm:ss”です。
- 変換元がNULLである場合は、CNV_TIMESTAMP関数の結果はNULLとなります。
- 時刻印書式がNULLである場合は、CNV_TIMESTAMP関数の結果はNULLとなります。
- CNV_TIMESTAMP関数の結果のデータ型はTIMESTAMP型です。
- 時刻印書式の一部の要素を省略した場合、CNV_TIMESTAMP関数は書式要素の組合せに応じて一定の値を返します。詳細は“CNV_DATE関数に指定可能な書式要素の組合せ”と“CNV_TIME関数に指定可能な書式要素の組合せ”の備考を参照してください。
- 時刻印書式には、書式要素の組合せを任意の順序で指定できます。また書式要素は任意のセパレータで区切って指定することができます。ただし、同じ書式要素を2度以上指定した場合やセパレータを2文字以上連続して記述することはできません。詳細は、“時刻印書式”、“CNV_TIMESTAMP関数に指定可能な書式要素の組合せ”、“セパレータ”を参照してください。
- CNV_TIMESTAMP関数の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.42 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
変換元	CHAR(19)
時刻印書式	CHAR(21)

例

CNV_TIMESTAMP関数

```
CNV_TIMESTAMP('20070414 141220','YYYYMMDD hh24mmss')
→結果はTIMESTAMP'2007-04-14 14:12:20'になります。
CNV_TIMESTAMP('2007-03-10 031450PM','YYYJ-MM-DD hhmmssPM')
→結果はTIMESTAMP'2008-03-10 15:14:50'になります。
CNV_TIMESTAMP('2007-04-14 00:00:00')
→結果はTIMESTAMP'2007-04-14 00:00:00'になります。
CNV_TIMESTAMP('2007','YYYY')
→結果はTIMESTAMP'2007-12-31 00:00:00'になります。
CNV_TIMESTAMP('Mondey 2007.04.23','DAY YYYY.MM.DD')
→結果はエラーになります。正しくはMondayです。
CNV_TIMESTAMP('MONJuly23200715','DYMONTHDDYYYYhh24')
→結果はTIMESTAMP'2007-07-23 15:00:00'になります。
```

時刻印書式

TIMESTAMP型は通常、「YYYY-MM-DD hh:mm:ss」のシステム標準書式で扱います。しかし、システム標準書式以外の書式でTIMESTAMP型を扱いたいような場合があります。このような場合に「時刻印書式」を利用します。

時刻印書式は、日付と時刻を表現する際の文字列の書式です。この時刻印書式は、TIMESTAMP型と文字型との間の変換関数の入力パラメタとして指定します。

書式要素を以下に示します。

書式要素	意味
YYYY	年(西暦: 1～9999) 年の開始日は1月1日 データの表現は4文字固定(0001～9999)
YYYJ	年度(西暦: 1～9999) 年度の開始日は4月1日 データの表現は4文字固定(0001～9999)
MM	月(1～12) データの表現は2文字固定(01～12)
MONTH	月の名称 (January,February,March,April,May,June,July,August,September,October,November,December) データの表現は9文字固定。不足分には空白を埋める
MON	省略形の月の名前 (JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC) データの表現は3文字固定
DD	日(1～31) データの表現は2文字固定(01～31)
DAY	曜日 (Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday) データの表現は9文字固定。不足分には空白を埋める
DY	省略形の曜日(SUN,MON,TUE,WED,THU,FRI,SAT) データの表現は3文字固定
H	半期(1～2)データの表現は1文字固定 「YYYJ」と共に記述した場合 1: 4月1日 ～ 9月30日(上期) 2: 10月1日 ～ 3月31日(下期) 「YYYY」と共に記述した場合 1: 1月1日 ～ 6月30日(上期) 2: 7月1日 ～ 12月31日(下期)

書式要素	意味
Q	四半期(1～4) データの表現は1文字固定 「YYYJ」と共に記述した場合 1: 4月1日 ～ 6月30日 (第1四半期) 2: 7月1日 ～ 9月30日 (第2四半期) 3: 10月1日 ～ 12月31日 (第3四半期) 4: 1月1日 ～ 3月31日 (第4四半期) 「YYYY」と共に記述した場合 1: 1月1日 ～ 3月31日 (第1四半期) 2: 4月1日 ～ 6月30日 (第2四半期) 3: 7月1日 ～ 9月30日 (第3四半期) 4: 10月1日 ～ 12月31日 (第4四半期)
hh	時間(1～12) 時間の開始は12
hh24	時間(0～23) 時間の開始は0
mm	分(0～59) 分の開始は0
ss	秒(0～59) 秒の開始は0
AMまたはPM	12時間表記の文字列を24時間の表記の時刻に変換します。

CNV_TIMESTAMP関数に指定可能な書式要素の組合せ

CNV_TIMESTAMP関数には、指定可能な書式要素の組合せがあります。

CNV_TIMESTAMP関数に指定可能な書式要素の組合せは“CNV_DATE関数に指定可能な書式要素の組合せ”と“CNV_TIME関数に指定可能な書式要素の組合せ”を参照してください。

セパレータ

セパレータに指定可能な文字はCNV_DATE関数と共通です。詳細はCNV_DATE関数の“セパレータ”を参照してください。

注意事項

時刻印書式には、上記の書式要素を任意の順序で指定することができます。また書式要素を任意のセパレータで区切って指定することもできます。ただし、以下のような指定をした場合、エラーとなります。

- 同じ書式要素を2度以上指定する。

```
CNV_TIMESTAMP('2007-04-10 14:12:20','YYYY.MM.DD hh24:mm:ss:ss') →エラー
```

- セパレータを2文字以上連続して記述する。

```
CNV_TIMESTAMP('2007-04-10 14:12:20','YYYY//MM///DD hh24::mm::ss') →エラー
```

- セパレータを時刻印書式の先頭、末尾に記述する。

```
CNV_TIMESTAMP('2007-04-10 14:12:20','YYYY/MM/DD hh24:mm:ss:') →エラー
```

- 書式要素にH、またはQを指定した場合、YYYYとYYYJを同時に指定することはできません。どちらか一方を指定してください。

```
CNV_TIMESTAMP('2007-04-10 14:12:20','YYYY-YYYJ-H-Q hh24:mm:ss') →エラー
```

- 書式要素にhhを指定した場合、AMとPMを同時に指定することはできません。どちらか一方を指定してください。

```
CNV_TIMESTAMP('2007-04-10 14:12:20','YYYY-MM-DD hh:mm:ss:AM:PM') →エラー
```

- CNV_TIMESTAMP関数に、書式要素MONTH、MON、DAYおよびDYを指定する場合、変換元の文字列は、以下の文字列と完全に一致していなければいけません。大文字、小文字の違いでもエラーとなりますので注意が必要です。

```
MONTH:January,February,March,April,May,June,July,August,September,October,November, December
MON:JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
DAY:Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
DY:SUN,MON,TUE,WED,THU,FRI,SAT
```

- 書式要素DAYおよびDYを指定した場合、指定した日付と曜日が正しいかどうかの妥当性チェックは行いません。誤っていても曜日は無視して処理されます。

使用例

例1

CURRENT DATE値関数(日別発注表にデータを1行追加します。処理日には現在の日付を設定します。)

```
INSERT INTO 在庫管理.日別発注表(取引先,取引製品,発注数量,処理日)
    VALUES(61,:PRODNO,:ORDERQTY,CURRENT_DATE)
```

例2

CURRENT TIME値関数(日別発注表にデータを1行追加します。処理時間には現在の時間を設定します。)

```
INSERT INTO 在庫管理.日別発注表(取引先,取引製品,発注数量,処理時間)
    VALUES(61,:PRODNO,:ORDERQTY,CURRENT_TIME)
```

例3

CURRENT TIMESTAMP値関数(日別発注表にデータを1行追加します。処理日時には現在の日時を設定します。)

```
INSERT INTO 在庫管理.日別発注表(取引先,取引製品,発注数量,処理日時)
    VALUES(61,:PRODNO,:ORDERQTY,CURRENT_TIMESTAMP)
```

例4

ROUND_DATE関数(丸めた月ごとの売上金額の合計を求めます。)

```
SELECT 丸め月,SUM(売上金) FROM 売上表
    GROUP BY ROUND_DATE(販売日,'MONTH') AS 丸め月
```

例5

TRUNC_DATE関数(四半期ごとの発注数量の平均を求めます。)

```
SELECT 四半期,AVG(発注数量) FROM 発注表
    GROUP BY TRUNC_DATE(発注日,'QUARTER') AS 四半期
```

例6

ADD_DATE関数(発注日から1ヶ月以上経過している発注番号を求めます。)

```
SELECT 発注番号 FROM 発注表
    WHERE 納品日 IS NULL AND ADD_DATE(発注日,1,'MONTH') < CURRENT_DATE
```

例7

LAST_DAY関数(発注日から月末までの日数を求めます。)

```
SELECT 発注番号,SPAN_DATE(発注日,LAST_DAY(発注日),'DAY')
    FROM 発注表 WHERE 納品日 IS NULL
```

例8

CNV_DATE関数(最低気温日付文字表の日付文字:Sunday January 14,2007を日付に変換します。)

```
SELECT 地区コード , CNV_DATE(日付文字, 'DAY MONTH DD,YYYY')
    FROM 最低気温日付文字表
```

例9

CNV_TIME関数(在庫管理表の更新時刻文字: 21:20:44を時刻に変換します。)

```
SELECT 商品コード , CNV_TIME(更新時刻文字, 'hh24:mm:ss')
    FROM 在庫管理表
```

例10

CNV_TIMESTAMP関数(会議室予約表の使用日時文字:20070114 21:20:44を時刻印に変換します。)

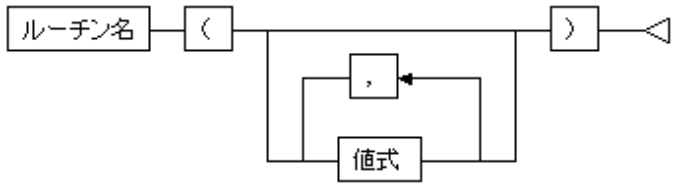
```
SELECT  予約番号, CNV_TIMESTAMP(使用日時文字, 'YYYYMMDD hh24:mm:ss')
    FROM 会議室予約表
```

2.5.5 ファンクションルーチン指定

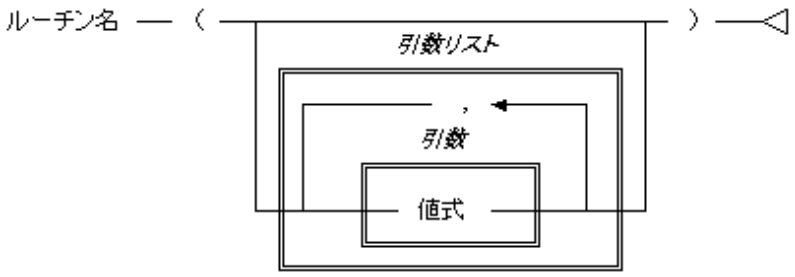
機能

ファンクションルーチンを指定します。

記述形式



構文の構成



権限

- ファンクションルーチンを記述したSQL文を実行できるのは、ファンクションルーチンの定義者とEXECUTE権を付与された人です。

一般規則

ルーチン名

ー ファンクションルーチンの名前を指定します。

ポイント

ルーチン名はスキーマ名修飾することを推奨します。

値式

ー 値式は、ファンクションルーチン定義時のパラメタのデータ型に対して比較可能なデータ型である必要があります。

使用例

例

DATE型のデータを‘yyyy年mm月dd日’の形式の文字列に変換するファンクションルーチンUSERFUNC001を定義します。列C1の値がDATE型で‘2007-08-03’のとき結果は‘2007年08月03日’になります。

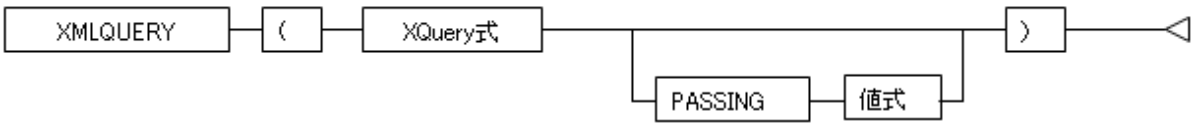
```
SELECT STOCKS.USERFUNC001(C1) FROM 在庫管理.発注表
    WHERE C2=10
```

2.5.6 XMLQUERY関数

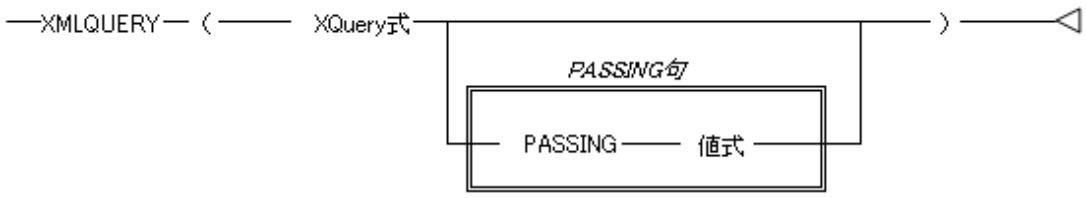
機能

XMLQUERY関数は、XQuery式の評価結果であるXMLデータを返却する関数です。

記述形式



構文要素の構成



参照項番

- 値式 → “2.11 値式”

一般規則

- XMLQUERY関数のデータ型は、BLOB型です。

- XMLQUERY関数は、XQuery式の評価結果であるシーケンスの項目を、先頭から順に連結したXMLデータを返します。
- XQuery式の評価結果が空シーケンスである場合、XMLQUERY関数はNULLを返します。
- XMLQUERY関数の結果は31キロバイト以下であることが必要です。31キロバイトを超えた場合はエラーになります。
- 以下の値式として、XMLQUERY関数を指定することはできません。
  - 問合せ式で選択リストにDISTINCTを指定した場合の選択リストに指定する値式(ただし、問合せ指定の結果がBLOB型でないならば、エラーにはなりません)
  - GROUP BY句に指定するグループ化関数、CASE式、またはファンクションルーチン指定の引数に指定する値式
  - ORDER BY句に指定する値式
  - ビュー表の問合せ指定に指定する値式
  - XMLQUERY関数のPASSING句に指定する値式
  - XMLEXISTS述語のPASSING句に指定する値式
  - 副問合せに指定する値式
  - トリガ定義の以下のいずれかに指定する値式
    - 被トリガSQL文
    - 被トリガ動作
  - 列の定義長の合計が32キロバイト以上の表を指定したSQL文に指定する値式
  - OBJECT構造の表を指定したSQL文に指定する値式
- XMLQUERY関数を問合せ指定に指定した場合、並列検索は行いません。並列検索が指定されても無視します。

XQuery式

- XMLデータの文字列定数で指定します。

参照

XQuery式に指定する検索式の詳細は、“XQueryリファレンス”を参照してください。

注意

XQuery式に指定する検索式は文字列定数で表現するため、検索式の記述内容に'(シングルクォーテーション)を指定することはできません。'(シングルクォーテーション)を含む文字列リテラルを指定する場合には、既定義エンティティー参照の'を指定してください。
既定義エンティティー参照の詳細は、“XQueryリファレンス”を参照してください。

例

A要素のテキストノードの値が「'ABC銀行'」を表すXQuery式

```
XMLQUERY('/A[text()="&apos;ABC銀行&apos;"]'
   PASSING 伝票)
```

- XQuery式に空文字または空白文字列のみ指定した場合、XMLQUERY関数はエラーになります。
- XQuery式の問合せの評価時にエラーが発生した場合、XMLQUERY関数はエラーになります。

PASSING句

- PASSING句の値式にはXQuery式の検索対象となるXMLデータを指定します。
- PASSING句の値式のデータ型は、BLOB型であることが必要です。

ー PASSING句に以下の値式を指定することはできません。

- 集合関数指定
- CAST指定
- XMLQUERY関数
- ファンクションルーチン

ー 値式に指定するXMLデータは、正しいデータ形式であることが必要です。誤ったデータ形式であることを検出した場合、XMLQUERY関数はNULLを返します。

参照

XMLデータの正しい形式に関しては、“アプリケーション開発ガイド(共通編)”の“SQL/XMLで扱えるデータ形式”を参照してください。

ー PASSING句を省略、またはPASSING句の値式の値がNULLである場合、XMLQUERY関数の評価結果は不定になります。

DESCRIBE情報について

ー 値式に動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.43 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
PASSING句の値式	BLOB型

使用例

製品番号が100番台の製品を発注している、会社番号が60番台の取引会社の会社名を検索します。

```
SELECT  会社表.会社番号,
        XMLQUERY('/取引先会社情報[発注/製品番号/text() >= 100 and
                  発注/製品番号/text() < 200]/会社名/text()'
                  PASSING  会社表.会社情報)  AS 会社名
FROM  会社表
WHERE  会社番号 BETWEEN 60 AND 69
```

上記の問合せにおける、取引先会社と発注製品の情報を以下に示します。

会社表

会社番号	会社情報
61	XML1
62	XML2
63	XML3
71	XML4
72	XML5

XML1

```
<取引先会社情報>
  <会社名>アダム電気</会社名>
  <発注>
    <製品番号>130</製品番号>
  </発注>
    :
</取引先会社情報>
```

XML2

```
<取引先会社情報>
  <会社名>アイデア商事</会社名>
  <発注>
    <製品番号>150</製品番号>
  </発注>
    :
</取引先会社情報>
```

XML3

```
<取引先会社情報>
  <会社名>大月産業</会社名>
  <発注>
    <製品番号>200</製品番号>
  </発注>
    :
</取引先会社情報>
```

XML4

```
<取引先会社情報>
  <会社名>川川電気</会社名>
  <発注>
    <製品番号>320</製品番号>
  </発注>
    :
</取引先会社情報>
```

XML5

```
<取引先会社情報>
  <会社名>竜巻産業</会社名>
  <発注>
    <製品番号>200</製品番号>
  </発注>
    :
</取引先会社情報>
```

XMLQUERY関数の指定により、会社表の会社情報列に格納された各XMLデータに対して、XQuery式'/取引先会社情報[発注/製品番号/text() >= 100 and 発注/製品番号/text() < 200]/会社名/text()'が評価されます。XQuery式の評価結果より、XMLQUERY関数は図中の会社番号が61と62の行に対して、XMLデータから抽出した会社名を返します。また、会社番号が63の行に対してはNULLを返します。このSQL文の結果を以下に示します。

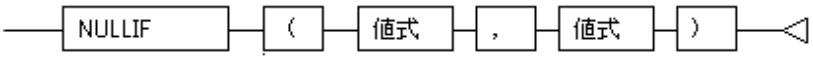
会社番号	会社名
61	アダム電気
62	アイデア商事
63	NULL

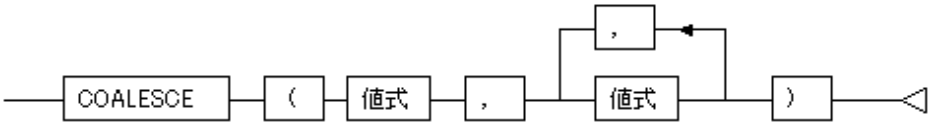
2.6 CASE式

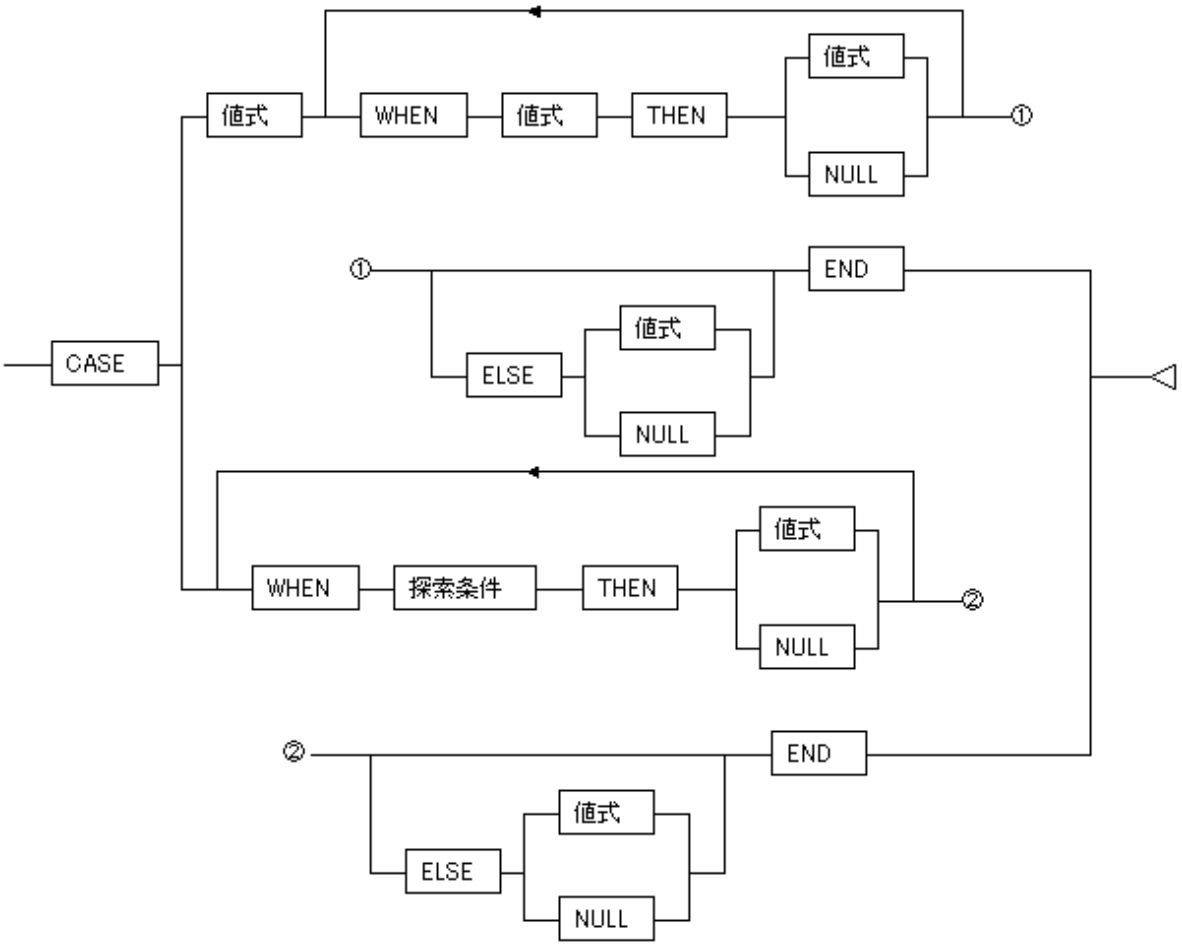
機能

条件付けられた値を指定します。

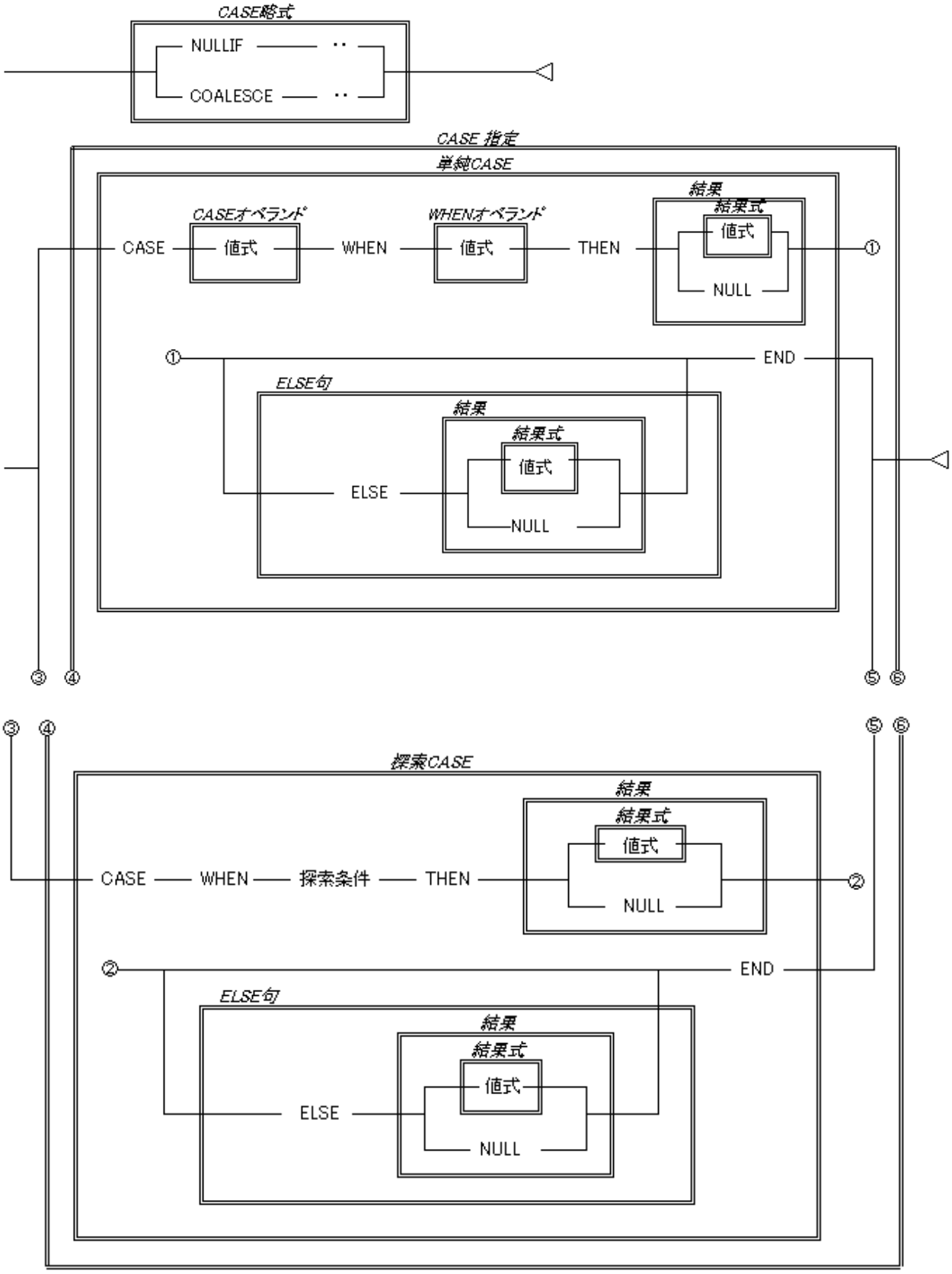
記述形式






CASE
値式
WHEN
値式
THEN
値式
NULL
①
END
ELSE
値式
NULL
WHEN
探索条件
THEN
値式
NULL
②
END
ELSE
値式
NULL

構文要素の構成



参照項番

- 値式 → “2.11 値式”

・ 探索条件 → “2.13 探索条件”

一般規則

CASE略式

NULLIF

- NULLIFは、第1オペランドと第2オペランドが等しい場合にNULL値、そうでない場合に第1オペランドの値を返却します。
- NULLIF(V1,V2)は、以下のCASE指定と同じです。

```
CASE WHEN V1=V2 THEN NULL ELSE V1 END
```

- NULLIFの値式の両方に動的パラメタ指定を指定することはできません。
- NULLIFの1番目または2番目の値式に動的パラメタ指定が指定された場合、DESCRIBE情報はもう一方の値式のデータ型となります。
- 値式として順序を指定することはできません。

COALESCE

- COALESCEは、指定したオペランドの値式のうちで最初に出現したNULL値でない値を返却します。すべての値式がNULL値の場合は、最後のオペランドの値を返却します。
- COALESCE (V1,V2)は、以下のCASE指定と同じです。

```
CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END
```

- COALESCE(V1,V2・・・,Vn)は、以下のCASE指定と同じです。

```
CASE WHEN V1 IS NOT NULL THEN V1 ELSE
          COALESCE(V2, ・・・, Vn) END
```

- COALESCEの値式すべてに動的パラメタ指定を指定することはできません。
- COALESCEの値式に動的パラメタ指定が指定された場合、DESCRIBE情報は他のすべての値式に対して、集合演算(UNION)におけるデータ変換で示す規約を適用することで決定されます。結果のデータ型については、“表3.9 集合演算(UNION)におけるデータ変換”を参照してください。
- 値式として順序を指定することはできません。

CASE指定

ー CASE指定が単純CASEの場合、以下のようになります。

- 各WHENオペランドの値式のデータ型は、CASEオペランドの値式のデータ型と比較可能でなければなりません。比較可能なデータ型に関する規則は、“2.12.1 比較述語”を参照してください。
- 単純CASE指定は、各探索WHEN句を“CASEオペランド=WHENオペランド”の形式で指定した探索CASEと同じです。

```
CASE V1 WHEN V2 THEN V3
        WHEN V4 THEN V5
                ELSE V6 END

          ↓同じ

CASE WHEN V1=V2 THEN V3
     WHEN V1=V4 THEN V5
                ELSE V6 END
```

ー CASEオペランド、WHENオペランドおよび探索条件に順序を指定することはできません。

ー CASE指定の少なくとも1つの結果には結果式を指定しなければなりません。

- ELSEの指定がない場合、ELSE NULLが指定されたものとみなします。
- CASE指定の結果のデータ型は、すべての結果に対して集合演算(UNION)におけるデータ変換で示す規約を適用することで決定されます。結果のデータ型については、“表3.9 集合演算(UNION)におけるデータ変換”を参照してください。
- 探索CASEの探索条件には副問合せは指定できません。
- CASE指定の結果式すべてに動的パラメタ指定を指定することはできません。
- 単純CASEのCASEオペランド、複数のWHENオペランドすべてに動的パラメタ指定を指定することはできません。
- 動的パラメタ指定が指定された場合のDESCRIBE情報は、以下のとおりです。
  - CASE指定の結果式に動的パラメタ指定が指定された場合、DESCRIBE情報は他の結果式に指定された値式に対して、集合演算(UNION)におけるデータ変換で示す規約を適用することで決定されます。結果のデータ型については、“表3.9 集合演算(UNION)におけるデータ変換”を参照してください。
  - 単純CASEのWHENオペランドに動的パラメタ指定が指定された場合、DESCRIBE情報はCASEオペランドと他のすべてのWHENオペランドに対して、集合演算(UNION)におけるデータ変換で示す規約を適用することで決定されます。結果のデータ型については、“表3.9 集合演算(UNION)におけるデータ変換”を参照してください。
  - 単純CASEのCASEオペランドに動的パラメタ指定が指定された場合、DESCRIBE情報はすべてのWHENオペランドに対して、集合演算(UNION)におけるデータ変換で示す規約を適用することで決定されます。結果のデータ型については、“表3.9 集合演算(UNION)におけるデータ変換”を参照してください。

使用例

例1

CASE略式、NULLIF指定の場合

利用者表から、名前の愛称を求めます。ただし、愛称は名前とは異なる内容の設定がある場合のみ取り出します。

```
SELECT 名前,NULLIF(愛称,名前) AS 愛称
    FROM 利用者表
```

利用者表

利用者	名前	愛称	メール	TEL	住所
USER-1	TANAKA	TANAKA	E-MAIL-A	-	OSAKA
USER-2	SUZUKI	NAME-B	-	TEL-B	TOKYO
USER-3	KATOU	NAME-C	E-MAIL-C	TEL-C	KYUSYU

⇒

名前	愛称
TANAKA	-
SUZUKI	NAME-B
KATOU	NAME-C

-:NULL値

例2

CASE略式、COALESCE指定の場合

利用者表から、メールアドレス、TEL番号、住所の優先順で最初のNULL値以外のデータを求めます。

```
SELECT 利用者, COALESCE(メール,TEL,住所) AS 連絡先
    FROM 利用者表
```

利用者表

利用者	名前	愛称	メール	TEL	住所
USER-1	TANAKA	TANAKA	E-MAIL-A	-	OSAKA
USER-2	SUZUKI	NAME-B	-	TEL-B	TOKYO
USER-3	KATOU	NAME-C	E-MAIL-C	TEL-C	KYUSYU

利用者	連絡先
USER-1	E-MAIL-A
USER-2	TEL-B
USER-3	E-MAIL-C

-:NULL値

例3

CASE指定、単純CASE指定の場合

履歴表から、利用者ごとに利用したサービスの回数をサービス別に求めます。

```
SELECT 利用者,COUNT(CASE サービス WHEN 1
                  THEN サービス ELSE NULL END) AS サービス1
              ,COUNT(CASE サービス WHEN 2
                  THEN サービス ELSE NULL END) AS サービス2
              ,COUNT(CASE サービス WHEN 3
                  THEN サービス ELSE NULL END) AS サービス3
   FROM 履歴表 GROUP BY 利用者
```

履歴表

履歴番号	利用者	サービス
1	USER-A	1
2	USER-B	2
3	USER-A	2
4	USER-A	1
5	USER-C	3

利用者	サービス1	サービス2	サービス3
USER-A	2	1	0
USER-B	0	1	0
USER-C	0	0	1

例4

CASE指定、探索CASE指定の場合

履歴表から、利用者ごとに利用したサービスの回数をサービス別に求めます。

```
SELECT 利用者,COUNT(CASE WHEN サービス = 1
                  THEN サービス ELSE NULL END) AS サービス１
              ,COUNT(CASE WHEN サービス = 2
                  THEN サービス ELSE NULL END) AS サービス2
              ,COUNT(CASE WHEN サービス = 3
                  THEN サービス ELSE NULL END) AS サービス3
   FROM 履歴表 GROUP BY 利用者
```

履歴表

履歴番号	利用者	サービス
1	USER-A	1
2	USER-B	2
3	USER-A	2
4	USER-A	1
5	USER-C	3

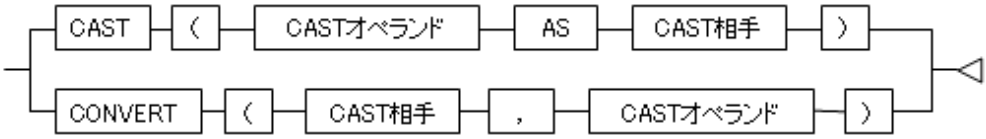
利用者	サービス1	サービス2	サービス3
USER-A	2	1	0
USER-B	0	1	0
USER-C	0	0	1

2.7 CAST指定

機能

データ変換を指定します。

記述形式



一般規則

- CASTは、CASTオペランドで指定した値をCAST相手で指定するデータ型に変換します。
- CASTオペランドは値式です。
- CAST相手は、データ型です。
- CONVERTは、CASTと同じ意味です。

CASTオペランドのデータ型が真数の場合の変換規則

- CAST相手のデータ型が真数または概数の場合、上位有効桁が失われた場合は、エラーになります。また、精度が失われた場合は、切捨てになります。
- CAST相手のデータ型が固定長文字の場合、位取りは、CASTオペランドのデータ型と同じになります。また、CAST相手の文字列長が長い場合は、右側に空白を入れ、CAST相手の文字列長が短い場合は、エラーになります。また、CASTオペランドが負数の場合、先頭に'ー'を付けます。
- CAST相手のデータ型が可変長文字の場合、文字列長が合わせられること以外は固定長文字と同じになります。
- CAST相手のデータ型が時間隔の場合、上位有効桁が失われた場合は、エラーになります。また、精度が失われた場合は、切捨てになります。

CASTオペランドのデータ型が概数の場合の変換規則

- CAST相手のデータ型が真数または概数の場合、上位有効桁が失われた場合は、エラーになります。また、精度が失われた場合は、切捨てになります。
- CAST相手のデータ型が固定長文字の場合、CAST相手の文字列長が長い場合は、右側に空白を入れ、CAST相手の文字列長が短い場合は、エラーになります。また、CASTオペランドが0の場合は、'0E0'になり、CASTオペランドが負数の場合は、先頭に'ー'を付けます。また、CASTオペランドの絶対値が1未満の場合は、指数部に'ー'を付けます。仮数部の先頭の正の符号'+'は省略します。
- CAST相手のデータ型が可変長文字の場合、文字列長が合わせられること以外は固定長文字と同じになります。

CASTオペランドのデータ型が固定長文字または可変長文字の場合の変換規則

- CAST相手のデータ型が真数または概数の場合、前後に空白がある場合は、空白は無視します。また、不当な文字がある場合は、エラーになります。ほかは文字を数値とみなした場合の変換規則と同じになります。
- CAST相手のデータ型が固定長文字の場合、CAST相手の文字列長が長い場合は、右側に空白を入れ、CAST相手の文字列長が短い場合は、右側が切捨てになります。
- CAST相手のデータ型が可変長文字の場合、文字列長が合わせられること以外は固定長文字と同じになります。
- CAST相手のデータ型が日時の場合、前後に空白がある場合は、空白は無視します。また、各日時フィールドの先行する0は省略可能になります。また、文字列が日時値として不当な場合は、エラーになります。
- CAST相手のデータ型が時間隔の場合、前後に空白がある場合は、空白は無視します。また、文字列がCAST相手の時間隔値として不当な場合は、エラーになります。また、上位有効桁が失われた場合は、エラーになります。

CASTオペランドのデータ型が各国語固定長文字または各国語可変長文字の場合の変換規則

- CAST相手のデータ型が各国語固定長文字の場合、CAST相手の文字列長が長い場合は、右側に空白を入れ、CAST相手の文字列長が短い場合は、右側が切捨てになります。
- CAST相手のデータ型が各国語可変長文字の場合、文字列長が合わせられること以外は各国語固定長文字と同じになります。

CASTオペランドのデータ型が日時の場合の変換規則

- CAST相手のデータ型が固定長文字の場合、先行する0は省略しません。また、CAST相手の文字列長が長い場合は、右側に空白を入れ、CAST相手の文字列長が短い場合は、エラーになります。
- CAST相手のデータ型が可変長文字の場合、文字列長が合わせられること以外は固定長文字と同じになります。
- CAST相手のデータ型が日時の場合、同じデータ型間では変換可能です。時刻印から日付の場合は、時刻印の日付部分になり、時刻印から時刻の場合は、時刻印の時刻部分になり、日付から時刻印の場合は、時刻部分は'00:00:00'になり、時刻から時刻印の場合は、日付部分は現在の日付になります。

CASTオペランドのデータ型が時間隔の場合の変換規則

- CAST相手のデータ型が真数の場合、上位有効桁が失われた場合は、エラーになります。
- CAST相手のデータ型が固定長文字の場合、CAST相手の文字列長が長い場合は、右側に空白を入れ、CAST相手の文字列長が短い場合は、エラーになります。また、CASTオペランドが負数の場合、先頭に'ー'を付けます。
- CAST相手のデータ型が可変長文字の場合、文字列長が合わせられること以外は固定長文字と同じになります。
- CAST相手のデータ型が時間隔の場合、上位有効桁が溢れた場合は、エラーになります。また、精度が失われた場合は、切捨てになります。

CAST指定により変換可能なデータ型の組み合わせ

- CAST指定により変換可能なデータ型の組合せを以下に示します。なお、BLOB型は指定できません。

表2.44 CAST指定により変換可能なデータ型の組合せ

			CAST相手データ型										
			真数	概数	文字型		各国語		日時			時間隔	
					固定長	可変長	固定長	可変長	日付	時刻	時刻印	年月	日時
CASTオペランドデータ型	真数		○	○	○	○	×	×	×	×	×	△	△
	概数		○	○	○	○	×	×	×	×	×	×	×
	文字型	固定長	○	○	○	○	×	×	○	○	○	○	○
		可変長	○	○	○	○	×	×	○	○	○	○	○
	各国語	固定長	×	×	×	×	○	○	×	×	×	×	×
		可変長	×	×	×	×	○	○	×	×	×	×	×
	日時	日付	×	×	○	○	×	×	○	×	○	×	×
		時刻	×	×	○	○	×	×	×	○	○	×	×
		時刻印	×	×	○	○	×	×	○	○	○	×	×

			CAST相手データ型										
			真数	概数	文字型		各国語		日時			時間隔	
					固定長	可変長	固定長	可変長	日付	時刻	時刻印	年月	日時
	時間隔	年月	△	×	○	○	×	×	×	×	×	○	×
		日時	△	×	○	○	×	×	×	×	×	×	○

○: 変換可能な組合せ

△: 時間隔が単一日時フィールドだけを含む場合に可能な組合せ

×: 変換不可能な組合せ

DESCRIBE情報について

－ CAST指定の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.45 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
CASTオペランド	CAST相手のデータ型

導出表にCAST指定を指定した場合の注意事項

導出表を記述したSQL文を実行した場合、Symfoware/RDBの最適化処理で入れ子をなくし平坦な形式のSQL文に変換する場合があります。このSQL文の変換では、変換前のSQL文の導出表の選択リストにCAST指定が指定されていると、CAST指定を条件式に移動することがあります。

このとき、データベースにCAST指定で変換できないデータが含まれると、導出表の条件式において変換できないデータを除外していたとしても、条件式に移動したCAST指定において、CAST指定の変換エラーが発生する場合があります。そのため、データベースにCAST指定で変換できないデータが含まれる場合は、CASE式を用いて、変換できないデータを除外することにより、CAST指定で変換エラーが発生しないように対処してください。
以下に例を示します。

実行するSQL文

```
SELECT T1_SUB2.CAST_TIMESTAMP
    FROM
     (SELECT CAST_TIMESTAMP
        FROM
         (SELECT CAST(CHAR_TIMESTAMP AS TIMESTAMP )
                    AS CAST_TIMESTAMP       ・・・・・(注1)
            FROM S1.T1
            WHERE  CHAR_TIMESTAMP <>'              '   ・・・・・(注2)
         ) AS T1_SUB (CAST_TIMESTAMP)
     ) T1_SUB2 (CAST_TIMESTAMP)
    WHERE T1_SUB2.CAST_TIMESTAMP > TIMESTAMP'2009-08-01 15:58:59'
```

注1) 選択リストにCAST指定を記述。

注2) 最も内側の導出表にて文字列の空白以外を取り出す判定。

最適化処理が変換したSQL文

Symfoware/RDBの最適化処理が変換したSQL文は、以下のようになります。

```
SELECT CAST(T1.CHAR_TIMESTAMP AS TIMESTAMP)
  FROM S1.T1
  WHERE T1.CHAR_TIMESTAMP <>'                    ' ・・・・・(注1)
    AND CAST(T1.CHAR_TIMESTAMP AS TIMESTAMP) > TIMESTAMP'2009-08-01 15:58:59' ・・・・・(注2)
```

注1) 文字列の空白以外を取り出す判定。

注2) 文字列をCASTし、変換結果を判定。

その結果、注2のCAST指定に指定された列の値が空白の場合、TIMESTAMP型にデータ型変換することができず、変換エラーになります。

対処例

CASE式を用いて、空白のデータを除外することにより、CASTで変換エラーが発生しないように対処してください。

```
SELECT T1_SUB2.CAST_TIMESTAMP
    FROM
    (SELECT CAST_TIMESTAMP
      FROM
        (SELECT (CASE WHEN CHAR_TIMESTAMP <>'                    '
                  THEN
                   CAST(CHAR_TIMESTAMP AS TIMESTAMP )
                  ELSE
                   NULL
                 END) AS CAST_TIMESTAMP
          FROM S1.T1
          WHERE  CHAR_TIMESTAMP <>'                    '
        ) AS T1_SUB (CAST_TIMESTAMP)
    ) T1_SUB2 (CAST_TIMESTAMP)
  WHERE
      T1_SUB2.CAST_TIMESTAMP > TIMESTAMP'2009-08-01 15:58:59'
```

使用例

例1

SMALLINT型に変換します。

```
CAST(10.13 AS SMALLINT)
→結果は10になります。
```

例2

CHAR型に変換します。

```
CAST(-99.99 AS CHAR(6))
→結果は-99.99になります。
```

例3

CHAR型に変換します。

```
CAST(INTERVAL '20:30' MINUTE TO SECOND AS CHAR(5))
→結果は20:30になります。
```

例4

営業所一覧表から、処理日に6ヶ月加算した日付を取り出します。文字列型のホスト変数“CDATA”には“6”が設定されているとします。

```
SELECT コード,処理日 + CAST(:CDATA AS INTERVAL MONTH)
    FROM 在庫管理.営業所一覧表
```

例5

SMALLINT型に変換します。

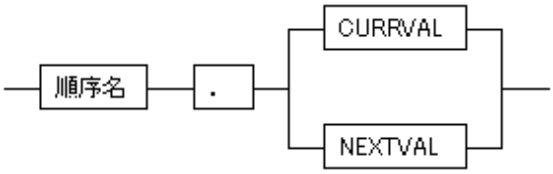
```
CONVERT(SMALLINT,10.13)
→結果は10になります。
```

2.8 順序

機能

順序はRDBシステム全体で一意の値を生成します。

記述形式



権限

・ 順序を使用したSQL文が実行できるのは、順序の定義者とSELECT権を付与された人のみです。

一般規則

・ CURRVALおよびNEXTVALはアプリケーションおよびストアドプロシジャの以下のSQL文の箇所に指定できます。

  ― 単一行SELECT文の選択リスト

  ― 問合せ指定の選択リスト:カーソル指定の問合せ式のみ

  ― UPDATE文のSET句

  ― INSERT文に指定した問合せ指定の選択リスト

  ― INSERT文のVALUES句

  ― ストアドプロシジャのSET文

  ― 被トリガSQL文

・ SQL文中に同一の順序のNEXTVALを複数個指定している場合は、すべて同一の順序番号が返却されます。また、SQL文中に同一の順序のNEXTVALとCURRVALの両方を指定している場合は、それらが文中で出現する順番にかかわらずNEXTVALとCURRVALの両方に対して、NEXTVALとして採番された同じ順序番号が返却されます。

- SELECT文に順序を指定すると、順序番号は検索結果の行に対して生成されます。
- UPDATE文に順序を指定すると、順序番号は更新行に対して生成されます。
- INSERT文に順序を指定すると、順序番号は挿入先の挿入行に対して生成されます。

順序名

― 順序の名前を指定します。

CURRVAL

― アプリケーションのセション内で最後に生成した値を返却します。なお、CURRVALを使用するときは、あらかじめNEXTVALで順序番号を生成している必要があります。

NEXTVAL

― RDBシステム内で最後に生成した値の後続の値を返却します。

注意

順序定義で順序保証指定“ORDER”を指定せずに割当順序数指定“CACHE”を指定している場合、ロードシェア機能利用時にシステムダウンが発生すると、順序番号が昇順または降順に生成されないことがあります。なお、順序番号の一意性は保証されます。

使用例

例

製品の納入に伴い、納品管理表および在庫表に製品データを格納します。その時、おのおの表の管理番号には順序を使用して生成した一意の番号を設定します。

NEXTVALを使用して新しい管理番号を生成します。

```
INSERT INTO STOCKS.納品管理表
    (管理番号,製品番号,製品名,数量)
    VALUES(STOCKS.順序1.NEXTVAL,110,'テレビ',86)
```

CURRVALを使用して納品管理表に設定した管理番号と同じ番号を格納します。

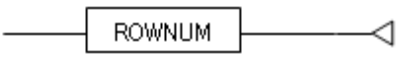
```
INSERT INTO STOCKS.在庫表
    (管理番号,製品番号,製品名,在庫数量,倉庫番号)
    VALUES(STOCKS.順序1.CURRVAL,110,'テレビ',86,2)
```

2.9 ROWNUM

機能

ROWNUMには、FROM句の結果の表に対して、WHERE句の探索条件を適用した結果の行に、1から振った順番号が入ります。

記述形式



一般規則

- ROWNUMのデータ型は、精度が18、位取りが0のNUMERIC型です。
- ROWNUMから値を読み込むことができますが、ROWNUMに対して値の挿入、更新はできません。

- ROWNUMには、FROM句の結果の表に対して、WHERE句の探索条件を適用した結果の行に1から振った順番号が入ります。1件目の取り出しでは、ROWNUMに1が入り、2件目の取り出しでは、ROWNUMに2が入ります。以降、これを繰り返します。WHERE句に“ROWNUM < 件数”を指定すると、結果として取り出す件数を限定することができます。ただし、“ROWNUM > 2”などを指定すると、1件目取り出しの判定が常に偽となり、ROWNUMに2以上の値が入らないため、検索結果として取り出す件数は0件となります。
- WHERE句を適用した結果の行の順序に規則性はないので、ROWNUMの値と行の対応関係もそのときの取り出し順に依存します。
- GROUP BY句、集合演算、またはORDER BY句は、ROWNUMに順番号を入れた後に行います。そのため、行の順が入れ代わり、ROWNUMの値の順に取り出すことができません。たとえば、ソートされたカーソルの先頭から、ある件数分を結果として取り出すような場合、ORDER BY句を導出表に指定し、その導出表を読み込むときにWHERE句にROWNUMを指定して件数を制限します。
- ROWNUMは以下の箇所にのみ指定可能です。

SQL文	
カーソル宣言	選択リスト
	導出表(カーソル宣言と同じ)
	WHERE句
	副問合せ(カーソル宣言と同じ)
単一行SELECT文(カーソル宣言と同じ)	
CREATE PROCEDURE文	カーソル宣言
	単一行SELECT文

使用例

例1

在庫表から在庫数量が100以下の行を、先頭から5行分取り出します。

```
SELECT 製品番号,製品名,在庫数量
  FROM 在庫管理.在庫表 WHERE 在庫数量 <= 100 AND ROWNUM < 6
```

例2

在庫表から、製品番号が100から300の製品の数を求めます。ただし、50行を超える検索はしません。

```
SELECT COUNT(*) FROM 在庫管理.在庫表
 WHERE 製品番号 BETWEEN 100 AND 300 AND ROWNUM < 51
```

例3

在庫表から、製品番号が100から199で在庫数量が多い順に3行取り出します。

```
SELECT 製品番号,製品名,在庫数量,倉庫番号
  FROM (SELECT 製品番号, 製品名,在庫数量,倉庫番号 FROM 在庫管理.在庫表
                  WHERE 製品番号 BETWEEN 100 AND 199
                  ORDER BY 在庫数量 DESC) AS D1(製品番号,製品名,在庫数量,倉庫番号)
                        (1)
 WHERE ROWNUM < 4
           (2)
```

(1) 在庫数量の降順にソートします。

(2) 在庫数量が多い順に3行取り出します。

例4

在庫表から、製品番号が100から200の行を取り出し、取り出した行に1から順に番号を振ります。

```
SELECT ROWNUM AS "ROWNUM",製品番号,製品名,在庫数量 FROM 在庫管理.在庫表
 WHERE 製品番号 BETWEEN 100 AND 200
```

例5

在庫表から、製品番号が100から300の行の、先頭から11件目以降を取り出します。

```
SELECT 製品番号,行番号
  FROM (SELECT 製品番号,ROWNUM FROM 在庫管理.在庫表
                              (1)
         WHERE 製品番号 BETWEEN 100 AND 300) AS D1(製品番号,行番号)
 WHERE 行番号 > 10
          (2)
```

(1) 問合せの結果に対し、1から順に行番号を振ります。

(2) ROWNUMで振った行番号を判定します。

誤った使い方の例

```
SELECT 製品番号
  FROM 在庫管理.在庫表 WHERE 製品番号 BETWEEN 100 AND 200 AND ROWNUM > 10
                                                              (1)
```

(1) 結果集合の行にROWNUMの値を1から順に振りますが、探索条件は常に偽になるため、2以上の値が振られることがありません。そのため、結果集合の行はすべて偽になり、検索結果は0件になります。

例6

取引先などあるソートキーの昇順に画面を次々にめくっていくような場合、前回の続きから取り出します。

```
SELECT 行番号,取引先,取引製品,仕入価格
  FROM (SELECT ROWNUM,取引先,取引製品,仕入価格
                 (3)
          FROM (SELECT 取引先,取引製品,仕入価格
                  FROM 在庫管理.発注表
                 WHERE 取引先 > 61
                 ORDER BY 取引先) AS D1(取引先,取引製品,仕入価格)
                       (1)
          WHERE ROWNUM <= 10)
                   (2)
       AS D2(行番号, 取引先,取引製品,仕入価格)
 WHERE 行番号 >= 6
          (4)
```

(1) 取引先の昇順にソートします。

(2) 取引先が小さい順に10行取り出します。

(3) 取引先でソートした結果に1から順番号を振ります。

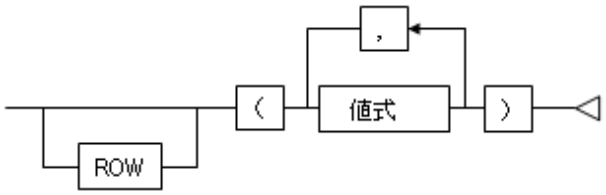
(4) 順番号について、6以上の行を取り出します。

2.10 行値構成子

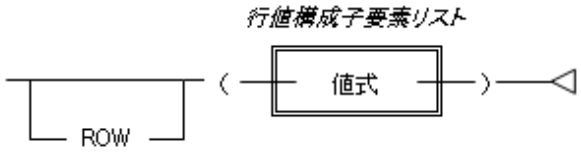
機能

順序付けられた列の並びを指定します。

記述形式



構文要素の構成



参照項番

- 値式 → “2.11 値式”

一般規則

- 行値構成子はWHERE句に指定した以下の述語に指定可能です。
  - 比較述語
  - BETWEEN述語

行値構成子要素リスト

- 行値構成子要素リストは、値式をカンマ(,)で区切って記述します。
- 行値構成子要素リストに行値構成子を指定することはできません。

使用例

例

売上表から、販売日が2011年1月から2011年3月までの売上金額の合計を求めます。

```
SELECT SUM(売上金)
    FROM 売上表
    WHERE (販売年,販売月) BETWEEN (2011,1) AND (2011,3)
```

2.11 値式

機能

値式は、値を指定します。

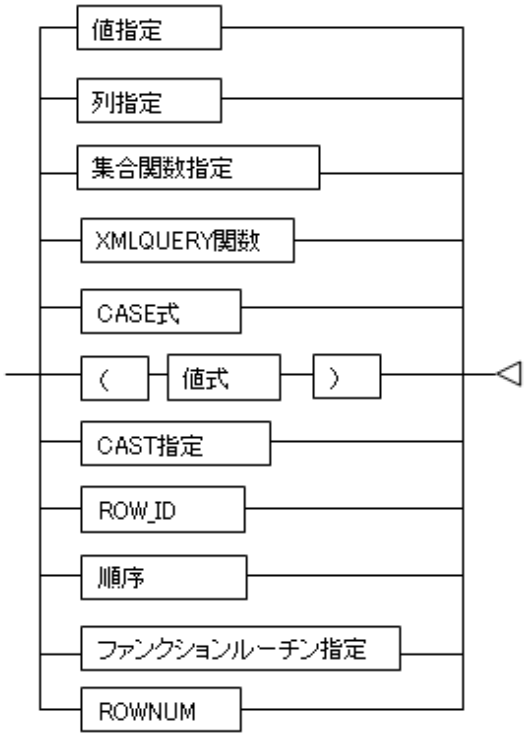
値式の種類について、以下に示します。

値式	機能
数値式	数値式は、数値を指定します。
データ列値式	データ列値式は、文字列値を指定します。
日時値式	日時値式は、日時値を指定します。
時間隔値式	時間隔値式は、時間隔値を指定します。

それぞれの値式の共通要素として値式一次子があります。

値式一次子

記述形式



参照項番

- 値指定 → “2.3 値指定と相手指定”
- 列指定 → “2.4 列指定”
- 集合関数指定 → “2.5.1 集合関数指定”
- XMLQUERY関数 → “2.5.6 XMLQUERY関数”
- CASE式 → “2.6 CASE式”
- CAST指定 → “2.7 CAST指定”
- 順序 → “2.8 順序”
- ファンクションルーチン指定→ “2.5.5 ファンクションルーチン指定”
- ROWNUM → “2.9 ROWNUM”

一般規則

- ROW_ID(行識別子)は以下の箇所にのみ指定可能です。なお、単一行SELECT文の選択リストおよび問合せ指定の選択リストは、更新可能な問合せに対してのみ指定可能です。
  - 単一行SELECT文の選択リスト
  - 問合せ指定の選択リスト(カーソル指定の問合せ式のみ)
  - UPDATE文:探索の探索条件
  - DELETE文:探索の探索条件
  - 問合せ指定のWHERE句の探索条件

使用例

例

ROW_ID(行識別子を使用してデータの更新を行います。)

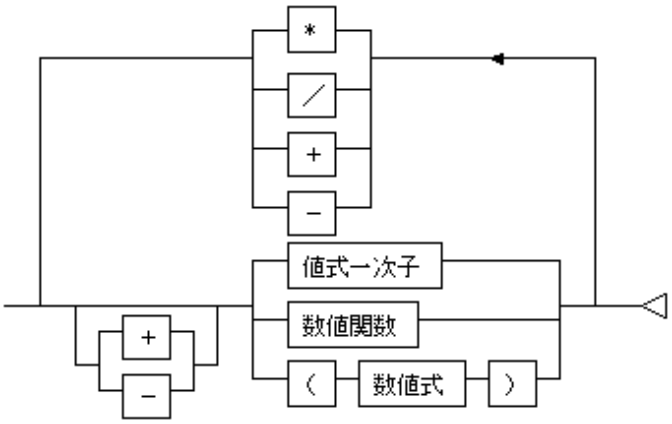
```
SELECT ROW_ID INTO :H_RID
    FROM 在庫管理.在庫表 WHERE 製品番号 = 351
                ...
UPDATE 在庫管理.在庫表
    SET 倉庫番号 = 3 WHERE ROW_ID = :H_RID
```

2.11.1 数値式

機能

数値式は、数値を指定します。

記述形式



参照項番

- 数値関数 → “2.5.2 数値関数”
- 値式一次子 → “2.11 値式”

一般規則

- 単項演算子を連続して指定することはできません。
- データ型が文字列、各国語文字列、日時型およびBLOB型の値指定、列指定、CAST指定、または集合関数は、演算子を指定することはできません。
- 値式に含まれる値指定、列指定、または集合関数の値がNULLの場合、値式の結果はNULLになります。
- 単項演算子＋は、直後に続く要素に何も作用しません。単項演算子－は、直後に続く要素の符号を反転します。単項演算子の結果のデータ型を以下に示します。

データ型	結果のデータ型
SMALLINT	INTEGER
INTEGER	
DECIMAL(p,q)	DECIMAL(p,q)
NUMERIC(p,q)	
REAL	REAL

データ型	結果のデータ型
DOUBLE PRECISION	DOUBLE PRECISION

p: 精度、q: 位取り

- 二項演算子+、-、*、/は、それぞれ、加算、減算、乗算、除算を示します。
- 値式は、括弧の中の式が最初に評価されます。括弧が指定されていない式は、単項演算子が乗除算より先に、乗除算が加減算より先に評価されます。また、同じ実行順序の式は左から右方向の順に評価されます。
- 除数が0であってはなりません。除数が0ならば、データ例外(ゼロ除算)になります。
- 二項演算子の左辺と右辺のデータ型が一致していない場合、それぞれのデータ型を変換することがあります。また、結果のデータ型は、左辺および右辺のデータ型と演算子により決定します。“表2.46 二項演算子の左辺および右辺のデータ型変換”にデータ型変換を、“表2.47 二項演算子の演算結果のデータ型”に結果のデータ型を示します。
- 値式一次子または数値関数に単項演算子+、-を付加したものを因子と呼びます。単項演算子は、省略することができます。
- 因子と二項演算子を組み合わせたものを項と呼びます。

表2.46 二項演算子の左辺および右辺のデータ型変換

変換前のデータ型		変換後のデータ型	
左辺	右辺	左辺	右辺
SMALLINT	SMALLINT	無変換	無変換
	INTEGER		
	DECIMAL(p,q)	DECIMAL(5,0)	
	NUMERIC(p,q)	DECIMAL(5,0)	DECIMAL(p,q)
	REAL	REAL	無変換
	DOUBLE PRECISION	DOUBLE PRECISION	
INTEGER	SMALLINT	無変換	無変換
	INTEGER		
	DECIMAL(p,q)	DECIMAL(10,0)	
	NUMERIC(p,q)	DECIMAL(10,0)	DECIMAL(p,q)
	REAL	REAL	無変換
	DOUBLE PRECISION	DOUBLE PRECISION	
DECIMAL(p1,q1)	SMALLINT	無変換	DECIMAL(5,0)
	INTEGER		DECIMAL(10,0)
	DECIMAL(p2,q2)		無変換
	NUMERIC(p2,q2)		DECIMAL(p2,q2)
	REAL	REAL	無変換
	DOUBLE PRECISION	DOUBLE PRECISION	
NUMERIC(p1,q1)	SMALLINT	DECIMAL(p1,q1)	DECIMAL(5,0)
	INTEGER	DECIMAL(p1,q1)	DECIMAL(10,0)
	DECIMAL(p2,q2)	DECIMAL(p1,q1)	無変換

変換前のデータ型		変換後のデータ型	
左辺	右辺	左辺	右辺
	NUMERIC(p2,q2)	DECIMAL(p1,q1)	DECIMAL(p2,q2)
	REAL	REAL	無変換
	DOUBLE PRECISION	DOUBLE PRECISION	
REAL	SMALLINT	無変換	REAL
	INTEGER		REAL
	DECIMAL(p2,q2)		REAL
	NUMERIC(p2,q2)		REAL
	REAL		無変換
	DOUBLE PRECISION		
DOUBLE PRECISION	SMALLINT	無変換	DOUBLE PRECISION
	INTEGER		DOUBLE PRECISION
	DECIMAL(p2,q2)		DOUBLE PRECISION
	NUMERIC(p2,q2)		DOUBLE PRECISION
	REAL		無変換
	DOUBLE PRECISION		

p : 精度、q : 位取り

p1: 精度、q1: 位取り

p2: 精度、q2: 位取り

表2.47 二項演算子の演算結果のデータ型

演算	変換後のデータ型		結果のデータ型
	左辺	右辺	
加減算	SMALLINT	SMALLINT	INTEGER
		INTEGER	
	INTEGER	SMALLINT	INTEGER
		INTEGER	
	DECIMAL(p1,q1)	DECIMAL(p2,q2)	DECIMAL(m,n) m: MIN(18,MAX(p1-q1,p2-q2)+MAX(q1,q2)+1) n: MAX(q1,q2)
	REAL	REAL	REAL
		DOUBLE PRECISION	DOUBLE PRECISION
	DOUBLE PRECISION	REAL	DOUBLE PRECISION

演算	変換後のデータ型		結果のデータ型
	左辺	右辺	
		DOUBLE PRECISION	
乗算	SMALLINT	SMALLINT	INTEGER
		INTEGER	
	INTEGER	SMALLINT	INTEGER
		INTEGER	
	DECIMAL(p1,q1)	DECIMAL(p2,q2)	DECIMAL(m,n) m: MIN(18,p1+p2) n: q1+q2 (注)
	REAL	REAL	REAL
		DOUBLE PRECISION	DOUBLE PRECISION
	DOUBLE PRECISION	REAL	DOUBLE PRECISION
		DOUBLE PRECISION	
除算	SMALLINT	SMALLINT	INTEGER
		INTEGER	
	INTEGER	SMALLINT	INTEGER
		INTEGER	
	DECIMAL(p1,q1)	DECIMAL(p2,q2)	DECIMAL(m,n) m: MIN(18,p1-q1+q2+MAX(5,q1)) n: MAX(5,q1)
	REAL	REAL	REAL
		DOUBLE PRECISION	DOUBLE PRECISION
	DOUBLE PRECISION	REAL	DOUBLE PRECISION
		DOUBLE PRECISION	

p1: 精度、q1: 位取り

p2: 精度、q2: 位取り

注) q1+q2>18となる場合は、エラーとなります。

DESCRIBE情報について

ー 数値式の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.48 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
二項演算子の左辺の値式	右辺の値式のデータ型
二項演算子の右辺の値式	左辺の値式のデータ型

備考. 二項演算子の両辺の値式が動的パラメタ指定の場合はエラーとなります。また、単行演算子の直後に動的パラメタ指定を指定した場合はエラーとなります。

使用例

例

数値式(列の値を四則演算する場合。仕入価格は20000、発注数量は100とします)

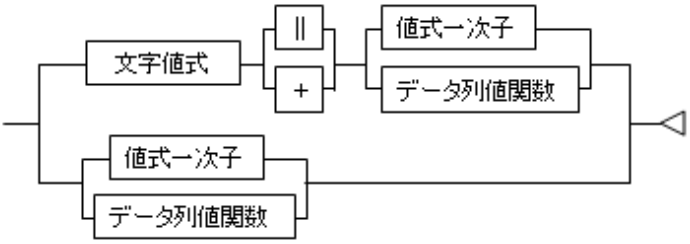
```
UPDATE 在庫管理.発注表
    SET 仕入価格 = 仕入価格 * 0.9,発注数量 = 発注数量 + 150
    WHERE 取引先 = 61
→結果は、仕入価格は18000、発注数量は250となります。
```

2.11.2 データ列値式

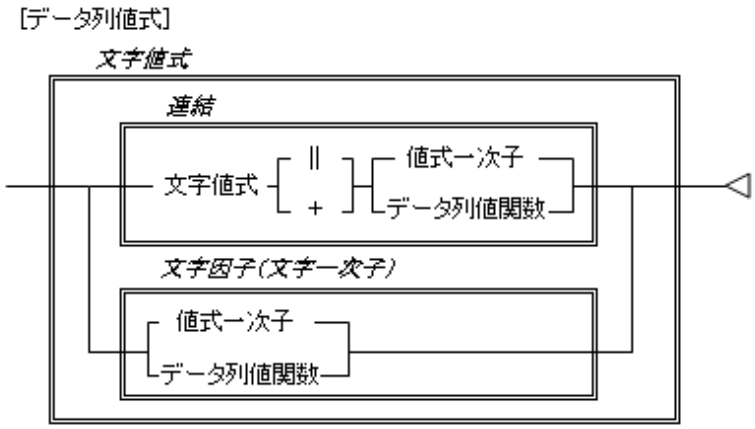
機能

データ列値式は、文字列値を指定します。

記述形式



構文要素の構成



参照項番

- データ列値関数 → “2.5.3 データ列値関数”
- 値式一次子 → “2.11 値式”

一般規則

- 文字値式は、データ列値式と同じ意味です。
- 文字因子は、データ列値式を構成する1つで値式一次子またはデータ列値関数を指定します。

- 連結の両項のデータ型は文字列型または各国語文字列型で、かつ比較可能であることが必要です。比較可能なデータ型は“表2.61 比較可能なデータ型”を参照してください。
- 連結は両項の文字列を結合します。
- いずれかの項がNULLの場合、結果はNULLになります。
- 文字列型の演算の場合の連結の結果を“表2.49 文字列型の演算の場合の連結の結果”に、また、各国語文字列型の演算の場合の連結の結果を“表2.50 各国語文字列型の演算の場合の連結の結果”に示します。

表2.49 文字列型の演算の場合の連結の結果

		右項	
		CHAR(n2)	VARCHAR(n2)
左項	CHAR(n1)	CHAR(n1+n2) Nf<n1+n2のときはエラー	VARCHAR(n) n: MIN(Nv,n1+n2)
	VARCHAR(n1)	VARCHAR(n) n: MIN(Nv,n1+n2)	

Nf: CHARのデータの最大長

Nv: VARCHARのデータの最大長

表2.50 各国語文字列型の演算の場合の連結の結果

		右項	
		NCHAR(n2)	NCHAR VARYING(n2)
左項	NCHAR(n1)	NCHAR(n1+n2) Nf<n1+n2のときはエラー	NCHAR VARYING(n) n: MIN(Nv,n1+n2)
	NCHAR VARYING(n1)	NCHAR VARYING(n) n: MIN(Nv,n1+n2)	

Nf: NCHARのデータの最大長

Nv: NCHAR VARYINGのデータの最大長

- 結果がVARCHARまたはNCHAR VARYINGのときで、連結した長さが最大長を超えた場合は、最大長以降の文字は切り捨てられます。ただし、切り捨てられる文字が空白でない場合は、エラーになります。
- 連結に動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。なお、両項が動的パラメタ指定の場合は、エラーになります。SQL記述子域のTYPEにVARCHARやNCHAR VARYINGの情報が設定されます。SQL記述子域のLENGTHにVARCHARやNCHAR VARYINGの最大文字数が設定されます。

表2.51 動的パラメタ指定が指定された場合のDESCRIBE情報

左項(または右項)のデータ型	DESCRIBE情報
CHAR(n)	VARCHAR(Nvc)
VARCHAR(n)	VARCHAR(Nvc)
NCHAR(n)	NCHAR VARYING(Nvn)
NCHAR VARYING(n)	NCHAR VARYING(Nvn)

Nvc: VARCHARのデータの最大長

Nvn: NCHAR VARYINGのデータの最大長

使用例

例1

連結演算子(C1が'ABCDE'の場合)

連結演算子として“||”を使用した場合

```
C1 || 'F'
→結果は'ABCDEF'になります。
```

連結演算子として“+”を使用した場合

```
C1 + 'F'
→結果は'ABCDEF'になります。
```

例2

連結演算子(仕入価格“20000”の先頭に¥を結合させる場合)

連結演算子として“||”を使用した場合

```
SELECT 取引製品,'¥' || CAST(仕入価格 AS CHARACTER(5))
    FROM 在庫管理.発注表 WHERE 取引先 = 61
→結果は'¥20000'になります。
```

連結演算子として“+”を使用した場合

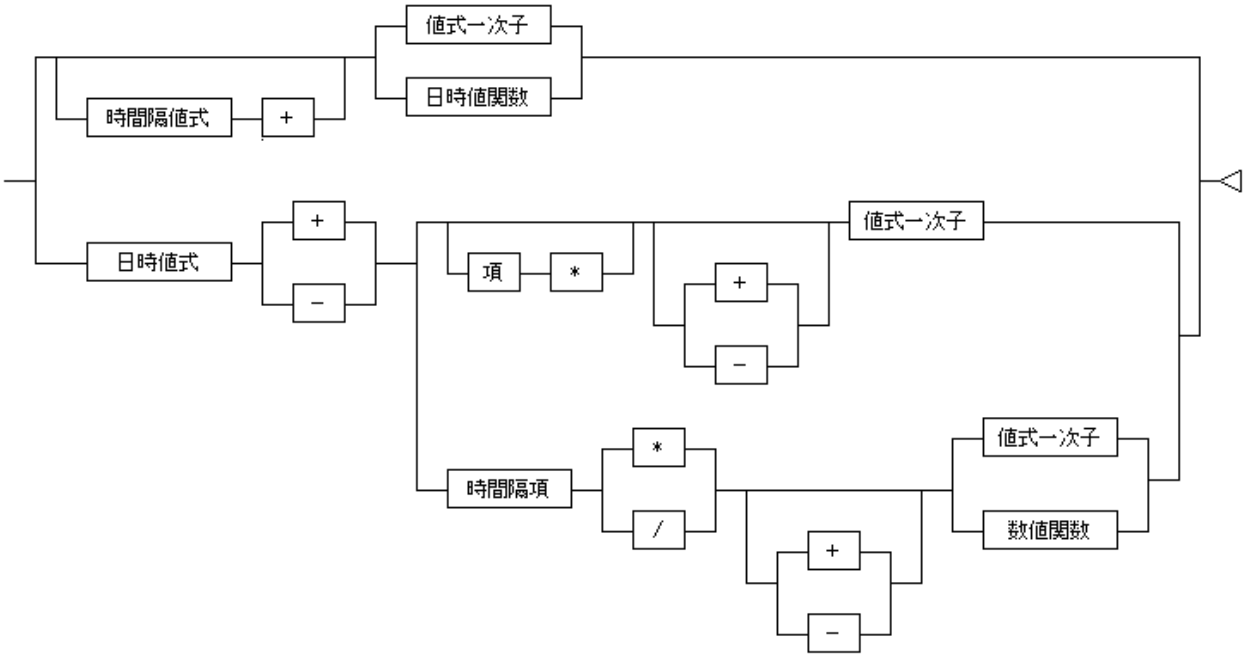
```
SELECT 取引製品,'¥'+ CAST(仕入価格 AS CHARACTER(5))
    FROM 在庫管理.発注表 WHERE 取引先 = 61
→結果は'¥20000'になります。
```

2.11.3 日時値式

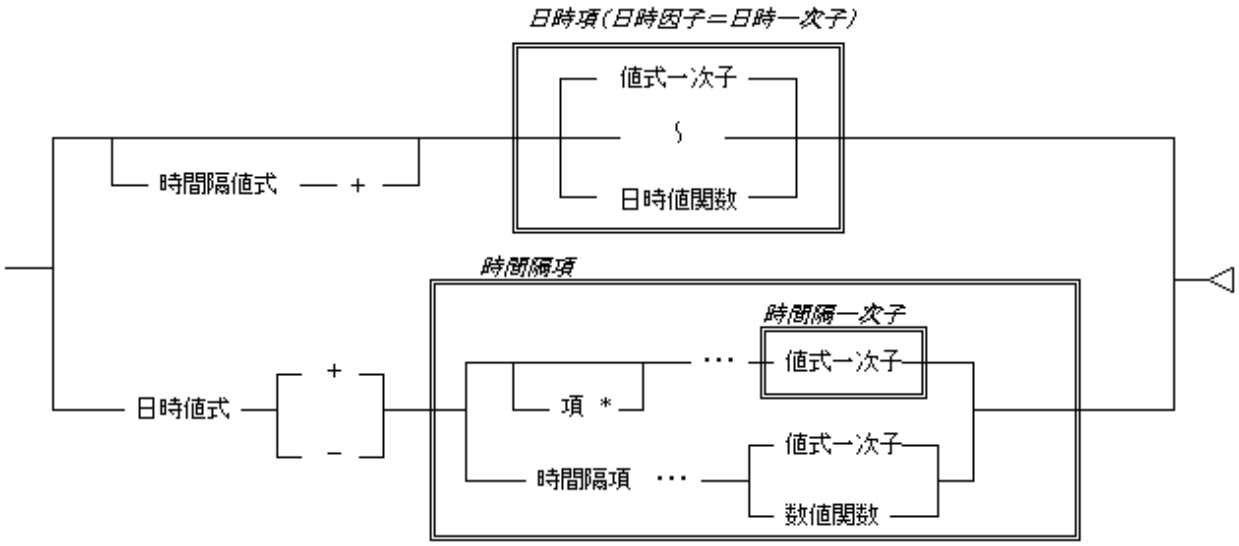
機能

日時値式は、日時値を指定します。

記述形式



構文要素の構成



参照項番

- 時間隔値式 → “2.11.4 時間隔値式”
- 数値関数 → “2.5.2 数値関数”
- 日時値関数 → “2.5.4 日時値関数”
- 値式一次子 → “2.11 値式”

一般規則

- 演算結果データ型を以下に示します。

表2.52 演算結果データ型

パターン	結果データ型
時間隔値式 ＋ 日時項	日時項に等しい
日時値式 ± 時間隔項	日時値式に等しい

- 日時演算を行う場合、時間隔項または時間隔値式は、日時項または日時値式内に含まれる日時フィールドのみを含むことが必要です。
- 日時演算に変数指定を使用する場合、CAST指定によって演算できる型にすることが必要です。
- 日時値式に含まれる任意の日時項、日時値式、時間隔値式がNULLの場合、日時値式の結果はNULLになります。
- 演算は、必要ならば各日時フィールドの桁送りを行います。
- 演算を行った結果、日時型の各日時フィールドに許される値を超えた場合は、日時フィールド溢れの例外事象になります。
- 月数の加算もしくは減算は、日時値式の月フィールドに対して行われます。演算結果が、暦上存在しない日付になった場合は、以下のように補正されます。

  ー 演算した結果が、その月に許される日数を超えた場合は、その月の最終日になります。

DESCRIBE情報について

ー 日時値式の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.53 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報	
時間隔値式 (? + 日時項の場合)	動的パラメタ指定に時間隔修飾子の指定あり	時間隔型 時間隔修飾子に従う
	動的パラメタ指定に時間隔修飾子の指定なし	INTERVAL YEAR(2) TO MONTH (注1) INTERVAL DAY(2) TO SECOND (注2) INTERVAL HOUR(2) TO SECOND (注3)
時間隔項 (日時値式 ± ? の場合)	動的パラメタ指定に時間隔修飾子の指定あり	時間隔型 時間隔修飾子に従う
	動的パラメタ指定に時間隔修飾子の指定なし	INTERVAL YEAR(2) TO MONTH (注1) INTERVAL DAY(2) TO SECOND (注2) INTERVAL HOUR(2) TO SECOND (注3)

注1) 時間隔項または時間隔値式がDATE型の場合

注2) 時間隔項または時間隔値式がTIME型の場合

注3) 時間隔項または時間隔値式がTIMESTAMP型の場合

備考. 時間隔値式 + ? または ? ± 時間隔項 の場合は時間隔値式と仮定されます。

使用例

例1

日時値式の演算1(C1がDATE'2007-10-17'の場合)

```
INTERVAL'15' DAY + C1
→結果はDATE'2007-11-01'になります。
```

例2

日時値式の演算2(C1がTIME型の場合)

```
C1 + INTERVAL'25' HOUR
→結果は桁溢れのため、エラーになります。
```

例3

日別発注表にデータを1行追加します。処理時間には現在の時間を設定します。

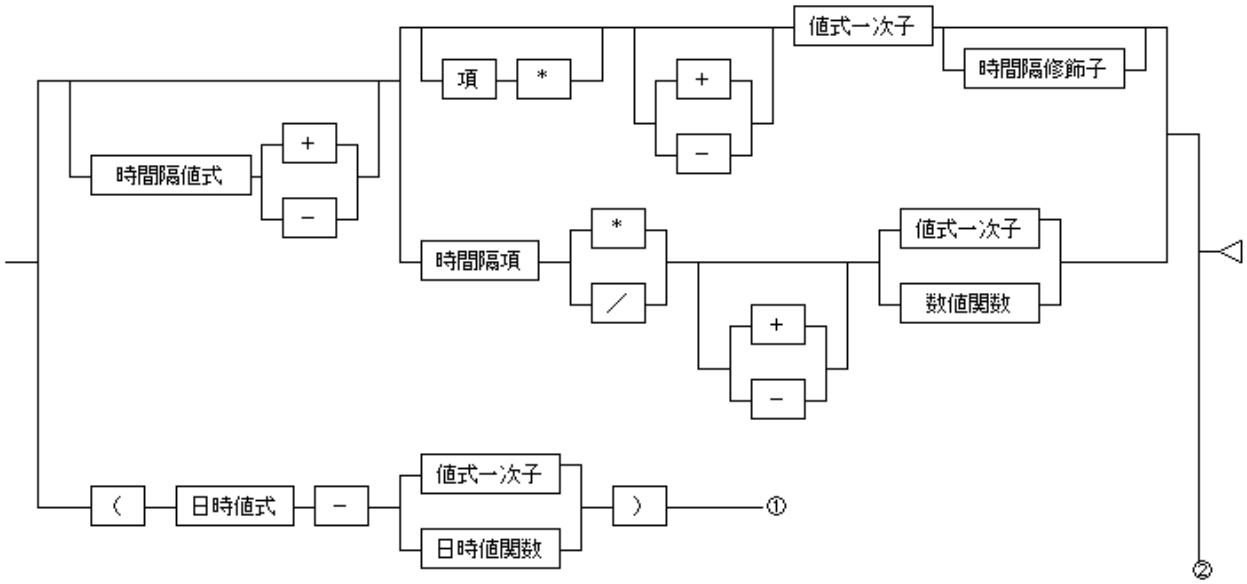
```
INSERT INTO 在庫管理.日別発注表(取引先,取引製品,発注数量,処理時間)
  VALUES(61,:PRODNO,:ORDERQTY,CURRENT_TIME)
```

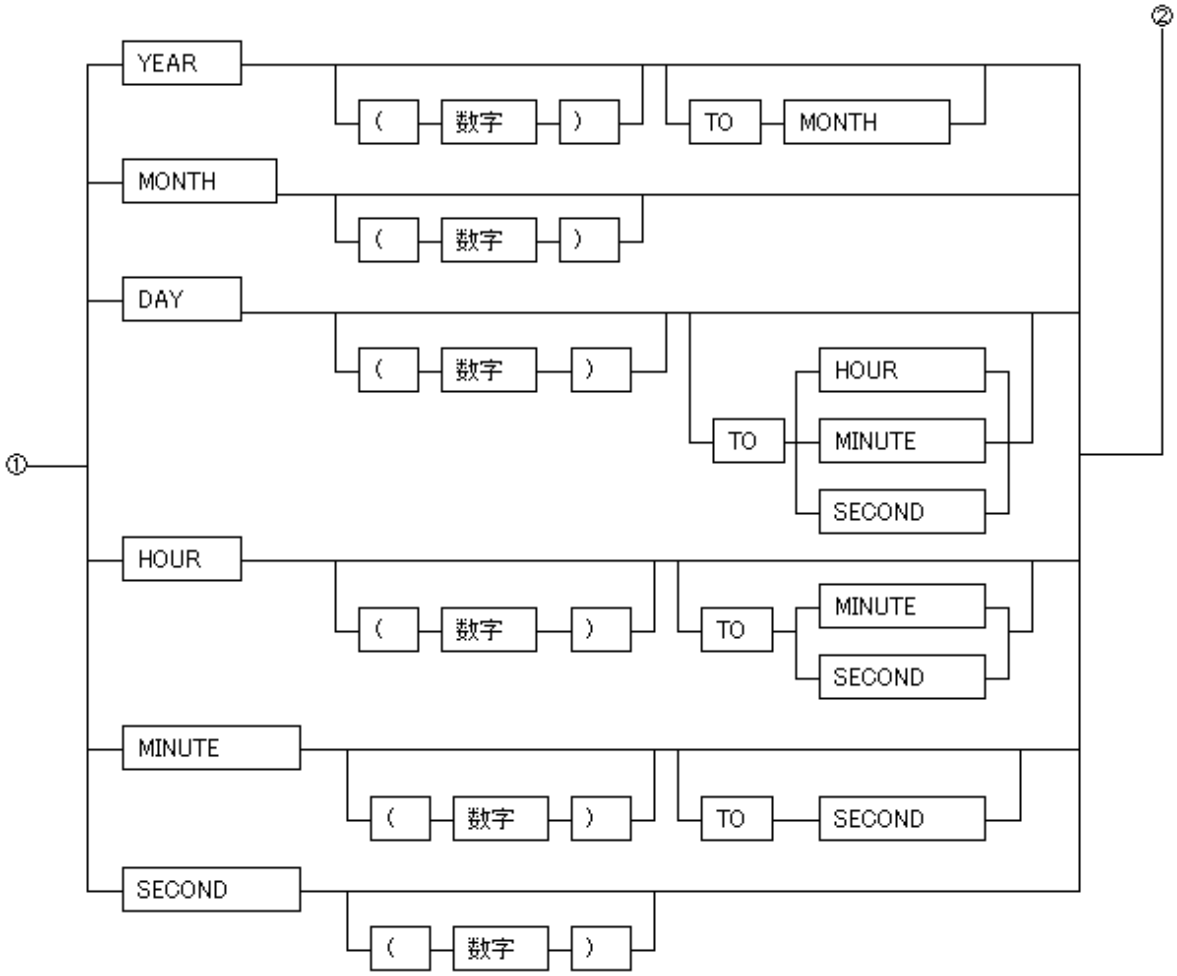
2.11.4 時間隔値式

機能

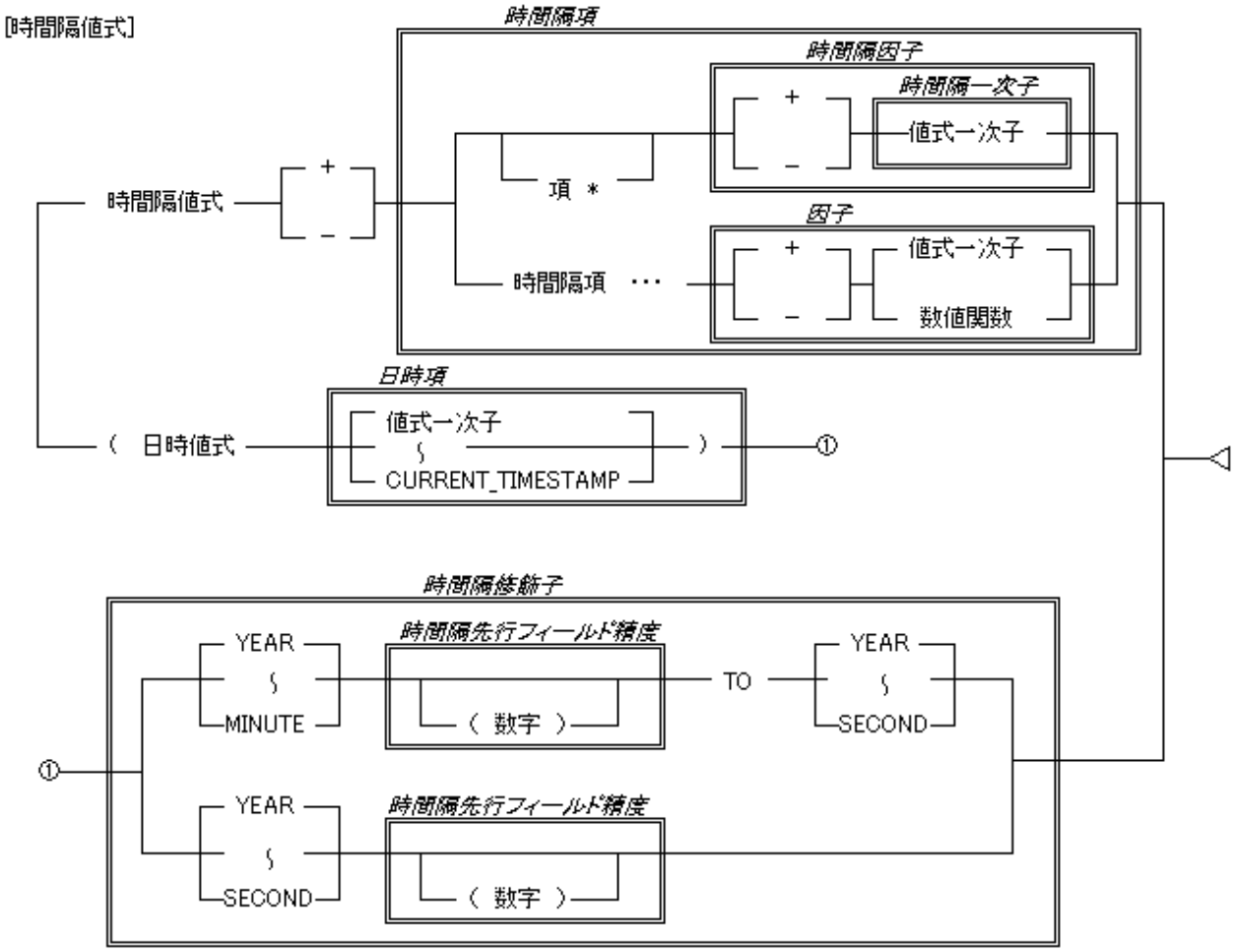
時間隔値式は、時間隔値を指定します。

記述形式




①
②
YEAR
(
数字
)
TO
MONTH
MONTH
(
数字
)
DAY
(
数字
)
TO
HOUR
MINUTE
SECOND
HOUR
(
数字
)
TO
MINUTE
SECOND
MINUTE
(
数字
)
TO
SECOND
SECOND
(
数字
)

構文要素の構成



参照項番

- 数値関数 → “2.5.2 数値関数”
- 日時値式 → “2.11.3 日時値式”
- 日時値関数 → “2.5.4 日時値関数”
- 数字 → “2.1.1 文字”
- 値式一次子 → “2.11 値式”

一般規則

- 演算結果データ型を以下に示します。

表2.54 演算結果データ型

パターン	結果データ型
時間隔値式 ± 時間隔項	時間隔型 演算する時間隔の最上位日時フィールド～最下位日時フィールドを持つ(注)
日時値式 - 日時項	時間隔型 時間隔修飾子に従う
時間隔項 * (/) 因子	時間隔型 時間隔項側の日時フィールドを持つ(注)
項 * 時間隔因子	時間隔型 時間隔因子側の日時フィールドを持つ(注)

注) 演算結果の有効桁を失わないような時間隔先行フィールドを持つ

- 時間隔値式±時間隔項を行う場合、年月クラスと年月クラスどうし、または、日時クラスと日時クラスどうしの演算のみ可能です。年月クラスと日時クラスの演算は行えません。
- 値式一次子が動的パラメタ指定の場合は、値式一次子の直後に時間隔修飾子を指定することができます。
- 日時値式ー日時項を行う場合、日時値式および日時項は比較可能であることが必要です。比較可能なデータ型は"表2.61 比較可能なデータ型"を参照してください。
- 時間隔演算に変数指定を使用する場合、CASTによるデータ型変換が必要です。
- 時間隔値式に含まれる任意の時間隔一次子、日時値式、日時項、因子がNULLの場合、時間隔値式の結果はNULLになります。
- 日時値式ー日時項を行う場合、時間隔値式の結果は、時間隔修飾子で指定された開始フィールドから終了フィールドを持つ時間隔型に変換されます。このとき、時間隔修飾子で指定した終了フィールドよりも下位の日時フィールドは、切り捨てられます。
- 時間隔値式±時間隔項、時間隔項* (/)因子および項*時間隔因子を行う場合の時間隔値式の結果の時間隔先行フィールド精度を以下に示します。

表2.55 時間隔値式の結果の時間隔先行フィールド精度

演算	左辺	右辺	時間隔先行フィールド精度
加減算	時間隔値式	時間隔項	MIN(9,MAX(p)+1)
乗算	時間隔項 因子 項	因子 時間隔項 時間隔因子	MIN(9,p + 因子の精度 - 因子の位取り)
除算	時間隔項	因子	MIN(9,p + 因子の位取り)

p: 時間隔先行フィールド精度

(左辺、右辺をそれぞれ演算結果の日時フィールドを持つ時間隔に変換した後の時間隔先行フィールド精度(1≦n≦9))

- 時間隔値式±時間隔項(年月型)の演算結果の時間隔型を以下に示します。

表2.56 時間隔値式±時間隔項(年月型)の演算結果の時間隔型

		時間隔項		
		YEAR(P2)	YEAR(p2) TO MONTH	MONTH(p2)
時間隔値式	YEAR(p1)	YEAR(p) p: MIN(9,MAX(p1,p2)+1)	YEAR(p) TO MONTH p: MIN(9,MAX(p1,p2)+1)	YEAR(p) TO MONTH p: MIN(9,MAX(p1+1,p2-1))
	YEAR(p1) TO MONTH	YEAR(p) TO MONTH p: MIN(9,MAX(p1,p2)+1)	YEAR(p) TO MONTH p: MIN(9,MAX(p1,p2)+1)	YEAR(p) TO MONTH p: MIN(9,MAX(p1+1,p2-1))
	MONTH(p1)	YEAR(p) TO MONTH p: MIN(9,MAX(p1-1,p2+1))	YEAR(p) TO MONTH p: MIN(9,MAX(p1-1,p2+1))	MONTH(p) p: MIN(9,MAX(p1,p2)+1)

p、p1、p2: 時間隔先行フィールド精度

・ 時間隔値式±時間隔項(日時型)の演算結果の時間隔型を以下に示します。

表2.57 時間隔値式±時間隔項(日時型)の演算結果の時間隔型

		時間隔項			
		DAY(p2) TO E2	HOUR(p2) TO E2	MINUTE(p2) TO E2	SECOND(p2)
時間隔値式	DAY(p1) TO E1	DAY(p) TO E p: MIN(9, MAX(p1,p2)+1) E: MIN(E1,E2)	DAY(p) TO E p: MIN(9, MAX(p1+1,p2-1)) E: MIN(E1,E2)	DAY(p) TO E p: MIN(9, MAX(p1+1,p2-3)) E: MIN(E1,E2)	DAY(p) TO SECOND p: MIN(9, MAX(p1+1,p2-5))
	HOUR(p1) TO E1	DAY(p) TO E p: MIN(9, MAX(p1-1,p2+1)) E: MIN(E1,E2)	HOUR(p) TO E p: MIN(9, MAX(p1,p2)+1) E: MIN(E1,E2)	HOUR(p) TO E p: MIN(9, MAX(p1+1,p2-1)) E: MIN(E1,E2)	HOUR(p) TO SECOND p: MIN(9, MAX(p1+1,p2-3))
	MINUTE(p1) TO E1	DAY(p) TO E p: MIN(9, MAX(p1-3,p2+1)) E: MIN(E1,E2)	HOUR(p) TO E p: MIN(9, MAX(p1-1,p2+1)) E: MIN(E1,E2)	MINUTE(p) TO E p: MIN(9, MAX(p1,p2)+1) E: MIN(E1,E2)	MINUTE(p) TO SECOND p: MIN(9, MAX(p1+1,p2-1))
	SECOND(p1)	DAY(p) TO SECOND p: MIN(9, MAX(p1-5,p2+1))	HOUR(p) TO SECOND p: MIN(9, MAX(p1-3,p2+1))	MINUTE(p) TO SECOND p: MIN(9, MAX(p1-1,p2+1))	SECOND(p) p: MIN(9, MAX(p1,p2)+1)

p、p1、p2 : 時間隔先行フィールド精度

E、E1、E2 : 日時フィールド

なお、日時フィールドには以下の大小関係があります。

DAY > HOUR > MINUTE > SECOND

例

```
MIN(DAY,MINUTE) = MINUTE
```

・ 時間隔項*(/)因子および項*時間隔因子の演算結果の時間隔型を以下に示します。

表2.58 時間隔項*(/)因子および項*時間隔因子の演算結果の時間隔型

		因子または項					
		SMALLINT (注1)	INTEGER (注2)	NUMERIC(p2,q2)	DECIMAL(p2,q2)	REAL	DOUBLE PRECISION
時間隔項または時間隔因子	YEAR(p1) TO E	YEAR(p) TO E　乗算 p: MIN(9,p1+p2)　除算 p: MIN(9,p1+q2)				YEAR(9) TO E	
	MONTH(p1)	MONTH(p)　乗算 p: MIN(9,p1+p2)　除算 p: MIN(9,p1+q2)				MONTH(9)	
	DAY(p1) TO E	DAY(p) TO E　乗算 p: MIN(9,p1+p2)　除算 p: MIN(9,p1+q2)				DAY(9) TO E	
	HOUR(p1) TO E	HOUR(p) TO E　乗算 p: MIN(9,p1+p2)　除算 p: MIN(9,p1+q2)				HOUR(9) TO E	
	MINUTE(p1) TO E	MINUTE(p) TO E　乗算 p: MIN(9,p1+p2)　除算 p: MIN(9,p1+q2)				MINUTE(9) TO E	
	SECOND(p1)	SECOND(p)　乗算 p: MIN(9,p1+p2)　除算 p: MIN(9,p1+q2)				SECOND(9)	

p、p1: 時間隔先行フィールド精度

p2 : 精度

q2 : 位取り

E : 日時フィールド

注1) 因子がSMALLINTの場合、p2=5、q2=0を表します。

注2) 因子がINTEGERの場合、p2=10、q2=0を表します。

DESCRIBE情報について

ー 時間隔値式の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.59 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報	
時間隔値式 (? ± 時間隔項 の場合)	動的パラメタ指定に時間隔修飾子の指定あり	時間隔型 時間隔修飾子に従う
	動的パラメタ指定に時間隔修飾子の指定なし	時間隔項のデータ型
時間隔項 (時間隔値式 ± ? の場合)	動的パラメタ指定に時間隔修飾子の指定あり	時間隔型 時間隔修飾子に従う
	動的パラメタ指定に時間隔修飾子の指定なし	時間隔値式のデータ型
因子 (時間隔項 *(/) ? の場合)	INTEGER	
項 (? * 時間隔因子 の場合)	INTEGER	

備考1. 日時値式 - ? および ? - 日時項 の場合はエラーとなります。

備考2. ? *(/) 因子 または 項 * ? の場合は数値式と仮定されます。

使用例

例1

時間隔値式の演算1(時間隔定数間の減算の場合)

```
INTERVAL '20' YEAR - INTERVAL '13' MONTH
→結果は、INTERVAL'18-11'YEAR(3) TO MONTHになります。
```

例2

時間隔値式の演算2(時間隔定数と数定数の乗算の場合)

```
INTERVAL '4' MONTH * 2
→結果は、INTERVAL'8'MONTH(7)になります。
```

例3

時間隔値式の演算3(時間隔型の列“記録”を時間隔値式として選択リストに指定した場合)

```
SELECT チーム名,記録 - MIN(記録) FROM 駅伝記録表
    WHERE チーム名='GRP01' GROUP BY 区間
→結果は、チーム名='GRP01'の区間最高記録とのタイム差を検索します。
```

2.12 述語

機能

述語には、“真”、“偽”または“不定”の真偽値の評価をするための条件を指定します。

述語の種類について、以下に示します。

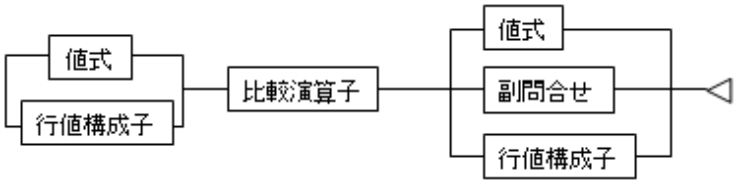
述語	機能
比較述語	2つの値についての比較を指定します。
BETWEEN述語	範囲比較を指定します。
IN述語	ある限定された値の集合について比較を指定します。
LIKE述語	文字型データについて、照合比較を指定します。
NULL述語	NULL値との比較を指定します。
限定述語	限定された値の集合との比較を指定します。
EXISTS述語	空集合との比較を指定します。
XMLEXISTS述語	XQuery式に指定した検索式に一致するXMLデータであるかを評価します。

2.12.1 比較述語

機能

比較述語は、2つの値についての比較を指定します。

記述形式



参照項番

- 値式 → “2.11 値式”
- 副問合せ → “2.14 副問合せ”
- 行値構成子→“2.10 行値構成子”

一般規則

比較演算子

ー 比較述語で指定する比較演算子の種類を以下に示します。

表2.60 比較演算子

比較演算子	機能(注)
＝	“＝”の左側の値式と右側の値式が等しい場合に真となり、等しくない場合に偽となります。
＜	“＜”の左側の値式が右側の値式より小さい場合に真となり、大きいまたは等しい場合に偽となります。
＜＝ ！＞	“＜＝”の左側の値式が右側の値式より小さいまたは等しい場合に真となり、大きい場合に偽となります。 “！＞”は“＜＝”と同じ意味です。
＞	“＞”の左側の値式が右側の値式より大きい場合に真となり、小さいまたは等しい場合に偽となります。
＞＝ ！＜	“＞＝”の左側の値式が右側の値式より大きいまたは等しい場合に真となり、小さい場合に偽となります。 “！＜”は“＞＝”と同じ意味です。
＜＞ ！＝	“＜＞”の左側の値式と右側の値式が等しくない場合に真となり、等しい場合に述語は偽となります。 “！＝”は“＜＞”と同じ意味です。

注) 比較演算子の左側または右側の値式がNULL値の場合、比較述語の結果は不定になります。

ー 比較演算子の左右の値式の対は、比較可能であることが必要です。比較可能なデータ型を以下に示します。なお、BLOB型およびROW_IDはほかの属性と比較することはできません。また、BLOB型同士の比較もできません。

表2.61 比較可能なデータ型

		右辺							
		文字列型	各国語文字列型	真数型・概数型	DATE型	TIME型	TIMESTAMP型	INTERVAL型(年月)	INTERVAL型(日時)
左辺	文字列型	○	×	×	○ (注1)	○ (注2)	○ (注3)	○ (注4)	○ (注5)
	各国語文	×	○	×	×	×	×	×	×

		右辺							
		文字列型	各国語文字列型	真数型・概数型	DATE型	TIME型	TIMESTAMP型	INTERVAL型(年月)	INTERVAL型(日時)
	字列型								
	真数型・概数型	×	×	○	×	×	×	○ (注6)	○ (注7)
	DATE型	○ (注1)	×	×	○	×	×	×	×
	TIME型	○ (注2)	×	×	×	○	×	×	×
	TIMESTAMP型	○ (注3)	×	×	×	×	○	×	×
	INTERVAL型(年月)	○ (注4)	×	○ (注6)	×	×	×	○	×
	INTERVAL型(日時)	○ (注5)	×	○ (注7)	×	×	×	×	○

○: 比較可能

×: 比較不可

注1) 比較相手が変数指定の場合のみ可能です。このとき変数の内容は日付列であることが必要です。

注2) 比較相手が変数指定の場合のみ可能です。このとき変数の内容は時刻列であることが必要です。

注3) 比較相手が変数指定の場合のみ可能です。このとき変数の内容は時刻印列であることが必要です。

注4) 比較相手が変数指定の場合のみ可能です。このとき変数の内容は比較元と同じ精度を持つ年月クラスの時間隔定数であることが必要です。

注5) 比較相手が変数指定の場合のみ可能です。このとき変数の内容は比較元と同じ精度を持つ日時クラスの時間隔定数であることが必要です。

注6) 比較相手が真数(位取り0)の変数指定の場合のみ可能です。このとき変数の内容は比較元と同じ単一日時フィールドからなる年月クラスの時間隔定数であることが必要です。

注7) 比較相手が真数(位取り0)の変数指定の場合のみ可能です。このとき変数の内容は比較元と同じ単一日時フィールドからなる日時クラスの時間隔定数であることが必要です。

- 値式がNULL値または副問合せの結果が空ならば、比較の結果は不定となります。
- 比較演算子の左辺と右辺がNULLでない場合、以下の条件が成立すれば真、成立しなければ偽となります。

A = B:

AとBは等しい

A < B:

AはBより小さい

A <= B:

AはBと等しいか、AはBより小さい

A > B:

AはBより大きい

A >= B:

AはBと等しいか、AはBより大きい

A <> B:

AはBに等しくない

- 値式のデータ型が文字列型、または各国語文字列型ならば、そのデータ型を取り扱う文字コード系(EUCコード、シフトJISコードまたはUNICODE)に基づいて表現した時の数値で、左から右の順に比較されます。長さが異なる文字列の比較は、短い文字列に空白を補い、長さを同じにして比較します。以下に例を示します。

表2.62 文字列型または各国語文字列型の比較の例

比較述語	実行される比較	結果
'AB'='ABbb'	'ABbb'='ABbb'	真
'AB'='ABCD'	'ABbb'='ABCD'	偽
'AB'>'AAAA'	'ABbb'>'AAAA'	真
'A'>'Abbb'	'Abbb'>'Abbb'	偽

b: 空白を示します。

副問合せ

- 副問合せを指定する場合、副問合せの結果は1行であることが必要です。

値式

- 値式に行識別子を指定した場合の記述形式は以下のとおりです。
  - 比較演算子は“＝”のみが指定可能です。
  - もう一方の値式には埋込み変数名または動的パラメタ指定のみが指定可能です。副問合せは指定できません。
  - 埋込み変数名はROW_IDと対応する変数であることが必要です。データ型と対応する変数定義については、“表6.1 SQLのデータ型と対応するC変数定義”および“表6.3 SQLのデータ型と対応するCOBOL変数定義”を参照してください。
- 値式に行識別子を指定した比較述語は、以下の探索条件にのみ指定可能です。
  - UPDATE文:探索の探索条件
  - DELETE文:探索の探索条件
  - 問合せ指定のWHERE句の探索条件
- 値式に行識別子を指定した比較述語は、副問合せには指定できません。
- 値式に行識別子を指定した比較述語を問合せ指定のWHERE句に指定した場合、問合せ指定は更新可能であることが必要です。
- 値式に行識別子を指定した場合、探索条件にブール演算子およびNOTを指定することはできません。

行値構成子

- 比較演算子の左右の行値構成子要素リストの値式の数は同じであることが必要です。
- 行値構成子を指定した比較述語は、ブール演算子を用いた比較述語と同じ意味になります。したがって、行値構成子を指定した場合の比較結果や比較可能なデータ型はブール演算子を用いた場合と同じになります。ブール演算子を用いた形式を以下に示します。

表2.63 行値構成子を用いた記述形式とブール演算子を用いた記述形式

比較演算子	行値構成子を用いた記述形式	ブール演算子を用いた記述形式
＝	(X1,X2,…,Xn)=(Y1,Y2,…,Yn)	X1=Y1 AND X2=Y2 AND…AND Xn=Yn

比較演算子	行値構成子を用いた記述形式	ブール演算子を用いた記述形式
＜	(X1,X2,…,Xn)<(Y1,Y2,…,Yn)	X1<Y1 OR (X1=Y1 AND X2<Y2) OR … OR (X1=Y1 AND X2=Y2 AND … AND Xn-1=Yn-1 AND Xn<Yn)
＜＝	(X1,X2,…,Xn)<=(Y1,Y2,…,Yn)	X1<Y1 OR (X1=Y1 AND X2<Y2) OR … OR (X1=Y1 AND X2=Y2 AND … AND Xn-1=Yn-1 AND Xn<=Yn)
＞	(X1,X2,…,Xn)>(Y1,Y2,…,Yn)	X1>Y1 OR (X1=Y1 AND X2>Y2) OR … OR (X1=Y1 AND X2=Y2 AND … AND Xn-1=Yn-1 AND Xn>Yn)
＞＝	(X1,X2,…,Xn)>=(Y1,Y2,…,Yn)	X1>Y1 OR (X1=Y1 AND X2>Y2) OR … OR (X1=Y1 AND X2=Y2 AND … AND Xn-1=Yn-1 AND Xn>=Yn)
＜＞	(X1,X2,…,Xn)<>(Y1,Y2,…,Yn)	X1<>Y1 OR X2<>Y2 OR … OR Xn<>Yn

DESCRIBE情報について

ー 比較述語の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.64 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
比較演算子の左側の値式	右側の値式のデータ型
比較演算子の右側の値式	左側の値式のデータ型

備考. 比較演算子の両側の値式が動的パラメタ指定の場合はエラーとなります。

使用例

表Tについて検索条件を与えます。

表T

	C1	C2	C3	C4
R1	10	15	30	JL
R2	15	20	30	CX
R3	10	25	20	PA
R4	10	20	70	QF
R5	20	25	50	TA
R6	20	10	60	CA

例1

列C1の値が15の行を検索します。

```
WHERE C1 = 15
→結果として、行R2が得られます。
```

例2

一方の辺が副問合せの場合

```
WHERE C1 = (SELECT MIN(C2) FROM T)
→結果として、行R1、R3、およびR4が得られます。
```

例3

副問合せの誤りの例

```
WHERE C1 >= (SELECT C2 FROM T WHERE C3 = 30)
→副問合せのC2の結果が15と20の2個なのでエラーとなります。
```

例4

列C1、C2の並びが20、10以上の行を検索します。

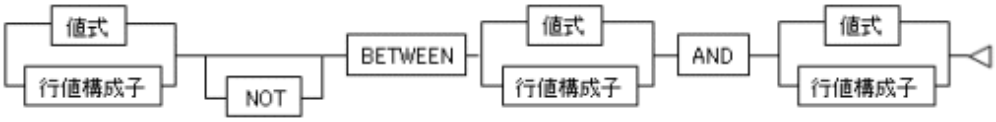
```
WHERE (C1,C2) >= (20,10)
→結果として、行R5、R6が得られます。
```

2.12.2 BETWEEN述語

機能

BETWEEN述語は、範囲比較を指定します。

記述形式



参照項番

- 値式 → “2.11 値式”
- 行値構成子→“2.10 行値構成子”

一般規則

- BETWEEN述語で指定する3つの値式のデータ型は、比較可能であることが必要です。比較可能なデータ型は“表2.61 比較可能なデータ型”を参照してください。
- BETWEEN述語で“x BETWEEN y AND z”と記述したものは、比較述語で“x>=y AND x<=z”と記述したのと同じ意味です。
- “x NOT BETWEEN y AND z”と記述したものは、“NOT (x BETWEEN y AND z)”と記述したのと同じ意味です。また、比較述語で“x<y OR x>z”と記述したのと同じ意味です。
- オペランドの一つに行値構成子を指定する場合、残りのオペランドも行値構成子でなければなりません。

- オペランドに行値構成子を指定した場合、3つのオペランドに指定した行値構成子要素リストの値式の数は、同じであることが必要です。
- 行値構成子要素リストに指定した値式のデータ型は、対応する他のオペランドの行値構成子要素リストに指定した値式のデータ型と、比較可能であることが必要です。比較可能なデータ型は“表2.61 比較可能なデータ型”を参照してください。

DESCRIBE情報について

ー BETWEEN述語の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.65 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
1番目の値式	2番目の値式のデータ型
2番目の値式	1番目の値式のデータ型
3番目の値式	1番目の値式のデータ型

備考. 1番目の値式が動的パラメタ指定で、2番目または3番目の値式が動的パラメタ指定の場合は、エラーとなります。

使用例

例

“比較述語”の使用例で示した表Tに対して、列C2の値が10以上かつ15以下の行を検索します。

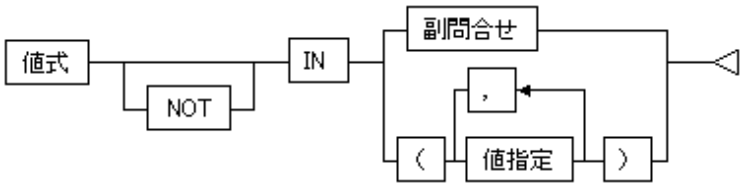
```
SELECT C1 FROM T WHERE C2 BETWEEN 10 AND 15
→この結果として、行R1とR6が選択されます。
```

2.12.3 IN述語

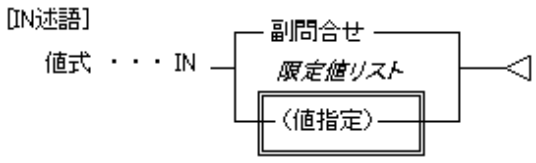
機能

IN述語は、ある限定された値の集合について比較を指定します。

記述形式



構文要素の構成



参照項番

- 値指定 → “2.3 値指定と相手指定”
- 副問合せ → “2.14 副問合せ”

一般規則

- 限定値リストは、値指定をカンマ(,)で区切って記述します。

- 値式のデータ型と、副問合せまたは限定値リストの中のすべての値指定のデータ型は、比較可能であることが必要です。比較可能なデータ型は“表2.61 比較可能なデータ型”を参照してください。
- “X NOT IN S”と記述したものは、“NOT(X IN S)”と記述したのと同じ意味です。また、Sが副問合せのとき、“X IN S”と記述したものは、限定述語の“X=ANY S”と記述したのと同じです。
- 値式の値が、副問合せの結果または限定値リストの集合の中に1つでも存在する場合、IN述語の結果は真になります。1つも存在しない場合、偽になります。値式がNULLの場合、および副問合せの結果または限定値リストの値が1つでもNULLの場合、不定となります。IN述語の結果については“表2.70 限定子ANYまたはSOMEを指定した場合の限定述語の結果”を参照してください。

DESCRIBE情報について

ー IN述語の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.66 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報	
値式	副問合せの場合	副問合せのデータ型
	限定値リストの場合	限定値リストの1番目の値指定のデータ型
限定値リスト	値式が動的パラメタ指定でない場合	値式のデータ型
	値式が動的パラメタ指定の場合	限定値リストの1番目の値指定のデータ型

備考. 限定値リストの1番目の値指定が動的パラメタ指定の場合はエラーとなります。

使用例

例

“比較述語”の使用例で示した表Tに対して、列C3の値が20、40または60の行を検索します。

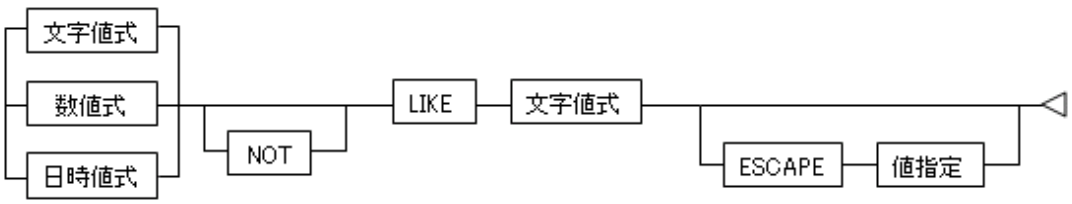
```
SELECT C1 FROM T
    WHERE C3 IN (20,40,60)
→この結果として、行R3とR6が選択されます。なお、列C3の値が40の行は
  存在しないため検索結果は2行となります。
```

2.12.4 LIKE述語

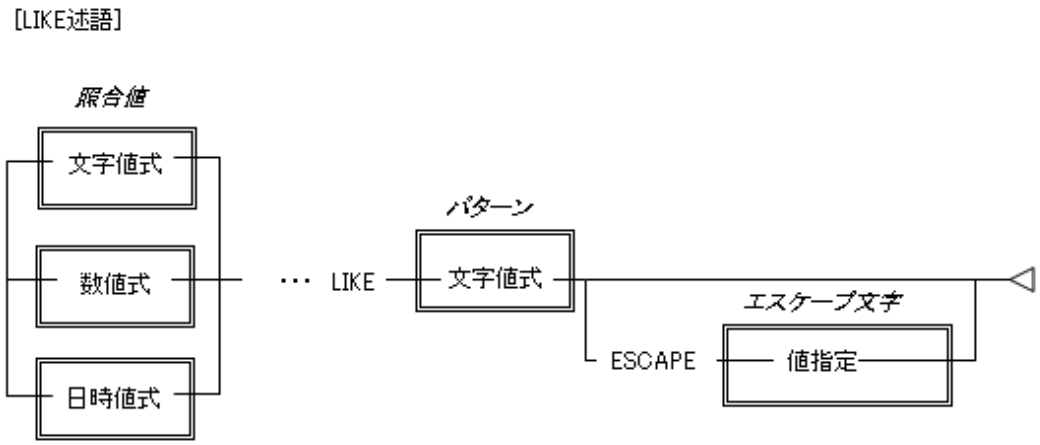
機能

LIKE述語は、文字型データについて、照合比較を指定します。

記述形式



構文要素の構成



参照項番

- 値指定 → “2.3 値指定と相手指定”
- 文字値式 → “2.11.2 データ列値式”
- 数値式 → “2.11.1 数値式”
- 日時値式 → “2.11.3 日時値式”

一般規則

- 照合値のデータ型と、パターンのデータ型、およびエスケープ文字のデータ型は比較可能であることが必要です。比較可能なデータ型は“表2.61 比較可能なデータ型”を参照してください。なお、照合値が真数型、概数型または日時型の場合、文字列型に暗黙的な型変換を行います。
- 照合値およびパターンの値がNULLの場合、LIKE述語の結果は不定になります。照合値の値とパターンが一致する場合は真になり、一致しない場合は偽になります。
- “x NOT LIKE y”と記述したものは、“NOT(x LIKE y)”と記述したのと同じ意味です。
- 照合値が数値データで、パターンに0を含んだ文字列データが指定された場合、LIKE述語の結果は偽になります。パターンには、照合値に指定した数値データと同じ形式の文字列データを指定する必要があります。

  例

  数値型の列“従業員番号”に、1が設定されているとします。
  従業員番号 LIKE '01'
  → 結果は偽になります。
  従業員番号 LIKE '1.00'
  → 結果は偽になります。

- 照合値が符号なしの数値データで、パターンに符号を含んだ文字列データが指定された場合、LIKE述語の結果は偽になります。パターンには、照合値に指定した数値データと同じ形式の文字列データを指定する必要があります。

  例

  数値型の列“従業員番号”に、1が設定されているとします。
  従業員番号 LIKE '1'
  → 結果は真になります。
  従業員番号 LIKE '+1'
  → 結果は偽になります。

・ 照合値が日時値データで、パターンに日時が1桁の文字列データが指定された場合、LIKE述語の結果は偽になります。パターンには、照合値に指定した日時値データを、年は4桁、月、日、時、分および秒は2桁の形式の文字列データで指定する必要があります。

例

日時型の列“営業日”に、DATE'2007-1-2'が設定されているとします。
営業日 LIKE '2007-01-02'
→ 結果は真になります。
営業日 LIKE '2007-1-2'
→ 結果は偽になります。

照合値

― 照合値のデータ型は、文字列型、各国語文字列型、真数型、概数型または日時型であることが必要です。

― 照合値の文字列または各国語文字列の中で、空白はデータの一部として照合比較の対象となります。

パターン

― 照合値が文字列型、真数型、概数型または日時型の場合、パターンは照合比較する文字列を指定します。文字列中で、特殊文字のパーセント記号文字“%”と下線文字“_”、およびエスケープ文字は、照合値との比較において以下のように扱われます。

- パーセント記号文字は、照合値のデータ中で0個以上の任意の文字の並びに対応します。
- 下線文字は、照合値のデータ中で1個の任意の文字に対応します。
- パーセント記号文字および下線文字を任意の文字でなく、その文字として扱う場合には、これらの文字の直前にエスケープ文字を指定します。

― 照合値が各国語文字列型の場合、パターンは照合比較する各国語文字列を指定します。各国語文字のパーセント記号“**%**”と下線文字“＿”、およびエスケープ文字の扱いは、文字列型の場合と同じです。

― パターンの文字列または各国語文字列の中で、空白はデータの一部として照合比較の対象となります。

エスケープ文字

― エスケープ文字は、1文字の文字列または各国語文字列で指定します。

― エスケープ文字には、NULLは指定できません。

DESCRIBE情報について

― LIKE述語の各オペランドに動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.67 動的パラメタ指定が指定された場合のDESCRIBE情報

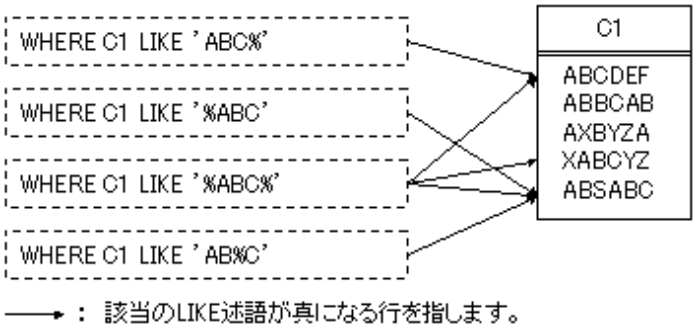
オペランド	DESCRIBE情報
パターン	照合値のデータ型が各国語文字列型の場合は各国語文字列型、それ以外の場合は文字列型
エスケープ文字	照合値のデータ型が各国語文字列型の場合は各国語文字列型、それ以外の場合は文字列型

備考. 照合値が動的パラメタ指定の場合はエラーとなります。

使用例

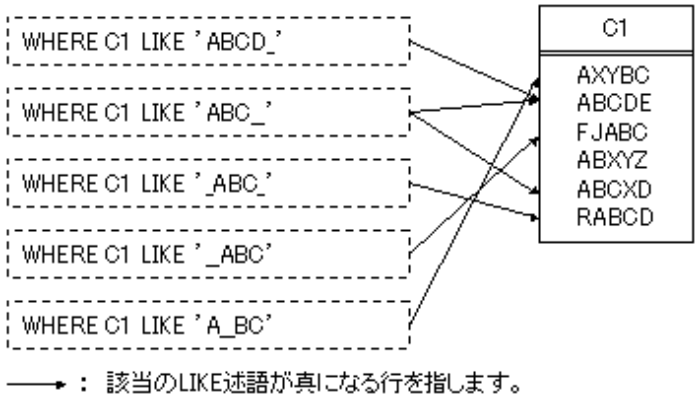
例1

パーセント記号文字をパターン中に指定する例です。



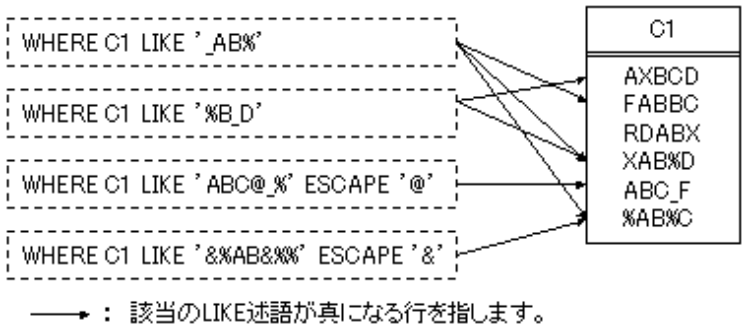
例2

下線文字をパターン中に指定する例です。



例3

エスケープ文字をパターン中に指定する例です。

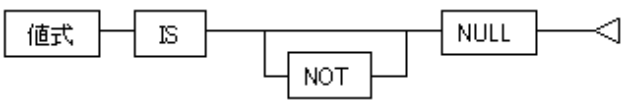


2.12.5 NULL述語

機能

NULL述語は、NULL値との比較を指定します。

記述形式



参照項番

- 値式 → “2.11 値式”

一般規則

・ 値式の値がNULLの場合、NULL述語の結果は真となり、NULLでない場合は偽となります。

NULL述語の結果を以下に示します。

表2.68 NULL述語の結果

対応する値	X IS NULL	X IS NOT NULL	NOT X IS NULL
NULL	真	偽	偽
NULLでない	偽	真	真

・ “x IS NOT NULL”と記述したものは、“NOT (x IS NULL)”と記述したのと同じ意味です。

・ NULL述語の値式が動的パラメタ指定の場合は、エラーとなります。

使用例

表Tに対して、以下のようなNULL値が存在するかどうかの検査を行います。

表T

	C1	C2	C3	C4
R1	a	15	－	x
R2	b	10	30	y
R3	c	－	－	z
R4	d	－	60	v

－： NULL値を表します。

例1

列C2の値がNULLの行を検索します。

```
WHERE C2 IS NULL
→結果として、行R3とR4が得られます。
```

例2

列C2の値がNULLでない行を検索します。

```
WHERE C2 IS NOT NULL
→結果として、行R1とR2が得られます。
```

例3

列C3の値がNULLでない行を検索します。

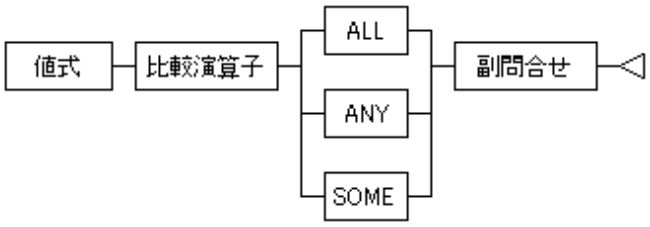
```
WHERE NOT C3 IS NULL
→結果として、行R2およびR4が得られます。
```

2.12.6 限定述語

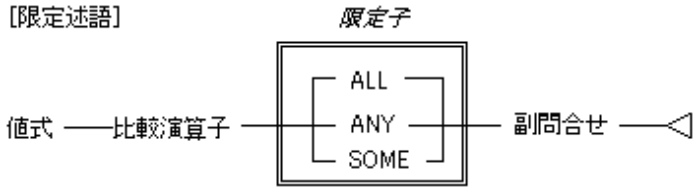
機能

限定述語は、限定された値の集合との比較を指定します。

記述形式



構文要素の構成



参照項番

- 値式 → “2.11 値式”
- 副問合せ → “2.14 副問合せ”

一般規則

- 比較演算子の種類については、“表2.60 比較演算子”を参照してください。
- 値式のデータ型と、副問合せのデータ型は、比較可能であることが必要です。比較可能なデータ型については、“表2.61 比較可能なデータ型”を参照してください。

ALL

ー 限定子ALLを指定した場合、限定述語の結果は次のようになります。

副問合せの結果が空のとき、または値式の値と副問合せの結果を比較し、副問合せの結果のすべての値について、比較演算が真になる場合、限定述語の結果は真となります。副問合せの結果の値について、1つでも比較演算が偽になる場合は、限定述語の結果は偽となります。限定述語の結果が真または偽でない場合、不定となります。これらの関係を以下に示します。

表2.69 限定子ALLを指定した場合の限定述語の結果

限定述語の結果	副問合せとの比較結果			備 考
	結果が真の行	結果が偽の行	結果が不定の行	
真	なし	なし	なし	副問合せの結果が空
	あり	なし	なし	
偽	あり	あり	あり	
	あり	あり	なし	
	なし	あり	あり	
	なし	あり	なし	
不定	あり	なし	あり	
	なし	なし	あり	

あり: 該当する結果がある

なし: 該当する結果はない

ANYまたはSOME

ー 限定子ANYまたはSOMEを指定した場合、限定述語の結果は次のようになります。値式の値と副問合せの結果を比較し、副問合せの結果の値について、1つでも比較演算が真の場合、限定述語の結果は真となります。副問合せの結果の値について、すべての比較演算が偽の場合、限定述語の結果は偽となります。限定述語の結果が真または偽でない場合、不定となります。これらの関係を以下に示します。

表2.70 限定子ANYまたはSOMEを指定した場合の限定述語の結果

限定述語の結果	副問合せとの比較結果			備考
	結果が真の行	結果が偽の行	結果が不定の行	
真	あり	あり	あり	
	あり	あり	なし	
	あり	なし	あり	
	あり	なし	なし	
偽	なし	なし	なし	副問合せの結果が空
	なし	あり	なし	
不定	なし	あり	あり	
	なし	なし	あり	

あり: 該当する結果がある

なし: 該当する結果はない

ー 限定述語で“値式 = ANY 副問合せ”と記述したものは、IN述語で“値式 IN副問合せ”と記述したのと同じ意味です。

DESCRIBE情報について

ー 値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、副問合せのデータ型になります。

使用例

例1

取引製品110を発注している会社名を検索します。

```
SELECT  会社名  FROM  会社表
WHERE   会社番号 = ANY
    (SELECT  取引先                          (1)
     FROM   発注表                          (1)
     WHERE  取引製品 = 110)                  (1)
```

(1)の問合せ式により導出される結果は、下のようになります。

会社表

会社番号	会社名	・・・・
61	アダム電気	
62	アイデア商事	
63	大月産業	
71	川川電気	
72	龍巻産業	
74	第第商事	

発注表

取引先	取引製品	・・・・
61	123	
61	124	
61	138	
62	110	
63	111	
71	140	
72	140	
74	110	
74	111	

会社表中の会社番号の値について、(1)で導出された結果と等しい値をもつ行が、少なくとも1つ存在する場合に真となります。したがってこの問合せ結果は下のようになります。

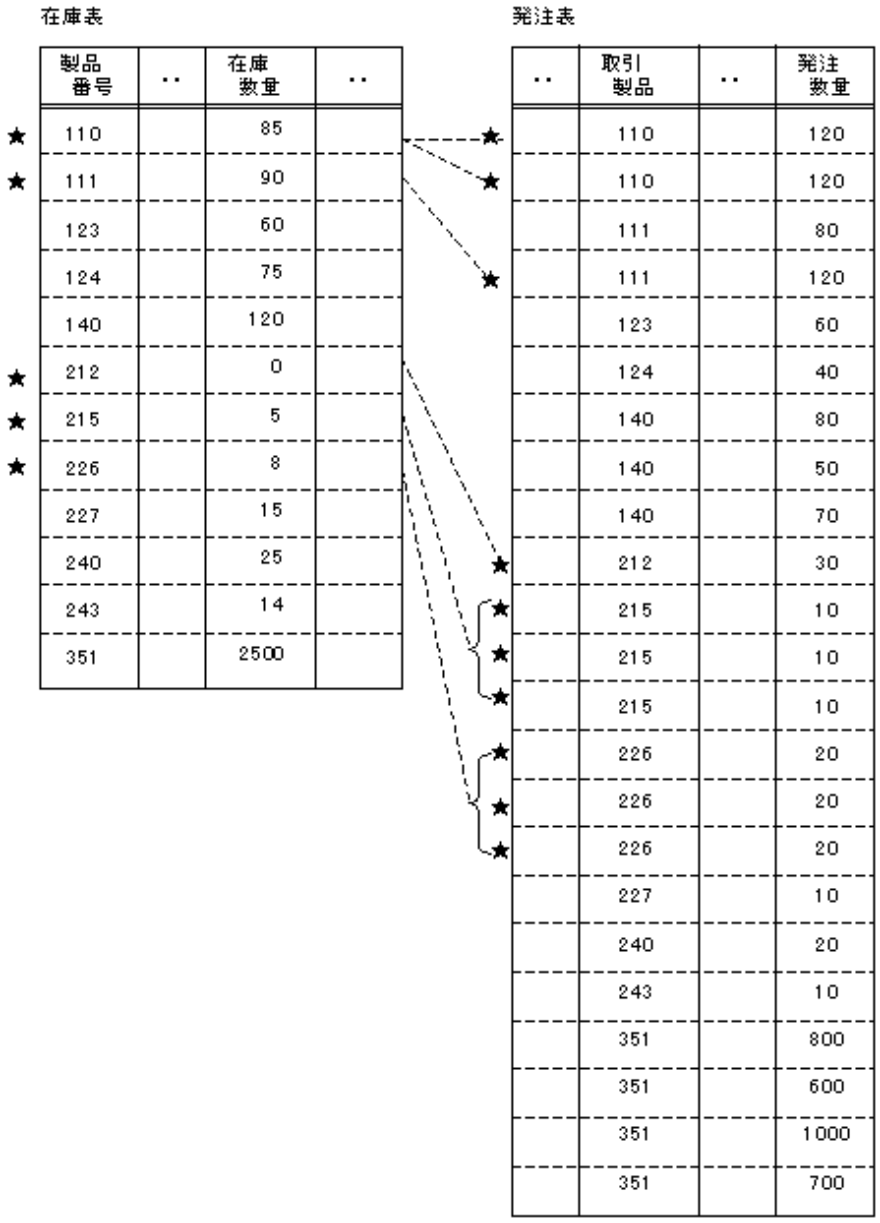
会社名
アイデア商事
第第商事

例2

在庫数量が、ある取引先からの発注数量を下回っている製品番号を検索します。

```
SELECT  製品番号  FROM  在庫表
WHERE  在庫数量 < ANY
    (SELECT  発注数量
     FROM    発注表
     WHERE   取引製品 = 製品番号)
```

上記の問合せ式において、在庫表の製品番号ごとの在庫数量と、発注表の発注数量との関係を次に示します。



発注表中から取引製品ごとに発注数量を取り出します。取り出した発注数量のどれか1つでも、在庫表中の在庫数量よりも多ければ真となります。このとき、比較対象となる在庫数量は、製品番号が発注表中の取引製品と同じである行の値です。上図において、★の部分は、“ANY”の結果が真となる行を表しています。したがって、この問合せの結果は次のようになります。

製品番号
110
111
212
215
226

例3

在庫数量が、すべての取引先の発注数量より下回っている製品番号を検索します。

```
SELECT  製品番号  FROM  在庫表
WHERE  在庫数量 < ALL
    (SELECT  発注数量
     FROM    発注表
     WHERE   取引製品 = 製品番号)
```

上記の副問合せにおいて、製品番号ごとの在庫表の在庫数量と、発注表の発注数量との行間の関係は、例2の場合と同じです。在庫表の製品番号と同じ取引製品をもつ発注表中のすべての行に対して、各取引先からの発注数量が、在庫数量を上回っている場合に真となります。したがって、この問合せの結果は次のようになります。

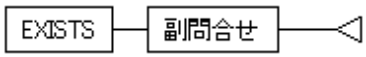
製品番号
110
212
215
226

2.12.7 EXISTS述語

機能

EXISTS述語は、空集合との比較を指定します。

記述形式



参照項番

- 副問合せ → “2.14 副問合せ”

一般規則

- 副問合せの結果が空でない場合、EXISTS述語の結果は、真となります。
- 副問合せの結果が空の場合、EXISTS述語の結果は、偽となります。

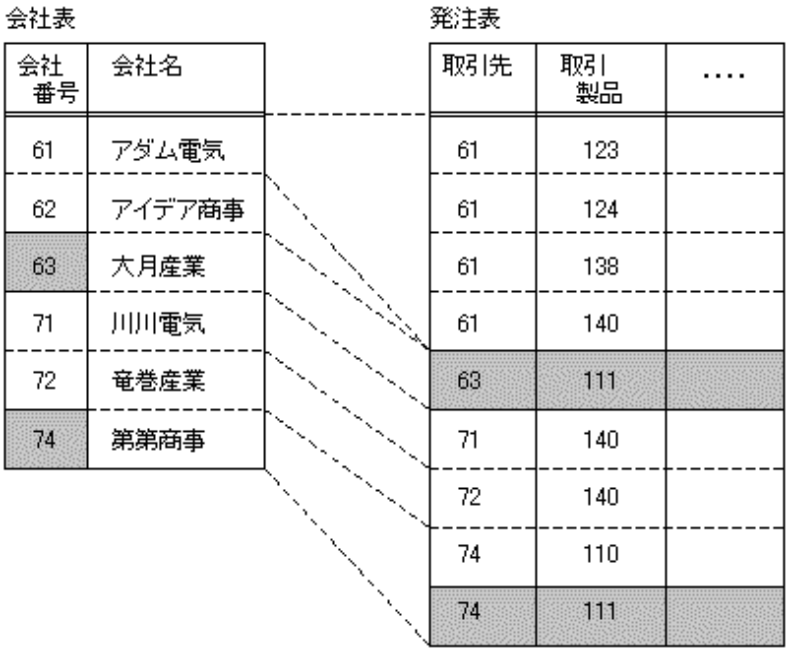
使用例

例

111の製品番号を発注している取引会社名を検索します。

```
SELECT  会社名  FROM  会社表
WHERE  EXISTS(SELECT * FROM  発注表
              WHERE  取引先 = 会社表.会社番号
              AND    取引製品 = 111)
```

上記の問合せにおいて、取引先会社ごとの取引製品の集合の関係は以下のようになります。



“EXISTS”が指定されているので、発注表の中に、各取引先が会社表の会社番号と同じで、かつ取引製品が111である行が存在している集合については真となります。つまり、図中の取引先が63、74を持つ行集合が真となります。したがって、この問合せの結果は次のようになります。

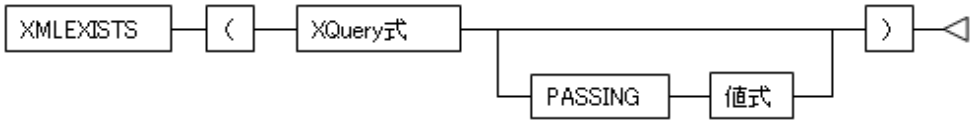
会社名
大月産業
第第商事

2.12.8 XMLEXISTS述語

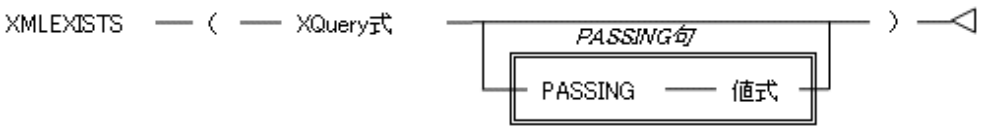
機能

XMLEXISTS述語は、XQuery式の評価結果が空シーケンスであるか否かを評価します。

記述形式



構文要素の構成



参照項番

- 値式 → “2.11 値式”

一般規則

- XMLEXISTS述語に指定されたXQuery式の評価結果が空シーケンスでない場合、XMLEXISTS述語の結果は真となります。
- XMLEXISTS述語に指定されたXQuery式の評価結果が空シーケンスである場合、XMLEXISTS述語の結果は偽となります。
- XMLEXISTS述語は、以下の探索条件にのみ指定可能です。
  - 問合せ指定のWHERE句の探索条件
  - UPDATE文およびDELETE文のWHERE句の探索条件
- OBJECT構造の表を指定したSQL文にXMLEXISTS述語は指定できません。
- XMLEXISTS述語を問合せ指定に指定した場合、並列検索は行いません。並列検索が指定されても無視します。

XQuery式

- XMLデータの検索式を文字列定数で指定します。

参照

XQuery式に指定する検索式の詳細は、“XQueryリファレンス”を参照してください。

注意

XQuery式に指定する検索式は文字列定数で表現するため、検索式の記述内容に'(シングルクォーテーション)を指定することはできません。'(シングルクォーテーション)を含む文字列リテラルを指定する場合には、既定義エンティティー参照の'を指定してください。
既定義エンティティー参照の詳細は、“XQueryリファレンス”を参照してください。

例

A要素のテキストノードの値が「'ABC銀行'」を表すXQuery式

```
XMLEXISTS('/A[text()="&apos;ABC銀行&apos;"]'
   PASSING 伝票)
```

- XQuery式に空文字または空白文字列のみ指定した場合、XMLEXIST述語はエラーになります。
- XQuery式の問合せの評価時にエラーが発生した場合、XMLEXIST述語はエラーになります。

PASSING句

- PASSING句の値式にはXQuery式の検索対象となるXMLデータを指定します。
- PASSING句の値式のデータ型は、BLOB型であることが必要です。
- PASSING句に以下の値式を指定することはできません。
  - 集合関数指定
  - CAST指定
  - XMLQUERY関数
  - ファンクションルーチン
- 値式に指定するXMLデータは、正しいデータ形式であることが必要です。誤ったデータ形式であることを検出した場合、XMLEXISTS述語の結果は偽となります。

参照

XMLデータの正しいXML形式に関しては、“アプリケーション開発ガイド(共通編)”の“SQL/XMLで扱えるデータ形式”を参照してください。

- PASSING句を省略、またはPASSING句の値式の値がNULLである場合、XMLEXISTS述語の評価結果は不定になります。

DESCRIBE情報について

- 値式に動的パラメタ指定が指定された場合のDESCRIBE情報を以下に示します。

表2.71 動的パラメタ指定が指定された場合のDESCRIBE情報

オペランド	DESCRIBE情報
PASSING句の値式	BLOB型

使用例

製品番号が200番の製品を発注している取引会社の会社番号を検索します。

```
SELECT  会社番号 FROM  会社表
WHERE  XMLEXISTS('/取引先会社情報[発注/製品番号/text() = 200]'
                        PASSING  会社表.会社情報)
```

上記の問合せにおける、取引先会社と発注製品の情報を以下に示します。

会社表

会社番号	会社情報
61	XML1
62	XML2
63	XML3
71	XML4
72	XML5

XML1

```
<取引先会社情報>
  <会社名>アダム電気</会社名>
  <発注>
    <製品番号>130</製品番号>
  </発注>
    :
</取引先会社情報>
```

XML2

```
<取引先会社情報>
  <会社名>アイデア商事</会社名>
  <発注>
    <製品番号>150</製品番号>
  </発注>
    :
</取引先会社情報>
```

XML3

```
<取引先会社情報>
  <会社名>大月産業</会社名>
  <発注>
    <製品番号>200</製品番号>
  </発注>
    :
</取引先会社情報>
```

XML4

```
<取引先会社情報>
  <会社名>川川電気</会社名>
  <発注>
    <製品番号>320</製品番号>
  </発注>
    :
</取引先会社情報>
```

XML5

```
<取引先会社情報>
  <会社名>竜巻産業</会社名>
  <発注>
    <製品番号>200</製品番号>
  </発注>
    :
</取引先会社情報>
```

XMLEXISTS述語の指定により、会社表の会社情報列に格納された各XMLデータに対して、XQuery式'/取引先会社情報[発注/製品番号/text()= 200]'が評価されます。XQuery式の評価結果より、XMLEXISTS述語は図中の会社番号63と72の行に対して真を返します。また、それ以外の行に対しては偽を返します。このSQL文の結果を以下に示します。

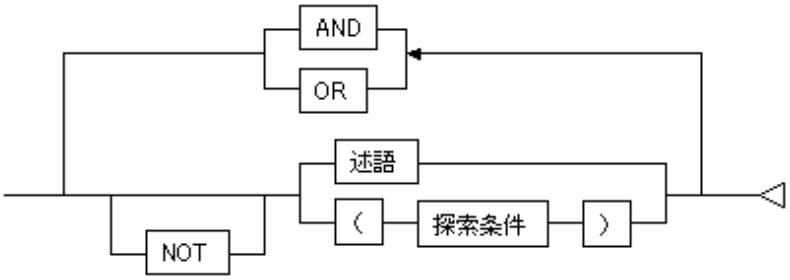
会社番号
63
72

2.13 探索条件

機能

ブール演算子を適用した結果によって、真、偽、または不定となる条件を指定します。

記述形式



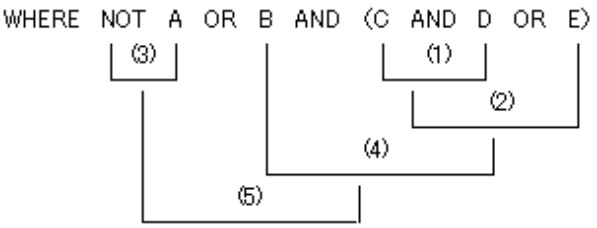
参照項番

- 述語 → “2.12 述語”

一般規則

- 各述語の評価結果に対してブール演算子(NOT、AND、OR)を適用することによって、探索条件の結果が導出されます。ブール演算子が指定されないならば、探索条件の結果は、指定された述語の結果となります。
- 探索条件中に括弧が指定されている場合は、括弧の中の式が最初に評価されます。括弧が指定されていない式は、NOT、AND、ORの順で評価されます。

  A、B、C、D、Eを述語とすると、下図に示す番号順に評価されます。



- 述語には、真、偽または不定の真偽値の評価ができるための条件を記述します。述語の真偽値をブール演算子を適用すると次のような結果となります。

AND

ー ANDが指定された場合は、以下の真理表に従って評価されます。

表2.72 ANDの真理表

		右辺		
		真	偽	不定
左辺	真	真	偽	不定

		右辺		
		真	偽	不定
	偽	偽	偽	偽
	不定	不定	偽	不定

OR

- ORが指定された場合は、以下の真理表に従って評価されます。

表2.73 ORの真理表

		右辺		
		真	偽	不定
左辺	真	真	真	真
	偽	真	偽	不定
	不定	真	不定	不定

NOT

- NOTが指定された場合、NOTに導かれるブール式が真ならばそのブール演算の結果は偽、偽ならば真、不定ならば不定となります。

使用例

例1

AND

```
SELECT *
INTO :CLSV,:HGHV,:LWV,:AVRV
FROM SCORETBL
WHERE AVR > (HIGH+LOW)/2
  AND AVR > 60
```

SCORETBL

CLASS	HIGH	LOW	AVR
A	88	15	54
B	90	20	60
C	96	30	65

⇨

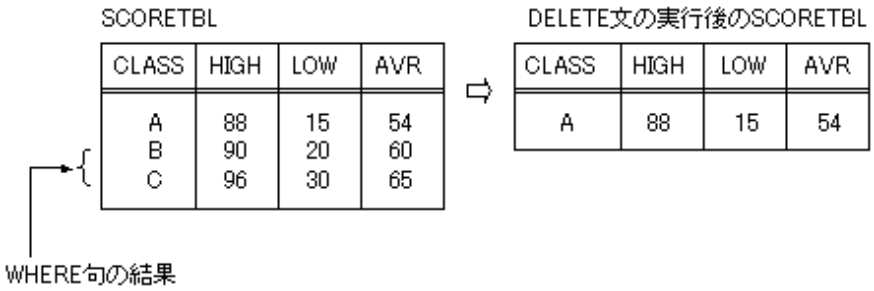
WHERE句の結果

CLASS	HIGH	LOW	AVR
C	96	30	65

例2

OR

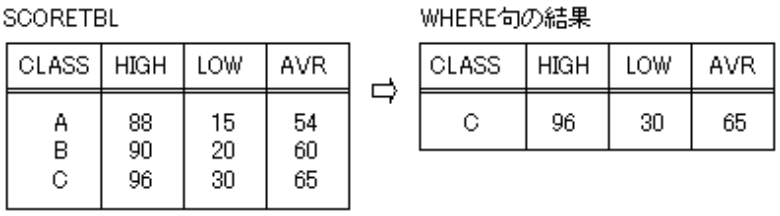
```
DELETE FROM SCORETBL
WHERE HIGH >= 90
   OR AVR > 60
```



例3

NOT

```
SELECT *
INTO :CLSV,:HGHV,:LWV,:AVRV
FROM SCORETBL
WHERE NOT LOW < 25
```

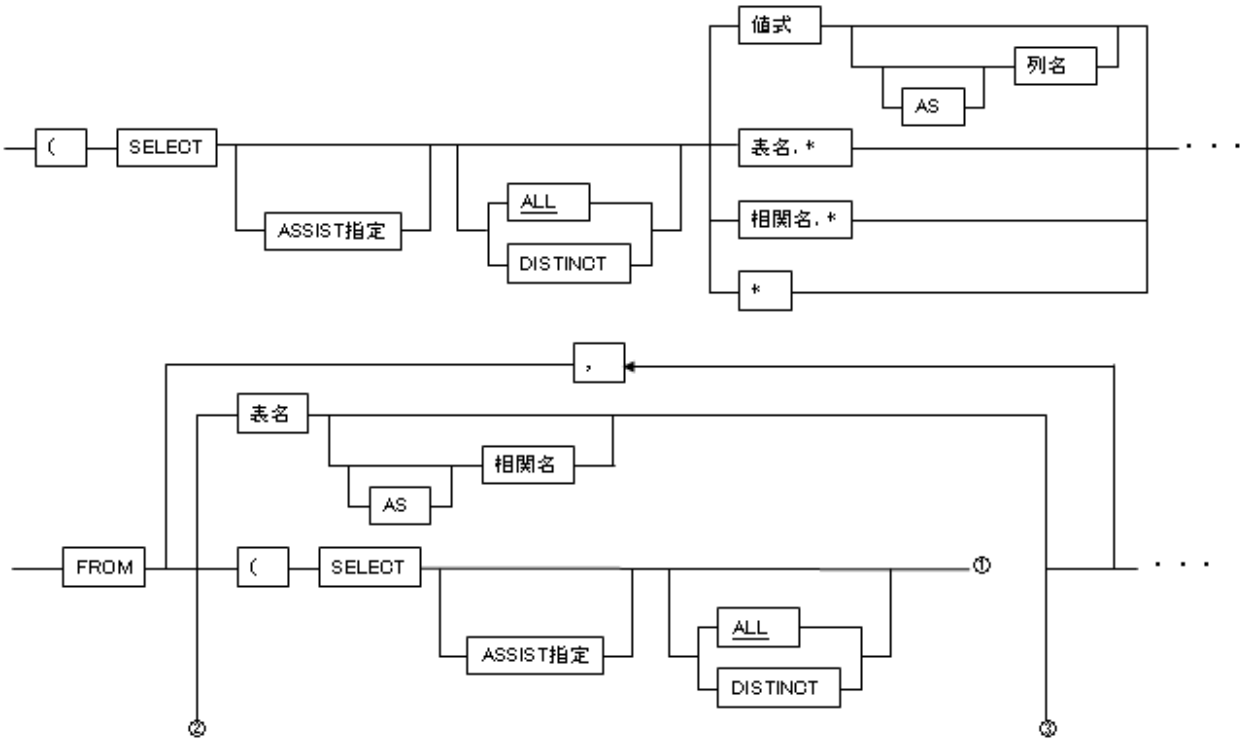


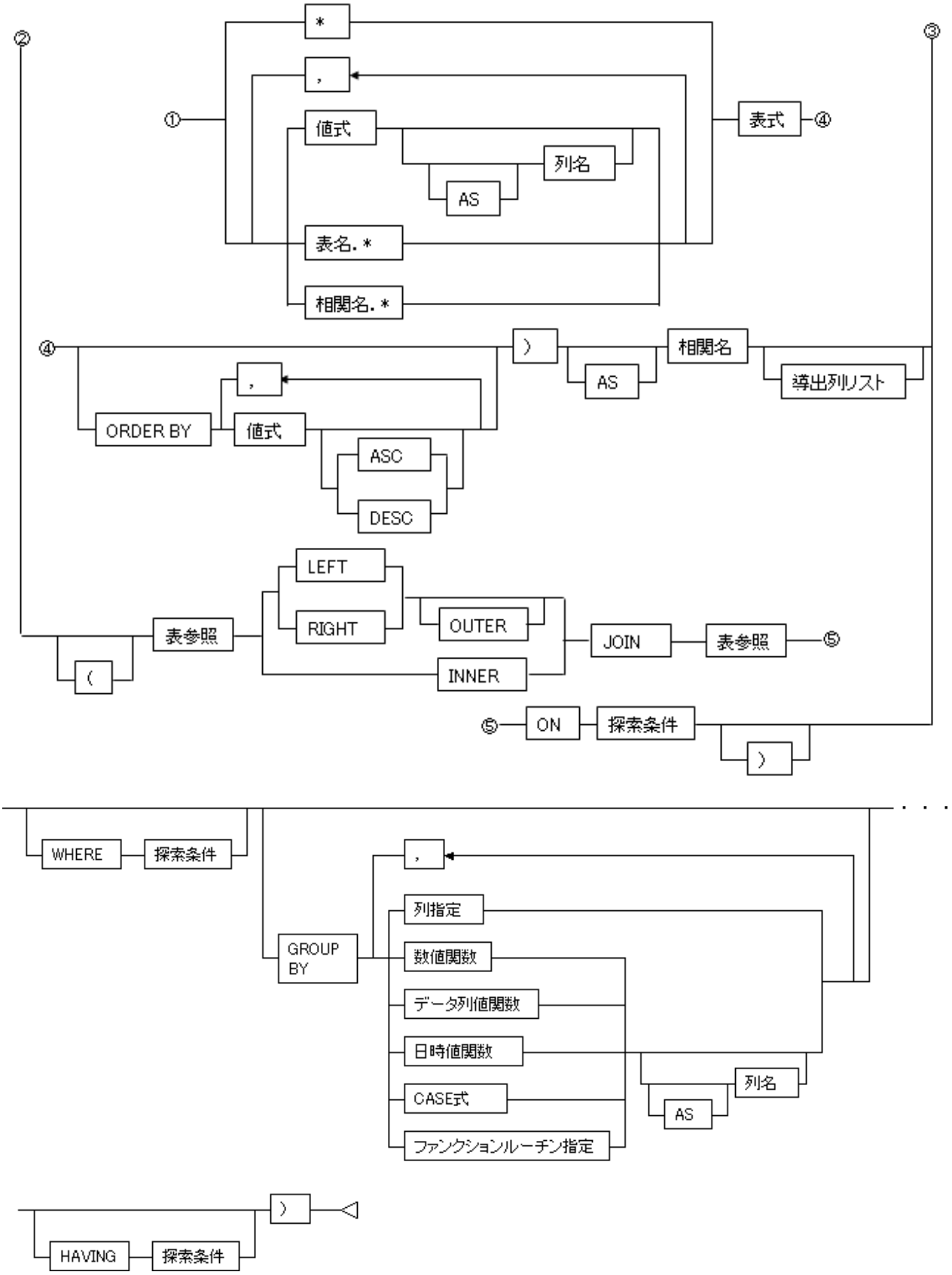
2.14 副問合せ

機能

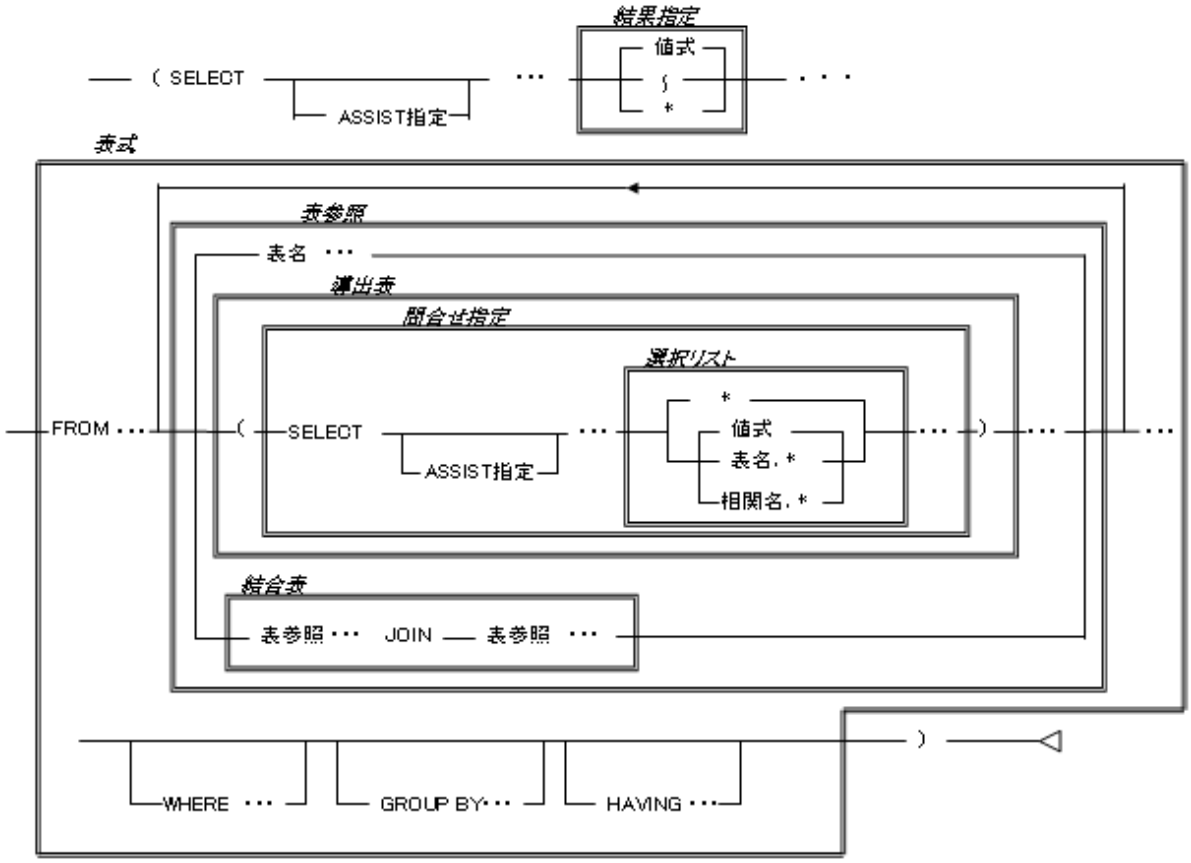
表式から導出される表を指定します。

記述形式


(
SELECT
ASSIST指定
ALL
DISTINCT
値式
列名
AS
表名.*
相関名.*
*
,
表名
相関名
AS
FROM
(
SELECT
ASSIST指定
ALL
DISTINCT
①
②
③


②
*
,
①
値式
列名
AS
表名.*
相関名.*
表式
④
③
④
)
AS
相関名
導出列リスト
,
ORDER BY
値式
ASC
DESC
LEFT
RIGHT
OUTER
INNER
(
表参照
JOIN
表参照
⑤
⑤
ON
探索条件
)
WHERE
探索条件
,
GROUP BY
列指定
数値関数
データ列値関数
日時値関数
CASE式
ファンクションルーチン指定
列名
AS
)
HAVING
探索条件

構文要素の構成



参照項番

- ASSIST指定 → “2.15 ASSIST指定”
- 値式 → “2.11 値式”
- 探索条件 → “2.13 探索条件”
- データ列値関数 → “2.5.3 データ列値関数”
- 数値関数 → “2.5.2 数値関数”
- 日時値関数 → “2.5.4 日時値関数”
- 列指定 → “2.4 列指定”
- CASE式 → “2.6 CASE式”
- ファンクションルーチン指定→ “2.5.5 ファンクションルーチン指定”

一般規則

- 副問合せの結果を以下に示します。
  - ― 副問合せにDISTINCTが指定されている場合、副問合せの結果の各行同士がまったく同じ値を持つ行がある場合、それらの複数行を1行とする処理が施されます。このとき、NULL値同士は等しいとみなされます。ALLを指定するか、または何も指定しない場合、副問合せの結果に変化はありません。
  - ― 表式の結果がグループ表でない場合、副問合せの結果は次のようになります。結果指定に集合関数指定がない場合、表式の結果の各行に結果指定の値式で示す演算が行われた、行の集合となります。結果指定に集合関数の指定がある場合の結果は、表式の結果を集合関数に適用した1行となります。

ー 表式の結果がグループ表の場合、副問合せの結果は次のようになります。結果の行の数はグループの数となります。各行の値は、結果指定に集合関数の指定がある場合には、グループを集合関数に適用した結果となり、列指定の場合には、グループ化列の値となり、グループ化関数の場合は、グループ化関数の値となります。なお、結果指定にGROUP BY句のAS句の列名を指定した場合は、対応するグループ化列、グループ化関数またはCASE式の値となります。

ー 表式の結果が空であれば、副問合せの結果は、空の表となります。

結果指定

ー 結果指定に“表名.*”、“相関名.*”、または“*”を指定する場合、以下の条件を満足しなければなりません。

- EXISTS述語の副問合せでは、指定の条件はありません。
- EXISTS述語以外の副問合せでは、“表名.*”または“相関名.*”を指定する場合には表名または相関名で示す表が持つ列の個数が1つであることが必要です。また、“*”を指定する場合にはFROM句に指定する表が1つで、その表の列の個数が1つであることが必要です。

値式

ー 値式の列指定は、FROM句で指定した表の列またはGROUP BY句のAS句の列名であることが必要です。

ー 値式は、定数または変数指定だけではいけません。必ず、列指定を1つ以上含むことが必要です。

その他の構文要素

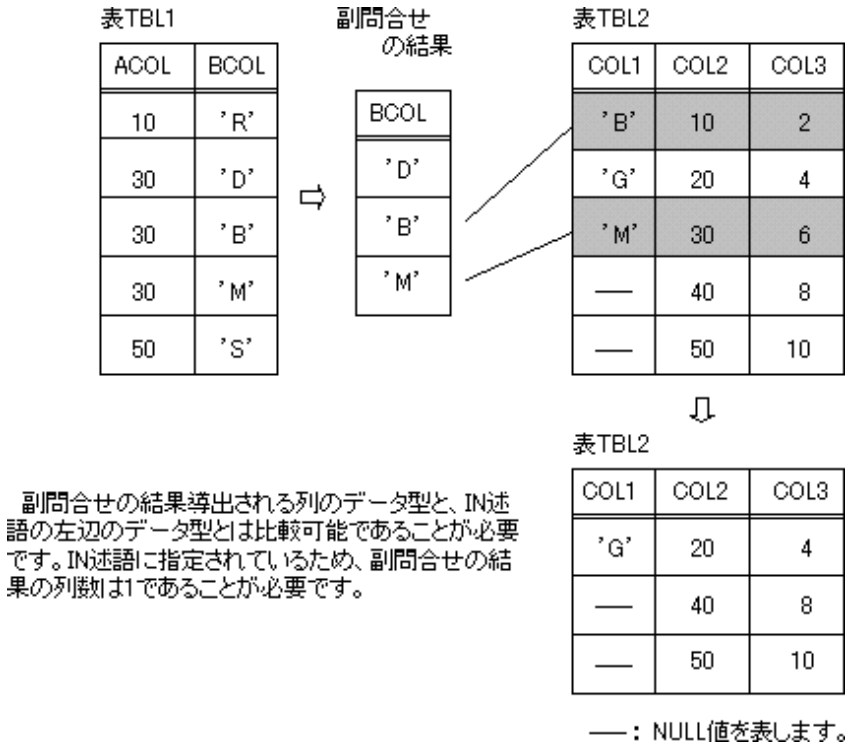
その他の構文要素の説明は、“3.26 DECLARE CURSOR(カーソル宣言)”を参照してください。

使用例

例

DELETE文:探索に適用した場合。

```
DELETE FROM TBL2 WHERE COL1 IN
(SELECT BCOL FROM TBL1 WHERE ACOL = 30)
```



2.15 ASSIST指定

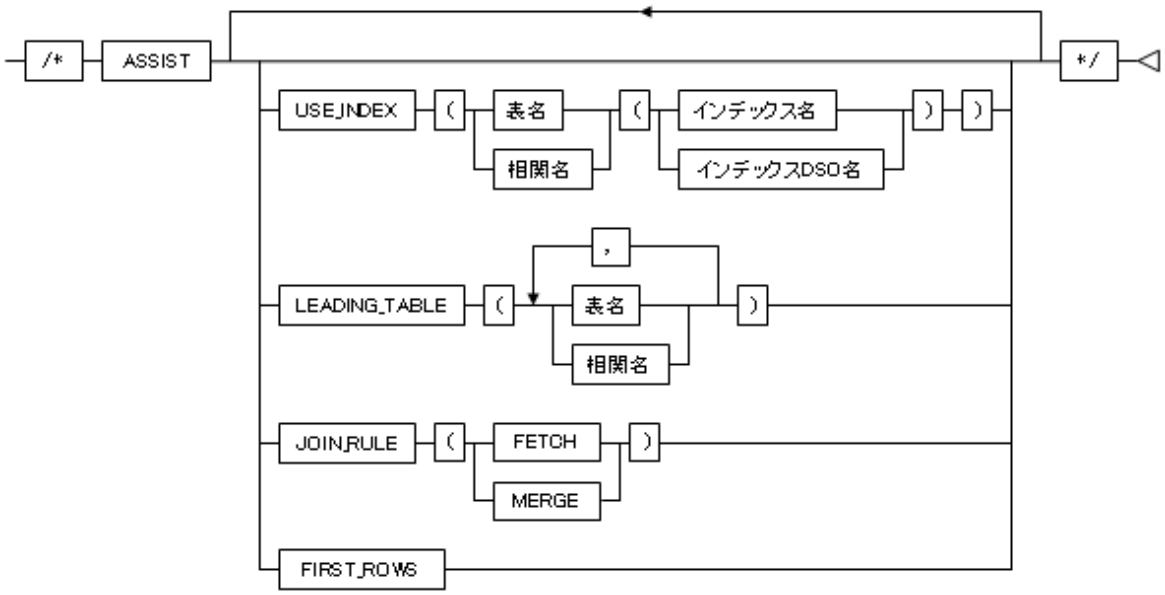
機能

SQL文を実行する際のアクセスプランを固定化する時に指定します。

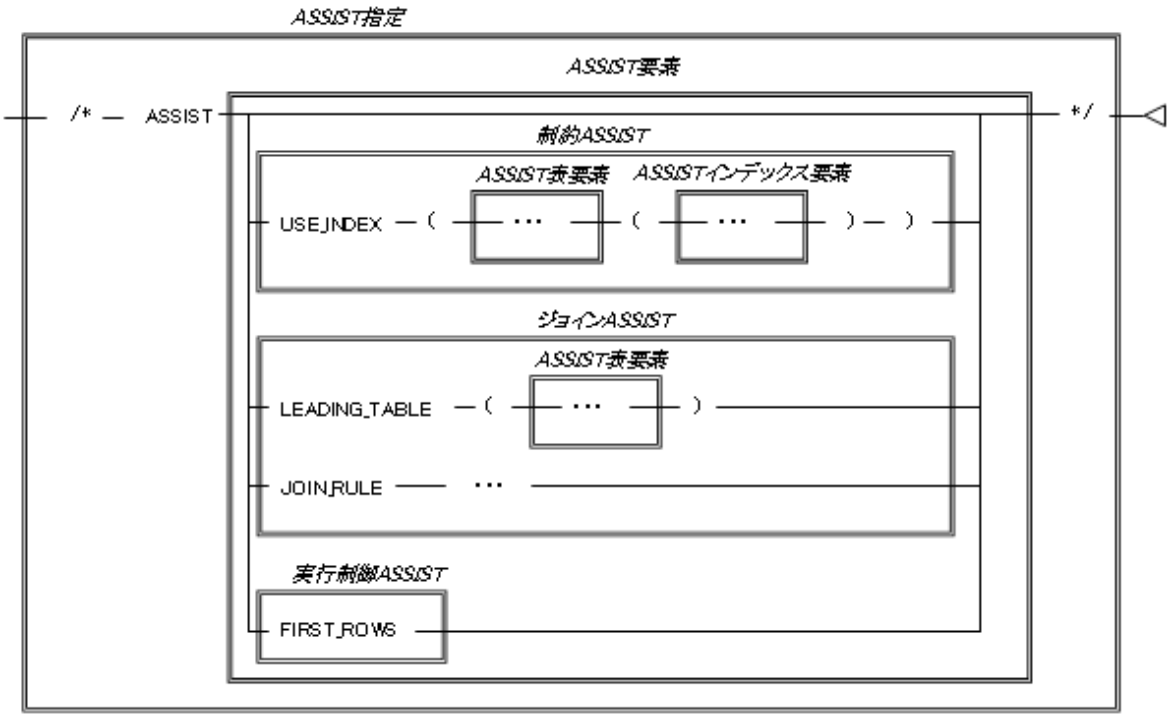
参照

アクセスプランについては、“アプリケーション開発ガイド(共通編)”を参照してください。

記述形式



構文要素の構成



参照項番

- /* → “2.1.3 トークン”
- */ → “2.1.3 トークン”

一般規則

ASSIST

- ASSIST要素に構文誤りがあった場合、構文誤りが存在するASSIST要素は無効となり、正しく指定されたASSIST要素のみが有効となります。
- ASSIST要素はASSIST指定を指定した問合せのみに適用されます。たとえば、ある問合せが副問合せを持つ場合、ある問合せに指定したASSIST指定は副問合せには適用されません。
- SQL文に以下のシステム表を指定した場合、ASSIST指定を指定することはできません。
  - RDBII_ASSISTTABLE
  - RDBII_DSI_STATUS
  - RDBII_CON_PRM
  - RDBII_CON_PRM_DIV
  - RDBII_SYS_PRM
  - RDBII_SYS_PRM_DIV
  - RDBII_CON_INF

USE_INDEX

- ASSIST表要素に指定された表に対して、インデックス名またはインデックスDSO名で指定したインデックスを使用します。
- 同一のASSIST表要素に対して、複数のUSE_INDEXを指定することはできません。
- 探索条件にROW_IDを指定した場合、USE_INDEXは指定できません。

参照

USE_INDEXの使用方法については、“アプリケーション開発ガイド(共通編)”の“制約ASSISTの使用方法”を参照してください。

LEADING_TABLE

- ASSIST表要素に指定された表の順にジョインします。結合表中の表も指定できます。
- 同じASSIST表要素を複数指定することはできません。
- LEADING_TABLEは1つのASSIST指定に1回だけ指定することができます。
- FROM句に指定した表、または導出表が1個の場合、LEADING_TABLEは指定できません。

参照

LEADING_TABLEの使用方法については、“アプリケーション開発ガイド(共通編)”の“ジョインASSISTの使用方法”を参照してください。

JOIN_RULE

― ジョインする方法を指定します。

FETCH

フェッチジョインのアクセスモデルを優先します。

MERGE

マージジョインのアクセスモデルを優先します。

参照

アクセスモデルについては、“アプリケーション開発ガイド(共通編)”を参照してください。

― JOIN_RULEは、1つのASSIST指定に1回だけ指定することができます。

― FROM句に指定した表、または導出表が1個の場合、JOIN_RULEは指定できません。

参照

JOIN_RULEの使用方法については、“アプリケーション開発ガイド(共通編)”の“ジョインASSISTの使用方法”を参照してください。

FIRST_ROWS

― 最初の1件の検索について最短の応答時間を目標にアクセスプランを選択します。

― FIRST_ROWSは、以下のいずれかに指定することができます。

- 最も外側の問合せ指定
- ORDER BY句を指定した導出表

参照

FIRST_ROWSの使用方法については、“アプリケーション開発ガイド(共通編)”の“実行制御ASSISTの使用方法”を参照してください。

表名

― ASSIST指定を適用する表の名前を指定します。

― 表名は、ASSIST指定と同一の問合せ指定のFROM句に指定した表名を指定します。

― スキーマ名による修飾を除いた表名でFROM句の表名を一意に識別可能な場合、スキーマ名は省略することができます。

― ビュー表名は指定できません。

相関名

― ASSIST指定を適用する表の相関名を指定します。

― 相関名は、ASSIST指定と同一の問合せ指定のFROM句に指定した相関名を指定します。

― FROM句に指定した表名に相関名を指定している場合、ASSIST表要素には相関名を指定しなければなりません。

― ビュー表の相関名は指定できません。

インデックス名

― ASSIST指定を適用するインデックスが、一時表に定義したインデックスまたは格納構造定義を行わないインデックスの場合、インデックス名を指定します。

インデックスDSO名

- ASSIST指定を適用するインデックスが、一時表に定義したインデックスまたは格納構造定義を行わないインデックス以外の場合、インデックスDSO名を指定します。
- データ構造がXMLのインデックスDSO名は指定できません。

使用例

ASSIST指定の使用例を以下に示します。“在庫管理.在庫表”および“在庫管理.発注表”という表を用いています。

在庫管理.在庫表

表の格納構造:SEQUENTIAL

インデックスのDSO名	構成列名
INDEX1	製品番号
INDEX2	倉庫番号

在庫管理.発注表

表の格納構造:SEQUENTIAL

インデックスのDSO名	構成列名
INDEX3	取引先
INDEX4	仕入価格
INDEX5	取引製品

例1

USE_INDEXの例を以下に示します。

インデックスDSOのINDEX4を使用して検索します。

```
SELECT /* ASSIST USE_INDEX(発注表(INDEX4)) */
     取引製品,仕入価格
  FROM 在庫管理.発注表
  WHERE 取引先 = 61
    AND 仕入価格 BETWEEN 8000 AND 9000
```

例2

LEADING_TABLEの使用誤りの例を以下に示します。

発注表、在庫表の順にジョインを行います。

```
SELECT /* ASSIST LEADING_TABLE(発注表,在庫表) */
     HA.仕入価格,HA.取引製品
  FROM 在庫管理.在庫表 ZAI,在庫管理.発注表 HA
  WHERE ZAI.製品番号 = HA.取引製品
    AND ZAI.倉庫番号 = 2
    AND HA.取引先 > 70
```

LEADING_TABLEのASSIST表要素に表名が指定されているため、エラーになります。相関名を使用している場合は、相関名を指定してください。

第3章 基本的なSQL文

SQL文に指定する各要素について説明します。

3.1 ALTER DSI文(DSI変更文)

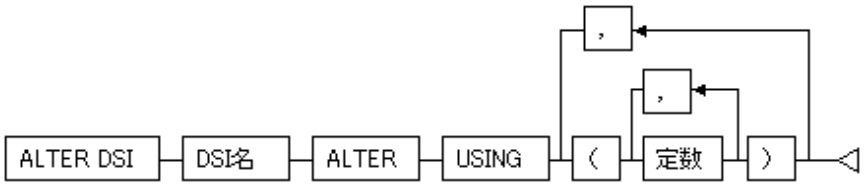
機能

DSIの分割値の変更またはDSIの容量拡張を行います。

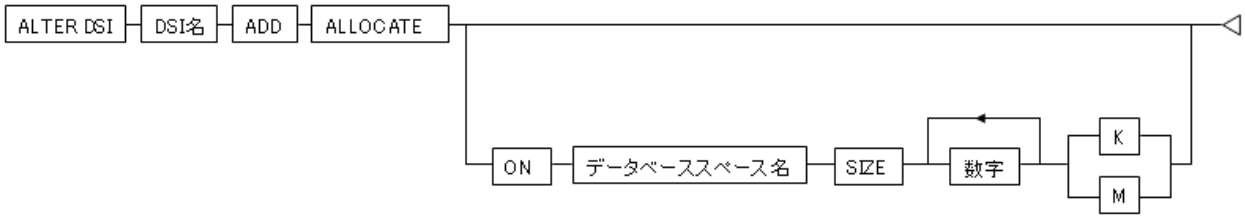
データベース簡単運用の場合は利用できません。

記述形式

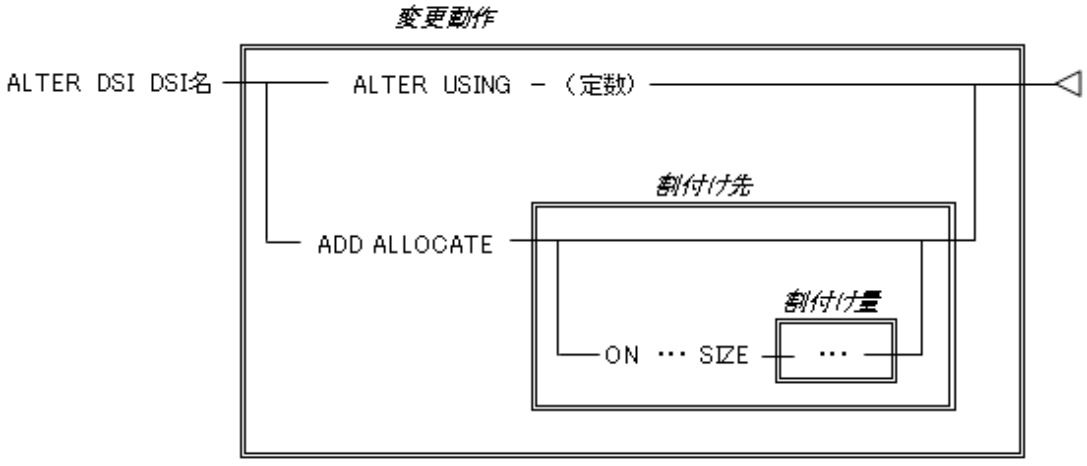
分割値の変更



容量拡張



構文の構成



参照項番

- 定数 → “2.1.2 定数”

権限

- DSIを変更できるのは、表の定義者、スキーマの定義者または表のALTER権の保持者です。また、容量拡張を行うためにはデータベーススペースに対してALLOCATE権が必要です。

一般規則

- 分割値を変更するDSIに関連する表をアプリケーションでアクセス中の場合は、rdbexdsiコマンドでDSIを運用から除外した後、DSIの分割値を変更します。rdbexdsiコマンドの対象となっていないユーザが表をアクセス中の場合、ALTER DSI文は異常終了します。

参照

詳細については、“RDB運用ガイド”の“動的定義変更”を参照してください。

- 容量拡張をアプリケーションから実施する場合、実行中のトランザクションはシステムによりコミットされます。
- 容量拡張を実施する場合、対象のDSIは初期化または創成済みである必要があります。
- メモリに常駐されたDSIは容量拡張ができません。メモリ常駐を解除してください。

参照

メモリ常駐の解除については、“RDB運用ガイド”の“メモリに常駐しているDSIの常駐解除”を参照してください。

- プレオープンされたDSIに対する定義変更はできません。プレオープンを解除してください。

参照

プレオープンの解除については、“クラスタ導入運用ガイド”の“プレオープンするDSIの変更”を参照してください。

DSI名

— 変更するDSIの名前を指定します。

USING

— 定数には、変更する分割値を指定します。

— 同じ表内のほかのDSIの分割値と重複するような変更はできません。

— すでにDSI内に格納されているデータに矛盾が生じるような変更はできません。

— 分割値を変更するDSIはアプリケーションなどで使用中であってはいけません。

— 変更動作は、表定義の変更内容を示します。

— USINGオペランドの一般規則については、“3.12 CREATE DSI文(表のDSI定義文)”を参照してください。

ADD ALLOCATE

— 割付け先のデータベーススペース名を指定します。

— 1つのDSIの割付け対象の割付け先として指定するデータベーススペース名は、同じロググループを使用するデータベーススペースでなければなりません。

— 1つのDSIの割付け対象の割付け先として指定するデータベーススペース名は、暗号化指定が統一されていなければなりません。

— 割付け量には、データベーススペース中に獲得する格納領域の大きさを、符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り上げます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。

- 割付け先を省略する場合、DSIに自動容量拡張の定義が設定されている必要があります。
- ロードシェア運用の場合、DSIに指定されたデータベーススペースと同じシステムに偏在するデータベーススペースでなければなりません。
- データベーススペースをローデバイスに作成している場合、ローデバイスのブロック長の倍数が、DSIの属するDSOのページ長でなければなりません。
- 割付け先のデータベーススペースに割付け量の空き領域があることを確認してください。

参照

詳細は“RDB運用ガイド”の“データベーススペースの容量監視”を参照してください。

- インデックス部の割付け量は、ベース部の5分の1の値となります。ベース部の5分の1がインデックス部のページ長の倍数でない場合、インデックス部のページ長の倍数に繰り上げます。
- 割付け量には、容量拡張を行った結果、DSIの割付け量の制限値を超えない範囲で指定してください。
  DSIの割付け量の制限値については、“C.1 Symfoware/RDBの定量制限”を参照してください。
- 割付け対象のデータベーススペースに、割付け量分の連続領域が存在しない場合、複数の領域に分割して割付けることがあります。複数の領域に分割されたかどうかは、rdbprtコマンドの出力結果で確認できます。DSI情報のAllocation informationのAllocate sizeに、指定した割付け量以下の値が表示された場合、複数の領域に分割して割付けられたと判断できます。

使用例

例1

“DSI1”の分割値を変更します。

```
CREATE DSO DSO1 … WHERE (年月日) BETWEEN (?) AND (?);
CREATE DSI DSI1 … USING (DATE'2005-8-1',DATE'2005-8-31'),
                        (DATE'2006-8-1',DATE'2006-8-31');
                  :
                  :
ALTER DSI DSI1 ALTER USING (DATE'2006-8-1',DATE'2006-8-31'),
                           (DATE'2007-8-1',DATE'2007-8-31');
```

例2

“DSI1”の容量拡張を実施します。

```
ALTER DSI DSI1 ADD ALLOCATE ON DBSP1 SIZE 1M;
```

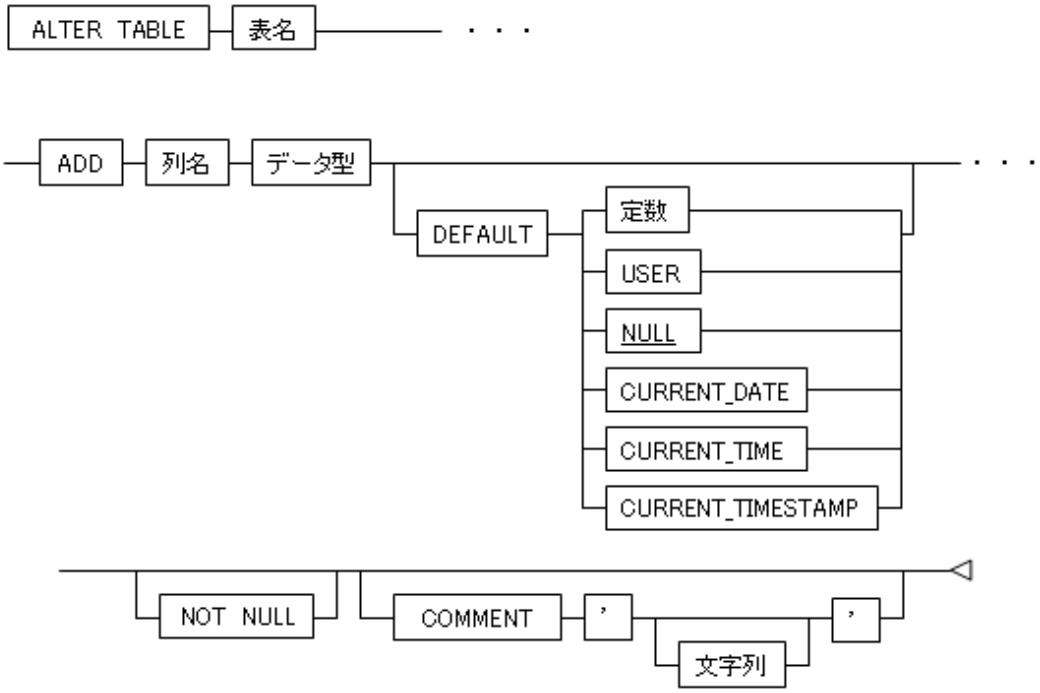
3.2 ALTER TABLE文(表定義変更文)

機能

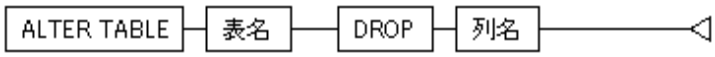
表の定義を変更します。列定義追加、列定義削除、表と列の注釈定義変更があります。

記述形式

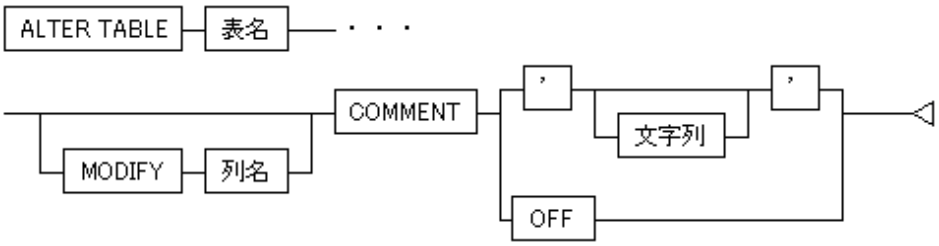
列定義追加



列定義削除



注釈定義変更



参照項番

- データ型 → “2.2 データ型”
- 定数 → “2.1.2 定数”

権限

- 表定義を変更できるのは、その表の定義者、スキーマの定義者またはその表のALTER権の保持者です。

一般規則

- 格納構造がOBJECTの場合、表定義変更文は実行できません。

列定義追加

ー 列定義によって定義される列が、変更対象表に追加されます。このとき、列の定義順は既存の列の後になります。

ー 列名は、変更対象表内で一意の名前であることが必要です。

- すでに存在する行に追加される列の値は、既定値になります。また、DEFAULT句にUSERを指定している場合は、定義文実行のユーザ名になります。
- NOT NULL制約を指定した場合、DEFAULT句にNULL以外を指定することが必要です。
- DEFAULT句を指定していない場合の列の値は、NULLになります。
- BLOB型の列は表のデータ構造がSEQUENTIALまたはOBJECTの場合のみ指定できます。
- 格納構造がSEQUENTIALの場合、BLOB型の列を追加するとき、表の行は、ページ長を超えてもかまいません。ただし、格納構造がRANDOMの場合はページ長を超える列の追加はできません。
- 以下のいずれかの条件に該当するSQL手続き文を含むプロシジャルーチンが定義されている場合は、列定義追加によって影響を受けるため、列を追加することはできません。
  - 問合せ式、問合せ指定および単一行SELECT文の選択リストの次数が増加して、実行できなくなるSQL手続き文
  - INSERT文の挿入列リストが省略されていて、挿入列の数が増加して実行できなくなるSQL手続き文
  - 列名が一意に決定できなくなるSQL手続き文
- 以下の条件に該当するSQL手続き文を含むトリガが定義されている場合は、列定義追加によって影響を受けるため、列を追加することはできません。
  - INSERT文の挿入列リストが省略されていて、挿入列の数が増加して実行できなくなるSQL手続き文
- その他の構文要素の説明は、“3.22 CREATE TABLE文(表定義)”を参照してください。

列定義削除

- 対象表に列が1つしかない場合は、指定することはできません。
- 対象列を構成列に含むインデックス、その列を参照しているビュー表、プロシジャルーチン、トリガまたは一意性制約が存在している場合は削除できません。

注釈定義変更

- 列名を指定した場合は、列の注釈を変更します。列名を指定していない場合は、表の注釈を変更します。
- 注釈定義変更で、OFFを指定した場合、注釈は削除されます。
- 注釈は256バイト以内の文字列(日本語記述可能)を指定することができます。

表名

- 変更する表の名前を指定します。

列名

- 変更する列の名前を指定します。

使用例

例1

表T1に列C3を追加します。

```
ALTER TABLE S1.T1 ADD C3 SMALLINT
```

例2

DEFAULT句を指定した列を追加します(NOT NULL制約を記述した新しい列C4を追加定義します)。

```
ALTER TABLE S1.T1 ADD C4 SMALLINT DEFAULT 1 NOT NULL
```

例3

表T1から列C3を削除します。

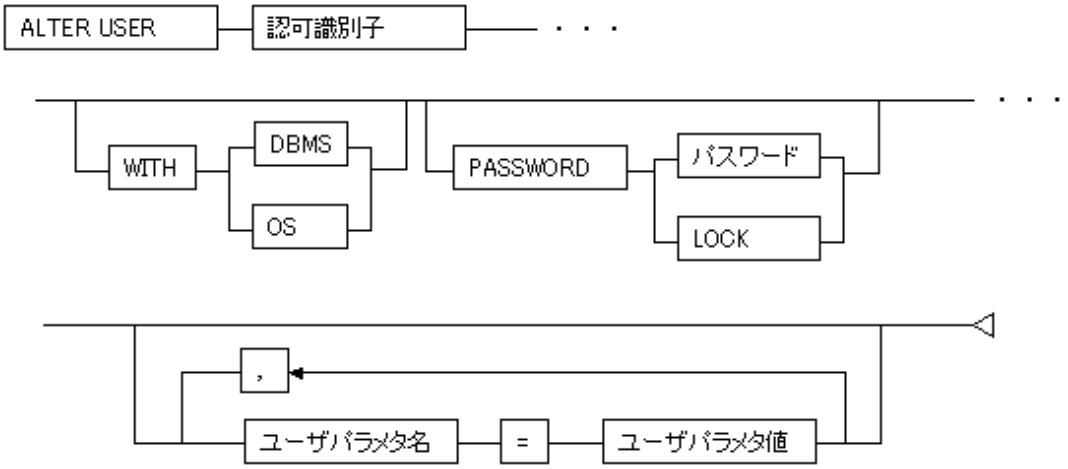
```
ALTER TABLE S1.T1 DROP C3
```

3.3 ALTER USER文(利用者変更文)

機能

利用者の属性を変更します。

記述形式



参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- 利用者の属性を変更できるのは、RDBディクショナリの定義者のみです。

一般規則

- 利用者登録の使用宣言をしている場合に実行できます。利用者登録の使用宣言を使用するか否かはSET SYSTEM PARAMETER文で指定します。
- 少なくとも1つのオペランドを指定しなければなりません。

認可識別子

- 利用者の属性を変更する認可識別子を指定します。
- 認可識別子は、18文字以内の先頭が英字で始まる英数字、または9文字以内の日本語文字列を指定します。

WITH句

- DBMSを指定した場合は、利用者の認可識別子とパスワードをSymfoware/RDBで管理するように変更します。この場合、PASSWORD句を指定しなければなりません。
- OSを指定した場合は、利用者の認可識別子をOSのログイン名に対応させて管理するように変更します。

PASSWORD句

- パスワードは、文字列定数で指定します。
- 認可識別子がSymfoware/RDBで管理されている場合のみパスワードを指定することができます。

ー PASSWORD句にパスワードを指定した場合、パスワードを指定された文字列に変更します。また、変更前のパスワードがロックされている場合は、これを解除します。PASSWORD句にLOCKを指定した場合、指定された認可識別子をロックした状態とします。

ー パスワードの指定形式の詳細については、“3.24 CREATE USER文(利用者定義文)”を参照してください。

ー WITH句にOSを指定した場合は、PASSWORD句の指定は無視されます。

ユーザパラメタ名

ー 利用者の認証情報のユーザパラメタのパラメタ名を指定します。指定可能なユーザパラメタは以下のとおりです。

- MAX_CONNECTION(注)
- MAX_MEMORY_USE(注)
- MAX_WORKFILE_USE(注)
- MAX_WORKFILE_NUM(注)
- MAX_TRAN_TIME(注)
- MAX_TRAN_MEM(注)
- MAX_WAIT_TIME(注)
- PASSWORD_CHANGE_TIME
- PASSWORD_LIMIT_TIME
- INVALID_PASSWORD_WAIT_TIME
- INVALID_PASSWORD_TIME
- MIN_PASSWORD_SIZE
- DEFAULT_ROLE

注) Symfoware Server Enterprise Extended Editionの場合のみ指定可能です。

ー DEFAULT_ROLE以外のユーザパラメタ名の指定は、WITH句にDBMSを指定して、利用者の認可識別子とパスワードをSymfoware/RDBで管理する場合のみ有効となります。

ー 利用者の認証情報のユーザパラメタは、SET SYSTEM PARAMETER文さらにCREATE USER文で設定した利用者ごとのユーザパラメタを変更する場合に指定します。
利用者の認証情報のユーザパラメタの詳細については、“3.61 SET SYSTEM PARAMETER文”を参照してください。

ー DEFAULT_ROLEは、ALTER USER文でのみ指定します。

ー ユーザパラメタの設定値は、次のコネクション接続時に有効になります。

ユーザパラメタ値

ー 利用者の認証情報のユーザパラメタのパラメタの値を指定します。

DEFAULT_ROLE

【指定形式】

DEFAULT_ROLE = ロール名[{,ロール名}…]

【ユーザパラメタの意味】

デフォルトロールのロール名を指定します。指定するロール名は、対象の認可識別子に付与されていなければなりません。

デフォルトロールとは、アプリケーション中でSET ROLE文を実行せずに有効となるロールを環境構築時にあらかじめ設定しておくものです。

使用例

例1

認可識別子“SUZUKI”をSymfoware/RDBで管理するように変更します。

```
ALTER USER SUZUKI
  WITH DBMS PASSWORD '777###FF'
```

例2

認可識別子“SUZUKI”のパスワードを変更します。

```
ALTER USER SUZUKI
  PASSWORD 'A@B#D5D1'
```

例3

認可識別子“SUZUKI”にデフォルトロール“STOCKS_A2”を設定します。

```
ALTER USER SUZUKI
  DEFAULT_ROLE=STOCKS_A2
```

例4

認可識別子“SUZUKI”に対して、1つのコネクションで使用可能なメモリ量“MAX_MEMORY_USE”を8メガバイト、1つのコネクションで使用可能な作業用ファイルの量“MAX_WORKFILE_USE”を100メガバイトに変更します。

```
ALTER USER SUZUKI
  MAX_MEMORY_USE=8,
  MAX_WORKFILE_USE=100
```

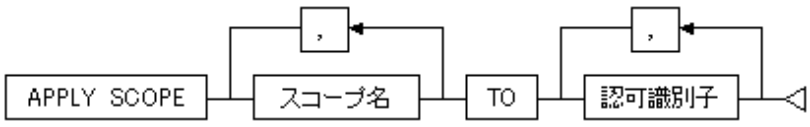
3.4 APPLY SCOPE文(スコープ適用文)

機能

スコープを表の利用者に適用します。

データベース簡単運用の場合は利用できません。

記述形式



参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- スコープを適用できるのは、スコープの定義者のみです。

一般規則

- 指定されたスコープは定義済であることが必要です。

・ 指定されたスコープが、指定された認可識別子に対してすでに適用されていてはいけません。

スコープ名

― 適用するスコープの名前を指定します。

― 同じスコープ名を複数指定できません。

認可識別子

― スコープを適用する利用者の認可識別子を指定します。

― 認可識別子は、18文字以内の先頭が英字で始まる英数字、または9文字以内の日本語文字列を指定します。

― ある認可識別子に対してスコープが適用されている場合、スコープ定義文のDSI名リストのDSIを1つでも含む表は、指定されたDSIがデータ操作の範囲となります。スコープ定義文のDSI名リストのDSIを1つも含まない表は、すべてのDSIがデータ操作の範囲となります。

― ある認可識別子に対して複数のスコープが適用されている場合、その認可識別子に適用されているスコープで指定されたすべてのDSIがデータ操作の範囲となります。

使用例

例

利用者YAMADAにスコープ“東京ＳＣＰ”を適用します。

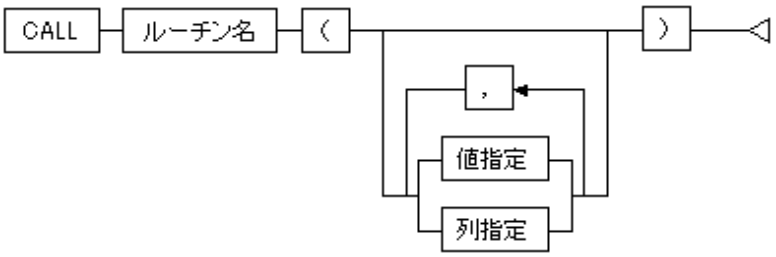
```
APPLY SCOPE 東京ＳＣＰ TO YAMADA
```

3.5 CALL文

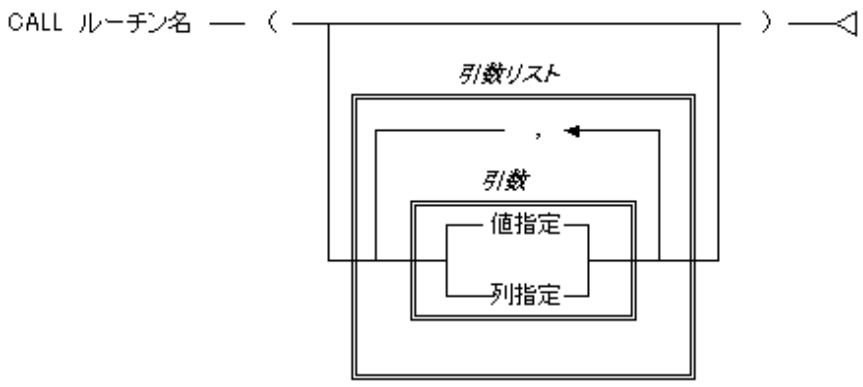
機能

プロシジャルーチンの呼出しを行います。

記述形式



構文の構成



参照項番

- 値指定 → “2.3 値指定と相手指定”
- 列指定 → “2.4 列指定”

権限

- CALL文を実行できるのは、そのプロシジャルーチンのEXECUTE権の保持者です。

一般規則

ルーチン名

- 呼び出すプロシジャルーチンの名前を指定します。
- ルーチン名はスキーマ名で修飾した識別子で記述しますが、スキーマ名は省略できる場合があります。スキーマ名修飾の詳細は、“D.2 名前に関する注意事項”を参照してください。

引数

- 引数は、プロシジャルーチン定義時のパラメタのデータ型に対して代入できることが必要です。
- 引数の値指定には定数、USER、SQL変数、パラメタ変数、埋込み変数または動的パラメタ指定のいずれかを指定します。
- プロシジャルーチン定義のパラメタモードにOUT、またはINOUTを指定した場合、引数の値指定にはSQL変数名、パラメタ変数名、埋込み変数名、動的パラメタ指定以外を指定することはできません。
- CALL文を被トリガSQL文に指定する場合のみ、引数に列指定を指定することができます。
- 指定する引数の個数は、プロシジャルーチン定義で指定したパラメタ宣言の個数に一致していることが必要です。
- プロシジャルーチン定義時のパラメタの順番に引数を指定することが必要です。
- CALL文が異常終了する場合、パラメタモードがOUTまたはINOUTのパラメタの値は呼出し元に返却されません。

プロシジャルーチンのトランザクションの扱い

- CALL文を実行することによってトランザクション区間が区切れることはなく、プロシジャルーチン内に記述されているSQL手続き文はCALL文の直前のトランザクションの状態で実行されます。
- プロシジャルーチン内でCOMMIT文またはROLLBACK文が実行されると、CALL文以前から継続しているトランザクションは完了します。
- トリガから実行されるプロシジャルーチンでは、COMMIT文またはROLLBACK文は実行できません。
- プロシジャルーチン内のSQL文の実行でエラーが起きた場合は、エラーを起こしたSQL文が無効となります。ただし、SQLSTATE値の例外コードが40の場合は、トランザクションはロールバックされます。
- プロシジャルーチン内でのSQL文の実行時は、動作環境ファイルのTRAN_SPECの指定は無効となります。アプリケーションに復帰した後は、TRAN_SPECの指定に従って制御されます。

DESCRIBE情報について

- 値指定に動的パラメタ指定が指定された場合のDESCRIBE情報は、対応するプロシジャルーチンのパラメタのデータ型になります。

  備考. CALL文で指定された動的パラメタ指定のDESCRIBE情報は、以下のように設定されます。

  - 対応するプロシジャルーチンのパラメタのパラメタモードがINまたはINOUTの場合は、入力DESCRIBE文でDESCRIBE情報が設定されます。
  - 対応するプロシジャルーチンのパラメタのパラメタモードがOUTまたはINOUTの場合は、出力DESCRIBE文でDESCRIBE情報が設定されます。

使用例

例1

アプリケーションからスキーマ名SCM1で修飾してルーチンPRC01を呼び出します。パラメタモードがINのパラメタに対して埋込み変数名を指定します。

アプリケーションの記述内容

```
EXEC SQL BEGIN DECLARE SECTION;
char  SQLSTATE[6];
Long  H1;
char  H2[21];
EXEC SQL END DECLARE SECTION;

H1 = 1000000;
strcpy(H2,"Shinkansen");
EXEC SQL CALL SCM1.PRC01(:H1,:H2);
・・・
```

ルーチンPRC01の定義内容

```
CREATE PROCEDURE SCM1.PRC01(
        IN  P1        INTEGER,
        IN  P2        CHAR(20)
)
BEGIN
        DECLARE SQLSTATE CHAR(5);
        DECLARE SQLMSG   CHAR(255);
        INSERT INTO SCM1.TBL1(CLM_INT,CLM_CHAR) VALUES(P1,P2);
END
```

例2

ルーチンPRC02からスキーマ名“在庫管理”で修飾してルーチン“テレビ在庫確認”を呼び出します。パラメタモードがINおよびOUTのパラメタに対してパラメタ名およびSQL変数名を指定します。

ルーチンPRC02の定義内容

```
CREATE PROCEDURE SCM1.PRC02(
        IN  Ｐ倉庫番号  SMALLINT,
        OUT Ｐ在庫数量  INTEGER
)
BEGIN
        DECLARE SQLSTATE    CHAR(5);
        DECLARE SQLMSG      CHAR(255);
        DECLARE Ｖ製品番号  SMALLINT    DEFAULT 100;
        DECLARE Ｖ製品名    NCHAR(10)   DEFAULT N'テレビ';

        CALL 在庫管理.テレビ在庫確認(
                Ｖ製品番号,Ｖ製品名,Ｐ倉庫番号,Ｐ在庫数量);
END
```

ルーチン“テレビ在庫確認”の定義内容

```
CREATE PROCEDURE 在庫管理.テレビ在庫確認(
        IN  検索番号    SMALLINT,
        IN  検索製品名  NCHAR(10),
        IN  検索倉庫    SMALLINT,
        OUT 結果数量    INTEGER
)
BEGIN
        DECLARE SQLSTATE CHAR(5);
        DECLARE SQLMSG   CHAR(255);
```

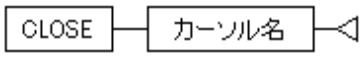
```
        SELECT 在庫数量 INTO 結果数量 FROM S1.在庫表
                WHERE 製品番号 = 検索番号 AND 製品名 = 検索製品名
                  AND 倉庫番号 = 検索倉庫;
        IF SQLSTATE <> '00000' THEN
                SET 結果数量 = NULL;-- 検索できなかった場合はNULL値を返却
        END IF;
END
```

3.6 CLOSE文

機能

カーソルを閉じることによってカーソルの機能を無効にします。

記述形式



権限

・ CLOSE文を実行できるのは、カーソルの処理対象の表に対するSELECT権の保持者です。

一般規則

・ CLOSE文で指定するカーソルは、開かれた状態であることが必要です。閉じられた状態ならば、例外(不当カーソル状態)となります。

カーソル名

ー カーソルの名前を指定します。

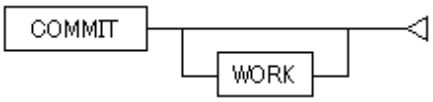
ー カーソル名は、同一コンパイル単位に含まれるカーソル宣言で定義されていることが必要です。

3.7 COMMIT文

機能

現行のデータベースの変更をすべて有効にしてトランザクションを終了します。

記述形式



一般規則

・ オープン中のカーソルは、以下のようになります。

ー SET TRANSACTION文でREAD COMMITTEDが指定され、カーソル宣言でFOR READ ONLYが指定された場合およびカーソル宣言でカーソルモードを指定した場合、COMMIT文が実行されてもカーソルはクローズされずに継続します。そうでない場合、COMMIT文が実行されるとカーソルはクローズされます。

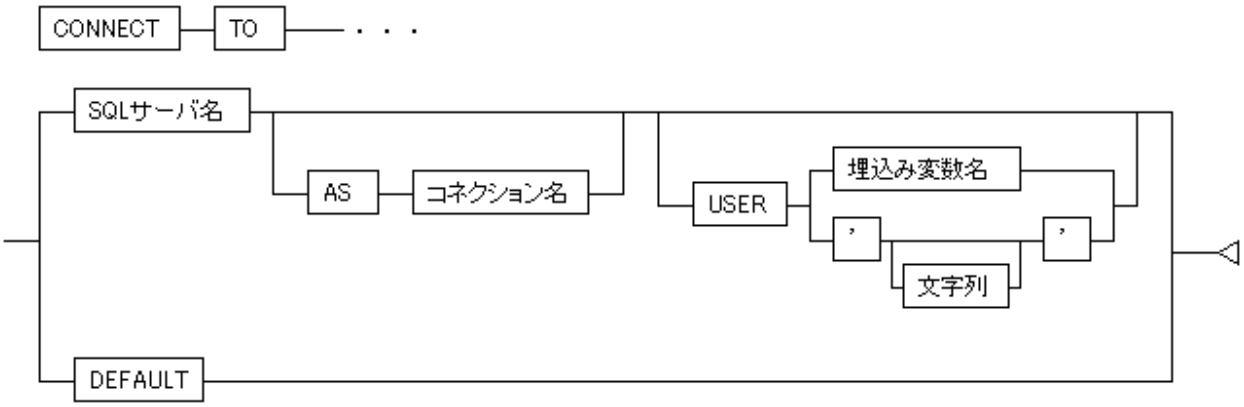
・ WORKは省略することもできます。

3.8 CONNECT文

機能

コネクションを接続します。

記述形式



参照項番

- 埋込み変数名 → “6.5 SQL埋込みホストプログラム”
- 文字 → “2.1.1 文字”

一般規則

- CONNECT文の実行により、サーバとのコネクションを接続します。
- CONNECT文が正常終了すると、現コネクションは、その時に接続されたコネクションとなります。すなわち、複数のコネクションを接続した場合、最後に実行したCONNECT文のコネクションが現コネクションとなります。
- 同じコネクション名を持つ複数のコネクションを接続することはできません。また、DEFAULTを指定した複数のコネクションを接続することもできません。
- SQLサーバ名、コネクション名、ログイン名およびパスワードの前方および後方の空白は無視します。たとえば、SQLサーバ名“ SV1”の記述は“SV1”と等価です。

SQLサーバ名

- SQLサーバ名は、ローカルにあるSymfoware/RDBと接続する場合とリモートにあるSymfoware/RDBと接続する場合で、指定する内容が異なります。
  - ローカルのSymfoware/RDBと接続する場合(マルチRDB運用でない場合)は、データベース名を指定します。
  - ローカルのSymfoware/RDBと接続する場合(マルチRDB運用の場合)は、“[RDBシステム名.]データベース名”を指定します。RDBシステム名には、データベースの存在するSymfoware/RDBのシステム名を指定します。省略した場合は、環境変数RDBNAMEに指定されたRDBシステム名が有効となります。
  - リモートのSymfoware/RDBと接続する場合は、任意の文字列を指定することができます。ただし、同じSQLサーバ名を持つSERVER_SPECがクライアント用の動作環境ファイルに指定されていることが必要です。
- SQLサーバ名は、文字列定数または文字列型の埋込み変数で指定します。

コネクション名

- コネクション名は、複数のコネクションを接続する場合に、コネクションを識別する名前を指定します。ただし、コネクション名は一意であることが必要です。
- コネクション名を省略した場合、SQLサーバ名がコネクション名になります。
- コネクション名は、文字列定数または文字列型の埋込み変数で指定します。

USER(ユーザ指定)

- サーバに接続する利用者を指定します。利用者は、埋込み変数または文字列で認可識別子とパスワードを斜線“/”で区切って指定します。ユーザ指定を省略した場合は、クライアント用の動作環境ファイルの実行パラメタ“DEFAULT_CONNECTION”に指定された認可識別子とパスワードでサーバに接続します。
- 認可識別子は、18文字以内の先頭が英字で始まる英数字、または9文字以内の日本語文字列を指定します。
- 以下の場合、本指定は不要です。

  この場合、アプリケーション実行時のログイン名が有効となります。

  - ローカルのSymfoware/RDBと接続する場合
  - 接続先ホスト名に自端末のIPアドレス、自端末のホスト名、“localhost”またはループバックアドレスを指定し、リモートのSymfoware/RDBと接続する場合
- Symfoware/RDBの認証機構を使用し、ローカルのSymfoware/RDBと接続する場合は、ユーザ指定を省略できません。
- リモートのSymfoware/RDBと接続するとき、ユーザ指定を省略した場合は、クライアント用の動作環境ファイルの実行パラメタ“DEFAULT_CONNECTION”に指定された認可識別子とパスワードでサーバに接続します。

DEFAULT

- DEFAULTを指定した場合は、クライアント用の動作環境ファイルの実行パラメタ“DEFAULT_CONNECTION”に指定された情報が有効となります。
- ローカルのSymfoware/RDBと接続する場合、本指定に関係なく実行時のログイン名とパスワードでサーバに接続します。

利用者の認証

不当な利用者がサーバに接続することを抑止するため、サーバに接続する時に利用者の認証が行われます。認証の方式は、以下の接続形式により異なります。

- OSのログイン名で接続する場合
- データベース専用利用者名で接続する場合

OSのログイン名で接続する場合

S L

- サーバに接続する利用者の認可識別子とパスワードは、以下に示すとおり、サーバにログイン可能で、かつデータベースの使用権限があるログイン名およびパスワードでなければなりません。

  ・ログイン名が有効な状態である

  ・ログイン名が有効期限内である

  ・ログイン名のパスワードがサーバシステムのパスワードと一致している

  ・ログイン名のサーバシステムのパスワードが有効期限内である

  注意

  OSのログイン名のパスワードが有効期限内であるかは、OSに依存します。また、パスワード有効期限の日数を設定した基準日には、OSによって以下の違いがあります。

  ・Solarisの場合：パスワードの設定日を日数にカウントする

  ・Linuxの場合：パスワードの設定日を日数にカウントしない

W

- サーバに接続する利用者の認可識別子とパスワードは、サーバにログイン可能で、かつデータベースの使用権限があるログイン名およびパスワードでなければなりません。

データベース専用利用者名で接続する場合

- サーバに接続する利用者は、CREATE USER文で登録した利用者で、かつデータベースの使用権限がある利用者でなければなりません。
- CREATE USER文でデータベース専用利用者として登録した利用者は、登録した認可識別子とパスワードを指定します。
- CREATE USER文でOSの利用者として登録した利用者は、サーバにログイン可能なログイン名およびパスワードを指定します。

ログイン名とパスワード

ログイン名とパスワードは、各サーバでは以下のように扱われます。

S L

Solaris/Linuxの場合
ログイン名: ログイン名
パスワード: ログイン名のパスワード

W

Windowsの場合
ログイン名: ユーザ名
パスワード: ユーザ名のパスワード

使用例

例1

ローカルのSymfoware/RDBと接続する場合(マルチRDB運用の場合)

```
CONNECT TO 'SYS1.DB01'
```

例2

ローカルのSymfoware/RDBと接続する場合(マルチRDB運用ではない場合)

```
CONNECT TO 'DB01'
```

例3

リモートのSymfoware/RDBと接続する場合

```
CONNECT TO 'SV1' USER 'USER1/777###FF'
```

備考. コネクション名を省略しているためコネクション名は、SQLサーバ名と同じ“SV1”となります。

例4

リモートのSymfoware/RDBと接続する場合で、SQLサーバ名、ログイン名およびパスワードをホスト変数で指定します。
なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR server[13];
    VARCHAR user[26];
EXEC SQL END DECLARE SECTION;
            :
    strcpy(server.sqlvar,"SV1");
    server.sqllen = strlen(server.sqlvar);
    strcpy(user.sqlvar,"USER1/777###FF");
    user.sqllen = strlen(user.sqlvar);
EXEC SQL CONNECT TO :server USER :user;
```

備考. コネクション名を省略しているためコネクション名は、SQLサーバ名と同じ“SV1”となります。

3.9 CREATE DATABASE文(データベース定義)

機能

データベース名を登録します。

記述形式

参照項番

・ 日本語文字列 → “2.1.3 トークン”

一般規則

データベース名

ー 登録するデータベースの名前を指定します。

ー データベース名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

ー 登録するデータベースの、データベース名を指定します。

ー データベース名は、Symfoware/RDBで一意の名前であることが必要です。

使用例

例

データベース“RDBDB1”を定義します。

```
CREATE DATABASE RDBDB1
```

3.10 CREATE DBSPACE文(データベーススペース定義)

機能

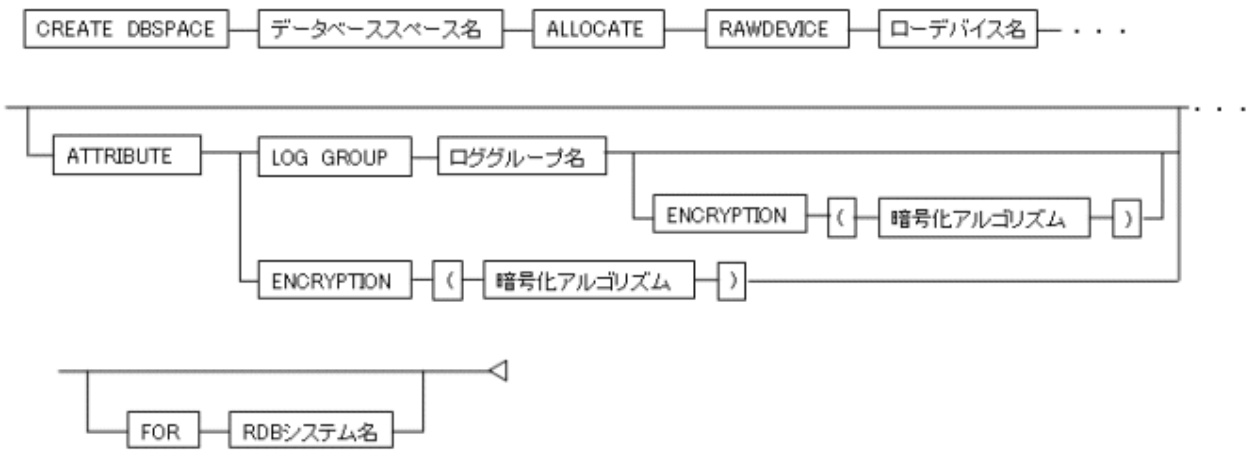
データベーススペースを作成します。

データベース簡単運用の場合は利用できません。

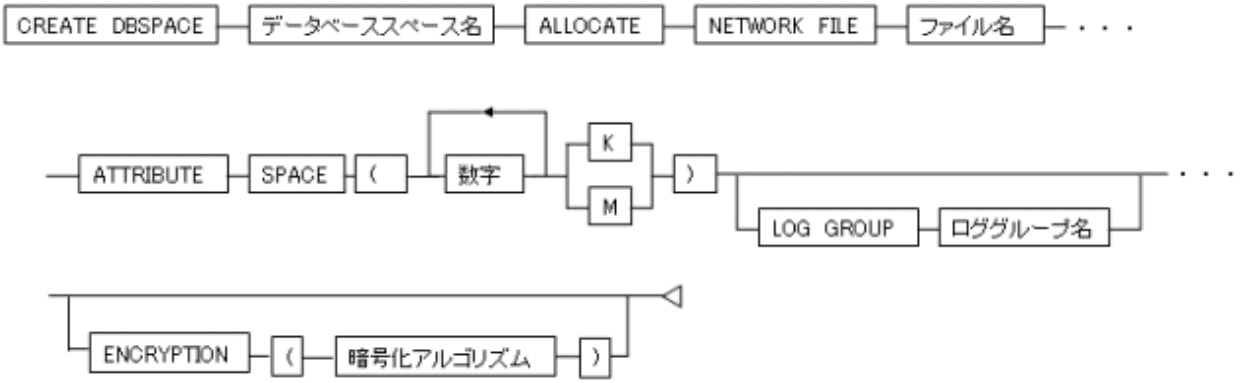
記述形式

S L

RAWDEVICE指定の場合



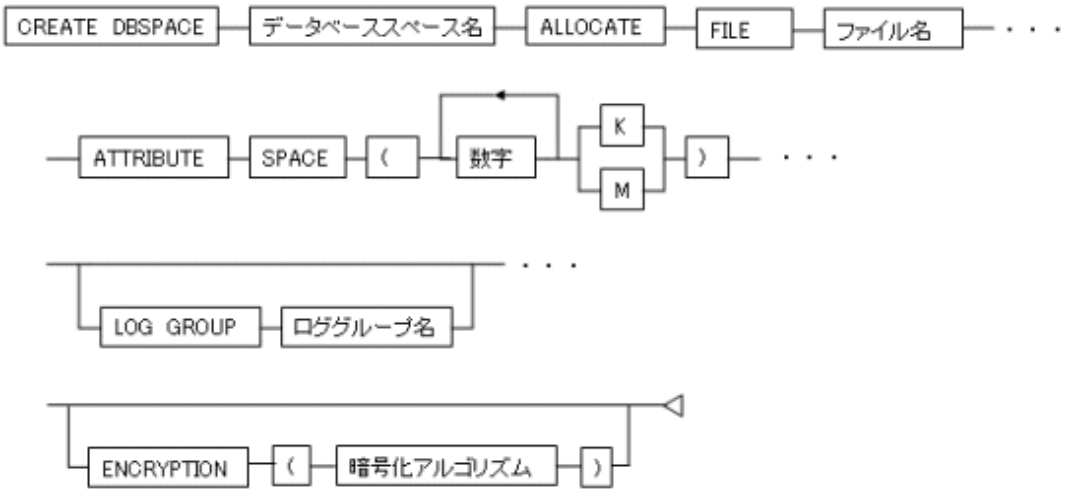
NETWORK FILE指定の場合



FILE指定の場合

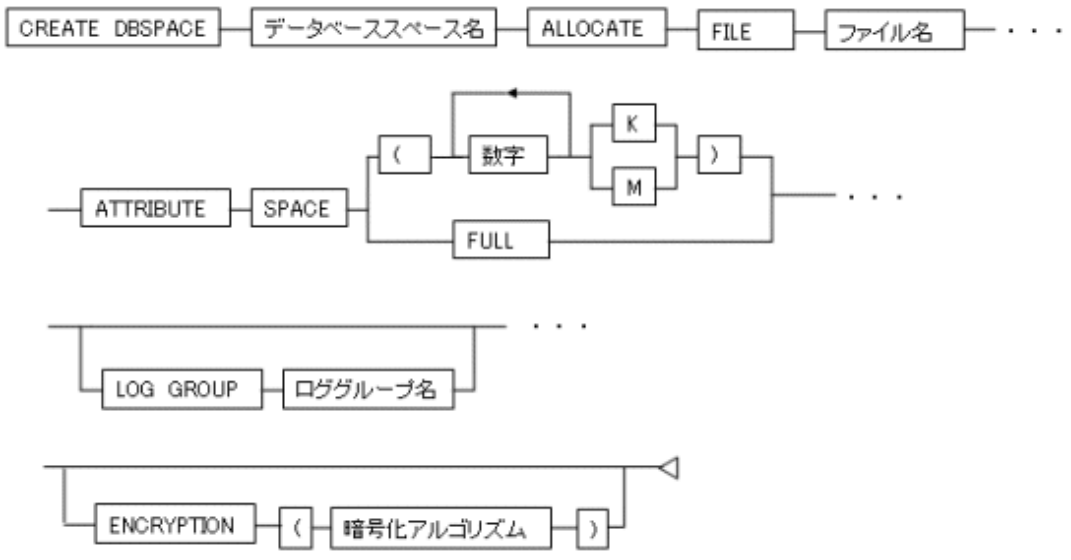
S L

Solaris/Linuxの場合



W

Windowsの場合



参照項番

- 数字 → “2.1.1 文字”
- 日本語文字列 → “2.1.3 トークン”

S L 一般規則(Solaris/Linuxの場合)

データベーススペース名

- 作成するデータベーススペースの名前を指定します。
- データベーススペース名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- データベーススペース名は、データベース内で一意の名前であることが必要です。

ロググループ名

- ロググループ名は、スケーラブルログ機能を利用する場合に、データベーススペースが使用するユーザログループの名前を指定します。
- ロググループ名は、英数字、“_”および各国語文字から構成される18文字以内の文字列を指定します。各国語文字の指定時でSymfoware/RDBの文字コード系がUNICODEの場合は12文字以内です。
- ロググループ名に“system”を指定した場合、またはロググループ名を省略した場合は、システムロググループ名が指定されたものとみなします。
- ロードシェア運用の場合は、ロググループ名を省略することはできません。ロードシェアシステム全体で一意となるように指定してください。

参照

スケーラブルログ機能の詳細については、“RDB運用ガイド”を参照してください。

暗号化アルゴリズム

- 格納するデータの暗号化アルゴリズムを指定します。

  AES256:

  暗号化キー長256ビットのAES暗号化アルゴリズムで暗号化します。

ローデバイス名

- データベーススペースをローデバイスに作成する場合、ローデバイス名は、絶対パス名で指定します。ローデバイス名に指定できる長さは、255バイト以内です。
- データベーススペースをディスク内のパーティションに作成するために、そのパーティションのローデバイス名を指定します。指定するローデバイスは、すでに作成済であることが必要です。データベーススペースの容量は、このパーティションの大きさとなります。
- 指定するローデバイスは、あらかじめSymfoware/RDBの起動者(rdbstartコマンドの実行者)の読込み許可および書込み許可を設定して作成してください。ローデバイスにこの許可を設定していない場合、エラーとなります。

RDBシステム名

- クラスタシステム上に登録されているRDBシステム名のみが指定できます。
- RDBシステム名を省略した場合は、CREATE DBSPACE文を実行したRDBシステムに対して、データベーススペースを定義します。
- ロードシェア機能は、Symfoware Server Enterprise Extended Editionの場合に使用できます。
- RDBシステム名は、9. x以前のロードシェア機能を利用する場合に、データベーススペースを定義するRDBシステム名を指定します。その場合、指定したローデバイスがそのRDBシステム上に存在しなければなりません。

参照

ロードシェア機能の詳細については、“クラスタ導入運用ガイド”を参照してください。

ファイル名

- データベーススペースをネットワークファイルまたはファイルに作成する場合、ファイル名は、ネットワークファイル名またはファイル名を絶対パス名で指定します。ファイル名に指定できる長さは、255バイト以内です。
- ネットワークファイルを指定する場合は、あらかじめNFSでマウントしておく必要があります。NFSマウントの方式は、ソフトマウントを指定してください。

  ### 参照

  ALLOCATE句でNETWORK FILEを指定して、データベーススペースをネットワーク上のストレージデバイスに作成する場合の詳細については、“セットアップガイド”を参照してください。

- ファイル名は、各オペレーティングシステムの文法に従って記述します。ただし、以下の文字を指定することはできません。
  - 空白
  - 改行
  - シングルクォーテーション
  - ダブルクォーテーション
  - セミコロン

ATTRIBUTE SPACE

- 作成するデータベーススペースのサイズを指定します。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1 キロバイトは1024バイト、1メガバイトは1024 キロバイトです。

一般規則(Windowsの場合)

データベーススペース名

- 作成するデータベーススペースの名前を指定します。
- データベーススペース名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- データベーススペース名は、データベース内で一意の名前であることが必要です。

ロググループ名

- ロググループ名は、スケーラブルログ機能を利用する場合に、データベーススペースが使用するユーザロググループの名前を指定します。
- ロググループ名は、英数字、“_”および各国語文字から構成される18文字以内の文字列を指定します。各国語文字の指定時でSymfoware/RDBの文字コード系がUNICODEの場合は12文字以内です。
- ロググループ名に“system”を指定した場合、またはロググループ名を省略した場合は、システムロググループ名が指定されたものとみなします。

参照

スケーラブルログ機能の詳細については、“RDB運用ガイド”を参照してください。

暗号化アルゴリズム

― 格納するデータの暗号化アルゴリズムを指定します。

AES256:

暗号化キー長256ビットのAES暗号化アルゴリズムで暗号化します。

ファイル名

― ALLOCATE句でFILEを指定する場合の規則は、以下のとおりです。

- ファイル名は、データベーススペースをNTFSに作成する場合に、NTFSファイル名を絶対パス名で指定します。ファイル名に指定できる長さは、255バイト以内です。
- ファイル名は、以下の形式で記述します。

```
ドライブ名:¥ファイル名
```

― ALLOCATE句でNETWORK FILEを指定する場合の規則は、以下のとおりです。

- ファイル名は、データベーススペースをネットワークファイルに作成する場合に、ファイル名を、絶対パス名で指定します。ファイル名に指定できる長さは、255バイト以内です。
- ファイル名は、以下の形式で記述します。

```
¥¥マシン名¥共有フォルダ名¥ファイル名
```

例

“RDBSV”というマシンの共有フォルダ“RDB2”に“DBSP01”という名前で作成する場合は以下の指定となります。

```
¥¥RDBSV¥RDB2¥DBSP01
```

注意

ネットワークファイル名としてドライブ名を指定した記述方法で指定することはできません。つまりネットワークドライブを割り当ててそのドライブ名を指定した記述方法では指定できません。

- ネットワークファイルを指定する場合にファイルを作成する共有フォルダをネットワークドライブとして割り当てる必要はありません。

― ファイル名は、各オペレーティングシステムの文法に従って記述しますが、それ以外に以下の文字を指定することはできません。

- 空白
- 改行
- シングルクォーテーション
- ダブルクォーテーション
- セミコロン

ATTRIBUTE SPACE

― 作成するデータベーススペースのサイズを指定します。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1 キロバイトは1024バイト、1メガバイトは1024 キロバイトです。

― FULLを指定した場合、指定したファイルのディレクトリの容量をデータベーススペースのサイズとして使用します。

― ATTRIBUTE SPACEにFULLを指定する場合は、ファイルはルートディレクトリ直下に作成してください。

使用例

例1

ローデバイスにデータベーススペース“RDBDBS1”を定義します。

S

Solarisの場合

```
CREATE DBSPACE RDBDBS1 ALLOCATE RAWDEVICE /dev/rdsk/c0t1d0s1
```

L

Linuxの場合

```
CREATE DBSPACE RDBDBS1 ALLOCATE RAWDEVICE /dev_symfoware/raw1
```

例2

ネットワークファイルにデータベーススペース“RDBS01”を定義します。

S L

Solaris/Linuxの場合

```
CREATE DBSPACE RDBS01
  ALLOCATE NETWORK FILE /RDB/DBS01
  ATTRIBUTE SPACE(500M)
```

W

Windowsの場合

```
CREATE DBSPACE RDBS01
  ALLOCATE NETWORK FILE ¥¥RDBSEV¥RDB2¥DBSP1
  ATTRIBUTE SPACE(2M)
```

例3

ファイルにデータベーススペース“RDBS01”を定義します。

S L

Solaris/Linuxの場合

```
CREATE DBSPACE RDBS01
  ALLOCATE FILE /RDB2/DBS01
  ATTRIBUTE SPACE(2M)
```

W

Windowsの場合

```
CREATE DBSPACE RDBS01
  ALLOCATE FILE C:¥RDB2¥DBSP1
  ATTRIBUTE SPACE(2M)
```

注意

- ロードシェア運用の場合、他ノードのRDBシステムまたは当該RDBシステムでRDBネットへの参入と離脱が行われていると、エラーとなります。
- ロードシェア運用の場合、ユーザログループの縮退または切り戻しが行われている場合、エラーとなります。

W

注意

- データベーススペースを作成するドライブのプロパティとして、“ドライブを圧縮してディスク領域を空ける”を設定しないでください。ドライブ圧縮を行った場合、通常のI/O処理と比較して性能上のオーバヘッドが発生します。これにより、I/O量に依存してOSリソース不足によるアクセスエラーが発生するためです。

- データベーススペースを作成するフォルダのプロパティとして[全般タブ]属性の[詳細設定]である、“圧縮属性または暗号化属性”を設定しないでください。
  圧縮属性とした場合、ドライブ圧縮と同様に通常のI/O処理と比較して性能上のオーバヘッドが発生します。これにより、I/O量に依存してOSリソース不足によるアクセスエラーが発生するためです。
  暗号化属性とした場合、Symfoware/RDBで使用する各資源の作成者（rdblogコマンド、rdbcrdicコマンドなどの実行ユーザ）とSymfoware Serverのサービス起動で動作するSymfoware Serverプロセスの実行者（ユーザ登録されないWindows(R)システムユーザ）が異なるため、暗号解除ができずアクセスエラーが発生するためです。

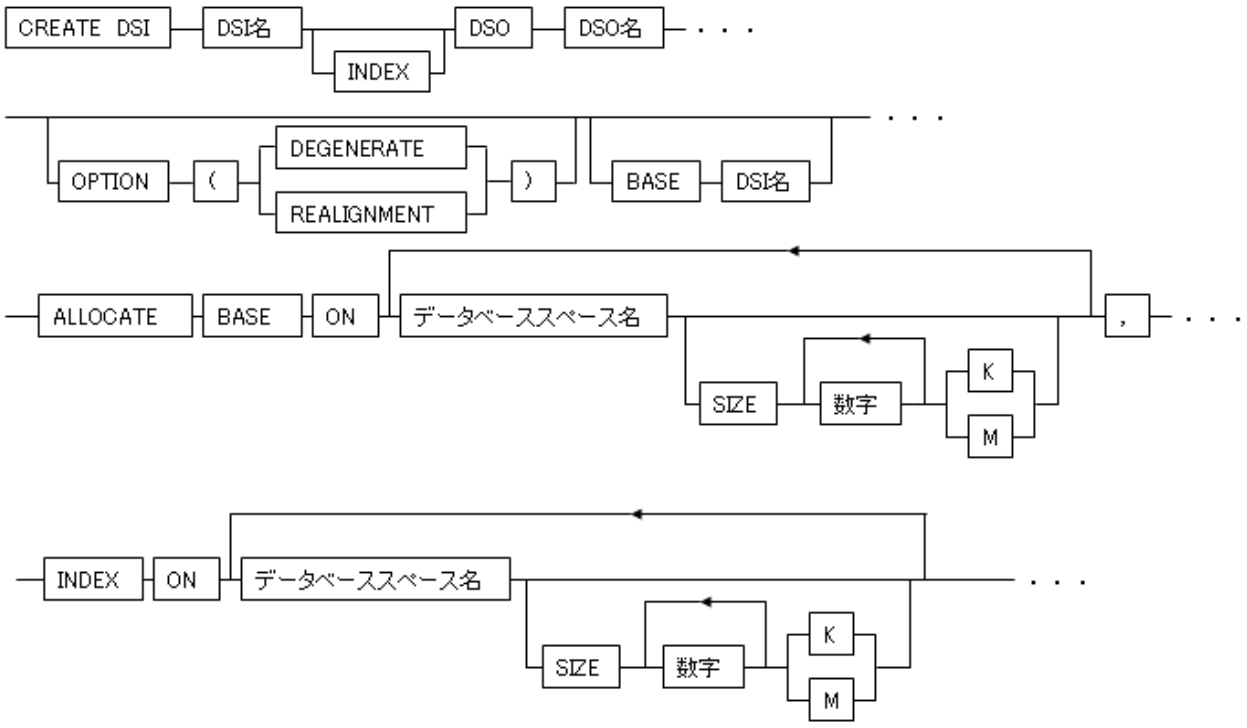
3.11 CREATE DSI文(インデックスのDSI定義文)

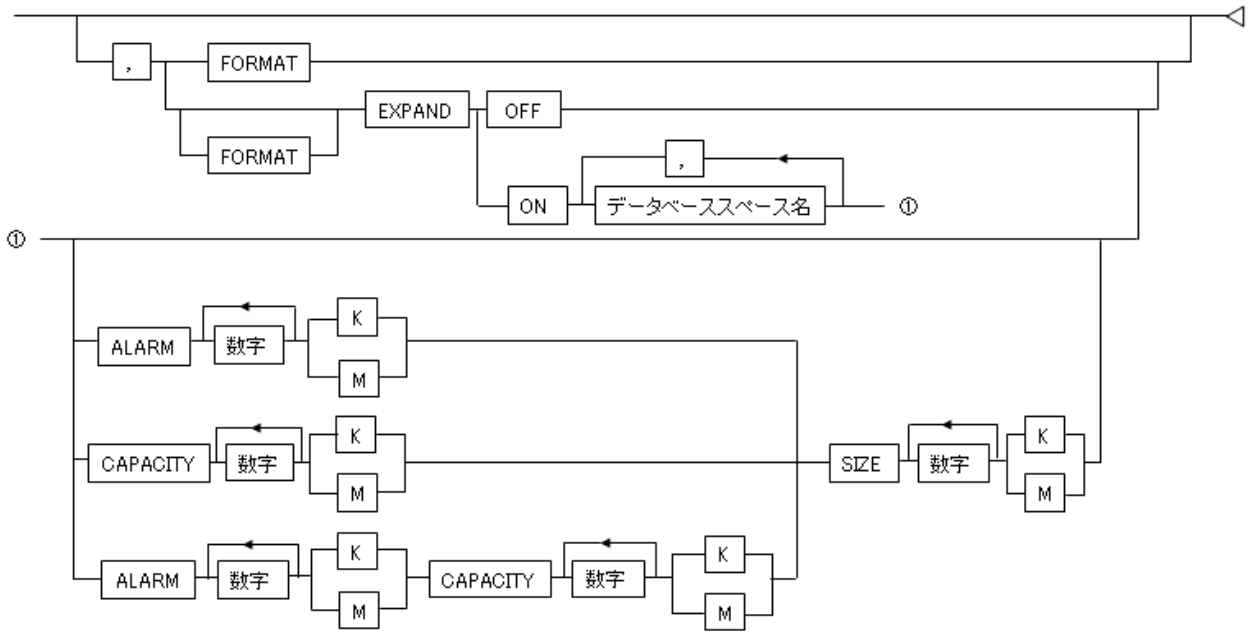
機能

表に付加するインデックスのデータを格納する領域を、データベーススペースに割り付けるために、DSIを定義します。

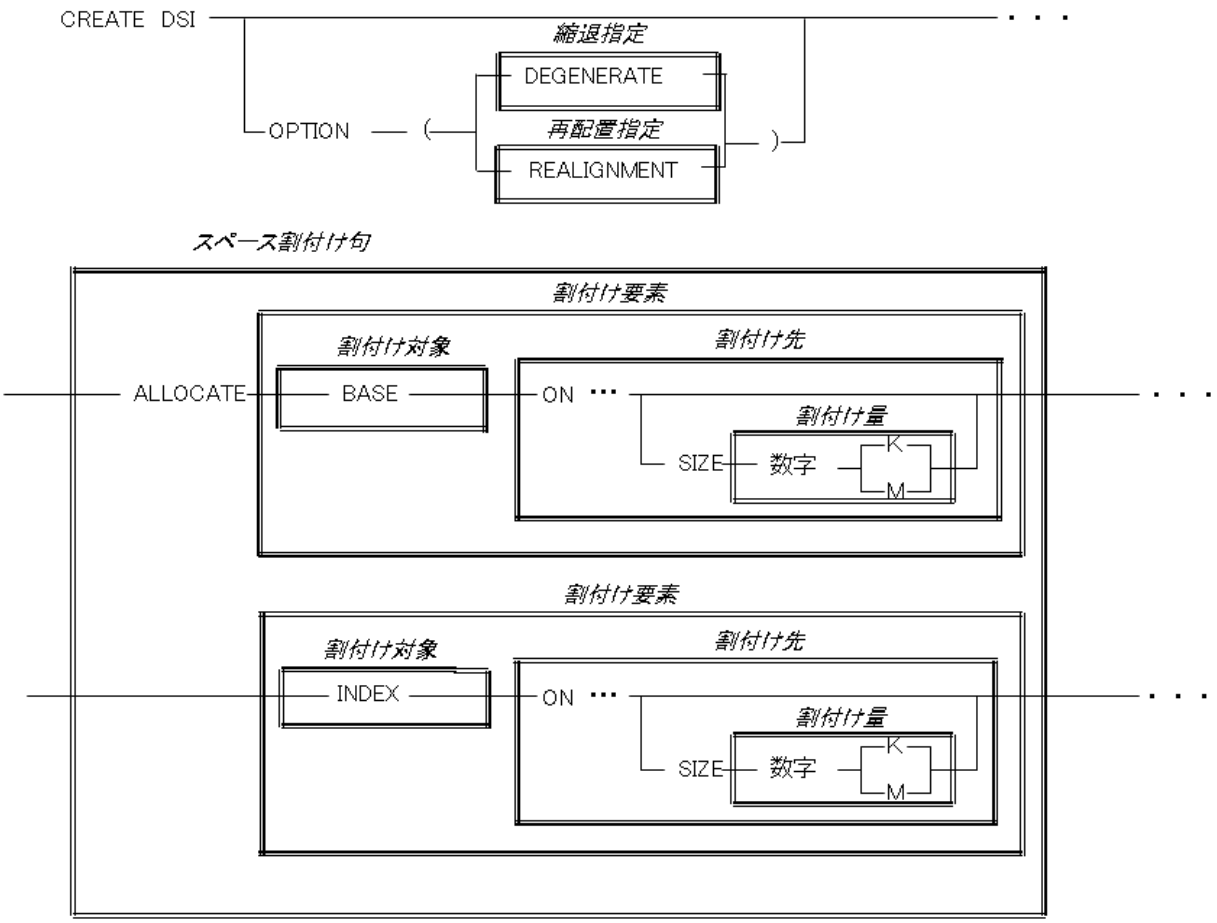
データベース簡単運用の場合は利用できません。

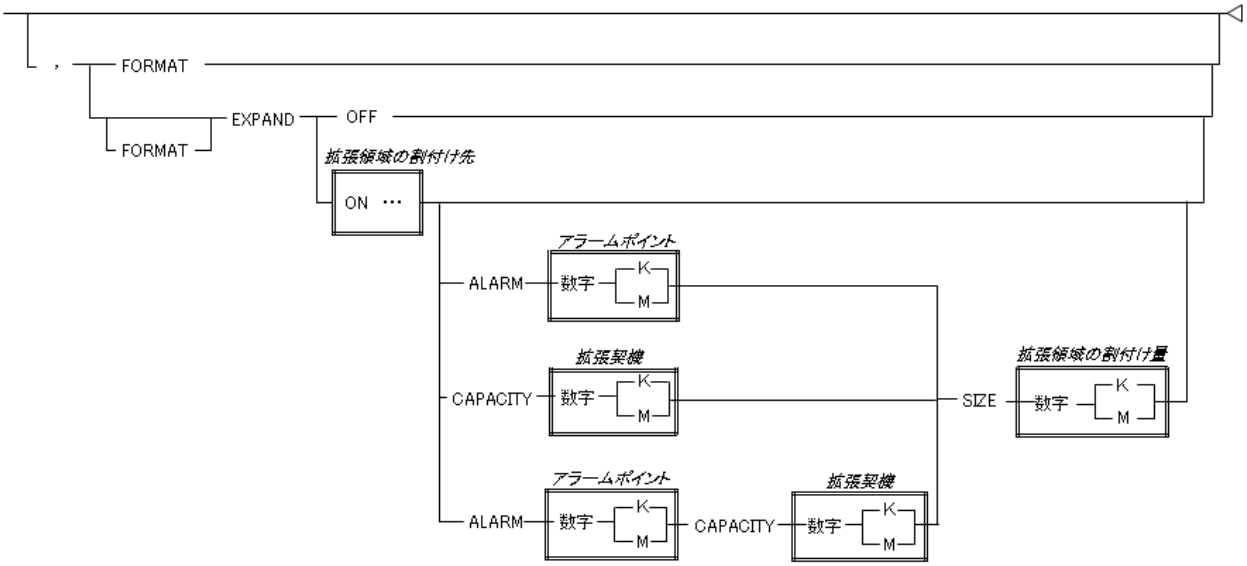
記述形式




,
FORMAT
FORMAT
EXPAND
OFF
ON
,
データベーススペース名
①
①
ALARM
数字
K
M
CAPACITY
数字
K
M
SIZE
数字
K
M
ALARM
数字
K
M
CAPACITY
数字
K
M

構文の構成


CREATE DSI
縮退指定
DEGENERATE
OPTION
(
再配置指定
REALIGNMENT
)
スペース割付け句
割付け要素
割付け対象
割付け先
ALLOCATE
BASE
ON
割付け量
SIZE
数字
K
M
割付け要素
割付け対象
割付け先
INDEX
ON
割付け量
SIZE
数字
K
M



参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- インデックスのDSIを定義できるのは、その表の定義者、スキーマの定義者またはその表のINDEX権の保持者です。また、データベーススペースに対してALLOCATE権が必要です。

一般規則

DSI名

- 作成するインデックスのDSIの名前を指定します。
- DSI名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- DSI名は、データベース内で一意の名前であることが必要です。

DSO名

- インデックスのDSI定義の対象とする表について、定義済であるインデックスのDSOの名前を指定します。

縮退指定(DEGENERATE)

- インデックスを縮退する場合に指定します。
- DEGENERATEを指定すると、行の削除によりインデックスのページ内に有効なデータがなくなった場合に、そのページを未使用として管理し、行の追加時に新たなページが必要になった時点で再使用を行います。DEGENERATEを指定しない場合には、有効なデータがなくなったページについてこの管理は行われません。

  DEGENERATEを指定することにより、インデックススペースの利用効率が向上し、インデックスの再創成の契機を減らすことが可能となります。
- DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表にインデックスを作成する場合、本オプションを指定することはできません。

再配置指定(REALIGNMENT)

- REALIGNMENTを指定すると、データの再配置を行うことでインデックスのアクセスを常に最適な状態に保ち、インデックスの再編成を不要にします。本機能は、24時間連続運転などの業務を中断して、インデックスの再編成が実施できない場合に指定します。
- DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表にインデックスを作成する場合、本オプションを指定することはできません。

BASE

- インデックスのDSIを定義する表に、分割条件を指定した表のDSOが定義されている場合、BASEオペランドを指定します。そうでない場合には、BASEオペランドは指定できません。
  - BASEオペランドのDSI名には、インデックスを作成する表に対して定義済である、表のDSIの名前を指定します。このとき、インデックスのDSIは、ここで指定した表のDSIに対応するインデックスのデータの格納領域を定義します。

割付け対象(BASEおよびINDEX)

- 割付け対象には、BASEとINDEXを指定します。
- スペース割付け句で指定する割付け対象の意味を“表3.1 インデックスのDSIに指定する割付け対象”に示します。

表3.1 インデックスのDSIに指定する割付け対象

指定する割付け対象	意味
BASE	インデックスのDSIの、データ部にスペースを割り付ける。
INDEX	インデックスのDSIの、インデックス部にスペースを割り付ける。

参照

データ部およびインデックス部の構造については、“RDB運用ガイド(データベース定義編)”を参照してください。

割付け先(ON)

- 1つの割付け要素について、同じデータベーススペースを2回以上指定することはできません。

データベーススペース名

- インデックスのデータを格納するデータベーススペースの名前を指定します。
- 1つのDSIの割付け対象の割付け先として指定するデータベーススペース名は、同じロググループのデータベーススペースでなければなりません。また、関連する表のDSIで割り付けたデータベーススペースと同じロググループを使用するものでなければなりません。
- データを暗号化する場合は、以下の注意事項があります。
  - 1つまたは複数のDSIの割付け先として、複数のデータベーススペースを指定する場合、データベーススペースの暗号化の指定を統一してください。
  - 表のDSIおよびインデックスのDSIを定義する場合、データベーススペースの暗号化の指定は、統一する必要がありません。

割付け量(SIZE)

- 割付け量には、データベーススペース中に獲得する格納領域の大きさを、符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り上げます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。
- インデックスのDSIのデータ部およびインデックス部に対する割付け量は、それぞれのページ長の2倍以上を指定することが必要です。たとえば、データ部のページ長が2キロバイトの場合、データ部に対する割付け量は4キロバイト以上になるように指定します。インデックス部のページ長が1キロバイトの場合は、インデックス部に対する割付け量は2キロバイト以上になるように指定します。
- 割付け量を省略した場合のインデックスのDSIのデータ部およびインデックス部に対する割付け量は、それぞれDSO定義時のページ長の84倍とデータ部の割付け量の1/5倍になります。
- 割付け対象のデータベーススペースに、割付け量分の連続領域が存在しない場合、複数の領域に分割して割付けることがあります。複数の領域に分割されたかどうかは、rdbprtコマンドの出力結果で確認できます。DSI情報のAllocation informationのAllocate sizeに、指定した割付け量以下の値が表示された場合、複数の領域に分割して割付けられたと判断できます。

FORMAT

― インデックスのDSIを初期化する場合に指定します。

― 表のDSIが定義直後の場合、FORMATは指定できません。

― rdbddlexコマンドの準備モードで定義する場合、FORMATは指定できません。

EXPAND OFF

― アラームポイントおよび容量拡張定義を設定しない場合に指定します。

拡張領域の割付け先

― 同じデータベーススペースを2回以上指定することはできません。

― 1つのDSIの割付け対象の割付け先として指定するデータベーススペース名は、同じロググループを使用するデータベーススペースでなければなりません。

― データを暗号化する場合は、以下の注意事項があります。

- 1つまたは複数のDSIの割付け先として、複数のデータベーススペースを指定する場合、データベーススペースの暗号化の指定を統一してください。

- 表のDSIおよびインデックスのDSIを定義する場合、データベーススペースの暗号化の指定は、統一する必要がありません。

― 拡張領域の割付け先には、拡張量に指定した値以上の空き領域を用意してください。
また拡張領域の割付け先を省略した場合、DSIのデータベーススペースを拡張領域の割付け先とするため、DSIのデータベーススペースに拡張量に指定した値以上の空き領域を用意してください。

― 拡張領域の割付け先を省略した場合、省略値は以下となります。

- アラームポイント:【ベース部の割付け量 × 80%】以下となるページ長の倍数の最大値

- 拡張契機:0キロバイト

- 拡張量:1024キロバイト

拡張領域の割付け量

― 割付け量は、ベース部のページ長以上の値を指定してください。

― 割付け量には、データベーススペース中に獲得する格納領域の大きさを、符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り上げます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。

― 割付け量は、ページ長単位に繰り上げますので、ベース部のページ長の倍数で指定してください。

― インデックス部の割付け量は、ベース部の5分の1の値となります。ベース部の5分の1がインデックス部のページ長の倍数でない場合、インデックス部のページ長の倍数に繰り上げます。

― 割付け対象のデータベーススペースに、拡張量分の連続領域が存在しない場合、複数の領域に分割して割付けることがあります。複数の領域に分割されたかどうかは、rdbprtコマンドの出力結果で確認できます。DSI情報のAllocation informationのAllocate sizeに、指定した拡張量以下の値が表示された場合、複数の領域に分割して割付けられたと判断できます。

アラームポイント

― アラームポイントの契機をDSIの使用容量で指定します。

― アラームポイントは、ページ長以上の値を指定してください。

― アラームポイントには、DSIの使用量を符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り捨てます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。

― アラームポイントを省略した場合、アラームポイントは設定されません。

拡張契機

- 自動容量拡張の拡張契機をDSIの空き容量で指定します。
- 拡張契機には、DSIの空き容量を符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り上げます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。
- 拡張契機を省略した場合、拡張契機の値は“DSIの容量 - アラームポイント”になります。

使用例

例1

会社表のインデックスのDSIを定義します。インデックスのDSO定義は、“3.13 CREATE DSO文(インデックスのDSO定義文)”の例3で定義されています。

```
CREATE  DSI 会社表インデックスＤＳＩ  INDEX
        DSO 会社表インデックスＤＳＯ
        ALLOCATE BASE  ON DBSPACE001 SIZE 4000K,
                 INDEX ON DBSPACE002 SIZE  100K
```

例2

在庫表のインデックスのDSIを定義します。なお、在庫表は製品名と倉庫番号の値で分割格納するものとします。この表のDSO定義は、“3.14 CREATE DSO文(表のDSO定義文)”の例4で、インデックスのDSO定義は、“3.13 CREATE DSO文(インデックスのDSO定義文)”の例4で、また、対応する表のDSIは、“3.12 CREATE DSI文(表のDSI定義文)”の例3で、それぞれ定義されています。

```
CREATE  DSI 在庫表インデックスＤＳＩテレビ２  INDEX
        DSO 在庫表インデックスＤＳＯ  BASE  在庫表ＤＳＩテレビ２
        ALLOCATE BASE  ON DBSPACE001 SIZE 300K,
                 INDEX ON DBSPACE002 SIZE  50K
```

例3

会社表のインデックスのDSIを定義します。インデックスのDSO定義は、“CREATE DSO文(インデックスのDSO定義文)”の例3で定義されています。アラームポイントの契機をDSIの使用容量150キロバイト、自動容量拡張の拡張契機を空きページ容量0キロバイト、拡張量を1メガバイトとします。

```
CREATE  DSI 会社表インデックスＤＳＩ  INDEX
        DSO 会社表インデックスＤＳＯ
        ALLOCATE BASE  ON DBSPACE001 SIZE 200K,
                 INDEX ON DBSPACE002 SIZE  40K,
        EXPAND ON DBSPACE001 ALARM 150K CAPACITY OK SIZE 1M
```

注意

- ロードシェア運用の場合、他ノードのRDBシステムまたは当該RDBシステムでRDBネットへの参入と離脱が行われていると、エラーとなります。
- ロードシェア運用の場合、ユーザロググループの縮退または切り戻しが行われている場合、エラーとなります。
- FORMATを指定した場合、割付け量に応じて時間がかかります。
- FORMATを指定したインデックスのDSI定義文を実行する場合、インデックスのDSIの定義処理中にはDSIを自動容量拡張しません。表のデータ量を確認した上で、適切な割付け量を指定してください。
- 縮退指定(DEGENERATE)および再配置指定(REALIGNMENT)は、インデックスのDSI定義文およびインデックスのDSO定義文でそれぞれ指定が可能です。それぞれで異なる指定をした場合、インデックスのDSI定義文で指定した値が

有効になります。インデックスのDSI定義文に縮退指定(DEGENERATE)および再配置指定(REALIGNMENT)を省略した場合は、インデックスのDSO定義文の指定に従います。

3.12 CREATE DSI文(表のDSI定義文)

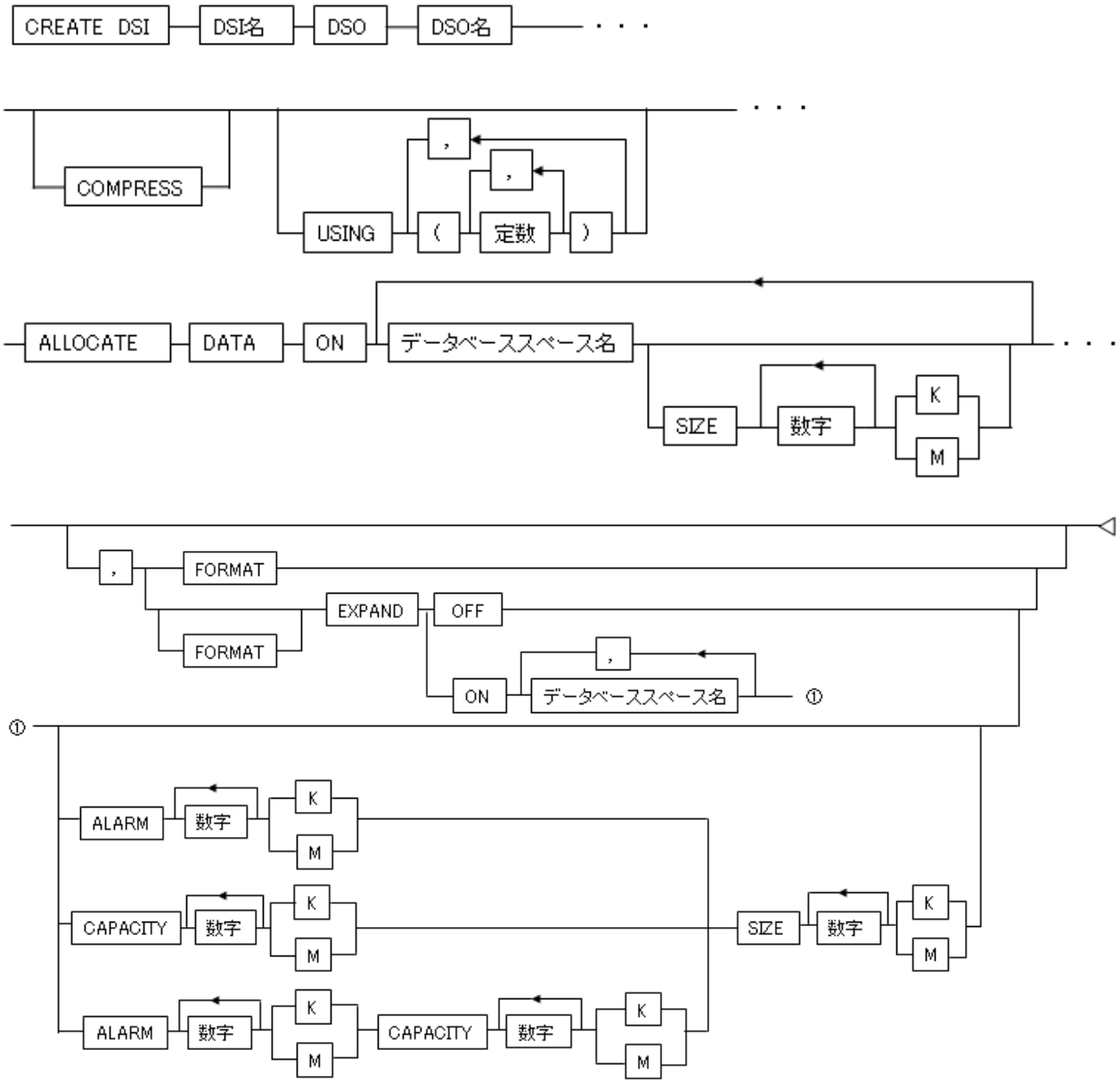
機能

表のデータを格納する領域を、データベーススペースに割り付けるために、DSI(Data Structure Instance)を定義します。

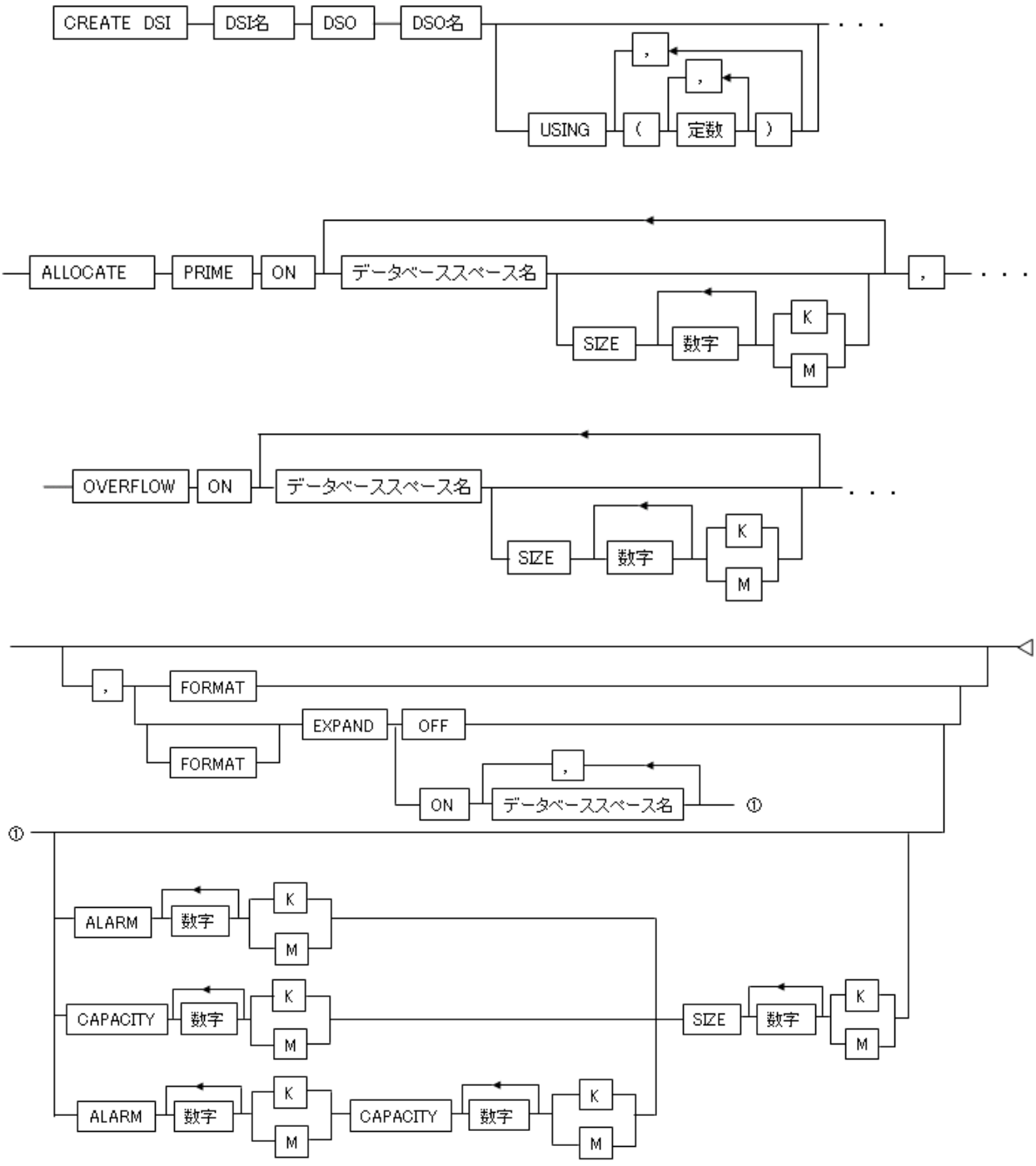
データベース簡単運用の場合は利用できません。

記述形式

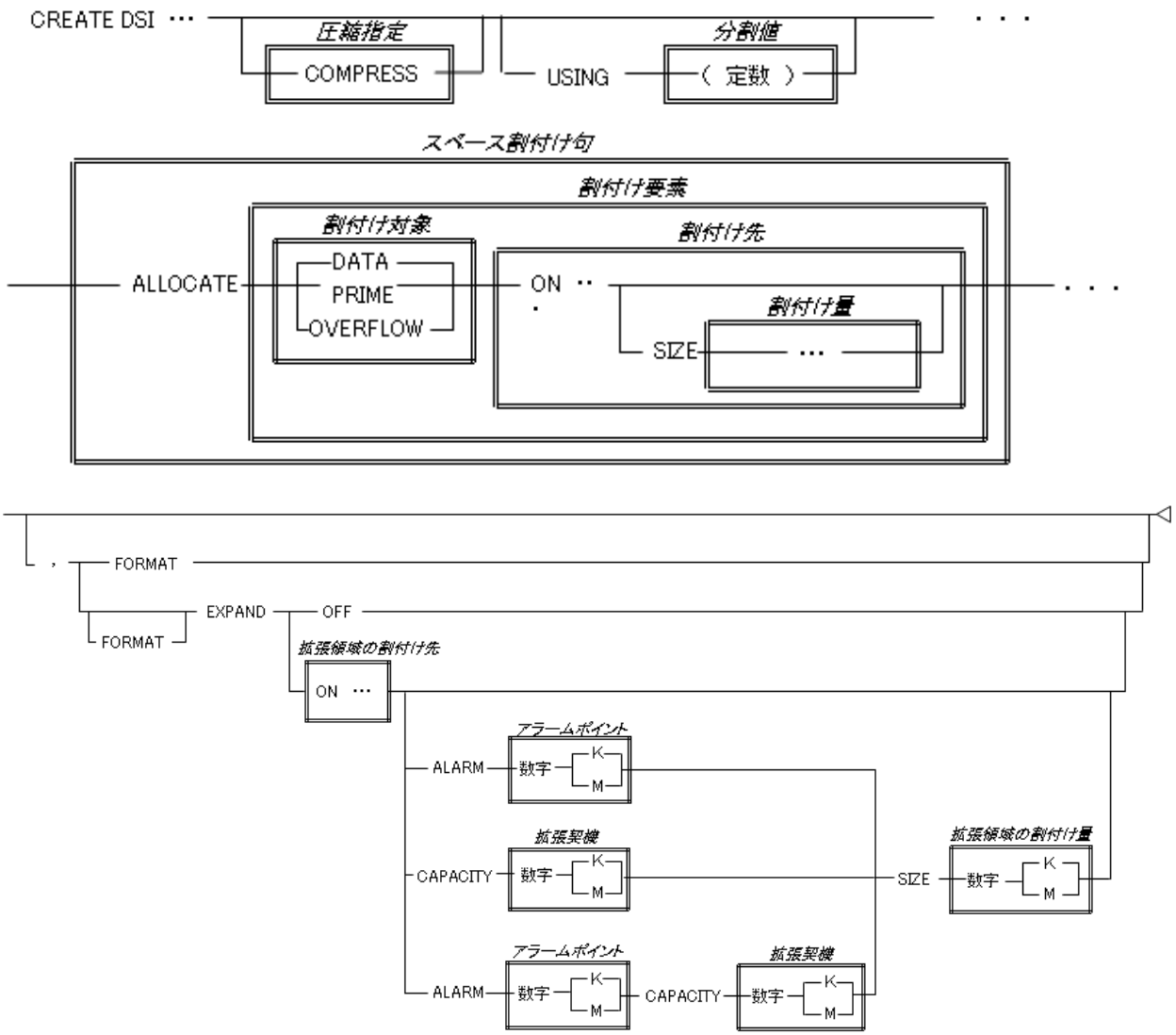
表のデータ構造がSEQUENTIALまたはOBJECTの場合



表のデータ構造がRANDOMの場合



構文の構成



参照項番

- 定数 → “2.1.2 定数”
- 日本語文字列 → “2.1.3 トークン”

権限

- 表のDSIを定義できるのは、その表の定義者、スキーマの定義者またはその表のALTER権の保持者です。また、データベーススペースに対してALLOCATE権が必要です。
- 未定義のDSIをスコープ定義したあとにそのDSI定義を行うとき、表のDSIの定義者とスコープ定義者が違う場合はエラーになります。

一般規則

DSI名

ー 作成する表のDSIの名前を指定します。

ー DSI名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

ー DSI名は、データベース内で一意の名前であることが必要です。

DSO名

― 表のDSI定義の対象とする表について、定義済である表のDSO名を指定します。

圧縮指定

― COMPRESSを指定した場合、表のDSIに対してデータの圧縮を行います。

― 表のデータ構造がSEQUENTIALの場合に指定可能です。

― Symfoware Server Enterprise Extended EditionまたはSymfoware Server Enterprise Editionを利用した場合のみ指定できます。

USING

― 表のDSO定義時に分割条件を指定した場合、USINGオペランドで分割値を指定します。そうでない場合には、USINGオペランドは指定できません。

- USINGオペランドの分割値は、DSO定義時に分割条件に指定した“?”に対する値を、定数で指定します。分割条件に複数の“?”が存在する場合には、その出現順序に対応して、定数をカンマ(,)で区切り指定します。分割条件に指定した“?”の個数と、定数の個数は同じであることが必要です。

- USINGオペランドの分割値に指定可能な定数の記述形式を以下に示します。

表3.2 分割値に指定可能な定数の記述形式

対応する列の属性	分割値に指定可能な定数の記述形式
SMALLINT	-32768～32767(小数点なし)
INTEGER	-2147483648 ～2147483647(小数点なし)
NUMERIC(p,q)	整数部(小数点の左側)の数字の数はp-q以下 (注) 小数部(小数点の右側)の数字の数はq以下
DECIMAL(p,q)	
CHARACTER(n)	長さがn以下の文字列定数
NATIONAL CHARACTER(n)	長さがn以下の各国語文字列定数
DATE	年から日までの10文字の日付
TIME	時から秒までの8文字の時刻
TIMESTAMP	年から秒までの19文字の時刻印
INTERVAL(年月)	年から月までの任意の単一または連続する日時フィールドをもつ時間隔
INTERVAL(日時)	日から秒までの任意の単一または連続する日時フィールドをもつ時間隔

p: 精度、q: 位取り、n: 長さ

注) 定数が小数点を含まない場合はすべて整数部となります。

- DSIによるデータベーススペースへの格納の単位は、DSO定義での分割条件と、USINGオペランドでの分割値の指定により、次のように決定されます。

[その1] 分割条件を“=”指定で行う場合

```
分割条件の指定:(製品名,倉庫番号) = (?,?)
分割値の指定  :(N'テレビ',2)
上記の場合、以下の条件式の結果が真になる行の集まりが格納の対象となります。
  格納対象    : 製品名 = テレビ  かつ  倉庫番号 = 2
```

[その2] 分割条件を“BETWEEN”指定で行う場合

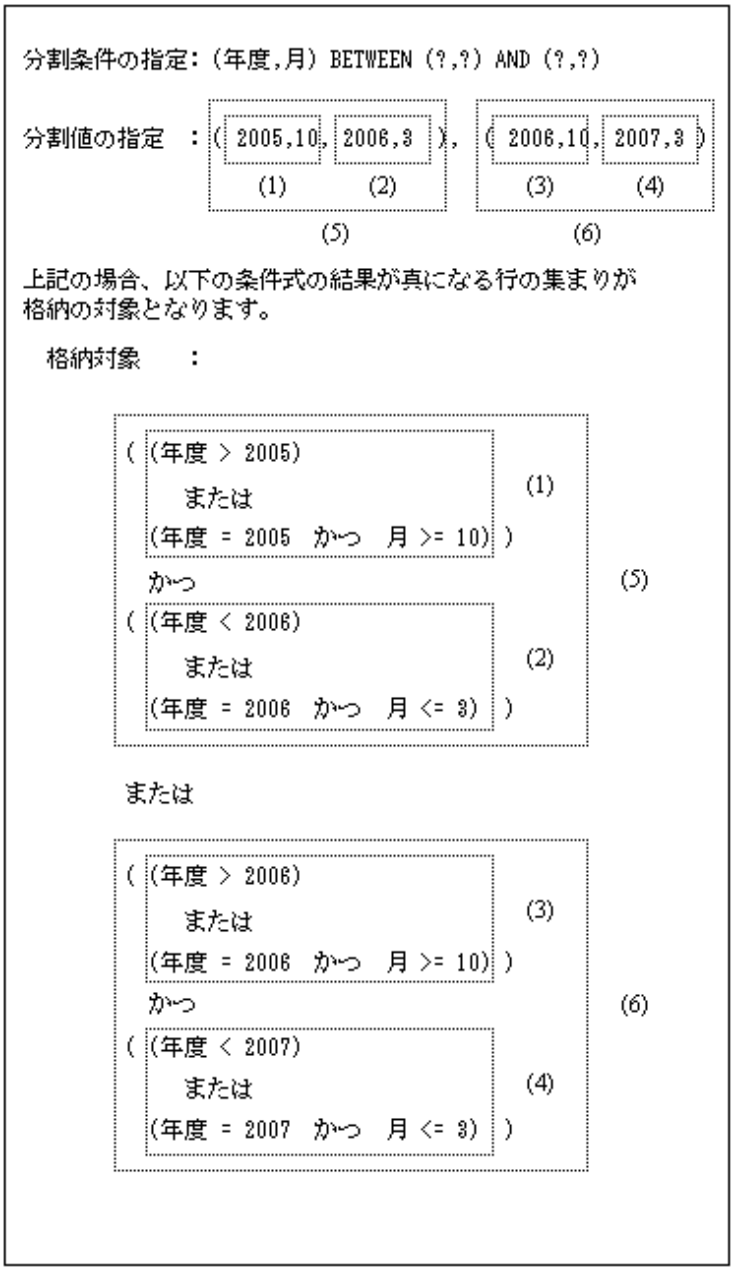
```
分割条件の指定:(製品番号) BETWEEN (?) AND (?)
分割値の指定  :(110,119)
```

```
上記の場合、以下の条件式の結果が真になる行の集まりが格納の対象となります。
  格納対象      : 110 <= 製品番号 <=119
```

[その3] 分割条件を“=”指定で、複数個指定する場合

```
分割条件の指定: (製品名,倉庫番号) = (?,?)
分割値の指定   : (N'テレビ',2),(N'冷蔵庫',1)
上記の場合、以下の条件式の結果が真になる行の集まりが格納の対象となります。
  格納対象      : (製品名 = テレビ   かつ   倉庫番号 = 2)
            または(製品名 = 冷蔵庫   かつ   倉庫番号 = 1)
```

[その4] 分割条件を“BETWEEN”指定で、複数個指定する場合(使用する表は使用例4の売上表とします)



- ある行(データ)の格納先が、複数のDSIとなるような、分割値の指定の仕方はできません。

割付け対象(DATA、PRIMEおよびOVERFLOW)

ー 表のデータ構造がSEQUENTIALまたはOBJECTの場合、割付け対象にDATAを指定します。RANDOMの場合は、PRIMEとOVERFLOWを指定します。

― スペース割付け句で指定する割付け対象の意味を以下に示します。

表3.3 表のデータ構造と指定する割付け対象

表のデータ構造	指定する割付け対象	意味
SEQUENTIAL	DATA	表のDSIのデータ部にスペースを割り付ける。
OBJECT		
RANDOM	PRIME	表のDSIのプライム部にスペースを割り付ける。
	OVERFLOW	表のDSIのオーバフロー部にスペースを割り付ける。

参照

プライム部、オーバフロー部およびデータ部の構造については、“RDB運用ガイド(データベース定義編)”を参照してください。

割付け先(ON)

― 1つの割付け要素について、同じデータベーススペースを2回以上指定することはできません。

データベーススペース名

― 表のDSIのデータを格納するデータベーススペースの名前を指定します。

― 1つのDSIの割付け対象の割付け先として指定するデータベーススペース名は、同じロググループを使用するデータベーススペースでなければなりません。

― データを暗号化する場合は、以下の注意事項があります。

- 1つまたは複数のDSIの割付け先として、複数のデータベーススペースを指定する場合、データベーススペースの暗号化の指定を統一してください。
- 表のDSIおよびインデックスのDSIを定義する場合、データベーススペースの暗号化の指定は、統一する必要がありません。

割付け量(SIZE)

― 割付け量には、データベーススペース中に獲得する格納領域の大きさを、符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り上げます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。

― 表のデータ構造がSEQUENTIALの場合、表のDSIのデータ部に対する割付け量は、ページ長の2倍以上を指定することが必要です。たとえば、データ部のページ長が2キロバイトの場合、データ部に対する割付け量は4キロバイト以上になるように指定します。

― 表のデータ構造がRANDOMの場合、表のDSIのプライム部およびオーバフロー部に対する割付け量は、ページ長の2倍以上を指定することが必要です。たとえば、プライム部のページ長が1キロバイトの場合、プライム部に対する割付け量は2キロバイト以上になるように指定します。オーバフロー部のページ長が2キロバイトの場合、オーバフロー部に対する割付け量は4キロバイト以上になるように指定します。

― 表のデータ構造がOBJECTの場合、表のDSIのデータ部に対する割付け量は、ページ長の2倍以上を指定することが必要です。データ部のページ長は32キロバイトであるため、データ部に対する割付け量は64キロバイト以上になるように指定します。

― 割付け量を省略した場合は、表のDSIのプライム部、オーバフロー部およびデータ部に対する割付け量は、それぞれDSO定義時のページ長の64倍になります。

― 割付け対象のデータベーススペースに、割付け量分の連続領域が存在しない場合、複数の領域に分割して割付けることがあります。複数の領域に分割されたかどうかは、rdbprtコマンドの出力結果で確認できます。DSI情報のAllocation informationのAllocate sizeに、指定した割付け量以下の値が表示された場合、複数の領域に分割して割付けられたと判断できます。

FORMAT

― 表のDSIを初期化する場合に指定します。

― rdbddlexコマンドの準備モードで定義する場合、FORMATは指定できません。

EXPAND OFF

― アラームポイントおよび容量拡張定義を設定しない場合に指定します。

拡張領域の割付け先

― 同じデータベーススペースを2回以上指定することはできません。

― 1つのDSIの割付け対象の割付け先として指定するデータベーススペース名は、同じロググループを使用するデータベーススペースでなければなりません。

― データを暗号化する場合は、以下の注意事項があります。

- 1つまたは複数のDSIの割付け先として、複数のデータベーススペースを指定する場合、データベーススペースの暗号化の指定を統一してください。
- 表のDSIおよびインデックスのDSIを定義する場合、データベーススペースの暗号化の指定は、統一する必要がありません。

― 拡張領域の割付け先には、拡張量に指定した値以上の空き領域を用意してください。
また拡張領域の割付け先を省略した場合、DSIのデータベーススペースを拡張領域の割付け先とするため、DSIのデータベーススペースに拡張量に指定した値以上の空き領域を用意してください。

― 拡張領域の割付け先を省略した場合、省略値は以下となります。

- アラームポイント:【データ部またはオーバーフロー部の割付け量 × 80%】以下となるページ長の倍数の最大値
- 拡張契機 : 0キロバイト
- 拡張量 : 1024キロバイト

拡張領域の割付け量

― 割付け量は、ページ長以上の値を指定してください。

― 割付け量には、データベーススペース中に獲得する格納領域の大きさを、符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り上げます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。

― 割付け対象のデータベーススペースに、拡張量分の連続領域が存在しない場合、複数の領域に分割して割付けることがあります。複数の領域に分割されたかどうかは、rdbprtコマンドの出力結果で確認できます。DSI情報のAllocation informationのAllocate sizeに、指定した拡張量以下の値が表示された場合、複数の領域に分割して割付けられたと判断できます。

アラームポイント

― アラームポイントの契機をDSIの使用容量で指定します。

― アラームポイントは、ページ長以上の値を指定してください。

― アラームポイントには、DSIの使用量を符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り捨てます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。

― アラームポイントを省略した場合、アラームポイントは設定されません。

拡張契機

― 自動容量拡張の拡張契機をDSIの空き容量で指定します。

― 拡張契機には、DSIの空き容量を符号なし整数と単位記号(KまたはM)の組合せで指定します。指定された値はページ長の整数倍に切り上げます。単位記号Kはキロバイト、単位記号Mはメガバイトを示します。1キロバイトは1024バイト、1メガバイトは1024キロバイトです。

－ 拡張契機を省略した場合、拡張契機の値は“DSIの容量 - アラームポイント”になります。

使用例

例1

会社表の表のDSIを定義します。この表のDSO定義は、“3.14 CREATE DSO文(表のDSO定義文)”の例1で定義されています。(表のデータ構造がSEQUENTIAL)

```
CREATE DSI 会社表ＤＳＩ  DSO 会社表ＤＳＯ
        ALLOCATE DATA ON DBSPACE001 SIZE 500K
```

例2

会社表の表のDSIを定義します。この表のDSO定義は、“3.14 CREATE DSO文(表のDSO定義文)”の例4で定義されています。(表のデータ構造がRANDOM)

```
CREATE DSI 会社表ＤＳＩ  DSO 会社表ＤＳＯ
        ALLOCATE PRIME ON DBSPACE001 SIZE 500K,
                OVERFLOW ON DBSPACE002 SIZE 300K
```

例3

在庫表の表のDSIを定義します。なお、在庫表は製品名と倉庫番号の値で分割格納するものとします。この表のDSO定義は、“3.14 CREATE DSO文(表のDSO定義文)”の例5で定義されています。(表のデータ構造がRANDOM)

```
CREATE DSI 在庫表ＤＳＩテレビ２  DSO 在庫表ＤＳＯ
        USING (N'テレビ',2)
        ALLOCATE PRIME ON DBSPACE001 SIZE 200K,
                OVERFLOW ON DBSPACE002 SIZE 100K
```

例4

2005年度の上半期の売上数量の売上表の表のDSIを定義します。なお、売上表は年度と月の値で分割格納するものとします。

```
CREATE DSO 売上表ＤＳＯ
        FROM S1.売上表
        TYPE SEQUENTIAL (PAGESIZE(4),ORDER(0))
        WHERE (年度,月) BETWEEN (?,?) AND (?,?);

CREATE DSI 上半期ＤＳＩ  DSO 売上表ＤＳＯ
        USING (2005,4,2005,9)
        ALLOCATE DATA ON DBSPACE001 SIZE 200K;
```

例5

2005年度の下半期と2006年度の下半期の売上数量の売上表のDSIを定義します。なお、売上表は年度と月の値で分割格納するものとします。また、分割値の指定を複数個指定するものとします。

```
CREATE DSO 売上表ＤＳＯ
        FROM S1.売上表
        TYPE SEQUENTIAL (PAGESIZE(4),ORDER(0))
        WHERE (年度,月) BETWEEN (?,?) AND (?,?);

CREATE DSI 下半期ＤＳＩ  DSO 売上表ＤＳＯ
        USING (2005,10,2006,3),(2006,10,2007,3)
        ALLOCATE DATA ON DBSPACE001 SIZE 200K;
```

売上表

年度	月	売上数量	
2005	4	85	2005年度上半期(4～9)のDSI
2005	5	70	
2005	6	40	
2005	7	60	
2005	8	55	
2005	9	80	
2005	10	100	2005年度下半期(10～3)のDSI
2005	11	85	
2005	12	50	
2006	1	75	
2006	2	60	
2006	3	110	
2006	4	60	下半期のDSI
⋮	⋮	⋮	
2006	10	120	2006年度下半期(10～3)のDSI
2006	11	85	
2006	12	70	
2007	1	80	
2007	2	65	
2007	3	80	

例4の上半期DSIと例5の2005年度と2006年度の下半期DSIで定義された表のDSIに格納されるデータです。ほかのデータを格納するためには、それぞれのDSIを定義することが必要です。

例6

会社表の表のDSIを定義します。この表のDSO定義は、“CREATE DSO文(表のDSO定義文)”の例1で定義されています。アラームポイントの契機をDSIの使用容量224キロバイト、自動容量拡張の拡張契機を空きページ容量0キロバイト、拡張量を1メガバイトとします。

```
CREATE DSI 会社表ＤＳＩ  DSO 会社表ＤＳＯ
        ALLOCATE DATA ON DBSPACE001 SIZE 280K,
        EXPAND ON DBSPACE001 ALARM 224K CAPACITY OK SIZE 1M
```

注意

- ロードシェア運用の場合、他ノードのRDBシステムまたは当該RDBシステムでRDBネットへの参入と離脱が行われていると、エラーとなります。
- ロードシェア運用の場合、ユーザログループの縮退または切り戻しが行われている場合、エラーとなります。
- FORMATを指定した場合、割付け量に応じて時間がかかります。

3.13 CREATE DSO文(インデックスのDSO定義文)

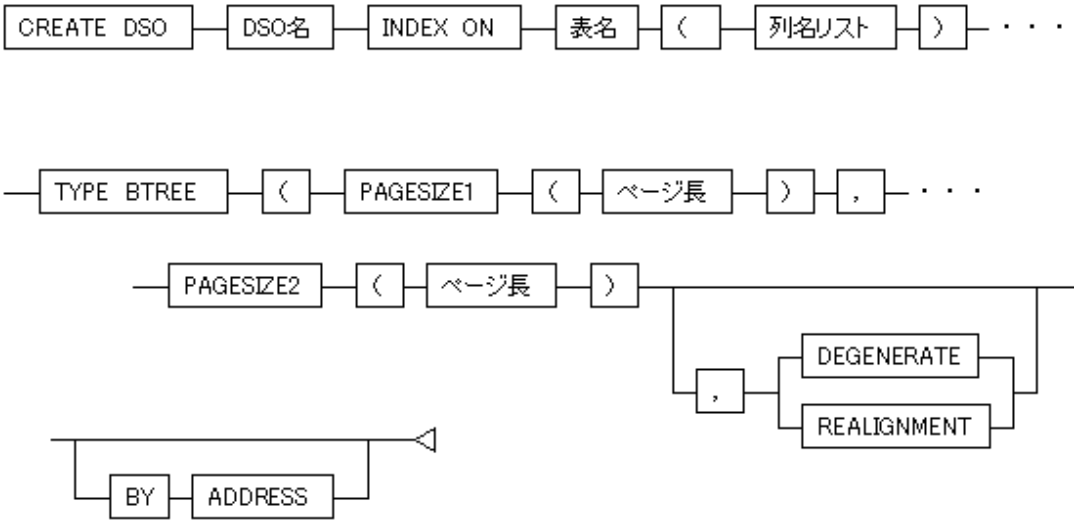
機能

インデックスのDSOを定義します。

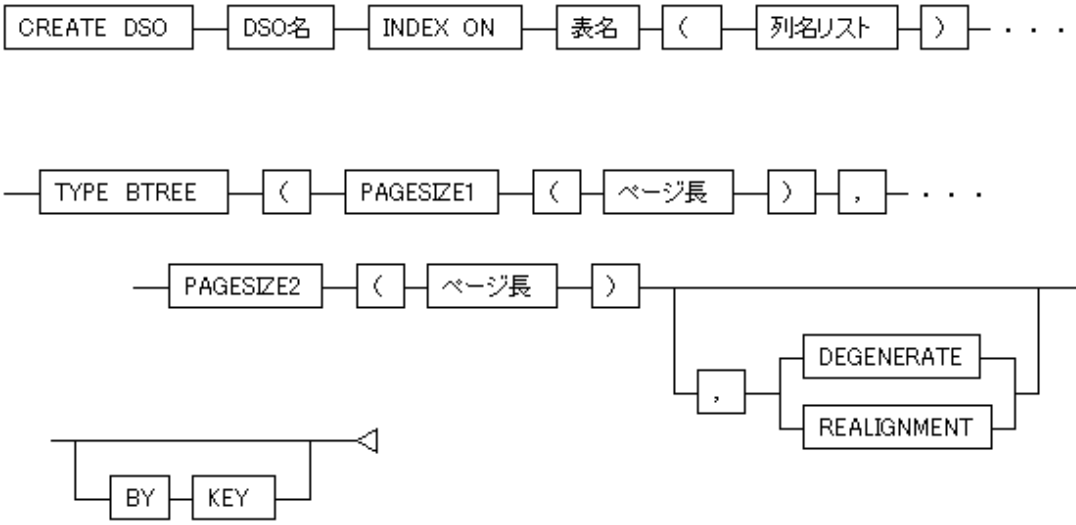
データベース簡単運用の場合は利用できません。

記述形式

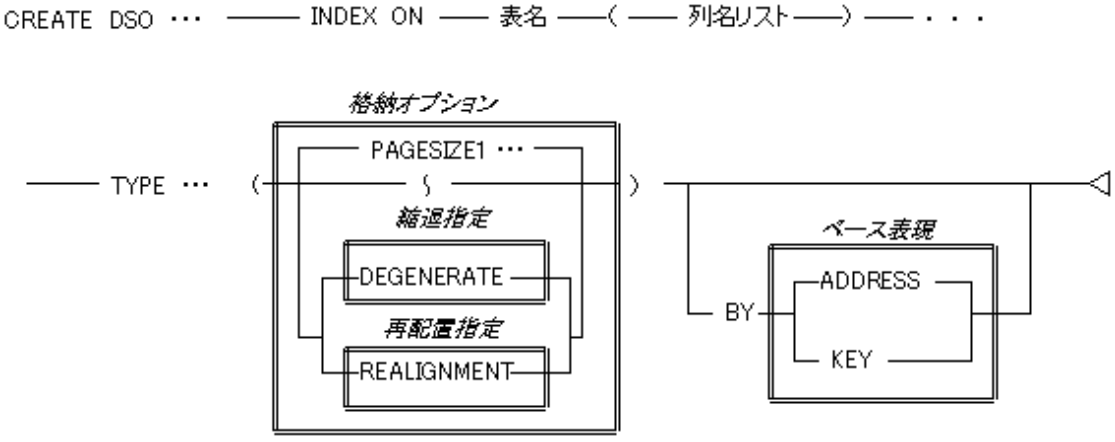
表のDSOのデータ構造がSEQUENTIALまたはOBJECTの場合



表のDSOのデータ構造がRANDOMの場合



構文の構成



参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- インデックスのDSOを定義できるのは、その表の定義者、スキーマの定義者またはその表のINDEX権の保持者です。

一般規則

DSO名

- 作成する表のDSOの名前を指定します。
- DSO名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- DSO名は、データベース内で一意の名前であることが必要です。

表名

- 表名は、インデックスを作成する表の名前を指定します。
- 表名は、スキーマ名で修飾されていることが必要です。

列名リスト

- 列名をカンマ(,)で区切り指定します。
- 列名リストは、インデックスを構成する列の組合せを指定します。
- 列名リストに指定できる列の数は、最大64個です。
- 列名リストの列名に対するデータ型で指定できる長さの合計は、最大1000バイトです。
- 列名リストに、データ型が概数型またはBLOB型の列の列名を指定することはできません。

格納オプション(PAGESIZE1、PAGESIZE2、DEGENERATE、REALIGNMENT)

- 指定可能な格納オプションを以下に示します。

表3.4 インデックスのDSOに指定可能な格納オプション

指定可能な格納オプション	意味	nに指定可能な値 (注)
PAGESIZE1(n)	データ部のページ長	1,2,4,8,16または32
PAGESIZE2(n)	インデックス部のページ長	
DEGENERATE	ページの再使用の有無	－
REALIGNMENT	ページの再配置の有無	－

注) ページ部に指定する値の単位は、キロバイトです。1キロバイトは1024バイトです。

参照

データ部およびインデックス部の構造については、“RDB運用ガイド(データベース定義編)”を参照してください。

縮退指定(DEGENERATE)

- インデックスを縮退する場合に指定します。
- DEGENERATEを指定すると、行の削除によりインデックスのページ内に有効なデータがなくなった場合に、そのページを未使用として管理し、行の追加時に新たなページが必要になった時点で再使用を行います。DEGENERATEを指定しない場合には、有効なデータがなくなったページについてこの管理は行われません。

  DEGENERATEを指定することにより、インデックススペースの利用効率が向上し、インデックスの再創成の契機を減らすことが可能となります。

－ DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表にインデックスを作成する場合、本オプションを指定することはできません。

再配置指定(REALIGNMENT)

－ REALIGNMENTを指定すると、データの再配置を行うことでインデックスのアクセスを常に最適な状態に保ち、インデックスの再編成を不要にします。本機能は、24時間連続運転などの業務を中断して、インデックスの再編成が実施できない場合に指定します。

－ DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表にインデックスを作成する場合、本オプションを指定することはできません。

ベース表現(ADDRESSまたはKEY)

－ ベース表現は、インデックスとベースの対応関係の持ち方を指定します。省略すると表のデータ構造がSEQUENTIALまたはOBJECTの場合には、ADDRESS、RANDOMの場合には、KEYが指定されたものとみなされます。

ADDRESS:

インデックスと表のレコードが、表のレコードの格納アドレスによって対応関係を持ちます。
表のデータ構造がSEQUENTIALまたはOBJECTの場合に指定します。

KEY:

インデックスと表のレコードが、表のレコードのクラスタキーによって対応関係を持ちます。
表のデータ構造がRANDOMの場合に指定します。

使用例

例1

会社表に会社名で構成するインデックスのDSOを定義します。(表のデータ構造がSEQUENTIAL)

```
CREATE DSO  会社表インデックスＤＳＯ
            INDEX ON S1.会社表（会社名）
            TYPE BTREE(PAGESIZE1(16),PAGESIZE2(1),REALIGNMENT)
            BY ADDRESS
```

例2

在庫表に製品名と倉庫番号から構成するインデックスのDSOを定義します。(表のデータ構造がSEQUENTIAL)

```
CREATE DSO  在庫表インデックスＤＳＯ
            INDEX ON S1.在庫表（製品名,倉庫番号）
            TYPE BTREE(PAGESIZE1(16),PAGESIZE2(1),REALIGNMENT)
            BY ADDRESS
```

例3

会社表に会社名で構成するインデックスのDSOを定義します。(表のデータ構造がRANDOM)

```
CREATE DSO  会社表インデックスＤＳＯ
            INDEX ON S1.会社表（会社名）
            TYPE BTREE(PAGESIZE1(16),PAGESIZE2(1),REALIGNMENT)
            BY KEY
```

例4

在庫表に製品名と倉庫番号から構成するインデックスのDSOを定義します。(表のデータ構造がRANDOM)

```
CREATE DSO  在庫表インデックスＤＳＯ
            INDEX ON S1.在庫表（製品名,倉庫番号）
```

```
TYPE BTREE(PAGESIZE1(16),PAGESIZE2(1),REALIGNMENT)
BY KEY
```

注意

縮退指定(DEGENERATE)および再配置指定(REALIGNMENT)は、インデックスのDSI定義文およびインデックスのDSO定義文でそれぞれ指定が可能です。それぞれで異なる指定をした場合、インデックスのDSI定義文で指定した値が有効になります。インデックスのDSI定義文に縮退指定(DEGENERATE)および再配置指定(REALIGNMENT)を省略した場合は、インデックスのDSO定義文の指定に従います。

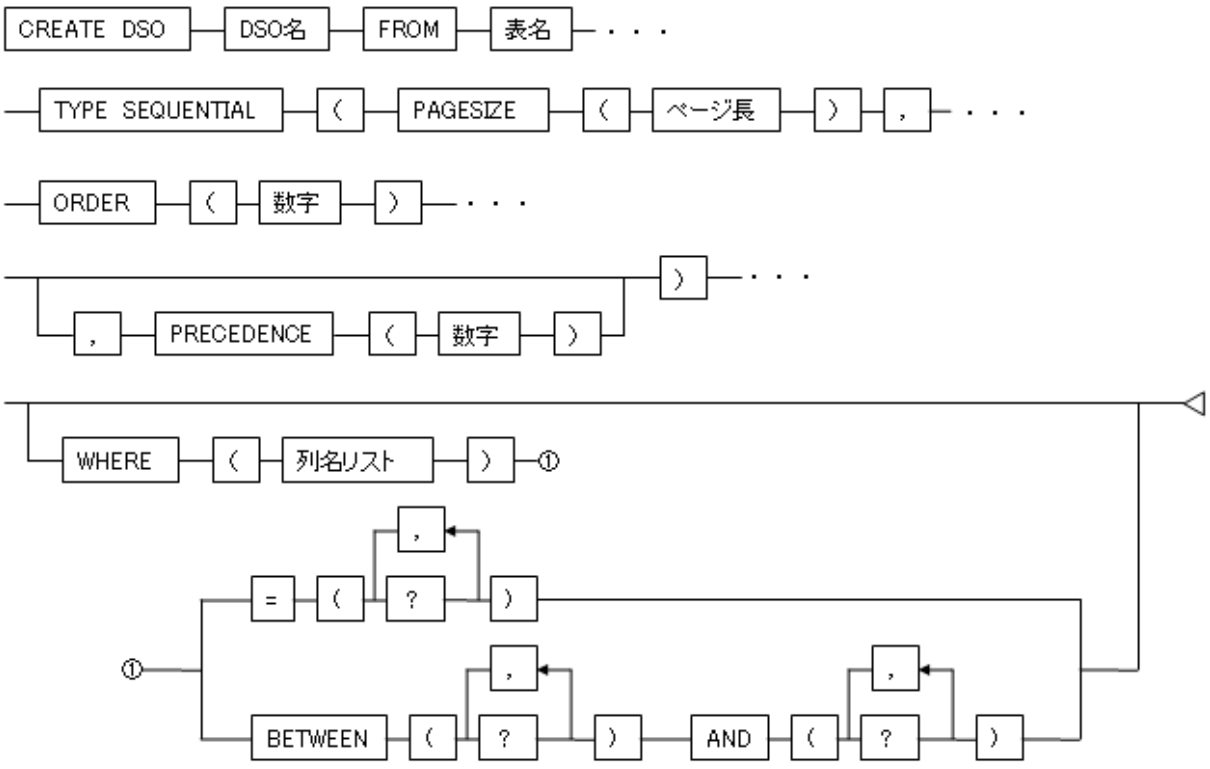
3.14 CREATE DSO文(表のDSO定義文)

機能

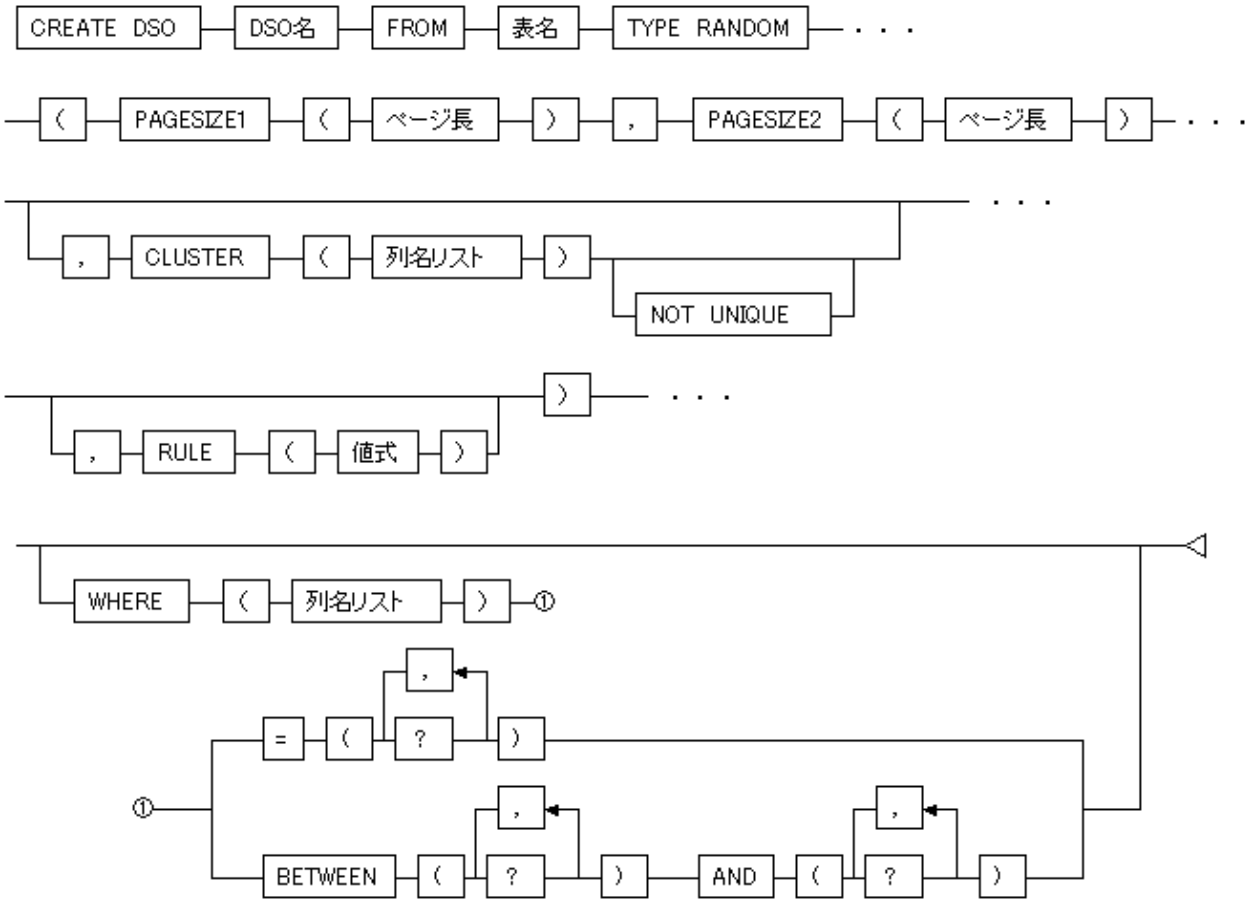
表のDSO(Data Structure Organization)を定義します。

データベース簡単運用の場合は利用できません。

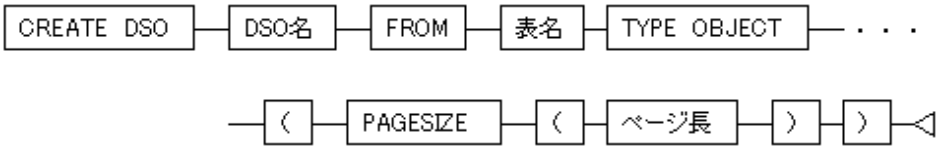
記述形式-1(表のデータ構造がSEQUENTIAL)



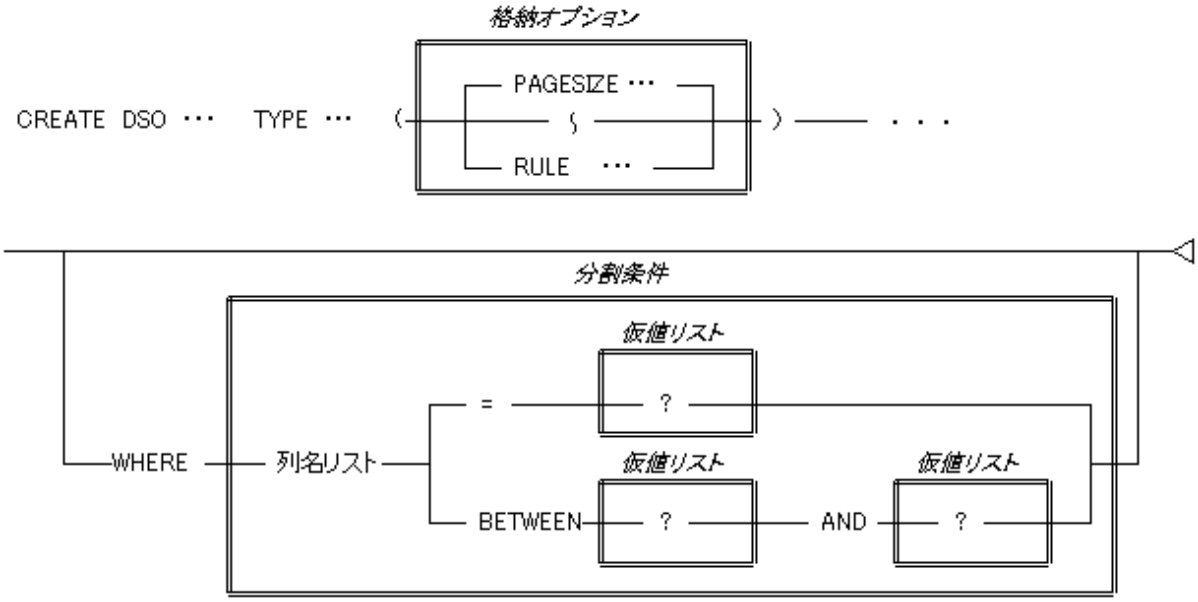
記述形式-2(表のデータ構造がRANDOM)



記述形式-3(表のデータ構造がOBJECT)



構文の構成



参照項番

- 値式 → “2.11 値式”
- 日本語文字列 → “2.1.3 トークン”

権限

- 表のDSOを定義できるのは、その表の定義者、スキーマの定義者またはその表のALTER権の保持者です。

一般規則

DSO名

- 作成する表のDSOの名前を指定します。
- DSO名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列指定します。
- DSO名は、データベース内で一意の名前であることが必要です。

表名

- 表名は、格納構造定義の対象とする表の名前を指定します。
- 表名は、スキーマ名で修飾されていることが必要です。

格納オプション(PAGESIZE(n)、ORDER(n)、PRECEDENCE(n)、CLUSTERおよびRULE)

- 指定可能な格納オプションを以下に示します。

表3.5 表のデータ構造と指定可能な格納オプション

表のデータ構造	指定可能な格納オプション	意味	nに指定可能な値
SEQUENTIAL	PAGESIZE(n)	データ部のページ長	1,2,4,8,16または32 (注1) (注2) (注3)
	ORDER(n)	削除領域の再使用の有無	0または1
	PRECEDENCE(n)	コミットデータ即時読込み機能の利用の有無	1のみ
RANDOM	PAGESIZE1(n)	プライム部のページ長	1,2,4,8,16または32 (注1)
	PAGESIZE2(n)	オーバフロー部のページ長	

表のデータ構造	指定可能な格納オプション	意味	nに指定可能な値
	CLUSTER 列名リスト	表のクラスタキー	―
	RULE(値式)	レコードの格納場所	―
OBJECT	PAGESIZE(n)	データ部のページ長	32 (注1)

注1) ページ長に指定する値の単位は、キロバイトです。1キロバイトは1024バイトです。

注2) 32キロバイト以上のBLOBを含む場合のページ長は、32キロバイトを推奨します。

注3) PRECEDENCE(1)を指定する場合のページ長は、32キロバイトを推奨します。

参照

データ部、プライム部、オーバフロー部および表のクラスタキーの構造については、“RDB運用ガイド(データベース定義編)”を参照してください。

― 表のデータ構造がOBJECTのDSO定義は、データにBLOB型でサイズが32K以上の値を指定した表に対して定義します。それ以外の表に対しては定義することができません。

ORDER

― 格納オプションのORDERは、表のデータ構造にSEQUENTIALを指定した場合、省略できません。

ORDER(0):

データの更新時に表のDSI領域の自動再配置を行いません。この場合、削除領域を再使用しません。

ORDER(1):

データの更新時に表のDSI領域の自動再配置を行います。この場合、削除領域を再使用します。

PRECEDENCE

― 格納オプションのPRECEDENCEは、コミットデータ即時読込み機能を利用して、表のデータに対して更新と参照が同時に動作した場合に、更新の完了を待たずにコミット済みのデータの参照を可能とする場合に指定します。

- 必ず1を指定します。
- 表のデータ構造がSEQUENTIALの場合に指定します。
- 格納オプションのORDER(1)と組み合わせて指定します。

参照

格納オプションにPRECEDENCE(1)が指定されたSEQUENTIAL構造の詳細については、“RDB運用ガイド(データベース定義編)”を参照してください。

CLUSTER

― 格納オプションのCLUSTERは、表のデータ構造がRANDOMの場合に、表の中でクラスタキーとなる列または列の並びを指定します。列名リストは列名をカンマ(,)で区切り指定します。

- NOT UNIQUEは、列名リストに指定する列の組が、一意性制約されていない場合に指定します。列名リストに指定する列の組が一意性制約されている場合に、NOT UNIQUEを指定するとエラーになります。列の組が一意性制約されているとは、表名で指定された表に同じ列の組から構成される一意性制約定義が存在することをいいます。
- 分割条件を指定する場合、CLUSTERの列名リストは、分割条件の列名リストで指定する列をすべて含むように指定することが必要です。なお、CLUSTERの列名リストは、分割条件の列名リストに等しいか、分割条件の列名リストのうしろに、列名の並びを追加して指定することが必要です。
- 表のデータ構造がRANDOMの場合に、CLUSTERを省略すると、表定義中のPRIMARY KEYで指定された列の組がクラスタキーになります。このとき、表にPRIMARY KEYが指定されていないとエラーになります。

- 列名リストに指定可能な列のデータ型を以下に示します。なお、CLUSTERを省略する場合は、表に定義されたPRIMARY KEYは以下に示すデータ型の列で構成されていることが必要です。

表3.6 CLUSTERおよび分割条件の列名リストに指定可能な列のデータ型

属性	精度	位取り	長さ	備考
SMALLINT	―	―	―	
INTEGER	―	―	―	
NUMERIC	1～18	0～精度	―	
DECIMAL	1～18	0～精度	―	
CHARACTER	―	―	1～1000 (注1)	VARYINGは指定不可
NATIONAL CHARACTER	―	―	1～500 (注2)	VARYINGは指定不可
DATE	―	―	―	
TIME	―	―	―	
TIMESTAMP	―	―	―	
INTERVAL	1～9	―	―	

注1) 分割条件に指定する場合は、1～254で指定します。

注2) 分割条件に指定する場合は、1～127で指定します。

― 列名リストに指定する列には、NOT NULLが指定されていることが必要です。

RULE

― 格納オプションのRULEは、表のデータ構造がRANDOMの場合に、レコードの格納場所を決定する値式を指定します。RULEに指定する値式の指定方法は以下のとおりです。

- 列名と定数による演算式が指定できます。ただし、定数は整数のみが指定可能で、小数点の指定はできません。
- 計算結果が負数となる値式は指定できません。
- 値式に指定する列の属性はINTEGER型またはSMALLINT型であることが必要です。
- 値式に指定する列はクラスタキーであることが必要です。

分割条件(WHERE)

― 分割条件の指定方法は以下のとおりです。

- 表を行の集まり単位に分割して格納する場合に、分割の条件を指定します。
- 実際の分割単位はDSI定義文で指定します。ここでは、分割のための条件のみを指定します。
- 各分割単位に格納される行は、列名リストに指定された各列に設定する値と、仮値リストの“?”にDSI定義文の分割値で指定された定数を代入した比較の結果により決定されます。
- “=”は、データを限定してDSIに格納する場合に指定します。
- BETWEENは、範囲比較でデータをDSIに格納する場合に指定します。
- DSO定義文で分割条件を指定する場合には、表定義時に一意性制約の指定方法について十分に注意が必要です。詳細については、“3.22 CREATE TABLE文(表定義)”を参照してください。
- 表のデータ構造がOBJECTのDSO定義には、分割条件は指定できません。

列名リスト

― 列名リストは列名をカンマ(,)で区切って指定します。列名リストに指定可能な列の数は、64個までです。列名リスト中の列名の個数と仮値リストの“?”の個数は、同じであることが必要です。

ー 列名リストに指定可能な列のデータ型は“表3.6 CLUSTERおよび分割条件の列名リストに指定可能な列のデータ型”のとおりです。

ー 列名リストに指定する列には、NOT NULLが指定されていることが必要です。

使用例

例1

会社表の表のDSOを定義します。(表のデータ構造がSEQUENTIAL)

```
CREATE DSO 会社表ＤＳＯ  FROM S1.会社表
           TYPE SEQUENTIAL(PAGESIZE(4),ORDER(1))
```

例2

在庫表を製品名と倉庫番号の値で分割格納する、表のDSOを定義します。(表のデータ構造がSEQUENTIAL)

```
CREATE DSO 在庫表ＤＳＯ  FROM S1.在庫表
           TYPE SEQUENTIAL(PAGESIZE(4),ORDER(1))
           WHERE (製品名,倉庫番号) = (?,?)
```

例3

会社表の表のDSOをコミットデータ即時読込み機能の利用ありで定義します。(表のデータ構造がSEQUENTIAL)

```
CREATE DSO 会社表ＤＳＯ  FROM S1.会社表
           TYPE SEQUENTIAL(PAGESIZE(32),ORDER(1), PRECEDENCE(1))
```

例4

会社表の表のDSOを定義します。なお、会社表の一意性指定にPRIMARY KEYが指定されているものとします。(表のデータ構造がRANDOM)

```
CREATE DSO 会社表ＤＳＯ  FROM S1.会社表
           TYPE RANDOM(PAGESIZE1(4),PAGESIZE2(16))
```

例5

在庫表を製品名と倉庫番号の値で分割格納する、表のDSOを定義します。なお、在庫表の一意性指定にPRIMARY KEYが指定されているものとします。(表のデータ構造がRANDOM)

```
CREATE DSO 在庫表ＤＳＯ  FROM S1.在庫表
           TYPE RANDOM(PAGESIZE1(4),PAGESIZE2(16))
           WHERE (製品名,倉庫番号) = (?,?)
```

例6

製品写真表に対する表のDSOを定義します。(表のデータ構造がOBJECT)

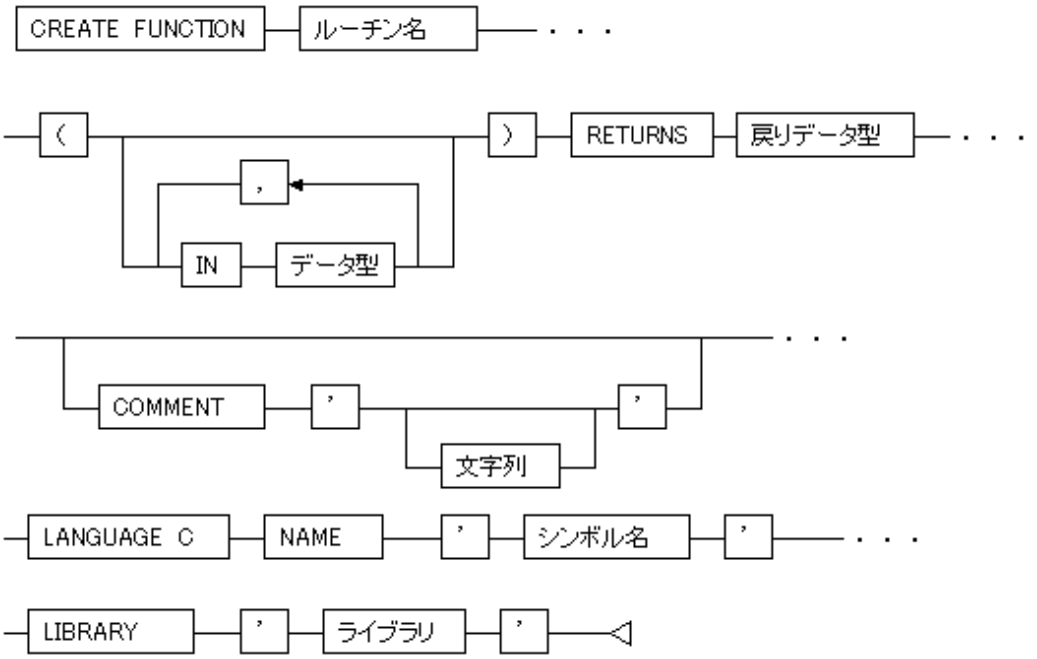
```
CREATE DSO 製品写真ＤＳＯ  FROM S1.製品写真表
           TYPE OBJECT(PAGESIZE(32))
```

3.15 CREATE FUNCTION文(ファンクションルーチン定義)

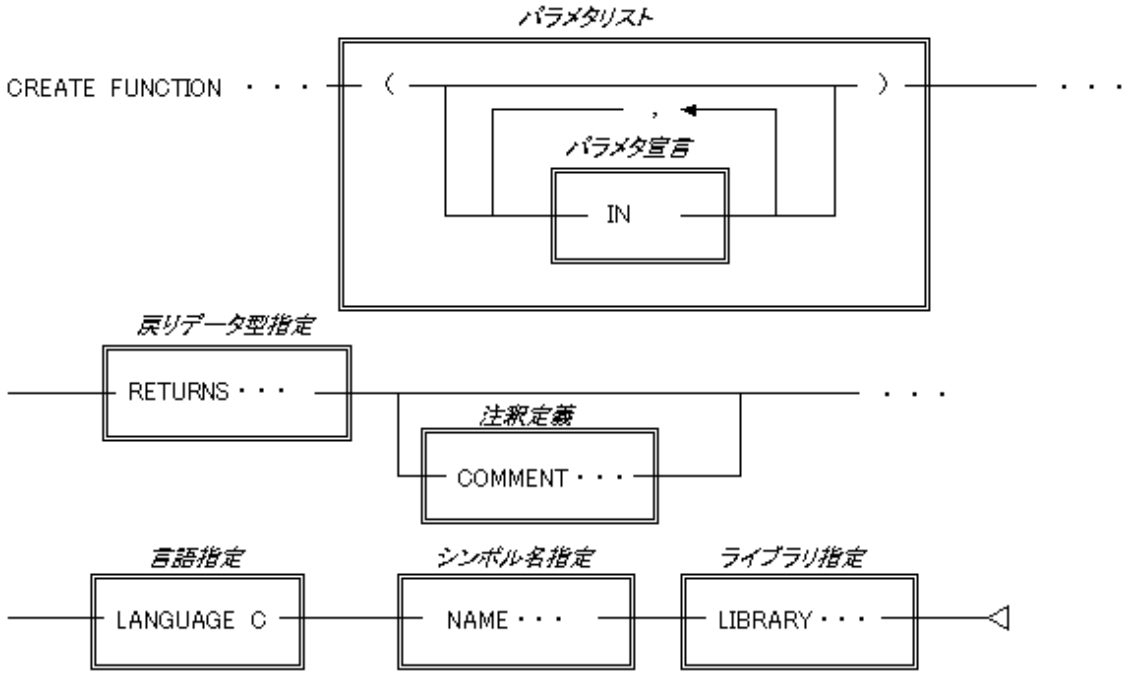
機能

ファンクションルーチンを定義します。引数のデータ型、処理結果のデータ型およびファンクションルーチンの処理を行うプログラムを指定します。

記述形式



構文の構成



参照項番

- データ型 → “2.2 データ型”
- 戻りデータ型 → “2.2 データ型”
- 日本語文字列 → “2.1.3 トークン”

権限

- ファンクションルーチン定義が実行できるのは、スキーマ定義者およびCREATE権を付与された人です。
- ファンクションルーチンを使用したSQL文を実行できるのは、ファンクションルーチンの定義者とEXECUTE権を付与された人です。

一般規則

ルーチン名

- 定義するファンクションルーチンの名前を指定します。
- ルーチン名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列指定します。
- ルーチン名はスキーマ内で、一意の名前でなければなりません。
- ファンクションルーチン定義がスキーマ定義に含まれる場合、ルーチン名のスキーマ名の修飾を省略した場合は、スキーマ定義で指定したスキーマ名で修飾したとみなされます。また、スキーマ名で修飾する場合は、スキーマ定義で指定したスキーマ名であることが必要です。

データ型

- ファンクションルーチンの引数となるデータ型を指定します。ファンクションルーチンの処理を行うC言語アプリケーションには、ここで指定したデータ型で引数が通知されます。

戻りデータ型

- ファンクションルーチンの処理結果のデータ型のデータ型を指定します。

シンボル名

- ファンクションルーチンの処理を行うC言語アプリケーションのオブジェクトのシンボル名を指定します。シンボル名に指定できる長さは、256バイト以内です。

ライブラリ

- ファンクションルーチンの処理を行うC言語アプリケーションの実行モジュールのライブラリのパスを指定します。ライブラリのパスに指定できる長さは、256バイト以内です。

クラスタシステムでの利用

- クラスタシステムでのファンクションルーチンの処理を行うプログラムの格納場所は、各ノードに同一のディレクトリを作成し、同じプログラムを格納します。

使用例

例

S L

Solaris/Linuxの場合

ファンクションルーチン“FUNC1”として、引数のデータ型、処理結果のデータ型、“FUNC1”を実行するCプログラムの関数“user_apl1”、“user_apl1”を格納しているライブラリ“/usr/local/lib/linmain1.so”を定義します。

```
CREATE FUNCTION S1.FUNC1
  (IN INTEGER,IN INTEGER)
  RETURNS INTEGER
  LANGUAGE C NAME 'user_apl1'
  LIBRARY '/usr/local/lib/linmain1.so'
```

W

Windowsの場合

ファンクションルーチン“FUNC1”として、引数のデータ型、処理結果のデータ型、“FUNC1”を実行するCプログラムの関数“user_apl1”、“user_apl1”を格納しているライブラリ“D:¥FORSYMFO¥FUNCLIB¥USERFUNC001.DLL”を定義します。

```
CREATE FUNCTION S1.FUNC1
  (IN INTEGER,IN INTEGER)
```

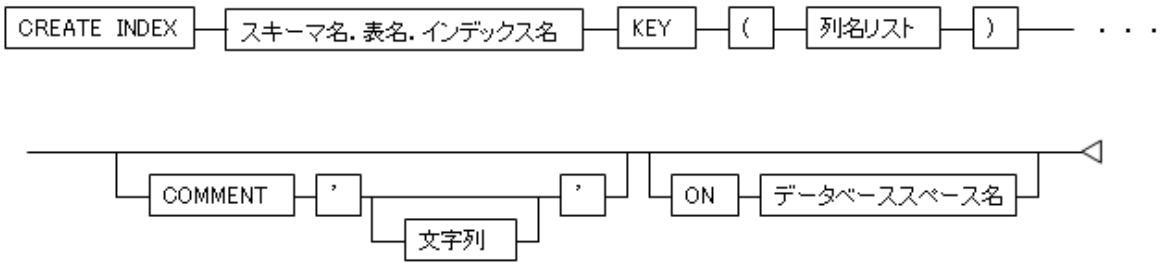
```
RETURNS INTEGER
LANGUAGE C NAME 'user_apl1'
LIBRARY 'D:¥FORSYMFO¥FUNCLIB¥USERFUNC001.DLL'
```

3.16 CREATE INDEX文(インデックス定義)

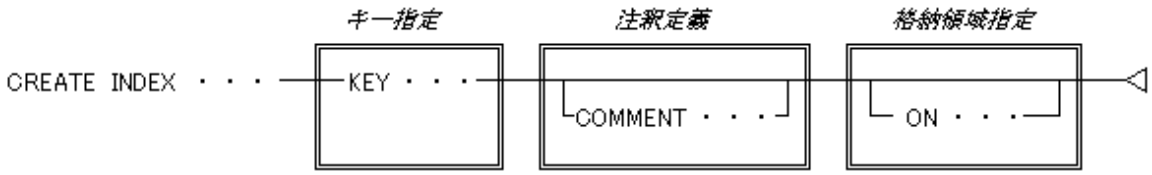
機能

インデックスのDSO定義とインデックスのDSI定義を簡略化してインデックスを定義します。

記述形式



構文の構成



参照項番

- 文字列定数 → “2.1.2 定数”
- 日本語文字列 → “2.1.3 トークン”

権限

- インデックスを定義できるのは、その表の定義者、スキーマの定義者またはその表のINDEX権の保持者です。

一般規則

- 表に一意性制約を指定した場合、表データを格納する前に当該の列および列群に対するインデックス定義を行うことが必要です。
  ただし、データベース簡単運用の場合は、表の一意性制約に対するインデックスをシステムが自動的に定義するため、インデックスを定義する必要はありません。なお、インデックス名は“@”、システムで採番する10桁の数字と“_PK”(PRIMARY KEYの場合)または“_UK”(UNIQUEの場合)を使用して命名します。
- 格納領域指定で指定するデータベーススペースに対するALLOCATE権が必要です。

表名

- インデックスを作成する表の名前を指定します。
- DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表を指定することはできません。
  PRECEDENCE(1)が指定されたSEQUENTIAL構造の表にインデックスを定義する場合は、インデックスのDSO定義文およびインデックスのDSI定義文で定義してください。

インデックス名

- 作成するインデックスの名前を指定します。

- インデックス名は、システム用の動作環境ファイルでDEFAULT_DSI_NAME=CODEを指定した場合は、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。CODEを指定しない場合は、8文字以内の先頭が英字で始まる英数字、または8文字以内の日本語文字列を指定します。
- 一時表の場合のインデックス名は、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- インデックス名は、スキーマ内で一意の名前であることが必要です。

キー指定(KEY)

- キー指定は、インデックスキーを構成する列および列の並びを指定します。キー指定中の列または列の並びの組合せが、表定義で指定した一意性制約に対応する列または列の並びの組合せと一致する場合、一意な値を持つインデックスとして定義されます。これ以外の場合、重複する値を持つインデックスとして定義します。
- 列名リストは、列名をカンマ(,)で区切って指定します。
- インデックスキーを構成する列に、データ型が概数型またはBLOB型の列名は指定できません。
- キー指定で、列の並びが同一のインデックスを定義することはできません。
- キー指定の指定方法は以下のとおりです。
  - 同じ列名を2回以上指定することはできません。
  - 列名は、表名で指定する実表の列であることが必要です。
  - 指定できる列の数は、最大64個です。
  - インデックスキーを構成する列の大きさは、1000バイト以下であることが必要です。

注釈定義(COMMENT)

- 注釈は256バイト以内の文字列(日本語記述可能)を指定することができます。

格納領域指定(ON)

- 格納領域指定は、インデックスの格納先のデータベーススペースを指定します。
  - 指定されたデータベーススペースは、定義済であることが必要です。インデックス定義により指定されたデータベーススペースに、Symfoware/RDBが自動的にインデックスの格納構造を定義します。

    定義される格納構造については、“D.3 格納領域指定時の格納構造”を参照してください。
  - 格納領域指定で指定するデータベーススペースは、関連する表で割り付けたデータベーススペースと同じロググループを使用するものでなければなりません。
  - インデックスを作成する表が一時表の場合、格納領域指定は指定できません。
  - インデックスを作成する表がデフォルトデータベーススペースに定義されている場合、格納領域指定は指定できません。

使用例

例

スキーマ名S1の表T1にインデックス“IXA”を定義します。

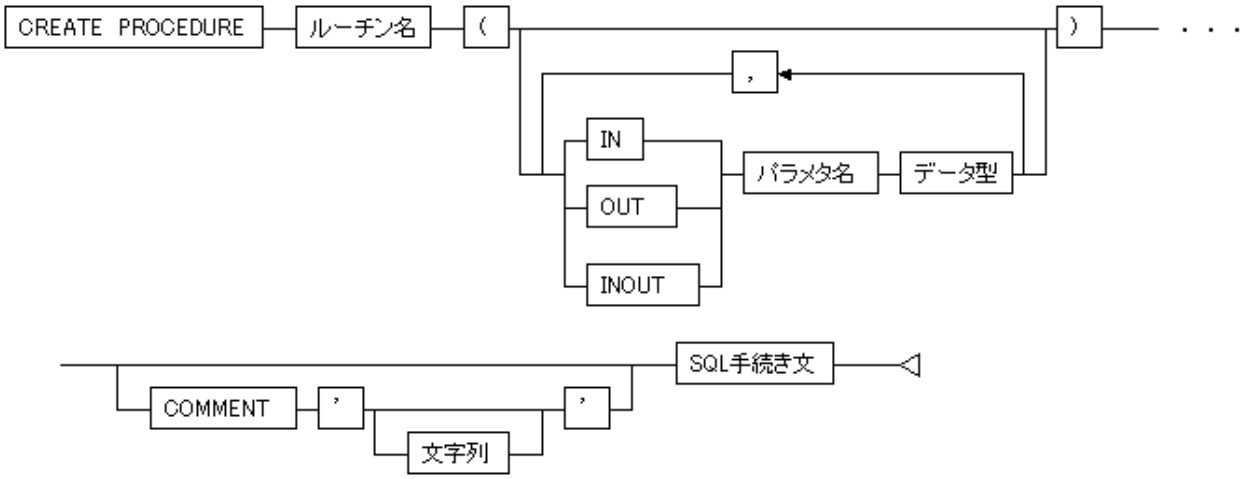
```
CREATE TABLE S1.T1(C1 INT NOT NULL,
                   C2 INT NOT NULL,
                   C3 INT NOT NULL) ON DBSPACE1;

CREATE INDEX S1.T1.IXA KEY(C1) ON DBSPACE2;
```

3.17 CREATE PROCEDURE文(プロシジャルーチン定義)

機能

プロシジャルーチンを定義します。

記述形式



参照項番

- SQL手続き文 → “5.1 ストアドプロシジャの概要”
- データ型 → “2.2 データ型”
- ルーチン名 → “2.1.4 名前”
- パラメタ名 → “2.1.4 名前”

権限

- プロシジャ定義の実行者は、以下の条件を満たすことが必要です。
  - スキーマに対するCREATE権を持っていることが必要です。
  - プロシジャルーチン内の各SQL文を実行するための権限を持っていることが必要です。

一般規則

- CREATE PROCEDURE文については、ストアドプロシジャで詳しく説明します。“5.1 ストアドプロシジャの概要”を参照してください。

使用例

例

プロシジャルーチン“ROUTINE1”を定義します。

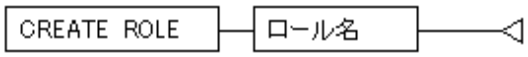
```
CREATE PROCEDURE S1.ROUTINE1(IN PARA1 INTEGER)
BEGIN
  DECLARE A INTEGER;
  SET A = PARA1 + 1;
  INSERT INTO S1.T1(C1) VALUES(A);
              :
END
```

3.18 CREATE ROLE文(ロール定義文)

機能

ロールを定義します。ロール定義文は、スキーマ要素に含まれます。

記述形式



参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- ロールを定義できるのは、RDBディクショナリの定義者のみです。
- 定義者にロールに関する付与権が与えられます。

一般規則

ロール名

ー 作成するロールの名前を指定します。

ー ロール名には、36バイト以内の先頭が英字で始まる英数字、または日本語文字列を指定します。日本語文字列に指定する文字コード系は、データベースの文字コード系によって長さが異なります。詳細は、“2.1.3 トークン”を参照してください。

ー ロール名は、RDBシステム内で一意でなければなりません。

使用例

例

ロール“STOCKS_A2”を定義します。

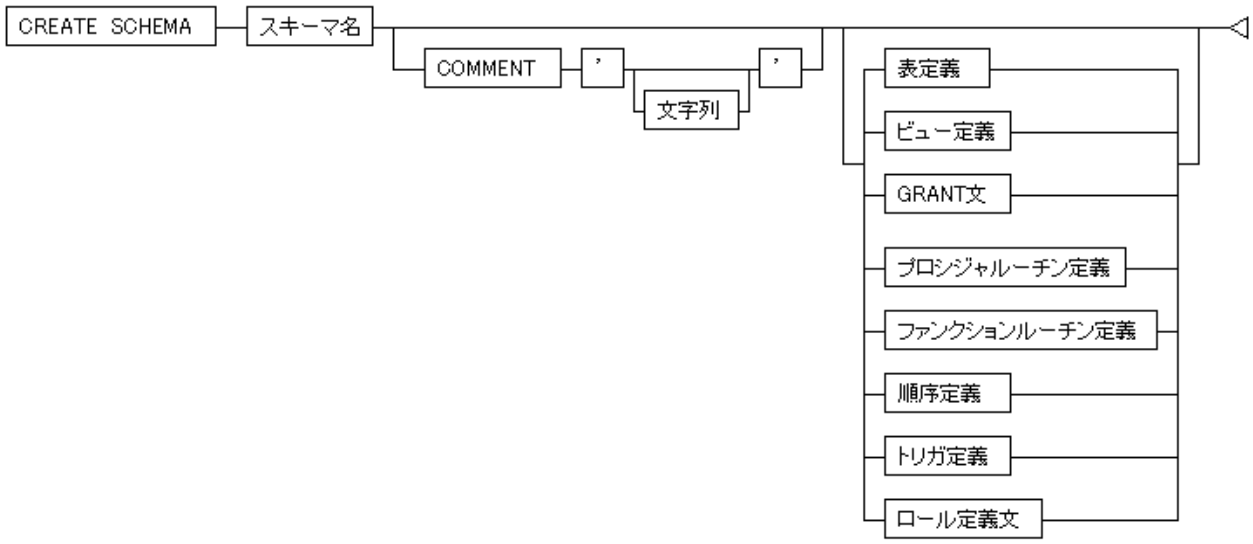
```
CREATE ROLE STOCKS_A2
```

3.19 CREATE SCHEMA文(スキーマ定義)

機能

スキーマ定義は、表の集まりを管理する単位となるスキーマと、スキーマに含まれる実表およびビュー表を定義します。

記述形式



構文の構成

参照項番

- 表定義 → “3.22 CREATE TABLE文(表定義)”
- ビュー定義 → “3.25 CREATE VIEW文(ビュー定義)”
- GRANT文 → “3.47 GRANT文”
- プロシジャルーチン定義 → “3.17 CREATE PROCEDURE文(プロシジャルーチン定義)”
- ファンクションルーチン定義 → “3.15 CREATE FUNCTION文(ファンクションルーチン定義)”
- 順序定義 → “3.21 CREATE SEQUENCE文(順序定義)”
- トリガ定義 → “3.23 CREATE TRIGGER文(トリガ定義)”
- ロール定義文 → “3.18 CREATE ROLE文(ロール定義文)”
- 日本語文字列 → “2.1.3 トークン”

一般規則

スキーマ名

- 作成するスキーマの名前を指定します。
- スキーマ名は、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- スキーマ名は、データベース内で一意の名前であることが必要です。
- 格納領域指定を指定する表を定義するスキーマは、スキーマ名を8文字以内で定義します。システム用の動作環境ファイルでDEFAULT_DSI_NAME=CODEを指定した場合は、スキーマ名は36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

注釈定義(COMMENT)

- 注釈は256バイト以内の文字列(日本語記述可能)を指定することができます。

スキーマ要素

- スキーマ定義にスキーマ要素が含まれる場合、すべての要素の定義が正しい場合にだけスキーマが定義されます。要素の定義に1つでもエラーがあれば、すべてのスキーマ要素およびスキーマは定義されません。

使用例

例

スキーマ“S1”を定義します。S1には在庫表を含んで定義します。

```
CREATE SCHEMA S1
CREATE TABLE  在庫表 (製品番号 SMALLINT NOT NULL,
                      製品名   NATIONAL CHARACTER(10) NOT NULL,
                      在庫数量 INTEGER,
                      倉庫番号 SMALLINT,
                      PRIMARY KEY (製品番号)) ;
```

3.20 CREATE SCOPE文(スコープ定義文)

機能

スコープ定義は、データ操作の範囲を定義します。

データベース簡単運用の場合は利用できません。

記述形式

参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- スコープを定義できるのは、指定したDSIが定義済の場合、DSIの定義者のみです。DSIをスコープ定義の後から定義する場合は、DSI定義者はスコープの定義者であることが必要です。

一般規則

- 指定されたDSIは未定義でもかまいません。未定義のDSIを指定した場合、表のDSIとみなします。ただし、未定義のDSIを指定すると、以下のようなエラーになる可能性があるため注意が必要です。
  - ― 未定義のDSIをスコープ定義したあとにそのDSI定義を行うとき、表のDSIの定義者とスコープ定義者が違う場合はエラーになります。
  - ― スコープ定義で、ほかのスコープですでに指定された未定義のDSI名を指定するとき、新規のスコープ定義者と既存のスコープ定義者が違う場合はエラーとなります。
- 指定されたDSIは、ほかのスコープ定義文で指定されていてもかまいません。
- スコープ機能で限定されるデータ操作の範囲は、アプリケーションからのSQL文、rdbuptコマンドおよびrdbunlsqlコマンドです。rdbsloaderコマンドなどのRDBコマンドでは有効となりません。

スコープ名

- ― 作成するスコープの名前を指定します。
- ― スコープ名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- ― スコープ名は、データベース内で一意の名前であることが必要です。

DSI名リスト

- ― データ操作の範囲を限定する表のDSIの名前を指定します。DSI名リストは、DSI名をカンマ(,)で区切って指定します。
- ― DSI名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- ― 同じDSI名を複数指定することはできません。
- ― 一時表のDSI名を指定することはできません。

使用例

例

発注表のデータ操作の範囲を東京DSIに限定するスコープ“東京ＳＣＰ”を定義します。

```
CREATE SCOPE 東京ＳＣＰ DSI（東京ＤＳＩ）
```

3.21 CREATE SEQUENCE文(順序定義)

機能

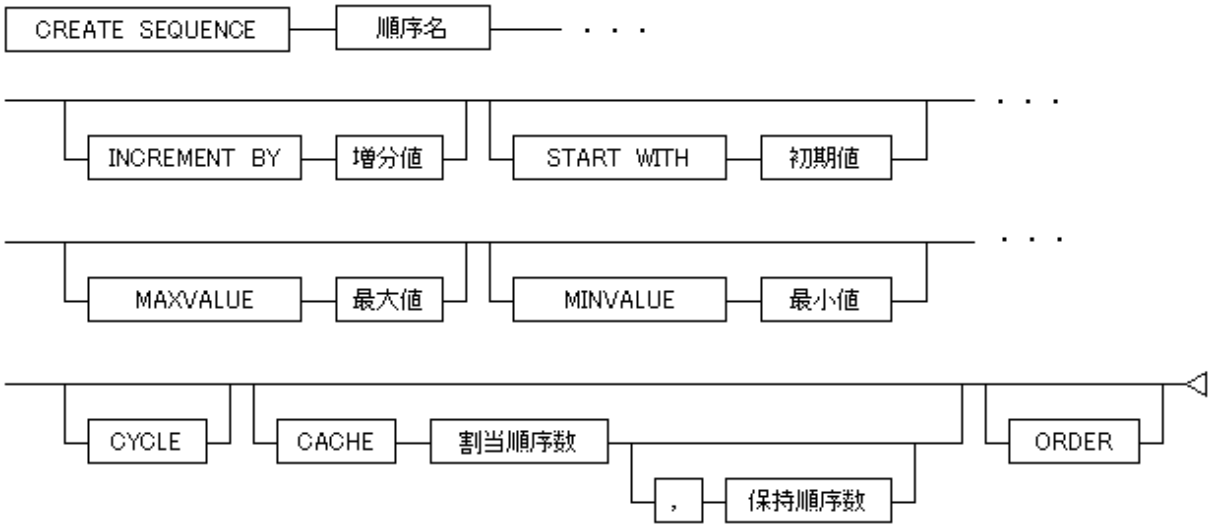
順序定義は、順序を定義します。

順序とは、RDBシステム全体で一意な値を生成する機能です。順序の主な用途は主キー値の作成です。

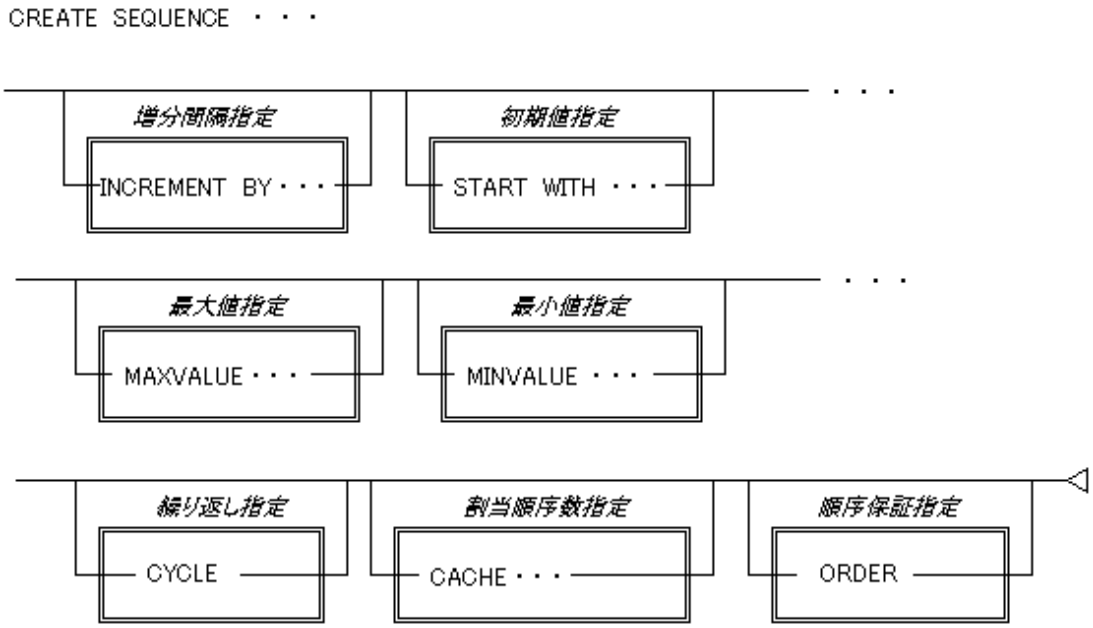
定義した順序は、アプリケーションのSQL文中に順序を直接指定して使用する方法と、表定義のDEFAULT句に順序を指定して、自動的に取得した値を表に挿入する方法があります。

順序を使用すると、アプリケーションでSQL文実行時に意識することなく主キーを作成することができます。SQL文中に指定した順序の属性は、18桁の整数のNUMERIC型として扱われます。

記述形式



構文の構成



参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- 順序定義文を実行できるのは、スキーマ定義者およびCREATE権を付与された人です。
- 順序を使用したSQL文を実行できるのは、順序の定義者とSELECT権を付与された人です。

一般規則

- 順序番号の生成は、トランザクションのロールバックに関係なく増分値だけ増加します。たとえば、SQL文を実行して順序番号を生成後、そのトランザクションがロールバックしたため再度SQL文を実行した場合、同じ順序番号は生成されません。
- CACHE指定により事前にメモリ上に保持した順序番号は、システムダウンすると消失します。システム再起動後は、次の順序番号から生成されます。
- 順序定義で順序保証指定“ORDER”を指定せずに割当順序数指定“CACHE”を指定している場合、ロードシェア機能利用時にシステムダウンが発生すると、順序番号が昇順または降順に生成されないことがあります。なお、順序番号の一意性は保証されます。

順序名

- 作成する順序の名前を指定します。
- 順序名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- 順序名は、スキーマ内で一意の名前であることが必要です。
- 順序定義がスキーマ定義に含まれる場合、順序名のスキーマ名の修飾を省略した場合は、スキーマ定義で指定したスキーマ名で修飾したとみなされます。また、スキーマ名で修飾する場合は、スキーマ定義で指定したスキーマ名であることが必要です。

増分間隔指定

- 増分間隔指定は、順序番号の増分間隔を指定します。増分値は正または負の整数です。この値が正の場合は昇順に、負の場合は降順になります。省略した場合は、1ずつ昇順に増加します。

初期値指定

- 初期値指定“START WITH”は、生成する順序番号の初期値を指定します。この句を指定した場合、順序番号の最小値より大きい値を初期値として昇順を開始することも、最大値よりも小さい値を初期値として降順を開始することもできます。この句を省略した場合、省略値は、昇順の場合は順序番号の最小値になります。降順の場合は順序番号の最大値になります。18桁以内の整数値を指定します。

最大値指定

- 最大値指定“MAXVALUE”は、順序番号の最大値を指定します。18桁以内の整数値を指定します。最大値は、初期値以上でかつ最小値を超える値である必要があります。この句を省略した場合、省略値には、昇順の場合は（10の18乗）-1、降順の場合は-1が指定されます。

最小値指定

- 最小値指定“MINVALUE”は、順序番号の最小値を指定します。18桁以内の整数値を指定します。最小値は、初期値以下でかつ最大値未満である必要があります。この句を省略した場合、省略値には、昇順の場合は1、降順の場合は-(10の18乗)+1が指定されます。

繰り返し指定

- 繰り返し指定“CYCLE”は、順序番号が最大値または最小値に達した場合、順序番号の生成を続行する場合に指定します。昇順の場合は、最大値に達すると最小値が生成されます。降順の場合は、最小値に達すると最大値が生成されます。

割当順序数指定

- 割当順序数指定“CACHE”は、より高速に順序番号を取得できるように、順序番号を事前にメモリ上に割り当てて保持し、メモリから取得する機能です。CACHEには割当順序数と保持順序数を指定します。
- 割当順序数は、メモリ上に割り当てておく順序番号の数を指定します。

ー 保持順序数は、新たに順序番号をメモリ上に割り当てる契機を指定します。メモリ上に割り当てた順序番号が取得され、順序番号の残数が保持順序数に達すると、再度、割当順序数までメモリ上に順序番号を追加割り当てします。

ー 割当順序数および保持順序数には、18桁以内の正の整数値を指定します。割当順序数に指定できる最小値は2です。また、保持順序数に指定できる最小値は0です。保持順序数は、割当順序数より小さい値を指定する必要があります。保持順序数を省略した場合、0が指定されたとみなします。

順序保証指定

ー クラスタシステムで順序を使用する場合、順序保証指定“ORDER”を指定すると、クラスタシステム全体で順序番号の生成順番を保証します。

使用例

例

順序“順序1”を定義します。順序の増分値と初期値は1を指定します。最大値は100を指定します。

```
CREATE SEQUENCE S1.順序１
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 100
```

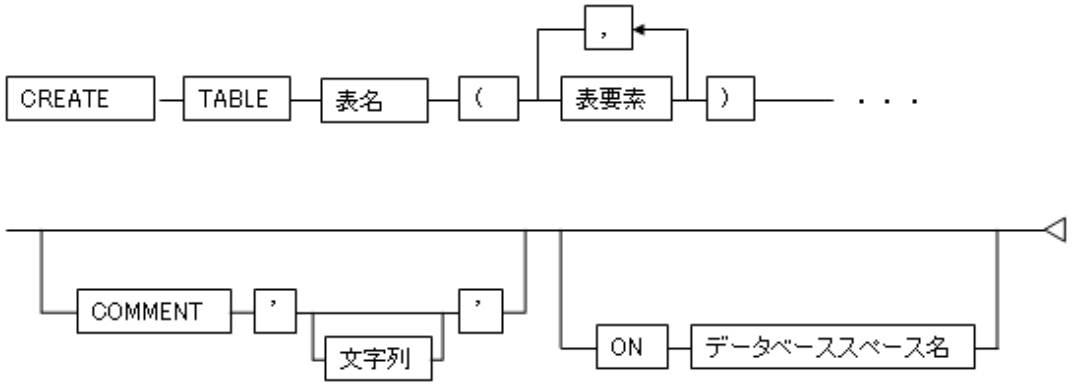
3.22 CREATE TABLE文(表定義)

機能

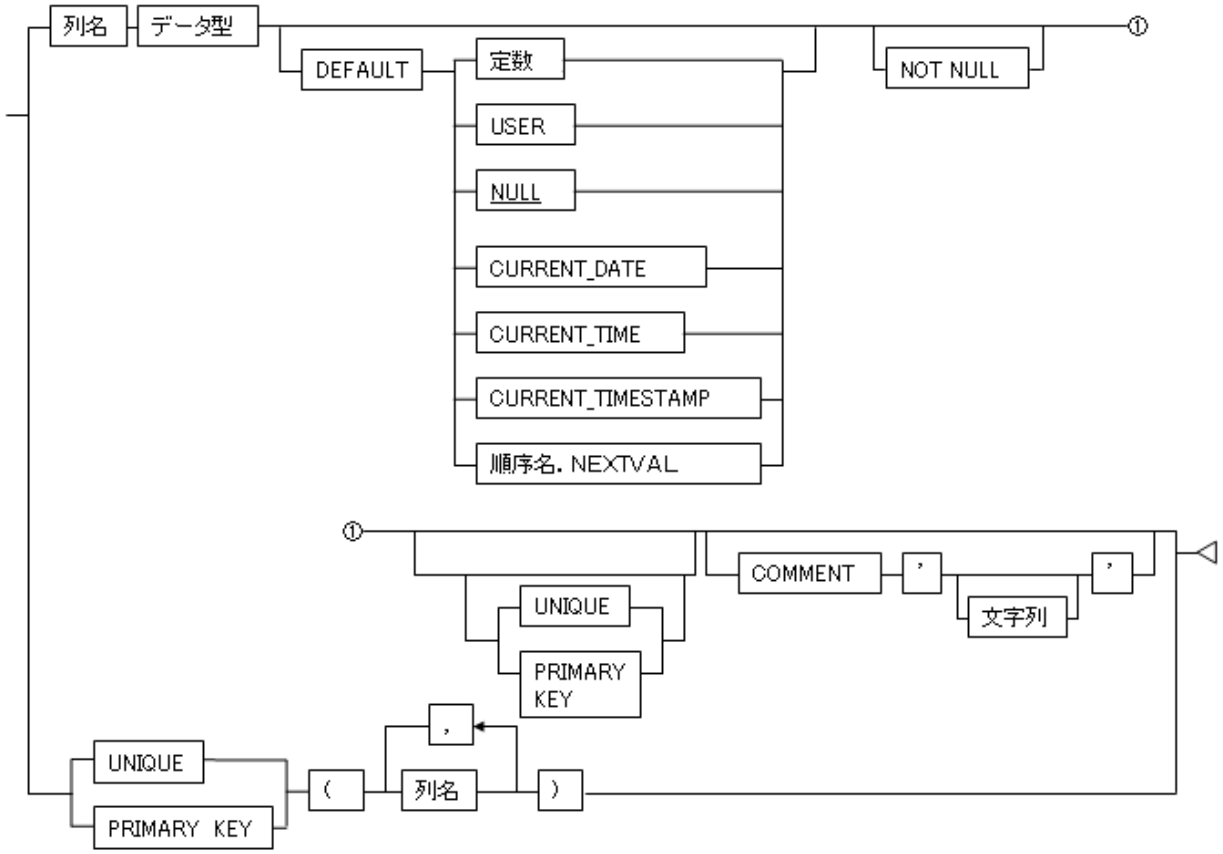
表定義は、実表および一時表の実表名と表を構成する列の属性を定義します。

記述形式

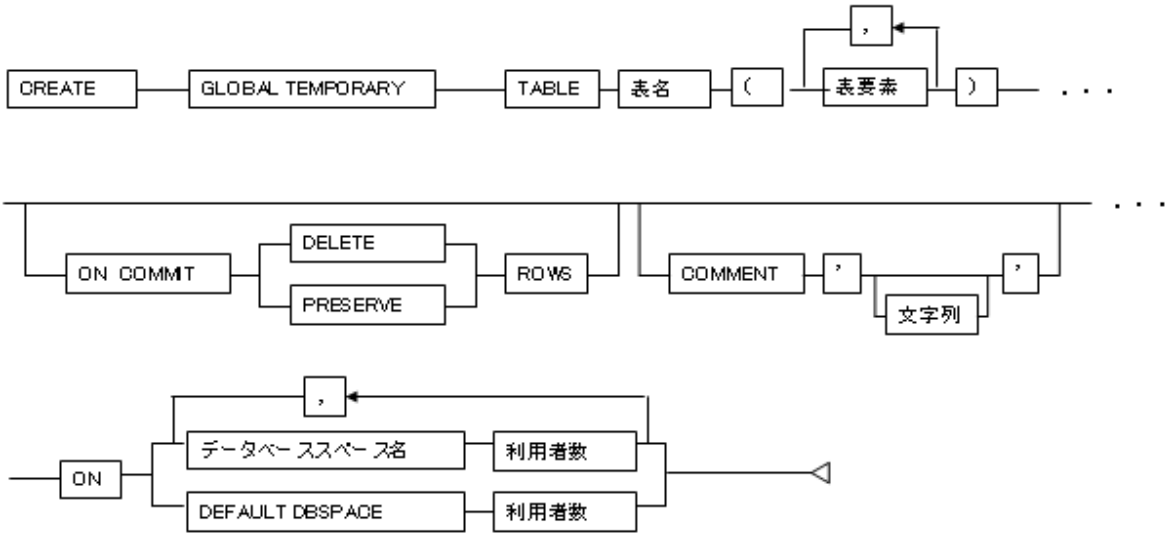
実表の場合



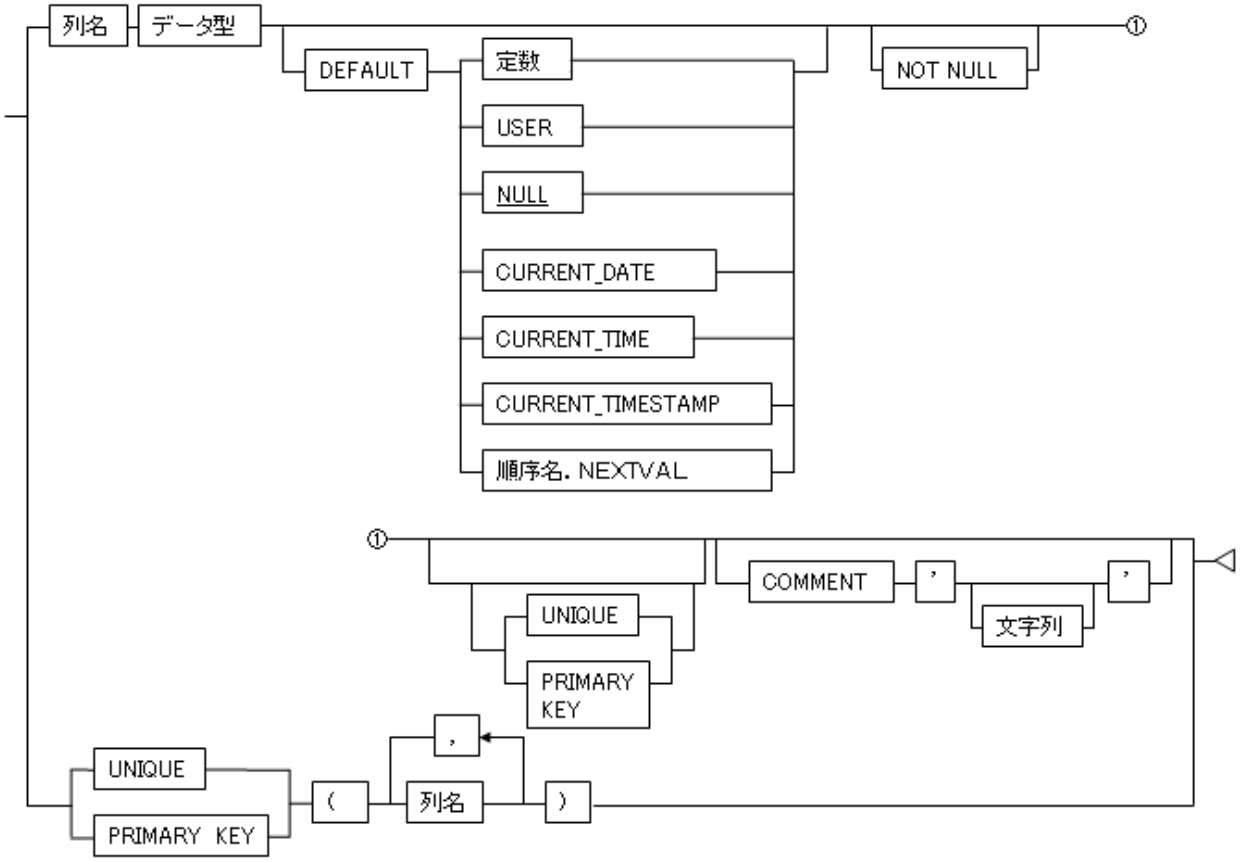
表要素


列名
データ型
DEFAULT
定数
USER
NULL
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
順序名. NEXTVAL
NOT NULL
①
UNIQUE
PRIMARY KEY
COMMENT
'
文字列
UNIQUE
PRIMARY KEY
(
,
列名
)

一時表の場合

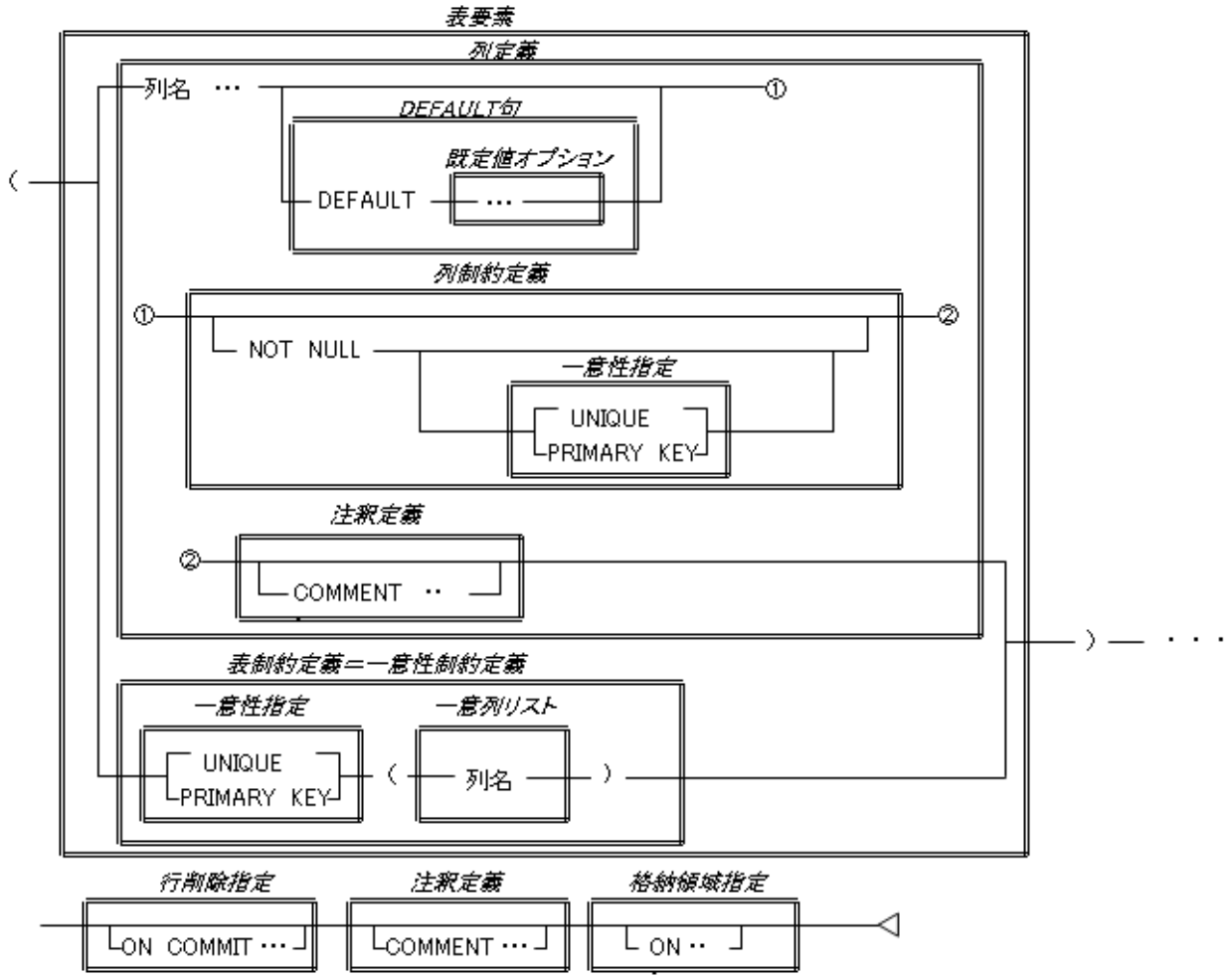

CREATE
GLOBAL TEMPORARY
TABLE
表名
(
,
表要素
)
ON COMMIT
DELETE
PRESERVE
ROWS
COMMENT
'
文字列
ON
,
データベーススペース名
利用者数
DEFAULT DBSPACE
利用者数

表要素


列名
データ型
DEFAULT
定数
USER
NULL
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
順序名.NEXTVAL
NOT NULL
①
UNIQUE
PRIMARY KEY
COMMENT
'
文字列
'
UNIQUE
PRIMARY KEY
(
,
列名
)

構文の構成


CREATE
TABLE ・・・
一時表指定
GLOBAL TEMPORARY



参照項番

- データ型 → “2.2 データ型”
- 定数 → “2.1.2 定数”
- 日本語文字列 → “2.1.3 トークン”

権限

- 表を定義できるのは、スキーマの定義者とCREATE権を付与された人のみです。
- 格納領域指定を行った場合は、データベーススペースに対するALLOCATE権が必要です。
- DEFAULT句に順序を指定する場合は、順序に対するSELECT権が必要です。

一般規則

一時表指定(GLOBAL TEMPORARY)

一時表を作成する場合に指定します。このとき、行削除指定で一時表内の行の削除時期および格納領域指定で一時表を作成するデータベーススペースまたはデフォルトデータベーススペース（DEAFAULT DBSPACE）を指定します。

なお、一時表には、SEQUENTIAL構造の格納構造(DSO、DSI)が自動的に作成されるため、格納構造を定義する必要はありません。また、作成される格納構造のDSO名、DSI名は“_TEMP”で始まる名前となります。

表名

ー 作成する表の名前を指定します。

ー 表名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

ー 表定義がスキーマ定義に含まれる場合、表名のスキーマ名の修飾を省略した場合は、スキーマ定義で指定したスキーマ名で修飾したとみなされます。また、スキーマ名で修飾する場合は、スキーマ定義で指定したスキーマ名であることが必要です。

ー 表名は、修飾したスキーマ内で一意の名前であることが必要です。

列定義

ー 列定義に指定する一意性指定は、その一意性指定を指定した一意性制約定義の一意列リストに該当列を1つだけ指定した、表制約定義と同じです。

ー 表に対して格納構造定義で分割格納をする場合、表に定義する一意性制約定義に分割キーを構成する列をすべて含むように指定することが必要です。また、列定義に一意性指定が指定できるのは分割キーの列と同じ列だけです。列定義に一意性指定を指定した場合、分割キーを構成する列の数は1つになります。

ー 列定義に行識別子“ROW_ID”は指定できません。

ー 格納構造がSEQUENTIALの表に、BLOB型の列を含む場合は、以下のようになります。

- BLOB型の列のサイズは、1キロバイトから最大2ギガバイトです。BLOB型以外の列の合計は最大32000バイトです。

ー 格納構造がOBJECTの表で、データ型にBLOB型を指定した場合は、以下のようになります。

- BLOB型の列以外の列は固定長属性であることが必要です。
- BLOB型の列はNOT NULL制約を指定することが必要です。
- BLOB型の列は表の最後の列として指定することが必要です。
- BLOB型で指定するサイズの単位は、G(ギガバイト)、M(メガバイト)およびK(キロバイト)です。
- BLOB型の列以外の列の合計は最大32000バイトです。

列名

ー 表の列の名前を指定します。

ー 列名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

DEFAULT句

ー DEFAULT句は、列の値に対する省略値を指定します。ここで指定する省略値を既定値オプションと呼びます。すなわち、表に行が挿入されるとき、該当列の値が指定されない場合に、列に設定する値を指定します。DEFAULT句に指定可能な値と、列に設定される値を以下に示します。

なお、表中のLは定義する長さを、Mは設定する定数の長さを表します。

表3.7 DEFAULT句に指定可能な値と、列に設定される値

DEFAULT句の指定	列定義のデータ型	挿入時に値が指定されない列に設定される値
なし	すべてのデータ型	NULL値
NULL	すべてのデータ型	NULL値
真数定数	真数型、概数型	定数の値
概数定数	真数型、概数型	定数の値
文字列定数(長さM)	文字列型(長さL=M)	定数の値
	文字列型(長さL>M)	データ型が固定長の場合は定数の右に空白を詰めた値。可変長の場合は定数の値。
各国語文字列定数(長さM)	各国語文字列型(長さL=M)	定数の値
	各国語文字列型(長さL>M)	データ型が固定長の場合は定数の右に空白を詰めた値。可変長の場合は定数の値。

DEFAULT句の指定	列定義のデータ型	挿入時に値が指定されない列に設定される値
日付定数	DATE型	定数の値
時刻定数	TIME型	定数の値
時刻印定数	TIMESTAMP型	定数の値
時間隔定数	INTERVAL型(年月)	定数の値
	INTERVAL型(日時)	定数の値
USER	文字列型(長さL>=18) (注)	表に行を挿入するアプリケーションのCONNECT文で指定したログイン名
日時値関数(CURRENT_DATE)	DATE型	挿入時の日付
日時値関数(CURRENT_TIME)	TIME型	挿入時の時刻
日時値関数(CURRENT_TIMESTAMP)	TIMESTAMP型	挿入時の時刻印
NEXTVAL	真数型、概数型	順序番号

注) データベースの文字コード系がUNICODEの場合、長さはL>=128です。

NOT NULL

- NOT NULLは、列がNULL値を許さないことを指定します。

一意性制約定義

- 一意性制約定義は、列または列の並びに、重複する行が発生しないことを指定します。このとき、対象となる列にはすべてNOT NULLが指定されていることが必要です。
- 一意性制約定義のうちPRIMARY KEYは、表の中で主キーとなる列または列の並びを指定します。PRIMARY KEYは、表定義中で1回だけ指定することができます。
- 一意性制約定義に指定できる列名の数は、最大64個です。
- 一意性制約された列名に対するデータ型で指定できる長さの合計は、最大1000バイトです。
- 一意性制約定義に指定した列または列の並びは、格納構造定義でインデックスの構成列とするか、あるいはクラスタキーとして定義することが必要です。このために、一意性制約にデータ型が概数型またはBLOB型の列を指定することはできません。
  ただし、データベース簡単運用の場合は、一意性制約定義に指定した列または列の並びに対応したインデックスをシステムが自動的に定義するため、インデックスを定義する必要はありません。なお、インデックス名は“@”、システムで採番する10桁の数字と“_PK”(PRIMARY KEYの場合)または“_UK”(UNIQUEの場合)を使用して命名します。
- 表に対して格納構造定義で分割格納をする場合、表に定義する一意性制約定義に分割キーを構成する列をすべて含むように指定することが必要です。また、列定義に一意性指定が指定できるのは分割キーの列と同じ列だけです。

  表に対して格納構造定義で分割をする場合、一意性制約は分割キーの指定を考慮してこれらの規則を満たすように指定してください。特に列定義に一意性制約を指定する場合は、分割キーとして、その列以外を指定できなくなるので注意してください。

行削除指定(ON COMMIT)

- 一時表内の行の削除時期を指定します。DELETE ROWSを指定するか、または本指定を省略した場合は、一時表はトランザクション内で利用できる表となり、一時表に格納されるデータは、トランザクション終了時に削除されます。
  PRESERVE ROWSを指定した場合は、一時表はセション内で利用できる表となり、一時表に格納されるデータは、セション終了時に削除されます。

注釈定義(COMMENT)

— 表および列に対して注釈が定義できます。

— 注釈は256バイト以内の文字列(日本語記述可能)を指定することができます。

格納領域指定(ON)

— 格納領域指定は、表のデータの格納先のデータベーススペースまたはデフォルトデータベーススペース(DEAFAULT DBSPACE)を指定します。

- 指定されたデータベーススペースにSymfoware/RDBが表の格納構造を定義します。なお、指定されたデータベーススペースは、定義済みであることが必要です。

  定義される格納構造については、"D.3 格納領域指定時の格納構造"を参照してください。

  1つの実表を複数のデータベーススペースに分割して格納するような場合には、格納領域指定を行うことができません。このような場合には、格納構造定義文でデータベーススペースを定義します。

- 格納領域指定を指定する場合、システム用の動作環境ファイルでDEFAULT_DSI_NAME=CODEを指定した場合は、表名およびスキーマ名に、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。CODEを指定しない場合は、8文字以内で定義します。

- 格納領域指定を指定する場合、デフォルトデータベーススペースに表を作成する場合または一時表を作成する場合の表名は、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

- デフォルトデータベーススペースに実表を定義する場合は、格納領域指定は指定できません。

データベーススペース名

— データベーススペース名は、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

利用者数

— データベーススペースまたはデフォルトデータベーススペースの利用者数を1～32767の範囲で指定します。各データベーススペースの利用者数の総和が、一時表を利用するアプリケーションの多重度となるように利用者数を指定してください。一時表を利用するアプリケーションは、セション終了まで利用者数にカウントされます。

使用例

例1

スキーマS1に実表"会社表"を定義します。

```
CREATE TABLE S1.会社表 (会社番号  SMALLINT NOT NULL PRIMARY KEY,
                       会社名    NATIONAL CHARACTER(10) NOT NULL,
                       電話番号  CHARACTER(14),
                       住所      NATIONAL CHARACTER(20))
```

例2

スキーマS1に実表"発注表"を定義します。なお、取引先、取引製品の並びを主キーとします。また、発注数量の省略値を10とします。

```
CREATE TABLE S1.発注表 (取引先     SMALLINT NOT NULL,
                       取引製品   SMALLINT NOT NULL,
                       仕入価格   INTEGER,
                       発注数量   SMALLINT DEFAULT 10,
                       PRIMARY KEY(取引先,取引製品))
```

例3

スキーマS1に実表"製品写真表"を定義します。

```
CREATE TABLE S1.製品写真表 (製品番号  SMALLINT NOT NULL PRIMARY KEY,
                            商品写真  BLOB(2G) NOT NULL)
```

例4

スキーマS1に一時表“一時在庫表”を定義します。一時在庫表の利用者数を10とします。

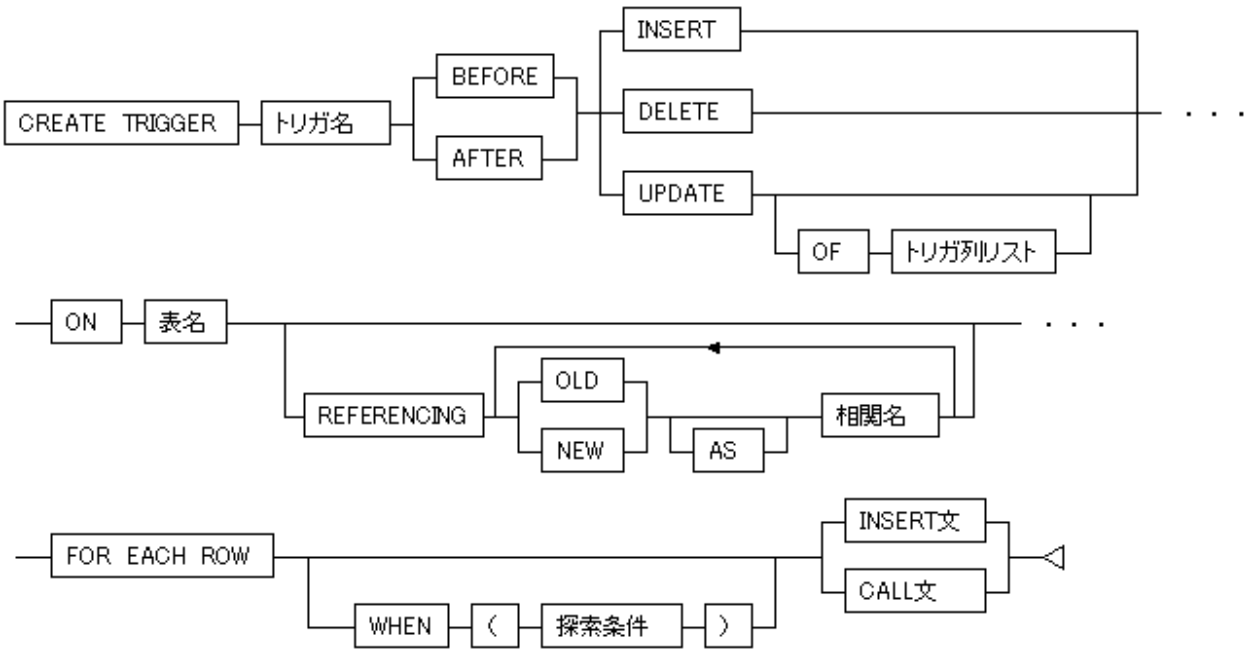
```
CREATE GLOBAL TEMPORARY TABLE S1.一時在庫表
      (製品番号      SMALLINT NOT NULL,
       製品名        NATIONAL CHARACTER(10) NOT NULL,
       在庫数量      INTEGER,
       倉庫番号      SMALLINT,
       PRIMARY KEY(製品番号))
       ON DBSPACE1 10
```

3.23 CREATE TRIGGER文(トリガ定義)

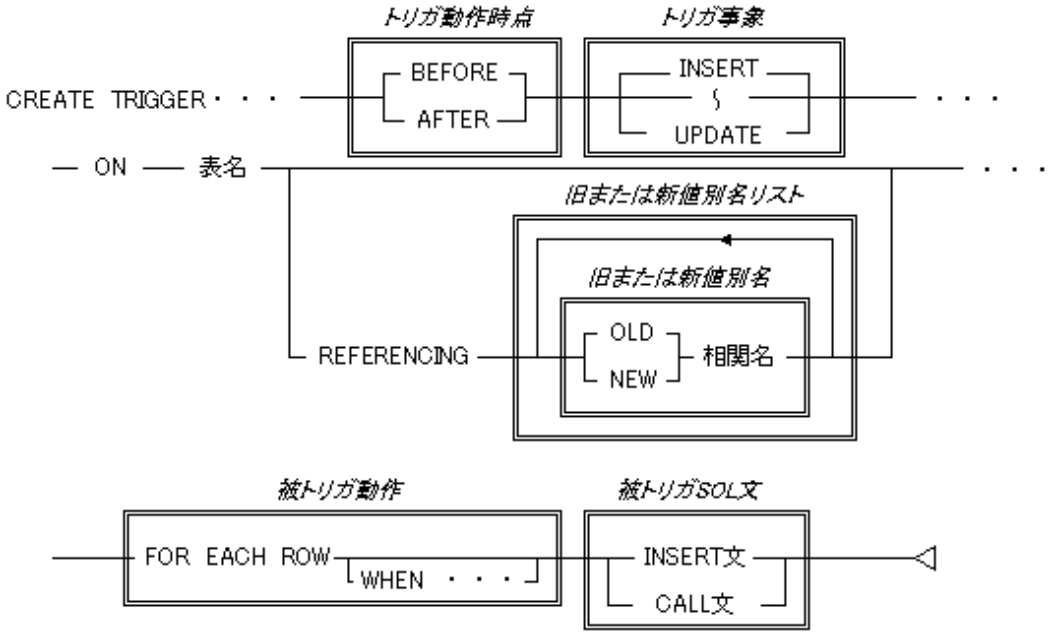
機能

トリガ定義は、データ操作に連動したほかの表へのデータ操作を定義します。

記述形式



構文の構成



参照項番

- 探索条件 → “2.13 探索条件”
- INSERT文 → “3.48 INSERT文”
- CALL文 → “3.5 CALL文”
- 日本語文字列 → “2.1.3 トークン”

権限

- トリガ定義の実行者は、以下の条件を満たすことが必要です。
  - ― スキーマに対するCREATE権を持っていることが必要です。
  - ― 表名で指定された表に対するTRIGGER権を持っていることが必要です。
  - ― 被トリガSQL文中に指定された表に対して、実施する操作に必要な各権限を持っていることが必要です。
  - ― 被トリガSQL文中に順序を指定した場合は、順序に対するSELECT権が必要です。
  - ― CALL文を指定する場合は、プロシジャルーチンに対するEXECUTE権が必要です。

一般規則

- CREATE TRIGGER文を準備可能文として使用する場合、ホスト変数または動的パラメタを指定できません。

トリガ名

- ― 作成するトリガの名前を指定します。
- ― トリガ名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- ― トリガ名は、スキーマ内で一意の名前であることが必要です。
- ― トリガ定義がスキーマ定義に含まれる場合、トリガ名のスキーマ名の修飾を省略した場合は、スキーマ定義で指定したスキーマ名で修飾したとみなされます。また、スキーマ名で修飾する場合は、スキーマ定義で指定したスキーマ名であることが必要です。

トリガ動作時点

BEFOREを指定した場合:

表名に指定した表を更新する前に被トリガ動作が実行されます。

AFTERを指定した場合:

表名に指定した表を更新したあとに被トリガ動作が実行されます。

トリガ事象

INSERTを指定した場合:

INSERT文で表に行を挿入すると、被トリガ動作が実行されます。

DELETEを指定した場合:

DELETE文で表の行を削除すると、被トリガ動作が実行されます。

UPDATEを指定した場合:

UPDATE文で表の列の値を更新すると、被トリガ動作が実行されます。トリガ列リストを指定した場合、指定した列のいずれかを更新すると、被トリガ動作が実行されます。

トリガ列リスト

- トリガ列リストは列名をカンマ(,)で区切って指定します。
- トリガ列リストの列名は同一の列名を指定することはできません。
- トリガ列リストに指定する列は、表名で指定した表に定義されていることが必要です。

表名

- トリガを作成する対象の表の名前を指定します。
- 表名にビュー表を指定することはできません。
- 表名は、定義するトリガと別のデータベースに属する表を指定することはできません。
- 表名および被トリガSQL文の表名にOBJECT構造の表を指定することはできません。

REFERENCING句

- REFERENCING句のOLDに指定する相関名を“旧値相関名”、NEWに指定する相関名を“新値相関名”と呼びます。
- 旧または新値別名は、表の状態としてトリガ事象の実行前後を識別するための別名です。旧または新値別名リストは、旧または新値別名の並びです。
- REFERENCING句を指定することにより、旧値相関名あるいは新値相関名を、OLDあるいはNEWから別のものに変更することができます。
- REFERENCING句を指定する場合、旧値相関名と新値相関名は互いに同じであってはなりません。また、被トリガSQL文中で指定された表名と同じであってはなりません。

相関名

- 相関名は、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列指定します。
- トリガ事象と、列を修飾する相関名の関係を以下に示します。

トリガ事象	列を修飾する相関名
INSERT	旧値相関名は指定できません。
DELETE	新値相関名は指定できません。
UPDATE	旧値相関名と新値相関名が指定できます。

被トリガ動作

- 被トリガ動作は、被トリガSQL文を実行する条件を指定します。

－ WHEN句の探索条件に副問合せおよびROWNUMを指定することはできません。

被トリガSQL文

－ 被トリガSQL文には問合せ指定を含まないINSERT文、または、CALL文を指定することができます。

－ 被トリガSQL文にCALL文を指定することにより、複数の文からなる一連の処理を被トリガSQL文として実行することができます。

－ 被トリガSQL文にCALL文を指定する場合、CALL文で呼び出されるプロシジャルーチンではトランザクション管理文を実行することができません。トランザクション管理文を実行するとプロシジャルーチン中のCOMMIT文やROLLBACK文の実行がエラーになります。

－ WHEN句の探索条件、あるいは被トリガSQL文に指定したトリガ対象表の列は、それがトリガ事象を実行する前の値か実行したあとの値かを示すため、旧値相関名であるOLD、あるいは新値相関名であるNEWで修飾しなければなりません。

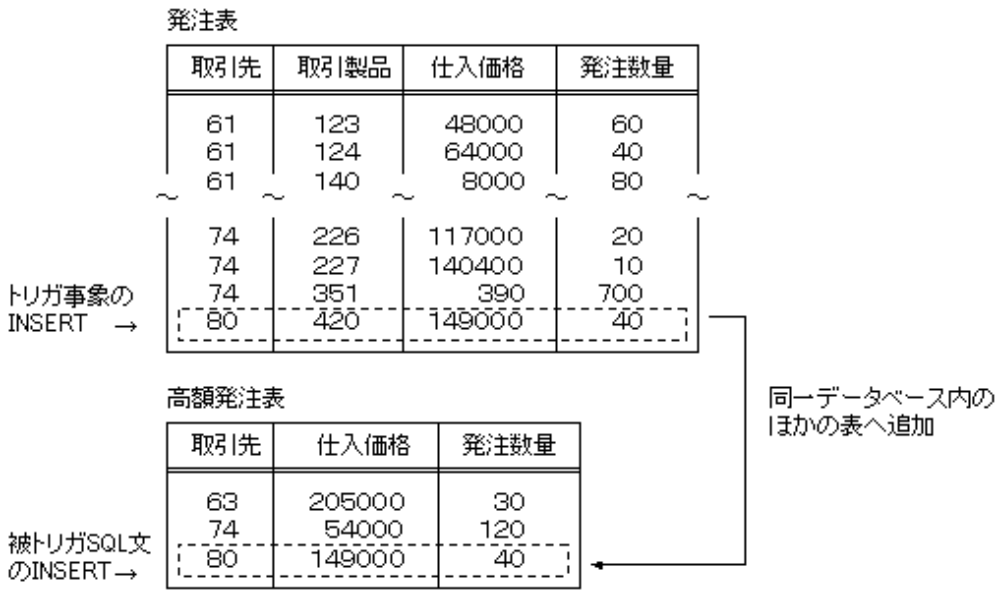
－ 被トリガSQL文にCALL文を指定する場合、CALL文で示すプロシジャルーチンのパラメタのパラメタモードは、INでなければなりません。

使用例

例1

発注表に追加された行の発注価格が500万を超えている場合、同時に、その製品の取引先、仕入価格および発注数量を高額発注表に追加します。なお、発注価格は、仕入価格×発注数量で求められます。

```
CREATE TRIGGER 在庫管理.発注トリガ
    AFTER INSERT ON 在庫管理.発注表
    REFERENCING NEW AS NEWREC
    FOR EACH ROW
        WHEN (NEWREC.仕入価格 * NEWREC.発注数量 > 5000000)
            INSERT INTO 在庫管理.高額発注表
                    VALUES(NEWREC.取引先,NEWREC.仕入価格,NEWREC.発注数量)
```



例2

在庫表の在庫数量を0より小さい値に変更しようした場合、SQL文をエラーにする処理をトリガの定義とプロシジャの定義で行います。

プロシジャの定義

```
CREATE PROCEDURE S1.在庫チェックルーチン(IN 在庫数量Ｖ INT)
BEGIN
  DECLARE SQLSTATE CHAR(5);
  DECLARE SQLMSG   CHAR(256);
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
    RESIGNAL;
    END;
   -- ルーチン本体
  IF (在庫数量Ｖ < 0) THEN
    SIGNAL SQLSTATE'60001' '在庫数量不当';
  END IF;
END
```

トリガの定義

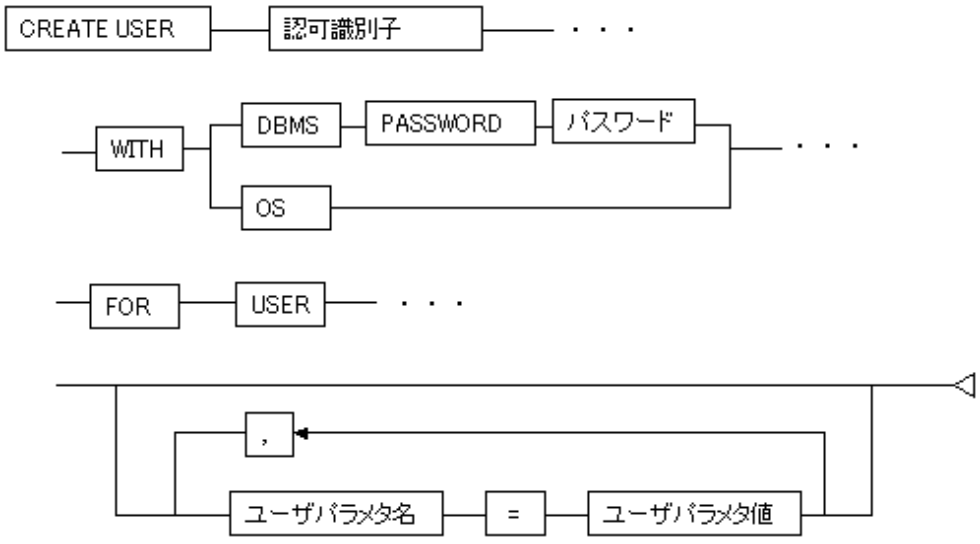
```
CREATE TRIGGER S1.在庫数チェック
  BEFORE UPDATE OF 在庫数量 ON S1.在庫表
  FOR EACH ROW
  CALL S1.在庫チェックルーチン(NEW.在庫数量)
```

3.24 CREATE USER文(利用者定義文)

機能

データベースシステムにアクセスする利用者を定義します。

記述形式



参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- 利用者を定義できるのは、RDBディクショナリの定義者のみです。

一般規則

・ 利用者登録の使用宣言をしている場合に実行できます。利用者登録の使用宣言はSET SYSTEM PARAMETER文で指定します。

認可識別子

― RDBシステムに登録する利用者の名前を指定します。

― 認可識別子は、18文字以内の先頭が英字で始まる英数字、または9文字以内の日本語文字列を指定します。

― 認可識別子は、RDBシステム内で一意でなければなりません。

― 英小文字の区切り識別子を認可識別子に指定することはできません。

WITH句

― DBMSを指定した場合は、利用者の認可識別子とパスワードをSymfoware/RDBで管理します。

― OSを指定した場合は、認可識別子をOSのユーザ名に対応させて管理します。

― USER_CONTROL=YESを指定している場合、rdbuptコマンドおよびrdbunlsqlコマンドは、CREATE USER文のWITH句でOSを指定した利用者のみ実行することができます。

PASSWORD句

パスワードは、以下の形式で指定します。

― パスワードは、以下の文字で構成される文字列定数で指定します。

- 英字
- 数字
- 特殊文字
- 拡張文字
- “^”

> **注意**
>
> このパスワードをJDBCドライバから使用して認証する場合は、“;”(セミコロン)を使用できません。

― パスワードは、最低長以上8バイト以下の長さで指定します。最低長はユーザパラメタ“MIN_PASSWORD_SIZE”で変更することができます。

― パスワードは、文字列中に2つ以上の英字を含まなければなりません。また、1つ以上の数字、特殊文字または拡張文字を含まなければなりません。

― パスワードは利用者名と比較して、同じもの、ずらしたものまたは反転したものであってはなりません。

FOR句

― 認可識別子を、利用者として定義します。

ユーザパラメタ名

― 利用者の認証情報のユーザパラメタのパラメタ名を指定します。指定可能なユーザパラメタは以下のとおりです。

- MAX_CONNECTION(注)
- MAX_MEMORY_USE(注)
- MAX_WORKFILE_USE(注)
- MAX_WORKFILE_NUM(注)
- MAX_TRAN_TIME(注)
- MAX_TRAN_MEM(注)

- MAX_WAIT_TIME(注)
- PASSWORD_CHANGE_TIME
- PASSWORD_LIMIT_TIME
- INVALID_PASSWORD_WAIT_TIME
- INVALID_PASSWORD_TIME
- MIN_PASSWORD_SIZE

注) Symfoware Server Enterprise Extended Editionの場合のみ指定可能です。

― ユーザパラメタ名の指定は、WITH句にDBMSを指定して、利用者の認可識別子とパスワードをSymfoware/RDBで管理する場合のみ有効となります。

― 利用者の認証情報のユーザパラメタは、SET SYSTEM PARAMETER文で設定した利用者全員のユーザパラメタのデフォルト値を、利用者ごとに設定する場合に指定します。

利用者の認証情報のユーザパラメタの詳細については、“3.61 SET SYSTEM PARAMETER文”を参照してください。

ユーザパラメタ値

― 利用者の認証情報のユーザパラメタのパラメタの値を指定します。

使用例

例1

認可識別子“SUZUKI”をデータベース専用利用者として定義します。

```
CREATE USER SUZUKI
  WITH DBMS PASSWORD 'ABC001@'
  FOR USER
```

例2

認可識別子“SATO”に対して、パスワードの期限“PASSWORD_LIMIT_TIME”を60日に定義します。

```
CREATE USER SATO
  WITH DBMS PASSWORD '888##FF'
  FOR USER
  PASSWORD_LIMIT_TIME=60
```

例3

認可識別子“SATO”に対して、1つのコネクションで使用可能なメモリ量“MAX_MEMORY_USE”を8メガバイト、1つのコネクションで使用可能な作業用ファイルの量“MAX_WORKFILE_USE”を100メガバイトに定義します。

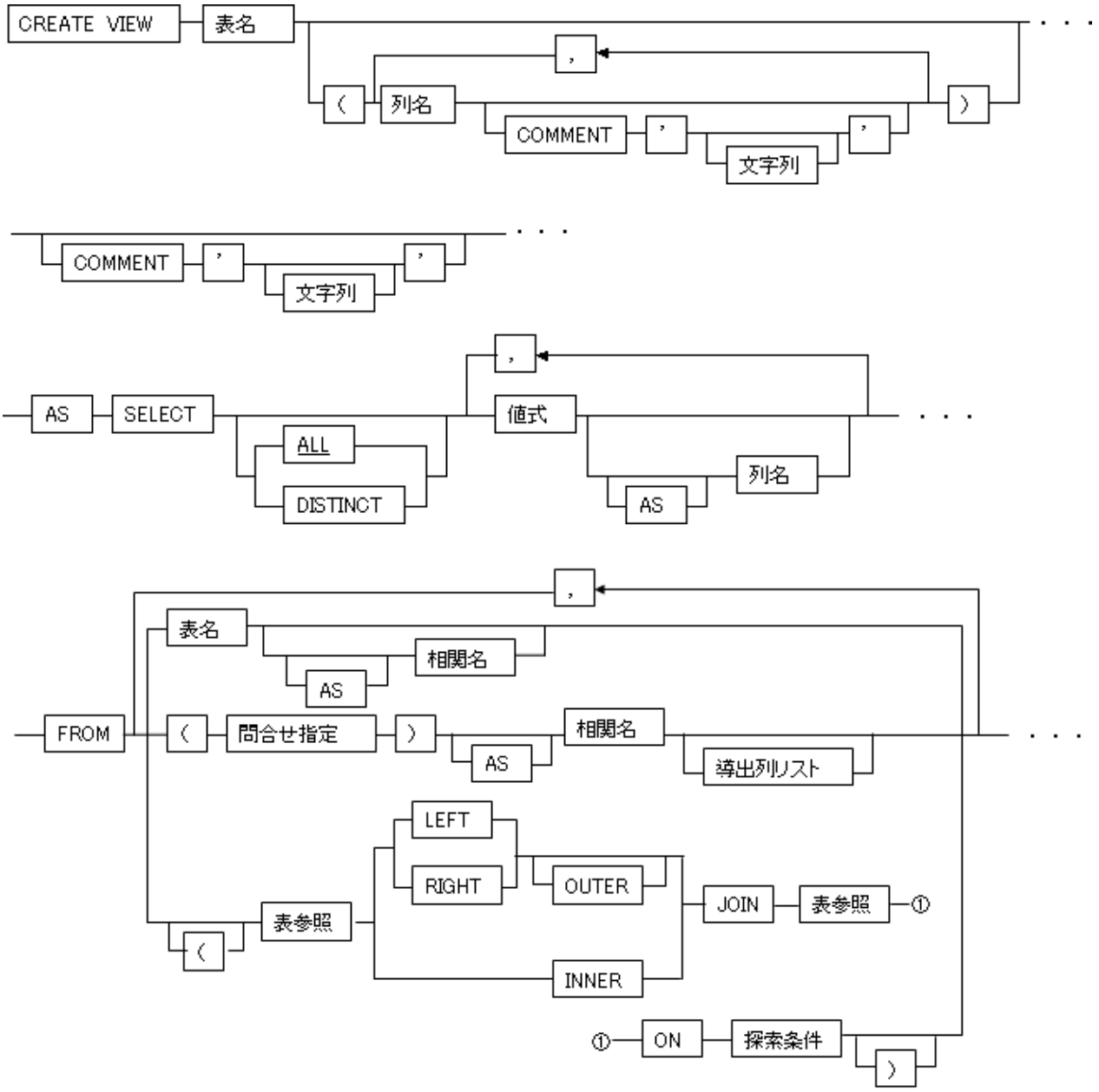
```
CREATE USER SATO
  WITH DBMS PASSWORD '888##FF'
  FOR USER
  MAX_MEMORY_USE=8,
  MAX_WORKFILE_USE=100
```

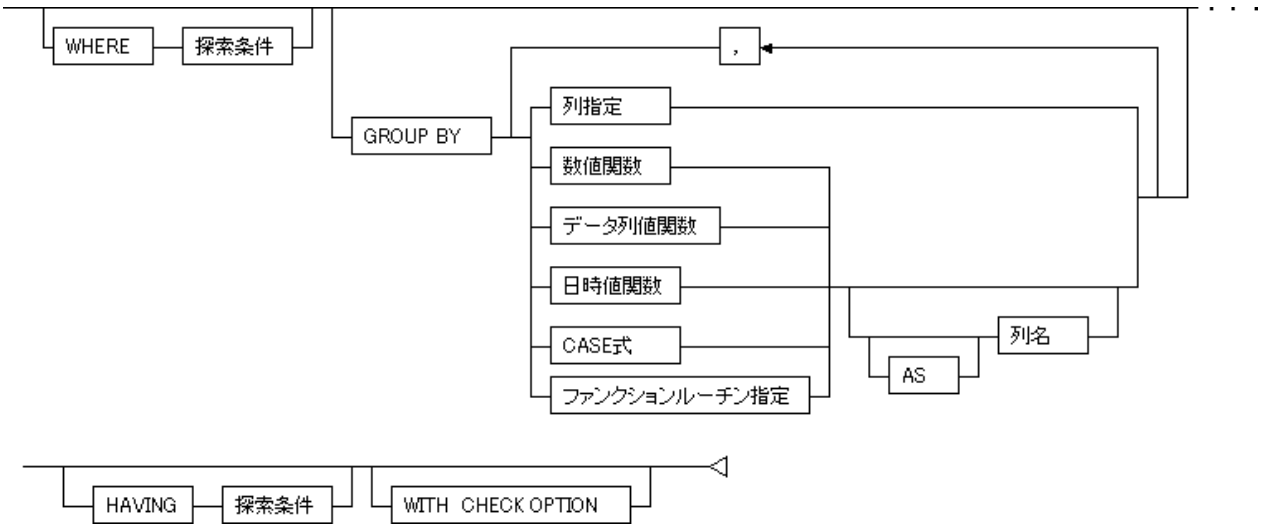
3.25 CREATE VIEW文(ビュー定義)

機能

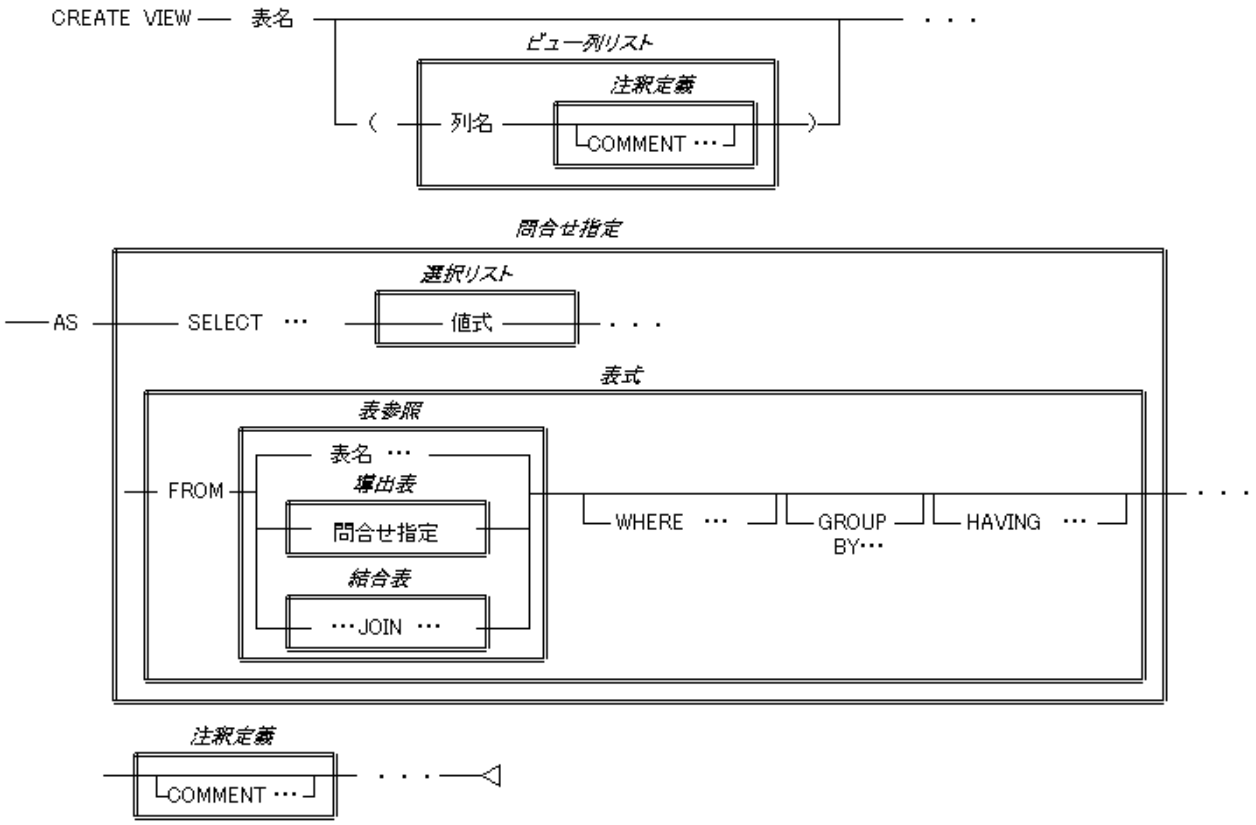
ビュー定義は、実表を基にビュー表を定義します。

記述形式


CREATE VIEW
表名
,
(
列名
)
COMMENT
'
文字列
'
COMMENT
'
文字列
'
AS
SELECT
,
値式
ALL
DISTINCT
AS
列名
,
表名
AS
相関名
FROM
(
問合せ指定
)
AS
相関名
導出列リスト
(
表参照
LEFT
RIGHT
OUTER
INNER
JOIN
表参照
①
①
ON
探索条件
)



構文の構成



参照項番

- 値式 → “2.11 値式”
- 探索条件 → “2.13 探索条件”
- 列指定 → “2.4 列指定”
- 数値関数 → “2.5.2 数値関数”
- データ列値関数 → “2.5.3 データ列値関数”
- 日時値関数 → “2.5.4 日時値関数”
- CASE式 → “2.6 CASE式”

- ファンクションルーチン指定→ “2.5.5 ファンクションルーチン指定”
- 日本語文字列 → “2.1.3 トークン”

権限

- ビュー表を定義するには、問合せ指定に指定した表に対するSELECT権が必要です。
- ビューを定義することで、ビューの定義者がビューの導出元の表に対して保持している権限は、ビュー表に対しても付与されます。また、権限をほかの人に与える権限(付与権)についても、ビューの定義者がビューの導出元の表に対して保持している権限が付与権付きの場合は付与権付きで、そうでない場合は付与権なしで付与されます。
- ビュー表を定義できるのは、スキーマの定義者とCREATE権を付与された人のみです。
- ビュー表の問合せ指定にファンクションルーチンを指定した場合は、ファンクションルーチンに対するEXECUTE権が必要です。

一般規則

- CREATE VIEW文を準備可能文として使用する場合、ホスト変数または動的パラメタを指定できません。
- CREATE VIEW文にROWNUMは指定できません。

表名

- ビュー表の名前を指定します。
- 表名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- 表名は、修飾したスキーマ内で一意な名前であることが必要です。また、“RDBII”で始まる表名は指定できません。
- ビュー定義がスキーマ定義に含まれる場合で、ビューの表名のスキーマ名の修飾を省略した場合は、スキーマ定義で指定したスキーマ名で修飾したとみなされます。また、スキーマ名で修飾する場合は、スキーマ定義で指定したスキーマ名であることが必要です。
- ビューの表名を修飾するスキーマ名と問合せ指定中の表名を修飾するスキーマ名は、同一のデータベース中に定義されていることが必要です。

ビュー列リスト

- ビュー列リストは、ビュー表を構成する列の名前を、カンマ(,)で区切って記述します。
- ビュー列リストの指定方法は以下のとおりです。
  - 列名は、ビュー表内で一意な名前であることが必要です。
  - 列名の数は最大32,000個です。
  - 列名の個数は、問合せ指定に指定する選択リストの個数と同じであることが必要です。
- ビュー列リストに行識別子“ROW_ID”は指定できません。
- 問合せ指定に以下の条件を含んでいる場合、ビュー列リストを省略することはできません。条件を含まない場合、ビュー列リストを省略することができます。このとき、問合せ指定の選択リストに指定した列名が、ビュー表を構成する列名となります。なお、選択リストにAS句を指定した場合、AS句の列名がビュー表を構成する列名となります。
  - 選択リストに列名以外を指定して、AS句を指定していない。
  - 選択リストに同一の列名を指定している。

列名

- 列名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

注釈定義(COMMENT)

- ビュー表および列に対して注釈が定義できます。
- 注釈は256バイト以内の文字列(日本語記述可能)を指定することができます。

問合せ指定

- 問合せ指定の文字列の最大長は32,000バイトです。
  システム表の表RDBII_SYSTEM.RDBII_DESCRIPTIONの列DESC_VALUEには30,000バイトまで格納します。
- 問合せ指定の結果を以下に示します。
  - 表式の結果がグループ表でない場合、問合せ指定の結果は次のようになります。選択リストに集合関数指定がない場合、表式の結果の各行に選択リストの値式で示す演算が施された、行の集合となります。選択リストに集合関数の指定がある場合の結果は、表式の結果を集合関数に適用した1行となります。
  - 表式の結果がグループ表の場合、問合せ指定の結果は次のようになります。結果の行の数はグループの数となります。各行の値は、選択リストに集合関数の指定がある場合には、グループを集合関数に適用した結果となり、列指定の場合には、グループ化列の値となり、グループ化関数の場合は、グループ化関数の値となります。なお、選択リストにGROUP BY句のAS句の列名を指定した場合は、対応するグループ化列、グループ化関数またはCASE式の値となります。
  - 表式の結果が空であれば、問合せ指定の結果は、空の表となります。
- 問合せ指定が更新可能ならば、ビュー表は更新可能な表となります。そうでなければ、ビュー表は読込み専用の表となります。
- 問合せ指定の中に順序のCURRVALおよびNEXTVALを指定することはできません。
- 問合せ指定の中にASSIST指定を指定することはできません。
- 問合せ指定にXMLQUERY関数を指定することはできません。
- 問合せ指定にXMLEXISTS述語を指定することはできません。

DISTINCTおよびALL

- DISTINCTは、問合せ指定の結果の各行同士がまったく同じ値を持つ行(重複行)がある場合、それらの行を1行とする場合に指定します。このとき、NULL値同士は等しいとみなされます。ALLは、重複行をそのまま残す場合に指定します。また、どちらも省略した場合は、ALLが指定されたものとみなします。

選択リスト

- 選択リスト中に含まれる列指定は、FROM句で指定した表の列またはGROUP BY句のAS句の列名であることが必要です。
- 選択リストに格納構造がOBJECTのBLOB型の列を複数指定することはできません。
- 表式の結果がグループ表の場合、選択リストの列指定は、グループ化列またはGROUP BY句のAS句の列名を指定するか、集合関数指定またはグループ化関数の引数で指定することが必要です。
- 表式の結果がグループ表でなく、選択リストに集合関数を指定する場合は、選択リストのすべてが集合関数指定であることが必要です。
- 選択リストのすべてが、定数または変数のみではいけません。必ず、列指定を含む値式を1つは指定することが必要です。
- 問合せ指定の選択リストに“*”を指定することはできません。

導出表

- 導出表にORDER BY句を指定することはできません。

WITH CHECK OPTION

- WITH CHECK OPTIONを指定した場合、問合せ指定の探索条件が真とならないようなINSERT文、UPDATE文:位置づけまたはUPDATE文:探索の実行は、例外(制約違反)となります。
- WITH CHECK OPTIONを指定するとき、ビュー表は更新可能であることが必要です。

その他の構文要素

その他の構文要素の説明は、“3.26 DECLARE CURSOR(カーソル宣言)”を参照してください。

使用例

例

実表“会社表”からビュー表“会社表1”を定義します。このビュー表の定義者は、実表“会社表”のSELECT権とINSERT権を付与権付きで保持しているとします。

```
CREATE VIEW  S1.会社表１ AS
    SELECT 会社名,電話番号,住所 FROM S1.会社表 WHERE 会社番号 <= 70
```

ビュー表 会社表1

会社名	電話番号	住　所
アダム電気	111－777－4444	東京都 小田区 新蒲田 7－8－9
アイデア商事	222－888－5555	神奈川県 小浜市 旭区 1－2－3
大月産業	333－999－6666	埼玉県 浦和町 大崎 1－1－1

備考.ビュー表の定義者に対してビュー表“会社表1”のSELECT権とINSERT権がそれぞれ付与権付きで付与され、UPDATE権とDELETE権は付与されません。

3.26 DECLARE CURSOR(カーソル宣言)

機能

カーソルを定義します。

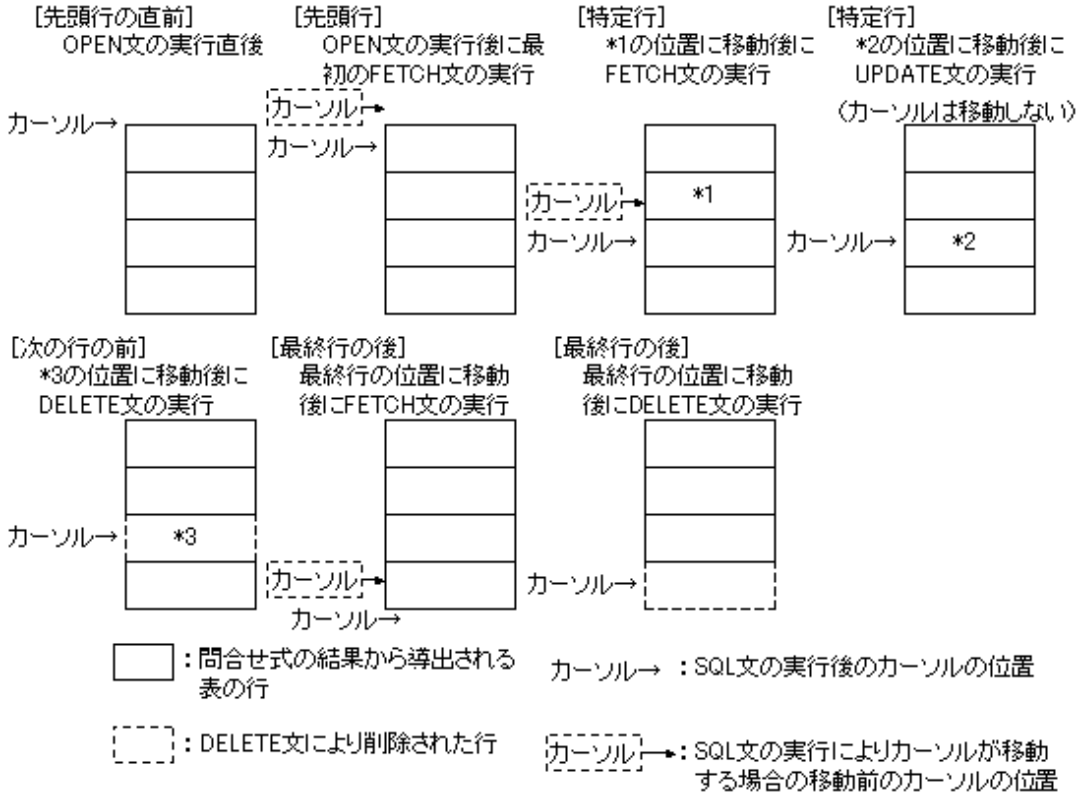
カーソルとは

ホスト言語プログラムは、表の複数行を一度に取り出して処理することはできません。これを解決するために、複数行のうちの1行をホストプログラムから、参照したり更新したりすることができるようにする手段としてカーソルがあります。カーソルは、一連の流れに対応したSQL文を使用して処理します。カーソルの使用方法を以下に示します。

- カーソルはカーソル宣言(DECLARE CURSOR)で、問合せ式を指定して行の集合を宣言します。
- カーソルをOPEN文で開くことにより、問合せ式の結果である行の集合から1行を取り出す準備ができます。
- 複数回のFETCH文の実行により、複数行をホストプログラムに取り出すことができます。
- FETCH文で取り出した行をUPDATE文:位置づけで更新したり、DELETE文:位置づけで削除したりすることができます。
- 最終行まで処理したカーソルは、CLOSE文により閉じます。

カーソルはOPEN文によって開かれた状態になり、FETCH文により次の行に移動していきます。カーソルの位置は、SQL文の実行により以下のように移動します。

図3.1 SQL文の実行によるカーソル位置の移動



あるカーソルと別のカーソルのカーソル指定に同じ表を指定した場合、または、あるカーソルのカーソル指定に指定した表と同じ表を非カーソル系のSQL文に指定した場合、先行するそのカーソルの操作によっては、後続の操作(別のカーソルによる操作、または非カーソル系のSQL文の実行)が不可能な場合があります。ただし、ここでいう表とは、SQL文に指定した表がビュー表の場合、そのビュー表に含まれる表のことです。この関係を以下に示します。

表3.8 同一の表に複数のカーソル系のSQL文または非カーソル系のSQL文でアクセスした場合の実行結果

				後続操作								
				カーソル系			非カーソル系					
				参照	更新		参照		更新			
				OPEN CSR2	UPDATE ・・・ CSR2	DELETE ・・・ CSR2	SELECT ・・・ FROM T1	INSERT INTO ・・・ SELECT FROM T1	INSERT INTO T1 ・・・	UPDATE T1 ・・・	DELETE FROM T1 ・・・
先行操作	カーソル系	参照	OPEN CSR1	○	× (注1)		○		× (注2)		
		更新	UPDATE ・・・	× (注3)			× (注4)		× (注5)		

				後続操作								
				カーソル系			非カーソル系					
				参照	更新		参照		更新			
				OPEN CSR2	UPDATE ・・・ CSR2	DELETE ・・・ CSR2	SELECT ・・・ FROM T1	INSERT INTO ・・・ SELECT FROM T1	INSERT INTO T1 ・・・	UPDATE T1 ・・・	DELETE FROM T1 ・・・
			CSR1								
			DELETE ・・・ CSR1								

○: 後続する操作は実行可能です。

×: 後続する操作は実行できません。

注1) 表T1を参照するカーソルCSR1をオープン後、カーソルCSR2により表T1を更新すると、次にカーソルCSR1のFETCH文で表T1の行を取り出すときに、更新前の状態で取り出されることと、更新後の状態で取り出されてしまうことがあります。このためカーソルCSR2による表T1の更新は、カーソルCSR1をクローズしてから実行してください。

注2) 表T1を参照するカーソルCSR1をオープン後、非カーソル系のSQL文により表T1の行を更新すると、次にカーソルCSR1のFETCH文で表T1の行を取り出すときに、更新前の状態で取り出されることと、更新後の状態で取り出されてしまうことがあります。このため非カーソル系のSQL文による表T1の更新は、カーソルCSR1をクローズしてから実行してください。

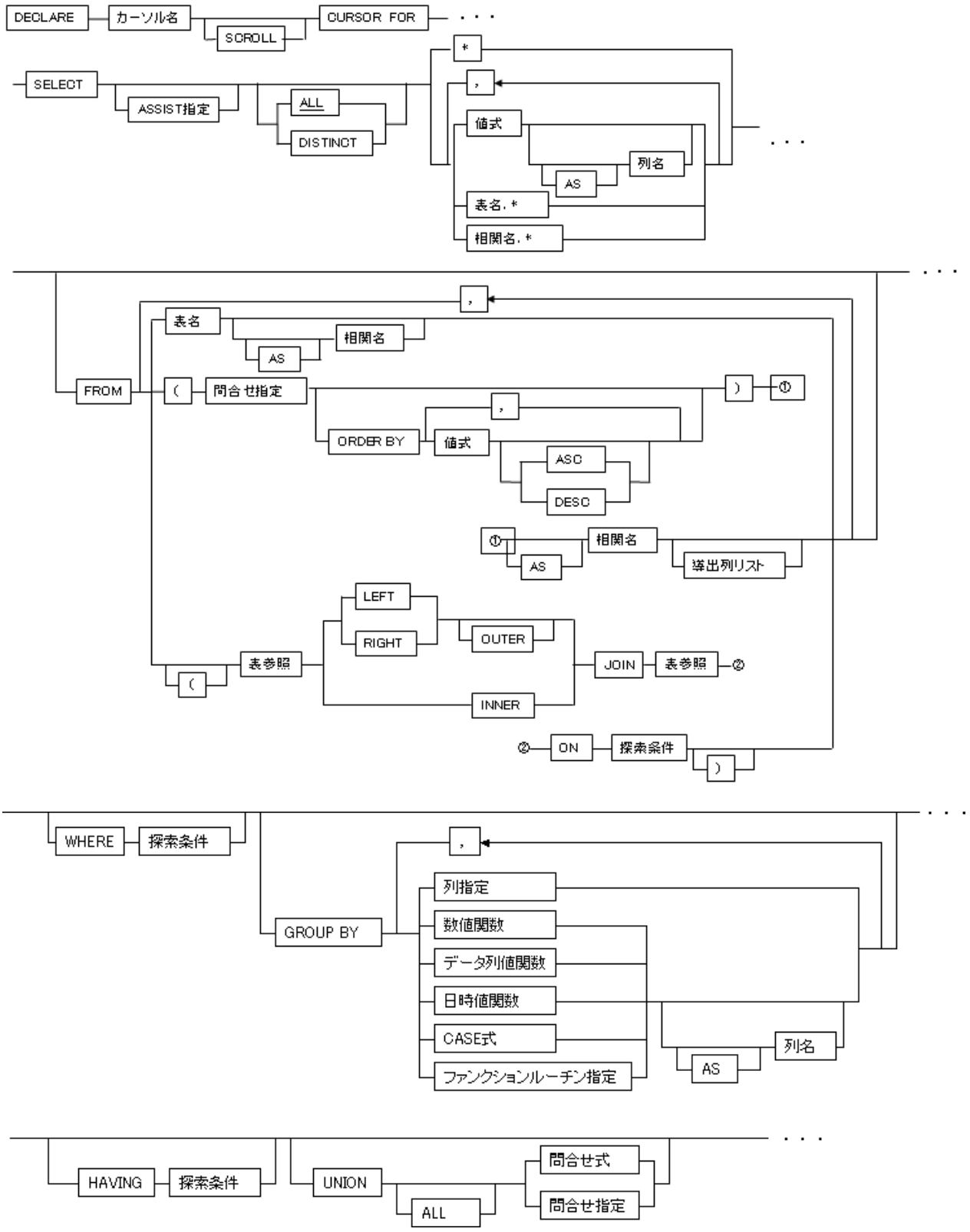
注3) カーソルCSR1により表T1を更新後、カーソルCSR2をオープンすると、次にカーソルCSR2のFETCH文で表T1の行を取り出すときに、更新前の状態で取り出されることと、更新後の状態で取り出されてしまうことがあります。このためカーソルCSR2のオープンは、カーソルCSR1をクローズしてから実行してください。なお、カーソルCSR1により表T1を更新後、カーソルCSR2で表T1を更新すると、データベースとカーソルの間で矛盾を起こすことがあります。

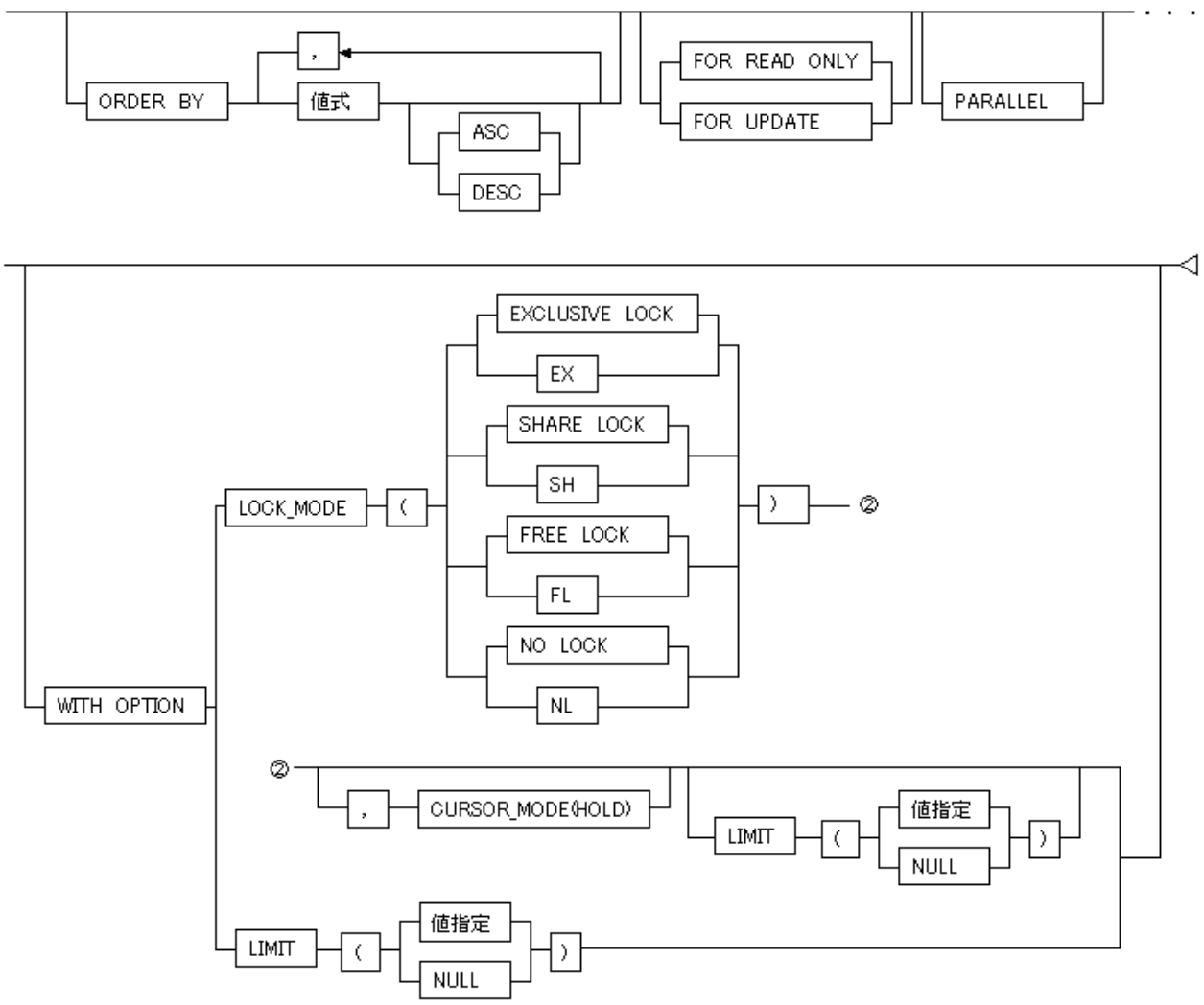
注4) カーソルCSR1により表T1を更新後、非カーソル系のSQL文で表T1を参照すると、更新前の状態で参照されることと、更新後の状態で参照されてしまうことがあります。このため非カーソル系のSQL文による表T1の参照は、カーソルCSR1をクローズしてから実行してください。

注5) カーソルCSR1により表T1を更新後、非カーソル系のSQL文で表T1を更新すると、次にカーソルCSR1のFETCH文で表T1の行を取り出すときに、データベースとカーソルの間で矛盾を起こすことがあります。このため非カーソル系のSQL文による表T1の更新は、カーソルCSR1をクローズしてから実行してください。

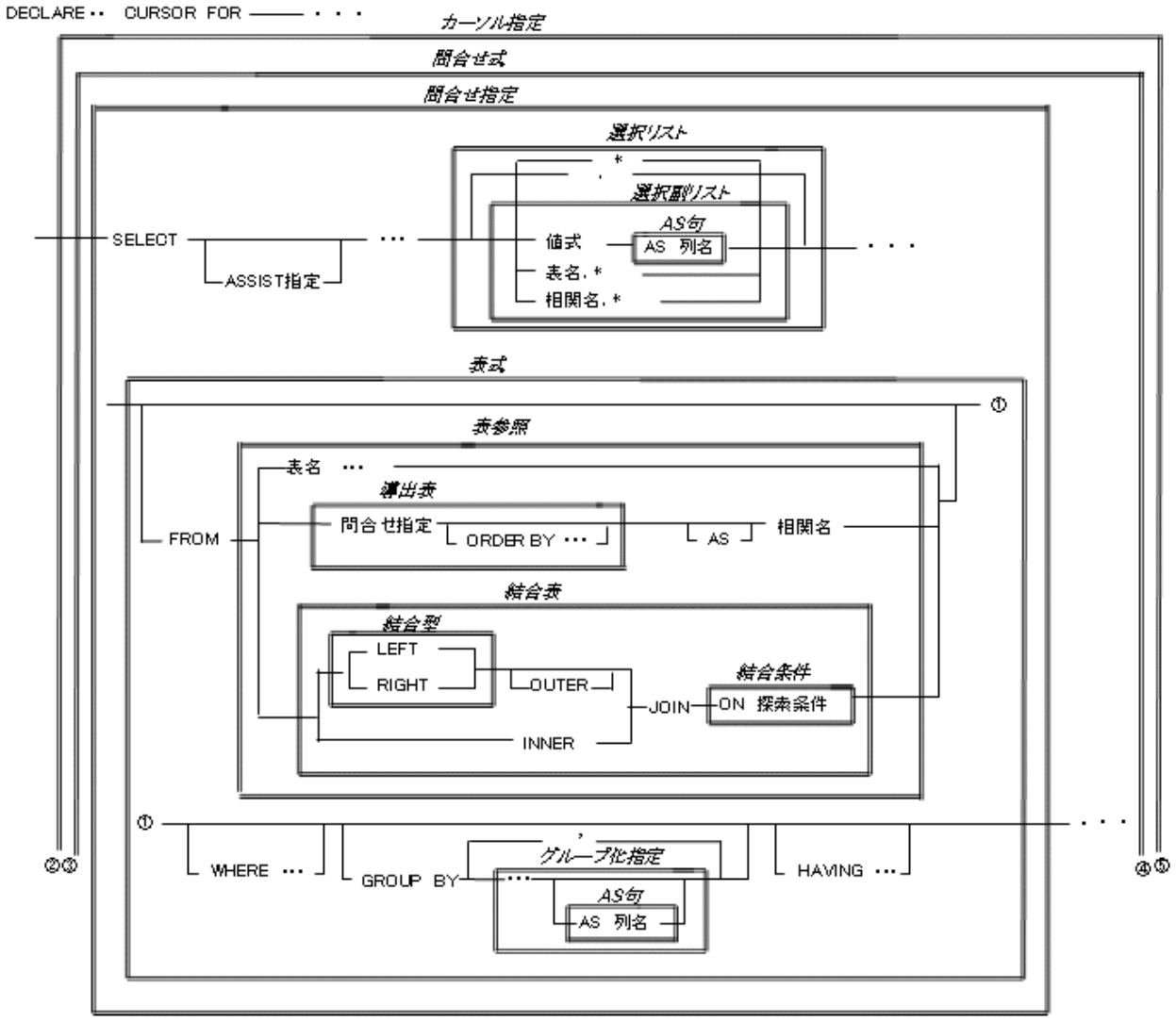
備考. カーソルCSR1とカーソルCSR2のカーソル宣言は、両方とも表T1を指定したとします。

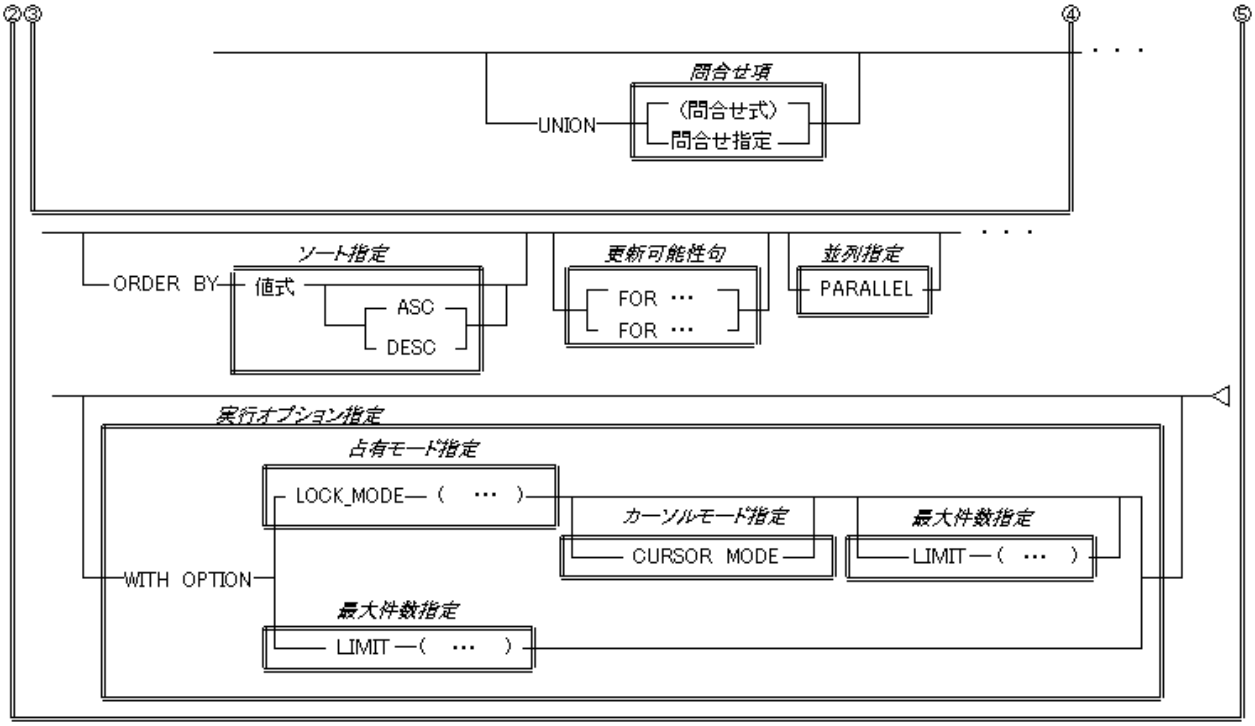
記述形式


DECLARE
カーソル名
SCROLL
CURSOR FOR
SELECT
ASSIST指定
ALL
DISTINCT
*
,
値式
AS
列名
表名.*
相関名.*
FROM
表名
AS
相関名
(
問合せ指定
)
①
ORDER BY
値式
ASC
DESC
導出列リスト
LEFT
RIGHT
OUTER
INNER
JOIN
表参照
②
ON
探索条件
WHERE
GROUP BY
列指定
数値関数
データ列値関数
日時値関数
CASE式
ファンクションルーチン指定
HAVING
UNION
ALL
問合せ式
問合せ指定


ORDER BY
,
値式
ASC
DESC
FOR READ ONLY
FOR UPDATE
PARALLEL
WITH OPTION
LOCK_MODE
(
EXCLUSIVE LOCK
EX
SHARE LOCK
SH
FREE LOCK
FL
NO LOCK
NL
)
②
②
,
CURSOR_MODE(HOLD)
LIMIT
(
値指定
NULL
)
LIMIT
(
値指定
NULL
)

構文の構成


DECLARE CURSOR FOR
カーソル指定
問合せ式
問合せ指定
選択リスト
*
選択副リスト
AS句
SELECT
ASSIST指定
値式
AS 列名
表名.*
相関名.*
表式
①
表参照
表名
導出表
問合せ指定
ORDER BY
AS
相関名
FROM
結合表
結合型
LEFT
RIGHT
OUTER
INNER
JOIN
結合条件
ON 探索条件
①
WHERE
GROUP BY
グループ化指定
AS句
AS 列名
HAVING
②③
④⑤



参照項番

- ASSIST指定 → “2.15 ASSIST指定”
- 列指定 → “2.4 列指定”
- データ列値関数 → “2.5.3 データ列値関数”
- 数値関数 → “2.5.2 数値関数”
- 日時値関数 → “2.5.4 日時値関数”
- 値式 → “2.11 値式”
- 探索条件 → “2.13 探索条件”
- CASE式 → “2.6 CASE式”
- ファンクションルーチン指定→ “2.5.5 ファンクションルーチン指定”
- 日本語文字列 → “2.1.3 トークン”

一般規則

カーソル名

- カーソル名は、カーソルの名前を指定します。
- カーソル名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

SCROLL

- FETCH文に取出し方向を指定する場合、カーソル宣言にSCROLLの指定が必要です。ただし、取出し方向にNEXTを指定した場合は、SCROLLの指定を省略することができます。
- 問合せ式で選択リストに、格納構造がOBJECTでBLOB型の列が指定された場合は、SCROLLは指定できません。
- プロシジャルーチン定義中にSCROLLは指定できません。

問合せ指定

- 問合せ指定は、表式の結果から導出される表を指定します。

ー 問合せ指定の結果を以下に示します。

- 表式の結果がグループ表でない場合、問合せ指定の結果は次のようになります。選択リストに集合関数指定がない場合、表式の結果の各行に選択リストの値式で示す演算が行われた、行の集合となります。選択リストに集合関数の指定がある場合の結果は、表式の結果を集合関数に適用した1行となります。
- 表式の結果がグループ表の場合、問合せ指定の結果は次のようになります。結果の行の数はグループの数となります。各行の値は、選択リストに集合関数の指定がある場合には、グループを集合関数に適用した結果となり、列指定の場合には、グループ化列の値となり、グループ化関数の場合は、グループ化関数の値となります。なお、選択リストにGROUP BY句のAS句の列名を指定した場合は、対応するグループ化列、グループ化関数またはCASE式の値となります。
- 表式の結果が空であれば、問合せ指定の結果は、空の表となります。

DISTINCTおよびALL

ー DISTINCTは、問合せ指定の結果の各行同士がまったく同じ値を持つ行がある場合、それらの行を1行とする処理が行われます。このとき、NULL値同士は等しいとみなされます。ALLは、重複行をそのまま残す場合に指定します。また、どらちも省略した場合は、ALLが指定されたものとみなします。

ー 問合せ式で選択リストに、BLOB型の列が指定された場合は、DISTINCTは指定できません。

ー 問合せ式で選択リストに、順序が指定された場合は、DISTINCTは指定できません。

選択リスト

ー 選択リストには、表式の結果から導出される表の列を指定します。

ー 選択リストに“*”を指定した場合、FROM句で指定した順に各表の各列のすべてを記述したのと同じ意味です。

ー 選択副リストは、選択リストにカンマ(,)で区切って指定する1つ1つの要素です。

ー 選択リストに“表名.*”または“相関名.*”が指定された場合、表のすべての列を記述したのと同じ意味です。“表名.”と“相関名.”は“修飾子”と呼びます。

ー 選択リスト中に含まれる列指定は、FROM句で指定した表の列またはGROUP BY句のAS句の列名であることが必要です。

ー 選択リストに格納構造がOBJECTのBLOB型の列を複数指定することはできません。

ー 表式の結果がグループ表の場合、選択リストの列指定は、グループ化列またはGROUP BY句のAS句の列名を指定するか、集合関数指定またはグループ化関数の引数で指定することが必要です。

ー 表式の結果がグループ表でなく、選択リストに集合関数を指定する場合は、選択リストのすべてが集合関数指定であることが必要です。

ー 選択リストにAS句が指定された場合、選択リストの結果の列名はAS句に指定された列名となります。

ー AS句の列名はORDER BY句のソート指定に列名として指定することができます。

FROM句

ー FROM句は、1つ以上の名前付きの表や問合せ指定の結果から導出される表(導出表)または結合表を指定します。FROM句に指定する表、導出表または結合表を表参照といいます。

ー FROM句の結果は、次のとおりです。

- FROM句で指定される表が1つならば、FROM句の結果は、その表となります。
- FROM句で指定される表が複数ならば、FROM句の結果は、それらの表の拡張直積となります。拡張直積は、指定された複数の表の列と行のすべてを組み合わせて導出します。拡張直積により導出される表の列数は、元となる表の列数の総和となります。列の順序は、指定された表の列の順序となります。また、表の行数は、元となる表の行数の積となります。

ー FROM句は省略することができます。省略した場合はFROM句にシステム表の RDBII_ASSISTTABLEが指定されたものとみなします。

例1

問合せ指定のFROM句に表TBL1と表TBL2の2つの表を指定した例です。

```
SELECT COL1,BCOL FROM TBL1,TBL2
```

FROM句の結果は、表TBL1と表TBL2の拡張直積となります。

表TBL1

COL1	COL2
10	A
15	B
20	C

表TBL2

ACOL	BCOL
A	1
A	2
B	2

⇨

FROM句の結果

COL1	COL2	ACOL	BCOL
10	A	A	1
10	A	A	2
10	A	B	2
15	B	A	1
15	B	A	2
15	B	B	2
20	C	A	1
20	C	A	2
20	C	B	2

⇦

問合せ指定の結果

COL1	BCOL
10	1
10	2
10	2
15	1
15	2
15	2
20	1
20	2
20	2

例2

問合せ指定のFROM句に結合表を指定した例です。結合する表をそれぞれT1、T2とし、探索条件を“T1.C1=T2.C4”とします。

T1

C1	C2	C3
1	A	あ
2	B	い
3	C	う
4	D	え
5	E	お

T2

C4	C5	C6	C7
2	W	か	さ
3	X	き	し
4	Y	く	す
7	Z	け	せ

```
SELECT C1, C4, C5 FROM
T1 LEFT OUTER JOIN T2 ON T1. C1 = T2. C4
```

C1	C4	C5
2	2	W
3	3	X
4	4	Y
1	－	－
5	－	－

```
SELECT C1, C4, C5 FROM
T1 RIGHT OUTER JOIN T2 ON T1. C1 = T2. C4
```

C1	C4	C5
2	2	W
3	3	X
4	4	Y
－	7	Z

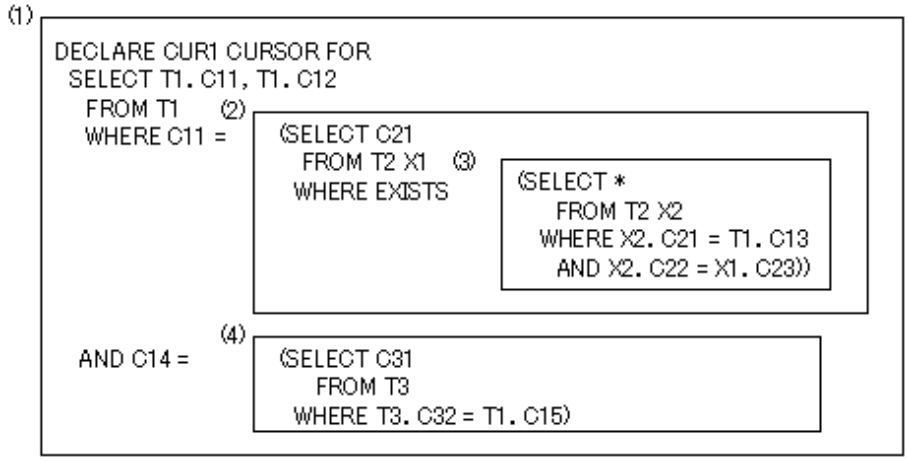
－：NULL値を表します。

表名

－ 検索対象とする表の名前を指定します。

相関名

- 相関名は、表の別名を指定します。
- 相関名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- 相関名が指定されていない表名は、ほかの相関名が指定されていない表名と同じであってはいけません。
- 相関名は、相関名指定のない表名と同じであってはいけません。また、ほかの相関名と同じであってはいけません。
- FROM句の中に指定される相関名と、相関名が指定されない表名の有効範囲は、FROM句が含まれる表式を含む最も内側の問合せ指定または副問合せになります。



(1) 表名T1の有効範囲

(2) 相関名X1の有効範囲

(3) 相関名X2の有効範囲

(4) 表名T3の有効範囲

導出表

- 導出表の問合せ指定が読込み専用ならば、導出表は読込み専用の表になります。問合せ指定が更新可能ならば、導出表は更新可能な表になります。問合せ指定が読込み専用または更新可能になる条件については、“表3.12 読込み専用カーソルとなる条件”を参照してください。
- 導出列リストは、導出表の列の別名を指定します。導出表の問合せ指定の選択リストに以下の条件を1つでも満たす値式を指定した場合、導出列リストは省略できません。
  - 列指定以外を指定した場合
  - 同じ列名を複数指定した場合
- 導出列リストは、列名をカンマで区切り括弧でくくって指定します。
- 導出表の問合せ指定の選択リストの個数と、導出列リストの列名の個数は同じであることが必要です。
- 導出列リストに記述できる列名の数は、最大32000個です。
- 導出表の選択リストに順序を指定することはできません。
- 導出表を記述したSQL文を実行した場合、Symfoware/RDBの最適化処理で入れ子をなくし平坦な形式のSQL文に変換する場合があります。このSQL文の変換では、変換前のSQL文の導出表の選択リストにCAST指定が指定されていると、CAST指定を条件式に移動することがあります。このとき、データベースにCAST指定で変換できないデータが含まれると、導出表の条件式において変換できないデータを除外していたとしても、条件式に移動したCAST指定において、CAST指定の変換エラーが発生する場合があります。そのため、データベースにCAST指定で変換できないデータが含まれる場合は、CASE式を用いて、変換できないデータを除外することにより、CAST指定で

変換エラーが発生しないように対処してください。
以下に例を示します。

実行するSQL文

```
SELECT T1_SUB2.CAST_TIMESTAMP
    FROM
     (SELECT CAST_TIMESTAMP
        FROM
         (SELECT CAST(CHAR_TIMESTAMP AS TIMESTAMP )
                    AS CAST_TIMESTAMP       ・・・・・(注1)
            FROM S1.T1
            WHERE  CHAR_TIMESTAMP <>'              '   ・・・・・(注2)
         ) AS T1_SUB (CAST_TIMESTAMP)
     ) T1_SUB2 (CAST_TIMESTAMP)
    WHERE T1_SUB2.CAST_TIMESTAMP > TIMESTAMP'2009-08-01 15:58:59'
```

注1) 選択リストにCAST指定を記述。

注2) 最も内側の導出表にて文字列の空白以外を取り出す判定。

最適化処理が変換したSQL文

Symfoware/RDBの最適化処理が変換したSQL文は、以下のようになります。

```
SELECT CAST(T1.CHAR_TIMESTAMP AS TIMESTAMP)
  FROM S1.T1
  WHERE T1.CHAR_TIMESTAMP <>'                    '  ・・・・・(注1)
    AND CAST(T1.CHAR_TIMESTAMP AS TIMESTAMP) > TIMESTAMP'2009-08-01 15:58:59' ・・・・・(注2)
```

注1) 文字列の空白以外を取り出す判定。

注2) 文字列をCASTし、変換結果を判定。

その結果、注2のCAST指定に指定された列の値が空白の場合、TIMESTAMP型にデータ型変換することができず、変換エラーになります。

対処例

CASE式を用いて、空白のデータを除外することにより、CASTで変換エラーが発生しないように対処してください。

```
SELECT T1_SUB2.CAST_TIMESTAMP
    FROM
      (SELECT CAST_TIMESTAMP
        FROM
          ( SELECT (CASE WHEN CHAR_TIMESTAMP <>'                    '
                      THEN
                       CAST(CHAR_TIMESTAMP AS TIMESTAMP )
                      ELSE
                       NULL
                     END) AS CAST_TIMESTAMP
            FROM S1.T1
            WHERE  CHAR_TIMESTAMP <>'                    '
          ) AS T1_SUB (CAST_TIMESTAMP)
      ) T1_SUB2 (CAST_TIMESTAMP)
    WHERE
        T1_SUB2.CAST_TIMESTAMP > TIMESTAMP'2009-08-01 15:58:59'
```

結合表(OUTER JOIN / INNER JOIN)

- 結合型は、結合表で結合の方法を示すものです。
- LEFTを指定した場合は、左側の表の列の値にない右側の表の列の値をNULL値に置き換えた行となります。
- RIGHTを指定した場合は、右側の表の列にない値と左側の表の列の値をNULL値に置き換えた行となります。
- INNERを指定した場合は、左側の表と右側の表の拡張直積となり、FROM句で指定される表が複数の場合と同じ結果になります。
- 左側の表または右側の表に相関名を指定する場合、結合表における相関名が指定されていない表名と同じであってはいけません。
- 探索条件中に直接含まれる列は、表参照で指定した列であることが必要です。
- 探索条件に副問合せは指定できません。
- 探索条件にROWNUMは指定できません。
- 結合表は、読込み専用の表であり、更新することはできません。

WHERE句

- WHERE句は、導出される表に対して探索条件を指定します。
- 探索条件に指定した列指定は、FROM句に指定した表の列を指定するか、または外への参照であることが必要です。

  外への参照は“2.4 列指定”を参照してください。
- WHERE句の結果は、FROM句の結果の表に対して、探索条件を適用した結果が真の行からなる表です。
- 探索条件に順序は指定できません。
- 探索条件に指定した値式が集合関数指定ならば、その探索条件は、HAVING句に指定された副問合せのWHERE句に指定したものであることが必要です。なお、集合関数指定の中の列指定は、外への参照であることが必要です。以下に例を示します。

```
SELECT C11,AVG(C12)
    FROM T1
    GROUP BY C11
    HAVING AVG(C12) < (SELECT COUNT(C21)
```

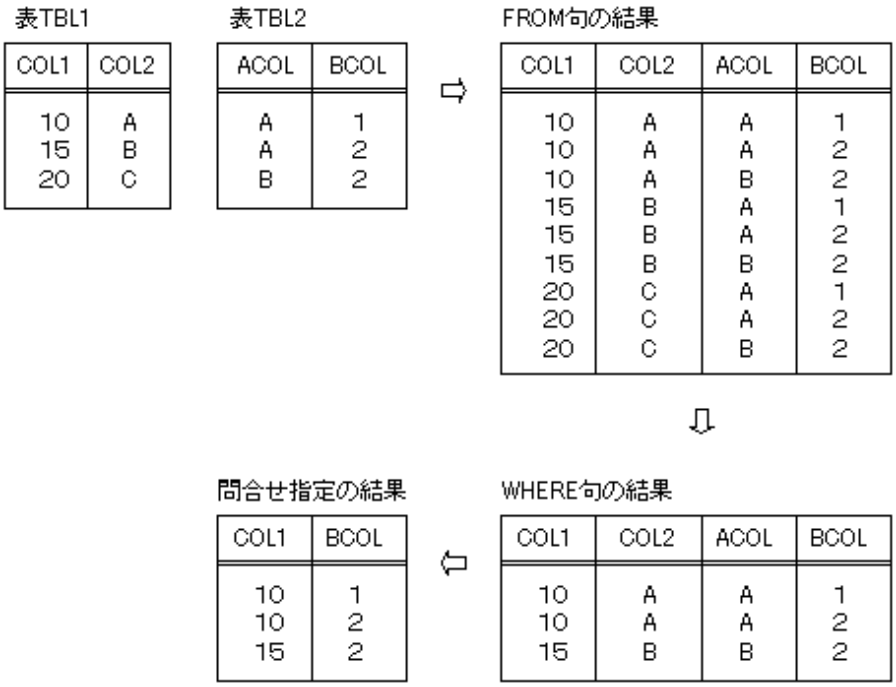
```
                    FROM T2
                    WHERE SUM(T1.C12) < T2.C22)
```

例

FROM句に指定された結果にWHERE句を適用することによって、必要な行だけを絞り込んだ表を導出します。

```
SELECT COL1,BCOL
    FROM TBL1,TBL2
    WHERE COL2 = ACOL
```

表TBL1と表TBL2の拡張直積からWHERE句の条件を満たす行が絞り込まれます。



GROUP BY句

- GROUP BY句は、導出された表に対してグループ化するための条件を指定(グループ化指定と呼びます)します。
- GROUP BY句に列指定を指定した場合、その列をグループ化列といいます。
- GROUP BY句にデータ列値関数、数値関数または日時値関数を指定した場合、その関数をグループ化関数といいます。
- GROUP BY句に指定する関数は、CURRENT DATE値関数、CURRENT TIME値関数、CURRENT TIMESTAMP値関数であってはいけません。
- GROUP BY句の結果は、指定したグループ化列、関数およびCASE式の値が同一の行の集まりを1つのグループとした、グループの集まりからなる表です。
- GROUP BY句に、グループ化関数を指定する場合、その関数中に指定する値式は、以下の条件を満たすことが必要です。
  - 関数内に指定する値式には、値指定、列指定、数値式、データ列値式、日時値式および時間隔値式を指定することができます。
  - 関数内に指定する値式には、集合関数、外への参照の列および動的パラメタ指定を指定することはできません。
  - 関数内に指定する値式には、列指定を必ず含むことが必要です。
- 列指定は、FROM句に指定した表の列であることが必要です。

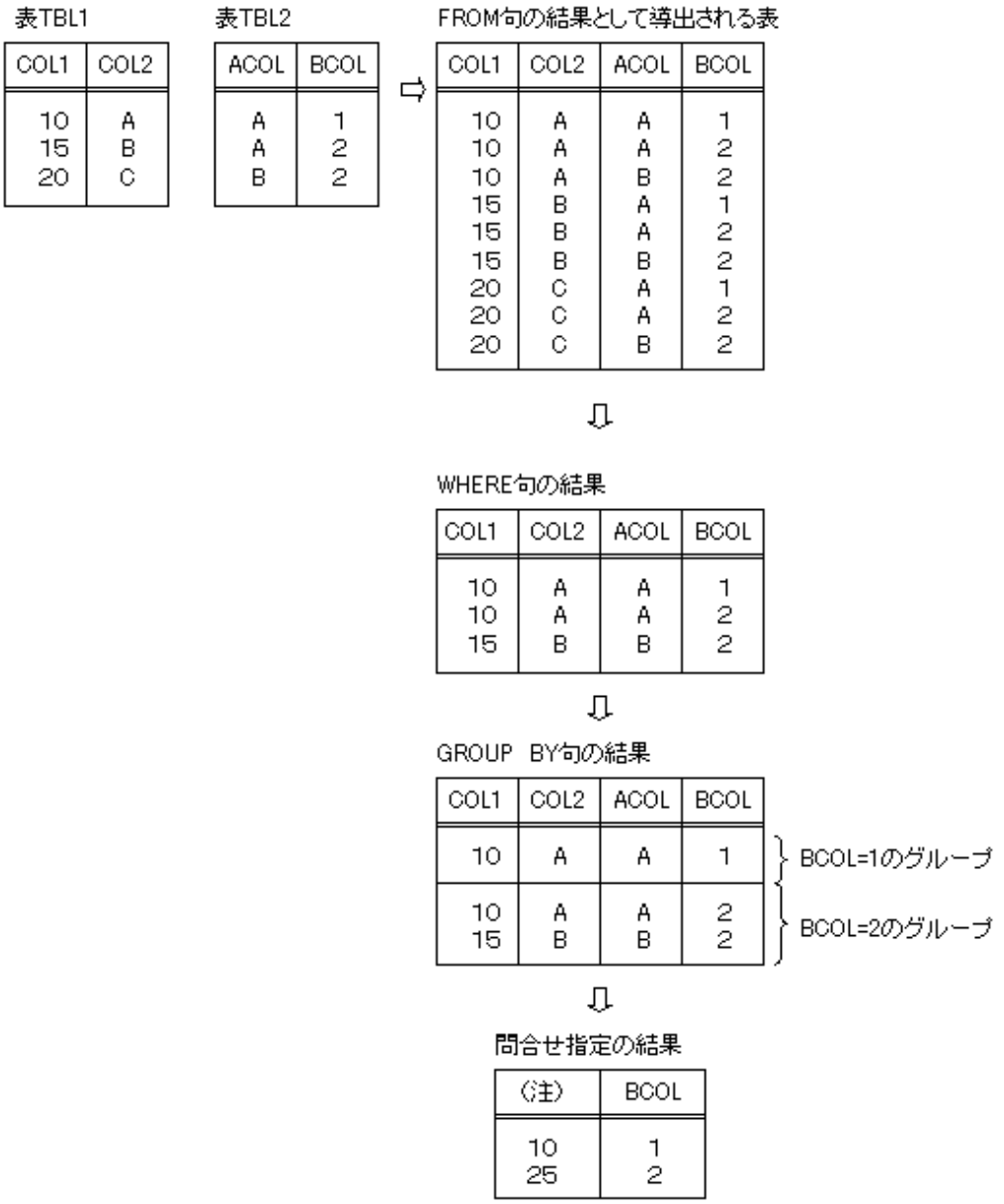
- GROUP BY句を指定した場合、HAVING句および問合せ指定の選択リストに指定する値式は、以下のいずれかでなければなりません。
  - 値指定
  - 集合関数
  - CURRENT DATE値関数、CURRENT TIME値関数、CURRENT TIMESTAMP値関数
  - グループ化列
  - AS句の列名
  - グループ化列またはAS句の列名を含む値式。ただし、値式に含まれるすべての列指定がグループ化列またはAS句の列名でなければなりません。
  - グループ化関数
- HAVING句および問合せ指定の選択リストにグループ化関数を指定する場合、以下の注意が必要です。
  - 値式の引数にグループ化関数を指定する場合、グループ化関数の引数に指定する列指定は、すべてグループ化列またはAS句の列名でなければなりません。また、副問合せ内で、グループ化関数の引数に外への参照の列指定を指定する場合、グループ化関数の引数に指定する列指定は、すべてグループ化列またはAS句の列名でなければなりません。
- 集合関数の引数にグループ化関数を指定することはできません。
- グループ化指定に、CASE式を指定する場合、そのCASE式中に指定する値式は以下の条件を満たすことが必要です。
  - CASE式内に指定する値式には、集合関数、外への参照の列および動的パラメタ指定を指定することはできません。
  - CASE式内に指定する値式には、列指定を必ず含むことが必要です。
- AS句の列名をHAVING句や問合せ指定の選択リストに指定した場合は、AS句の列名に対応する列指定、関数またはCASE式が指定されたものとみなされます。
- グループ化指定に指定したCASE式をそのままHAVING句や問合せ指定の選択リストに指定しても、グループ化したCASE式とはみなされません。CASE式を指定する場合は、AS句の列名を使用してください。
- AS句の列名は、FROM句の表の列名と同じであってはいけません。また、GROUP BY句において一意でなければなりません。
- AS句の列名を値式として指定する場合には以下の規約があります。
  - 外への参照でない場合、HAVING句、選択リスト、ORDER BY句以外に指定することはできません。
  - 外への参照の場合、結合表の結合条件、WHERE句、HAVING句以外に指定することはできません。また、AS句の列名を含む副問合せが、HAVING句に含まれていなければなりません。
- GROUP BY句にはBLOB型の列は指定できません。
- GROUP BY句にROWNUMは指定できません。

例

問合せ指定にFROM句、WHERE句およびGROUP BY句を適用した例を示します。

```
SELECT SUM(COL1),BCOL
    FROM TBL1,TBL2
    WHERE COL2 = ACOL
    GROUP BY BCOL
```

表TBL1と表TBL2の拡張直積からWHERE句の条件を満たす行が絞り込まれます。さらに、GROUP BY句で指定された列の値が同一の行を1つのグループとします。この結果、問合せ指定の結果が2件となります。



注) 集合関数の結果の列には名前がありません。

HAVING句

- HAVING句は、GROUP BY句またはFROM句の結果であるグループ表に対し、探索条件に合うグループだけを選びだす条件を指定します。
- 探索条件に含まれる列指定は、FROM句で指定した表の列またはGROUP BY句のAS句の列名であることが必要です。
- 探索条件に含まれる列指定は、グループ化列またはGROUP BY句のAS句の列名を指定するか、集合関数指定またはグループ化関数の引数で指定することが必要です。

ー 探索条件に含まれる列指定が外への参照の場合、副問合せが指定される句がWHERE句か、HAVING句かにより、指定のしかたが異なります。以下に外への参照の指定のしかたについて示します。外への参照は、“2.4 列指定”を参照してください。

- WHERE句に指定した副問合せの場合、副問合せのHAVING句の外への参照は直接の列指定であることが必要です。以下にWHERE句に指定した副問合せにHAVING句を指定した例を示します。

```
SELECT C11,C12
FROM T1
WHERE T1.C13 = (SELECT SUM(C22) FROM T2
                GROUP BY C21
                HAVING T2.C21 = T1.C14)
                                  (1)
```

(1) 列指定

- HAVING句に指定した副問合せの場合、副問合せのHAVING句に指定する外への参照は、グループ化列またはGROUP BY句のAS句の列名を直接指定するか、またはグループ化列またはGROUP BY句のAS句の列名以外を集合関数で指定する必要があります。

ー HAVING句の結果は、GROUP BY句の結果のグループ表に対して、探索条件を適用した結果が真の行からなるグループ表です。以下にHAVING句に指定した副問合せにHAVING句を指定した例を示します。

```
SELECT C11,AVG(C12)
FROM T1
GROUP BY C11
HAVING AVG(C13) = (SELECT SUM(C22) FROM T2
                   GROUP BY C21
                   HAVING T2.C21 = T1.C11 AND AVG(T2.C22) = AVG(T1.C12))
                            (1)                   (2)
```

(1) 列指定

(2) 集合関数指定

ー 探索条件にROWNUMは指定できません。

例1

HAVING句にグループ化列と集合関数を指定した例です。

```
SELECT COL2,SUM(COL3)
FROM T1
GROUP BY COL2
HAVING COL2 = N'冷蔵庫' AND AVG(COL3) < 40
       (1)                    (2)
```

(1) グループ化列

(2) 集合関数指定

在庫表

COL1	COL2	COL3	COL4
110	テレビ	85	2
111	テレビ	90	2
123	冷蔵庫	60	1
124	冷蔵庫	75	1
212	テレビ	0	2
215	ビデオ	5	2
226	冷蔵庫	8	1
227	冷蔵庫	15	1

⇩

GROUP BY句で導出される表

COL1	COL2	COL3	COL4	
110	テレビ	85	2	1グループ
111	テレビ	90	2	
212	テレビ	0	2	
215	ビデオ	5	2	1グループ
123	冷蔵庫	60	1	1グループ
124	冷蔵庫	75	1	
226	冷蔵庫	8	1	
227	冷蔵庫	15	1	

⇩

HAVING句の結果(問合せ指定の結果)

COL2	(注)
冷蔵庫	158

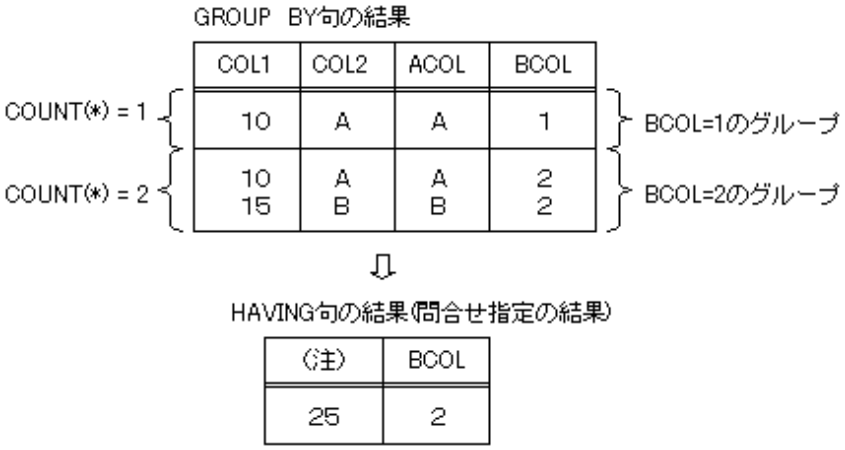
注) 集合関数の結果の列には名前がありません。

例2

“GROUP BY句”で示した例にHAVING句を追加した例を示します。

```
SELECT SUM(COL1),BCOL
FROM TBL1,TBL2
WHERE COL2 = ACOL
GROUP BY BCOL
HAVING COUNT(*) > 1
```

表TBL1と表TBL2の拡張直積からWHERE句の条件を満たす行が絞り込まれます。そして、GROUP BY句で指定された列の値が同一の行を1つのグループとします。さらに、HAVING句でグループが限定されます。この結果、問合せ指定の結果は1件となります。



注) 集合関数の結果の列には名前がありません。

問合せ式(UNION)

- UNIONを指定すると、問合せ式の結果は、左辺と右辺の表の集合演算を行った表となります。UNIONを“集合演算子”と呼びます。集合演算の結果は次のようになります。
  - ALLを指定すると、左辺と右辺の表の集合和となります。
  - ALLを指定しないと、ALLを指定した結果の各行同士に、まったく同じ値を持つ行がある場合、それらの複数行を1行とする処理が行われます。このとき、NULL値同士は等しいとみなされます。
- 問合せ項は、問合せ式を構成する1つの要素です。
- 集合演算子は、括弧を優先し、左から右に適用されます。
- 問合せ式にUNIONを含む場合、集合演算の結果の表は一番左側の問合せ指定の列名になります。
- 集合演算子UNIONを指定する場合、集合演算の対象となるそれぞれの問合せ指定は、以下の条件を満たすことが必要です。
  - 選択リストの結果の個数は同じであることが必要です。
  - 選択リストの結果の集合演算で対応する列同士は、比較可能であることが必要です。比較可能なデータ型に関する規則は、“表2.61 比較可能なデータ型”を参照してください。
- 最終結果に採用されるデータ型の一覧を“表3.9 集合演算(UNION)におけるデータ変換”、“表3.10 時間隔型(YEAR,MONTH) の集合演算(UNION)におけるデータ変換”および“表3.11 時間隔型(DAY,HOUR,MINUTE,SECOND)の集合演算(UNION)におけるデータ変換”に示します。

表3.9 集合演算(UNION)におけるデータ変換

集合演算対象のデータ型		集合演算結果のデータ型
1項	2項	
SMALLINT	SMALLINT	SMALLINT(無変換)
	INTEGER	INTEGER
	DECIMAL(p,q)	DECIMAL(r,s) r: MIN(18,MAX(p-q,5)+q) s: q
	NUMERIC(p,q)	
	REAL	REAL
	DOUBLE PRECISION	DOUBLE PRECISION
INTEGER	SMALLINT	INTEGER
	INTEGER	INTEGER(無変換)

集合演算対象のデータ型		集合演算結果のデータ型
1項	2項	
	DECIMAL(p,q)	DECIMAL(r,s) r: MIN(18,MAX(p-q,10)+q) s: q
	NUMERIC(p,q)	
	REAL	REAL
	DOUBLE PRECISION	DOUBLE PRECISION
DECIMAL(p1,q1)	SMALLINT	DECIMAL(r,s) r: MIN(18,MAX(p1-q1,5)+q1) s: q1
	INTEGER	DECIMAL(r,s) r: MIN(18,MAX(p1-q1,10)+q1) s: q1
	DECIMAL(p2,q2)	DECIMAL(r,s) r: MIN(18,MAX(p1-q1,p2-q2)+MAX(q1,q2)) s: MAX(q1,q2)
	NUMERIC(p2,q2)	
	REAL	REAL
	DOUBLE PRECISION	DOUBLE PRECISION
NUMERIC(p1,q1)	SMALLINT	DECIMAL(r,s) r: MIN(18,MAX(p1-q1,5)+q1) s: q1
	INTEGER	DECIMAL(r,s) r: MIN(18,MAX(p1-q1,10)+q1) s: q1
	DECIMAL(p2,q2)	DECIMAL(r,s) r: MIN(18,MAX(p1-q1,p2-q2)+MAX(q1,q2)) s: MAX(q1,q2)
	NUMERIC(p2,q2)	
	REAL	REAL
	DOUBLE PRECISION	DOUBLE PRECISION
REAL	SMALLINT	REAL
	INTEGER	REAL
	DECIMAL(p,q)	REAL
	NUMERIC(p,q)	REAL
	REAL	REAL(無変換)
	DOUBLE PRECISION	DOUBLE PRECISION
DOUBLE PRECISION	SMALLINT	DOUBLE PRECISION
	INTEGER	DOUBLE PRECISION
	DECIMAL(p,q)	DOUBLE PRECISION
	NUMERIC(p,q)	DOUBLE PRECISION
	REAL	DOUBLE PRECISION

集合演算対象のデータ型		集合演算結果のデータ型
1項	2項	
	DOUBLE PRECISION	DOUBLE PRECISION(無変換)
CHAR(n1)	CHAR(n2)	CHAR(m) m: MAX(n1,n2)
	VARCHAR(n2)	VARCHAR(m) m: MAX(n1,n2)
VARCHAR(n1)	CHAR(n2)	VARCHAR(m) m: MAX(n1,n2)
	VARCHAR(n2)	VARCHAR(m) m: MAX(n1,n2)
NCHAR(n1)	NCHAR(n2)	NCHAR(m) m: MAX(n1,n2)
	NCHAR VARYING(n2)	NCHAR VARYING(m) m: MAX(n1,n2)
NCHAR VARYING(n1)	NCHAR(n2)	NCHAR VARYING(m) m: MAX(n1,n2)
	NCHAR VARYING(n2)	NCHAR VARYING(m) m: MAX(n1,n2)
DATE	DATE	DATE
TIME	TIME	TIME
TIMESTAMP	TIMESTAMP	TIMESTAMP
INTERVAL	INTERVAL	INTERVAL (注)

注) 集合演算結果のデータ型の詳細については、表3.10 時間隔型(YEAR,MONTH)の集合演算(UNION)におけるデータ変換および表3.11 時間隔型(DAY,HOUR,MINUTE,SECOND)の集合演算(UNION)におけるデータ変換を参照してください。

表3.10 時間隔型(YEAR,MONTH)の集合演算(UNION)におけるデータ変換

		2項		
		YEAR(p2)	YEAR(p2) TO MONTH	MONTH(p2)
1項	YEAR(p1)	YEAR(p) p: MAX(p1,p2)	YEAR(p) TO MONTH p: MAX(p1,p2)	YEAR(p) TO MONTH p: MIN(9, MAX(p1,p2-1))
	YEAR(p1) TO MONTH	YEAR(p) TO MONTH p: MAX(p1,p2)	YEAR(p) TO MONTH p: MAX(p1,p2)	YEAR(p) TO MONTH p: MIN(9, MAX(p1,p2-1))
	MONTH(p1)	YEAR(p) TO MONTH p: MIN(9, MAX(p1-1,p2))	YEAR(p) TO MONTH p: MIN(9, MAX(p1-1,p2))	MONTH(p) p: MAX(p1,p2)

p、p1、p2: 時間隔先行フィールド精度

表3.11 時間隔型(DAY,HOUR,MINUTE,SECOND)の集合演算(UNION)におけるデータ変換

		2項			
		DAY(p2) TO E2	HOUR(p2) TO E2	MINUTE(p2) TO E2	SECOND(p2)
1項	DAY(p1) TO E1	DAY(p) TO E p: MAX(p1,p2) E: MIN(E1,E2)	DAY(p) TO E p: MIN(9, MAX(p1,p2-1)) E: MIN(E1,E2)	DAY(p) TO E p: MIN(9, MAX(p1,p2-3)) E: MIN(E1,E2)	DAY(p) TO SECOND p: MIN(9, MAX(p1,p2-5))
	HOUR(p1) TO E1	DAY(p) TO E p: MIN(9, MAX(p1-1,p2)) E: MIN(E1,E2)	HOUR(p) TO E p: MAX(p1,p2) E: MIN(E1,E2)	HOUR(p) TO E p: MIN(9, MAX(p1,p2-1)) E: MIN(E1,E2)	HOUR(p) TO SECOND p: MIN(9, MAX(p1,p2-3))
	MINUTE(p1) TO E1	DAY(p) TO E p: MIN(9, MAX(p1-3,p2)) E: MIN(E1,E2)	HOUR(p) TO E p: MIN(9, MAX(p1-1,p2)) E: MIN(E1,E2)	MINUTE(p) TO E p: MAX(p1,p2) E: MIN(E1,E2)	MINUTE(p) TO SECOND p: MIN(9, MAX(p1,p2-1))
	SECOND(p1)	DAY(p) TO SECOND p: MIN(9, MAX(p1-5,p2))	HOUR(p) TO SECOND p: MIN(9, MAX(p1-3,p2))	MINUTE(p) TO SECOND p: MIN(9, MAX(p1-1,p2))	SECOND(p) p: MAX(p1,p2)

p、p1、p2: 時間隔先行フィールド

E、E1、E2: 日時フィールド

なお、日時フィールドには以下の大小関係があります。

DAY > HOUR > MINUTE > SECOND

例

```
MIN(DAY,MINUTE) = MINUTE
```

カーソル指定

- カーソル指定で導出される表は、カーソルの定義方法によって、更新可能または読込み専用となります。カーソル指定で導出される表の扱いは次のとおりです。
  - カーソル指定の問合せ式が読込み専用の場合、そのカーソル指定で導出される表は読込み専用となります。
  - 上記以外の場合、そのカーソル指定で導出される表は更新可能となります。
- カーソル宣言の形式が、“表3.12 読込み専用カーソルとなる条件”のいずれかに該当する場合、読込み専用カーソルになります。いずれにも当てはまらない場合は、更新可能カーソルになります。

  “表3.12 読込み専用カーソルとなる条件”のa)からt)は、問合せ指定で指定する要素です。また、u)は問合せ式で指定する要素です。v)からx)は、カーソル宣言で指定する要素です。w)とx)は、更新可能性句にFOR UPDATEを指定したときのみ更新可能カーソルとなります。

表3.12 読込み専用カーソルとなる条件

読込み専用カーソルとなるカーソル宣言の条件	例
a) DISTINCTを指定している。	DECLARE CSR CURSOR FOR SELECT DISTINCT COLA, COLB FROM SCM.TBL1
b) 選択リストに演算式を指定している。	DECLARE CSR CURSOR FOR SELECT COLC,COLA + COLB FROM SCM.TBL1

読込み専用カーソルとなるカーソル宣言の条件	例
c) 選択リストに集合関数を指定している。	DECLARE CSR CURSOR FOR SELECT SUM(COLB) FROM SCM.TBL1
d) 選択リストに同じ列を指定している。	DECLARE CSR CURSOR FOR SELECT COLA,COLB,COLA,COLC FROM SCM.TBL1
e) 選択リストに定数またはホスト変数を指定している。	DECLARE CSR CURSOR FOR SELECT COLA,100 FROM SCM.TBL1
f) 選択リストにデータ列関数を指定している。	DECLARE CSR CURSOR FOR SELECT SUBSTRING(COL1 FROM 1 FOR 3) FROM SCM.TBL1
g) 選択リストに数値関数を指定している。	DECLARE CSR CURSOR FOR SELECT ABS(COL1) FROM SCM.TBL1
h) 選択リストに日時値関数を指定している。	DECLARE CSR CURSOR FOR SELECT TRUNC_DATE(COL1,'MONTH') FROM SCM.TBL1
i) 選択リストにXMLQUERY関数を指定している	DECLARE CSR CURSOR FOR SELECT XMLQUERY('/a[b/text() = 1]' PASSING COLX) FROM SCM.TBL1
j) 選択リストにCASE式を指定している。	DECLARE CSR CURSOR FOR SELECT CASE WHEN COL1=1 THEN 'ONE' END FROM SCM.TBL1
k) 選択リストにCAST指定を指定している。	DECLARE CSR CURSOR FOR SELECT CAST(COL1 AS CHAR(6)) FROM SCM.TBL1
l) 選択リストに連結演算子を指定している。	DECLARE CSR CURSOR FOR SELECT COL1 || COL2 FROM SCM.TBL1
m) 選択リストにファンクションルーチンを指定している。	DECLARE CSR CURSOR FOR SELECT USERFUNC1(C1,C2) FROM SCM.TBL1
n) 選択リストに順序を指定している。	DECLARE CSR CURSOR FOR SELECT SCM.順序1.NEXTVAL FROM SCM.TBL1
o) 選択リストにROWNUMを指定している。	DECLARE CSR CURSOR FOR SELECT ROWNUM, COLA FROM SCM.TBL1
p) FROM句に2つ以上の表または結合表を指定している。	DECLARE CSR CURSOR FOR SELECT COLA, COLZ FROM SCM.TBL1, SCM.TBL2
q) FROM句に、ORDER BY句を指定した導出表を指定している。	DECLARE CSR CURSOR FOR SELECT DCOLA FROM (SELECT COLA FROM SCM.TBL1 ORDER BY COLA) AS D1(DCOLA)
r) FROM句にROWNUMを指定した導出表を指定している。	DECLARE CSR CURSOR FOR SELECT DCOLA FROM (SELECT COLA FROM SCM.TBL1 WHERE ROWNUM < 10) AS D1(DCOLA)

読込み専用カーソルとなるカーソル宣言の条件	例
s) GROUP BY句またはHAVING句を指定している。	DECLARE CSR CURSOR FOR SELECT COLA FROM SCM.TBL1 GROUP BY COLA
t) WHERE句に副問合せを含んでいる。	DECLARE CSR CURSOR FOR SELECT * FROM SCM.TBL1 WHERE COLA IN (SELECT COLZ FROM SCM.TBL2 WHERE COLY > 2000)
u) UNIONを指定している。	DECLARE CSR CURSOR FOR SELECT COLC FROM SCM.TBL1 UNION SELECT COLX FROM SCM.TBL2
v) 更新可能性句にFOR READ ONLYを指定している。	DECLARE CSR CURSOR FOR SELECT COLA, COLB FROM SCM.TBL1 FOR READ ONLY
w) ORDER BY句を指定し、更新可能性句を指定していない。	DECLARE CSR CURSOR FOR SELECT COLA, COLB FROM SCM.TBL1 ORDER BY COLA
x) SCROLLを指定し、更新可能性句を指定していない。	DECLARE CSR SCROLL CURSOR FOR SELECT COLA, COLB FROM SCM.TBL1

ORDER BY句

- ORDER BY句を指定すると、カーソルの行の順序は、ORDER BY句のソート指定の順番になります。
- ORDER BY句を省略すると、カーソルの行の順序に規則性はありません。
- ORDER BY句が指定された場合、ソート指定の指定方法は次のとおりです。
  - ソート指定には値式を指定します。値式に指定可能な項目を以下に示します。

表3.13 ソート指定の値式に指定可能な項目

項目			指定可否
列指定			○
符号なし整数			○
値指定			
	変数指定		×
	定数		×
	USER		×
	動的パラメタ指定		×
	項目参照		
		パラメタ名	×
		SQL変数名	×
集合関数指定			
	COUNT(*)		○
	DISTINCT集合関数		○
	ALL集合関数		○
数値式			○
数値関数			○

項目	指定可否
データ列値式	○
データ列値関数	○
日時値式	○
日時値関数	○ (注)
ファンクションルーチン指定	○
時間隔値式	○
CASE式	○
CAST指定	○
ROW_ID	×
順序	×
XMLQUERY関数	×
ROWNUM	×

○: 指定可 ×: 指定不可

注) CURRENT_DATE値関数、CURRENT_TIME値関数およびCURRENT_TIMESTAMP値関数は除く。

- ソート指定に符号なし整数を指定すると、選択リストの左からの順番に対応する値式を指定したのと同じ意味です。
- ソート指定に指定した値式が符号なし整数でない場合、列指定を含むことが必要です。
- ソート指定に指定した値式が選択リストに指定されていない場合、問合せ指定にDISTINCTが指定されていてはなりません。
- ソート指定に指定した値式が選択リストに指定されていない場合でグループ表の場合、グループ化列、GROUP BY句のAS句の列名、グループ化列またはGROUP BY句のAS句の列名を含む値式、グループ化関数あるいは集合関数指定であることが必要です。
- 導出表にORDER BY句を指定する場合、ソート指定には、選択リストに指定していない値式を指定することはできません。
- 導出表にORDER BY句を指定する場合、ソート指定には、列指定または符号なし整数を指定しなければなりません。
- カーソル指定がUNIONを含み、ソート指定を符号なし整数以外で指定する場合は、一番左側の問合せ指定の選択リストが対象となります。この場合、選択リストに指定していない値式を指定することはできません。
- ソート指定に同じ値式を複数指定することはできません。
- ソート指定に指定した値式が選択リストに複数指定されていてはなりません。

― ORDER BY句が指定された場合の扱いは次のとおりです。

- ASCが指定された場合の行の順序は昇順となります。DESCが指定された場合は、降順となります。ASCもDESCも指定されない場合は、ASCが指定されたとみなされます。
- ソートの昇順および降順の順序づけは、“2.12.1 比較述語”の比較の方法に従います。ただし、NULL値は、最大の値として処理します。そのため、昇順では末尾に、降順では先頭に扱われます。
- ソートの優先順位は、指定された順になります。
- ソート対象となる列データが同値である行の順序に規則性はありません。

― ORDER BY句にBLOB型の値を指定することはできません。

― ORDER BY句に順序は指定できません。また選択リストに指定した順序を符号なし整数で指定することもできません。

― ORDER BY句にROWNUMは指定できません。

― 問合せ指定の選択リストに“*”を指定した場合、ソート指定には列指定または符号なし整数でなければなりません。

FOR READ ONLYおよびFOR UPDATE(更新可能性句)

- FOR READ ONLYおよびFOR UPDATEは、“更新可能性句”と呼びます。
- 更新可能性句にFOR READ ONLYが指定された場合、そのカーソルは読込み専用カーソルとなります。
- 更新可能性句にFOR UPDATEが指定された場合、そのカーソルは更新可能カーソルとなります。
- 更新可能性句が省略された場合の扱いは次のとおりです。
  - ORDER BY句を指定している、SCROLLを指定している、または問合せ式が読込み専用となる場合は、そのカーソルは読込み専用カーソルとなります。
  - 上記以外の場合、そのカーソルは更新可能カーソルとなります。
- カーソルの問合せ式が読込み専用の場合、FOR UPDATEは指定できません。
- カーソルの問合せ式が更新可能である場合、ORDER BY句を指定する、またはSCROLLを指定するカーソルは、更新可能性句にFOR UPDATEを指定したときのみ更新可能カーソルとなります。
- 更新可能性句にFOR READ ONLYを指定し、SET TRANSACTION文にREAD COMMITTEDが指定された場合、トランザクションを超越するカーソルとなります。ただし、トランザクションを超越するカーソルとして、32K以上のBLOB型は指定できません。

PARALLEL

- PARALLELは、“並列指定”と呼びます。
- 並列指定を指定した場合、表を並列に検索します。

実行オプション指定

- 実行オプション指定は、占有モード、カーソルモードまたは最大件数を指定します。

占有モード指定

- 占有モード指定は、問合せ指定を指定した場合に、問合せ指定のデータベース資源の占有の方法を指定します。
- 占有モードを指定したSQL文が読み込んだデータベース資源は、占有モード指定により、以下のようにデータベース資源を占有します。なお、更新する行は占有モード指定にかかわらず、非共有モードでトランザクションの終了までデータベース資源を占有します。

占有モード指定	占有の方法
EXCLUSIVE LOCK	非共有モードでトランザクション終了までデータベース資源を占有します。
SHARE LOCK	共有モードでトランザクション終了までデータベース資源を占有します。
FREE LOCK	共有モードでSQL文終了までデータベース資源を占有します。ただし、DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表にアクセスする場合には、データベース資源を占有しません。
NO LOCK	データベース資源を占有しません。

- 占有モードを指定したSQL文によって読み込まれた資源は、SET TRANSACTION文で指定された内容にかかわらず、以下のようになります。

占有モード指定	読込み水準
EXCLUSIVE LOCK	他のトランザクションによって占有されていない行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
SHARE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。当該

占有モード指定	読込み水準
	SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
FREE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。ただし、DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表にアクセスする場合には、他のトランザクションでの占有の有無に関わらず、SQL文実行時にコミット済みの行のデータを読み込みます。いずれの場合も、当該SQL文で読み込んだ行は、他のトランザクションに更新されることがあるため、同一トランザクションで再検索すると最新の結果を検索することができます。
NO LOCK	他のトランザクションでどのような占有をされた資源でも参照することが可能なため、同一トランザクションで再検索すると最新の結果を検索することができます。

- 占有モードの単位は、R_LOCKがYESの場合は行単位となります。R_LOCKがNOの場合は、ページ単位またはDSI単位の占有になります。
- 動作環境パラメタにDSO_LOCKを指定したり、環境変数でRDBDSOを指定した場合、占有モードを指定したSQL文は実行できません。

カーソルモード指定

- カーソルモードを指定することにより、占有モードを指定したカーソルをトランザクションを越えて保持することができます。カーソルモードを指定したカーソルは、トランザクションをコミットしてもトランザクションを越えてオープン状態を保持することができます。ただし、ROLLBACK文でトランザクションを終了させた場合や、トランザクションを取り消すようなエラーが発生した場合は、カーソルは自動的にクローズされます。
- カーソルモード指定は、占有モード指定でFREE LOCKまたはNO LOCKの場合に指定できます。
- 更新可能性句にFOR UPDATEを指定できません。
- 32K以上のBLOB型の列を指定できません。

最大件数指定

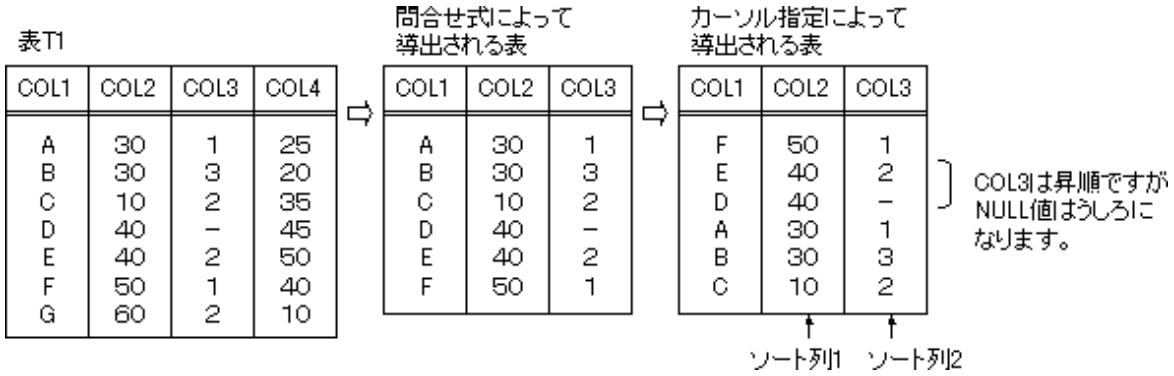
- 最大件数指定は、検索結果として取り出す件数を限定します。最大件数指定を指定すると、それ以上の行は返されません。
- 最大件数指定に指定された値が0以下の場合、検索結果として取り出す件数は0件となります。
- 最大件数指定に指定された値がNULLの場合、検索結果として取り出す件数は限定されず、すべての行が返されます。
- ORDER BY句が指定されている場合、ソート指定に指定された順序に並び替えられた後、その先頭から、最大件数指定で指定された件数までが結果として取り出されます。ORDER BY句を指定しない場合は任意の順序で取り出されます。
- 値指定にUSERは指定できません。
- 指定値に指定できる値は、-2147483648から2147483647までの整数値です。
- 値指定に変数指定または項目参照を指定した場合、データ型はINTEGER型またはSMALLINT型が指定可能です。
- 値指定に動的パラメタ指定が指定された場合のDESCRIBE情報は、INTEGER型になります。

使用例

例1

カーソル宣言にORDER BY句を指定した例を示します。

```
DECLARE CSR1 CURSOR FOR
        SELECT COL1,COL2,COL3 FROM T1 WHERE COL4 > 10
        ORDER BY COL2 DESC,COL3 ASC
```



例2

スキーマ“S1”の表“T2”を検索します。読込みの水準として、他のトランザクションが更新中のデータを読むことができます。また、他のトランザクションの更新を止めず、排他を獲得せずに読み込むことができます。

```
DECLARE CUR1 CURSOR FOR
        SELECT C1,C2 FROM S1.T2
        WITH OPTION LOCK_MODE(NO LOCK)
```

例3

占有モード指定のカーソルを、トランザクションを越えて保持します。

スキーマ“S1”の表“T3”を検索します。検索を行うカーソルをオープンしたまま、トランザクションをコミットしてもカーソルは閉じないようにします。

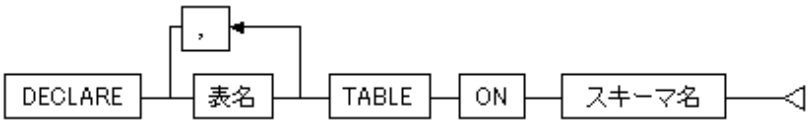
```
EXEC SQL DECLARE CUR1 CURSOR FOR
    SELECT C1,C2  FROM S1.T3  WHERE C3 > DATE'2007-04-01'
    WITH OPTION LOCK_MODE(FREE LOCK) , CURSOR_MODE(HOLD);
      ～
EXEC SQL OPEN CUR1;
EXEC SQL FETCH CUR1 INTO :OUTVAL1, :OUTVAL2;
      ～
EXEC SQL COMMIT WORK;  →COMMIT文を実行してもカーソルはクローズされません。
      ～
EXEC SQL FETCH CUR1 INTO :OUTVAL1, :OUTVAL2;
      ～  →データの終了
EXEC SQL CLOSE CUR1;
```

3.27 DECLARE TABLE(表宣言)

機能

表宣言は、スキーマ名の修飾をしない表名を定義します。表宣言で定義した表名を、それ以降のSQL文で記述する場合、スキーマ名を省略することができます。

記述形式



一般規則

- 同一の表名に対する表宣言は、SQL埋込みプログラム中に複数記述できません。
- 表宣言は、その表宣言で宣言する修飾なしの表名を指定するSQL文より先に指定することが必要です。

表名

- スキーマ名を省略して使用する表の名前を指定します。
- 表名はスキーマ名で修飾することはできません。

スキーマ名

- 表を含むスキーマの名前を指定します。

使用例

例

SELECT文の表名“在庫表”のスキーマ名の修飾を省略できるようにするため、表名“在庫表”を表宣言します。

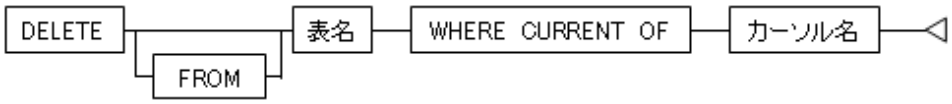
```
DECLARE 在庫表 TABLE ON SCH1
SELECT  在庫数量 INTO :STOCK FROM 在庫表 WHERE 製品番号 = 200
```

3.28 DELETE文:位置づけ

機能

カーソルによって位置づけられた1行を削除します。

記述形式



権限

- DELETE文:位置づけを実行できるのは、処理対象の表のDELETE権の保持者です。

一般規則

- カーソルにより位置づけられている行が削除されます。
- カーソルは、開かれた状態であり、FETCH文により位置づけられていることが必要です。

表名

- 行を削除する表の名前を指定します。
- 表名で指定する表は、カーソル宣言のカーソル指定に指定されていることが必要です。
- カーソル宣言のカーソル指定に導出表が記述されている場合、表名には導出表の元となる表を指定します。

カーソル名

- カーソルの名前を指定します。
- カーソル名は、同一コンパイル単位に含まれるカーソル宣言で定義されていることが必要です。

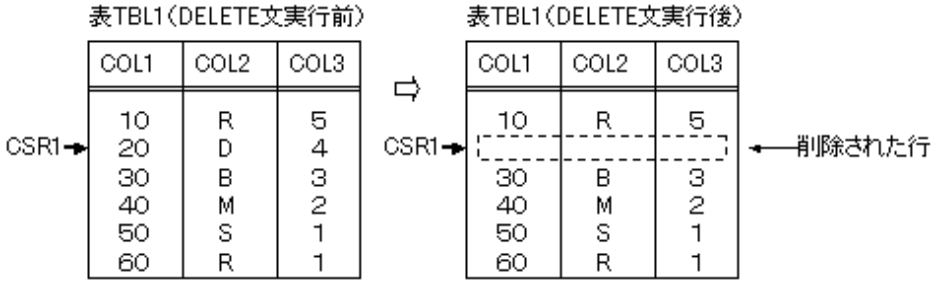
使用例

例

表TBL1からカーソルが位置づけられている行を削除する例を示します。

```
DELETE FROM TBL1
       WHERE CURRENT OF CSR1
```

CSR1は、このDELETE文と同じ同一コンパイル単位内で定義されたカーソルとします。現在、カーソルCSR1が位置づけられている行を削除します。

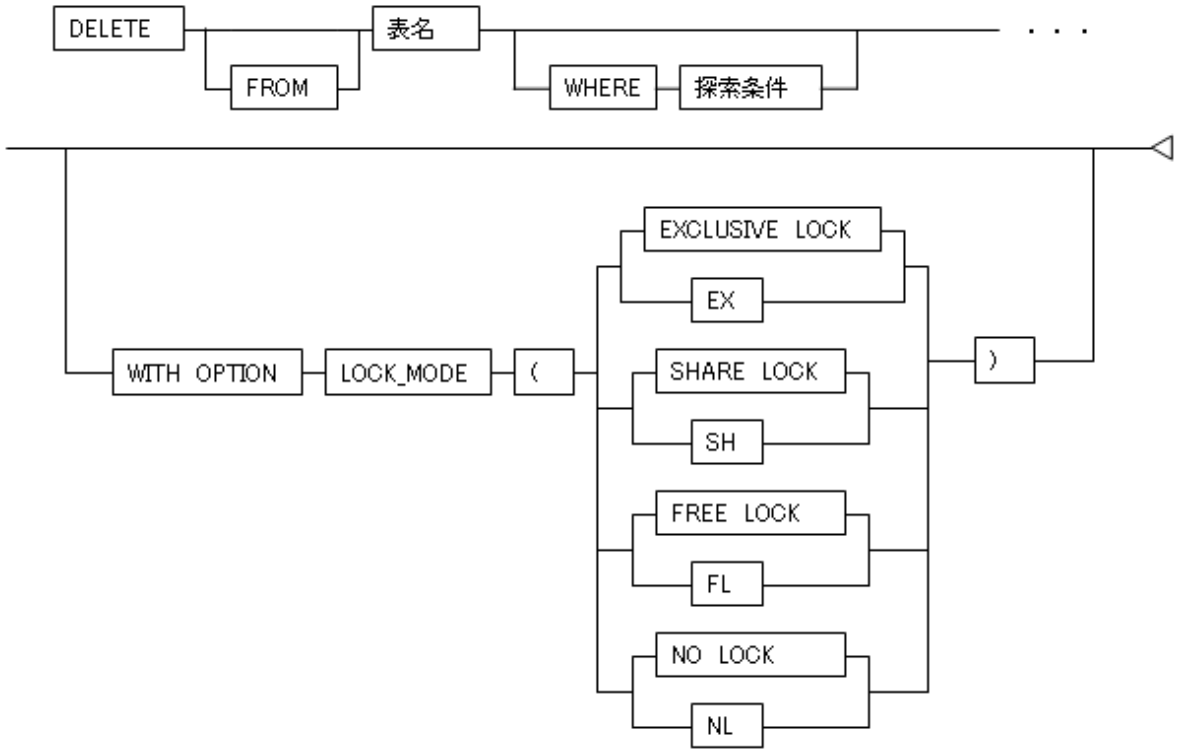


3.29 DELETE文:探索

機能

表から探索条件を満たす行を削除します。

記述形式



構文の構成

参照項番

- 探索条件 → “2.13 探索条件”

権限

- DELETE文:探索を実行できるのは、処理対象の表のDELETE権の保持者です。ただし、探索条件に副問合せを指定する場合、その副問合せに指定する表のSELECT権が必要です。

一般規則

- DELETE文にROWNUMは指定できません。

表名

ー 行を削除する表の名前を指定します。

ー 表名で指定した表がビュー表の場合、そのビュー表に含まれる表が対象となります。

ー 表名の有効範囲はDELETE文:探索の全体となります。

探索条件

ー 探索条件を指定した場合、探索条件が真となる行を削除します。

- 探索条件を省略した場合、表のすべての行を削除します。
- 探索条件によって真となる行が存在しなかった場合、または表に行が存在しなかった場合、例外(データなし)となります。
- 探索条件に副問合せを指定する場合、表名で指定した表を、副問合せのFROM句に指定することはできません。
- 探索条件の副問合せに指定した導出表にはORDER BY句を指定することはできません。
- 表名で指定した表に、トリガ事象にDELETEを指定したトリガを定義している場合、被トリガSQL文中に指定した表名と同じ表名を探索条件の副問合せのFROM句に指定することはできません。表名がビュー表の場合、そのビュー表に含まれる表が対象となります。

実行オプション指定

- 実行オプション指定は、占有モードを指定します。

占有モード指定

- 占有モード指定は、問合せ指定を指定した場合に、問合せ指定のデータベース資源の占有の方法を指定します。
- 占有モードを指定したSQL文が読み込んだデータベース資源は、占有モード指定により、以下のようにデータベース資源を占有します。なお、更新する行は占有モード指定にかかわらず、非共有モードでトランザクションの終了までデータベース資源を占有します。

占有モード指定	占有の方法
EXCLUSIVE LOCK	非共有モードでトランザクション終了までデータベース資源を占有します。
SHARE LOCK	共有モードでトランザクション終了までデータベース資源を占有します。
FREE LOCK	共有モードでSQL文終了までデータベース資源を占有します。
NO LOCK	データベース資源を占有しません。

- 占有モードを指定したSQL文によって読み込まれた資源は、SET TRANSACTION文で指定された内容にかかわらず、以下のようになります。

占有モード指定	読込み水準
EXCLUSIVE LOCK	他のトランザクションによって占有されていない行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
SHARE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
FREE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。当該SQL文で読み込んだ行は、他のトランザクションに更新されることがあるため、同一トランザクションで再検索すると最新の結果を検索することができます。
NO LOCK	他のトランザクションでどのような占有をされた資源でも参照することが可能なため、同一トランザクションで再検索すると最新の結果を検索することができます。

- 占有モードの単位は、R_LOCKがYESの場合は行単位となります。R_LOCKがNOの場合は、ページ単位またはDSI単位の占有になります。

― 動作環境パラメタにDSO_LOCKを指定したり、環境変数でRDBDSOを指定した場合、占有モードを指定したSQL文は実行できません。

― DELETE文にROWNUMは指定できません。

使用例

例1

表TBL1から探索条件が真となる行を削除する例を示します。

```
DELETE FROM TBL1
       WHERE COL2 = 'R'
         AND COL3 = 1
```

表TBL1が以下に示すデータを持っているとします。表TBL1の行のうち、探索条件で指定された条件を満たす6番目の行が削除されます。探索条件が指定されなかった場合には、表TBL1の全行が削除されます。

表TBL1(DELETE文実行前)

COL1	COL2	COL3
10	R	5
20	D	4
30	B	3
40	M	2
50	S	1
60	R	1

⇨

表TBL1(DELETE文実行後)

COL1	COL2	COL3	
10	R	5	
20	D	4	
30	B	3	
40	M	2	
50	S	1	
			←削除された行

例2

SQL文実行中の読み込まれた資源を共用モードで占有する例を示します。

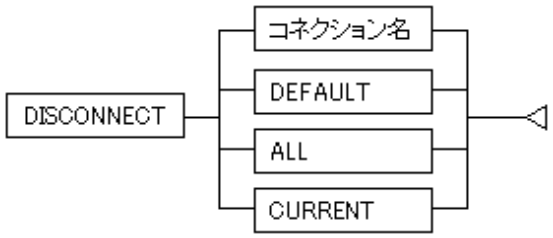
```
DELETE FROM 在庫管理.在庫表 WHERE COL1 = 1
          WITH OPTION LOCK_MODE(FREE LOCK)
```

3.30 DISCONNECT文

機能

コネクションを切断します。

記述形式



一般規則

- トランザクションが終了していることが必要です。
- 現コネクションを切断した場合、現コネクションがない状態となります。

コネクション名

- コネクション名を指定した場合、指定されたコネクションを切断します。この場合、同名のコネクション名を指定したCONNECT文により、コネクションが接続状態であることが必要です。
- コネクション名は、文字列定数または文字列型の埋込み変数で指定します。

DEFAULT

- DEFAULTを指定した場合、DEFAULTを指定したCONNECT文で接続されたコネクションを切断します。この場合、DEFAULTを指定したCONNECT文により、コネクションが接続状態であることが必要です。

ALL

- ALLを指定した場合、接続状態のすべてのコネクションを切断します。

CURRENT

- CURRENTを指定した場合、現コネクションを切断します。

使用例

例1

コネクション名に定数を指定します。

```
CONNECT TO 'SV1' AS 'C1' USER 'USER1/PASS1'
        :
DISCONNECT 'C1'
```

例2

ALLを指定します。

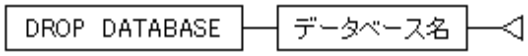
```
CONNECT TO DEFAULT
CONNECT TO 'SV2' AS 'C2' USER 'USER2/PASS2'
        :
DISCONNECT ALL
```

3.31 DROP DATABASE文(データベース削除文)

機能

データベース名を削除します。

記述形式



権限

- データベースを削除できるのは、RDBディクショナリの定義者のみです。

一般規則

- 削除するデータベースに、スキーマ、データベーススペースが存在しているとエラーになります。この場合には、該当のデータベースに属するスキーマ、データベーススペースを削除した後、データベース名を削除してください。

データベース名

- 削除するデータベースの名前を指定します。

使用例

例

データベース“RDBDB1”を削除します。

```
DROP DATABASE RDBDB1
```

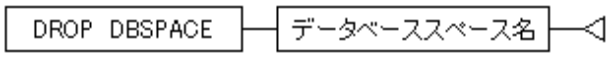
3.32 DROP DBSPACE文(データベーススペース削除文)

機能

データベーススペースを削除します。

データベース簡単運用の場合は利用できません。

記述形式



権限

- データベーススペースを削除できるのは、そのデータベーススペースの定義者のみです。

一般規則

- 削除するデータベーススペースに、表のDSIまたはインデックスのDSIが定義されていると削除することはできません。この場合、これらのDSIを削除してから、データベーススペースを削除することが必要です。
- UNIX系を使用する場合、削除対象のデータベーススペースが作成されている、ローデバイスのパーティションは削除されません。
- データベーススペース削除文でエラーが発生した場合、および、トランザクションがロールバックした場合、元の状態にはリカバリされず、アクセス禁止状態となります。エラーが発生した場合にはエラー原因を取り除いて、再度、データベーススペース削除文を実行することが必要です。

データベーススペース名

ー 削除するデータベーススペースの名前を指定します。

使用例

例

データベーススペース“RDBDBS1”を削除します。

```
DROP DBSPACE RDBDBS1
```

注意

- ロードシェア運用の場合、他ノードのRDBシステムまたは当該RDBシステムでRDBネットへの参入と離脱が行われていると、エラーとなります。
- ロードシェア運用の場合、ユーザログループの縮退または切り戻しが行われている場合、エラーとなります。

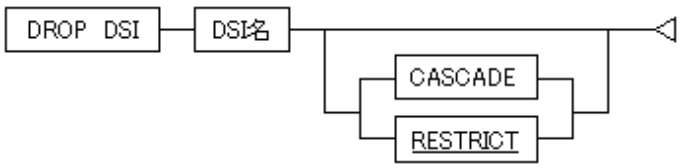
3.33 DROP DSI文(DSI削除文)

機能

DSI定義を削除します。

データベース簡単運用の場合は利用できません。

記述形式



権限

- DSIを削除できるのは、表の定義者、スキーマの定義者、表のDROP権の保持者またはスキーマのDROP権の保持者です。

一般規則

- DSI削除文でエラーが発生した場合、および、トランザクションがロールバックした場合、元の状態にはリカバリされず、アクセス禁止状態となります。エラーが発生した場合にはエラー原因を取り除いて、再度、DSI削除文を実行することが必要です。

DSI名

— 削除するDSIの名前を指定します。

CASCADEおよびRESTRICT(削除動作)

— CASCADEおよびRESTRICTは、“削除動作”と呼びます。

— CASCADEを指定した場合、削除するDSIによる格納対象が表である場合に、それに関連するインデックスのDSIはすべて削除されます。

— RESTRICTを指定した場合、または削除動作を省略した場合、表のDSIに関連するインデックスのDSIが定義されていると、表のDSIは削除できません。このようなDSIを削除する場合には、まず関連するDSIを削除してから、DSIを削除することが必要です。

使用例

例

“発注表DSI”を削除します。

```
DROP DSI 発注表ＤＳＩ
```

注意

- ロードシェア運用の場合、他ノードのRDBシステムまたは当該RDBシステムでRDBネットへの参入と離脱が行われていると、エラーとなります。
- ロードシェア運用の場合、ユーザロググループの縮退または切り戻しが行われている場合、エラーとなります。

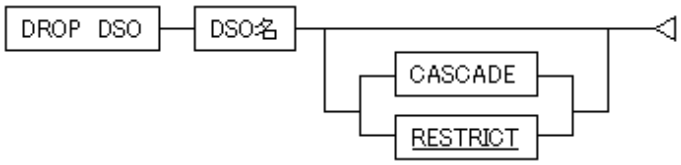
3.34 DROP DSO文(DSO削除文)

機能

表およびインデックスのDSO定義を削除します。

データベース簡単運用の場合は利用できません。

記述形式



権限

- DSOを削除できるのは、その表の定義者、スキーマの定義者、その表のDROP権の保持者またはスキーマのDROP権の保持者です。

一般規則

DSO名

ー 削除するDSOの名前を指定します。

CASCADEおよびRESTRICT(削除動作)

ー CASCADEおよびRESTRICTは、“削除動作”と呼びます。

ー CASCADEを指定した場合、削除するDSOに関連するDSIはすべて削除されます。また、実表のDSOを削除する場合、その表に関連するインデックスのDSOおよびそれに関連するDSIはすべて削除されます。

ー RESTRICTを指定した場合、または削除動作を省略した場合、DSIが定義されているDSOを削除することはできません。また、表に関連するインデックスのDSOが定義されていると、表のDSOを削除することはできません。このようなDSOを削除する場合には、まず関連するDSIおよびDSOを削除してから、DSOを削除することが必要です。

使用例

例

“会社表DSO”を削除します。

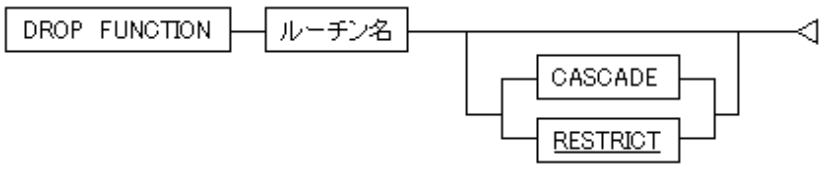
```
DROP DSO 会社表ＤＳＯ
```

3.35 DROP FUNCTION文 (ファンクションルーチン削除文)

機能

ファンクションルーチンを削除します。

記述形式



権限

- ファンクションルーチンを削除することができるのは、スキーマの定義者、ファンクションルーチンの定義者とDROP権を付与された人です。

一般規則

ルーチン名

― 削除するファンクションルーチンの名前を指定します。

CASCADEおよびRESTRICT(削除動作)

― CASCADEおよびRESTRICTは、“削除動作”と呼びます。

― CASCADEを指定した場合、削除対象のファンクションルーチンをSQL文に指定したプロシジャルーチンやトリガもすべて削除されます。

― RESTRICTを指定した場合、または削除動作を省略した場合、削除対象のファンクションルーチンをSQL文中に指定したプロシジャルーチンやトリガが定義されていないことが必要です。このような場合は、削除対象のファンクションルーチンを指定するプロシジャルーチンやトリガを削除してから、ファンクションルーチンを削除することが必要です。

使用例

例

ファンクションルーチン“FUNC1”を削除します。

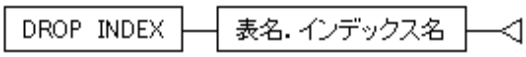
```
DROP FUNCTION S1.FUNC1
```

3.36 DROP INDEX文(インデックス削除文)

機能

インデックス定義を削除します。

記述形式



権限

・ インデックスを削除できるのは、そのインデックスを定義した表の定義者、スキーマの定義者、そのインデックスを定義した表のDROP権の保持者またはスキーマのDROP権の保持者です。

一般規則

・ インデックス定義を削除した場合は、インデックスの格納構造定義はSymfoware/RDBが自動的に削除します。

・ インデックス削除文でエラーが発生した場合、および、トランザクションがロールバックした場合、元の状態にはリカバリされず、アクセス禁止状態となります。エラーが発生した場合にはエラー原因を取り除いて、再度、インデックス削除文を実行することが必要です。

表名

― インデックスが定義されている表の名前を指定します。

インデックス名

― 削除するインデックスの名前を指定します。

使用例

例

スキーマS1の表T1に対して定義されたインデックスIXAを削除します。

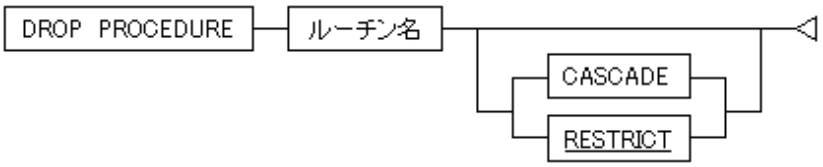
```
DROP INDEX S1.T1.IXA
```

3.37 DROP PROCEDURE文(プロシジャルーチン削除文)

機能

プロシジャルーチンを削除します。

記述形式



権限

- プロシジャルーチンに対してすべての権限が付与されている場合も、そのプロシジャルーチンを削除できるのは、そのプロシジャルーチンの定義者、スキーマの定義者、そのプロシジャルーチンのDROP権の保持者またはスキーマのDROP権の保持者です。
- プロシジャルーチンを削除した場合、プロシジャルーチンに対する権限もすべて消去されます。

一般規則

ルーチン名

ー 削除するプロシジャルーチンの名前を指定します。

CASCADEおよびRESTRICT(削除動作)

ー CASCADEおよびRESTRICTは、“削除動作”と呼びます。

ー CASCADEを指定した場合、削除対象のルーチンを呼び出しているルーチンもすべて削除されます。

ー CASCADEを指定した場合、削除対象のプロシジャルーチンをSQL手続き文中に指定したプロシジャルーチンはすべて削除されます。

ー RESTRICTを指定した場合、または削除動作を省略した場合、削除対象のルーチンを呼び出しているルーチンが存在していると削除することはできません。削除する場合は、呼び出しているルーチンから削除することが必要です。

ー RESTRICTを指定した場合、または削除動作を省略した場合、削除対象のプロシジャルーチンをSQL手続き文中に指定したプロシジャルーチンが定義されていないことが必要です。このような場合は、削除対象のプロシジャルーチンを指定するプロシジャルーチンを削除してから、プロシジャルーチンを削除することが必要です。

使用例

例

プロシジャルーチン“ROUTINE1”を削除します。

```
DROP PROCEDURE S1.ROUTINE1
```

3.38 DROP ROLE文(ロール削除文)

機能

ロールを削除します。

記述形式

権限

・ ロールを削除できるのは、RDBディクショナリの定義者のみです。

一般規則

ロール名

― 削除するロールの名前を指定します。

使用例

例

ロール“STOCKS_A2”を削除します。

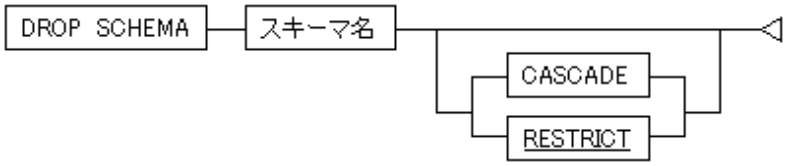
```
DROP ROLE STOCKS_A2
```

3.39 DROP SCHEMA文(スキーマ削除文)

機能

スキーマを削除します。

記述形式



権限

・ スキーマを削除できるのは、そのスキーマの定義者またはそのスキーマのDROP権の保持者です。

・ スキーマ定義を削除した場合、その資源に対する権限も削除されます。

一般規則

スキーマ名

― 削除するスキーマの名前を指定します。

CASCADEおよびRESTRICT(削除動作)

― CASCADEおよびRESTRICTは、“削除動作”と呼びます。

― CASCADEを指定した場合、削除対象のスキーマ内の実表、一時表、ビュー表、プロシジャルーチン、トリガ、ファンクションルーチン、順序、DSO、DSIおよびインデックスがすべて削除されます。

― RESTRICTを指定した場合、または削除動作を省略した場合、スキーマ内に実表、一時表、ビュー表、プロシジャルーチン、トリガ、ファンクションルーチン、順序、DSO、DSIまたはインデックスが定義されているスキーマを削除することはできません。これらがスキーマ内に定義されているスキーマを削除する場合には、まずそれらの定義を削除してからスキーマを削除することが必要です。

使用例

例

スキーマ“S1”を削除します。

```
DROP SCHEMA S1
```

3.40 DROP SCOPE文(スコープ削除文)

機能

スコープを削除します。

データベース簡単運用の場合は利用できません。

記述形式

権限

・ スコープを削除できるのは、スコープの定義者のみです。

一般規則

・ スコープ削除文が実行されると、利用者に対して適用されているすべてのスコープ情報は削除されます。

スコープ名

ー 削除するスコープの名前を指定します。

使用例

例

スコープ“東京ＳＣＰ”を削除します。

```
DROP SCOPE 東京ＳＣＰ
```

3.41 DROP SEQUENCE文 (順序削除文)

機能

順序を削除します。

記述形式

権限

・ 順序を削除できるのは、スキーマ定義者、順序定義者およびDROP権を付与された人です。

一般規則

・ 削除対象の順序を指定した表、プロシジャルーチン、トリガが定義されている場合、順序を削除することはできません。これらの定義を削除してから、順序を削除することが必要です。

順序名

ー 削除する順序の名前を指定します。

使用例

例

順序“順序1”を削除します。

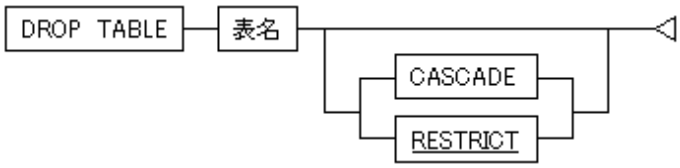
```
DROP SEQUENCE S1.順序１
```

3.42 DROP TABLE文(表削除文)

機能

表を削除します。

記述形式



権限

- 表を削除できるのは、その表の定義者、スキーマの定義者、その表のDROP権の保持者またはスキーマのDROP権の保持者です。
- 表を削除した場合、その資源に対する権限も削除されます。

一般規則

表名

ー 削除する実表または一時表の名前を指定します。

CASCADEおよびRESTRICT(削除動作)

ー CASCADEおよびRESTRICTは、“削除動作”と呼びます。

ー CASCADEを指定した場合、削除対象の表を指定したビュー表、プロシジャルーチンおよびトリガはすべて削除されます。

ー CASCADEを指定した場合、削除する表に関連するDSO、インデックスのDSO、そのDSOに関連するDSI、インデックスのDSIおよび表に付加されたインデックスはすべて削除されます。

ー RESTRICTを指定した場合または削除動作を省略した場合、削除対象の表を指定したビュー表、プロシジャルーチン、トリガ、削除する表に関連するDSO、インデックスのDSO、そのDSOに関連するDSI、インデックスのDSIおよび表に付加されたインデックスが定義されている表を削除することはできません。これらが定義されている表を削除する場合には、まずそれらの定義を削除してから、表を削除することが必要です。
ただし、システムが自動的に定義したインデックスは削除する必要はありません。表と合わせて削除されます。

使用例

例

スキーマS1の表“発注表”を削除します。

```
DROP TABLE S1.発注表
```

3.43 DROP TRIGGER文(トリガ削除文)

機能

トリガを削除します。

記述形式

権限

- トリガを削除できるのは、そのトリガの定義者、スキーマの定義者、そのトリガのDROP権の保持者またはスキーマのDROP権の保持者です。

一般規則

トリガ名

ー 削除するトリガの名前を指定します。

使用例

例

トリガ“TRG1”を削除します。

```
DROP TRIGGER S1.TRG1
```

3.44 DROP USER文(利用者削除文)

機能

利用者の定義を削除します。

記述形式

権限

- 利用者を削除できるのは、RDBディクショナリの定義者のみです。

一般規則

認可識別子

ー 削除する認可識別子を指定します。

ー 利用者登録の使用宣言を行っている場合、監査ログパラメタのAUDITに指定した利用者は削除できません。監査ログパラメタのAUDITから削除する利用者の指定を外してください。監査ログパラメタの詳細については、“3.61 SET SYSTEM PARAMETER文”を参照してください。

使用例

例

認可識別子“SUZUKI”を削除します。

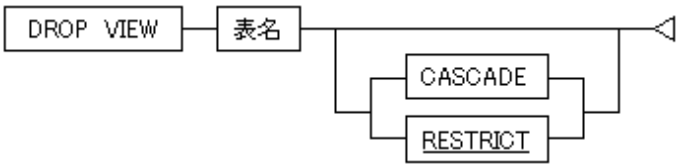
```
DROP USER SUZUKI
```

3.45 DROP VIEW文(ビュー削除文)

機能

ビュー表を削除します。

記述形式



権限

- ビュー表を削除した場合、その資源に対する権限も削除されます。
- ビュー表を削除できるのは、そのビュー表の定義者、スキーマの定義者、そのビュー表のDROP権の保持者またはスキーマのDROP権の保持者です。

一般規則

表名

ー 削除するビュー表の名前を指定します。

CASCADEおよびRESTRICT(削除動作)

ー CASCADEおよびRESTRICTは、“削除動作”と呼びます。

ー CASCADEを指定した場合、削除対象のビュー表を指定したビュー表、プロシジャルーチンおよびトリガはすべて削除されます。

ー RESTRICTを指定した場合または削除動作を省略した場合、削除対象のビュー表を問合せ指定に指定したビュー表、プロシジャルーチンおよびトリガが定義されていないことが必要です。削除対象のビュー表を問合せ指定に指定したビュー表が定義されているビュー表を削除する場合には、まず、それらの定義を削除してからビュー表を削除することが必要です。

使用例

例

スキーマS1のビュー表“会社表1”を削除します。

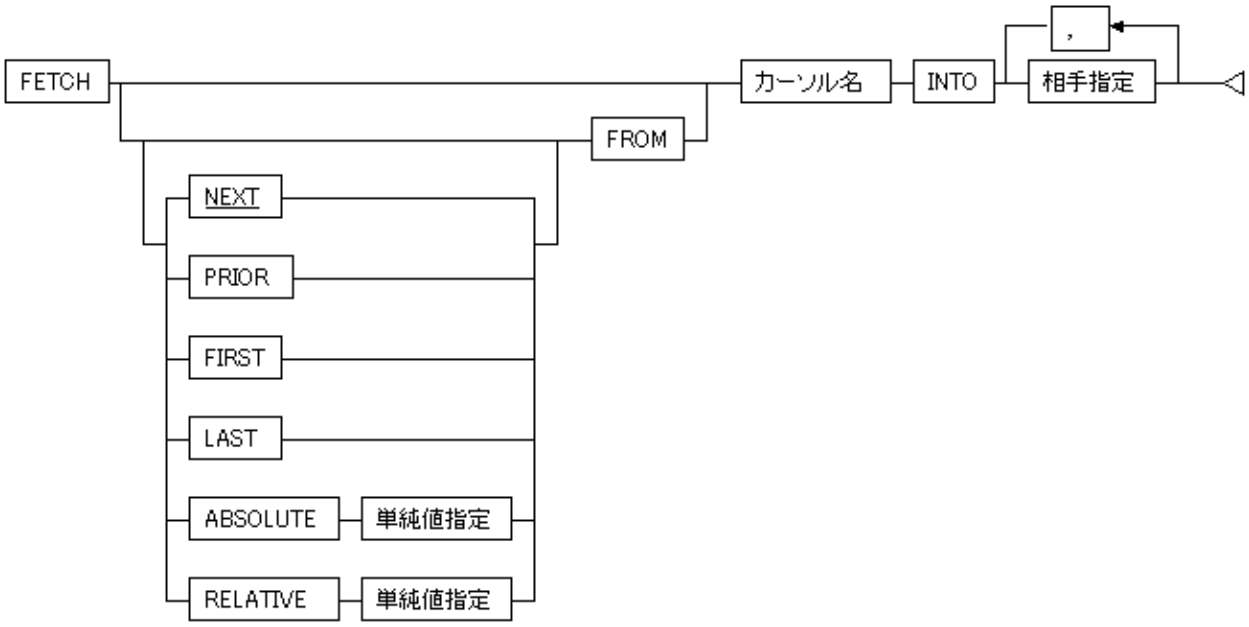
```
DROP VIEW S1.会社表１
```

3.46 FETCH文

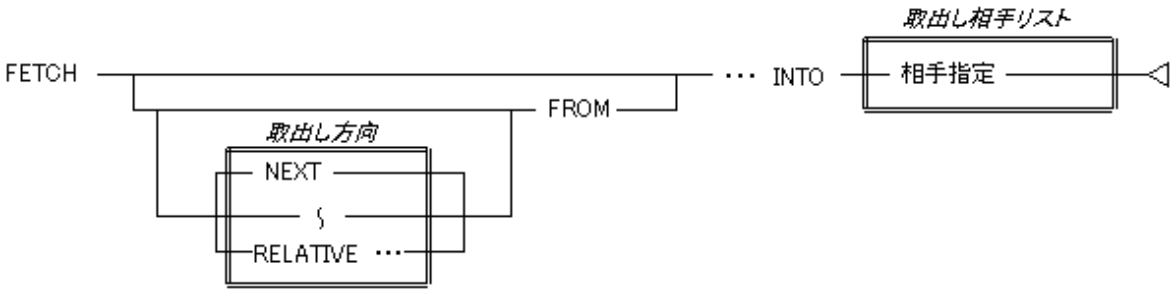
機能

カーソルを次の行に位置づけ、その行から値を取り出します。

記述形式



構文の構成



参照項番

- 単純値指定 → “2.3 値指定と相手指定”
- 相手指定 → “2.3 値指定と相手指定”

権限

- FETCH文を実行できるのは、カーソルの処理対象の表に対するSELECT権の保持者です。

一般規則

- カーソルは、開かれた状態であることが必要です。
- カーソルは、次の行に位置づけられます。
- 相手指定の変数に値を代入する途中に誤りが起こると、データ例外(代入エラー)になります。
- カーソルが、導出された表の最終行に位置づけられている状態でFETCH文を発行すると例外(データなし)となります。

取出し方向

ー NEXT、PRIOR、FIRST、LAST、ABSOLUTEおよびRELATIVEは、“取出し方向”と呼びます。

ー 取出し方向を指定した場合、カーソル宣言にSCROLLの指定が必要です。ただし、NEXTを指定した場合は、カーソル宣言のSCROLLの指定を省略することができます。

ー 先頭行より前の行を取り出すと例外(データなし)となり、カーソルは先頭行の直前に位置づけられます。

ー 最終行よりうしろの行を取り出すと例外(データなし)となり、カーソルは最終行の直後に位置づけられます。

ー 取出し方向を省略するとNEXTが指定されたものとみなします。

NEXT

- 取出し方向にNEXTを指定した場合、カーソルが位置づけられている行の次の行を取り出します。

PRIOR

- 取出し方向にPRIORを指定した場合、カーソルが位置づけられている行の直前の行を取り出します。

FIRST

- 取出し方向にFIRSTを指定した場合、先頭の行を取り出します。

LAST

- 取出し方向にLASTを指定した場合、最後の行を取り出します。

ABSOLUTE

- 取出し方向にABSOLUTEを指定した場合、先頭行から単純値指定で示す値の分だけうしろの行を取り出します。単純値指定に負の値を指定した場合は、最終行から単純値指定で示す値の絶対値分だけ前の行を取り出します。

RELATIVE

- 取出し方向にRELATIVEを指定すると、カーソルが位置づけられている行に単純値指定で示す値を加算した行を取り出します。

単純値指定

- 単純値指定はINTEGER型またはSMALLINT型とします。
- 単純値指定に標識変数は指定できません。

カーソル名

ー カーソルの名前を指定します。

ー カーソル名は、同一コンパイル単位に含まれるカーソル宣言で定義されていることが必要です。

相手指定(取出し相手リスト)

ー 取出し相手リストは、相手指定をカンマ(,)で区切って記述します。

ー カーソル宣言で指定した問合せ指定の選択リストの個数と、取出し相手リストに指定する相手指定の個数は、同じであることが必要です。

ー カーソル宣言で指定した問合せ指定の選択リストの結果のデータ型に対して、相手指定に指定可能なデータ型の条件を以下に示します。

選択リストのデータ型が文字列型の場合:

相手指定のデータ型は文字列型であることが必要です。

選択リストのデータ型が各国語文字列型の場合:

相手指定のデータ型は各国語文字列型であることが必要です。

選択リストのデータ型が真数型の場合:

相手指定のデータ型は真数型または概数型であることが必要です。

選択リストのデータ型が概数型の場合:

相手指定のデータ型は真数型または概数型であることが必要です。

選択リストのデータ型が日時型の場合:

相手指定のデータ型は文字列型であることが必要です。

選択リストのデータ型が時間隔型の場合:

相手指定のデータ型は文字列型または真数型であることが必要です。ただし、真数型が指定できるのは選択リストのデータ型が単一日時フィールドで指定された時間隔型の場合のみです。

選択リストのデータ型がBLOB型の場合:

相手指定のデータ型はBLOB型であることが必要です。

— カーソル宣言で指定した問合せ指定の選択リストにROW_IDを指定した場合、相手指定にはROW_IDと対応する変数を指定することが必要です。データ型と対応する変数定義については、“表6.1 SQLのデータ型と対応するC変数定義”および“表6.3 SQLのデータ型と対応するCOBOL変数定義”を参照してください。

— カーソル指定の列がNULL値の場合の扱いは次のとおりです。

- 相手指定に標識変数が指定されていれば、その標識変数には-1が設定されます。
- 相手指定に標識変数が指定されていなければ、データ例外(NULL値、標識なし)となります。

— カーソル指定の列がNULL値でなく、相手指定に標識変数が指定されているならば、その標識変数には次の値が入ります。

- 相手指定のデータ型が固定長または可変長の文字列型の場合で、カーソル指定の列の長さが相手指定の文字列の最大長よりも長ければ、カーソル指定の列の文字数が設定されます。
- 固定長または可変長の各国語文字列型の場合も、同様に設定されます。
- そうでなければ、標識変数には0が設定されます。

— カーソル指定の列が、相手指定に設定されるときの規則は次のようになります。

文字列型データの場合:

カーソル指定の列のデータ長が相手指定の長さと同じ場合は、文字データがそのまま設定されます。カーソル指定の列のデータ長が相手指定の長さよりも短い場合は、空白が相手指定の右側に設定されます。カーソル指定の列のデータ長が相手指定の長さよりも長い場合は、相手指定の左側から相手指定の長さだけの文字データが設定されます。

各国語文字列型の場合:

文字列型の場合と同じように設定されます。

真数型の場合:

相手指定のデータ型に変換して設定されます。カーソル指定の列の整数部が相手指定の精度と位取りで表現できる場合は、相手指定の精度と位取りに従って設定されます。上位の桁落ちが発生する場合は例外となり、下位の位が落ちる場合は切り捨てられます。

概数型のデータの場合:

相手指定には、カーソル指定の列の概数値が設定されます。

日時型のデータの場合:

相手指定のデータ型に変換して設定されます。

時間隔型のデータの場合:

相手指定のデータ型に変換して設定されます。

BLOB型のデータの場合:

相手指定には、カーソル指定の列のバイナリ値が設定されます。

ROW_IDの場合:

相手指定には、行識別子が設定されます。

使用例

例1

カーソル宣言およびカーソルを開いてから閉じるまでの一連の流れを、表T1で説明します。

1. 表T1の検索のためにカーソルCSR1を定義します。

```
DECLARE CSR1 CURSOR FOR
    SELECT COL1,COL2,COL3 FROM T1 WHERE COL4 > 10
      ORDER BY COL2 DESC,COL3 ASC
```

カーソル宣言は非実行文です。アプリケーション中でこのカーソルを使用するOPEN文、FETCH文およびCLOSE文よりも前に記述します。実行の順序とは関係ありません。

2. カーソルを開いた状態にします。

```
OPEN CSR1
```

カーソルは導出された表の先頭行よりも前の(1)に位置づけられます。

3. カーソルを位置づけます。相手指定のTGT1、TGT2およびTGT3には位置づけられた行の各列の値が取り出されます。

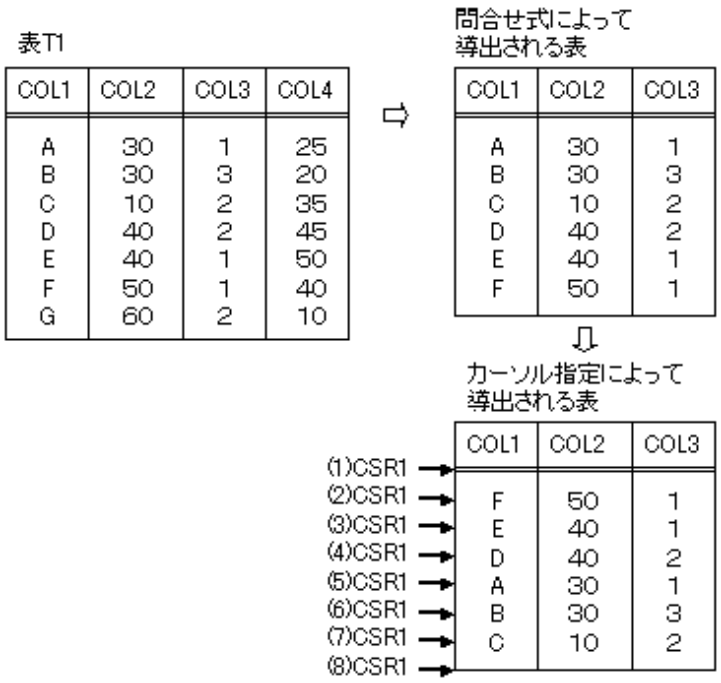
```
FETCH CSR1 INTO :TGT1,:TGT2,:TGT3
```

カーソルを開いてから最初のFETCH文の実行で、カーソルは(2)に位置づけられます。FETCH文を繰り返して実行することにより、カーソルは(3)から(7)に順に位置づけられ、探索条件を満たす行が次々と取り出されます。また、カーソルが最終行を位置づけている状態でFETCH文を実行すると、カーソルは(8)に位置づけられます。このとき、状態変数(SQLSTATE)にデータなしの例外コードが設定されます。SQLSTATEの値については、“付録A SQLSTATE値”を参照してください。

4. カーソルを閉じた状態にします。

```
CLOSE CSR1
```

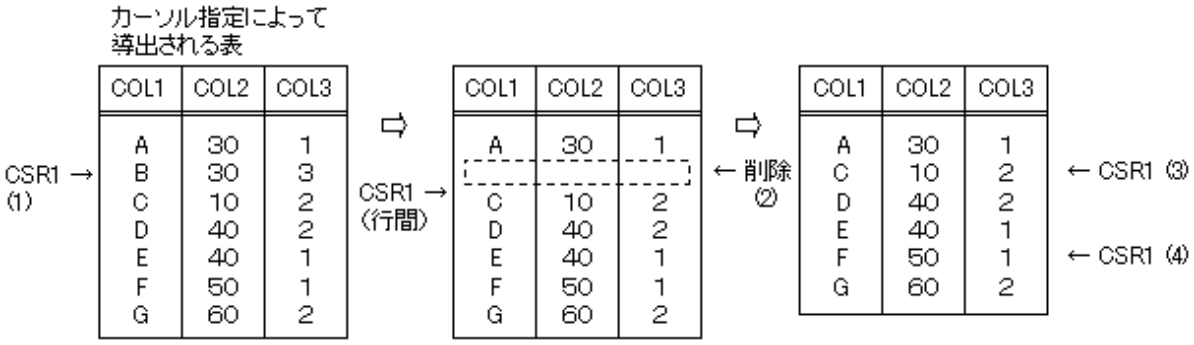
カーソルは閉じられ、再び開いた状態にするまで使用できません。なお、カーソルはどの行に位置づけられた状態でも閉じることができます。



例2

カーソルによって位置づけられた行を削除したあと、次のFETCH文で位置づけられる行について説明します。なお、(2)のFETCH文を実行する前は、カーソルは先頭行を位置づけている状態とします。

```
DECLARE CSR1 SCROLL CURSOR FOR
       SELECT COL1,COL2,COL3 FROM T1 FOR UPDATE
     :
OPEN CSR1
     :
FETCH NEXT FROM CSR1 INTO :H1,:H2,:H3                    (1)
DELETE FROM T1 WHERE CURRENT OF CSR1                     (2)
FETCH NEXT FROM CSR1 INTO :H1,:H2,:H3                    (3)
FETCH ABSOLUTE 5 FROM CSR1 INTO :H1,:H2,:H3              (4)
```



備考．DELETE文:位置づけで削除した行が詰められたあとのカーソル表で、次に位置づける行が決められます。

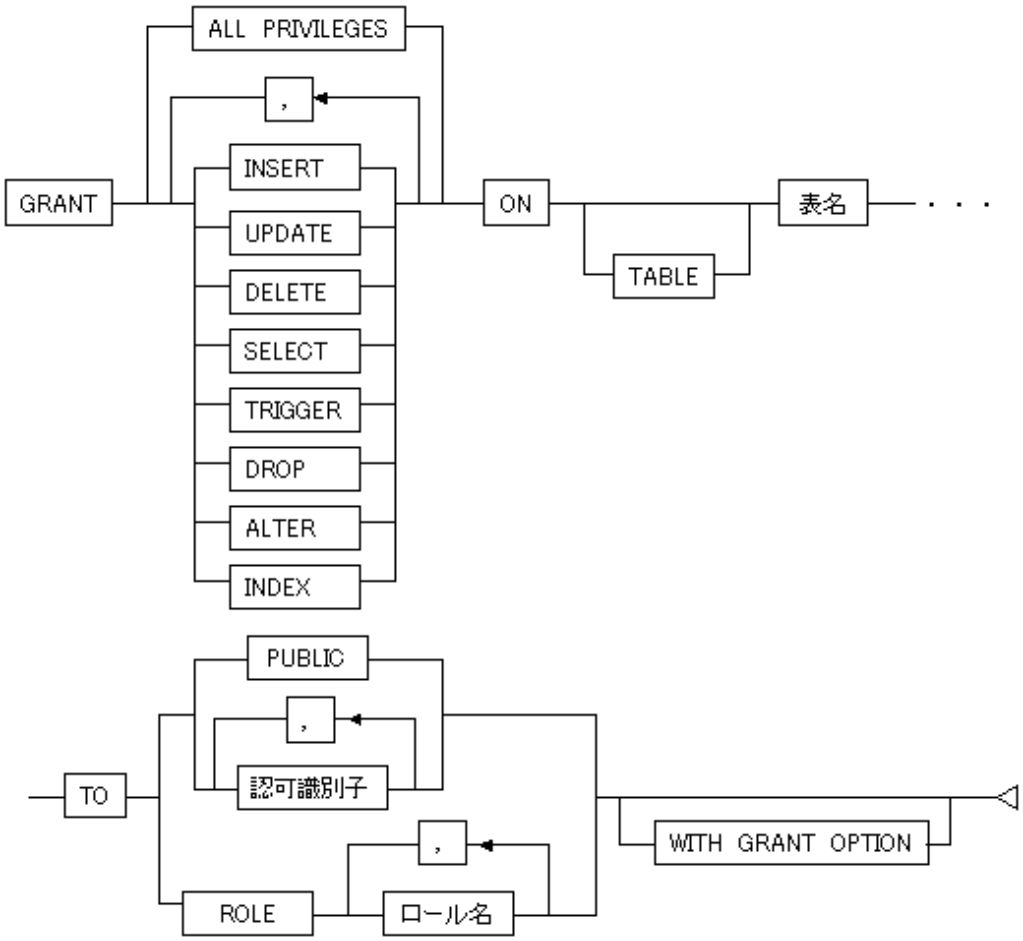
3.47 GRANT文

機能

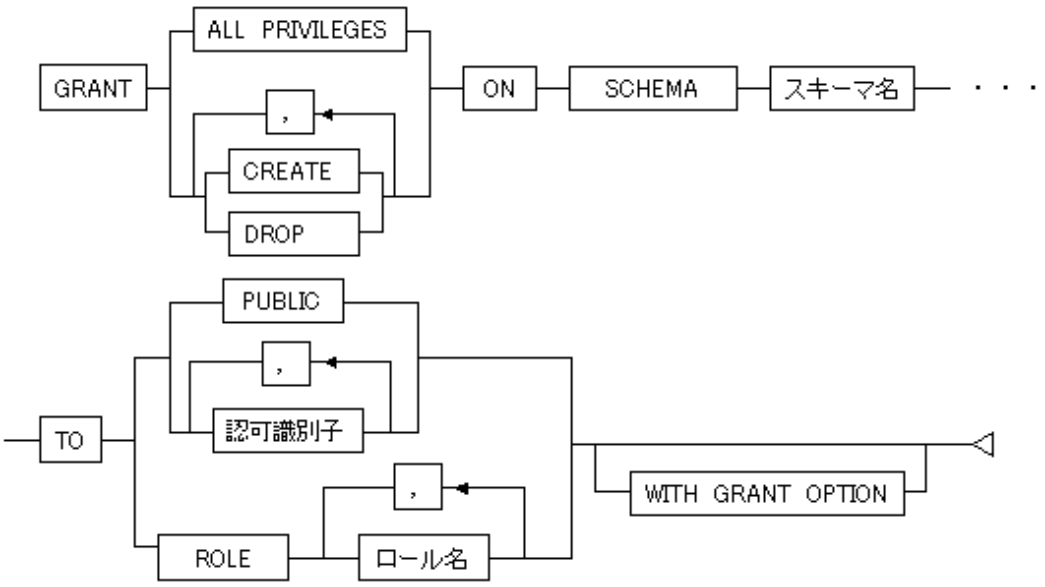
権限を定義します。

記述形式

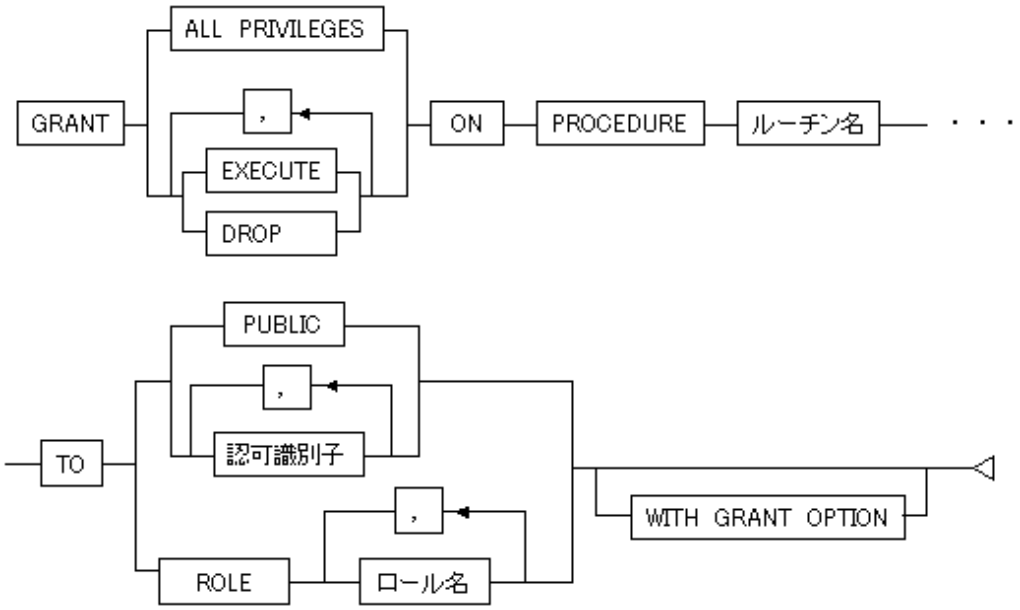
表に対する権限を認可識別子またはロールに付与



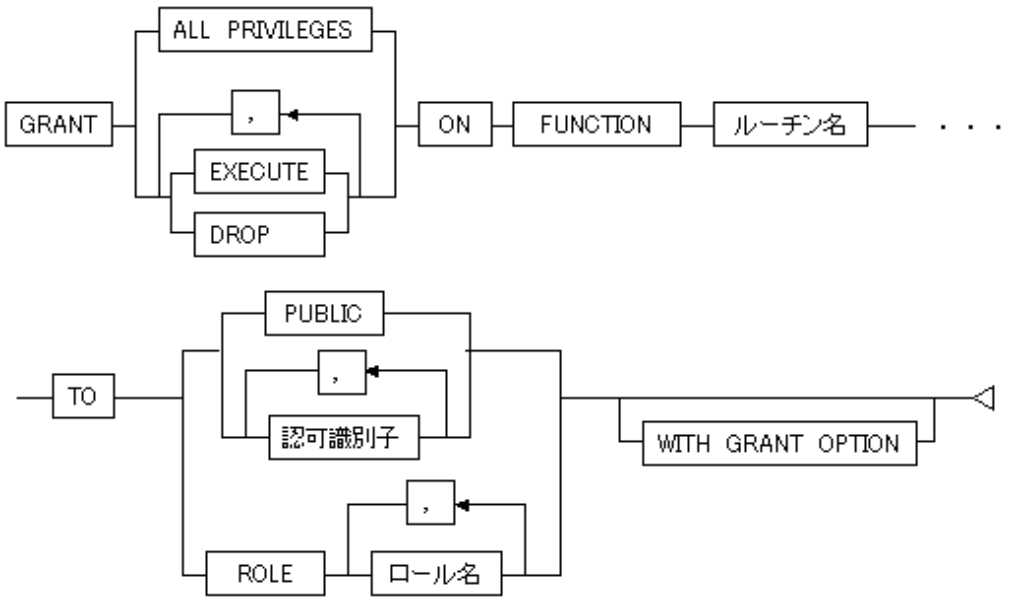
スキーマに対する権限を認可識別子またはロールに付与



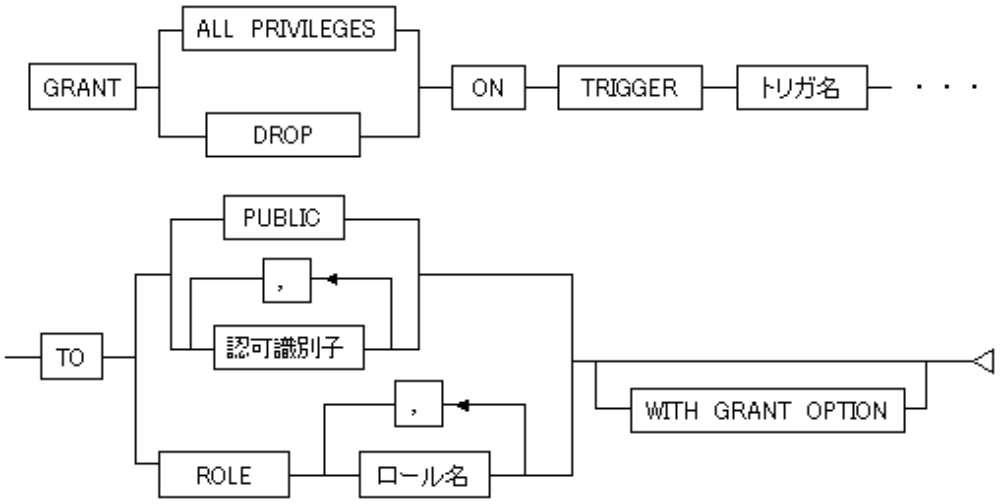
プロシジャルーチンに対する権限を認可識別子またはロールに付与



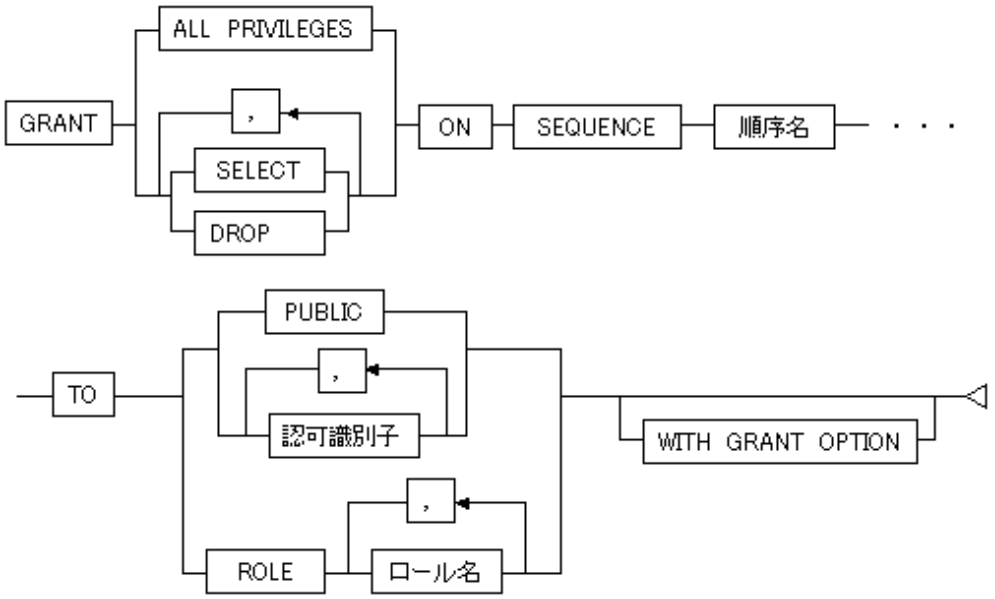
ファンクションルーチンに対する権限を認可識別子またはロールに付与



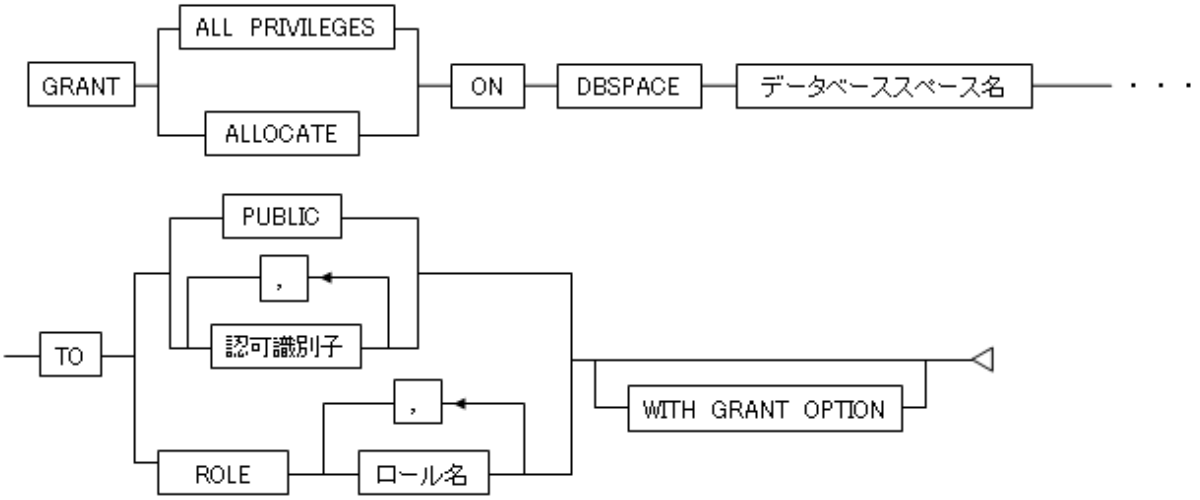
トリガに対する権限を認可識別子またはロールに付与



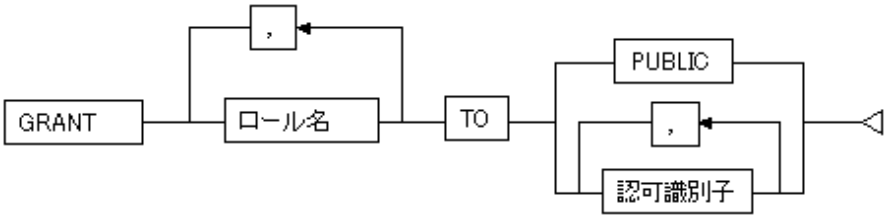
順序に対する権限を認可識別子またはロールに付与



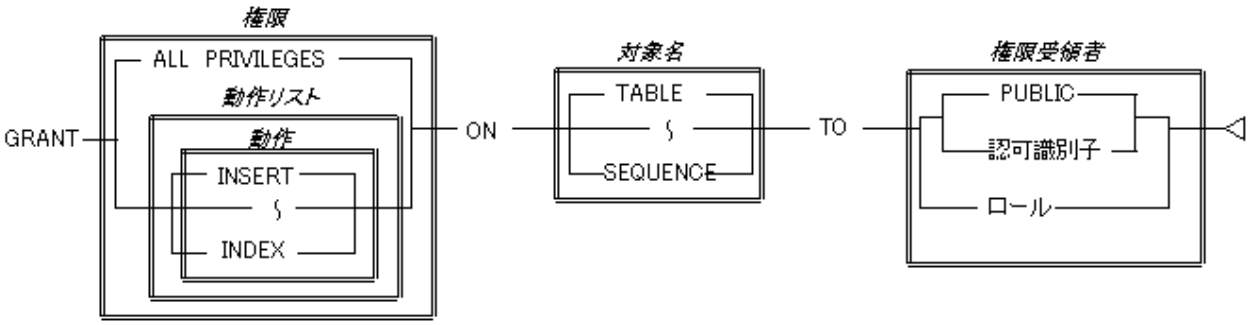
データベーススペースに対する権限を認可識別子またはロールに付与



ロールを他の利用者に付与



構文の構成



参照項番

- 日本語文字列 → “2.1.3 トークン”

一般規則

- 存在しない資源を指定することはできません。
- 付与者が付与可能な権限を持たないなら権限を付与することはできません。
- ビューで参照する表に対するINSERT権、DELETE権、UPDATE権が、ビュー表の定義者に付与されたとき、そのビュー表が更新可能なビューならば、ビュー表に対する同じ権限もビュー表の定義者に付与されます。

ALL PRIVILEGES

ー ALL PRIVILEGESが指定された場合、付与者が付与可能なすべての権限が、指定された資源に設定できる範囲で付与されます。

ー すでに付与者が指定された資源に対し個別に権限を設定している場合、ALL PRIVILEGESは指定できません。

動作リスト

ー 動作は、権限の対象となる動作です。

INSERT

- 権限受領者に対して表に行を挿入する権限(INSERT権)が付与されます。

UPDATE

- 権限受領者に対して表の行を更新する権限(UPDATE権)が付与されます。

DELETE

- 権限受領者に対して表の行を削除する権限(DELETE権)が付与されます。

SELECT

- 権限受領者に対して表の行または順序を参照する権限(SELECT権)が付与されます。

EXECUTE

- 権限受領者に対してプロシジャルーチンまたはファンクションルーチンを実行する権限(EXECUTE権)が付与されます。

CREATE

- 権限受領者に対してスキーマや表などの資源を定義する権限(CREATE権)が付与されます。

ALLOCATE

- 権限受領者に対してデータベーススペースに対して領域を割り当てる権限(ALLOCATE権)が付与されます。

TRIGGER

- 権限受領者に対してその表にトリガを定義する権限(TRIGGER権)が付与されます。
- ビュー表の場合は付与できません。

DROP

- 権限受領者に対してスキーマや表などの資源を削除する権限(DROP権)が付与されます。

ALTER

- 権限受領者に対して表定義を更新する権限(ALTER権)が付与されます。
- ビュー表の場合は付与できません。

INDEX

- 権限受領者に対して表にインデックスを定義する権限(INDEX権)が付与されます。
- ビュー表の場合は付与できません。

対象名

ー 対象名は、権限の対象となる資源の名前です。

表名

- 権限を付与する表の名前を指定します。

ルーチン名

- 権限を付与するプロシジャルーチンまたはファンクションルーチンの名前を指定します。

スキーマ名

- 権限を付与するスキーマの名前を指定します。

データベーススペース名

- 権限を付与するデータベーススペースの名前を指定します。

トリガ名

- 権限を付与するトリガの名前を指定します。

順序名

- 権限を付与する順序の名前を指定します。

ロール名(ロールを利用者に付与する場合に指定)

- 同じロール名を複数指定することはできません。
- 指定したロールは定義済でなければなりません。
- 実行者は、指定したロールの付与権を保持していなければなりません。

権限受領者

PUBLIC

- PUBLICは、データベースをアクセスするすべての人を意味します。
- PUBLICは、1回のみ指定できます。

注意

PUBLICとは、暗黙に定義された「データベースをアクセスするすべての利用者の集合」を意味します。

個々の利用者は、認可識別子を指定して直接許可された権限と、PUBLICを指定して許可された権限を合わせて受領していることに注意してください。

したがって、たとえば、PUBLICからSELECT権を剥奪することは、必ずしもその資源に対するSELECT権をすべての利用者が失うことを意味しません。直接権限を付与された人は、その権限を持ち続けます。

認可識別子

- 権限を付与する認可識別子を指定します。
- 認可識別子は、18文字以内の先頭が英字で始まる英数字、または9文字以内の日本語文字列を指定します。
- 英小文字の区切り識別子を認可識別子に指定することはできません。

ロール名(スキーマ、表、プロシジャルーチン、ファンクションルーチン、トリガ、順序およびデータベーススペースに対する権限をロールに付与する場合に指定)

- ロール名は、権限を付与するロールの名前を指定します。
- 同じロール名を複数指定することはできません。
- 指定したロールは定義済でなければなりません。
- 実行者は、指定した権限の付与権を保持していなければなりません。

WITH GRANT OPTION

- WITH GRANT OPTIONを指定すると、指定した権限をほかの人に与える権限(付与権)も与えることができます。
- ROLEの場合は指定できません。
- 権限受領者にPUBLICを指定している場合、PUBLICに対して付与権は付与されません。

使用例

例1

利用者“YAMADA”および“TANAKA”に、在庫表に対する全権限を付与します。

```
GRANT ALL PRIVILEGES ON TABLE STOCKS.在庫表 TO YAMADA,TANAKA
```

例2

ロール“STOCKS_A2”に在庫表に対する必要な権限を付与します。

```
GRANT SELECT,UPDATE,INSERT ON STOCKS.在庫表 TO ROLE STOCKS_A2
```

例3

ロール“STOCKS_A2”を利用者“SUZUKI”に付与します。

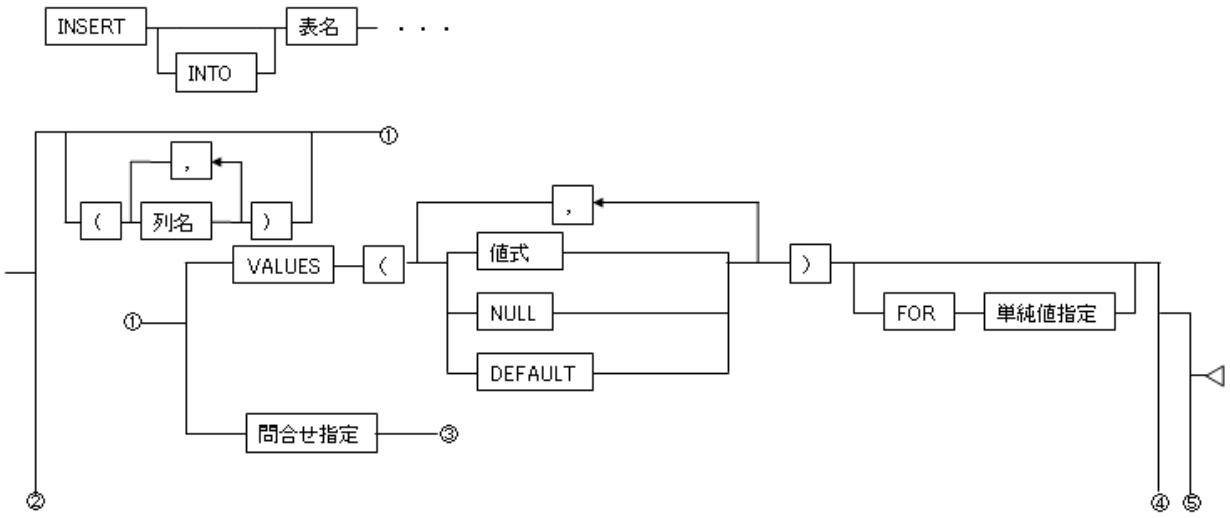
```
GRANT STOCKS_A2 TO SUZUKI
```

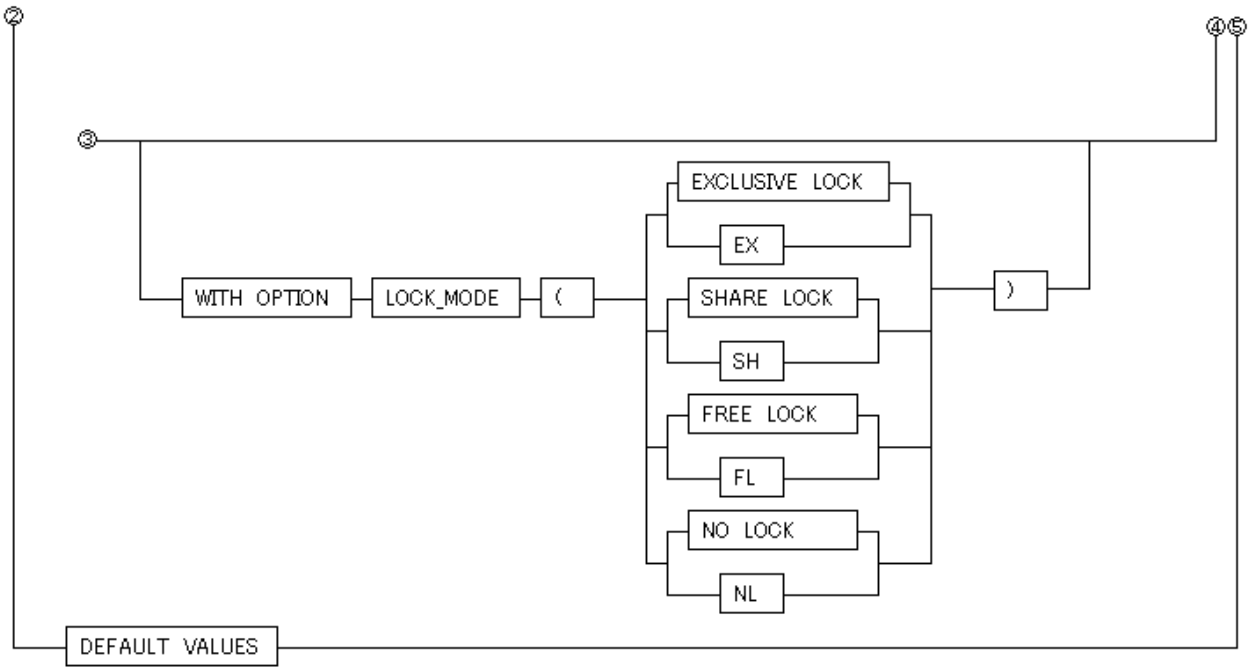
3.48 INSERT文

機能

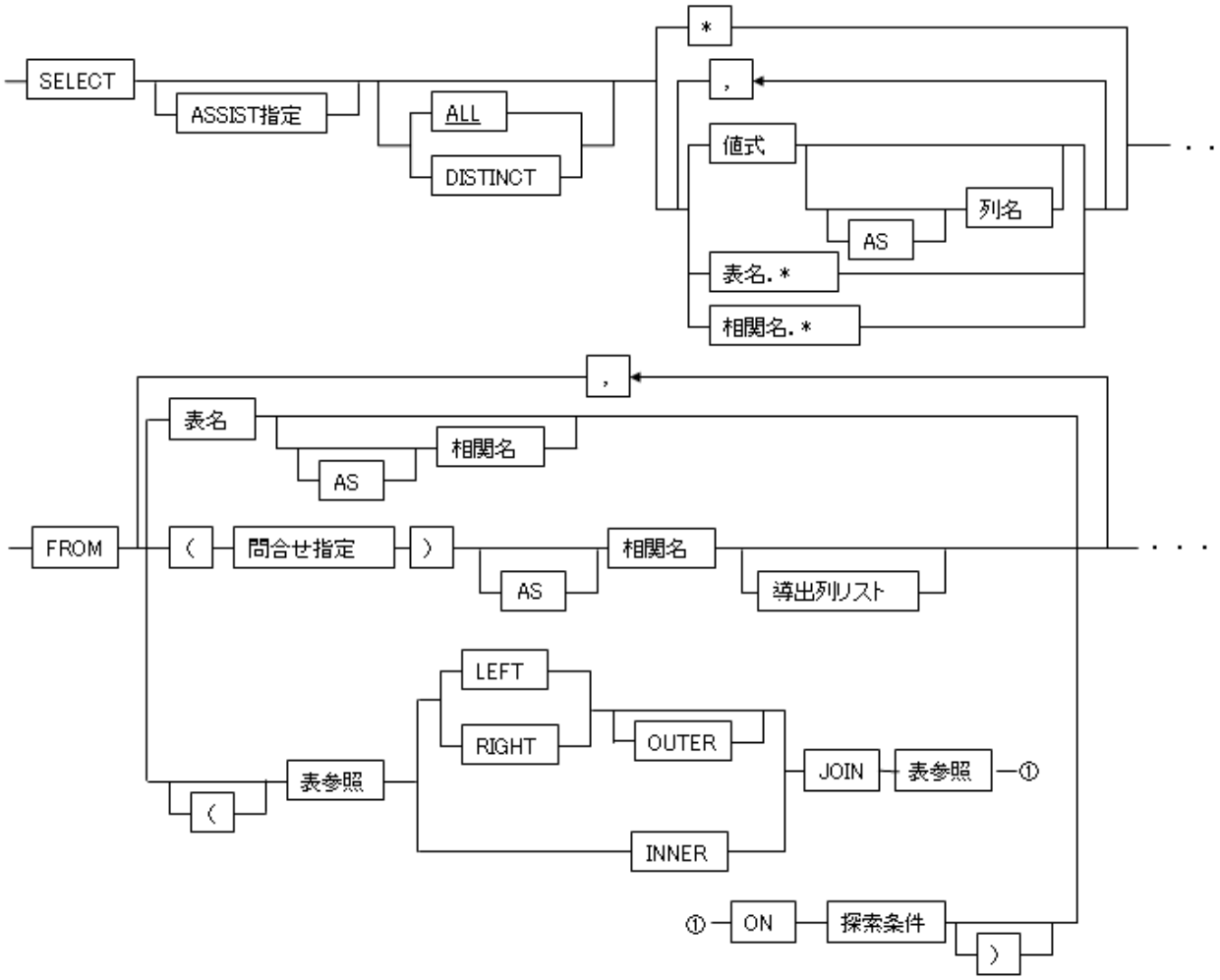
表に新しい行を挿入します。

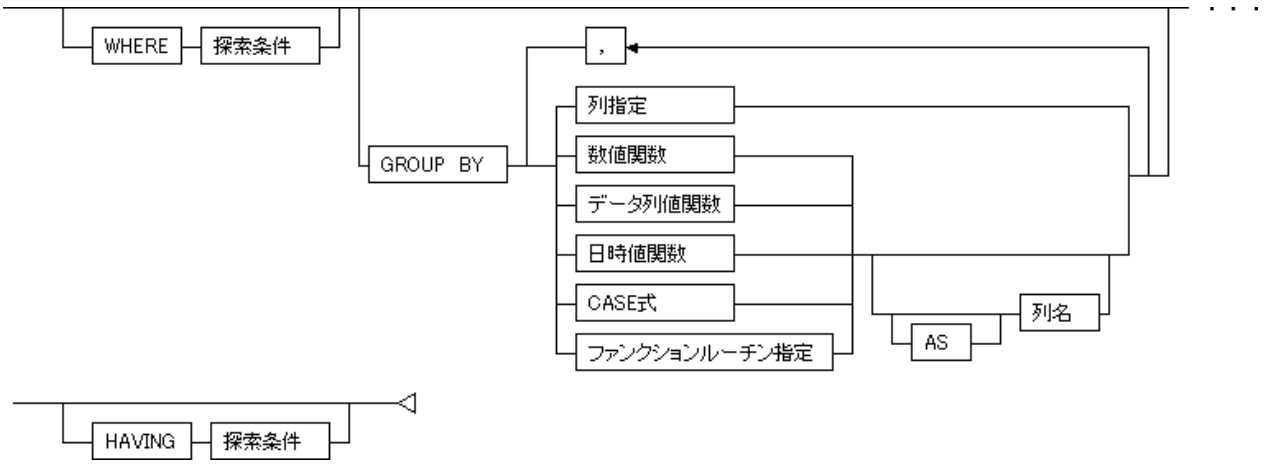
記述形式



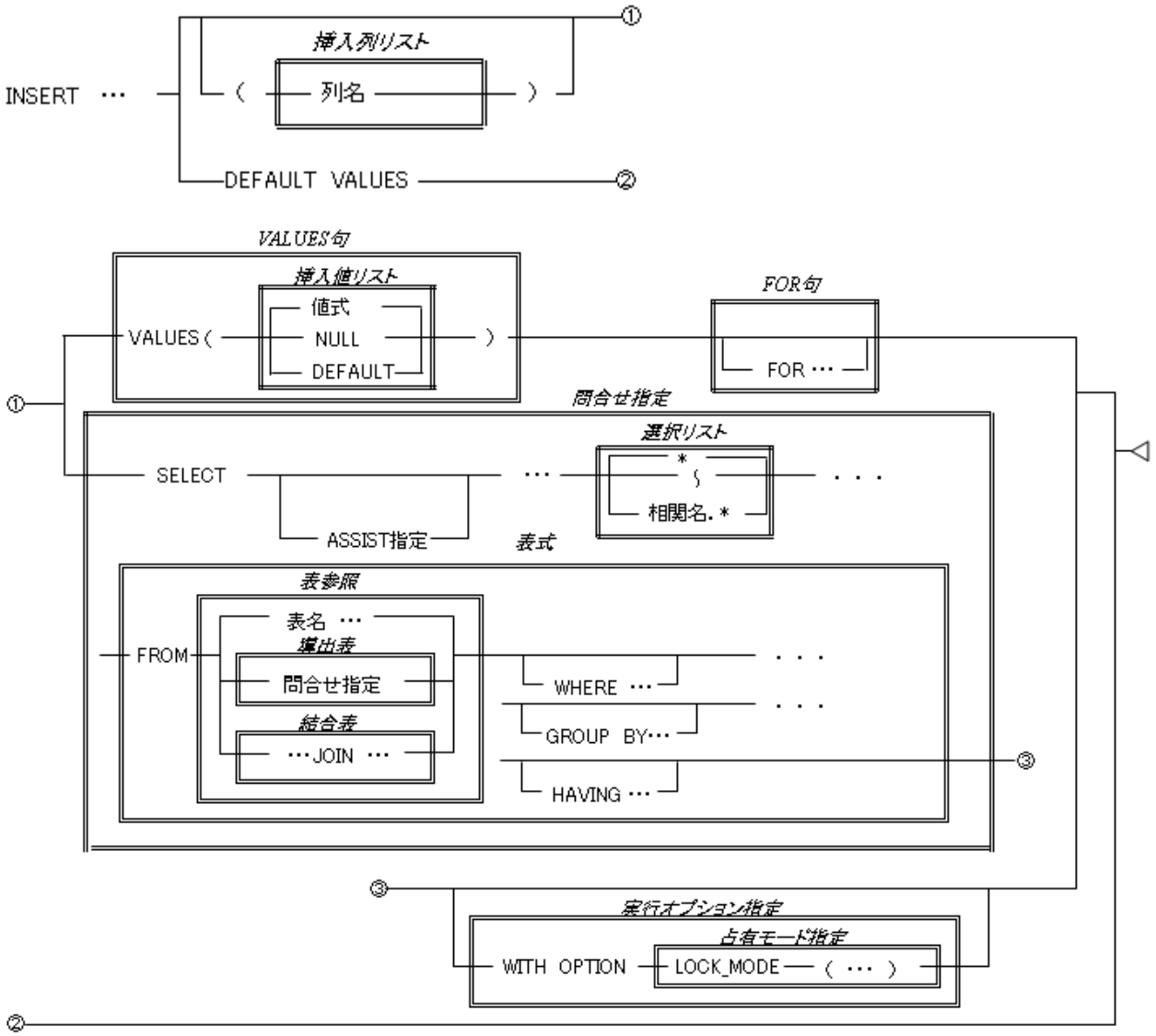

②
④⑤
③
WITH OPTION
LOCK_MODE
(
EXCLUSIVE LOCK
EX
SHARE LOCK
SH
FREE LOCK
FL
NO LOCK
NL
)
DEFAULT VALUES

問合せ指定


SELECT
ASSIST指定
ALL
DISTINCT
*
,
値式
AS
列名
表名.*
相関名.*
,
表名
AS
相関名
FROM
(
問合せ指定
)
AS
相関名
導出列リスト
LEFT
RIGHT
OUTER
INNER
JOIN
表参照
①
(
表参照
①
ON
探索条件
)



構文の構成



参照項番

- 値式 → “2.11 値式”
- 問合せ指定 →“3.26 DECLARE CURSOR(カーソル宣言)”

- ASSIST指定 → “2.15 ASSIST指定”
- 単純値指定 → “2.3 値指定と相手指定”
- 値指定 → “2.3 値指定と相手指定”
- 列指定 → “2.4 列指定”
- 数値関数 → “2.5.2 数値関数”
- データ列値関数 → “2.5.3 データ列値関数”
- 日時値関数 → “2.5.4 日時値関数”
- CASE式 → “2.6 CASE式”
- 順序 → “2.8 順序”
- ファンクションルーチン指定→ “2.5.5 ファンクションルーチン指定”

権限

- INSERT文を実行できるのは、処理対象の表のINSERT権の保持者です。ただし、問合せ指定が指定された場合、その問合せ指定に指定された表のSELECT権が必要です。

一般規則

- INSERT文にROWNUMは指定できません。

表名

— 行を挿入する表の名前を指定します。

— 表名で指定した表を、問合せ指定のFROM句に指定することはできません。表名がビュー表の場合、そのビュー表に含まれる表が対象になります。

— 表名で指定した表に、トリガ事象にINSERTを指定したトリガを定義している場合、被トリガSQL文中に指定した表名と同じ表名を副問合せのFROM句に指定することはできません。表名がビュー表の場合、そのビュー表に含まれる表が対象になります。

挿入列リスト

— 挿入列リストは、表名で指定した表の列を指定します。同じ列を複数回指定することはできません。

— 挿入列リストを省略した場合、表のすべての列が指定されたとみなされます。

— 挿入列リストの個数と、挿入値リストの個数または問合せ指定の選択リストの個数は、同じであることが必要です。

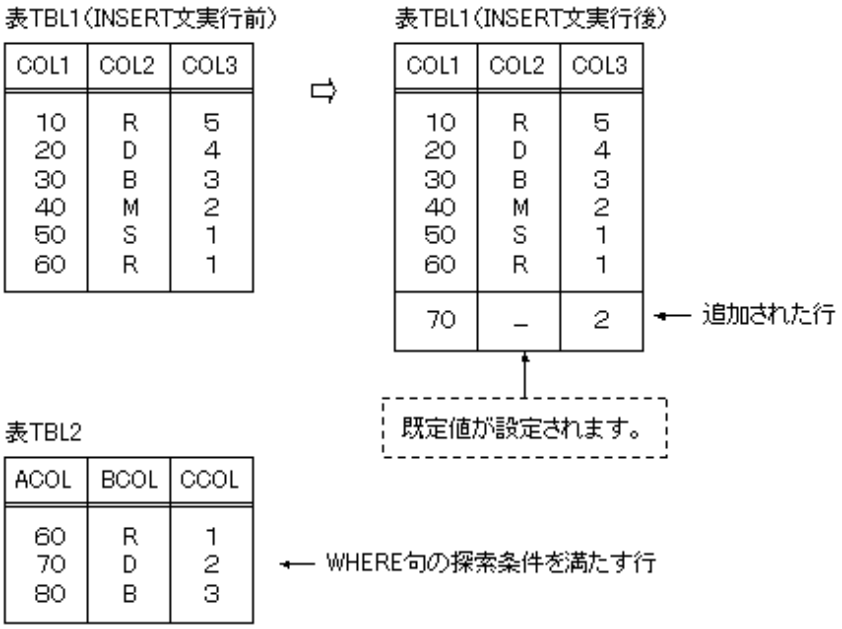
— 挿入列リストが指定された場合、挿入列リストに指定されていない列には、その列の定義で指定されたDEFAULT句の値が設定されます。列の定義でDEFAULT句が指定されていない場合には、NULL値が設定されます。

例

挿入列リストを指定する例を示します。

```
INSERT INTO TBL1(COL1,COL3)
       SELECT ACOL,CCOL
       FROM TBL2
       WHERE ACOL = 70
```

表TBL2の行のうち、WHERE句の条件を満たす2番目の行のACOL、CCOLの値が、表TBL1の指定された列COL1、COL3に挿入されます。このとき指定されなかった列COL2には表定義で指定された既定値が設定されます。もし、表定義でDEFAULT句が指定されていなかった場合は、NULL値が設定されます。ただし、列COL2がNULL値を許さない列である場合はエラーとなります。



VALUES句

- VALUES句を指定した場合、挿入値リストの値が入った1行を表に挿入します。

挿入値リスト

- NULLを指定した場合、NULL値が設定されます。
- DEFAULTを指定した場合、その列の定義で指定されたDEFAULT句の値が設定されます。列の定義でDEFAULT句が指定されていない場合には、NULL値が設定されます。
- 挿入値リストの値式にCURRENT_DATE、CURRENT_TIME、CURRENT_TIMESTAMPを指定した場合、それぞれ現在の日付、現在の時刻、現在の時刻印が設定されます。
- 挿入値リストの値式にCURRENT_DATEを指定した場合、挿入する値のデータ型は、DATE型です。
- 挿入値リストの値式にCURRENT_TIMEを指定した場合、挿入する値のデータ型は、TIME型です。
- 挿入値リストの値式にCURRENT_TIMESTAMPを指定した場合、挿入する値のデータ型は、TIMESTAMP型です。
- 挿入値リストに指定する挿入値のデータ型または問合せ指定で示される各列のデータ型は、挿入する表の対応する各列のデータ型にそれぞれ代入可能であることが必要です。代入可能な条件を以下に示します。

  NULLを指定する場合:

  対応する列がNULL値を許していることが必要です。

  列のデータ型が文字列型の場合:

  挿入する値のデータ型は、文字列型であることが必要です。

  列のデータ型が各国語文字列型の場合:

  挿入する値のデータ型は、各国語文字列型であることが必要です。

  列のデータ型が真数型の場合:

  指定される列のデータ型が真数型ならば、挿入する値のデータ型は、真数型または概数型であることが必要です。

  列のデータ型が概数型の場合:

  指定される列のデータ型が概数型ならば、挿入する値のデータ型は、真数型または概数型であることが必要です。

  列のデータ型がDATE型の場合:

  挿入する値のデータ型は、DATE型であることが必要です。ただし、挿入値リストを変数指定で指定する場合は文字列型で指定することが必要です。

列のデータ型がTIME型の場合:

挿入する値のデータ型は、TIME型であることが必要です。ただし、挿入値リストを変数指定で指定する場合は文字列型で指定することが必要です。

列のデータ型がTIMESTAMP型の場合:

挿入する値のデータ型は、TIMESTAMP型であることが必要です。ただし、挿入値リストを変数指定で指定する場合は文字列型で指定することが必要です。

列のデータ型が年月クラスの時間隔型の場合:

挿入する値のデータ型は、年月クラスの時間隔型であることが必要です。ただし、挿入値リストを変数指定で指定する場合は、文字列型または位取り0の真数型で指定することが必要です。位取り0の真数型は、列のデータ型が単一日時フィールドで指定された時間隔の場合のみ指定できます。

列のデータ型が日時クラスの時間隔型の場合:

挿入する値のデータ型は、日時クラスの時間隔型であることが必要です。ただし、挿入値リストを変数指定で指定する場合は、文字列型または位取り0の真数型で指定することが必要です。位取り0の真数型は、列のデータ型が単一日時フィールドで指定された時間隔の場合のみ指定できます。

列のデータ型がBLOB型の場合:

挿入する値のデータ型は、BLOB型であることが必要です。

- 表名で指定した表が“WITH CHECK OPTION”を指定したビュー表の場合、挿入する値はビュー表の探索条件に対して真でなければなりません。そうでなければ、例外(WITH CHECK OPTIONに違反)となります。
- 挿入値が対象列に設定されるときの規則は次のようになります。

文字列型データの場合:

挿入値の右側の空白を除くデータ長が対象列の長さと同じ場合は、文字データがそのまま設定されます。挿入値の右側の空白を除くデータ長が対象列の長さよりも短い場合は、挿入値が対象列の左側から設定され、対象列の残り部分には空白が設定されます。挿入値の右側の空白を除くデータ長が対象列の長さよりも長い場合は、データ例外(文字データの右トランケート)となります。

各国語文字列型の場合:

文字列型の場合と同じように設定されます。

真数型の場合:

対象列のデータ型に変換して設定されます。挿入値の整数部が対象列の精度と位取りで表現できる場合は、対象列の精度と位取りに従って設定されます。上位の桁落ちが発生する場合は例外となり、下位の位が落ちる場合は切り捨てられます。

概数型のデータの場合:

対象列には、挿入値の概数値が設定されます。

日時型のデータの場合:

対象列のデータ型に変換して設定されます。

時間隔型のデータの場合:

対象列のデータ型に変換して設定されます。ただし、時間隔型が年月の場合は年月のデータ型に、時間隔型が日時の場合は日時のデータ型にそれぞれ設定されます。

BLOB型のデータの場合:

対象列には、挿入値のバイナリ値が設定されます。

- INSERT文を被トリガSQL文に指定する場合のみ、挿入値リストの挿入値に列指定を指定することができます。
- 挿入値リストに指定する値式に集合関数指定を指定することはできません。
- 挿入値リストに指定する値式にROW_IDを指定することはできません。

FOR句

複数行のデータを一括挿入する場合に、挿入する行数を指定します。

SQL埋込みCプログラムの場合

- VALUES句の挿入値には、ホスト変数定義で配列指定された構造体変数またはポインタ指定された構造体変数を1つ指定します。
- VALUES句に指定した構造体変数の配列要素数と同じ値をFOR句に指定してください。
- FOR句に指定する単純値指定のSQLデータ型は、INTEGERまたはSMALLINTとしてください。
- FOR句を指定する場合、挿入列リストの列名にBLOB型の列は指定できません。
- ホスト変数定義で配列指定された構造体変数またはポインタ指定された構造体変数に標識変数は指定できません。
- FOR句を用いたINSERT文は準備可能文としては利用できません。

SQL埋込みCOBOLプログラムの場合

- VALUES句の挿入値には、集団項目繰り返しホスト変数を1つ指定します。
- VALUES句に指定した集団項目繰り返しホスト変数の配列要素数と同じ値をFOR句に指定してください。
- FOR句に指定する単純値指定のSQLデータ型は、INTEGERまたはSMALLINTとしてください。
- FOR句を指定する場合、挿入列リストの列名にBLOB型の列は指定できません。
- 集団項目繰り返しホスト変数に標識変数は指定できません。
- FOR句を用いたINSERT文は準備可能文としては利用できません。

DESCRIBE情報について

― 挿入値リストに動的パラメタ指定が指定された場合のDESCRIBE情報は、対応する列のデータ型になります。

問合せ指定

― 問合せ指定を指定した場合、問合せ指定の結果の表すべての行を表に挿入します。

― 問合せ指定の選択リストに、格納構造がOBJECTのBLOB型の列を指定することはできません。

― 問合せ指定の導出表にORDER BY句を指定することはできません。

― 問合せ指定の結果が空ならば、例外(データなし)になります。

― 挿入する表がOBJECT構造の場合、問合せ指定を指定することはできません。

実行オプション指定

― 実行オプション指定は、占有モードを指定します。

占有モード指定

― 占有モード指定は、問合せ指定を指定した場合に、問合せ指定のデータベース資源の占有の方法を指定します。

― 占有モードを指定したSQL文が読み込んだデータベース資源は、占有モード指定により、以下のようにデータベース資源を占有します。なお、更新する行は占有モード指定にかかわらず、非共有モードでトランザクションの終了までデータベース資源を占有します。

占有モード指定	占有の方法
EXCLUSIVE LOCK	非共有モードでトランザクション終了までデータベース資源を占有します。
SHARE LOCK	共有モードでトランザクション終了までデータベース資源を占有します。
FREE LOCK	共有モードでSQL文終了までデータベース資源を占有します。
NO LOCK	データベース資源を占有しません。

- 占有モードを指定したSQL文によって読み込まれた資源は、SET TRANSACTION文で指定された内容にかかわらず、以下のようになります。

占有モード指定	読込み水準
EXCLUSIVE LOCK	他のトランザクションによって占有されていない行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
SHARE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
FREE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。当該SQL文で読み込んだ行は、他のトランザクションに更新されることがあるため、同一トランザクションで再検索すると最新の結果を検索することができます。
NO LOCK	他のトランザクションでどのような占有をされた資源でも参照することが可能なため、同一トランザクションで再検索すると最新の結果を検索することができます。

- 占有モードの単位は、R_LOCKがYESの場合は行単位となります。R_LOCKがNOの場合は、ページ単位またはDSI単位の占有になります。
- 動作環境パラメタにDSO_LOCKを指定したり、環境変数でRDBDSOを指定した場合、占有モードを指定したSQL文は実行できません。

DEFAULT VALUES

- DEFAULT VALUESが指定された場合、すべての列には、その列の定義で指定されたDEFAULT句の値が設定されます。列の定義でDEFAULT句が指定されていない場合には、NULL値が設定されます。

その他の構文要素

その他の構文要素の説明は、“3.26 DECLARE CURSOR(カーソル宣言)”を参照してください。

使用例

例1

挿入する値として問合せ指定を使用する例を示します。

```
INSERT INTO TBL1
       SELECT *
       FROM TBL2
       WHERE ACOL = 60
```

表TBL2の行のうちWHERE句の探索条件を満たす行が、表TBL1に挿入されます。問合せ指定の選択リストに“*”が指定されると、FROM句で指定された表のすべての列が指定されたとみなされます。

表TBL1(INSERT文実行前)

COL1	COL2	COL3
10	R	5
20	D	4
30	B	3
40	M	2
50	S	1

⇨

表TBL1(INSERT文実行後)

COL1	COL2	COL3	
10	R	5	
20	D	4	
30	B	3	
40	M	2	
50	S	1	
60	R	1	← 追加された行

表TBL2

ACOL	BCOL	CCOL	
60	R	1	← WHERE句の探索条件を満たす行
70	D	2	
80	B	3	

例2

挿入値に集団項目繰り返しホスト変数を指定する例を示します。

SQL埋込みCOBOLプログラム

```
              :
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01  SQLSTATE     PIC X(5).
01  SQLMSG       PIC X(255).
01  G1.
  02 RVAL    OCCURS 10.
    03 HOST1    PIC S9(4) BINARY.
    03 HOST2    PIC S9(4) BINARY.
    03 HOST3    PIC X(10).
    EXEC SQL END DECLARE SECTION END-EXEC.
PROCEDURE DIVISION.

  COMPUTE HOST1 OF RVAL OF G1 (1) = 1.
  COMPUTE HOST1 OF RVAL OF G1 (2) = 2.
              :
  COMPUTE HOST2 OF RVAL OF G1 (1) = 10.
  COMPUTE HOST2 OF RVAL OF G1 (2) = 20.
              :
  MOVE "A" TO HOST3 OF RVAL OF G1(1).
  MOVE "B" TO HOST3 OF RVAL OF G1(2).
              :
  EXEC SQL INSERT INTO S1.T1 VALUES (:G1.RVAL) FOR 10 END-EXEC.
              :
```

例3

SQL文の実行中のみ参照資源を共用モードで占有する例を示します。

```
INSERT INTO 在庫管理.在庫表 SELECT * FROM 在庫管理.発注表
       WHERE COL1 = 10
       WITH OPTION LOCK_MODE(FREE LOCK)
```

例4

COMMITされていない更新中のデータを参照して挿入する例を示します。

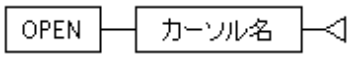
```
INSERT INTO 在庫管理.在庫表 SELECT * FROM 在庫管理.発注表
        WHERE COL1 = 10
        WITH OPTION LOCK_MODE(NO LOCK)
```

3.49 OPEN文

機能

カーソルを開くことによってカーソルを有効な状態にします。

記述形式



参照項番

- 日本語文字列 → “2.1.3 トークン”

権限

- OPEN文を実行できるのは、カーソルの処理対象の表に対するSELECT権の保持者です。

一般規則

- カーソルは、閉じられた状態であることが必要です。開かれた状態であれば、例外(不当カーソル状態)となります。
- OPEN直後のカーソルは、先頭行の直前に位置づけられます。
- OPEN文を実行できるのは、処理対象の表のSELECT権の保持者です。
- カーソル宣言にFOR READ ONLYを指定し、SET TRANSACTION文にREAD COMMITTEDが指定された場合、32K以上のBLOB型の項目を参照するカーソルは指定できません。

カーソル名

- ー カーソルの名前を指定します。
- ー カーソル名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- ー カーソル名は、同一コンパイル単位に含まれるカーソル宣言で定義されていることが必要です。

使用例

例

カーソル“CUR1”を開きます。

```
OPEN CUR1
```

3.50 PRINT STATISTICS文

機能

現在設定されている最適化情報をファイルに出力します。

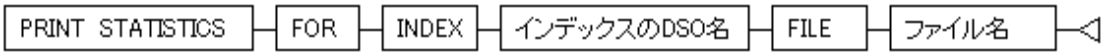
データベース簡単運用の場合は利用できません。

記述形式

DSI指定の場合

表指定の場合

インデックスのDSO指定の場合



一般規則

DSI名

－ 最適化情報を出力するDSI名を指定します。

表名

－ 最適化情報を出力する表名を指定します。

インデックスのDSO名

－ 最適化情報を出力するインデックスのDSO名を指定します。

ファイル名

－ 最適化情報の出力先のファイル名を指定します。ファイル名に指定できる長さは、256バイト以内です。

参照

ファイルの出力形式については、“RDB運用ガイド(データベース定義編)”を参照してください。

使用例

例1

関東発注表DSIの最適化情報を出力します。表のデータ構造は、SEQUENTIALです。

S L

Solaris/Linuxの場合

```
PRINT STATISTICS FOR DSI 関東発注表ＤＳＩ
    FILE /rdb2/statistics/DB01/関東発注表ＤＳＩ
```

W

Windowsの場合

```
PRINT STATISTICS FOR DSI 関東発注表ＤＳＩ
    FILE c:¥rdb¥data¥関東発注表ＤＳＩ.txt
```

出力例

```
DATABASE      = DB01

DSI           = 関東発注表ＤＳＩ
RECORD        = 30000000
PAGE          = 150000
```

例2

製品名IXDSIのインデックスDSIの最適化情報を出力します。

S L

Solaris/Linuxの場合

```
PRINT STATISTICS FOR DSI 製品名ＩＸＤＳＩ
    FILE /rdb2/statistics/DB01/製品名ＩＸＤＳＩ
```

W

Windowsの場合

```
PRINT STATISTICS FOR DSI 製品名ＩＸＤＳＩ
    FILE c:¥rdb¥data¥製品名ＩＸＤＳＩ.txt
```

出力例

```
DATABASE      = DB01

DSI           = 製品名ＩＸＤＳＩ
PAGE          = 5
INDEX HEIGHT  = 2
DIFFERENT KEY = 150
```

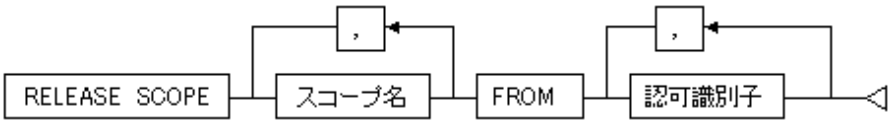
3.51 RELEASE SCOPE文(スコープ解除文)

機能

適用されたスコープを解除します。

データベース簡単運用の場合は利用できません。

記述形式



権限

- スコープを解除できるのは、スコープの定義者のみです。

一般規則

- 指定されたスコープは、指定された認可識別子に対して適用されていることが必要です。

スコープ名

ー 解除するスコープの名前を指定します。

ー 同じスコープ名を複数指定できません。

認可識別子

ー スコープの適用を解除する利用者のログイン名を指定します。

使用例

例

利用者YAMADAに適用されたスコープ“東京ＳＣＰ”を解除します。

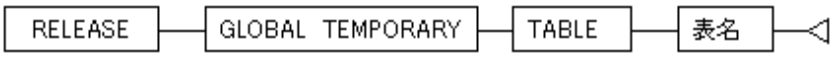
```
RELEASE SCOPE 東京ＳＣＰ FROM YAMADA
```

3.52 RELEASE TABLE文

機能

一時表に格納したデータを削除し、一時表の領域を解放します。

記述形式



権限

・ RELEASE TABLE文を実行できるのは、処理対象の表のDELETE権の保持者です。

一般規則

・ トランザクションは、終了していることが必要です。

表名

ー 解放する一時表の名前を指定します。

使用例

例

一時表S1.T1を解放します。

```
RELEASE GLOBAL TEMPORARY TABLE S1.T1
```

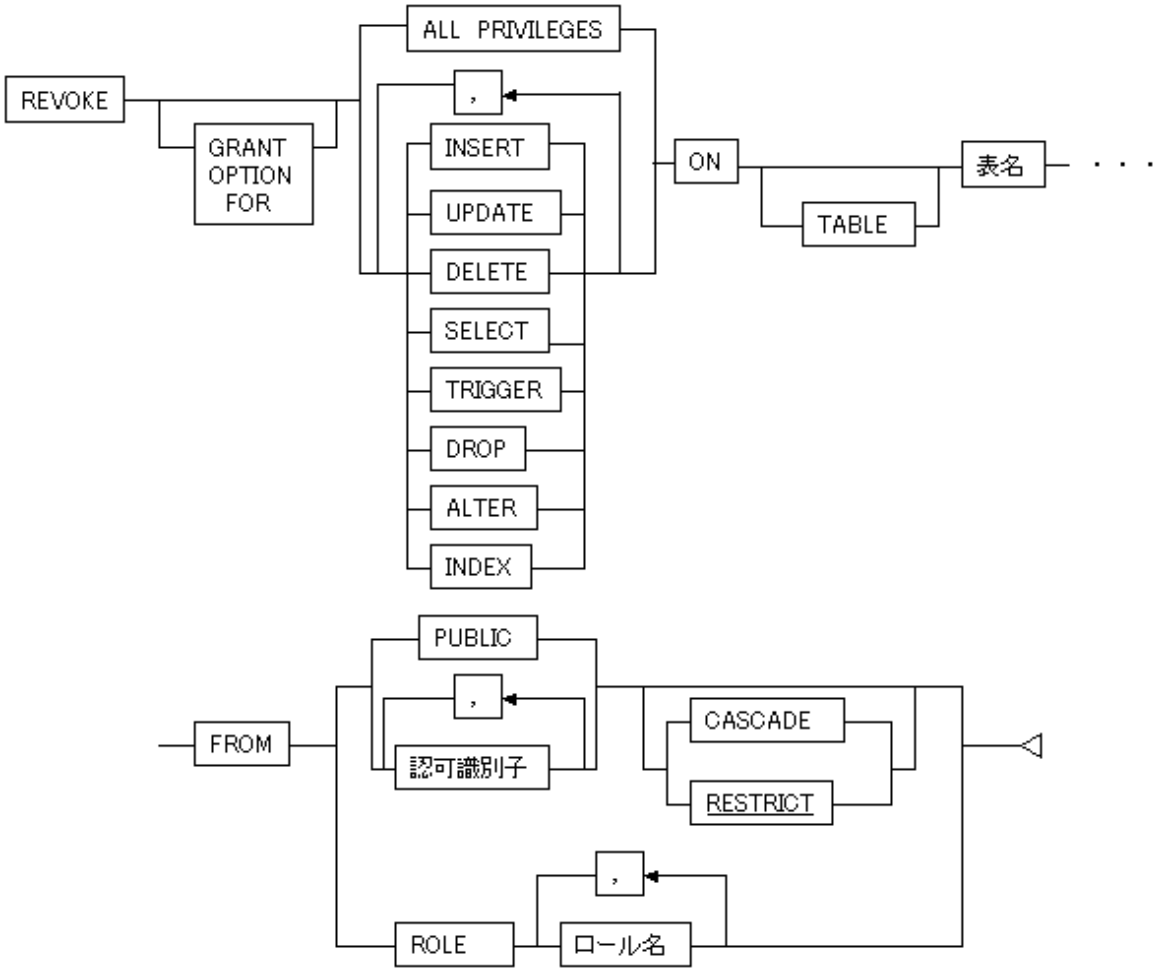
3.53 REVOKE文

機能

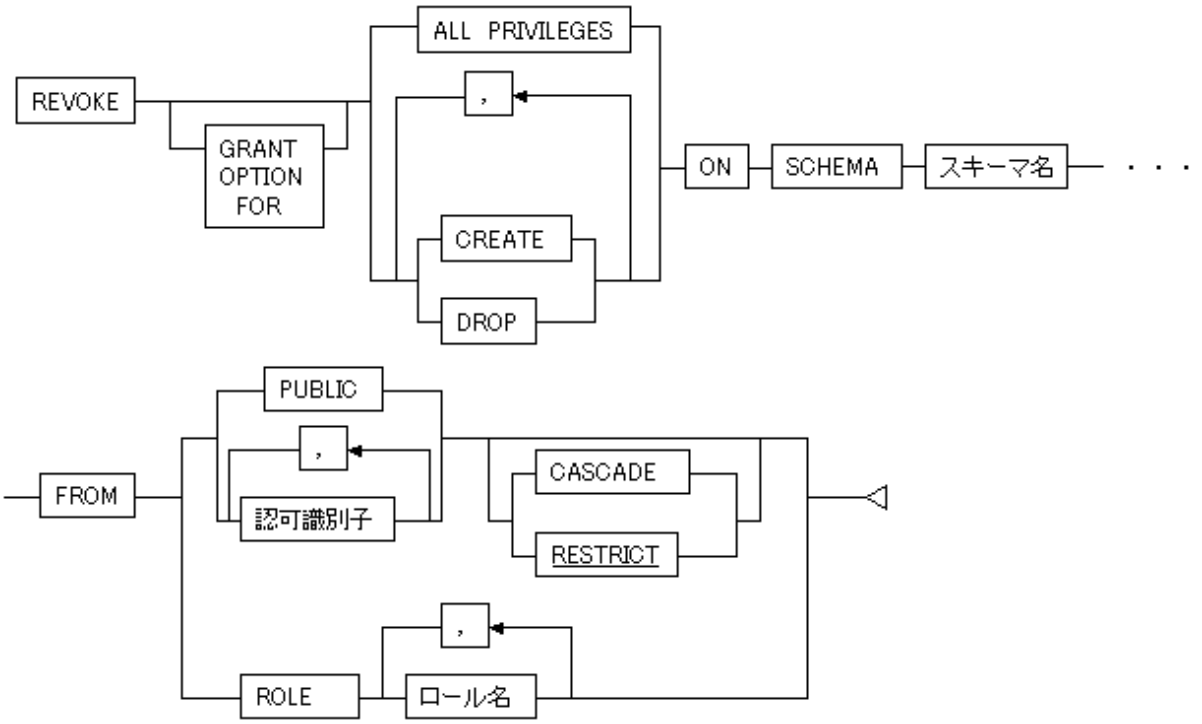
権限を削除します。

記述形式

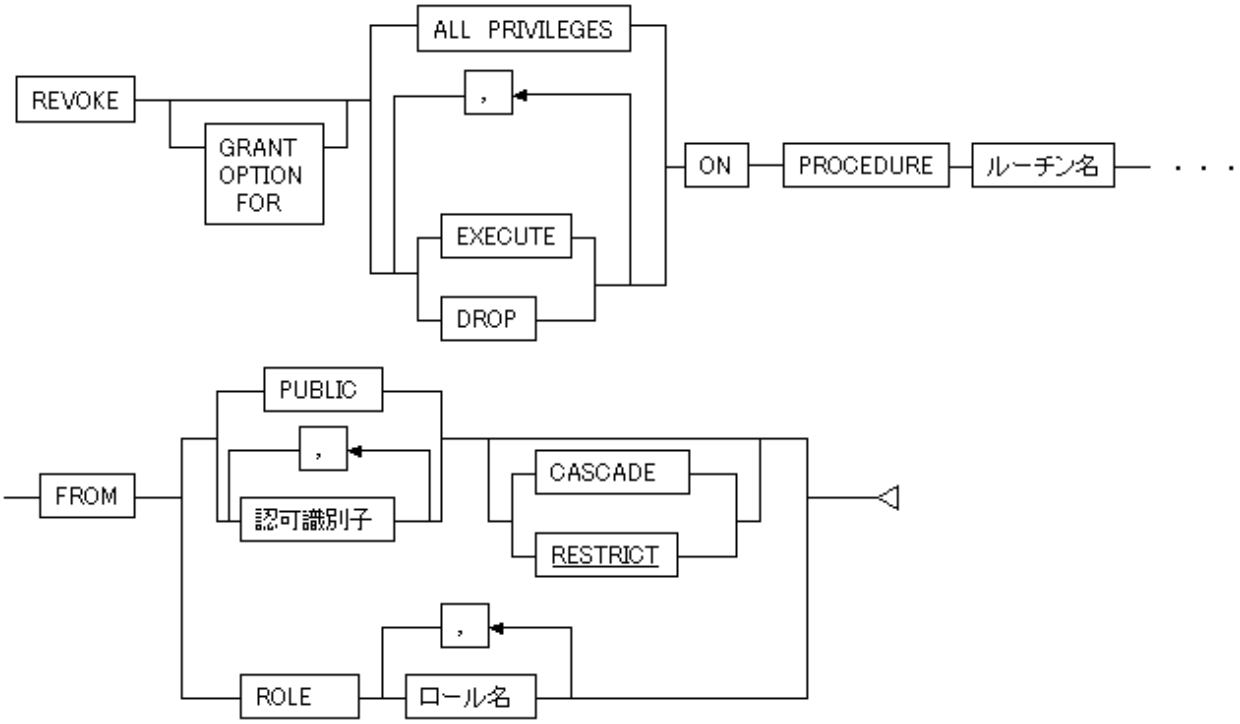
表に対する権限の削除またはロールから権限を削除



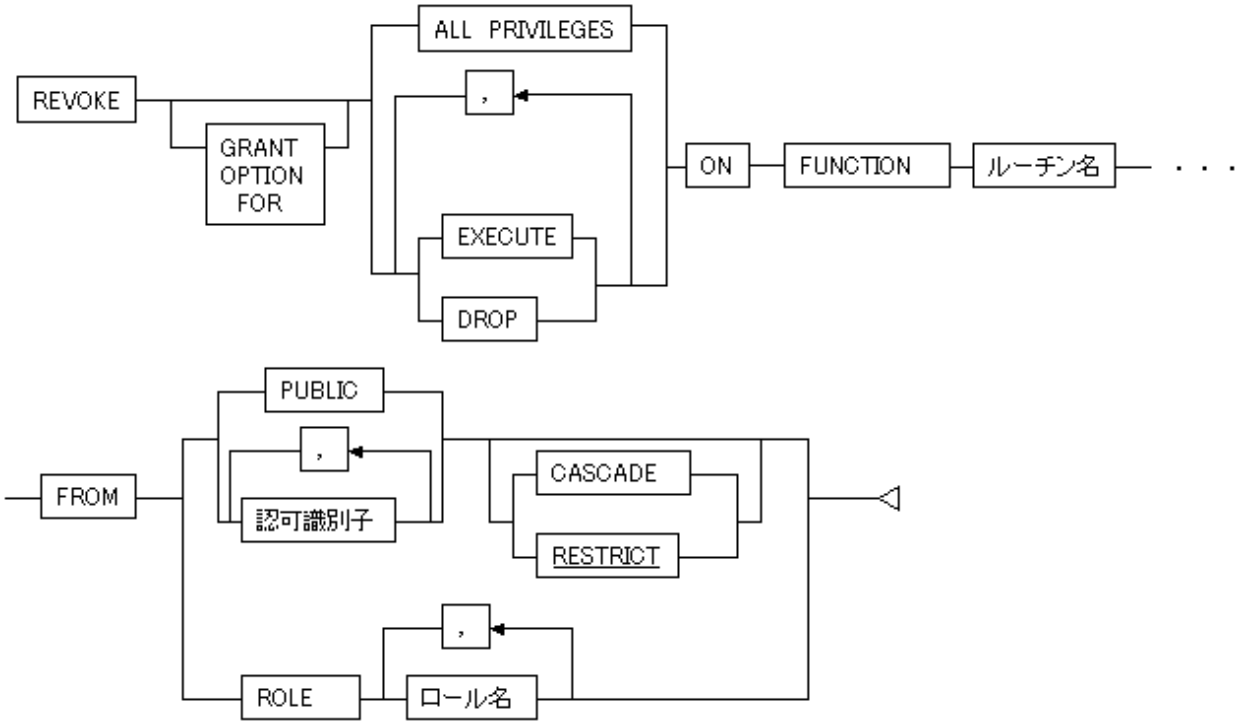
スキーマに対する権限の削除またはロールから権限を削除



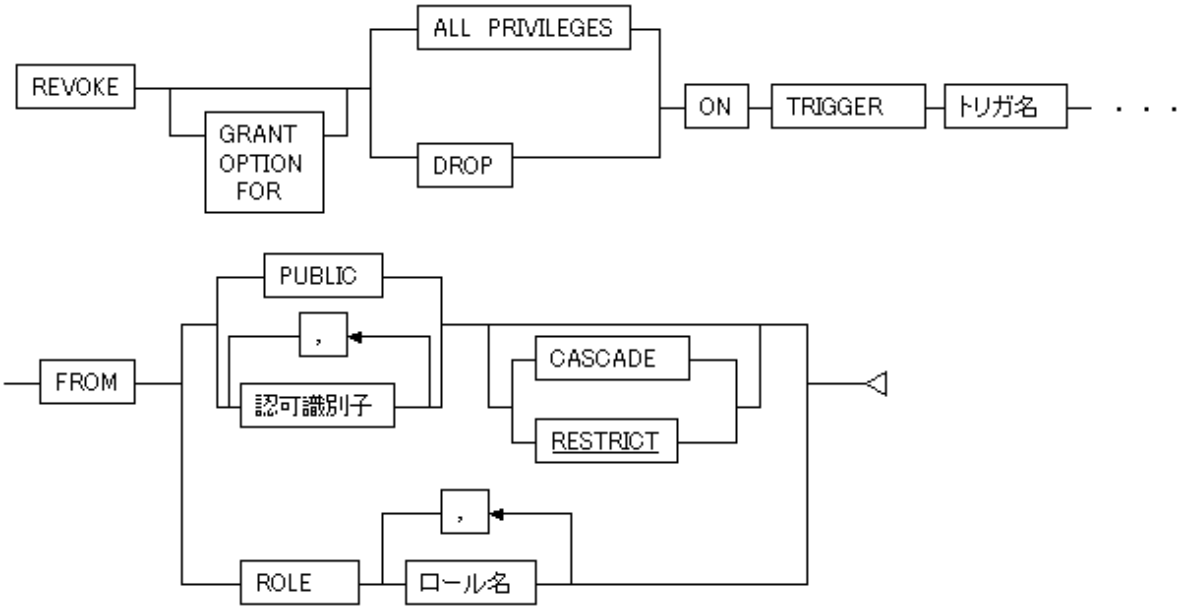
プロシジャルーチンに対する権限の削除またはロールから権限を削除



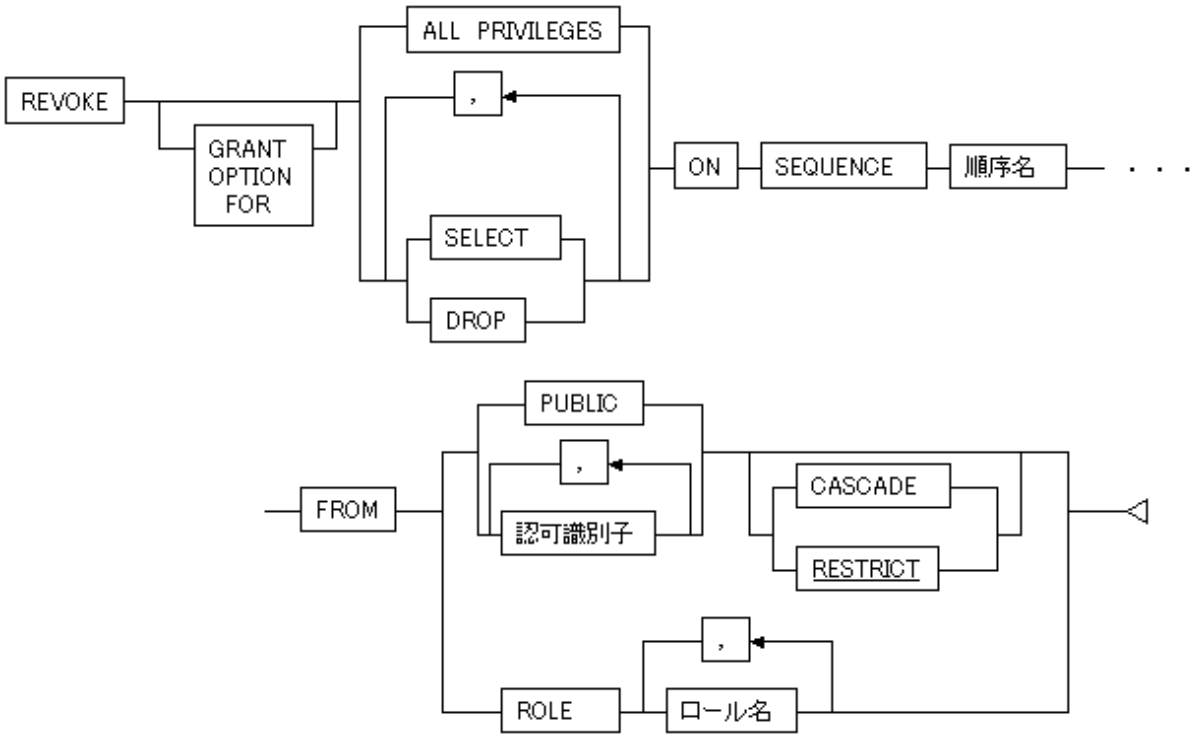
ファンクションルーチンに対する権限の削除またはロールから権限を削除



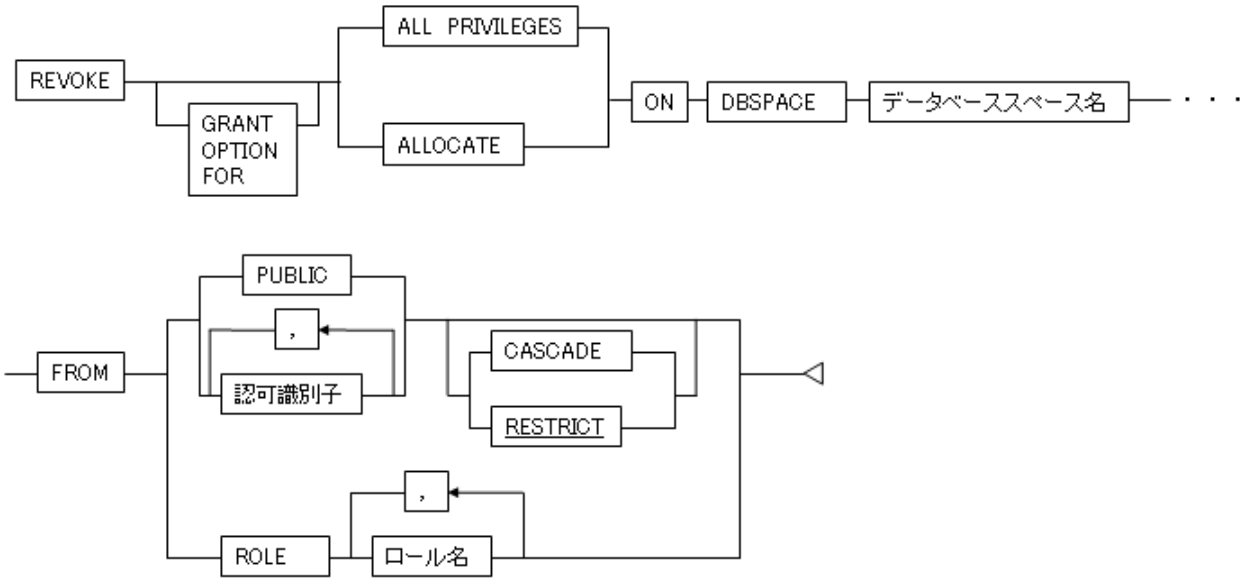
トリガに対する権限の削除またはロールから権限を削除



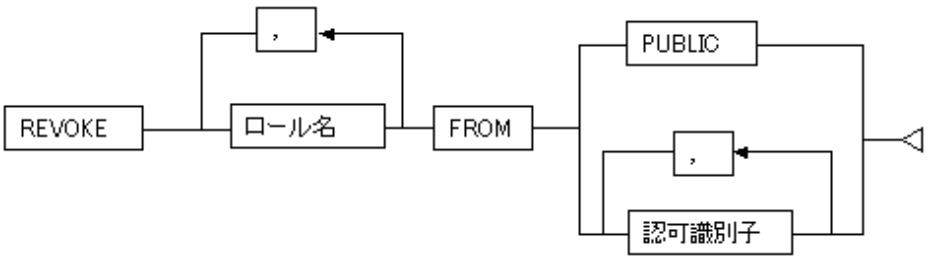
順序に対する権限の削除またはロールから権限を削除



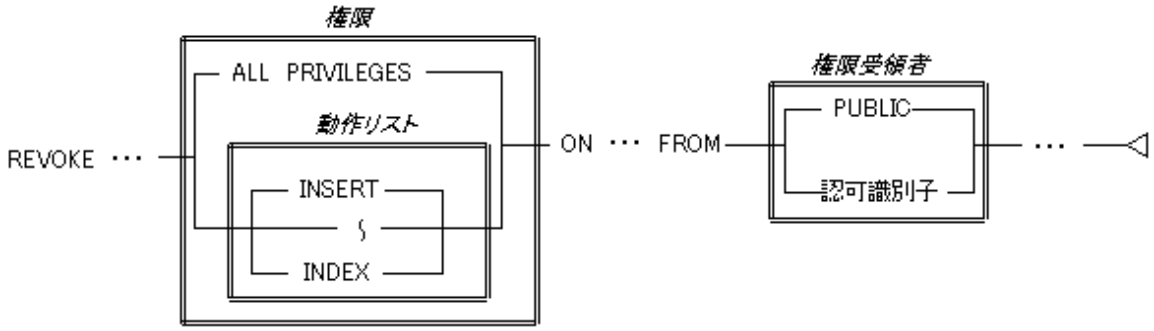
データベーススペースに対する権限の削除またはロールから権限を削除



ロールを他の利用者から削除



構文の構成



一般規則

GRANT OPTION FOR

- GRANT OPTION FORが指定された場合は、付与権のみが削除されます。
- ROLEの場合は指定できません。

ALL PRIVILEGES

- ALL PRIVILEGESが指定された場合、REVOKE文実行者から付与されたすべての権限が削除されます。

動作リスト

動作リストに指定する権限の一般規則を以下に示します。

INSERT

- 権限受領者に対して表に行を挿入する権限(INSERT権)が削除されます。

UPDATE

- 権限受領者に対して表の行を更新する権限(UPDATE権)が削除されます。

DELETE

- 権限受領者に対して表の行を削除する権限(DELETE権)が削除されます。

SELECT

- 権限受領者に対して表の行または順序を参照する権限(SELECT権)が削除されます。

EXECUTE

- 権限受領者に対してプロシジャルーチンまたはファンクションルーチンを実行する権限(EXECUTE権)が削除されます。

CREATE

- 権限受領者に対してスキーマや表などの資源を定義する権限(CREATE権)が削除されます。

ALLOCATE

- 権限受領者に対してデータベーススペースに対して領域を割り当てる権限(ALLOCATE権)が削除されます。

TRIGGER

- 権限受領者に対してその表にトリガを定義する権限(TRIGGER権)が削除されます。

DROP

- 権限受領者に対してスキーマや表などの資源を削除する権限(DROP権)が削除されます。

ALTER

- 権限受領者に対して表定義を更新する権限(ALTER権)が削除されます。

INDEX

- 権限受領者に対して表にインデックスを定義する権限(INDEX権)が削除されます。

対象名

対象名は、権限の対象となる資源の名前です。

表名

- 権限を削除する表の名前を指定します。

ルーチン名

- 権限を削除するプロシジャルーチンまたはファンクションルーチンの名前を指定します。

スキーマ名

- 権限を削除するスキーマの名前を指定します。

データベーススペース名

- 権限を削除するデータベーススペースの名前を指定します。

トリガ名

- 権限を削除するトリガの名前を指定します。

順序名

- 権限を削除する順序の名前を指定します。

ロール名(ロールを他の利用者から削除する場合に指定)

- ロール名は、権限受領者から削除するロールの名前を指定します。
- 同じロール名を複数指定することはできません。
- 実行者は、ロールの付与権を保持していなければなりません。

権限受領者

PUBLIC

- PUBLICは、データベースをアクセスするすべての人を意味します。

PUBLICとは、暗黙に定義された「データベースをアクセスするすべての利用者の集合」を意味します。

個々の利用者は、認可識別子を指定して直接許可された権限と、PUBLICを指定して許可された権限を合わせて受領していることに注意してください。

したがって、たとえば、PUBLICからSELECT権を剥奪することは、必ずしもその資源に対するSELECT権をすべての利用者が失うことを意味しません。直接権限を付与された人は、その権限を持ち続けます。

認可識別子

- 権限を削除する認可識別子を指定します。
- 認可識別子は、18文字以内の先頭が英字で始まる英数字、または9文字以内の日本語文字列を指定します。
- 英小文字の区切り識別子を認可識別子に指定することはできません。

ロール名(ロールから権限を削除する場合に指定)

- ロール名は、権限を削除するロールの名前を指定します。
- 同じロール名を複数指定できません。

CASCADEおよびRESTRICT(削除動作)

CASCADEおよびRESTRICTは、“削除動作”と呼びます。

CASCADE

- CASCADEを指定した場合、付与した権限が削除されます。権限受領者が定義したビュー表、プロシジャルーチンおよびトリガに必要な権限を削除すると、これらの資源はすべて削除されます。
- ビューで参照する表に対するINSERT権、DELETE権、UPDATE権が、ビュー表の定義者から剥奪されるとき、ビュー表に対する同じ権限もビュー表の定義者から剥奪されます。

RESTRICT

- RESTRICTを指定した場合、または削除動作を省略した場合、権限受領者が定義した資源に必要な権限を削除することはできません。以下のような場合は、対象となる資源を削除してから、権限を削除することが必要です。
  - 権限受領者が定義したビュー表が参照する表またはビュー表のSELECT権を削除する場合
  - 権限受領者が定義した表またはビュー表が参照する順序のSELECT権を削除する場合
  - 権限受領者が定義したビュー表が参照するファンクションルーチンのEXECUTE権を削除する場合
  - 権限受領者が定義したプロシジャルーチンがSQL手続き文中で指定する表またはビュー表に対する必要な権限を削除する場合
  - 権限受領者が定義したプロシジャルーチンがSQL手続き文中で指定する順序に対するSELECT権を削除する場合
  - 権限受領者が定義したプロシジャルーチンがSQL手続き文中で指定するプロシジャルーチンのEXECUTE権を削除する場合
  - 権限受領者が定義したプロシジャルーチンがSQL手続き文中で指定するファンクションルーチンのEXECUTE権を削除する場合
  - 権限受領者が定義した表のトリガ権を削除する場合
  - 被トリガSQL文で必要な権限を削除する場合
- 権限受領者が他の利用者に付与した権限がある場合、RESTRICTを指定して権限を削除することはできません。削除動作にCASCADEを指定して削除してください。
- RESTRICTを指定した場合、または削除動作を省略した場合で以下の場合は、処理を実行することはできません。
  - ロールから権限を削除することで、ビュー表またはプロシジャルーチンの定義者からこれらの資源に必要な権限がなくなる場合

使用例

例1

在庫表の利用者“YAMADA”から全権限を削除します。

```
REVOKE ALL PRIVILEGES ON STOCKS.在庫表 FROM YAMADA
```

例2

ロール“STOCKS_A2”から権限を削除します。

```
REVOKE SELECT,UPDATE,INSERT ON STOCKS.在庫表 FROM ROLE STOCKS_A2
```

例3

ロール“STOCKS_A2”を他の利用者“SUZUKI”から削除します。

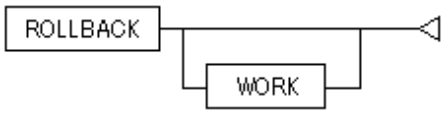
```
REVOKE STOCKS_A2 FROM SUZUKI
```

3.54 ROLLBACK文

機能

現行のデータベースの変更をすべて無効にしてトランザクションを終了します。

記述形式



一般規則

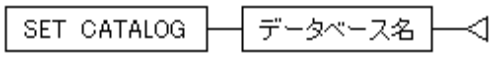
・ オープン中のすべてのカーソルを閉じます。

3.55 SET CATALOG文

機能

動的SQLの被準備文の対象となるデータベース名を変更します。

記述形式



一般規則

・ SET CATALOG文で指定したデータベースの対象は、PREPARE文および、EXECUTE IMMEDIATE文の被準備文です。

・ 変更したデータベース名は、SET CATALOG文を実行したセション内でのみ有効です。

データベース名

ー 新しいデータベースの名前を指定します。データベース名は、文字列定数または文字列型の埋込み変数で指定します。

ー 指定された前後の空白を取り除いた値がデータベース名となります。

使用例

例1

データベース名を“RDBDB2”に変更します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR STMSTR[100];
EXEC SQL END DECLARE SECTION;
strcpy(STMSTR.sqlvar,"SELECT * FROM 在庫表");
STMSTR.sqllen = strlen(STMSTR.sqlvar);
EXEC SQL CONNECT TO 'RDBDB1' USER 'USER1/PASS1';
EXEC SQL PREPARE STM1 FROM :STMSTR;                    (1)
EXEC SQL SET CATALOG 'RDBDB2';                         (2)
EXEC SQL PREPARE STM1 FROM :STMSTR;                    (3)
```

(1) 参照するデータベース名: RDBDB1

(2) データベース名変更

(3) 参照するデータベース名: RDBDB2

例2

データベース名を“RDBDB1”または“RDBDB2”に変更します。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR STMSTR1[100];
     VARCHAR STMSTR2[100];
EXEC SQL END DECLARE SECTION;
strcpy(STMSTR1.sqlvar,"SELECT * FROM 在庫表");
STMSTR1.sqllen = strlen(STMSTR1.sqlvar);
strcpy(STMSTR2.sqlvar,"SELECT * FROM 会社表");
STMSTR2.sqllen = strlen(STMSTR2.sqlvar);
EXEC SQL CONNECT TO 'RDBDB' USER 'USER1/PASS1';
EXEC SQL SET CATALOG 'RDBDB1';
EXEC SQL PREPARE STM1 FROM :STMSTR1;                    (1)
EXEC SQL EXECUTE STM1;                                  (1)
EXEC SQL PREPARE STM2 FROM :STMSTR2;                    (1)
EXEC SQL SET CATALOG 'RDBDB2';
EXEC SQL EXECUTE STM2;                                  (1)
EXEC SQL EXECUTE IMMEDIATE :STMSTR2;                    (2)
```

(1) 対象となるデータベース: RDBDB1

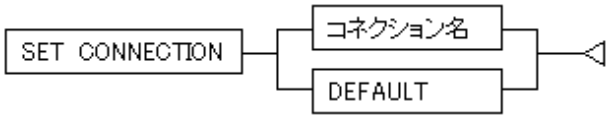
(2) 対象となるデータベース: RDBDB2

3.56 SET CONNECTION文

機能

現コネクションを変更します。

記述形式



一般規則

コネクション名

- コネクションの名前を指定します。コネクション名は、文字列定数または文字列型の埋込み変数で指定します。
- コネクション名を指定した場合、現コネクションが指定したコネクションに変更されます。この場合、同名のコネクション名を指定したCONNECT文により、コネクションが接続状態であることが必要です。
- 変更したコネクション名はセション終了まで有効です。
- コネクション名に空白を含めた場合は、前後の空白を取り除いた値がコネクション名になります。

DEFAULT

- DEFAULTを指定した場合、現コネクションがDEFAULTを指定したCONNECT文で接続されたコネクションに変更されます。この場合、DEFAULTを指定したCONNECT文により、コネクションが接続状態であることが必要です。

使用例

例1

コネクション名に定数を指定します。

```
CONNECT TO 'SV1' AS 'C1' USER 'USER1/PASS1'
CONNECT TO 'SV2' AS 'C2' USER 'USER2/PASS2'
        :
SET CONNECTION 'C1'
```

例2

DEFAULTを指定します。

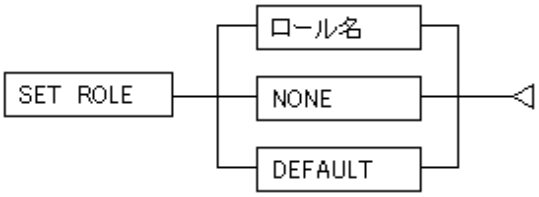
```
CONNECT TO DEFAULT
CONNECT TO 'SV2' AS 'C2' USER 'USER2/PASS2'
        :
SET CONNECTION DEFAULT
```

3.57 SET ROLE文

機能

現行SQLセションに対してロールを有効にします。

記述形式



一般規則

・トランザクションは終了していることが必要です。

ロール名

ー 現行SQLセションに対して有効にするロールの名前を文字列定数または文字列型の埋込み変数で指定します。

ー 実行者はロールの権限受領者でなければなりません。

ー ロール名に空白を含めた場合は、前後の空白を取り除いた値がロール名になります。

NONE

ー NONEが指定されたとき、現行SQLセションで有効となっているロールを無効とします。

DEFAULT

ー DEFAULTが指定されたとき、現行SQLセションでデフォルトロールの設定を有効とします。

使用例

例

アプリケーションの実行中にロール“STOCKS_A2”を有効にします。

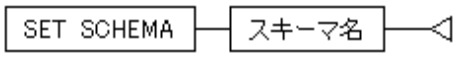
```
SET ROLE 'STOCKS_A2'
```

3.58 SET SCHEMA文

機能

動的SQLの被準備文の省略したスキーマ名を変更します。

記述形式



一般規則

- SET SCHEMA文で指定したスキーマの対象は、PREPARE文およびEXECUTE IMMEDIATE文の被準備文です。
- 変更したスキーマ名は、SET SCHEMA文を実行したセション内でのみ有効です。

スキーマ名

- 新しいスキーマの名前を指定します。スキーマ名は、文字列定数または文字列型の埋込み変数で指定します。
- 指定された前後の空白を取り除いた値が被準備文中の省略したスキーマ名となります。
- データベース名で修飾したスキーマ名が指定できます。データベース名で修飾したスキーマ名が指定された場合、以下のSQL文が暗に実行されます。

```
SET SCHEMA 'RDBDB1.SCH1'
```

上記のSQL文を指定した場合、以下のSQL文を順番に実行したのと同じ意味です。

1. SET CATALOG 'RDBDB1'
2. SET SCHEMA 'SCH1'

使用例

例1

スキーマ名を“SCH1”に変更します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR STMSTR[100];
EXEC SQL END DECLARE SECTION;
strcpy(STMSTR.sqlvar,"SELECT * FROM 在庫表");          (1)
STMSTR.sqllen = strlen(STMSTR.sqlvar);
EXEC SQL CONNECT TO 'RDBDB1' USER 'USER1/PASS1';
EXEC SQL PREPARE STM1 FROM :STMSTR;                    (2)
EXEC SQL SET SCHEMA 'SCH1';                            (3)
EXEC SQL PREPARE STM1 FROM :STMSTR;                    (4)
```

(1) スキーマ名省略

(2) 参照するスキーマ名: USER1

(3) スキーマ名変更

(4) 参照するスキーマ名: SCH1

例2

スキーマ名を“SCH1”または“SCH2”に変更します。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR STMSTR1[100];
     VARCHAR STMSTR2[100];
EXEC SQL END DECLARE SECTION;
strcpy(STMSTR1.sqlvar,"SELECT * FROM 在庫表");
STMSTR1.sqllen = strlen(STMSTR1.sqlvar);
strcpy(STMSTR2.sqlvar,"SELECT * FROM 会社表");
```

```
STMSTR2.sqllen = strlen(STMSTR2.sqlvar);
EXEC SQL CONNECT TO 'RDBDB1' USER 'USER1/PASS1';
EXEC SQL SET SCHEMA 'SCH1';
EXEC SQL PREPARE STM1 FROM :STMSTR1;                    (1)
EXEC SQL EXECUTE STM1;                                  (1)
EXEC SQL PREPARE STM2 FROM :STMSTR2;                    (1)
EXEC SQL SET SCHEMA 'SCH2';
EXEC SQL EXECUTE STM2;                                  (1)
EXEC SQL EXECUTE IMMEDIATE :STMSTR2;                    (2)
```

(1) 対象となるスキーマ名: SCH1

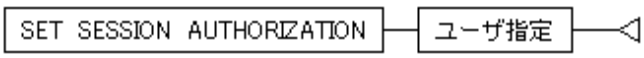
(2) 対象となるスキーマ名: SCH2

3.59 SET SESSION AUTHORIZATION文

機能

現行セションの対象となる利用者(認可識別子)を変更します。現行セションの対象となる利用者とは、USER定数の値および権限チェックのチェック対象となる利用者を指します。

記述形式



一般規則

- トランザクションは、終了していることが必要です。
- 変更した利用者は、SET SESSION AUTHORIZATION文を実行したセション内でのみ有効です。

ユーザ指定

ー サーバに接続する利用者を文字列定数または文字列型の埋込み変数で指定します。利用者は、認可識別子とパスワードを斜線“/”で区切って指定します。

ー ユーザ指定に空白を含めた場合は、斜線“/”で区切った認可識別子とパスワードのそれぞれについて前後の空白を取り除いた値が、認可識別子とパスワードになります。

ー 認可識別子は、18文字以内の先頭が英字で始まる英数字、または9文字以内の日本語文字列を指定します。

利用者の認証

不当な利用者がサーバに接続することを抑止するため、サーバに接続する時に利用者の認証が行われます。認証の方式は、以下の接続形式により異なります。

ー OSのログイン名で接続する場合

ー データベース専用利用者名で接続する場合

OSのログイン名で接続する場合

S L

- サーバに接続する利用者の認可識別子とパスワードは、以下に示すとおり、サーバにログイン可能で、かつデータベースの使用権限があるログイン名およびパスワードでなければなりません。

・ログイン名が有効な状態である

・ログイン名が有効期限内である

・ログイン名のパスワードがサーバシステムのパスワードと一致している

・ログイン名のサーバシステムのパスワードが有効期限内である

> **注意**
>
> OSのログイン名のパスワードが有効期限内であるかは、OSに依存します。また、パスワード有効期限の日数を設定した基準日には、OSによって以下の違いがあります。
>
> ・Solarisの場合:パスワードの設定日を日数にカウントする
>
> ・Linuxの場合:パスワードの設定日を日数にカウントしない

W

- サーバに接続する利用者の認可識別子とパスワードは、サーバにログイン可能で、かつデータベースの使用権限があるログイン名およびパスワードでなければなりません。

データベース専用利用者名で接続する場合

- サーバに接続する利用者は、CREATE USER文で登録した利用者で、かつデータベースの使用権限がある利用者でなければなりません。
- CREATE USER文でデータベース専用利用者として登録した利用者は、登録した認可識別子とパスワードを指定します。
- CREATE USER文でOSの利用者として登録した利用者は、サーバにログイン可能なログイン名およびパスワードを指定します。

ログイン名とパスワード

ログイン名とパスワードは、各サーバでは以下のように扱われます。

S L

Solaris/Linuxの場合
ログイン名: ログイン名
パスワード: ログイン名のパスワード

W

Windowsの場合
ログイン名: ユーザ名
パスワード: ユーザ名のパスワード

使用例

例

利用者を変更します。

```
SET SESSION AUTHORIZATION 'USER1/777###FF'
```

3.60 SET STATISTICS文

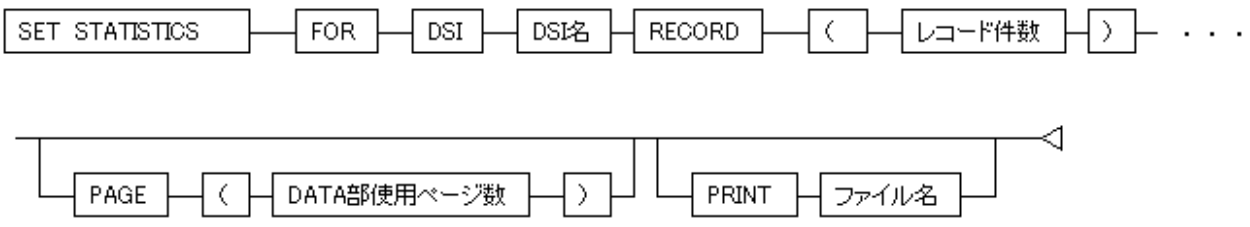
機能

データベースに最適化情報を設定します。

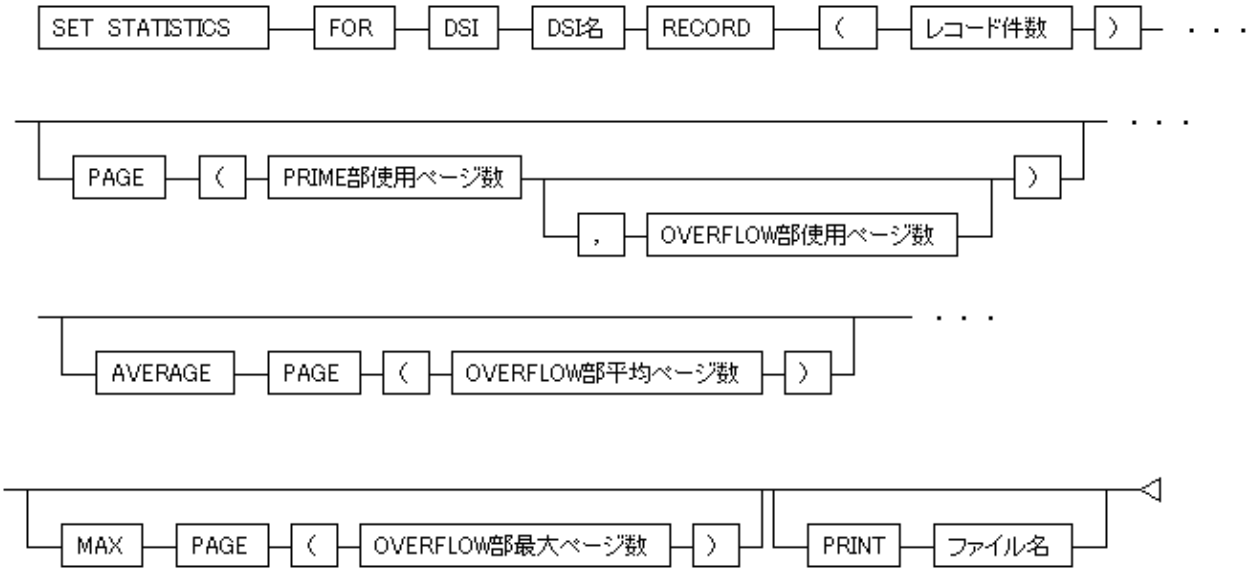
データベース簡単運用の場合は利用できません。

記述形式

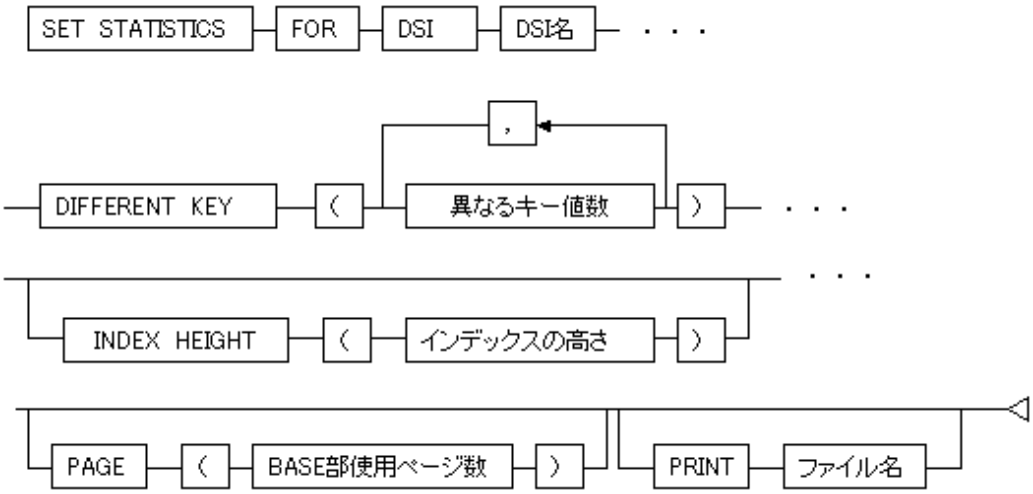
表のデータ構造がSEQUENTIALまたはOBJECTの表のDSI指定の場合



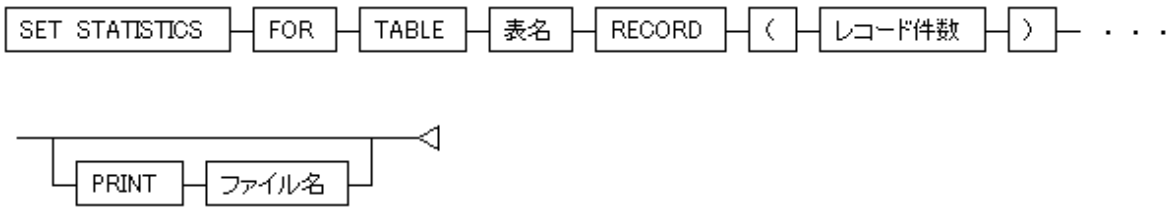
表のデータ構造がRANDOMの表のDSI指定の場合



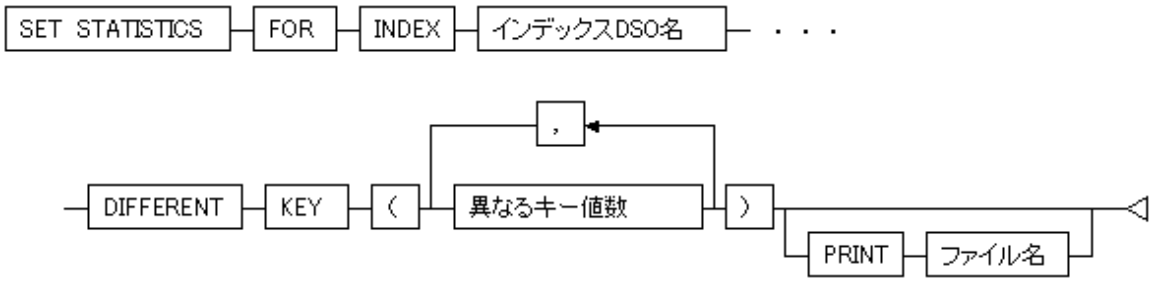
インデックスのDSI指定の場合



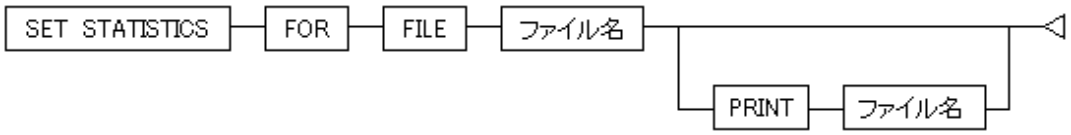
表指定の場合



インデックスのDSO指定の場合



ファイル指定の場合



権限

最適化情報を設定できるのは、指定した表、インデックスDSOまたはDSIの定義者のみです。

一般規則

DSI名

- 最適化情報を設定するDSI名を指定します。

表名

- 最適化情報を設定する表名を指定します。

インデックスDSO名

- 最適化情報を設定するインデックスのDSO名を指定します。

FILE

- 設定する最適化情報を記述したファイル名を指定します。PRINT STATISTICS文で出力した最適化情報を編集して、最適化情報を設定することができます。ファイル名に指定できる長さは、256バイト以内です。

RECORD

- DSIまたは表に格納するレコード件数を指定します。
- 表指定の場合で表が分割されている場合、指定されたレコード件数を表を構成するDSI数で割った値を各DSIに設定します。

PAGE

- レコードを格納するページ数を指定します。
- 表のデータ構造がSEQUENTIALまたはOBJECTの場合は、DATA部のページ数を指定します。
- 表のデータ構造がRANDOMの場合は、PRIME部とOVERFLOW部のページ数を設定します。
- インデックスの場合は、BASE部のページ数を設定します。
- 各ページ数は、“RDB運用ガイド(データベース定義編)”のデータベーススペースの所要量の見積りを参照して算出してください。
- ページ数の指定を省略した場合は、RECORDで指定されたレコード件数から、システムが算出した値を設定します。“RDB運用ガイド(データベース定義編)”の“データベーススペースの所要量の見積り”での算出値とは異なる値が設定される場合があります。可変長の列を含む場合は、最大レコード長で算出します。

AVERAGE PAGE

― 表のデータ構造がRANDOMの場合に、DSO定義のRULEで指定した規則(省略時はシステムで決定)によって分類された各バケットの平均のOVERFLOW部のページ数を設定します。以前に、rdbupsコマンドを使用して、データベースにデータが格納された状態から最適化情報を設定するなどして、値が判っている場合に設定します。通常は省略してください。省略した場合は0を設定します。

MAX PAGE

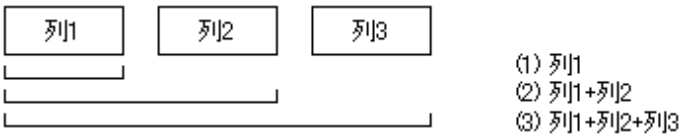
― 表のデータ構造がRANDOMの場合に、DSO定義のRULEで指定した規則(省略時はシステムで決定)によって分類された各バケットのうち、最大のOVERFLOW部のページ数を設定します。以前にrdbupsコマンドを使用して、データベースにデータが格納された状態から最適化情報を設定するなどして、値が判っている場合に設定します。通常は省略してください。省略した場合は0を設定します。

DIFFERENT KEY

― インデックスの場合に、インデックスを構成する各列の組合せごとに、異なるキー値の数を設定します。設定する異なるキー値数の情報は、インデックスを構成する列について、先頭からの列を必ず含む組合せのすべてについて設定する必要があります。

例

インデックスが列1、列2、列3で構成される場合



インデックスを構成する列が以下の場合の設定例を示します。

- インデックス構成列

  会社コード、部コード、課コード

- 設定する異なるキー値の情報

  - 会社コードの異なるキー値数(5種類)

  - 会社コード、部コードの組合せでの異なるキー値数(20種類)

  - 会社コード、部コード、課コードの組合せでの異なるキー値数(150種類)

指定方法

```
DIFFERENT  KEY(5,20,150)
```

― インデックスのDSO指定の場合でインデックスのDSOが分割されている場合、指定された異なるキー値数をインデックスのDSOを構成するDSI数で割った値を各DSIに設定します。

INDEX HEIGHT

― インデックスのINDEX部の高さを指定します。以前にrdbupsコマンドを使用して、データベースにデータが格納された状態から最適化情報を設定するなどして、値が判っている場合に設定し、通常は省略してください。省略した場合は、表のレコード件数に対応した値をシステムが決定して設定します。

PRINT

― 設定した最適化情報の内容を出力する場合に、出力先のファイル名を指定します。

最適化情報のファイルの記述形式

DSI指定の場合(SEQUENTIAL型またはOBJECT型)

```
DATABASE  = データベース名
------------------- SEQUENTIAL/OBJECT型DSI情報 ---
DSI       = DSI名
RECORD    = レコード件数
PAGE      = DATA部の使用ページ数 (注)
-----------------------------------------------------
```

注) 省略可能です。

DSI指定の場合(RANDOM型)

```
DATABASE       = データベース名
---------------------------- RANDOM型DSI情報 ----------
DSI            = DSI名
RECORD         = レコード件数
PAGE           = DATA部のページ数, OVERFLOW部のページ数(注)
AVERAGE PAGE   = OVERFLOW部の平均ページ数              (注)
MAX PAGE       = OVERFLOW部の最大ページ数              (注)
-------------------------------------------------------------
```

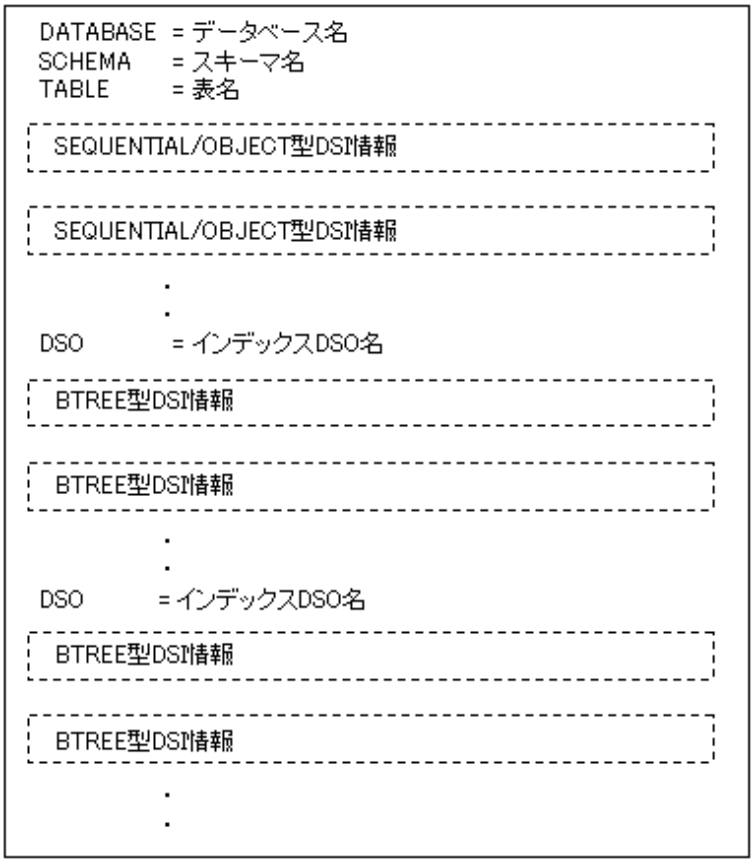
注) 省略可能です。

DSI指定の場合(BTREE型)

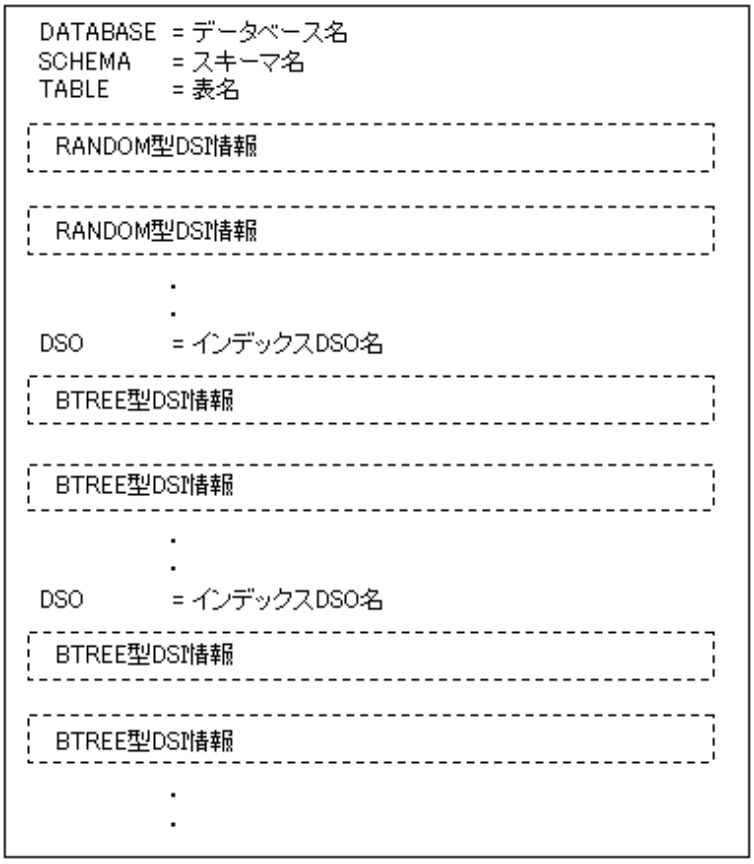
```
DATABASE       = データベース名
---------------------------- BTREE型DSI情報 -----------
DSI            =DSI名
PAGE           =BASE部のページ数    (注)
INDEX HEIGHT   =インデックスの高さ    (注)
DIFFERENT KEY  =異なるキー値数
                異なるキー値数
                    .
                    .
-------------------------------------------------------------
```

注) 省略可能です。

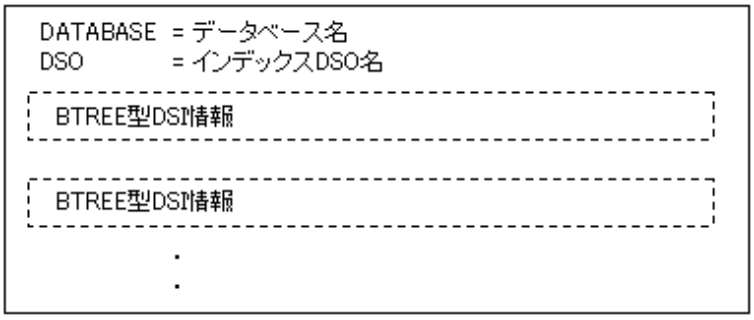
表指定の場合(SEQUENTIAL型またはOBJECT型)



表指定の場合(RANDOM型)



インデックスDSO指定の場合



使用例

例1

関東発注表DSIに最適化情報を設定します。表のデータ構造はSEQUENTIAL型です。

```
SET STATISTICS FOR DSI 関東発注表ＤＳＩ
    RECORD(30000000)
```

例2

製品名IXDSIのインデックスDSIに最適化情報を設定します。

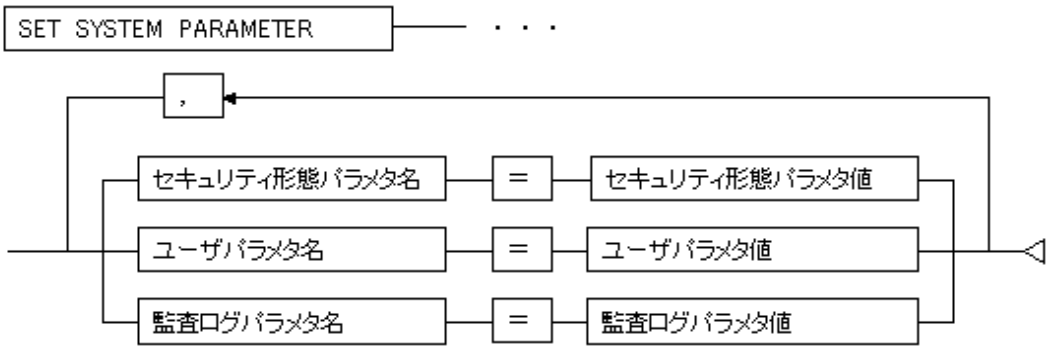
```
SET STATISTICS FOR DSI 製品名ＩＸＤＳＩ
    DIFFERENT KEY(150)
```

3.61 SET SYSTEM PARAMETER文

機能

SET SYSTEM PARAMETER文では、セキュリティパラメタを設定します。

記述形式



権限

- パラメタを設定できるのは、RDBディクショナリの定義者のみです。

一般規則

- すでにパラメタ値が設定されている場合は指定された値に置き換えます。
- 同じパラメタを複数指定することはできません。
- 監査ログデータベースに接続した状態からは実行できません。監査ログデータベース以外のデータベースに接続してから実行してください。

セキュリティ形態パラメタ名とセキュリティ形態パラメタ値

USER_CONTROL

【指定形式】

USER_CONTROL = {YES | NO}

【セキュリティ形態パラメタの意味】

利用者登録の使用宣言を行うかどうかを指定します。利用者登録の使用宣言は、登録された利用者のみにデータベースのアクセスを許可するシステムにする場合に指定します。省略した場合は、NOが指定されたものとみなします。NOを指定して実行した場合、登録されている利用者は削除されます。

【パラメタの意味】

YES:

利用者登録の使用宣言を行います。

NO:

利用者登録の使用宣言を行いません。

注意

監査ログパラメタのAUDITに利用者が指定されている場合、利用者登録の使用宣言をNOからYESに変更できません。監査ログパラメタのAUDITから利用者の指定を外してください。

ROLE_RANGE

【指定形式】

ROLE_RANGE = { LEVEL1 | LEVEL0 }

【セキュリティ形態パラメタの意味】

本パラメタは、運用系コマンド(rdbfmt、rdbsloader、rdbsaloader、rdbsuloader、rdbunlまたはrdbunlxコマンド)に対するロールの使用を有効にするかどうかを指定します。運用系コマンドに対するロールの使用宣言は、各コマンドで必要な権限を含むロールを付与されている利用者に対し、運用系コマンドの実行を許可する場合に指定します。
LEVEL1を指定することにより、コマンドの実行者への権限付与がロールを使用して行えるため、権限管理作業のコストを削減させることができます。
省略された場合は、標準セキュリティ運用時、標準運用時ともにLEVEL0が指定されたとみなします。
本パラメタは、CREATE USER文、ALTER USER文での利用者ごとの設定はできません。

【パラメタの意味】

LEVEL1:

上記の運用系コマンドに対するロールの使用を有効にします。

LEVEL0:

上記の運用系コマンドに対するロールの使用を有効にしません。

注意

LEVEL1は標準運用時のみ指定可能です。
標準セキュリティ運用時に、LEVEL1を指定した場合は無視されます。
パラメタ値にLEVEL1を指定すると、運用系コマンドの実行に必要な権限を含むロールが付与されている利用者は、運用系コマンドが実行できるようになります。意図しない利用者に運用系コマンドが実行されないように注意して、パラメタ値を変更してください。

権限の付与はGRANT文で行います。詳細は“3.47 GRANT文”を参照してください。

ユーザパラメタ名とユーザパラメタ値

ユーザパラメタには、以下のパラメタが指定できます。なお、SET SYSTEM PARAMETER文ではDEFAULT_ROLEを指定することはできません。

ユーザパラメタは利用者ごとに設定が可能です。利用者ごとの設定は、CREATE USER文またはALTER USER文で行います。これらのパラメタは、“USER_CONTROL=YES”を指定している場合に指定できます。

ユーザパラメタの設定値は、次のコネクション接続時に有効になります。

注意

以下のユーザパラメタは、Symfoware Server Enterprise Extended Editionを利用した場合のみ指定できます。ただし、ロードシェア運用時は使用できません(指定時はJYP4441Eを出力してエラーとなります)。

- MAX_CONNECTION
- MAX_MEMORY_USE
- MAX_WORKFILE_USE
- MAX_WORKFILE_NUM
- MAX_TRAN_TIME
- MAX_TRAN_MEM
- MAX_WAIT_TIME
- RESOURCE_LIMIT_CHECK

PASSWORD_CHANGE_TIME

【指定形式】

PASSWORD_CHANGE_TIME = 日数

【ユーザパラメタの意味】

何日前からパスワードの変更を勧めるかの日数を指定します。指定できる範囲は、0～128です。単位は日数です。省略した場合は、0が指定されたとみなします。0を指定すると変更催促を行いません。この日数を超えてサーバに接続すると、サーバに接続しSQL文を実行することはできますが、CONNECT文または最初に実行するSQL文で変更督促が行われます。

PASSWORD_LIMIT_TIME

【指定形式】

PASSWORD_LIMIT_TIME = 日数

【ユーザパラメタの意味】

パスワードの期限を指定します。指定できる範囲は、0～128および-1です。単位は日数です。省略した場合は、0が指定されたとみなします。0を指定すると無制限になります。-1を指定すると1回だけパスワードが有効となります。この日数を超えてサーバに接続すると、CONNECT文または最初に実行するSQL文がエラーになりサーバに接続できません。

INVALID_PASSWORD_WAIT_TIME

【指定形式】

INVALID_PASSWORD_WAIT_TIME = 待ち時間

【ユーザパラメタの意味】

パスワードが誤りの時の待ち時間を指定します。指定できる範囲は、0～5です。単位は秒です。省略した場合は、4が指定されたとみなします。誤ったパスワードを指定してCONNECT文を実行すると、CONNECT文の処理は指定した秒数だけ待ち状態になりエラーとなります。

INVALID_PASSWORD_TIME

【指定形式】

INVALID_PASSWORD_TIME = 回数

【ユーザパラメタの意味】

パスワードを連続して失敗できる回数を指定します。指定できる範囲は、0～10です。単位は回数です。省略した場合は、5が指定されたとみなします。

MIN_PASSWORD_SIZE

【指定形式】

MIN_PASSWORD_SIZE = パスワードサイズ

【ユーザパラメタの意味】

パスワードに最低限必要なバイト数を指定します。指定できる範囲は、6～8です。省略した場合は、6が指定されたとみなします。
本パラメタはSET USER PASSWORD文でパスワードを変更する場合に変更後のパスワードの長さがチェックされます。

MAX_CONNECTION

【指定形式】

MAX_CONNECTION = コネクション数

【ユーザパラメタの意味】

1人の利用者がSymfoware/RDBシステムに対して、同時に接続可能なコネクション数を指定します。
指定できる範囲は、0～32767です。省略した場合は、1が指定されたとみなします。0を指定すると無制限になります。

MAX_MEMORY_USE

【指定形式】

MAX_MEMORY_USE = メモリの最大量

【ユーザパラメタの意味】

1つのコネクションで使用可能なメモリ量を指定します。
指定できる範囲は、0～32767です。単位はメガバイトです。省略した場合は、16が指定されたとみなします。0を指定すると無制限になります。
コネクションの接続が完了するまでのメモリの最大量は、Symfoware/RDBシステムに対して設定された値になります。

MAX_WORKFILE_USE

【指定形式】

MAX_WORKFILE_USE = 作業用ファイルの最大量

【ユーザパラメタの意味】

1つのコネクションで使用可能な作業用ファイルの量を指定します。
指定できる範囲は、0～32767です。単位はメガバイトです。省略した場合は、0が指定されたとみなします。0を指定すると無制限になります。

MAX_WORKFILE_NUM

【指定形式】

MAX_WORKFILE_NUM = 作業用ファイルの数

【ユーザパラメタの意味】

1つのコネクションで使用可能な作業用ファイルの数を指定します。
指定できる範囲は、0～32767および-1です。単位は個です。省略した場合は、16が指定されたとみなします。0を指定すると無制限になります。-1を指定すると作業用ファイルを作成することはできません。

MAX_TRAN_TIME

【指定形式】

MAX_TRAN_TIME = 実行時間

【ユーザパラメタの意味】

1つのトランザクションで使用可能な時間を指定します。
指定できる範囲は、0～32767です。単位は秒です。省略した場合は、300が指定されたとみなします。0を指定すると無制限になります。
時間超過を検出した場合には、接続中のコネクションが切断されます。

MAX_TRAN_MEM

【指定形式】

MAX_TRAN_MEM = メモリ量

【ユーザパラメタの意味】

1つのトランザクションで使用可能なトランザクション用メモリ量を指定します。
指定できる範囲は、0～32767です。単位はキロバイトです。省略した場合は、1024が指定されたとみなします。0を指定すると無制限になります。

MAX_WAIT_TIME

【指定形式】

MAX_WAIT_TIME = 時間

【ユーザパラメタの意味】

アプリケーションの無応答待ち時間を指定します。
指定できる範囲は、0～32767です。単位は分です。省略した場合は、無制限が指定されたとみなします。0を指定すると無制限になります。

時間超過を検出した場合には、接続中のコネクションが切断されます。コネクションの接続が完了するまでの最大の無応答の待ち時間は、Symfoware/RDBシステムに対して設定された値になります。

RESOURCE_LIMIT_CHECK

【指定形式】

RESOURCE_LIMIT_CHECK = {USER | CONNECTION}

【ユーザパラメタの意味】

SET SESSION AUTHORIZATION文の実行時に、SET SESSION AUTHORIZATION文で指定した利用者の使用可能な資源量を変更するかどうかを指定します。
省略した場合は、USERが指定されたものとみなします。
本パラメタは、CREATE USER文、ALTER USER文での利用者ごとの設定はできません。

【パラメタの意味】

USER:

使用可能な資源量を変更します。

CONNECTION:

使用可能な資源量を変更しません。

監査ログパラメタ名と監査ログパラメタ値

監査ログパラメタには、以下のパラメタが指定できます。

注意

監査ログパラメタは、ロードシェア運用時は使用できません(設定しても無視されます)。

AUDIT_SESSION_SUCCESS

【指定形式】

AUDIT_SESSION_SUCCESS = {YES | NO}

【監査ログパラメタの意味】

接続に成功したアプリケーションの実行に関する監査ログを取得するかどうかを指定します。省略した場合は、NOが指定されたとみなします。

【パラメタの意味】

YES:

監査ログを取得します。

NO:

監査ログを取得しません。

AUDIT_SESSION_FAIL

【指定形式】

AUDIT_SESSION_FAIL = {YES | NO}

【監査ログパラメタの意味】

接続に失敗したアプリケーションの実行に関する監査ログを取得するかどうかを指定します。省略した場合は、NOが指定されたとみなします。

【パラメタの意味】

YES:

監査ログを取得します。

NO:

監査ログを取得しません。

AUDIT_ACCESS_SUCCESS

【指定形式】

AUDIT_ACCESS_SUCCESS = {YES | NO}

【監査ログパラメタの意味】

表およびプロシジャルーチンなどの資源に対するアクセスで成功したものの監査ログを取得するかどうかを指定します。省略した場合は、NOが指定されたとみなします。

【パラメタの意味】

YES:

監査ログを取得します。

NO:

監査ログを取得しません。

AUDIT_ACCESS_FAIL

【指定形式】

AUDIT_ACCESS_FAIL = {YES | NO}

【監査ログパラメタの意味】

表およびプロシジャルーチンなどの資源に対するアクセスで失敗したものの監査ログを取得するかどうかを指定します。省略した場合は、NOが指定されたとみなします。

【パラメタの意味】

YES:

監査ログを取得します。

NO:

監査ログを取得しません。

AUDIT_SQL

【指定形式】

AUDIT_SQL = {YES | NO}

【監査ログパラメタの意味】

アプリケーションから実行されるSQL文とSQL文の実行時間に関する監査ログを取得するかどうかを指定します。省略した場合は、NOが指定されたとみなします。
AUDIT_SQL=NOを指定した場合、AUDIT_SQLBINDの指定に関わらず、SQL文の入力に関する情報の監査ログは取得されません。
監査ログのビュー表AUDIT_SQLおよびAUDIT_SQLBINDが存在しない監査ログ表の場合は、本パラメタの値に関係なく、アプリケーションから実行されるSQL文とSQL文の実行時間に関する監査ログは取得されません。これらの情報を取得する場合は、rdbauditコマンドのVLオプションで監査ログ表のバージョンを確認し、バージョンが0の場合は、“インストールガイド(サーバ編)”の“Symfoware Server移行時の作業手順”の“環境の作成”を参照して監査ログ表を再作成してください。

【パラメタの意味】

YES:

監査ログを取得します。

NO:

監査ログを取得しません。

AUDIT_SQLBIND

【指定形式】

AUDIT_SQLBIND = {YES | NO}

【監査ログパラメタの意味】

アプリケーションから実行されるSQL文の監査ログを取得する場合に、SQL文の入力に関する情報の監査ログを取得するかどうかを指定します。省略した場合は、NOが指定されたとみなします。
AUDIT_SQL=NOを指定した場合、AUDIT_SQLBINDの指定に関わらず、SQL文の入力に関する情報の監査ログは取得されません。
監査ログのビュー表AUDIT_SQLおよびAUDIT_SQLBINDが存在しない監査ログ表の場合は、本パラメタの値に関係なく、SQL文の入力に関する情報の監査ログは取得されません。これらの情報を取得する場合は、rdbauditコマンドのVLオプションで監査ログ表のバージョンを確認し、バージョンが0の場合は、“インストールガイド(サーバ編)”の“Symfoware Server移行時の作業手順”の“環境の作成”を参照して監査ログ表を再作成してください。

【パラメタの意味】

YES:

監査ログを取得します。

NO:

監査ログを取得しません。

AUDIT_MANAGE

【指定形式】

AUDIT_MANAGE = {YES | NO}

【監査ログパラメタの意味】

管理者の実行に関する監査ログを取得するかどうかを指定します。省略した場合は、NOが指定されたとみなします。

【パラメタの意味】

YES:

監査ログを取得します。

NO:

監査ログを取得しません。

AUDIT_ERROR

【指定形式】

AUDIT_ERROR = {YES | NO}

【監査ログパラメタの意味】

システムにおける重大なエラー、その他の事象の監査ログを取得するかどうかを指定します。省略した場合は、NOが指定されたとみなします。

【パラメタの意味】

YES:

監査ログを取得します。

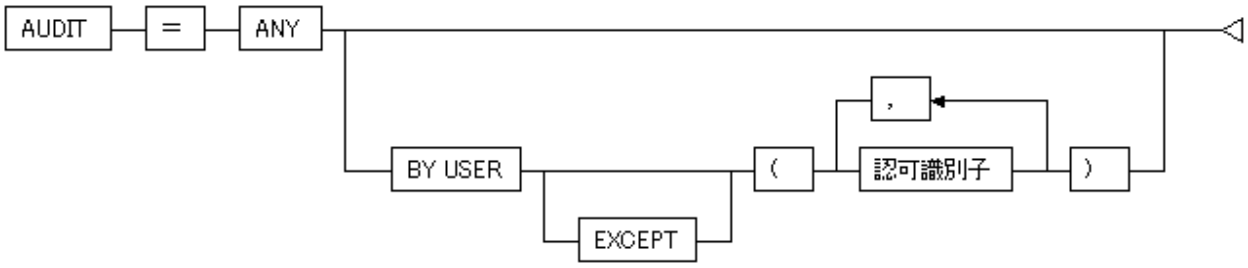
NO:

監査ログを取得しません。

AUDIT

【指定形式】

AUDIT = ANY [BY USER [EXCEPT] (認可識別子[{, 認可識別子}…])]



【監査ログパラメタの意味】

- 監査ログを取得する利用者を限定します。省略した場合はすべての利用者の監査ログを取得します。
- 利用者をデータベースで管理する場合とOSで管理する場合のどちらの運用でも設定することができます。
- AUDITパラメタの設定は、実行中のアプリケーションまたはコマンドに対して即時に有効となります。
- オペランドに“ANY”のみを指定した場合は、すべての利用者の監査ログを取得します。
- 同じ利用者を複数指定することはできません。
- 利用者登録の使用宣言を行っている場合、定義されていない利用者は指定できません。利用者定義の詳細については、“3.24 CREATE USER文(利用者定義文)”を参照してください。

【パラメタの意味】

EXCEPT:

本パラメタを指定した場合、指定した利用者を除いた利用者の監査ログを取得します。

認可識別子:

監査ログを取得する利用者を限定します。

EXCEPTを省略した場合

監査ログを取得する利用者名を指定します。

EXCEPTを指定した場合

監査ログを取得しない利用者名を指定します。

18文字以内の先頭が英字で始まる英数字、または9文字以内の日本語文字列を指定します。

以下の情報は、AUDITパラメタの設定に関係なく常に取得されます。

ー セションに関する情報（接続失敗時）

ー エラーに関する情報

注意

誤った利用者を設定すると監査ログが正しく取得できません。
設定後、正しく設定されているか必ず確認してください。
設定されている内容は、rdbprtコマンドのmオプションにPARAMを指定して実行することで確認できます。

AUDIT_LOG_FULL

【指定形式】

AUDIT_LOG_FULL = {STOP | REUSE | CANCEL}

【監査ログパラメタの意味】

監査ログデータベースが満杯時の対処方法を指定します。省略した場合は、CANCELが指定されたとみなします。

【パラメタの意味】

STOP:

Symfoware/RDBを強制停止します。

REUSE:

一番古い監査ログエレメントを再利用します。

注意

監査ログエレメントは満杯になると循環利用され、古い監査ログの内容が上書きされて失われます。そのため、監査ログエレメントが満杯になる事象を検出して監査ログを外部媒体にバックアップする運用を行う必要があります。詳細は“RDB運用ガイド”の“監査ログデータベースのバックアップと初期化”を参照してください。

CANCEL:

監査ログの取得を停止して、メッセージログファイルに出力します。

注意

- メッセージログファイルへの出力は、監査ログデータベースへの出力と比べて時間がかかります。そのため、監査ログエレメントが満杯になる事象を検出して監査ログを外部媒体にバックアップする運用を行う必要があります。詳細は“RDB運用ガイド”の“監査ログデータベースのバックアップと初期化”を参照してください。
- 監査ログデータベースを暗号化している場合でも、メッセージログファイルに出力するログは暗号化しません。そのため、機密情報が漏洩する可能性があります。このような場合の危険性を回避するためには、“STOP”を指定するか、監査ログの取得範囲にSQL文を含めずに、監査ログ運用を行ってください。

使用例

例

利用者制御を使用し、パスワードの期限“PASSWORD_LIMIT_TIME”を30日に設定します。

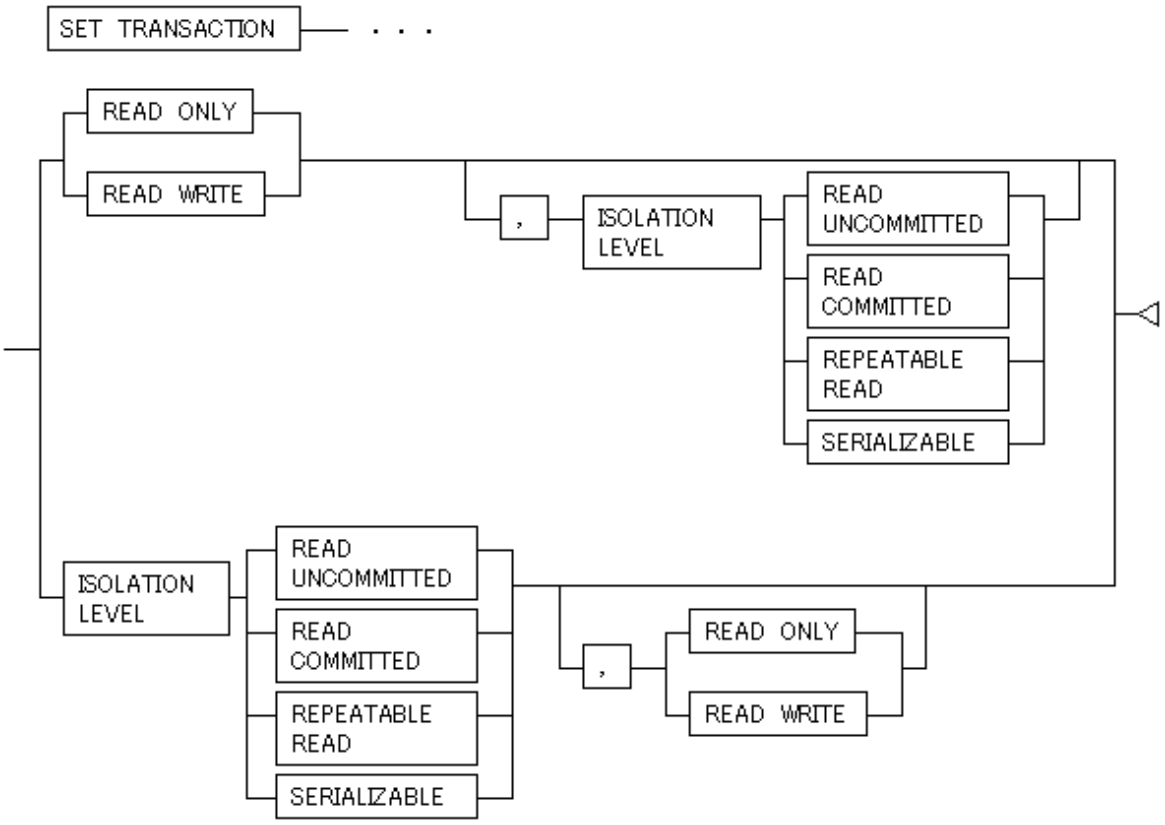
```
SET SYSTEM PARAMETER
  USER_CONTROL = YES,
  PASSWORD_LIMIT_TIME = 30
```

3.62 SET TRANSACTION文

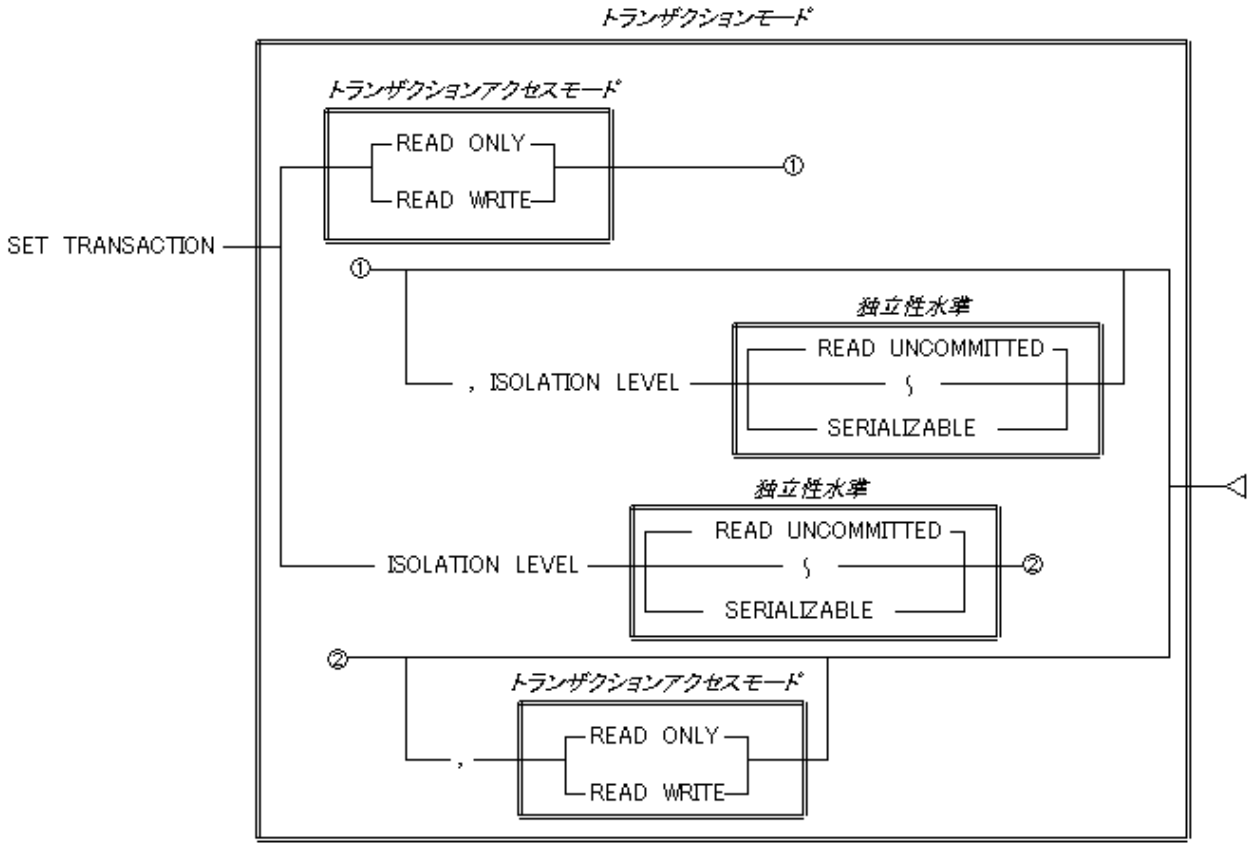
機能

トランザクションモードを切り替えます。

記述形式


SET TRANSACTION
READ ONLY
READ WRITE
,
ISOLATION LEVEL
READ UNCOMMITTED
READ COMMITTED
REPEATABLE READ
SERIALIZABLE
ISOLATION LEVEL
READ UNCOMMITTED
READ COMMITTED
REPEATABLE READ
SERIALIZABLE
,
READ ONLY
READ WRITE

構文の構成


トランザクションモード
トランザクションアクセスモード
READ ONLY
READ WRITE
①
SET TRANSACTION
①
独立性水準
READ UNCOMMITTED
, ISOLATION LEVEL
SERIALIZABLE
独立性水準
READ UNCOMMITTED
ISOLATION LEVEL
②
SERIALIZABLE
②
トランザクションアクセスモード
READ ONLY
,
READ WRITE

一般規則

- トランザクションは、終了していることが必要です。
- 動作環境ファイルにDSO_LOCKを指定した場合は、SET TRANSACTION文を指定することはできません。
- 動作環境ファイルのR_LOCKがNOの場合、SET TRANSACTION文にREPEATABLE READを指定しても、独立性水準はSERIALIZABLEになります。
- 動作環境ファイルのR_LOCKがYESの場合、SET TRANSACTION文にSERIALIZABLEを指定しても、独立性水準はREPEATABLE READになります。

トランザクションモード

ー 同じトランザクションモードは2つ以上指定することはできません。

ー トランザクションモードは、SET TRANSACTION文によってのみ変更可能です。それ以外の契機で変更されることはありません。

READ ONLYおよびREAD WRITE(トランザクションアクセスモード)

ー READ ONLYを指定した場合、トランザクションアクセスモードは、読込み専用モードになります。READ ONLYは、トランザクション内で更新系のSQL文を許しません。

ー READ WRITEを指定した場合、トランザクションアクセスモードは更新可能になります。

ー トランザクションアクセスモードを省略した場合は、READ WRITEとなります。

READ UNCOMMITTED

ー READ UNCOMMITTEDを指定した場合、ほかのトランザクションで追加および更新されたデータをそのトランザクションがCOMMITされる前に検索することができます。
ただし、以下の条件の時は、直前の検索結果と異なる結果を得る場合があります。

- ほかの利用者の更新系のトランザクションがCOMMITされる前に、その更新したデータの検索をした場合
- 検索後のデータにほかの利用者が更新および削除をしてCOMMITした場合
- 探索条件を指定したSQL文を実行し、その後、ほかの利用者が探索条件に一致した行を追加した場合

READ COMMITTED

ー READ COMMITTEDを指定した場合、ほかのトランザクションで追加および更新されたデータはそのトランザクションがCOMMITするまで検索することはできません。
ただし、検索する表が、DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の場合には、ほかのトランザクションがCOMMITするまでの間は、追加および更新がなされる前のデータを検索することができます。

いずれの場合も、以下の条件の時は、直前の検索結果と異なる結果を得る場合があります。

- 検索後のデータにほかの利用者が更新および削除をしてCOMMITした場合
- 探索条件を指定したSQL文を実行し、その後、ほかの利用者が探索条件に一致した行を追加した場合

REPEATABLE READ

ー REPEATABLE READを指定した場合、ほかのトランザクションで追加および更新されたデータはそのトランザクションがCOMMITするまで検索することはできません。
ただし、以下の条件の時は、直前の検索結果と異なる結果を得る場合があります。

- 探索条件を指定したSQL文を実行し、その後、ほかの利用者が探索条件に一致した行を追加した場合

SERIALIZABLE

ー SERIALIZABLEを指定した場合、ほかのトランザクションで追加および更新されたデータは、そのトランザクションがCOMMITされるまで検索することはできません。

使用例

例

読込み専用モードでREAD COMMITTEDを指定した場合の例を示します。

```
SET TRANSACTION READ ONLY,
    ISOLATION LEVEL READ COMMITTED
```

3.63 SET USER PASSWORD文

機能

現行セションの利用者(認可識別子)のパスワードを変更します。

記述形式

一般規則

- トランザクションは、終了していることが必要です。
- CREATE USER文でWITH句にDBMSを指定した利用者のみ実行できます。

パスワード

- 変更後のパスワードを指定します。文字列定数または文字列型の埋込み変数で指定します。
- パスワードは、現在使用しているパスワードと比較して3文字以上違うものでなければなりません。ただし、大文字と小文字は同じ文字として扱います。
- パスワードの指定形式の詳細については、“3.24 CREATE USER文(利用者定義文)”を参照してください。
- パスワードに空白を含めた場合は、前後の空白を取り除いた値が、パスワードになります。

使用例

例

現行セションの利用者のパスワードを777###FFに変更します。

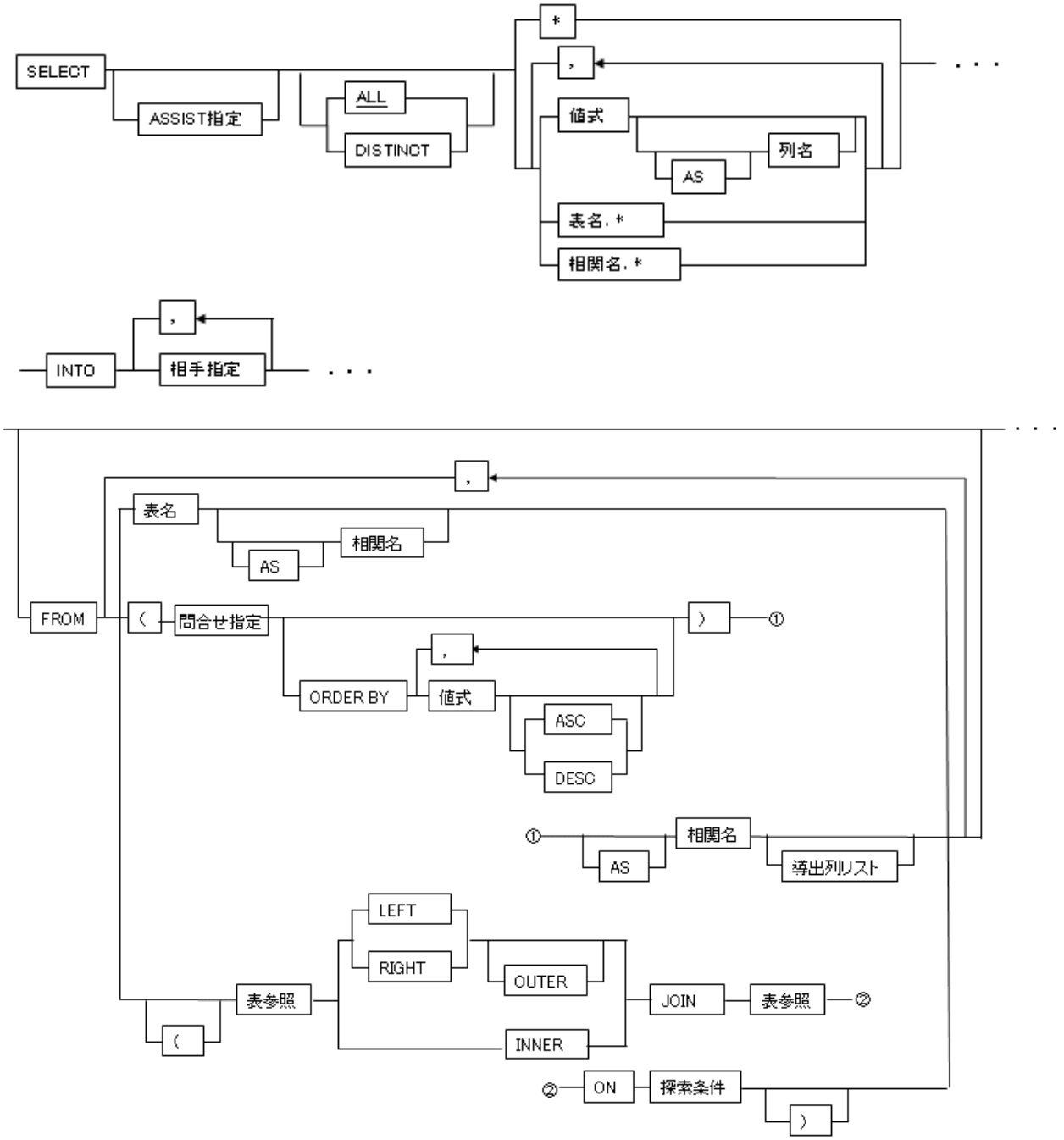
```
SET USER PASSWORD '777###FF'
```

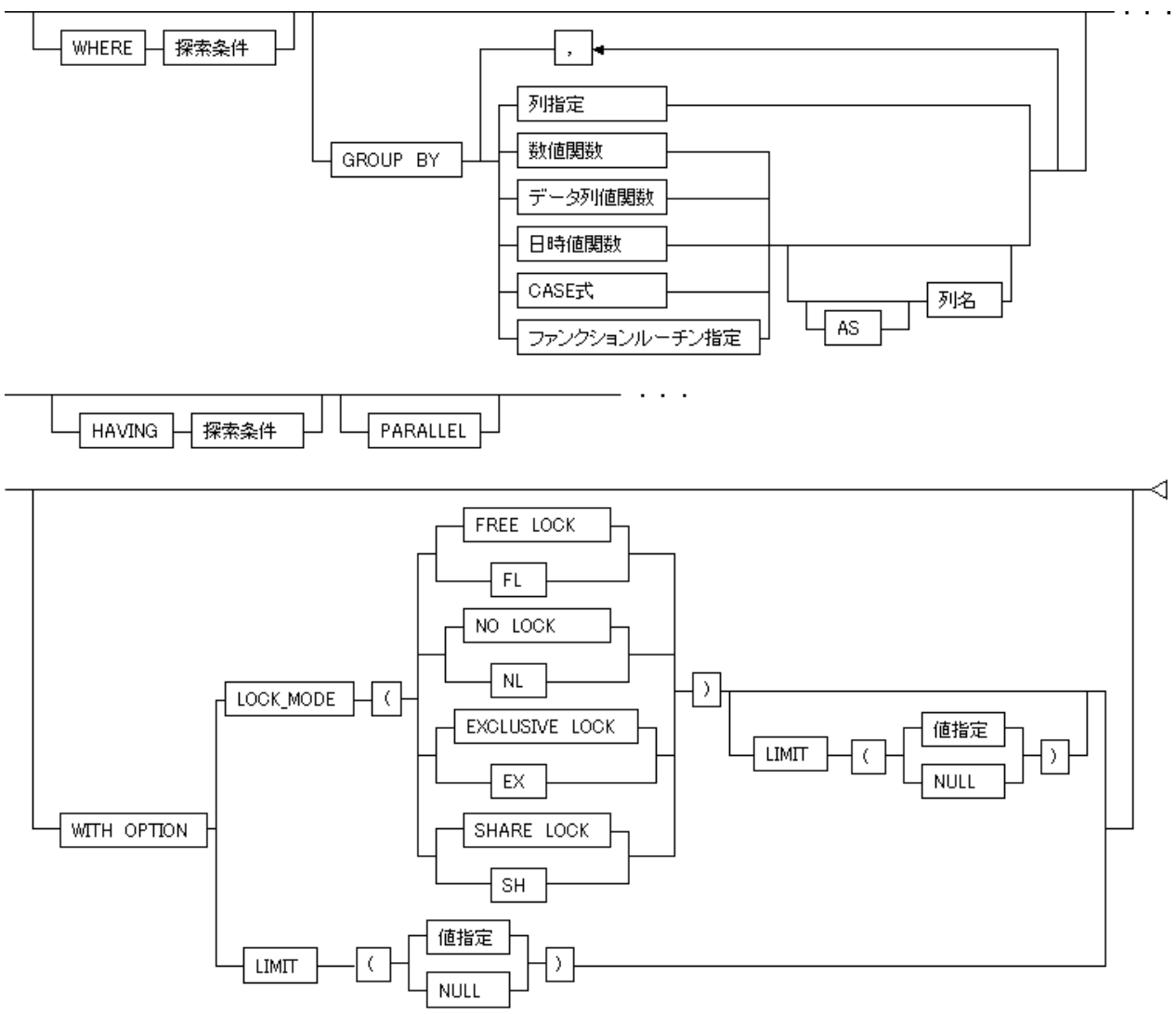
3.64 Single row SELECT文(単一行SELECT文)

機能

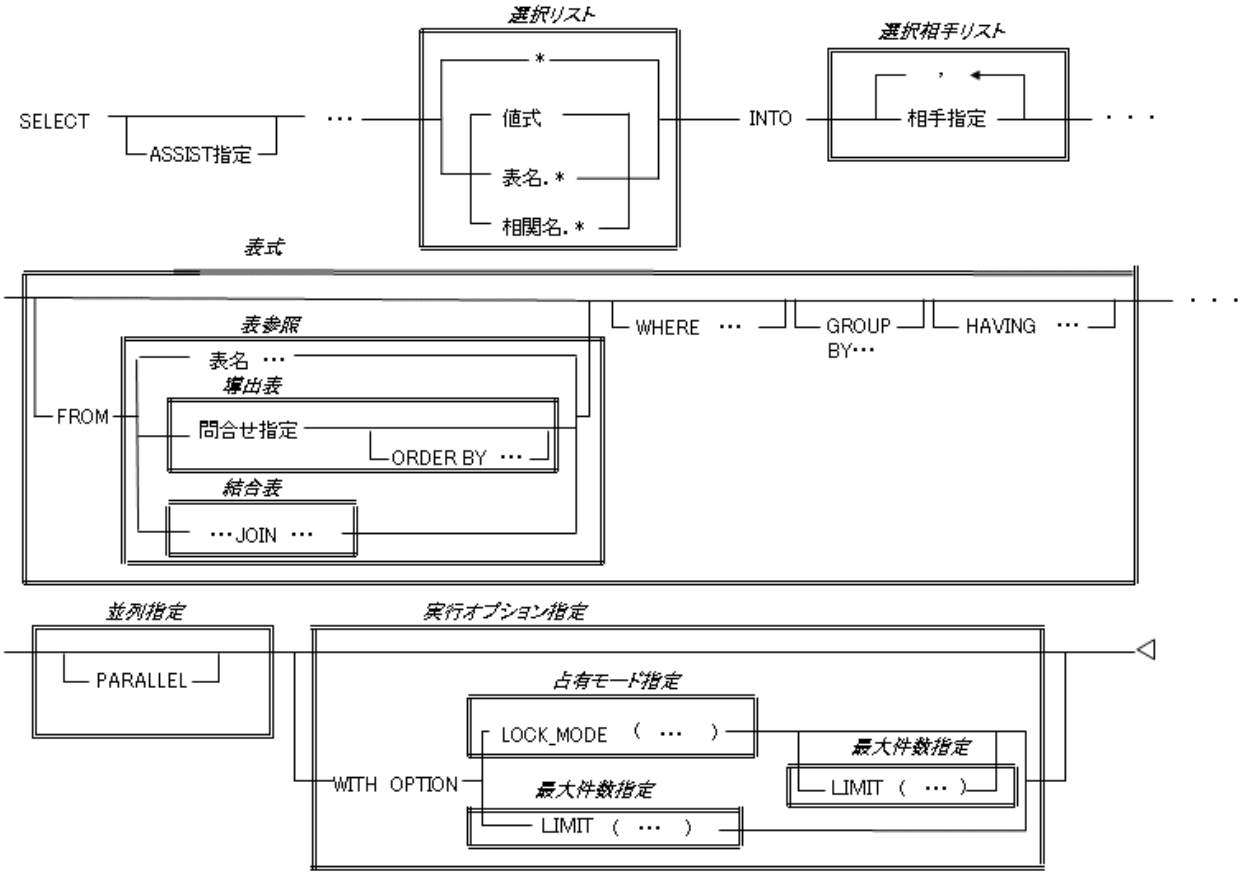
表の指定された行から値を取り出します。

記述形式


SELECT
ASSIST指定
ALL
DISTINCT
*
,
値式
AS
列名
表名.*
相関名.*
INTO
,
相手指定
FROM
,
表名
AS
相関名
(
問合せ指定
)
①
ORDER BY
,
値式
ASC
DESC
①
AS
相関名
導出列リスト
LEFT
RIGHT
OUTER
INNER
JOIN
表参照
②
表参照
(
②
ON
探索条件
)


WHERE
探索条件
GROUP BY
,
列指定
数値関数
データ列値関数
日時値関数
CASE式
ファンクションルーチン指定
AS
列名
HAVING
探索条件
PARALLEL
WITH OPTION
LOCK_MODE
(
FREE LOCK
FL
NO LOCK
NL
EXCLUSIVE LOCK
EX
SHARE LOCK
SH
)
LIMIT
(
値指定
NULL
)
LIMIT
(
値指定
NULL
)

構文の構成



参照項番

- ASSIST指定 → “2.15 ASSIST指定”
- 値式 → “2.11 値式”
- 相手指定 → “2.3 値指定と相手指定”
- 探索条件 → “2.13 探索条件”
- 列指定 → “2.4 列指定”
- 数値関数 → “2.5.2 数値関数”
- データ列値関数 → “2.5.3 データ列値関数”
- 日時値関数 → “2.5.4 日時値関数”
- CASE式 → “2.6 CASE式”
- ファンクションルーチン指定→ “2.5.5 ファンクションルーチン指定”

権限

- 単一行SELECT文を実行できるのは、処理対象の表のSELECT権の保持者です。

一般規則

選択リスト

ー 選択リストには、表式の結果から導出される表の列を指定します。

- 選択リストの個数と、選択相手リストに指定されている相手指定の個数は、同じであることが必要です。選択相手リストは、選択リストに対応する検索結果の取り出し先です。
- 選択リストに“*”を指定した場合、FROM句で指定した順に各表の各列のすべてを記述したのと同じ意味です。
- 選択リストに“表名.*”または“相関名.*”が指定された場合、表のすべての列を記述したのと同じ意味です。“表名.”または“相関名.”は、“修飾子”と呼びます。
- 選択リスト中に含まれる列指定は、FROM句で指定した表の列またはGROUP BY句のAS句の列名であることが必要です。
- 選択リストに格納構造がOBJECTのBLOB型の列を複数指定することはできません。
- 表式の結果がグループ表の場合、選択リストの列指定は、グループ化列またはGROUP BY句のAS句の列名を指定するか、集合関数指定またはグループ化関数の引数で指定することが必要です。
- 表式の結果がグループ表でなく、選択リストに集合関数を指定する場合は、選択リストのすべてが集合関数指定であることが必要です。
- 選択リストにAS句が指定された場合、選択リストの結果の列名はAS句に指定された列名となります。
- 単一行SELECT文に以下の指定がある場合は、選択リストにROW_IDは指定できません。
  - 集合関数
  - DISTINCT
  - GROUP BY句
  - HAVING句
  - ビュー表の検索
- 選択リストにROW_IDを指定した場合、相手指定にはROW_IDと対応する変数を指定することが必要です。データ型と対応する変数定義については、“表6.1 SQLのデータ型と対応するC変数定義”および“表6.3 SQLのデータ型と対応するCOBOL変数定義”を参照してください。

相手指定

- 選択リストの結果のデータ型に対して、相手指定に指定可能なデータ型の条件を以下に示します。

  選択リストのデータ型が文字列型の場合:

  相手指定のデータ型は文字列型であることが必要です。

  選択リストのデータ型が各国語文字列型の場合:

  相手指定のデータ型は各国語文字列型であることが必要です。

  選択リストのデータ型が真数型の場合:

  相手指定のデータ型は真数型または概数型であることが必要です。

  選択リストのデータ型が概数型の場合:

  相手指定のデータ型は真数型または概数型であることが必要です。

  選択リストのデータ型が日時型の場合:

  相手指定のデータ型は文字列型であることが必要です。

  選択リストのデータ型が時間隔型の場合:

  相手指定のデータ型は文字列型または真数型であることが必要です。ただし、真数型が指定できるのは選択リストのデータ型が単一日時フィールドで指定された時間隔型の場合のみです。

  選択リストのデータ型がBLOB型の場合:

  相手指定のデータ型はBLOB型であることが必要です。

- 相手指定の変数に値を代入する途中に誤りが起こると、データ例外(代入エラー)となります。

— 検索結果が相手指定に設定されるときの規則は次のようになります。

文字列型データの場合:

検索結果のデータ長が相手指定の長さと同じ場合は、文字データがそのまま設定されます。検索結果のデータ長が相手指定の長さよりも短い場合は、空白が相手指定の右側に設定されます。検索結果のデータ長が相手指定の長さよりも長い場合は、相手指定の左側から相手指定の長さだけの文字データが設定されます。

各国語文字列型の場合:

文字列型の場合と同じように設定されます。

真数型の場合:

相手指定のデータ型に変換して設定されます。検索結果の整数部が相手指定の精度と位取りで表現できる場合は、相手指定の精度と位取りに従って設定されます。上位の桁落ちが発生する場合は例外となり、下位の位が落ちる場合は切り捨てられます。

概数型のデータの場合:

相手指定には、検索結果の概数値が設定されます。

日時型のデータの場合:

相手指定のデータ型に変換して設定されます。

時間隔型のデータの場合:

相手指定のデータ型に変換して設定されます。

BLOB型のデータの場合:

相手指定には、検索結果のバイナリ値が設定されます。

ROW_IDの場合:

相手指定には、行識別子が設定されます。

表式

— 表式の結果は1行になることが必要です。結果が空の場合は、例外(データなし)、複数行の場合は、例外(基数違反)となります。単一行SELECT文の結果を以下に示します。

- DISTINCTが指定されている場合、単一行SELECT文の結果の各行同士がまったく同じ値を持つ行がある場合、それらの複数行を1行とする処理が行われます。このとき、NULL値同士は等しいとみなされます。ALLが指定されている場合、同じ値を持つ複数行を1行とする処理は行われません。また、どちらも指定されていない場合は、ALLが指定されたのと同じになります。
- 表式の結果が、グループ表でない場合、単一行SELECT文の結果は次のようになります。選択リストに集合関数指定がない場合、表式の結果の各行に結果指定の値式で示す演算が行われた、行の集合となります。選択リストに集合関数の指定がある場合の結果は、表式の結果を集合関数に適用した1行となります。
- 表式の結果がグループ表の場合、単一行SELECT文の結果は次のようになります。結果の行の数はグループの数となります。各行の値は、結果指定に集合関数の指定がある場合には、グループを集合関数に適用した結果となり、列指定の場合には、グループ化列の値となり、グループ化関数の場合は、グループ化関数の値となります。なお、結果指定にGROUP BY句のAS句の列名を指定した場合は、対応するグループ化列、グループ化関数またはCASE式の値となります。
- 表式の結果が空であれば、単一行SELECT文の結果は、空の表となります。

— 表式の結果がNULL値の場合の扱いは以下のとおりです。

- 相手指定に標識変数が指定されていれば、その標識変数には-1が設定されます。
- 相手指定に標識変数が指定されていなければ、データ例外(NULL値、標識なし)となります。

— 表式の結果がNULL値でなく、相手指定に標識変数が指定されているならば、その標識変数には次の値が入ります。

- 相手指定のデータ型が固定長または可変長の文字列型の場合で、検索結果の長さが相手指定の文字列の最大長よりも長ければ、検索結果の文字数が設定されます。
- 固定長または可変長の各国語文字列型の場合も、同様に設定されます。

- そうでなければ、標識変数には0が設定されます。

ー FROM句は省略することができます。省略した場合はFROM句にシステム表の RDBII_ASSISTTABLEが指定されたものとみなします。

PARALLEL

ー PARALLELは、“並列指定”と呼びます。

ー 並列指定を指定した場合、表を並列に検索します。

実行オプション指定

ー 実行オプション指定は、占有モードまたは最大件数を指定します。

占有モード指定

- 占有モード指定は、問合せ指定を指定した場合に、問合せ指定のデータベース資源の占有の方法を指定します。
- 占有モードを指定したSQL文が読み込んだデータベース資源は、占有モード指定により、以下のようにデータベース資源を占有します。なお、更新する行は占有モード指定にかかわらず、非共有モードでトランザクションの終了までデータベース資源を占有します。

占有モード指定	占有の方法
EXCLUSIVE LOCK	非共有モードでトランザクション終了までデータベース資源を占有します。
SHARE LOCK	共有モードでトランザクション終了までデータベース資源を占有します。
FREE LOCK	共有モードでSQL文終了までデータベース資源を占有します。ただし、DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表にアクセスする場合には、データベース資源を占有しません。
NO LOCK	データベース資源を占有しません。

- 占有モードを指定したSQL文によって読み込まれた資源は、SET TRANSACTION文で指定された内容にかかわらず、以下のようになります。

占有モード指定	読込み水準
EXCLUSIVE LOCK	他のトランザクションによって占有されていない行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
SHARE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
FREE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。ただし、DSO定義でPRECEDENCE(1)が指定されたSEQUENTIAL構造の表にアクセスする場合には、他のトランザクションでの占有の有無に関わらず、SQL文実行時にコミット済みの行のデータを読み込みます。いずれの場合も、当該SQL文で読み込んだ行は、他のトランザクションに更新されることがあるため、同一トランザクションで再検索すると最新の結果を検索することができます。

占有モード指定	読込み水準
NO LOCK	他のトランザクションでどのような占有をされた資源でも参照することが可能なため、同一トランザクションで再検索すると最新の結果を検索することができます。

- 占有モードの単位は、R_LOCKがYESの場合は行単位となります。R_LOCKがNOの場合は、ページ単位またはDSI単位の占有になります。
- 動作環境パラメタにDSO_LOCKを指定したり、環境変数でRDBDSOを指定した場合、占有モードを指定したSQL文は実行できません。

最大件数指定

- 最大件数指定は、検索結果として取り出す件数を限定します。最大件数指定を指定すると、それ以上の行は返されません。
- 最大件数指定に指定された値が0以下の場合、検索結果として取り出す件数は0件となります。
- 最大件数指定に指定された値がNULLの場合、検索結果として取り出す件数は限定されず、すべての行が返されます。
- 値指定にUSERは指定できません。
- 指定値に指定できる値は、-2147483648から2147483647までの整数値です。
- 値指定に2以上の値を指定し、結果が複数行の場合は、例外(基数違反)となります。
- 値指定に変数指定または項目参照を指定した場合、データ型はINTEGER型またはSMALLINT型が指定可能です。
- 値指定に動的パラメタ指定が指定された場合のDESCRIBE情報は、INTEGER型になります。

その他の構文要素

その他の構文要素の説明は、“3.26 DECLARE CURSOR(カーソル宣言)”を参照してください。

使用例

例1

在庫数の合計および一番多い個数を求めます。その際、在庫数の合計はホスト変数H1に、また一番多い個数はホスト変数H2に格納します。

```
SELECT SUM(在庫量),MAX(在庫量) INTO :H1,:H2 FROM T1
```

例2

スキーマ“S1”の表“T1”を1件検索します。この時、非共有モードで行を占有して、他のトランザクションからの参照または更新を抑止します。

```
SELECT C1,C2 INTO :OUTVAL1,:OUTVAL2
    FROM S1.T1 WHERE C3 = :INVAL1
    WITH OPTION LOCK_MODE(EXCLUSIVE LOCK)
```

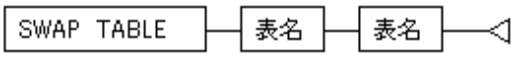
3.65 SWAP TABLE文(表交換文)

機能

指定された2つの表のデータを交換します。

データベース簡単運用の場合は利用できません。

記述形式



参照項番

なし。

権限

- 表交換文を実行できるのは、表の定義者だけです。

一般規則

- 指定した2つの表は、構成列数が同じであることが必要です。
- 指定した2つの表は、対応する列のデータ属性が同じであることが必要です。
- 指定した2つの表は、対応する列のNULL属性が同じであることが必要です。
- 指定した2つの表は、一意性制約定義が同じであることが必要です。
- 指定した2つの表は、表およびインデックスのDSIを割り付けているデータベーススペースの暗号化指定が同じであることが必要です。
- 指定された表は格納領域指定を指定して作成された表であってはいけません。
- 指定された表はALTER TABLE文(表定義変更文)が実行されていてはいけません。
- 指定された表は、必要なDSIの定義が完結していることが必要です。
- 指定された表はデフォルトデータベーススペースに作成された表であってはいけません。

表名

ー 交換する実表の名前を指定します。

使用例

例

表T1と表T2のデータを交換します。

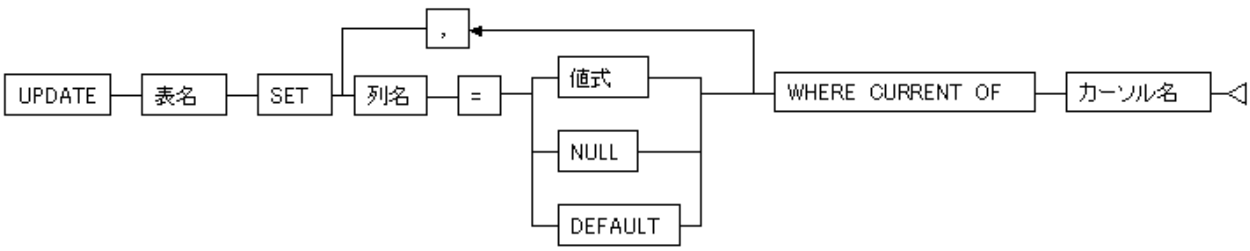
```
SWAP TABLE S1.T1 S1.T2
```

3.66 UPDATE文:位置づけ

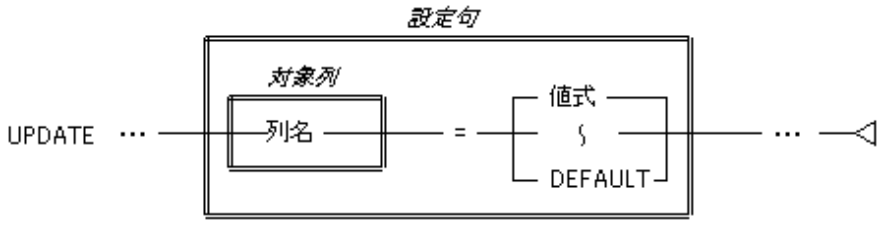
機能

カーソルによって位置づけられた1行を更新します。

記述形式



構文の構成



参照項番

- 値式 → “2.11 値式”

権限

- UPDATE文:位置づけを実行できるのは、処理対象の表のUPDATE権の保持者です。

一般規則

- カーソルは、開かれた状態であり、FETCH文により位置づけられていることが必要です。
- カーソルにより位置づけられている行が更新されます。
- UPDATE文にROWNUMは指定できません。

表名

— 更新する表の名前を指定します。

— 表名で指定する表は、カーソル宣言のカーソル指定に指定されていることが必要です。

— カーソル宣言のカーソル指定に導出表が記述されている場合、表名には導出表の元となる表を指定します。

— 表名にOBJECT構造の表を指定することはできません。

— 表名で指定した表が、“WITH CHECK OPTION”を指定したビュー表の場合、更新する値はビュー表の探索条件に対して真でなければなりません。そうでなければ、例外(WITH CHECK OPTIONに違反)となります。

設定句

— 設定句で指定した列は、値式、NULL値または既定値に変更されます。

— 設定句の値式に含まれる列指定が、UPDATE文:位置づけの対象となる表の列を指定しているとき、取り出される列の値は、更新される前の値です。

列名

- 列名は、更新する列の名前を指定します。
- 設定句に指定する列名は、表名で指定した表の列であることが必要です。また、同じ列名を複数回指定することはできません。

値式

- 設定句で指定する値式に集合関数を指定することはできません。
- 設定句で指定する値式のデータ型は、更新対象の列のデータ型に代入可能であることが必要です。
- 更新対象の列のデータ型が文字列型で、値式のデータ型が真数型、概数型または日時型の場合、値式のデータを文字列型に暗黙的に型変換して代入します。
- 更新対象の列のデータ型が真数型で、値式のデータ型が文字列型の場合、値式のデータを真数型に暗黙的に型変換して代入します。
- 更新対象の列のデータ型が概数型で、値式のデータ型が文字列型の場合、値式のデータを概数型に暗黙的に型変換して代入します。

- 更新対象の列のデータ型が日時型で、値式のデータ型が文字列型の場合、値式のデータを日時型に暗黙的に型変換して代入します。

列のデータ型が文字列型の場合:

値式のデータ型は文字列型、真数型、概数型または日時型であることが必要です。真数型/概数型または日時型が指定できるのは、値式を定数または変数指定で指定した場合のみです。

列のデータ型が各国語文字列型の場合:

値式のデータ型は各国語文字列型であることが必要です。

列のデータ型が真数型の場合:

値式のデータ型は真数型、概数型または文字列型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは符号つき数定数の形式になるように指定してください。

例1

UPDATE 会社管理.従業員表 SET 従業員番号 = '123' WHERE CURRENT OF CSR1
→ 文字列データ'123'を真数型に型変換して代入します。

例2

UPDATE 会社管理.従業員表 SET 従業員番号 = 'F123' WHERE CURRENT OF CSR1
→ 結果はエラーとなります。文字列データは、符号つき数定数の形式で指定する必要があります。

列のデータ型が概数型の場合:

値式のデータ型は真数型、概数型または文字列型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは符号つき数定数の形式になるように指定してください。

列のデータ型がDATE型の場合:

値式のデータ型は文字列型またはDATE型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは日時定数の形式になるように指定してください。

例1

UPDATE 会社管理.従業員表 SET 営業日 = '2007-01-02' WHERE CURRENT OF CSR1
→ 文字列データ'2007-01-02'をDATE型に型変換して代入します。

例2

UPDATE 会社管理.従業員表 SET 営業日 = '2007/1/2' WHERE CURRENT OF CSR1
→ 結果はエラーとなります。文字列データは、日時定数の形式で指定する必要があります。

列のデータ型がTIME型の場合:

値式のデータ型は文字列型またはTIME型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは日時定数の形式になるように指定してください。

列のデータ型がTIMESTAMP型の場合:

値式のデータ型は文字列型またはTIMESTAMP型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは日時定数の形式になるように指定してください。

列のデータ型が時間隔型の場合:

値式のデータ型は文字列型、真数型または時間隔型であることが必要です。文字列型、真数型が指定できるのは、値式を変数指定で指定した場合のみです。真数型を指定する場合、更新対象の列のデータ型が単一日時フィールドで指定された時間隔型でなければなりません。

列のデータ型がBLOB型の場合:

値式のデータ型はBLOB型であることが必要です。

NULL

― NULLを指定するときは、対応する列がNULL値を許していることが必要です。

DEFAULT

― DEFAULTを指定する場合は、表定義時にDEFAULT句を設定していないと、NULLが設定されます。表定義時にDEFAULT句を指定していればそのDEFAULT値が設定されます。

対象列

― 更新値が対象列に設定されるときの規則は次のようになります。

文字列型データの場合:

更新値の右側の空白を除くデータ長が対象列の長さと同じ場合は、文字データがそのまま設定されます。更新値の右側の空白を除くデータ長が対象列の長さよりも短い場合は、更新値が対象列の左側から設定され、対象列の残り部分には空白が設定されます。更新値の右側の空白を除くデータ長が対象列の長さよりも長い場合は、データ例外(文字データのトランケート)となります。

各国語文字列型の場合:

文字列型の場合と同じように設定されます。

真数型の場合:

対象列のデータ型に変換して設定されます。更新値の整数部が対象列の精度と位取りで表現できる場合は、対象列の精度と位取りに従って設定されます。上位の桁落ちが発生する場合は例外となり、下位の位が落ちる場合は切り捨てられます。

概数型のデータの場合:

対象列には、更新値の概数値が設定されます。

日時型のデータの場合:

対象列のデータ型に変換して設定されます。

時間隔型のデータの場合:

対象列のデータ型に変換して設定されます。ただし、時間隔型が年月の場合は年月のデータ型に、時間隔型が日時の場合は日時のデータ型にそれぞれ設定されます。

BLOB型のデータの場合:

対象列には、更新値のバイナリ値が設定されます。

カーソル名

― カーソルの名前を指定します。

― カーソル名は、同一コンパイル単位に含まれるカーソル宣言で定義されていることが必要です。

DESCRIBE情報について

― 設定句の値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、対応する列のデータ型になります。

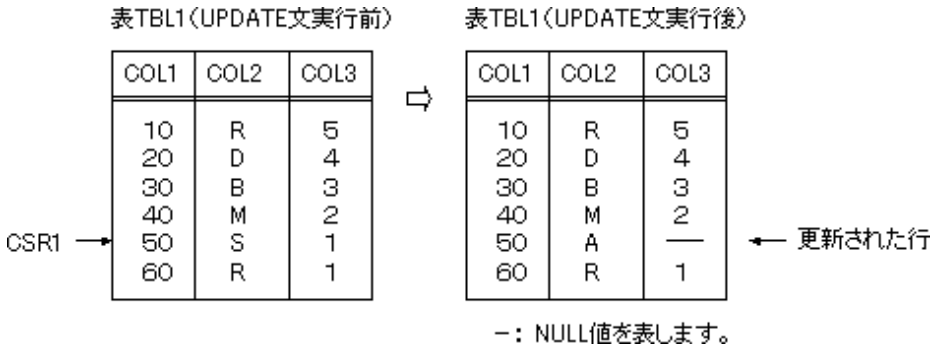
使用例

例

表TBL1のカーソルが位置づけられている行を更新する例を示します。

```
UPDATE TBL1
       SET COL2 = 'A',COL3 = NULL
       WHERE CURRENT OF CSR1
```

CSR1は、このUPDATE文と同一コンパイル単位内で定義されたカーソルとします。現在、カーソルCSR1が位置づけられている行を更新します。

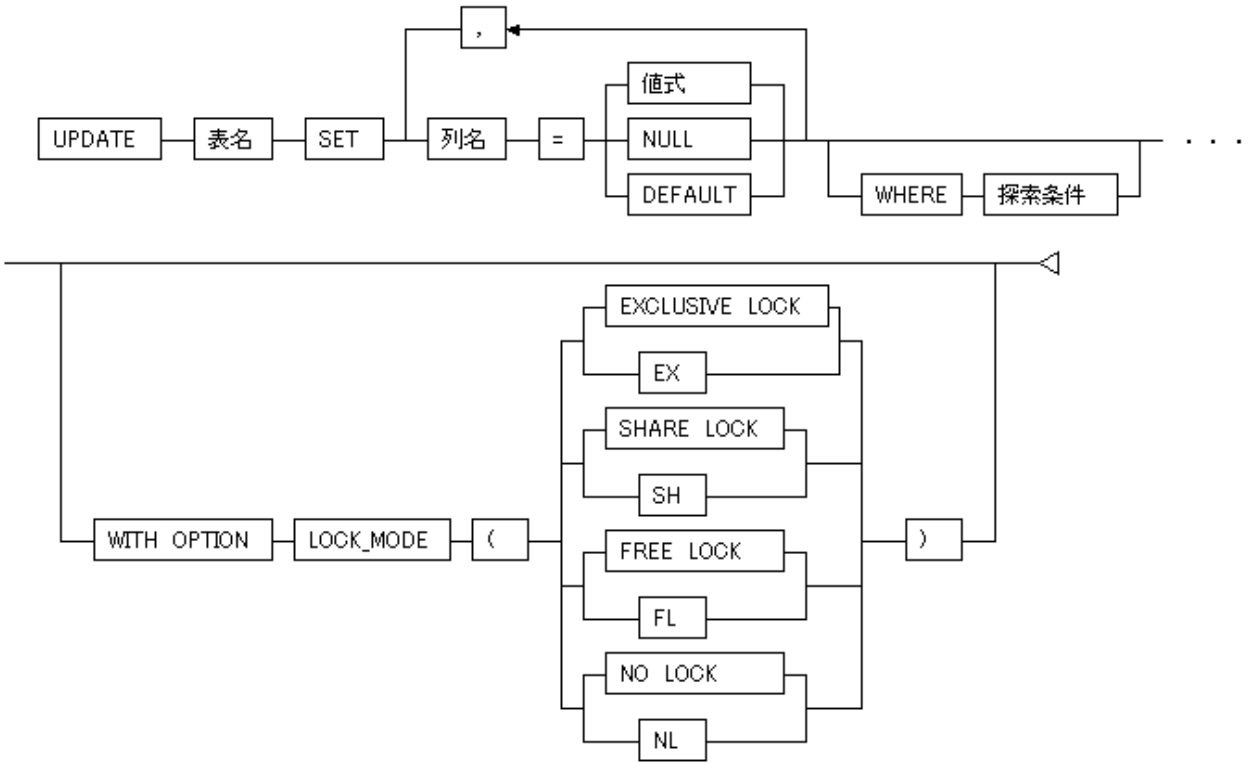


3.67 UPDATE文:探索

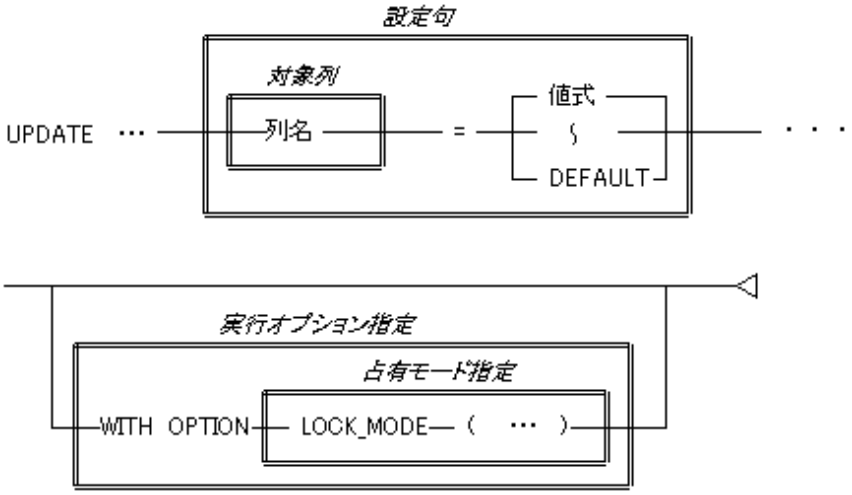
機能

探索条件を満たす列の値を更新します。

記述形式



構文の構成



参照項番

- 探索条件 → “2.13 探索条件”
- 値式 → “2.11 値式”

権限

- UPDATE文:探索を実行できるのは、処理対象の表のUPDATE権の保持者です。ただし、探索条件に副問合せが指定された場合、その副問合せに指定された表のSELECT権が必要です。

一般規則

- UPDATE文にROWNUMは指定できません。

表名

— 更新する表の名前を指定します。

— 表名の有効範囲はUPDATE文:探索の全体となります。

— 表名で指定した表が“WITH CHECK OPTION”を指定したビュー表の場合、更新する値はビュー表の探索条件に対して真でなければなりません。そうでなければ、例外(制約違反)となります。

— 表名で指定した表を、副問合せのFROM句に指定することはできません。このとき、表名で指定した表がビュー表の場合、そのビュー表に含まれる表が対象となります。

— 表名で指定した表に、トリガ事象にUPDATEを指定したトリガを定義している場合、被トリガSQL文中に指定した表名と同じ表名を副問合せのFROM句に指定することはできません。表名がビュー表の場合、そのビュー表に含まれる表が対象になります。

— 表名にOBJECT構造の表を指定することはできません。

設定句

— 設定句で指定した列は、値式、NULL値または既定値に変更されます。

— 設定句の値式に含まれる列指定が、UPDATE文:探索の対象となる表の列を指定しているとき、取り出される列の値は、更新される前の値です。

列名

- 列名は、更新する列の名前を指定します。
- 設定句に指定する列名は、表名で指定した表の列であることが必要です。また、同じ列名を複数回指定することはできません。

値式

- 設定句で指定する値式に集合関数指定を指定することはできません。
- 設定句で指定する値式のデータ型は、更新対象の列のデータ型に代入可能であることが必要です。
- 更新対象の列のデータ型が文字列型で、値式のデータ型が真数型、概数型または日時型の場合、値式のデータを文字列型に暗黙的に型変換して代入します。
- 更新対象の列のデータ型が真数型で、値式のデータ型が文字列型の場合、値式のデータを真数型に暗黙的に型変換して代入します。
- 更新対象の列のデータ型が概数型で、値式のデータ型が文字列型の場合、値式のデータを概数型に暗黙的に型変換して代入します。
- 更新対象の列のデータ型が日時型で、値式のデータ型が文字列型の場合、値式のデータを日時型に暗黙的に型変換して代入します。

列のデータ型が文字列型の場合:

値式のデータ型は文字列型、真数型、概数型または日時型であることが必要です。真数型/概数型または日時型が指定できるのは、値式を定数または変数指定で指定した場合のみです。

列のデータ型が各国語文字列型の場合:

値式のデータ型は各国語文字列型であることが必要です。

列のデータ型が真数型の場合:

値式のデータ型は真数型、概数型または文字列型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは符号つき数定数の形式になるように指定してください。

例1

UPDATE 会社管理.従業員表 SET 従業員番号 = '123'
→ 文字列データ'123'を真数型に型変換して代入します。

例2

UPDATE 会社管理.従業員表 SET 従業員番号 = 'F123'
→ 結果はエラーとなります。文字列データは、符号つき数定数の形式で指定する必要があります。

列のデータ型が概数型の場合:

値式のデータ型は真数型、概数型または文字列型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは符号つき数定数の形式になるように指定してください。

列のデータ型がDATE型の場合:

値式のデータ型は文字列型またはDATE型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは日時定数の形式になるように指定してください。

例1

UPDATE 会社管理.従業員表 SET 営業日 = '2007-01-02'
→ 文字列データ'2007-01-02'をDATE型に型変換して代入します。

例2

UPDATE 会社管理.従業員表 SET 営業日 = '2007/1/2'
→ 結果はエラーとなります。文字列データは、日時定数の形式で指定する必要があります。

列のデータ型がTIME型の場合:

値式のデータ型は文字列型またはTIME型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは日時定数の形式になるように指定してください。

列のデータ型がTIMESTAMP型の場合:

値式のデータ型は文字列型またはTIMESTAMP型であることが必要です。文字列型が指定できるのは、値式を定数または変数指定で指定した場合のみです。値式のデータ型が文字列型の場合、データは日時定数の形式になるように指定してください。

列のデータ型が時間隔型の場合:

値式のデータ型は文字列型、真数型または時間隔型であることが必要です。文字列型、真数型が指定できるのは、値式を変数指定で指定した場合のみです。真数型を指定する場合、更新対象の列のデータ型が単一日時フィールドで指定された時間隔型でなければなりません。

列のデータ型がBLOB型の場合:

値式のデータ型はBLOB型であることが必要です。

NULL

- NULLを指定するときは、対応する列がNULL値を許していることが必要です。

DEFAULT

- DEFAULTを指定する場合は、表定義時にDEFAULT句を設定していないと、NULLが設定されます。表定義時にDEFAULT句を指定していればそのDEFAULT値が設定されます。

対象列

ー 更新値が対象列に設定されるときの規則は次のようになります。

文字列型データの場合:

更新値の右側の空白を除くデータ長が対象列の長さと同じ場合は、文字データがそのまま設定されます。更新値の右側の空白を除くデータ長が対象列の長さよりも短い場合は、更新値が対象列の左側から設定され、対象列の残り部分には空白が設定されます。更新値の右側の空白を除くデータ長が対象列の長さよりも長い場合は、データ例外(文字データのトランケート)となります。

各国語文字列型の場合:

文字列型の場合と同じように設定されます。

真数型の場合:

対象列のデータ型に変換して設定されます。更新値の整数部が対象列の精度と位取りで表現できる場合は、対象列の精度と位取りに従って設定されます。上位の桁落ちが発生する場合は例外事象となり、下位の位が落ちる場合は切り捨てられます。

概数型のデータの場合:

対象列には、更新値の概数値が設定されます。

日時型のデータの場合:

対象列のデータ型に変換して設定されます。

時間隔型のデータの場合:

対象列のデータ型に変換して設定されます。ただし、時間隔型が年月の場合は年月のデータ型に、時間隔型が日時の場合は日時のデータ型にそれぞれ設定されます。

BLOB型のデータの場合:

対象列には、更新値のバイナリ値が設定されます。

探索条件

ー 探索条件を指定した場合、探索条件が真となる行が更新の対象となります。

ー 探索条件を省略した場合、表のすべての行が更新の対象となります。

ー 探索条件によって真となる行が存在しなかった場合、または表に行が存在しなかった場合、例外(データなし)となります。

ー 探索条件の副問合せに指定した導出表にはORDER BY句を指定することはできません。

DESCRIBE情報について

― 設定句の値式に動的パラメタ指定が指定された場合のDESCRIBE情報は、対応する列のデータ型になります。

実行オプション指定

― 実行オプション指定は、占有モードを指定します。

占有モード指定

― 占有モード指定は、問合せ指定を指定した場合に、問合せ指定のデータベース資源の占有の方法を指定します。

― 占有モードを指定したSQL文が読み込んだデータベース資源は、占有モード指定により、以下のようにデータベース資源を占有します。なお、更新する行は占有モード指定にかかわらず、非共有モードでトランザクションの終了までデータベース資源を占有します。

占有モード指定	占有の方法
EXCLUSIVE LOCK	非共有モードでトランザクション終了までデータベース資源を占有します。
SHARE LOCK	共有モードでトランザクション終了までデータベース資源を占有します。
FREE LOCK	共有モードでSQL文終了までデータベース資源を占有します。
NO LOCK	データベース資源を占有しません。

― 占有モードを指定したSQL文によって読み込まれた資源は、SET TRANSACTION文で指定された内容にかかわらず、以下のようになります。

占有モード指定	読込み水準
EXCLUSIVE LOCK	他のトランザクションによって占有されていない行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
SHARE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。当該SQL文で読み込んだ行は、トランザクション終了まで他のトランザクションに更新されることはないため、一度読み込んだ行は他のトランザクションによって更新されないことが保証されます。
FREE LOCK	他のトランザクションによって占有されていないか、または共有モードで資源を占有されている行を読み込みます。当該SQL文で読み込んだ行は、他のトランザクションに更新されることがあるため、同一トランザクションで再検索すると最新の結果を検索することができます。
NO LOCK	他のトランザクションでどのような占有をされた資源でも参照することが可能なため、同一トランザクションで再検索すると最新の結果を検索することができます。

― 占有モードの単位は、R_LOCKがYESの場合は行単位となります。R_LOCKがNOの場合は、ページ単位またはDSI単位の占有になります。

― 動作環境パラメタにDSO_LOCKを指定したり、環境変数でRDBDSOを指定した場合、占有モードを指定したSQL文は実行できません。

使用例

例1

表TBL1の探索条件が真となる行を更新する例を示します。

```
UPDATE TBL1
       SET COL1 = 100, COL2 = NULL
       WHERE COL1 = 20 OR COL3 = 5
```

表TBL1の行のうち探索条件を満たす行の列COL1と列COL2の値が、それぞれ100とNULL値になります。このとき、COL2はNULL値が許されない列であってはいけません。

表TBL1(UPDATE文実行前)

COL1	COL2	COL3
10	R	5
20	D	4
30	B	3
40	M	2
50	S	1
60	R	1

⇨

表TBL1(UPDATE文実行後)

COL1	COL2	COL3	
100	−	5	← 更新された行
100	−	4	← 更新された行
30	B	3	
40	M	2	
50	S	1	
60	R	1	

−: NULL値を表します。

例2

他のアプリケーションの処理待ちを少なくする例を示します。

```
UPDATE 在庫管理.在庫表 SET COL1 = 2 WHERE COL1 = 20
       WITH OPTION LOCK_MODE(NO LOCK)
```

例3

デッドロックの抑止を考慮した使い方の例を示します。インデックスの構成列はCOL1とします。

```
UPDATE 在庫管理.在庫表 SET COL1 = 2 WHERE COL1 = 20
       WITH OPTION LOCK_MODE(EXCLUSIVE LOCK)
```

第4章 動的SQL文

動的SQLは、アプリケーションの実行時に、SQL文を動的に指定することを可能にします。

ホストプログラムに直接SQL文を記述しないので、動的SQL文を使用すると、対話的に入力したSQL文を実行したり、ある情報から生成したSQL文を実行するアプリケーションを作成することができます。

4.1 動的SQL文の概要

動的SQLは、アプリケーションの実行時に、SQL文を動的に指定することを可能にします。

ホストプログラムに直接SQL文を記述しないので、動的SQL文を使用すると、対話的に入力したSQL文を実行したり、ある情報から生成したSQL文を実行するアプリケーションを作成することができます。

アプリケーションの実行時に入力するSQL文を準備可能文といいます。準備可能文として扱うことのできるSQL文を以下に示します。なお、動的SELECT文は、カーソル宣言のカーソル指定に対応し、動的単一行SELECT文は、単一行SELECT文に対応します。

- 動的SELECT文
- 動的単一行SELECT文
- DELETE文:探索
- UPDATE文:探索
- INSERT文
- 準備可能動的DELETE文:位置づけ
- 準備可能動的UPDATE文:位置づけ
- スキーマ定義文
  - CREATE SCHEMA文(スキーマ定義)
  - CREATE TABLE文(表定義)
  - CREATE VIEW文(ビュー定義)
  - CREATE PROCEDURE文(プロシジャルーチン定義)
  - CREATE FUNCTION文(ファンクションルーチン定義)
  - CREATE SEQUENCE文(順序定義)
  - CREATE INDEX文(インデックス定義)
  - CREATE TRIGGER文(トリガ定義)
- スキーマ操作文
  - DROP SCHEMA文(スキーマ削除文)
  - DROP TABLE文(表削除文)
  - ALTER TABLE文(表定義変更文)
  - DROP VIEW文(ビュー削除文)
  - DROP PROCEDURE文(プロシジャルーチン削除文)
  - DROP FUNCTION文(ファンクションルーチン削除文)
  - DROP SEQUENCE文(順序削除文)
  - DROP INDEX文(インデックス削除文)
  - DROP TRIGGER文(トリガ削除文)

  - SWAP TABLE文(表交換文)
- 格納構造定義文
  - CREATE DSO文(表のDSO定義文)
  - CREATE DSO文(インデックスのDSO定義文)
  - CREATE DSI文(表のDSI定義文)
  - CREATE DSI文(インデックスのDSI定義文)
  - CREATE SCOPE文(スコープ定義文)
- 格納構造操作文
  - DROP DSO文(DSO削除文)
  - DROP DSI文(DSI削除文)
  - ALTER DSI文(DSI変更文)
  - DROP SCOPE文(スコープ削除文)
  - APPLY SCOPE文(スコープ適用文)
  - RELEASE SCOPE文(スコープ解除文)
- 利用者制御文
  - CREATE USER文(利用者定義文)
  - DROP USER文(利用者削除文)
  - ALTER USER文(利用者変更文)
- アクセス制御文
  - CREATE ROLE文(ロール定義文)
  - DROP ROLE文(ロール削除文)
  - GRANT文
  - REVOKE文
- システム制御文
  - SET SYSTEM PARAMETER文
- CALL文

準備可能文はPREPARE文で準備します。PREPARE文で準備されたSQL文を被準備文といいます。被準備文はPREPARE文で指定したSQL文識別子によって識別されます。被準備文が動的SELECT文の場合、動的カーソル宣言でカーソルを定義し、動的OPEN文、動的FETCH文、動的CLOSE文、動的DELETE文:位置づけおよび動的UPDATE文:位置づけで操作します。また、準備可能動的DELETE文:位置づけおよび準備可能動的UPDATE文:位置づけでも操作します。被準備文が動的単一行SELECT文、DELETE文:探索、UPDATE文:探索、INSERT文、準備可能動的DELETE文:位置づけ、または準備可能動的UPDATE文:位置づけの場合は、EXECUTE文で実行します。

被準備文に含まれる動的パラメタ指定の値の設定や行の値の取出しには、SQL記述子域、SQLDA構造体または引数を使用します。DESCRIBE文により被準備文の選択リストの情報や動的パラメタ指定の情報を、SQL記述子域またはSQLDA構造体に設定することができます。SQL記述子域の情報は、DESCRIPTOR取得文で取得し、DESCRIPTOR設定文で設定することができます。動的OPEN文およびEXECUTE文では、SQL記述子域、SQLDA構造体または引数を介して動的パラメタ指定の値を指定することができます。動的FETCH文および動的単一行SELECT文を被準備文とするEXECUTE文では、SQL記述子域、SQLDA構造体または引数を介して行の値を取り出すことができます。また、被準備文がDELETE文:探索、UPDATE文:探索、INSERT文、準備可能動的DELETE文:位置づけ、または準備可能動的UPDATE文:位置づけで、動的パラメタ指定を含まないときは、EXECUTE IMMEDIATE文で直接実行することができます。

さらに、被準備文がスキーマ定義文、スキーマ操作文、格納構造定義文、格納構造操作文の場合は、EXECUTE文またはEXECUTE IMMEDIATE文で実行することができます。

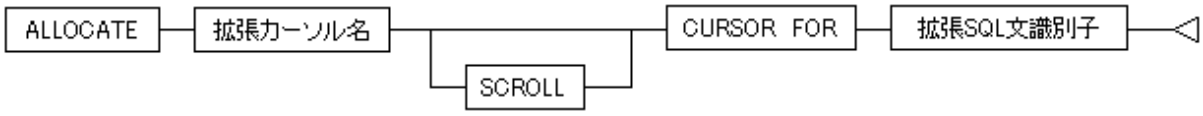
通常、動的SQLで指定するカーソル名やSQL文識別子は、識別子として静的に指定します。しかし、操作するカーソルの数や実行するSQL文の数が動的に変化する場合は、カーソル名やSQL文識別子を静的に指定することが困難です。この場合は、カーソル名やSQL文識別子をホスト変数で指定し、アプリケーションの実行時にカーソル名やSQL文識別子の値を動的に指定します。ホスト変数で指定するカーソル名を拡張カーソル名、SQL文識別子を拡張SQL文識別子といいます。拡張カーソル名でカーソルを定義する場合は、ALLOCATE CURSOR文を実行します。拡張カーソル名や拡張SQL文識別子を使用すると、動的OPEN文などのカーソル操作文やPREPARE文をカーソル名やSQL文識別子ごとに指定する必要がなくなり、共通化できます。このため、アプリケーションのコーディング量を削減する効果も期待できます。

4.2 ALLOCATE CURSOR文

機能

PREPARE文によって用意された文に基づいて、拡張カーソル名で指定したカーソルを定義します。

記述形式



参照項番

- SCROLL → “3.26 DECLARE CURSOR(カーソル宣言)”
- 日本語文字列 → “2.1.3 トークン”

一般規則

- 拡張SQL文識別子に対応する被準備文は、動的SELECT文であることが必要です。
- ALLOCATE CURSOR文で定義したカーソルは、拡張SQL文識別子に対応するDEALLOCATE PREPARE文が実行されるか、拡張SQL文識別子に対応するPREPARE文が再度実行されるまで有効です。

拡張カーソル名

- カーソルの名前を指定するための変数を指定します。
- 拡張カーソル名は、文字列型の埋込み変数で指定します。拡張カーソル名に指定した埋込み変数の値がカーソルの名前になります。拡張カーソル名の値には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。英字の大文字と小文字は区別されます。
- 拡張カーソル名の値に空白を含む場合、前後の空白を取り除いた値がカーソルの名前になります。
- 拡張カーソル名の値に指定したカーソルと同じ名前のカーソルが別のALLOCATE CURSOR文で定義されていないことが必要です。
- カーソル宣言や動的カーソル宣言で指定したカーソル名と、ALLOCATE CURSOR文で定義した拡張カーソル名の値が同じでも、異なるカーソルとして区別されます。

拡張SQL文識別子

- 拡張カーソル名の値に指定したカーソルと対応づける被準備文の名前を指定するための変数を指定します。
- 拡張SQL文識別子は、文字列型の埋込み変数で指定します。拡張SQL文識別子に指定した埋込み変数の値が被準備文の名前になります。拡張SQL文識別子の値には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。英字の大文字と小文字は区別されます。
- 拡張SQL文識別子の値に空白を含む場合は、前後の空白を取り除いた値が被準備文の名前になります。
- 拡張SQL文識別子で指定した被準備文の名前は、PREPARE文で定義されていることが必要です。
- SQL文識別子と拡張SQL文識別子の値が同じでも、異なる被準備文として区別されます。

使用例

例

ALLOCATE CURSOR文の拡張カーソル名でカーソルを定義します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CURVAR[4];
     CHAR CMDVAR[4];
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
          :
strcpy(CURVAR,"CU1");                                          (1)
strcpy(CMDVAR,"CMD");                                          (2)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL");    (3)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;                        (4)
EXEC SQL ALLOCATE :CURVAR CURSOR FOR :CMDVAR;                  (5)
```

(1) ホスト変数CURVARにカーソルの名前CU1を設定します。

(2) ホスト変数CMDVARに被準備文の名前CMDを設定します。

(3) ホスト変数CMDAREAに動的SELECT文を設定します。

(4) ホスト変数CMDAREAが示すSQL文を拡張SQL文識別子CMDVARの値CMDに対応づけます。

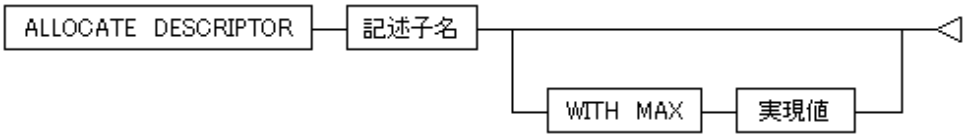
(5) 拡張カーソル名CURVARの値CU1と拡張SQL文識別子CMDVARの値CMDを対応づけます。

4.3 ALLOCATE DESCRIPTOR文

機能

SQL記述子域を割り当てます。

記述形式



一般規則

記述子名

- SQL記述子域の名前を指定します。SQL構造体変数の変数名は指定できません。
- 記述子名は文字列定数または文字列型の埋込み変数で指定します。
- すでに同名の記述子名が、割り当てられていてはいけません。

WITH MAX句

- 割り当てられたSQL記述子域は、実現値に指定された個数の項目記述子域を所有します。
- 実現値は真数定数または2進の精度を持つ埋込み変数で指定します。
- WITH MAX句を省略した場合、100が指定されたものとみなします。
- 実現値は1から32767までの値であることが必要です。

使用例

例

50個の項目記述子域を持つSQL記述子域を割り当てます。SQL記述子域の記述子名をDESC1とします。

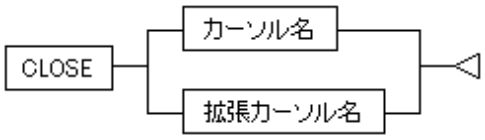
```
ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50
```

4.4 CLOSE文(動的CLOSE文)

機能

動的カーソル宣言またはALLOCATE CURSOR文で指定したカーソルを閉じます。

記述形式



一般規則

- CLOSE文で指定するカーソルは、開かれた状態であることが必要です。閉じられた状態ならば、例外(不当カーソル状態)となります。

カーソル名または拡張カーソル名

- カーソルの名前を指定します。
- 動的カーソル宣言で定義したカーソルを閉じる場合は、カーソル名を指定します。
- ALLOCATE CURSOR文で定義したカーソルを閉じる場合は、拡張カーソル名を指定し、拡張カーソル名の値にカーソルの名前を指定します。
- 拡張カーソル名は、文字列型の埋込み変数で指定します。拡張カーソル名の値に空白を含む場合は、前後の空白を取り除いた値がカーソルの名前になります。
- カーソル名を指定した場合、カーソル名は、同一コンパイル単位に含まれる動的カーソル宣言で定義されていることが必要です。
- カーソル名と拡張カーソル名の値が同じでも、異なるカーソルとして区別されます。

使用例

例1

動的OPEN文によりオープンしたカーソルを、動的CLOSE文によりクローズします。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                       (2)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = 38");(3)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                        (4)
EXEC SQL DESCRIBE OUTPUT CMD USING SQL DESCRIPTOR 'DESC1';                 (5)
                :
  (SQL記述子域に必要な情報を設定)
                :
```

```
EXEC SQL OPEN CU1;                                                    (6)
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR 'DESC1';                       (7)
                :
EXEC SQL CLOSE CU1;                                                   (8)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREAに動的SELECT文を設定します。

(4) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(5) SQL文識別子CMDに対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(6) 動的OPEN文によりカーソルをオープンします。

(7) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(8) 動的CLOSE文によりカーソルをクローズします。

例2

拡張カーソル名を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CURVAR[4];
     CHAR CMDVAR[4];
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
               :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                     (1)
strcpy(CURVAR,"CU1");                                                 (2)
strcpy(CMDVAR,"CMD");                                                 (3)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = 38");(4)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;                               (5)
EXEC SQL ALLOCATE :CURVAR CURSOR FOR :CMDVAR;                         (6)
EXEC SQL DESCRIBE OUTPUT :CMDVAR USING SQL DESCRIPTOR 'DESC1';        (7)
               :
  (SQL記述子域に必要な情報を設定)
               :
EXEC SQL OPEN :CURVAR;                                                (8)
EXEC SQL FETCH :CURVAR INTO SQL DESCRIPTOR 'DESC1';                   (9)
               :
EXEC SQL CLOSE :CURVAR;                                              (10)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CURVARにカーソルの名前CU1を設定します。

(3) ホスト変数CMDVARに被準備文の名前CMDを設定します。

(4) ホスト変数CMDAREAに動的SELECT文を設定します。

(5) ホスト変数CMDAREAが示すSQL文を拡張SQL文識別子CMDVARの値CMDに対応づけます。

(6) 拡張カーソル名CURVARの値CU1と拡張SQL文識別子CMDVARの値CMDを対応づけます。

(7) 拡張SQL文識別子CMDVARの値CMDに対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(8) 動的OPEN文によりカーソルをオープンします。

(9) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(10) 動的CLOSE文によりカーソルをクローズします。

4.5 DEALLOCATE DESCRIPTOR文

機能

SQL記述子域を解放します。

記述形式

一般規則

・ DEALLOCATE DESCRIPTOR文により、記述子名で指定されたSQL記述子域を解放します。

記述子名

ー 解放するSQL記述子域の名前を指定します。SQL構造体変数の変数名は指定できません。

ー 記述子名は、文字列定数または文字列型の埋込み変数で指定します。

ー 指定した記述子名のSQL記述子域がALLOCATE DESCRIPTOR文ですでに割り当てられていることが必要です。

ー 同一の記述子名のUSING記述子が動的OPEN文に指定されている場合は、動的OPEN文に指定しているカーソルは閉じられた状態であることが必要です。

使用例

例

DESC1の記述子名のSQL記述子域を割り当てた後、その領域を解放します。

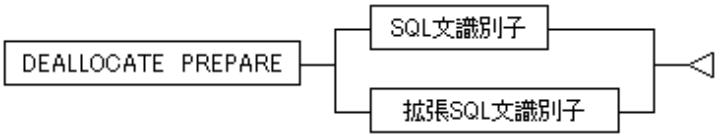
```
ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50
DEALLOCATE DESCRIPTOR 'DESC1'
```

4.6 DEALLOCATE PREPARE文

機能

PREPARE文で準備されているSQL文を解放します。

記述形式



一般規則

・ DEALLOCATE PREPARE文で指定しているSQL文識別子または拡張SQL文識別子と対応する被準備文が動的SELECT文の場合、同じSQL文識別子または拡張SQL文識別子に対応するカーソルは、閉じられた状態であることが必要です。

・ DEALLOCATE PREPARE文を実行することにより、SQL文識別子または拡張SQL文識別子に対応する被準備文は解放されます。

・ 拡張SQL文識別子に対応する被準備文が動的SELECT文の場合、拡張SQL文識別子に対応するALLOCATE CURSOR文で定義したカーソルも同時に解放されます。

- SQL文識別子または拡張SQL文識別子と対応する被準備文が動的SELECT文の場合、SQL文識別子または拡張SQL文識別子に対応するカーソルを参照している準備可能動的DELETE文:位置づけおよび準備可能動的UPDATE文:位置づけの被準備文も同時に解放されます。
- PREPARE文を実行後、EXECUTE文を実行した後は、DEALLOCATE PREPARE文により、SQL文の実行で使用した資源を回収してください。
  何らかの原因で通信切断が発生したコネクションで、DEALLOCATE PREPARE文を実行しないでアプリケーションの処理を継続すると、次にSQL文を実行した契機で、アプリケーションに通信切断のエラーが返却されます。

SQL文識別子または拡張SQL文識別子

- 解放する被準備文の名前を指定します。
- 解放する被準備文を準備したPREPARE文がSQL文識別子を指定している場合は、SQL文識別子を指定します。
- 解放する被準備文を準備したPREPARE文が拡張SQL文識別子を指定している場合は、拡張SQL文識別子を指定し、拡張SQL文識別子の値に被準備文の名前を指定します。
- 拡張SQL文識別子は、文字列型の埋込み変数で指定します。拡張SQL文識別子の値に空白を含む場合は、前後の空白を取り除いた値が被準備文の名前になります。
- SQL文識別子を指定した場合、SQL文識別子は、同一コンパイル単位に含まれるPREPARE文で定義されていることが必要です。
- SQL文識別子と拡張SQL文識別子の値が同じでも、異なる被準備文として区別されます。

使用例

例1

DELETE文:探索をPREPARE文で準備し、EXECUTE文で実行した後、そのSQL文識別子に対応する被準備文を解放します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
              :
strcpy(CMDAREA.sqlvar,"DELETE FROM S1.TBL WHERE COL1 = 39");   (1)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                            (2)
EXEC SQL EXECUTE CMD;                                          (3)
EXEC SQL DEALLOCATE PREPARE CMD;                               (4)
```

(1) ホスト変数CMDAREAにDELETE文:探索を設定します。

(2) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(3) 被準備文を実行します。

(4) SQL文識別子CMDに対応する被準備文を解放します。

例2

拡張SQL文識別子を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CMDVAR[4];
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
              :
strcpy(CMDVAR,"CMD");                                          (1)
strcpy(CMDAREA.sqlvar,"DELETE FROM S1.TBL WHERE COL1 = 39");   (2)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;                        (3)
EXEC SQL EXECUTE :CMDVAR;                                      (4)
EXEC SQL DEALLOCATE PREPARE :CMDVAR;                           (5)
```

(1) ホスト変数CMDVARに被準備文の名前CMDを設定します。

(2) ホスト変数CMDAREAにDELETE文:探索を設定します。

(3) ホスト変数CMDAREAが示すSQL文を拡張SQL文識別子CMDVARの値CMDに対応づけます。

(4) 被準備文を実行します。

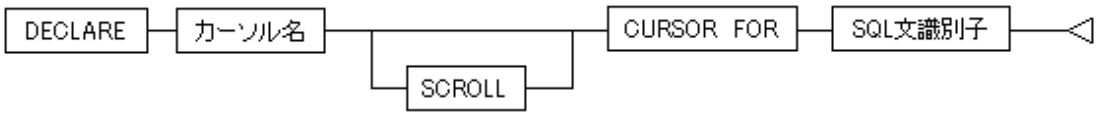
(5) 拡張SQL文識別子CMDVARの値CMDに対応する被準備文を解放します。

4.7 DECLARE CURSOR(動的カーソル宣言)

機能

PREPARE文によって用意された文に基づいて、カーソルを定義します。

記述形式



参照項番

- SCROLL → “3.26 DECLARE CURSOR(カーソル宣言)”
- 日本語文字列 → “2.1.3 トークン”

一般規則

- 同じSQL文識別子を指定したPREPARE文によって用意された文は、動的SELECT文であることが必要です。

カーソル名

- カーソルの名前を指定します。
- カーソル名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- カーソル名は、コンパイル単位内でほかのカーソル宣言、または動的カーソル宣言のカーソル名と異なっていることが必要です。
- カーソル名とALLOCATE CURSOR文で指定した拡張カーソル名の値が同じでも、異なるカーソルとして区別されます。

SQL文識別子

- PREPARE文で用意するSQL文の識別子を指定します。
- SQL文識別子には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

使用例

例

被準備文が動的SELECT文の場合、カーソル宣言における問合せ式の部分にSQL文識別子を指定します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL DECLARE CU1 CURSOR FOR CMD;                              (1)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL");       (2)
```

```
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                    (3)
```

(1) カーソルCU1が、SQL文識別子CMDで指定された問合せ式によって定義されます。

(2) ホスト変数CMDAREAに動的SELECT文を設定します。

(3) ホスト変数CMDAREAが示す動的SELECT文をSQL文識別子CMDに対応づけます。

これは、次のSQL文と同じです。

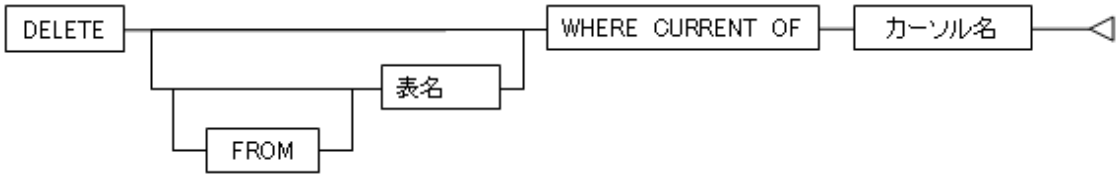
```
EXEC SQL DECLARE CU1 CURSOR FOR SELECT COL1,COL2,COL3 FROM S1.TBL;
```

4.8 DELETE文(準備可能動的DELETE文:位置づけ)

機能

動的カーソル宣言またはALLOCATE CURSOR文で指定したカーソルによって位置づけられた1行を削除します。

記述形式



参照項番

- 区切り識別子 → “2.1.3 トークン”

権限

- 準備可能動的DELETE文:位置づけを実行できるのは、処理対象の表のDELETE権の保持者です。

一般規則

- 準備可能動的DELETE文:位置づけは、このSQL文をホスト変数に格納し、PREPARE文で実行の準備を行った後、EXECUTE文により実行します。また、このSQL文をホスト変数に格納した後、EXECUTE IMMEDIATE文で実行します。埋込みSQL文として、実行することはできません。必ず、準備してから実行する必要があります。
- 表名、カーソル名および上記以外の一般規則は、“3.28 DELETE文:位置づけ”を参照してください。

表名

— 行を削除する表の名前を指定します。

— 表名を省略した場合、対象の表は、カーソルに対応する被準備文の内容である動的SELECT文のFROM句で指定された表になります。

カーソル名

— 動的カーソル宣言で定義したカーソル名、またはALLOCATE CURSOR文の拡張カーソル名で定義したカーソルの名前を指定します。

— カーソル名に英字の小文字を指定した場合、対応する英字の大文字に変換されます。英字の大文字と小文字を区別する場合は、区切り識別子で指定します。詳細は、“2.1.3 トークン”を参照してください。

ー 準備可能動的DELETE文:位置づけに指定したカーソル名は、以下のいずれかと同じであることが必要です。ただし、両方と同じである場合は、例外(あいまいなカーソル名)となります。

- 同一コンパイル単位に含まれる動的カーソル宣言のカーソル名
- 同一セションで実行されたALLOCATE CURSOR文の拡張カーソル名の値

使用例

例1

CU1のカーソルで位置づけられた行を削除します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR CMDAREA1[100];
     VARCHAR CMDAREA2[100];
EXEC SQL END DECLARE SECTION;
               :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                    (1)
EXEC SQL DECLARE CU1 CURSOR FOR CMD1;                                (2)
strcpy(CMDAREA1.sqlvar,"SELECT COL1,COL2 FROM S1.TBL WHERE COL4 = 38");(3)
CMDAREA1.sqllen = strlen(CMDAREA1.sqlvar);
strcpy(CMDAREA2.sqlvar,"DELETE FROM S1.TBL WHERE CURRENT OF CU1");   (4)
CMDAREA2.sqllen = strlen(CMDAREA2.sqlvar);
EXEC SQL PREPARE CMD1 FROM :CMDAREA1;                                (5)
EXEC SQL DESCRIBE OUTPUT CMD1 USING SQL DESCRIPTOR 'DESC1';          (6)
               :
  (SQL記述子域に必要な情報を設定)
               :
EXEC SQL OPEN CU1;                                                   (7)
EXEC SQL PREPARE CMD2 FROM :CMDAREA2;                                (8)
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR 'DESC1';                      (9)
EXEC SQL EXECUTE CMD2;                                               (10)
EXEC SQL CLOSE CU1;                                                  (11)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREA1に動的SELECT文を設定します。

(4) ホスト変数CMDAREA2に準備可能動的DELETE文:位置づけを設定します。

(5) ホスト変数CMDAREA1が示すSQL文をSQL文識別子CMD1に対応づけます。

(6) SQL文識別子CMD1に対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(7) 動的OPEN文によりカーソルをオープンします。

(8) ホスト変数CMDAREA2が示すSQL文をSQL文識別子CMD2に対応づけます。

(9) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(10) 準備可能動的DELETE文:位置づけにより位置づけ行を削除します。

(11) 動的CLOSE文によりカーソルをクローズします。

例2

拡張カーソル名を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CURVAR[4];
     CHAR CMDVAR1[5];
     CHAR CMDVAR2[5];
     VARCHAR CMDAREA1[100];
     VARCHAR CMDAREA2[100];
```

```
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
strcpy(CURVAR,"CU1");                                                      (2)
strcpy(CMDVAR1,"CMD1");                                                    (3)
strcpy(CMDVAR2,"CMD2");                                                    (4)
strcpy(CMDAREA1.sqlvar,"SELECT COL1,COL2 FROM S1.TBL WHERE COL4 = 38");(5)
CMDAREA1.sqllen = strlen(CMDAREA1.sqlvar);
strcpy(CMDAREA2.sqlvar,"DELETE FROM S1.TBL WHERE CURRENT OF CU1");      (6)
CMDAREA2.sqllen = strlen(CMDAREA2.sqlvar);
EXEC SQL PREPARE :CMDVAR1 FROM :CMDAREA1;                                  (7)
EXEC SQL ALLOCATE :CURVAR CURSOR FOR :CMDVAR1;                             (8)
EXEC SQL DESCRIBE OUTPUT :CMDVAR1 USING SQL DESCRIPTOR 'DESC1';            (9)
                :
  (SQL記述子域に必要な情報を設定)
                :
EXEC SQL OPEN :CURVAR;                                                     (10)
EXEC SQL PREPARE :CMDVAR2 FROM :CMDAREA2;                                  (11)
EXEC SQL FETCH :CURVAR INTO SQL DESCRIPTOR 'DESC1';                        (12)
EXEC SQL EXECUTE :CMDVAR2;                                                 (13)
EXEC SQL CLOSE :CURVAR;                                                    (14)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CURVAR1にカーソルの名前CU1を設定します。

(3) ホスト変数CMDVAR1に被準備文の名前CMD1を設定します。

(4) ホスト変数CMDVAR2に準備可能動的DELETE文:位置づけの名前CMD2を設定します。

(5) ホスト変数CMDAREA1に動的SELECT文を設定します。

(6) ホスト変数CMDAREA2に準備可能動的DELETE文:位置づけを設定します。

(7) ホスト変数CMDAREA1が示すSQL文を拡張SQL文識別子CMDVAR1の値CMD1に対応づけます。

(8) 拡張カーソル名CURVAR1の値CU1と拡張SQL文識別子CMDVAR1の値CMD1を対応づけます。

(9) 拡張SQL文識別子CMDVAR1の値CMD1に対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(10) 動的OPEN文によりカーソルをオープンします。

(11) ホスト変数CMDAREA2が示すSQL文を拡張SQL文識別子CMDVAR2の値CMD2に対応づけます。

(12) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(13) 準備可能動的DELETE文:位置づけにより位置づけ行を削除します。

(14) 動的CLOSE文によりカーソルをクローズします。

4.9 DELETE文(動的DELETE文:位置づけ)

機能

動的カーソル宣言またはALLOCATE CURSOR文で指定したカーソルによって位置づけられた1行を削除します。

記述形式

権限

- 動的DELETE文:位置づけを実行できるのは、処理対象の表のDELETE権の保持者です。

一般規則

- カーソル名または拡張カーソル名以外の規則は、“3.28 DELETE文:位置づけ”の一般規則を参照してください。

カーソル名または拡張カーソル名

- カーソルの名前を指定します。
- 動的カーソル宣言で定義したカーソルの行を削除する場合は、カーソル名を指定します。
- ALLOCATE CURSOR文で定義したカーソルの行を削除する場合は、拡張カーソル名を指定し、拡張カーソル名の値にカーソルの名前を指定します。
- 拡張カーソル名は、文字列型の埋込み変数で指定します。拡張カーソル名の値に空白を含む場合は、前後の空白を取り除いた値がカーソルの名前になります。
- カーソル名を指定した場合、カーソル名は、同一コンパイル単位に含まれる動的カーソル宣言で定義されていることが必要です。
- カーソル名と拡張カーソル名の値が同じでも、異なるカーソルとして区別されます。

使用例

例1

CU1のカーソルで位置づけられた行を削除します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                       (2)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = 38");(3)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                        (4)
EXEC SQL DESCRIBE OUTPUT CMD USING SQL DESCRIPTOR 'DESC1';                 (5)
                :
  (SQL記述子域に必要な情報を設定)
                :
EXEC SQL OPEN CU1;                                                         (6)
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR 'DESC1';                            (7)
EXEC SQL DELETE FROM S1.TBL WHERE CURRENT OF CU1;                          (8)
EXEC SQL CLOSE CU1;                                                        (9)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREAに動的SELECT文を設定します。

(4) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(5) SQL文識別子CMDに対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(6) 動的OPEN文によりカーソルをオープンします。

(7) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(8) 動的DELETE文:位置づけにより位置づけ行を削除します。

(9) 動的CLOSE文によりカーソルをクローズします。

例2

拡張カーソル名を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CURVAR[4];
     CHAR CMDVAR[4];
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
               :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                    (1)
strcpy(CURVAR,"CU1");                                                (2)
strcpy(CMDVAR,"CMD");                                                (3)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2 FROM S1.TBL WHERE COL4 = 38"); (4)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;                              (5)
EXEC SQL ALLOCATE :CURVAR CURSOR FOR :CMDVAR;                        (6)
EXEC SQL DESCRIBE OUTPUT :CMDVAR USING SQL DESCRIPTOR 'DESC1';       (7)
               :
  (SQL記述子域に必要な情報を設定)
               :
EXEC SQL OPEN :CURVAR;                                               (8)
EXEC SQL FETCH :CURVAR INTO SQL DESCRIPTOR 'DESC1';                  (9)
EXEC SQL DELETE FROM S1.TBL WHERE CURRENT OF :CURVAR;                (10)
EXEC SQL CLOSE :CURVAR;                                              (11)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CURVARにカーソルの名前CU1を設定します。

(3) ホスト変数CMDVARに被準備文の名前CMDを設定します。

(4) ホスト変数CMDAREAに動的SELECT文を設定します。

(5) ホスト変数CMDAREAが示すSQL文を拡張SQL文識別子CMDVARの値CMDに対応づけます。

(6) 拡張カーソル名CURVARの値CU1と拡張SQL文識別子CMDVARの値CMDを対応づけます。

(7) 拡張SQL文識別子CMDVARの値CMDに対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(8) 動的OPEN文によりカーソルをオープンします。

(9) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(10) 動的DELETE文:位置づけにより位置づけ行を削除します。

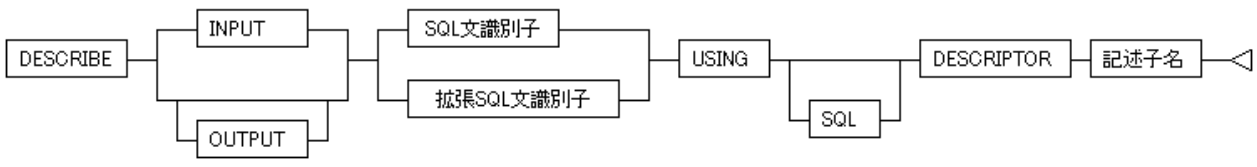
(11) 動的CLOSE文によりカーソルをクローズします。

4.10 DESCRIBE文

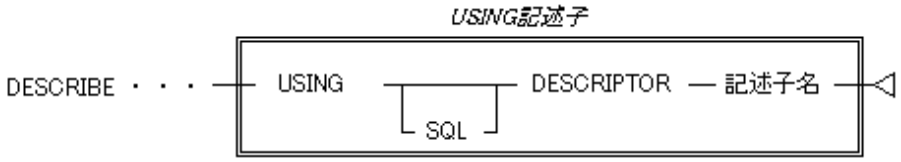
機能

被準備文中の動的パラメタ指定または選択リストについての情報を取り出します。

記述形式



構文の構成



一般規則

INPUT(入力DESCRIBE文)

- キーワードINPUTを指定したDESCRIBE文を、“入力DESCRIBE文”と呼びます。
- 入力DESCRIBE文は、記述子域に被準備文である動的SELECT文、動的単一行SELECT文、DELETE文:探索、UPDATE文:探索、INSERT文、UPDATE文:位置づけの動的パラメタ指定の情報が取り込まれます。結果は、記述子域に次のように格納されます。格納する内容については、“表4.1 取得(設定)識別子に設定される値”を参照してください。
  - 被準備文中にある動的パラメタ指定の個数が、SQLDA構造体のメンバSQLNの数を超えている、またはALLOCATE DESCRIPTOR文で指定した実現値の数を超えていると、項目記述子域には値が設定されません。そうでなければ、項目記述子域の先頭から、動的パラメタ指定が記述されている順序に対応して値が設定されます。

OUTPUT(出力DESCRIBE文)

- キーワードOUTPUTを指定した、またはキーワードを省略したDESCRIBE文を、“出力DESCRIBE文”と呼びます。
- 出力DESCRIBE文は、記述子域に被準備文である動的SELECT文または動的単一行SELECT文の選択リストの情報が取り込まれます。結果は、記述子域に次のように格納されます。格納する内容については、“表4.1 取得(設定)識別子に設定される値”を参照してください。
  - 選択リストの個数が、SQLDA構造体のメンバSQLNの数を超えている、またはALLOCATE DESCRIPTOR文で指定した実現値の数を超えていると、項目記述子域には値が設定されません。そうでなければ、項目記述子域の先頭から、選択リストの順序に対応して値が設定されます。

表4.1 取得(設定)識別子に設定される値

設定識別子		出力DESCRIBE	入力DESCRIBE	OPEN,EXECUTE	FETCH,EXECUTE
SQLDA構造体	SQL記述子域				
SQLD	COUNT	選択リストの個数	被準備中にある動的パラメタ指定の個数	動的パラメタ指定の個数を設定する。(注1)	相手指定の個数を設定する。(注1)
SQLTYPE	TYPE	データ型のコード	動的パラメタ指定のデータ型に対応するコード(注2)	動的パラメタ指定のデータ型に対応するコードを設定する。(注1)	相手指定のデータ型に対応するコードを設定する。(注1)
―	LENGTH	・文字列の最大文字長(文字列型または各国語文字列型の場合) ・データ長(真数型、概数型、日時型、時間隔型、BLOB型、またはROW_IDの場合)(注3)		・文字列の最大文字長を設定する。(文字列型または各国語文字列	(注1)

設定識別子		出力 DESCRIBE	入力 DESCRIBE	OPEN,EXECUTE	FETCH,EXECUTE
SQLDA構造体	SQL記述子域				
				型の場合)(注1) ・無視(真数型または概数型の場合)(注1) ・データ長を設定する。(BLOB型、ROW_IDの場合)(注1)(注4)	
SQLLEN	OCTET_LENGTH	・文字列の最大バイト数(文字列型または各国語文字列型の場合) ・データ長(真数型、概数型、日時型、時間隔型、BLOB型、またはROW_IDの場合)(注3)		・文字列の最大バイト数を設定する。(文字列型または各国語文字列型の場合)(注1) ・無視(真数型、概数型またはBLOB型の場合)(注1)(注4)	(注1)
SQLPRECISION.PRECISION	PRECISION	・精度(真数型または概数型の場合) ・0(真数型でも概数型でもない場合)		・精度を設定する。(真数型または概数型の場合)(注1) ・無視(真数型でも概数型でもない場合)(注1)	(注1)
SQLSCALE(注2)	SCALE	・位取り(真数型または概数型の場合) ・0(真数型でも概数型でもない場合)		・位取りを設定する。(真数型または概数型の場合) ・無視(真数型でも概	(注1)

設定識別子		出力 DESCRIBE	入力 DESCRIBE	OPEN,EXECUTE	FETCH,EXECUTE
SQLDA構造体	SQL記述子域				
				数型でもない場合) (注1)	
SQLNULLABLE	NULLABLE	・1(NULL値を許す場合) ・0(NULL値を許さない場合)			
－	INDICATOR	なし		・標識変数に対する値を設定する。動的パラメタ指定の値がNULL値なら負を設定する。(NULLABLEの値が1の場合) (注5) ・無視(NULLABLEの値が1でない場合) (注5)	相手指定に対する標識変数が設定される。
－	DATA	なし		動的パラメタ指定に対する値を設定する。 (注5)	相手指定に対する値が設定される。
SQLNAME	NAME	選択リストの値式の列指定の列名が設定される。列指定でない場合は値は設定されない。値式にAS句が指定された場合は、AS句の列名が設定される。	動的パラメタ指定に対応する列指定の列名、または動的パラメタ指定に対応する値式の列指定の列名が設定される。動的パラメタ指定に対応する値式が列指定でない場合は、または動的パラメタ指定が演算式中に含まれる場合は、値は設定されない。	無視	
－	CHARACTER_S	選択リストの値式のデータ型が文字列	動的パラメタ指定のデータ型が文字列型ま	動的パラメタ指定のデータ型が文字列型ま	相手指定のデータ型が文字列型ま

設定識別子		出力 DESCRIBE	入力 DESCRIBE	OPEN,EXECUTE	FETCH,EXECUTE
SQLDA構造体	SQL記述子域				
	ET_NAME	型または各国語文字列型の場合、データ型に対応する文字セット名が設定される。そうでない場合は、空白が設定される。	たは各国語文字列型の場合、データ型に対応する文字セット名が設定される。そうでない場合は、空白が設定される。	たは各国語文字列型の場合、データ型に対応する文字セット名を設定する。(注1)	たは各国語文字列型の場合、データ型に対応する文字セット名を設定する。(注1)
SQLSCALE (注2)	DATETIME_INTERVAL_CODE	日時型、時間隔型以外 ・ 0 日時型 ・ 1(DATE型の場合) ・ 2(TIME型の場合) ・ 3(TIMESTAMP型の場合) 時間隔型 ・ 1(時間隔修飾子がYEARの場合) ・ 2(時間隔修飾子がMONTHの場合) ・ 3(時間隔修飾子がDAYの場合) ・ 4(時間隔修飾子がHOURの場合) ・ 5(時間隔修飾子がMINUTEの場合) ・ 6(時間隔修飾子がSECONDの場合) ・ 7(時間隔修飾子がYEAR TO MONTHの場合) ・ 8(時間隔修飾子がDAY TO HOURの場合) ・ 9(時間隔修飾子がDAY TO MINUTEの場合) ・ 10(時間隔修飾子がDAY TO SECONDの場合) ・ 11(時間隔修飾子がHOUR TO MINUTEの場合) ・ 12(時間隔修飾子がHOUR TO SECONDの場合)		日時型、時間隔型以外 (注1) ・ 0を設定する。 日時型 (注1) ・ 1を設定する。(DATE型の場合) ・ 2を設定する。(TIME型の場合) ・ 3を設定する。(TIMESTAMP型の場合) 時間隔型(注1) ・ 1を設定する。(時間隔修飾子がYEARの場合) ・ 2を設定する。(時間隔修飾子がMONTHの場合) ・ 3を設定する。(時間隔修飾子がDAYの場合) ・ 4を設定する。(時間隔修飾子がHOURの場合) ・ 5を設定する。(時間隔修飾子がMINUTEの場合) ・ 6を設定する。(時間隔修飾子がSECONDの場合) ・ 7を設定する。(時間隔修飾子がYEAR TO MONTHの場合) ・ 8を設定する。(時間隔修飾子がDAY TO HOURの場合) ・ 9を設定する。(時間隔修飾子がDAY TO MINUTEの場合) ・ 10を設定する。(時間隔修飾子がDAY TO SECONDの場合)	

設定識別子		出力 DESCRIBE	入力 DESCRIBE	OPEN,EXECUTE	FETCH,EXECUTE
SQLDA構造体	SQL記述子域				
		・13(時間隔修飾子がMINUTE TO SECONDの場合)		・11を設定する。(時間隔修飾子がHOUR TO MINUTEの場合) ・12を設定する。(時間隔修飾子がHOUR TO SECONDの場合) ・13を設定する。(時間隔修飾子がMINUTE TO SECONDの場合)	
SQLPRECISION.INTERVAL	DATETIME_INTERVAL_PRECISION	精度(時間隔型の場合) ・0(時間隔型でない場合)		精度を設定する。(時間隔型の場合) (注1) ・無視(時間隔型でない場合) (注1)	

注1) 入力DESCRIBE文または出力DESCRIBE文で設定される値

注2) SQLTYPE=10(INTERVAL)の場合、SQLSCALEにDATETIME_INTERVAL_CODEの値が格納されます。

注3) 日時型、時間隔型は文字列で表現したデータ長

注4) 日時型・時間隔型は、文字列として設定します。

注5) 利用者が設定する値

備考1.(注1)、(注5)の項目は、SQL文を実行する前に設定しておく必要のある項目です。

備考2.(注1)、(注5)のない項目は、SQL文の実行時に設定される項目です。

備考3.動的パラメタ指定のデータ型は、以下のように決定されます。

ー 動的パラメタ指定が演算式中に含まれる場合、動的パラメタ指定のデータ型は演算の相手の値式のデータ型となります。

ー 動的パラメタ指定が演算式中に含まれない場合、動的パラメタ指定のデータ型は比較や代入の相手の値式のデータ型となります。たとえば、C2=?の場合、動的パラメタ指定のデータ型はC2のデータ型となります。

SQL文識別子または拡張SQL文識別子

ー 情報を取り出す被準備文の名前を指定します。

ー 情報を取り出す被準備文を準備したPREPARE文がSQL文識別子を指定している場合は、SQL文識別子を指定します。

ー 情報を取り出す被準備文を準備したPREPARE文が拡張SQL文識別子を指定している場合は、拡張SQL文識別子を指定し、拡張SQL文識別子の値に被準備文の名前を指定します。

ー 拡張SQL文識別子は、文字列型の埋込み変数で指定します。拡張SQL文識別子の値に空白を含む場合は、前後の空白を取り除いた値が被準備文の名前になります。

ー SQL文識別子を指定した場合、SQL文識別子は、同一コンパイル単位に含まれるPREPARE文で定義されていることが必要です。

ー SQL文識別子と拡張SQL文識別子の値が同じでも、異なる被準備文として区別されます。

USING記述子

- 同一の記述子名のUSING記述子が動的OPEN文に指定されている場合は、動的OPEN文に指定しているカーソルは、閉じられた状態であることが必要です。
- 同一の記述子名のUSING記述子が動的OPEN文、動的FETCH文、EXECUTE文に指定され、これらの文の実行が終了している場合にDESCRIBE文を実行すると、先のSQL記述子域の内容は失われます。

記述子名

- 記述子名には、ALLOCATE DESCRIPTOR文で割当てを行ったSQL記述子域の名前またはSQLDA構造体の変数名を指定します。
- SQL記述子域の値は、DESCRIPTOR取得文とDESCRIPTOR設定文で、それぞれ取得と設定を行います。
- SQL記述子域の名前は、文字列定数または文字列型の埋込み変数で指定します。
- SQLDA構造体は、動的パラメタ指定または相手指定に関する情報を示します。

  SQLDA構造体は、項目記述子域(SQL記述子域の各要素)SQLVAR、SQLVARの最大要素数SQLNおよび有効要素数SQLDで構成されています。SQLNおよびSQLDのデータ型は2進の精度を持つ真数です。

  SQLDA構造体の形式を以下に示します。

```
#define SQL_DESC_ENT_NUM 1

typedef struct sqlextinf {
    int     SQLPOSITION;          /* CALL文のパラメタ位置 */
    char    SQLSYSTEMRSV[252];    /* 予約域 */
} sqlextinf;

typedef struct sqlvar {
    char    *SQLDATA;             /* 変数データ域 */
    short   *SQLIND;              /* 標識変数データ域 */
    int      SQLLEN;              /* データ長 */
    short    SQLTYPE;             /* データ型のコード */
    short    SQLSCALE;            /* 位取り */
    union{
          short  PRECISION;
          struct {
            char    PRECISION;
            char    DATETIME_INTERVAL_PRECISION;
          } INTERVAL;
    } SQLPRECISION;               /* 精度 */
    short    SQLNULLABLE;         /* NULL可能性 */
    struct {
        short  SQLNAMEL;          /* 列名長 */
        char   SQLNAMEC[128];     /* 列名 */
    } SQLNAME;
    sqlextinf *SQLEXTINF;         /* 拡張情報 */
} sqlvar;

typedef struct SQLDA {
    short   SQLN;                 /* SQLVARの実配列数 */
    short   SQLD;                 /* SQLVARのデータの設定数 */
    sqlvar  SQLVAR[SQL_DESC_ENT_NUM];
} SQLDA;
```

- SQL記述子域は動的パラメタ指定または相手指定に関する情報を示します。SQL記述子域は識別子COUNTと0以上の項目記述子域(SQL記述子域の各要素)で構成されています。COUNTはSQL記述子域の動的パラメタ指定または相手指定の数を示し、そのデータ型は2進の精度を持つ真数です。
- 項目記述子域は以下の内容で構成されています。

表4.2 項目記述子域の内容

項目記述子域		意味	データ型
SQLDA構造体のメンバ名	SQL記述子域の識別子		
SQLDATA	－	ホスト変数のアドレス	－
SQLIND	－	標識変数のアドレス	－
－	DATA	データ値	TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, CHARACTER_SET_NAME, DATETIME_INTERVAL_CODEで指定されるデータ型に一致
－	INDICATOR	標識変数の値	2進の精度を持つ真数
SQLLEN	OCTET_LENGTH	長さ(バイト数)	2進の精度を持つ真数
－	LENGTH	長さ(文字数)	2進の精度を持つ真数
SQLTYPE	TYPE	データ型のコード	2進の精度を持つ真数
SQLSCALE	SCALE	位取りまたは日時型・時間隔型のコード	2進の精度を持つ真数
SQLPRECISION.PRECISION	PRECISION	精度	2進の精度を持つ真数
SQLPRECISION.INTERVAL	DATETIME_INTERVAL_PRECISION	時間隔先行フィールド精度	2進の精度を持つ真数
－	DATETIME_INTERVAL_CODE	日時型・時間隔型のコード	2進の精度を持つ真数
SQLNULLABLE	NULLABLE	NULL値を許すかどうか	2進の精度を持つ真数
SQLNAME	NAME	列指定の列名	SQLDA構造体の場合: 長さ108バイト以下の文字列型 SQL記述子域の場合: 長さ38バイト以上108バイト以下の文字列型
－	CHARACTER_SET_NAME	文字セット名	長さ38バイト以上108バイト以下の文字列型
SQLEXTINF	－	拡張情報	－

－ SQLDA構造体のSQLTYPEの値（データ型のコード）またはSQL記述子域のTYPEの値(データ型のコード)とSQLのデータ型の対応は以下の通りです。

表4.3 SQLTYPEまたはTYPEの値とSQLのデータ型の対応

コード	データ型
1	CHARACTERまたはNATIONAL CHARACTER (注1)
2	NUMERIC
3	DECIMAL
4	INTEGER
5	SMALLINT
7	REAL
8	DOUBLE PRECISION
9	DATE、TIMEまたはTIMESTAMP
10	INTERVAL
11	NATIONAL CHARACTER (注1)
12	CHARACTER VARYING またはNATIONAL CHARACTER VARYING (注1)
13	NATIONAL CHARACTER VARYING (注1)
30	BLOB
31	(注2)
50	ROW_ID

注1) データ型がNATIONAL CHARACTER、NATIONAL CHARACTER VARYINGの場合、SQLDA構造体の場合は、コードが11と13になります。 SQL記述子域の場合は、コードが1と12になります。CHARACTERとNATIONAL CHARACTERの区別は、CHARACTER_SET_NAMEの値で識別します。

注2) SQL埋込みCプログラムにおいて、8バイトの整数型(long long型など)と対応します。DESCRIBE文では、コードに31は返却されません。

－ SQLDA構造体のSASQLTYPEの値が9を示す場合は、メンバSQLSCALEには以下のDATE_TIME_INTERVAL_CODEを格納します。また、SQL記述子域のTYPEの値が9を示す場合は、以下のDATETIME_INTERVAL_CODEを格納します。

表4.4 SQLTYPEの値が9を示す場合のDATETIME_INTERVAL_CODEの値

コード	データ型
1	DATE
2	TIME
3	TIMESTAMP

－ SQLDA構造体のSQLTYPEの値が10を示す場合は、メンバSQLSCALEには以下のDATE_TIME_INTERVAL_CODEを格納します。また、SQL記述子域のTYPEの値が10を示す場合は、以下のDATETIME_INTERVAL_CODEを格納します。

表4.5 SQLTYPEの値が10を示す場合のDATETIME_INTERVAL_CODEの値

コード	時間隔修飾子
1	YEAR
2	MONTH

コード	時間隔修飾子
3	DAY
4	HOUR
5	MINUTE
6	SECOND
7	YEAR TO MONTH
8	DAY TO HOUR
9	DAY TO MINUTE
10	DAY TO SECOND
11	HOUR TO MINUTE
12	HOUR TO SECOND
13	MINUTE TO SECOND

- TYPEの値が1または12のとき、CHARACTER_SET_NAMEの値(文字セット名)は文字列型と各国語文字列型で以下のように異なります。

  文字列型:

  BASIC

  各国語文字列型:

  NCHAR

- SQLDA構造体の場合、DESCRIBE文を実行すると、選択リストまたは動的パラメタ指定のデータ型に対応する精度、位取りおよび長さが、それぞれのSQLDA構造体のメンバSQLPRECISION、SQLSCALEおよびSQLLENに設定されます。

  また、SQL記述子域の場合は、DESCRIBE文を実行すると、動的パラメタ指定または選択リストのデータ型に対応する精度、位取り、長さ(文字数)および長さ(バイト数)が、それぞれのSQL記述子域の識別子のPRECISION、SCALE、LENGTH、OCTET_LENGTH、DATETIME_INTERVAL_PRECISIONに設定されます。

  SQLDA構造体の場合、データ型に対応するそれぞれの値は、以下のとおりです。

表4.6 SQLのデータ型とSQLDA構造体の値の対応

SQLのデータ型(注1)	SQLTYPE	SQLPRECISION	SQLSCALE	SQLLEN	DATETIME_INTERVAL_PRECISION
CHARACTER	1	0	0	1～32000	0
NUMERIC	2	1～18	0～18	1～19	0
DECIMAL	3	1～18	0～18	1～10	0
INTEGER	4	31	0	4	0
SMALLINT	5	15	0	2	0
REAL	7	23	0	4	0
DOUBLE PRECISION	8	52	0	8	0
DATE	9	0	1 (注2)	10	0
TIME	9	0	2 (注2)	8	0
TIMESTAMP	9	0	3 (注2)	19	0
INTERVAL	10	(注3)	(注4)	2～19	1～19

SQLのデータ型(注1)	SQLTYPE	SQLPRECISION	SQLSCALE	SQLLEN	DATETIME_INTERVAL_PRECISION
NATIONAL CHARACTER	11	0	0	2～32000	0
CHARACTER VARYING	12	0	0	3～32002 (注5)	0
NATIONAL CHARACTER VARYING	13	0	0	4～32002 (注5)	0
BLOB	30	0	0	n*1024 (注6)	0
(注7)	31	63	0	8	0
ROW_ID	50	0	0	24	0

注1) SQLデータ型に対応するCの変数定義については“6.3 SQL埋込みCプログラム”を参照してください。SQLデータ型に対応するCOBOLの変数定義については“6.4 SQL埋込みCOBOLプログラム”を参照してください。

注2) DATE、TIMEまたはTIMESTAMPでは、SQLSCALE=0ですがDATETIME_INTERVAL_CODE値を代替として利用します。

注3) INTERVALでのSQLPRECISIONはPRECISION値(0)とDATETIME_INTERVAL_PRECISIONとなります。

注4) INTERVALでは、SQLSCALE=0ですがDATETIME_INTERVAL_CODE値を代替として利用します。

注5) 先頭に2バイトの長さが含まれます。

注6) n=1～2097150を示します。

注7) SQL埋込みCプログラムにおいて、8バイトの整数型(long long型など)と対応します。DESCRIBE文では、コードに31は返却されません。

SQL記述子域の場合、データ型に対応するそれぞれの値は、以下のとおりです。

表4.7 SQLのデータ型とSQL記述子域の値の対応

SQLのデータ型(注1)	TYPE	PRECISION	SCALE	OCTET_LENGTH	LENGTH	DATETIME_INTERVAL_PRECISION
CHARACTER	1	0	0	1～32000	1～32000	0
NATIONAL CHARACTER	1	0	0	2～32000	2～32000	0
NUMERIC	2	1～18	0～18	1～19	1～19	0
DECIMAL	3	1～18	0～18	1～10	1～10	0
INTEGER	4	31	0	4	4	0
SMALLINT	5	15	0	2	2	0
REAL	7	23	0	4	4	0
DOUBLE PRECISION	8	52	0	8	8	0
DATE	9	0	0	10	10	0
TIME	9	0	0	8	8	0
TIMESTAMP	9	0	0	19	19	0
INTERVAL	10	0	0	2～19	2～19	1～19
CHARACTER VARYING	12	0	0	1～32000	1～32000	0

SQLのデータ型(注1)	TYPE	PRECISION	SCALE	OCTET_LENGTH	LENGTH	DATETIME_INTERVAL_PRECISION
NATIONAL CHARACTER VARYING	12	0	0	2～32000	1～16000	0
BLOB	30	0	0	n*1024 (注2)	n*1024 (注2)	0
(注3)	31	63	0	8	8	0
ROW_ID	50	0	0	24	24	0

注1) SQLデータ型に対応するCの変数定義については“6.3 SQL埋込みCプログラム”を参照してください。SQLデータ型に対応するCOBOLの変数定義については“6.4 SQL埋込みCOBOLプログラム”を参照してください。

注2) n=1～2097150を示します。

注3) SQL埋込みCプログラムにおいて、8バイトの整数型(long long型など)と対応します。DESCRIBE文では、コードに31は返却されません。

使用例

例1

SQLDA構造体を使用して、SQL文識別子CMDに対応する被準備文の動的パラメタ指定および選択リストの情報を取得します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
#include "qdbnosis.h"

EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR  CMDAREA[100];
    SQLDA    *sqlda_in;
    SQLDA    *sqlda_out;
EXEC SQL END DECLARE SECTION;
                :
size = sizeof(SQLDA) + sizeof(sqlvar) * (50-1);

sqlda_in = (SQLDA *)malloc(size);                                    (1)
memset(sqlda_in, 0x00, size);
sqlda_in->SQLN = 50;

sqlda_out = (SQLDA *)malloc(size);                                   (2)
memset(sqlda_out, 0x00, size);
sqlda_out->SQLN = 50;

EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                     (3)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4=?");  (4)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);

EXEC SQL PREPARE CMD FROM :CMDAREA;                                  (5)

EXEC SQL DESCRIBE INPUT CMD USING SQL DESCRIPTOR :sqlda_in;          (6)
EXEC SQL DESCRIBE OUTPUT CMD USING SQL DESCRIPTOR :sqlda_out;        (7)
                :
  (SQLDA構造体に必要な情報を設定)
                :
EXEC SQL OPEN CU1 USING SQL DESCRIPTOR :sqlda_in;
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR :sqlda_out;                   (8)
                :
EXEC SQL CLOSE CU1;
```

(1) SQLDA構造体sqlda_inの領域を割り当てます。

(2) SQLDA構造体sqlda_outの領域を割り当てます。

(3) 動的カーソル宣言によりCU1のカーソルを宣言します。

(4) ホスト変数CMDAREAに動的SELECT文を設定します。

(5) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(6) SQL文識別子CMDに対応する被準備文の動的パラメタ指定の情報をSQLDA構造体sqlda_inに取り込みます。

(7) SQL文識別子CMDに対応する被準備文の選択リストの情報をSQLDA構造体sqlda_outに取り込みます。

(8) 動的OPEN文、動的FETCH文、動的CLOSE文によりカーソルの操作を行います。

例2

SQL記述子域を使用して、SQL文識別子CMDに対応する被準備文の動的パラメタ指定および選択リストの情報を取得します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR  CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
EXEC SQL ALLOCATE DESCRIPTOR 'DESC2' WITH MAX 50;                          (2)
EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                       (3)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = ?"); (4)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                        (5)
EXEC SQL DESCRIBE INPUT CMD USING SQL DESCRIPTOR 'DESC1';                  (6)
EXEC SQL DESCRIBE OUTPUT CMD USING SQL DESCRIPTOR 'DESC2';                 (7)
                :
  (SQL記述子域に必要な情報を設定)
                :
EXEC SQL OPEN CU1 USING SQL DESCRIPTOR 'DESC1';                            (8)
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR 'DESC2';                            (8)
                :
EXEC SQL CLOSE CU1;
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) 記述子名DESC2のSQL記述子域を割り当てます。

(3) 動的カーソル宣言によりCU1のカーソルを宣言します。

(4) ホスト変数CMDAREAに動的SELECT文を設定します。

(5) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(6) SQL文識別子CMDに対応する被準備文の動的パラメタ指定の情報を記述子名DESC1のSQL記述子域に取り込みます。

(7) SQL文識別子CMDに対応する被準備文の選択リストの情報を記述子名DESC2のSQL記述子域に取り込みます。

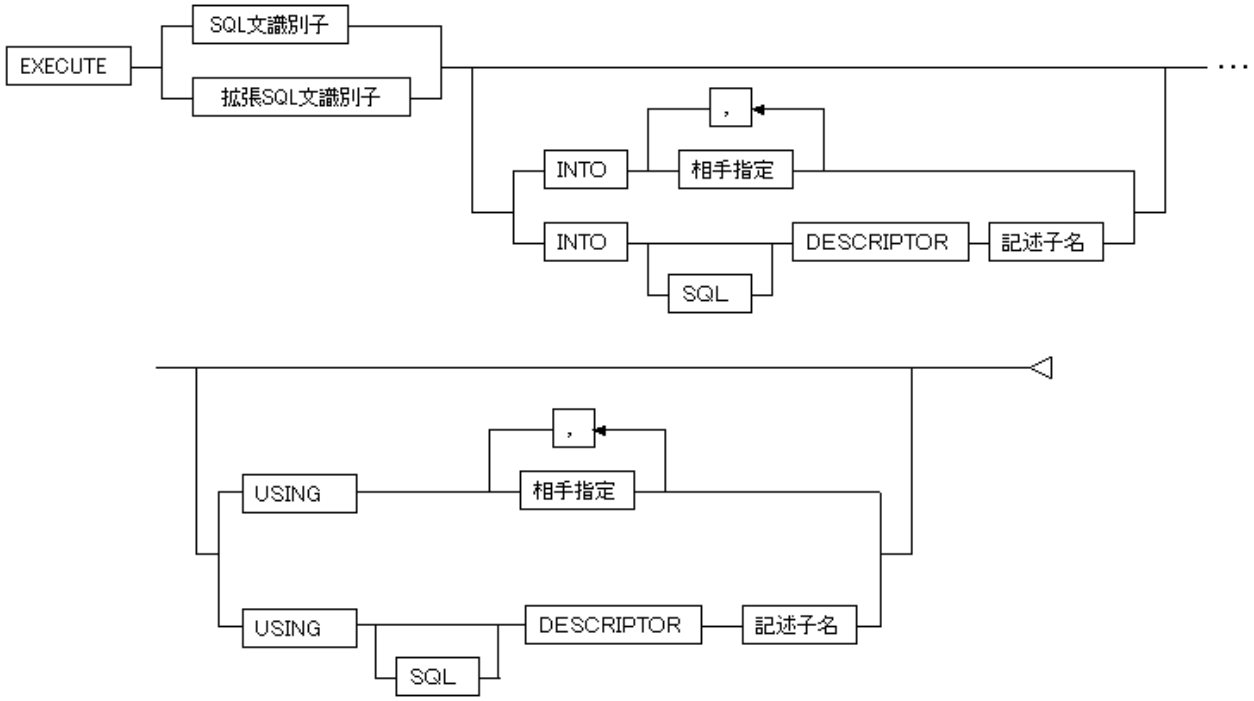
(8) 動的OPEN文、動的FETCH文、動的CLOSE文によりカーソルの操作を行います。

4.11 EXECUTE文

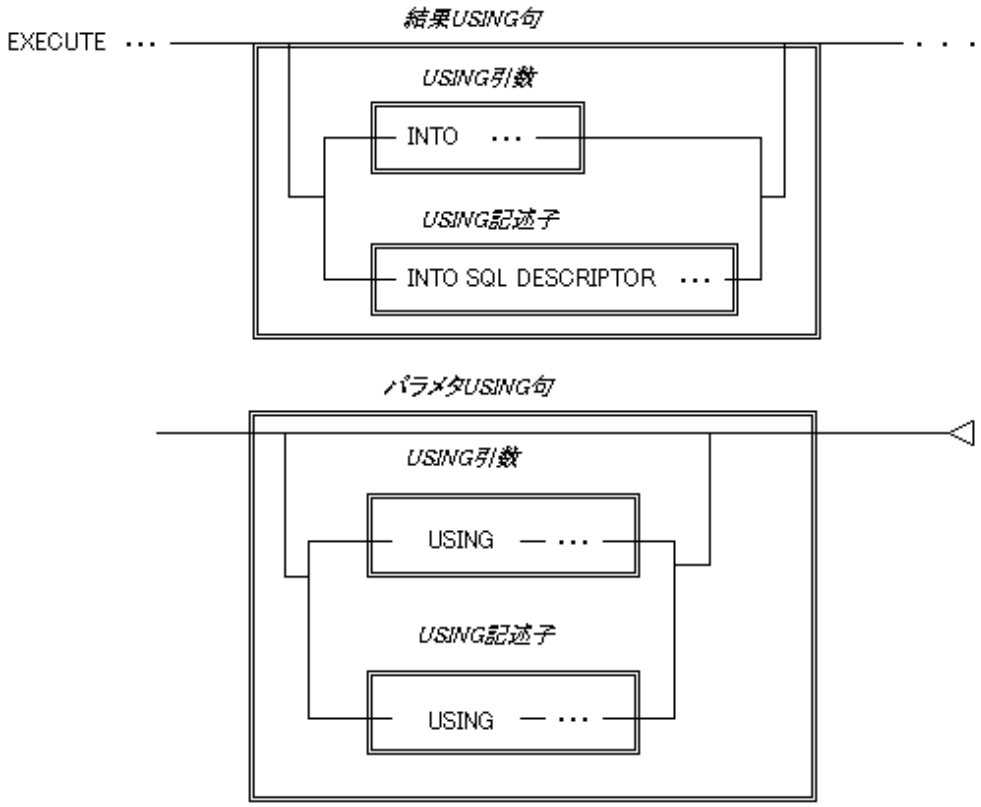
機能

入力変数と出力変数を被準備文に関連づけてからその文を実行します。

記述形式



構文の構成



参照項番

- 相手指定 → “2.3 値指定と相手指定”

権限

・ EXECUTE文を実行できるのは、被準備文に対する権限の保持者です。

一般規則

・ EXECUTE文により、パラメタUSING句に指定された動的パラメタ指定の値を関連づけて、被準備文が実行されます。ただし、被準備文が動的単一行SELECT文の場合は、結果USING句に実行結果の値が設定されます。

・ PREPARE文により準備された被準備文がプロシジャルーチンの呼出し(CALL文)である場合、USING句は以下のように指定します。

ー キーワードINTOには、プロシジャルーチンのパラメタモードがOUTまたはINOUTのパラメタ宣言に対応する引数を指定します。

ー キーワードUSINGには、プロシジャルーチンのパラメタモードがINまたはINOUTのパラメタ宣言に対応する引数を指定します。

SQL文識別子または拡張SQL文識別子

ー 実行する被準備文の名前を指定します。

ー 実行する被準備文を準備したPREPARE文がSQL文識別子を指定している場合は、SQL文識別子を指定します。

ー 実行する被準備文を準備したPREPARE文が拡張SQL文識別子を指定している場合は、拡張SQL文識別子を指定し、拡張SQL文識別子の値に被準備文の名前を指定します。

ー 拡張SQL文識別子は、文字列型の埋込み変数で指定します。拡張SQL文識別子の値に空白を含む場合は、前後の空白を取り除いた値が被準備文の名前になります。

ー SQL文識別子を指定した場合、SQL文識別子は、同一コンパイル単位に含まれるPREPARE文で定義されていることが必要です。

ー SQL文識別子と拡張SQL文識別子の値が同じでも、異なる被準備文として区別されます。

ー PREPARE文により準備された被準備文は以下のものであることが必要です。

- 動的単一行SELECT文
- UPDATE文:探索
- INSERT文
- DELETE文:探索
- 準備可能動的UPDATE文:位置づけ
- 準備可能動的DELETE文:位置づけ
- スキーマ定義文
- スキーマ操作文
- 格納構造定義文
- 格納構造操作文
- 利用者制御文
- アクセス制御文
- システム制御文
- CALL文

パラメタUSING句

ー PREPARE文により準備された被準備文に動的パラメタ指定がある場合、パラメタUSING句を指定します。

- パラメタUSING句にUSING引数を指定した場合、引数には被準備文の動的パラメタ指定の値を設定します。設定する内容については、“表4.1 取得(設定)識別子に設定される値”を参照してください。
  - 引数の順序は動的パラメタの順序に対応していることが必要です。
  - 引数の個数は動的パラメタ指定の個数と同じであることが必要です。
  - 引数のデータ型は動的パラメタ指定のデータ型と比較可能であることが必要です。
- パラメタUSING句にUSING記述子を指定した場合、記述子域には被準備文の動的パラメタ指定の値の情報を指定します。
  - 項目記述子域の順序は動的パラメタ指定の順序に対応していることが必要です。
  - SQLDA構造体のメンバSQLTYPE、SQLLEN、SQLPRECISION.PRECISION、SQLSCALE、SQLNAME、SQLPRECISION.INTERVALまでの情報は、入力DESCRIBE文を実行することにより設定されているので、記述子域の内容を変更しない限り設定する必要はありません。ただし、日時型、時間隔型は文字列型に変更する必要があります。
  - SQL記述子域の識別子TYPE、LENGTH、OCTET_LENGTH、PRECISION、SCALE、CHARACTER_SET_NAME、NAME、DATETIME_INTERVAL_CODE、DATETIME_INTERVAL_PRECISIONまでの情報は、入力DESCRIBE文を実行することにより設定されているので、記述子域の内容を変更しない限り設定する必要はありません。ただし、日時型、時間隔型は文字列型に変更する必要があります。

結果USING句

- PREPARE文により準備された被準備文が動的単一行SELECT文である場合、結果USING句を指定します。
- 結果USING句にUSING引数を指定した場合、引数には相手指定を設定します。設定する内容については、“表4.1 取得(設定)識別子に設定される値”を参照してください。
  - 引数の順序は相手指定の順序に対応していることが必要です。
  - 引数の個数は相手指定の個数と同じであることが必要です。
  - 引数のデータ型は相手指定のデータ型と代入可能であることが必要です。
- 結果USING句にUSING記述子を指定した場合、記述子域には相手指定の情報を指定します。
  - 項目記述子域の順序は相手指定の順序に対応していることが必要です。
  - SQLDA構造体のメンバSQLTYPE、SQLLEN、SQLPRECISION.PRECISION、SQLSCALE、SQLNAME、SQLPRECISION.INTERVALまでの情報は、出力DESCRIBE文を実行することにより設定されているので、記述子域の内容を変更しない限り設定する必要はありません。ただし、日時型、時間隔型は文字列型に変更する必要があります。
  - SQL記述子域のTYPE、LENGTH、OCTET_LENGTH、PRECISION、SCALE、CHARACTER_SET_NAME、NAME、DATETIME_INTERVAL_CODE、DATETIME_INTERVAL_PRECISIONまでの情報は、出力DESCRIBE文を実行することにより設定されているので、記述子域の内容を変更しない限り設定する必要はありません。ただし、日時型、時間隔型は文字列型に変更する必要があります。

記述子名

- 記述子名には、SQL記述子域の名前またはSQLDA構造体変数の変数名を指定します。
- 記述子名は、文字列定数または文字列型の埋込み変数で指定します。

使用例

例1

DELETE文:探索を動的に実行します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
    short INVAR1;
EXEC SQL END DECLARE SECTION;
```

```
                    :
strcpy(CMDAREA.sqlvar,"DELETE FROM S1.TBL WHERE COL1 = ?");
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;
                    :
  (変数"INVAR1"に値を設定)
                    :
EXEC SQL EXECUTE CMD USING :INVAR1;
```

例2

単一行SELECT文を動的に実行します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                    :
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = ?");
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;
                    :
  (SQL記述子域に必要な情報を設定)
                    :
EXEC SQL EXECUTE CMD INTO SQL DESCRIPTOR 'DESC1'
                     USING SQL DESCRIPTOR 'DESC2';
```

例3

SQLDA構造体を使用して、単一行SELECT文を動的に実行します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
#include "qdbnosis.h"

EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR CMDAREA[100];
  SQLDA   *sqlda_out;
  SQLDA   *sqlda_in;
EXEC SQL END DECLARE SECTION;
             :
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4=?");
CMDAREA.sqllen = strlen( CMDAREA.sqlvar );
EXEC SQL PREPARE CMD FROM :CMDAREA;
             :
  (SQLDA構造体に必要な情報を設定)
             :
EXEC SQL EXECUTE CMD1 INTO SQL DESCRIPTOR :sqlda_out
                      USING SQL DESCRIPTOR :sqlda_in;
```

例4

拡張SQL文識別子を指定して単一行SELECT文を動的に実行します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    CHAR CMDVAR[4];
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                    :
strcpy(CMDVAR,"CMD");
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = ?");
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
```

```
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;
                 :
  (SQL記述子域に必要な情報を設定)
                 :
EXEC SQL EXECUTE :CMDVAR INTO SQL DESCRIPTOR 'DESC1'
                         USING SQL DESCRIPTOR 'DESC2';
```

4.12 EXECUTE IMMEDIATE文

機能

SQL文を動的に準備して実行します。

記述形式

参照項番

・ 埋込み変数名 → “6.5 SQL埋込みホストプログラム”

権限

・ EXECUTE IMMEDIATE文を実行できるのは、準備可能文に対する権限の保持者です。

一般規則

・ SQL文変数にSQL文を設定して、EXECUTE IMMEDIATE文を実行することにより、そのSQL文が実行されます。

SQL文変数

ー SQL文が設定されている埋込み変数名を指定します。

ー SQL文変数の内容は以下のものであることが必要です。

- UPDATE文:探索
- INSERT文
- DELETE文:探索
- 準備可能動的UPDATE文:位置づけ
- 準備可能動的DELETE文:位置づけ
- スキーマ定義文
- スキーマ操作文
- 格納構造定義文
- 格納構造操作文
- 利用者制御文
- アクセス制御文
- システム制御文
- CALL文

ー SQL文変数の内容が準備可能動的DELETE文:位置づけまたは準備可能動的UPDATE文:位置づけの場合、指定したカーソル名は、以下のいずれかと同じであることが必要です。ただし、両方と同じである場合は、例外(あいまいなカーソル名)となります。

- 同一コンパイル単位に含まれる動的カーソル宣言のカーソル名

- 同一セションで実行されたALLOCATE CURSOR文の拡張カーソル名の値

また、指定したカーソルがオープンされていることが必要です。

ー SQL文変数に設定するSQL文には、SQL先頭子(EXEC SQL)、SQL終了子(END-EXECまたは;)、注釈、ホスト変数および動的パラメタ指定を含むことはできません。

使用例

例

被準備文が、DELETE文:探索の場合を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
strcpy(CMDAREA.sqlvar,"DELETE FROM S1.TBL WHERE COL1 = 39");          (1)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
                :
EXEC SQL EXECUTE IMMEDIATE :CMDAREA;                                  (2)
```

(1) ホスト変数CMDAREAに被準備文を設定します。

(2) ホスト変数CMDAREAに対応づけられた被準備文を実行します。

これは、次のSQL文と同じです。

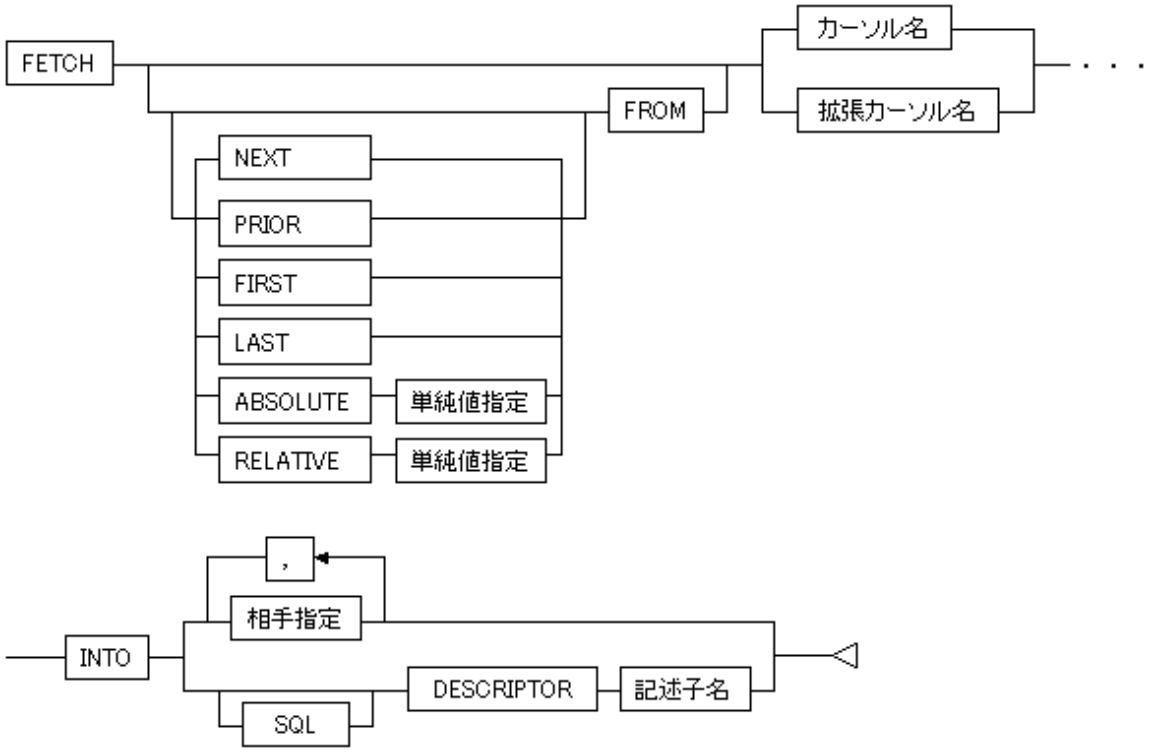
```
EXEC SQL DELETE FROM S1.TBL WHERE COL1 = 39;
```

4.13 FETCH文(動的FETCH文)

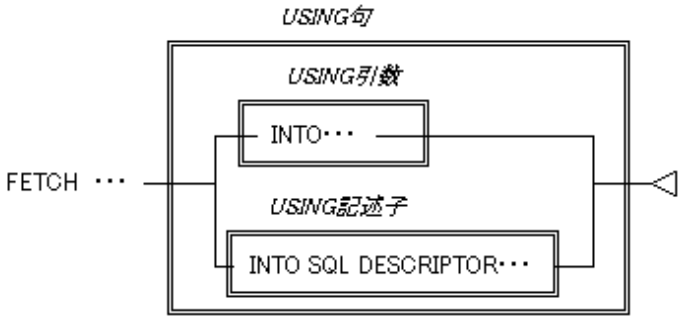
機能

動的カーソル宣言またはALLOCATE CURSOR文で指定したカーソルを次の行に位置づけ、その行から値を取り出します。

記述形式



構文の構成



参照項番

- 単純値指定 → “2.3 値指定と相手指定”
- 相手指定 → “2.3 値指定と相手指定”

権限

- FETCH文(動的FETCH文)を実行できるのは、カーソルの処理対象の表に対するSELECT権の保持者です。

一般規則

- カーソル名または拡張カーソル名、およびUSING句以外の規則は、“3.46 FETCH文”の一般規則を参照してください。

カーソル名または拡張カーソル名

ー カーソルの名前を指定します。

ー 動的カーソル宣言で定義したカーソルの値を取り出す場合は、カーソル名を指定します。

- ALLOCATE CURSOR文で定義したカーソルの値を取り出す場合は、拡張カーソル名を指定し、拡張カーソル名の値にカーソルの名前を指定します。
- 拡張カーソル名は、文字列型の埋込み変数で指定します。拡張カーソル名の値に空白を含む場合は、前後の空白を取り除いた値がカーソルの名前になります。
- カーソル名を指定した場合、カーソル名は、同一コンパイル単位に含まれる動的カーソル宣言で定義されていることが必要です。
- カーソル名と拡張カーソル名の値が同じでも、異なるカーソルとして区別されます。

USING句

- USING句にUSING引数を指定した場合、引数には相手指定を設定します。設定する内容については、"表4.1 取得(設定)識別子に設定される値"を参照してください。
  - 引数の順序は相手指定の順序に対応していることが必要です。
  - 引数の個数は相手指定の個数と同じであることが必要です。
  - 引数のデータ型は相手指定のデータ型と代入可能であることが必要です。
- USING句にUSING記述子を指定した場合、記述子域には相手指定の情報を指定します。
  - 項目記述子域の順序は相手指定の順序に対応していることが必要です。
  - SQLDA構造体のメンバSQLTYPE、SQLLEN、SQLPRECISION.PRECISION、SQLSCALE、SQLNAME、SQLPRECISION.INTERVALまでの情報は、出力DESCRIBE文を実行することにより設定されているので、記述子域の内容を変更しない限り設定する必要はありません。ただし、日時型、時間隔型は文字列型に変更する必要があります。
  - SQL記述子域のTYPE、LENGTH、OCTET_LENGTH、PRECISION、SCALE、CHARACTER_SET_NAME、NAME、DATETIME_INTERVAL_CODE、DATETIME_INTERVAL_PRECISIONまでの情報は、出力DESCRIBE文を実行することにより設定されているので、記述子域の内容を変更しない限り設定する必要はありません。ただし、日時型、時間隔型は文字列型に変更する必要があります。

使用例

例1

SQLDA構造体を使用して、SQL文識別子CMDに対応する被準備文の選択リストの情報を取得します。なお、可変長文字型の展開規則は、"6.3 SQL埋込みCプログラム"を参照してください。

```
#include "qdbnosis.h"

EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
    SQLDA *sqlda_out;
EXEC SQL END DECLARE SECTION;
            :
size = sizeof(SQLDA) + sizeof(sqlvar) * (50 - 1 );

sqlda_out = (SQLDA *)malloc(size);                                    (1)
memset( sqlda_out, 0x00, size );
sqlda_out->SQLN = 50;

EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                      (2)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4=38");  (3)
CMDAREA.sqllen = strlen( CMDAREA.sqlvar );

EXEC SQL PREPARE CMD FROM :CMDAREA;                                   (4)

EXEC SQL DESCRIBE OUTPUT CMD USING SQL DESCRIPTOR :sqlda_out;         (5)
            :
  (SQLDA構造体に必要な情報を設定)
            :
EXEC SQL OPEN CU1;                                                    (6)
```

```
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR :sqlda_out;                    (7)
            :
EXEC SQL CLOSE CU1;
```

(1) SQLDA構造体sqlda_outの領域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREAに動的SELECT文を設定します。

(4) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(5) SQL文識別子CMDに対応する被準備文の選択リストの情報をSQLDA構造体sqlda_outに取り込みます。

(6) 動的OPEN文によりカーソルをオープンします。

(7) 動的FETCH文によりSQLDA構造体sqlda_outに実行結果を取り込みます。

例2

SQL記述子域を使用して、SQL文識別子CMDに対応する被準備文の選択リストの情報を取得します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                     (1)
EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                  (2)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = 38");  (3)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                   (4)
EXEC SQL DESCRIBE OUTPUT CMD USING SQL DESCRIPTOR 'DESC1';            (5)
                :
  (SQL記述子域に必要な情報を設定)
                :
EXEC SQL OPEN CU1;                                                    (6)
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR 'DESC1';                       (7)
                :
EXEC SQL CLOSE CU1;
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREAに動的SELECT文を設定します。

(4) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(5) SQL文識別子CMDに対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(6) 動的OPEN文によりカーソルをオープンします。

(7) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

例3

拡張カーソル名を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CURVAR[4];
     CHAR CMDVAR[4];
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                     (1)
strcpy(CURVAR,"CU1");                                                 (2)
```

```
strcpy(CMDVAR,"CMD");                                                    (3)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = 38");(4)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;                                  (5)
EXEC SQL ALLOCATE :CURVAR CURSOR FOR :CMDVAR;                            (6)
EXEC SQL DESCRIBE OUTPUT :CMDVAR USING SQL DESCRIPTOR 'DESC1';           (7)
                    :
  (SQL記述子域に必要な情報を設定)
                    :
EXEC SQL OPEN :CURVAR;                                                   (8)
EXEC SQL FETCH :CURVAR INTO SQL DESCRIPTOR 'DESC1';                      (9)
                    :
EXEC SQL CLOSE :CURVAR;                                                 (10)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CURVARにカーソルの名前CU1を設定します。

(3) ホスト変数CMDVARに被準備文の名前CMDを設定します。

(4) ホスト変数CMDAREAに動的SELECT文を設定します。

(5) ホスト変数CMDAREAが示すSQL文を拡張SQL文識別子CMDVARの値CMDに対応づけます。

(6) 拡張カーソル名CURVARの値CU1と拡張SQL文識別子CMDVARの値CMDを対応づけます。

(7) 拡張SQL文識別子CMDVARの値CMDに対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(8) 動的OPEN文によりカーソルをオープンします。

(9) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

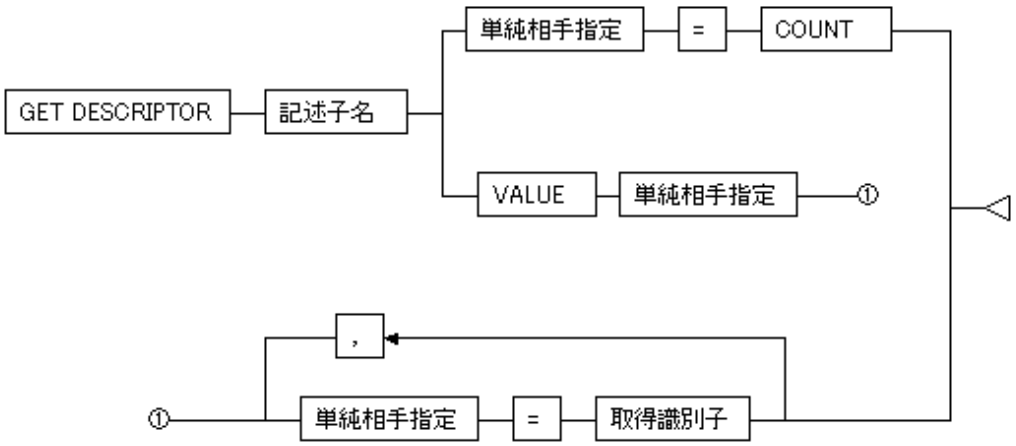
(10) 動的CLOSE文によりカーソルをクローズします。

4.14 GET DESCRIPTOR文(DESCRIPTOR取得文)

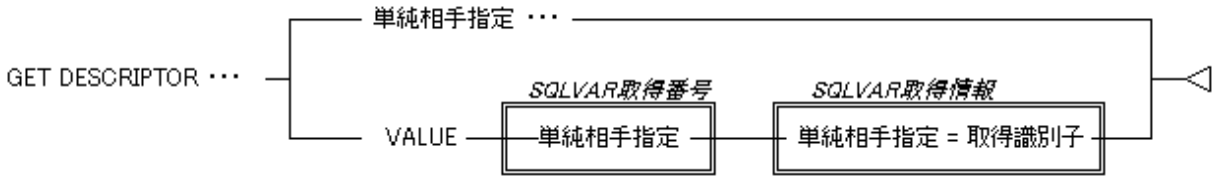
機能

SQL記述子域から情報を取得します。

記述形式



構文の構成



参照項番

・ 単純相手指定 → “2.3 値指定と相手指定”

一般規則

・ 指定された記述子名を持つSQL記述子域が割り当てられていることが必要です。

記述子名

― SQL記述子域の名前を指定します。SQLDA構造体変数の変数名は指定できません。

― 記述子名は文字列定数または文字列型の埋込み変数で指定します。

COUNT

― COUNTを指定した場合は、単純相手指定には、項目記述子域に設定された動的パラメタ指定、または相手指定の個数が設定されます。

― COUNTを指定した場合の単純相手指定のデータ型は2進の精度を持つ真数であることが必要です。

SQLVAR取得番号

― 取得する項目記述子域の番号を指定します。

― SQLVAR取得番号のデータ型は、2進の精度を持つ真数であることが必要です。

― SQLVAR取得番号は、1からCOUNTの値の範囲内であることが必要です。

SQLVAR取得情報

― SQLVAR取得情報は、DESCRIPTOR取得文で取得する情報です。

― 取得識別子には、SQLVAR取得番号で指定された、項目記述子域の内容を取得します。取得される内容および一般規則を以下に示します。

- INDICATORの値が負の場合は、INDICATORを取得せずにDATAの値を取得することはできません。
- 相手指定に対するDATAの値を取得するには、同一の記述子名のUSING記述子が指定された動的FETCH文またはEXECUTE文が実行されていることが必要です。

取得識別子	取得する内容	データ型
TYPE	データ型のコード	2進の精度を持つ真数
LENGTH	・ 文字列の最大文字長（文字列型または各国語文字列型の場合）（注1） ・ データ長（真数型、概数型、日時型、時間隔型、ROW_IDまたはBLOB型の場合）（注2）	2進の精度を持つ真数
OCTET_LENGTH	・ 文字列の最大バイト数（文字列型または各国語文字列型の場合）（注1）	2進の精度を持つ真数

取得識別子	取得する内容	データ型
	・ データ長（真数型、概数型、日時型、時間隔型、ROW_IDまたはBLOB型の場合） （注2）	
PRECISION	・ 精度(真数型または概数型の場合) ・ 0 (真数型でも概数型でもない場合)	2進の精度を持つ真数
SCALE	・ 位取り(真数型または概数型の場合) ・ 0 (真数型でも概数型でもない場合)	2進の精度を持つ真数
NULLABLE	・ 1（NULL値を許す場合） ・ 0（NULL値を許さない場合）	2進の精度を持つ真数
INDICATOR	指定された項目記述子域に対応する動的パラメタ指定または相手指定に対する標識変数の値（注3）	2進の精度を持つ真数
DATA	・ 指定された項目記述子域に対応する動的パラメタ指定または相手指定の値 ・ INDICATOR の値が負の場合、DATAの値はNULL値となる	TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, CHARACTER_SET_NAME, DATETIME_INTERVAL_CODEで指定されるデータ型に一致
NAME	動的パラメタ指定または相手指定に対応する列名	長さ38バイト以上の文字列型
CHARACTER_SET_NAME	・ BASIC（文字列型） ・ NCHAR（各国語文字列型） ・ 空白(文字列型でも各国語文字列型でもない場合)	長さ38バイト以上の文字列型
DATETIME_INTERVAL_CODE	・ TYPEの値が9の場合のデータ型のコード ・ TYPEの値が10の場合の時間隔修飾子のコード	2進の精度を持つ真数
DATETIME_INTERVAL_PRECISION	・ 0 (時間隔型でない場合) ・ 時間隔先行フィールド精度 (時間隔型の場合)	2進の精度を持つ真数

注1) 取得した値が32,000を超える場合は、LENGTHまたはOCTET_LENGTHには、32,000が設定されます。

注2) 日時型、時間隔型は文字列で表現したデータ長が設定されます。

注3) 取得した値が32,000を超える場合は、0または32,000が設定されます。この場合、コード変換が発生しない環境で動作させてください。

データ型に対応する各々の値は、“表4.7 SQLのデータ型とSQL記述子域の値の対応”を参照してください。

使用例

例

SQL文識別子CMDに対応する被準備文の動的パラメタ指定の情報を取得します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
strcpy(CMDAREA.sqlvar,"DELETE FROM S1.TBL WHERE COL1 = ? AND COL2 = ?");   (2)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                        (3)
EXEC SQL DESCRIBE INPUT CMD USING SQL DESCRIPTOR 'DESC1';                  (4)
EXEC SQL GET DESCRIPTOR 'DESC1':VARCOUNT = COUNT;                          (5)
for(VARWCOUNT = 1 ; VARWCOUNT <= VARCOUNT ; ++VARWCOUNT) {
    EXEC SQL GET DESCRIPTOR 'DESC1' VALUE :VARWCOUNT                       (6)
      :VARTYPE = TYPE,:VARLENG = LENGTH,:VAROCTET = OCTET_LENGTH,
      :VARPREC = PRECISION,:VARSCALE = SCALE,:VARNULL = NULLABLE,
      :VARNAME = NAME,:VARCHAR = CHARACTER_SET_NAME;
}
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CMDAREAにDELETE文:探索を設定します。

(3) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(4) SQL文識別子CMDに対応する被準備文の動的パラメタ指定の情報を記述子名DESC1のSQL記述子域に取り込みます。

(5) 記述子名DESC1のSQL記述子域からCOUNTの値を取得します。

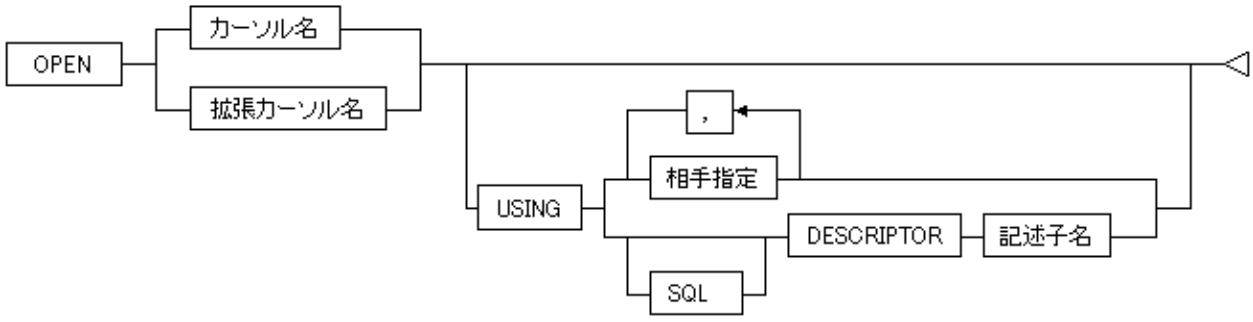
(6) 記述子名DESC1のSQL記述子域の項目記述子域からTYPE、LENGTH、OCTET_LENGTH、PRECISION、SCALE、NULLABLE、NAME、CHARACTER_SET_NAMEの値を取得します。

4.15 OPEN文(動的OPEN文)

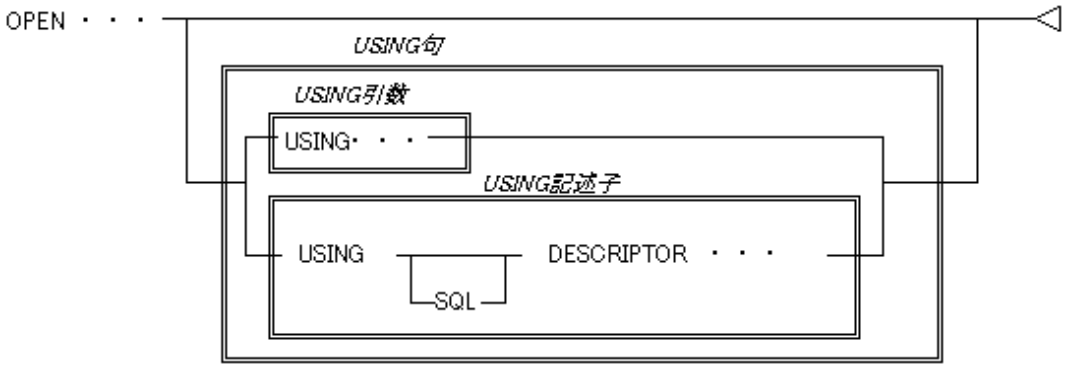
機能

動的OPEN文は、入力変数とカーソル指定を関連づけ、動的カーソル宣言またはALLOCATE CURSOR文で指定したカーソルを開きます。

記述形式



構文の構成



参照項番

・ 相手指定 → “2.3 値指定と相手指定”

権限

・ OPEN文(動的OPEN文)を実行できるのは、カーソルの処理対象の表に対するSELECT権の保持者です。

一般規則

・ カーソル名および拡張カーソル名に関連する被準備文は動的SELECT文であることが必要です。

・ カーソル名または拡張カーソル名、およびUSING句以外の規則は、“3.49 OPEN文”の一般規則を参照してください。

カーソル名または拡張カーソル名

― カーソルの名前を指定します。

― 動的カーソル宣言で定義したカーソルを開く場合は、カーソル名を指定します。

― ALLOCATE CURSOR文で定義したカーソルを開く場合は、拡張カーソル名を指定し、拡張カーソル名の値にカーソルの名前を指定します。

― 拡張カーソル名は、文字列型の埋込み変数で指定します。拡張カーソル名の値に空白を含む場合は、前後の空白を取り除いた値がカーソルの名前になります。

― カーソル名を指定した場合、カーソル名は、同一コンパイル単位に含まれる動的カーソル宣言で定義されていることが必要です。

― カーソル名と拡張カーソル名の値が同じでも、異なるカーソルとして区別されます。

USING句

― カーソル名に関連する被準備文が動的パラメタ指定を含む場合、USING句が指定されていることが必要です。

― USING句にUSING引数を指定した場合、引数には被準備文の動的パラメタ指定の値を設定します。設定する内容については、“表4.1 取得(設定)識別子に設定される値”を参照してください。

- 引数の順序は動的パラメタの順序に対応していることが必要です。

- 引数の個数は動的パラメタ指定の個数と同じであることが必要です。

- 引数のデータ型は動的パラメタ指定のデータ型と比較可能であることが必要です。

― USING句にUSING記述子を指定した場合、記述子域には被準備文の動的パラメタ指定の値の情報を指定します。

- 項目記述子域の順序は動的パラメタ指定の順序に対応していることが必要です。

- SQLDA構造体のメンバSQLTYPE、SQLLEN、SQLPRECISION.PRECISION、SQLSCALE、SQLNAME、SQLPRECISION.INTERVALまでの情報は、入力DESCRIBE文を実行することにより設定されているので、記述子域の内容を変更しない限り設定する必要はありません。ただし、日時型、時間隔型は文字列型に変更する必要があります。

- SQL記述子域のSQLTYPE、LENGTH、OCTET_LENGTH、PRECISION、SCALE、CHARACTER_SET_NAME、NAME、DATETIME_INTERVAL_CODE、DATETIME_INTERVAL_PRECISIONまでの情報は、入力DESCRIBE文を実行することにより設定されているので、記述子域の内容を変更しない限り設定する必要はありません。ただし、日時型、時間隔型は文字列型に変更する必要があります。

使用例

例1

SQLDA構造体を使用して、SQL文識別子CMDに対応する被準備文の動的パラメタ指定の情報を取得します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
#include "qdbnosis.h"

EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
    SQLDA   *sqlda_in;
EXEC SQL END DECLARE SECTION;
            :
size = sizeof(SQLDA) + sizeof(sqlvar) * (50 - 1 );

sqlda_in = (SQLDA *)malloc(size);                                    (1)
memset( sqlda_in, 0x00, size );
sqlda_in->SQLN = 50;

EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                    (2)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4=?");  (3)
CMDAREA.sqllen = strlen( CMDAREA.sqlvar );

EXEC SQL PREPARE CMD FROM :CMDAREA;                                  (4)

EXEC SQL DESCRIBE INPUT CMD USING SQL DESCRIPTOR :sqlda_in;          (5)
            :
  (SQLDA構造体に必要な情報を設定)
            :
EXEC SQL OPEN CU1 USING SQL DESCRIPTOR :sqlda_in;                    (6)
EXEC SQL FETCH CU1 INTO :COL1,:COL2,:COL3;
            :
EXEC SQL CLOSE CU1;
```

(1) SQLDA構造体sqlda_inの領域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREAに動的SELECT文を設定します。

(4) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(5) SQL文識別子CMDに対応する被準備文の動的パラメタの情報をSQLDA構造体sqlda_inに取り込みます。

(6) USING句を指定して動的パラメタ指定の値を関連づけて、動的OPEN文によりカーソルをオープンします。

例2

SQL記述子域を使用して、SQL文識別子CMDに対応する被準備文の動的パラメタ指定の情報を取得します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                    (1)
EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                 (2)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = ?");  (3)
```

```
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                    (4)
EXEC SQL DESCRIBE INPUT CMD USING SQL DESCRIPTOR 'DESC1';              (5)
                    :
  (SQL記述子域に必要な情報を設定)
                    :
EXEC SQL OPEN CU1 USING SQL DESCRIPTOR 'DESC1';                        (6)
EXEC SQL FETCH CU1 INTO :COL1,:COL2,:COL3;
                    :
EXEC SQL CLOSE CU1;
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREAに動的FETCH文を設定します。

(4) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(5) SQL文識別子CMDに対応する被準備文の動的パラメタ指定の情報を記述子名DESC1のSQL記述子域に取り込みます。

(6) USING句を指定して動的パラメタ指定の値を関連づけて、動的OPEN文によりカーソルをオープンします。

例3

拡張カーソル名を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CURVAR[4];
     CHAR CMDVAR[4];
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                    :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                      (1)
strcpy(CURVAR,"CU1");                                                  (2)
strcpy(CMDVAR,"CMD");                                                  (3)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2,COL3 FROM S1.TBL WHERE COL4 = ?");(4)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;                                (5)
EXEC SQL ALLOCATE :CURVAR CURSOR FOR :CMDVAR;                          (6)
EXEC SQL DESCRIBE INPUT :CMDVAR USING SQL DESCRIPTOR 'DESC1';          (7)
                    :
  (SQL記述子域に必要な情報を設定)
                    :
EXEC SQL OPEN :CURVAR USING SQL DESCRIPTOR 'DESC1';                    (8)
EXEC SQL FETCH :CURVAR INTO SQL DESCRIPTOR 'DESC2';
                    :
EXEC SQL CLOSE :CURVAR;
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CURVARにカーソルの名前CU1を設定します。

(3) ホスト変数CMDVARに被準備文の名前CMDを設定します。

(4) ホスト変数CMDAREAに動的SELECT文を設定します。

(5) ホスト変数CMDAREAが示すSQL文を拡張SQL文識別子CMDVARの値CMDに対応づけます。

(6) 拡張カーソル名CURVARの値CU1と拡張SQL文識別子CMDVARの値CMDを対応づけます。

(7) 拡張SQL文識別子CMDVARの値CMDに対応する被準備文の動的パラメタ指定の情報を記述子名DESC1のSQL記述子域に取り込みます。

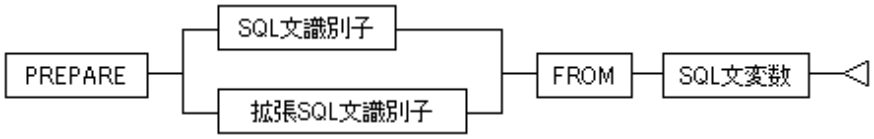
(8) USING句を指定して動的パラメタ指定の値を関連づけて、動的OPEN文によりカーソルをオープンします。

4.16 PREPARE文

機能

動的に実行するSQL文を準備します。

記述形式



参照項番

- 日本語文字列 → “2.1.3 トークン”
- 区切り識別子 → “2.1.3 トークン”

権限

- PREPARE文を実行できるのは、準備可能文に対する権限の保持者です。

一般規則

- SQL文変数にSQL文を設定して、PREPARE文を実行することにより、そのSQL文の実行の準備が完了します。SQL文変数に指定できるSQL文を準備可能文といい、PREPARE文により実行の準備が完了した文を被準備文といいます。
- すでに、PREPARE文で定義されたSQL文識別子または拡張SQL文識別子に対して、再度PREPARE文を実行する場合は、先の被準備文が解放され、新たに被準備文が準備されます。

  先の被準備文が動的SELECT文の場合、拡張SQL文識別子に対応するALLOCATE CURSOR文で定義したカーソルも同時に解放されます。

  先の被準備文が動的SELECT文の場合、SQL文識別子または拡張SQL文識別子に対応するカーソルを参照している準備可能動的DELETE文:位置づけおよび準備可能動的UPDATE文:位置づけの被準備文も同時に解放されます。
- トランザクションが異なる場合でも、SQL文識別子または拡張SQL文識別子は有効となります。

SQL文識別子または拡張SQL文識別子

- 被準備文の名前を指定します。
- SQL文識別子には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。英字の小文字は対応する英字の大文字に変換されます。英字の大文字と小文字を区別する場合は、区切り識別子で指定します。詳細は、“2.1.3 トークン”を参照してください。
- 拡張SQL文識別子は、文字列型の埋込み変数で指定します。拡張SQL文識別子の値が被準備文の名前になります。拡張SQL文識別子の値には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。英字の大文字と小文字は区別されます。
- 拡張SQL文識別子の値に空白を含む場合は、前後の空白を取り除いた値が被準備文の名前になります。
- SQL文識別子と拡張SQL文識別子の値が同じでも、異なる識別子として区別されます。

SQL文変数

- SQL文を設定した変数(埋込み変数)の名前を指定します。
- SQL文変数のデータ型は、文字列型であることが必要です。
- SQL文変数に設定するSQL文には、SQL先頭子(EXEC SQL)、SQL終了子(END-EXECまたは;)、注釈およびホスト変数を含むことはできません。

ー SQL文変数に設定するSQL文に埋込み変数を指定する場合には、動的パラメタ指定で指定します。

動的パラメタ指定は“2.3 値指定と相手指定”を参照してください。

ー SQL文変数の内容が準備可能動的DELETE文:位置づけまたは準備可能動的UPDATE文:位置づけの場合、指定したカーソル名は、以下のいずれかと同じであることが必要です。ただし、両方と同じである場合は、例外(あいまいなカーソル名)となります。

- 同一コンパイル単位に含まれる動的カーソル宣言のカーソル名
- 同一セションで実行されたALLOCATE CURSOR文の拡張カーソル名の値

また、指定したカーソルがオープンされていることが必要です。

ー SQL文変数に設定したSQL文が動的SELECT文の場合、PREPARE文で指定しているSQL文識別子または拡張SQL文識別子に対応するカーソルは、閉じられた状態であることが必要です。

使用例

例1

準備可能文が、DELETE文:探索の場合を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
strcpy(CMDAREA.sqlvar," DELETE FROM S1.TBL WHERE COL1 = 39");        (1)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                  (2)
EXEC SQL EXECUTE CMD;                                                (3)
```

(1) ホスト変数CMDAREAにDELETE文:探索を設定します。

(2) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(3) SQL文識別子CMDに対応づけられた被準備文を実行します。

これは、次のSQL文と同じです。

```
EXEC SQL DELETE FROM S1.TBL WHERE COL1 = 39;
```

例2

拡張SQL文識別子を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CMDVAR[4];
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
               :
strcpy(CMDVAR,"CMD");                                          (1)
strcpy(CMDAREA.sqlvar,"DELETE FROM S1.TBL WHERE COL1 = 39");   (2)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;                        (3)
EXEC SQL EXECUTE :CMDVAR;                                      (4)
EXEC SQL DEALLOCATE PREPARE :CMDVAR;                           (5)
```

(1) ホスト変数CMDVARに被準備文の名前CMDを設定します。

(2) ホスト変数CMDAREAにDELETE文:探索を設定します。

(3) ホスト変数CMDAREAが示すSQL文を拡張SQL文識別子CMDVARの値CMDに対応づけます。

(4) 被準備文を実行します。

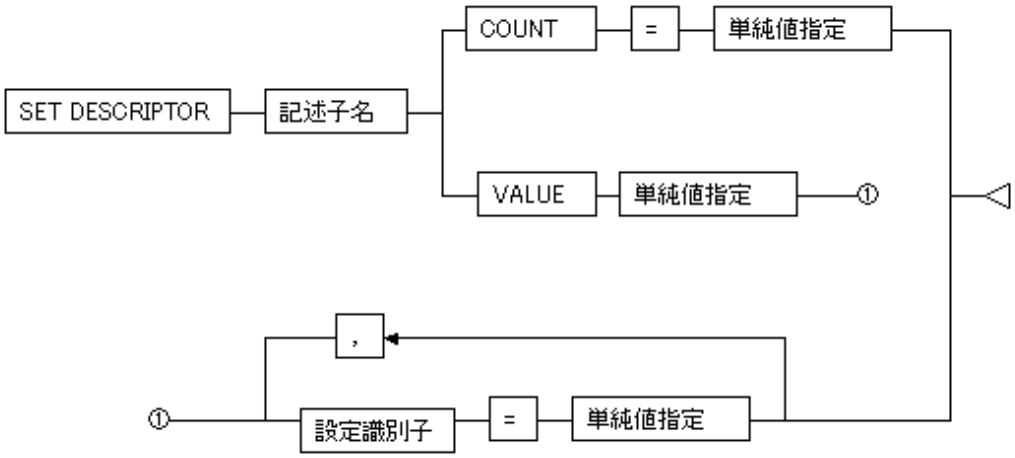
(5) 拡張SQL文識別子CMDVARの値CMDに対応する被準備文を解放します。

4.17 SET DESCRIPTOR文(DESCRIPTOR設定文)

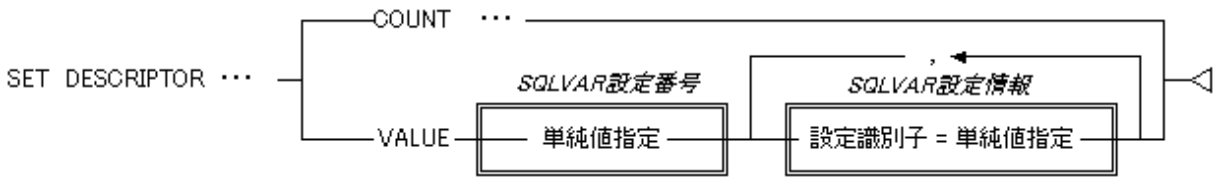
機能

SQL記述子域に情報を設定します。

記述形式



構文の構成



参照項番

- 単純値指定 → “2.3 値指定と相手指定”

一般規則

- 指定された記述子名を持つSQL記述子域が割り当てられていることが必要です。
- 同一の記述子名のUSING記述子が動的OPEN文に指定されている場合は、動的OPEN文に指定しているカーソルは、閉じられた状態であることが必要です。
- 同一の記述子名のUSING記述子が動的OPEN文、動的FETCH文、EXECUTE文に指定され、これらの文の実行が終了している場合にDESCRIPTOR設定文でDATAまたはINDICATOR以外の値を設定すると、先のSQL記述子域の内容は失われます。

記述子名

ー SQL記述子域の名前を指定します。記述子名は、文字列定数または埋込み変数で指定します。SQLDA構造体変数の変数名は指定できません。

COUNT

ー COUNTを指定した場合は、単純値指定には、項目記述子域に設定する動的パラメタ指定、または相手指定の個数を設定します。

ー COUNTを指定した場合の単純値指定のデータ型は2進の精度を持つ真数であることが必要です。

SQLVAR設定番号

ー 設定する項目記述子域の番号を指定します。SQLVAR設定番号は、1からALLOCATE DESCRIPTOR文で指定した実現値の値の範囲内であることが必要です。

ー SQLVAR設定番号のデータ型は、2進の精度を持つ真数であることが必要です。

SQLVAR設定情報

ー SQLVAR設定情報は、DESCRIPTOR設定文で設定する情報です。

ー 設定識別子には、SQLVAR設定番号で指定された、項目記述子域の情報を設定します。設定する内容および一般規則を以下に示します。

- DATAまたはINDICATOR以外の値を設定する場合、DATAの値は未定義になります。DATAの値が1つのDESCRIPTOR設定文の中で設定する複数の値のうちの1つである場合、DATAの値は最後に代入されます。

設定識別子	設定する内容	データ型
TYPE	データ型のコード	2進の精度を持つ真数
LENGTH	・文字列の最大文字長(文字列型または各国語文字列型の場合) ・データ長（BLOB型、ROW_IDの場合） ・無視(真数型または概数型の場合) （注1）	2進の精度を持つ真数
OCTET_LENGTH	・文字列の最大バイト数(文字列型または各国語文字列型の場合) ・無視(真数型または概数型の場合) （注1）	2進の精度を持つ真数
PRECISION	・精度(真数型・概数型) ・無視(真数型でも概数型でもない場合)	2進の精度を持つ真数
SCALE	・位取り(真数型・概数型) ・無視(真数型でも概数型でもない場合)	2進の精度を持つ真数
NULLABLE	・1(動的パラメタ指定に対するINDICATORの値を有効にする場合) ・0(動的パラメタ指定に対するINDICATORの値を無視する場合)	2進の精度を持つ真数
INDICATOR	指定された項目記述子域に対応する動的パラメタ指定に対する標識変数の値	2進の精度を持つ真数
DATA	・指定された項目記述子域に対応する動的パラメタ指定の値	TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, CHARACTER_SET_NA

設定識別子	設定する内容	データ型
	・ INDICATORの値が負の場合、DATAの値はNULL値になる	ME, DATETIME_INTERVAL_CODEで指定されるデータ型に一致
NAME	設定することはできない	長さ38バイト以上の文字列型
CHARACTER_SET_NAME	・ BASIC(文字列型) ・ NCHAR(各国語文字列型) ・ 空白(文字列型でも各国語文字列型でもない場合)	長さ38バイト以上の文字列型
DATETIME_INTERVAL_CODE	・ 0 (日時型でも時間隔型でもない場合) ・ 日時型の場合のデータ型のコード ・ 時間隔型の場合の時間隔修飾子のコード （注2）	2進の精度を持つ真数
DATETIME_INTERVAL_PRECISION	時間隔先行フィールド精度	2進の精度を持つ真数

注1) 日時型、時間隔型は文字列で表現したデータ長

注2) コードの値は“表4.5 SQLTYPEの値が10を示す場合のDATETIME_INTERVAL_CODEの値”を参照してください。

データ型に対応する各々の値は、“表4.7 SQLのデータ型とSQL記述子域の値の対応”を参照してください。

使用例

例

SQL文識別子CMDに対応する被準備文の動的パラメタ指定の値を設定します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
strcpy(CMDAREA.sqlvar,"DELETE FROM S1.TBL WHERE COL1 = ? AND COL2 = ?");   (2)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                        (3)
EXEC SQL DESCRIBE INPUT CMD USING SQL DESCRIPTOR 'DESC1';                  (4)
                        :
        (DESC1の内容を取得します)
                        :
for(VARWCOUNT = 1 ; VARWCOUNT <= VARCOUNT ; ++VARWCOUNT) {
    EXEC SQL SET DESCRIPTOR 'DESC1' VALUE :VARWCOUNT                       (5)
    DATA = :VARDATA;
}
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CMDAREAにDELETE文:探索を設定します。

(3) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(4) SQL文識別子CMDに対応する被準備文の動的パラメタ指定の情報を記述子名DESC1のSQL記述子域に取り込みます。

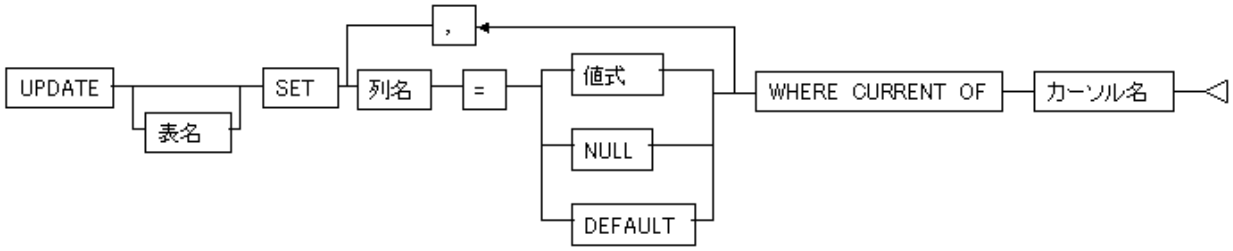
(5) 記述子名DESC1のSQL記述子域の項目記述子にDATAの値を設定します。

4.18 UPDATE文(準備可能動的UPDATE文:位置づけ)

機能

動的カーソル宣言またはALLOCATE CURSOR文で指定したカーソルによって位置づけられた1行を更新します。

記述形式



参照項番

- 区切り識別子 → “2.1.3 トークン”
- 値式 → “2.11 値式”

権限

- 準備可能動的UPDATE文:位置づけを実行できるのは、処理対象の表のUPDATE権の保持者です。

一般規則

- 準備可能動的UPDATE文:位置づけは、このSQL文をホスト変数に格納し、PREPARE文で実行の準備を行った後、EXECUTE文により実行します。また、このSQL文をホスト変数に格納した後、EXECUTE IMMEDIATE文で実行します。埋込みSQL文として、実行することはできません。必ず、準備してから実行する必要があります。
- 表名、カーソル名および上記以外の規則は、“3.66 UPDATE文:位置づけ”を参照してください。

表名

— 更新する表の名前を指定します。

— 表名を省略した場合、対象の表は、カーソルに対応する被準備文の内容である動的SELECT文のFROM句で指定された表になります。

カーソル名

— 動的カーソル宣言で定義したカーソル名、またはALLOCATE CURSOR文の拡張カーソル名で定義したカーソルの名前を指定します。

— カーソル名に英字の小文字を指定した場合、対応する英字の大文字に変換されます。英字の大文字と小文字を区別する場合は、区切り識別子で指定します。詳細は、“2.1.3 トークン”を参照してください。

— 準備可能動的UPDATE文:位置づけに指定したカーソル名は、以下のいずれかと同じであることが必要です。ただし、両方と同じである場合は、例外(あいまいなカーソル名)となります。

- 同一コンパイル単位に含まれる動的カーソル宣言のカーソル名
- 同一セションで実行されたALLOCATE CURSOR文の拡張カーソル名の値

使用例

例1

CU1のカーソルで位置づけられた行を更新します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA1[100];
    VARCHAR CMDAREA2[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
EXEC SQL DECLARE CU1 CURSOR FOR CMD1;                                      (2)
strcpy(CMDAREA1.sqlvar,"SELECT COL1,COL2 FROM S1.TBL WHERE COL4 = 38");    (3)
CMDAREA1.sqllen = strlen(CMDAREA1.sqlvar);
strcpy(CMDAREA2.sqlvar,"UPDATE S1.TBL SET T1 = 10 WHERE CURRENT OF CU1");  (4)
CMDAREA2.sqllen = strlen(CMDAREA2.sqlvar);
EXEC SQL PREPARE CMD1 FROM :CMDAREA1;                                      (5)
EXEC SQL DESCRIBE OUTPUT CMD1 USING SQL DESCRIPTOR 'DESC1';                (6)
                :
  (SQL記述子域に必要な情報を設定)
                :
EXEC SQL OPEN CU1;                                                         (7)
EXEC SQL PREPARE CMD2 FROM :CMDAREA2;                                      (8)
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR 'DESC1';                            (9)
EXEC SQL EXECUTE CMD2;                                                     (10)
EXEC SQL CLOSE CU1;                                                        (11)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREA1に動的SELECT文を設定します。

(4) ホスト変数CMDAREA2に準備可能動的UPDATE文:位置づけを設定します。

(5) ホスト変数CMDAREA1が示すSQL文をSQL文識別子CMD1に対応づけます。

(6) SQL文識別子CMD1に対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(7) 動的OPEN文によりカーソルをオープンします。

(8) ホスト変数CMDAREA2が示すSQL文をSQL文識別子CMD2に対応づけます。

(9) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(10) 準備可能動的UPDATE文:位置づけにより位置づけ行を更新します。

(11) 動的CLOSE文によりカーソルをクローズします。

例2

拡張カーソル名を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CURVAR[4];
     CHAR CMDVAR1[5];
     CHAR CMDVAR2[5];
     VARCHAR CMDAREA1[100];
     VARCHAR CMDAREA2[100];
EXEC SQL END DECLARE SECTION;
               :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
strcpy(CURVAR,"CU1");                                                      (2)
strcpy(CMDVAR1,"CMD1");                                                    (3)
strcpy(CMDVAR2,"CMD2");                                                    (4)
strcpy(CMDAREA1.sqlvar,"SELECT COL1,COL2 FROM S1.TBL WHERE COL4 = 38");    (5)
```

```
CMDAREA1.sqllen = strlen(CMDAREA1.sqlvar);
strcpy(CMDAREA2.sqlvar,"UPDATE S1.TBL SET T1 = 10 WHERE CURRENT OF CU1");(6)
CMDAREA2.sqllen = strlen(CMDAREA2.sqlvar);
EXEC SQL PREPARE :CMDVAR1 FROM :CMDAREA1;                                    (7)
EXEC SQL ALLOCATE :CURVAR CURSOR FOR :CMDVAR1;                               (8)
EXEC SQL DESCRIBE OUTPUT :CMDVAR1 USING SQL DESCRIPTOR 'DESC1';              (9)
                :
  (SQL記述子域に必要な情報を設定)
                :
EXEC SQL OPEN :CURVAR;                                                      (10)
EXEC SQL PREPARE :CMDVAR2 FROM :CMDAREA2;                                   (11)
EXEC SQL FETCH :CURVAR INTO SQL DESCRIPTOR 'DESC1';                         (12)
EXEC SQL EXECUTE :CMDVAR2;                                                  (13)
EXEC SQL CLOSE :CURVAR;                                                     (14)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CURVAR1にカーソルの名前CU1を設定します。

(3) ホスト変数CMDVAR1に被準備文(動的SELECT文)の名前CMD1を設定します。

(4) ホスト変数CMDVAR2に被準備文(準備可能動的UPDATE文:位置づけ)の名前CMD2を設定します。

(5) ホスト変数CMDAREA1に動的SELECT文を設定します。

(6) ホスト変数CMDAREA2に準備可能動的UPDATE文:位置づけを設定します。

(7) ホスト変数CMDAREA1が示すSQL文を拡張SQL文識別子CMDVAR1の値CMD1に対応づけます。

(8) 拡張カーソル名CURVAR1の値CU1と拡張SQL文識別子CMDVAR1の値CMD1を対応づけます。

(9) 拡張SQL文識別子CMDVAR1の値CMD1に対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(10) 動的OPEN文によりカーソルをオープンします。

(11) ホスト変数CMDAREA2が示すSQL文を拡張SQL文識別子CMDVAR2の値CMD2に対応づけます。

(12) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(13) 準備可能動的UPDATE文:位置づけにより位置づけ行を更新します。

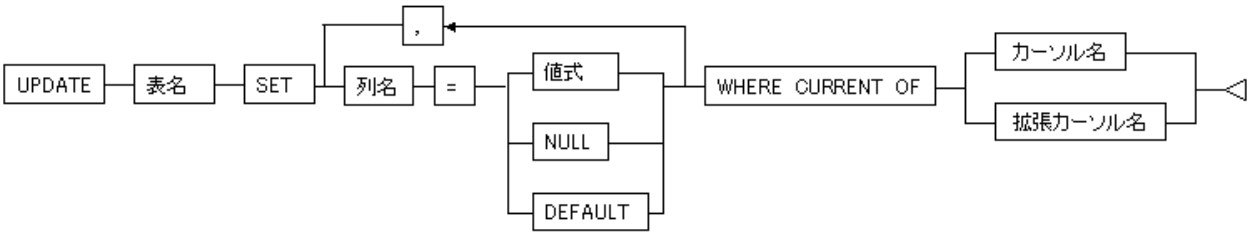
(14) 動的CLOSE文によりカーソルをクローズします。

4.19 UPDATE文(動的UPDATE文:位置づけ)

機能

動的カーソル宣言またはALLOCATE CURSOR文で指定したカーソルによって位置づけられた1行を更新します。

記述形式



参照項番

- 値式 → “2.11 値式”

権限

- 動的UPDATE文:位置づけを実行できるのは、処理対象の表のUPDATE権の保持者です。

一般規則

- カーソル名または拡張カーソル名以外の規則は、“3.66 UPDATE文:位置づけ”の一般規則を参照してください。

カーソル名または拡張カーソル名

- カーソルの名前を指定します。
- 動的カーソル宣言で定義したカーソルを更新する場合は、カーソル名を指定します。
- ALLOCATE CURSOR文で定義したカーソルを更新する場合は、拡張カーソル名を指定し、拡張カーソル名の値にカーソルの名前を指定します。
- 拡張カーソル名は、文字列型の埋込み変数で指定します。拡張カーソル名の値に空白を含む場合は、前後の空白を取り除いた値がカーソルの名前になります。
- カーソル名を指定した場合、カーソル名は、同一コンパイル単位に含まれる動的カーソル宣言で定義されていることが必要です。
- カーソル名と拡張カーソル名の値が同じでも、異なるカーソルとして区別されます。

使用例

例1

CU1のカーソルで位置づけられた行を更新します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
                :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                    (1)
EXEC SQL DECLARE CU1 CURSOR FOR CMD;                                 (2)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2 FROM S1.TBL WHERE COL4 = 38");  (3)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE CMD FROM :CMDAREA;                                  (4)
EXEC SQL DESCRIBE OUTPUT CMD USING SQL DESCRIPTOR 'DESC1';           (5)
                :
  (SQL記述子域に必要な情報を設定)
                :
EXEC SQL OPEN CU1;                                                   (6)
EXEC SQL FETCH CU1 INTO SQL DESCRIPTOR 'DESC1';                      (7)
EXEC SQL UPDATE S1.TBL SET COL1 = 10,COL2 = 20 WHERE CURRENT OF CU1;    (8)
EXEC SQL CLOSE CU1;                                                  (9)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) 動的カーソル宣言によりCU1のカーソルを宣言します。

(3) ホスト変数CMDAREAに動的SELECT文を設定します。

(4) ホスト変数CMDAREAが示すSQL文をSQL文識別子CMDに対応づけます。

(5) SQL文識別子CMDに対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(6) 動的OPEN文によりカーソルをオープンします。

(7) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(8) 動的UPDATE文:位置づけにより位置づけ行を更新します。

(9) 動的CLOSE文によりカーソルをクローズします。

例2

拡張カーソル名を指定した場合の例を示します。なお、可変長文字型の展開規則は、“6.3 SQL埋込みCプログラム”を参照してください。

```
EXEC SQL BEGIN DECLARE SECTION;
     CHAR CURVAR[4];
     CHAR CMDVAR[4];
     VARCHAR CMDAREA[100];
EXEC SQL END DECLARE SECTION;
               :
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1' WITH MAX 50;                          (1)
strcpy(CURVAR,"CU1");                                                      (2)
strcpy(CMDVAR,"CMD");                                                      (3)
strcpy(CMDAREA.sqlvar,"SELECT COL1,COL2 FROM S1.TBL WHERE COL4 = 38");     (4)
CMDAREA.sqllen = strlen(CMDAREA.sqlvar);
EXEC SQL PREPARE :CMDVAR FROM :CMDAREA;                                    (5)
EXEC SQL ALLOCATE :CURVAR CURSOR FOR :CMDVAR;                              (6)
EXEC SQL DESCRIBE OUTPUT :CMDVAR USING SQL DESCRIPTOR 'DESC1';             (7)
               :
  (SQL記述子域に必要な情報を設定)
               :
EXEC SQL OPEN :CURVAR;                                                     (8)
EXEC SQL FETCH :CURVAR INTO SQL DESCRIPTOR 'DESC1';                        (9)
EXEC SQL UPDATE S1.TBL SET COL1 = 10,COL2 = 20 WHERE CURRENT OF :CURVAR;   (10)
EXEC SQL CLOSE :CURVAR;                                                    (11)
```

(1) 記述子名DESC1のSQL記述子域を割り当てます。

(2) ホスト変数CURVARにカーソルの名前CU1を設定します。

(3) ホスト変数CMDVARに被準備文の名前CMDを設定します。

(4) ホスト変数CMDAREAに動的SELECT文を設定します。

(5) ホスト変数CMDAREAが示すSQL文を拡張SQL文識別子CMDVARの値CMDに対応づけます。

(6) 拡張カーソル名CURVARの値CU1と拡張SQL文識別子CMDVARの値CMDを対応づけます。

(7) 拡張SQL文識別子CMDVARの値CMDに対応する被準備文の選択リストの情報を記述子名DESC1のSQL記述子域に取り込みます。

(8) 動的OPEN文によりカーソルをオープンします。

(9) 動的FETCH文により記述子名DESC1のSQL記述子域に実行結果を取り込みます。

(10) 動的UPDATE文:位置づけにより位置づけ行を更新します。

(11) 動的CLOSE文によりカーソルをクローズします。

第5章 ストアドプロシジャ

クライアント側にあるアプリケーションをサーバで実行する形態の場合は、クライアントからサーバにSQL文を送信し、サーバ側の実行結果をクライアント側で受信して処理を行っています。このとき、SQL文単位で送信・受信を繰り返すため、クライアント側に通信負荷がかかります。大規模なアプリケーション開発の場合、このようなクライアント側の性能限界を解消し、開発/保守の生産性を向上させるためにはプロシジャを利用します。プロシジャとは、サーバに登録する処理手続きのことです。プロシジャをサーバに登録しておいて、クライアント側からプロシジャのルーチンを呼出し、サーバ側で一連のトランザクション処理を実行します。

5.1 ストアドプロシジャの概要

機能

ストアドプロシジャとは、複数のSQL文で記述された一連の処理手続きです。ストアドプロシジャはサーバに登録し、クライアントからCALL文で呼出して実行します。この処理手続きの定義体をプロシジャルーチンと呼びます。

プロシジャルーチンでは変数を扱うことができます。プロシジャルーチン内で使用する変数をSQL変数と呼びます。SQL変数はアプリケーションの埋込み変数と異なりデータベースのデータ型を持つため、変数の取扱いが簡単です。例えば、VARCHAR型の変数にデータを代入する場合、SQL埋込みCプログラムのように長さを設定する必要はなく直接変数に代入することができます。また、SQL変数はNULL値を設定することができるため、プロシジャルーチン内では標識変数を指定する必要はありません。

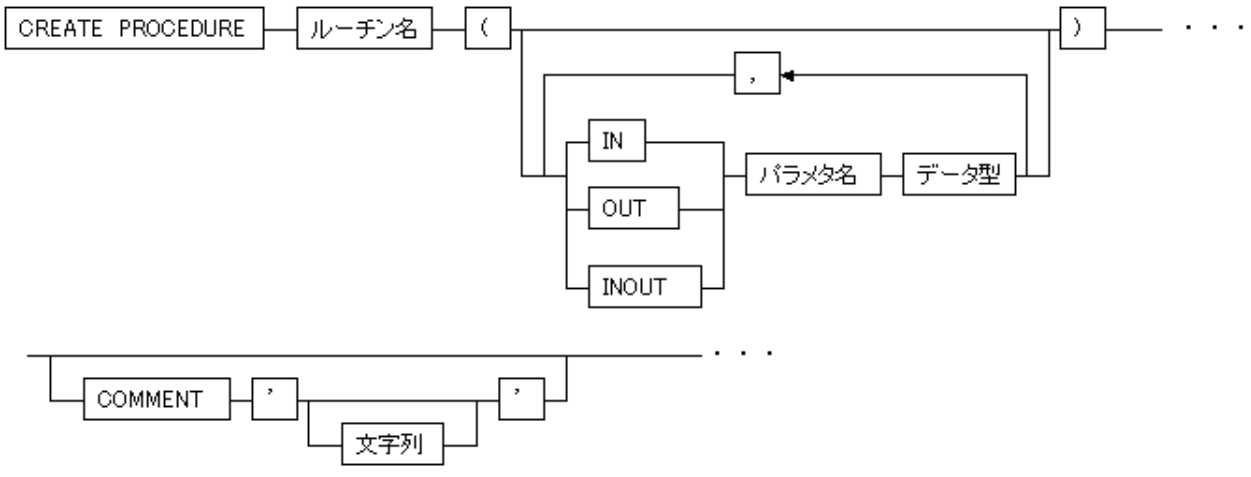
プロシジャルーチンは、INSERT文やSELECT文などのデータ操作文の他にSQL制御文を指定することができます。SQL制御文には、SQL変数に値を代入するSET文、条件に応じて処理の流れを制御するIF文やWHILE文、無条件に処理を分岐するGOTO文などがあります。これらのSQL制御文をプロシジャルーチンに指定して実行することで、アプリケーションのような処理をサーバで行うことができます。

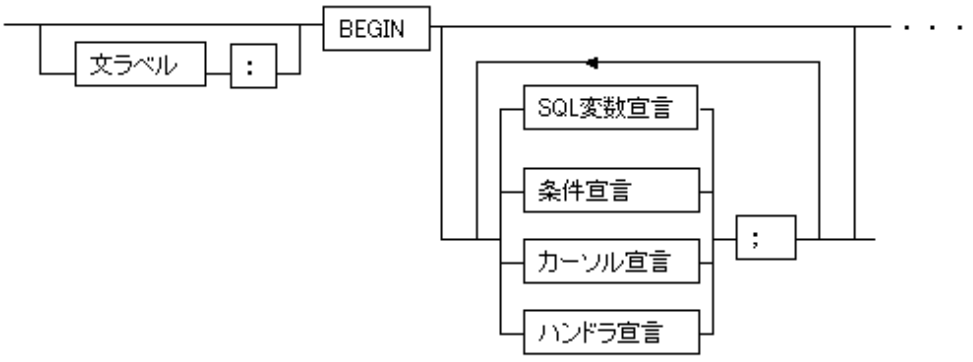
プロシジャルーチンとアプリケーションのデータの受渡しはプロシジャルーチンのパラメタで行います。アプリケーションでは、プロシジャルーチンのパラメタに対応してCALL文の引数に定数および埋込み変数を指定してデータの受渡しを行います。

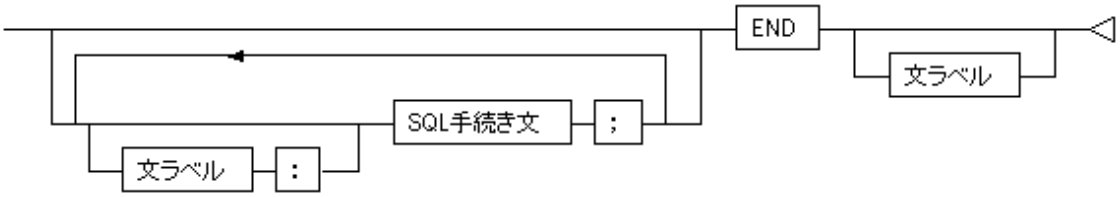
プロシジャルーチン内では、ハンドラを使用することできます。ハンドラとは、プロシジャルーチン実行中にエラーが発生した際に実行されるサブルーチンです。プロシジャルーチン実行中にエラーが発生した際にハンドラが呼び出され、特定の処理を行い、プロシジャルーチンの処理を回復することができます。ハンドラの有効範囲は複合文内です。

記述形式

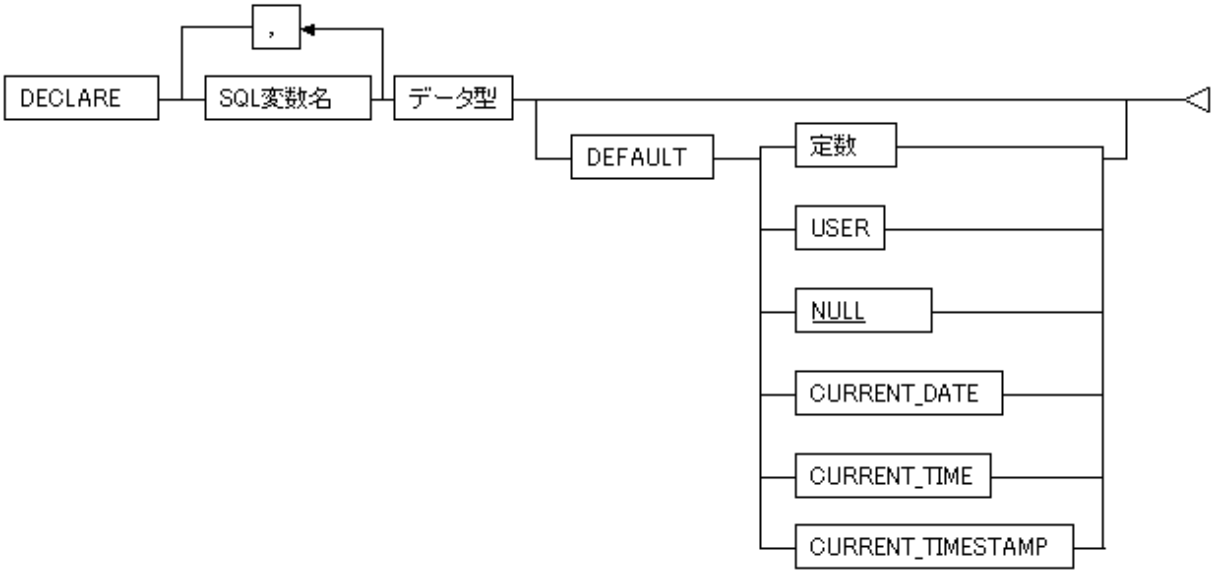
CREATE PROCEDURE



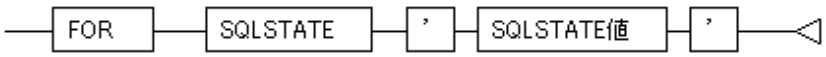




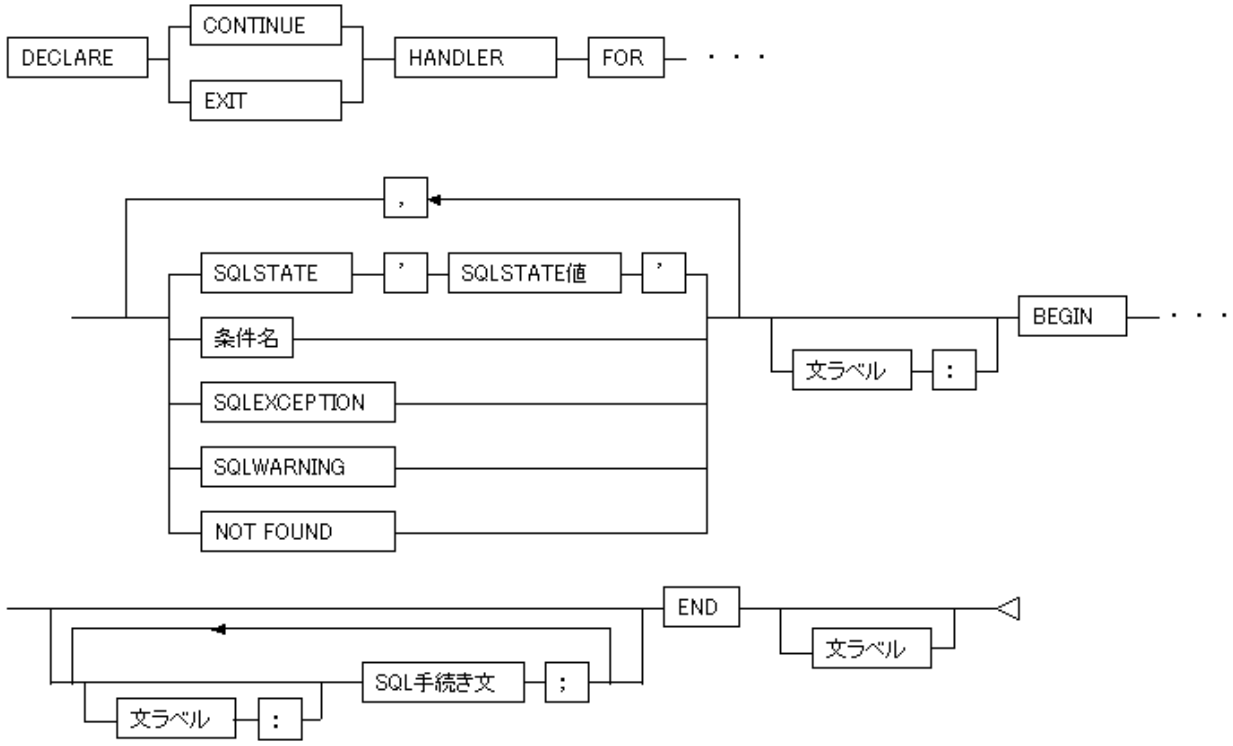
SQL変数宣言



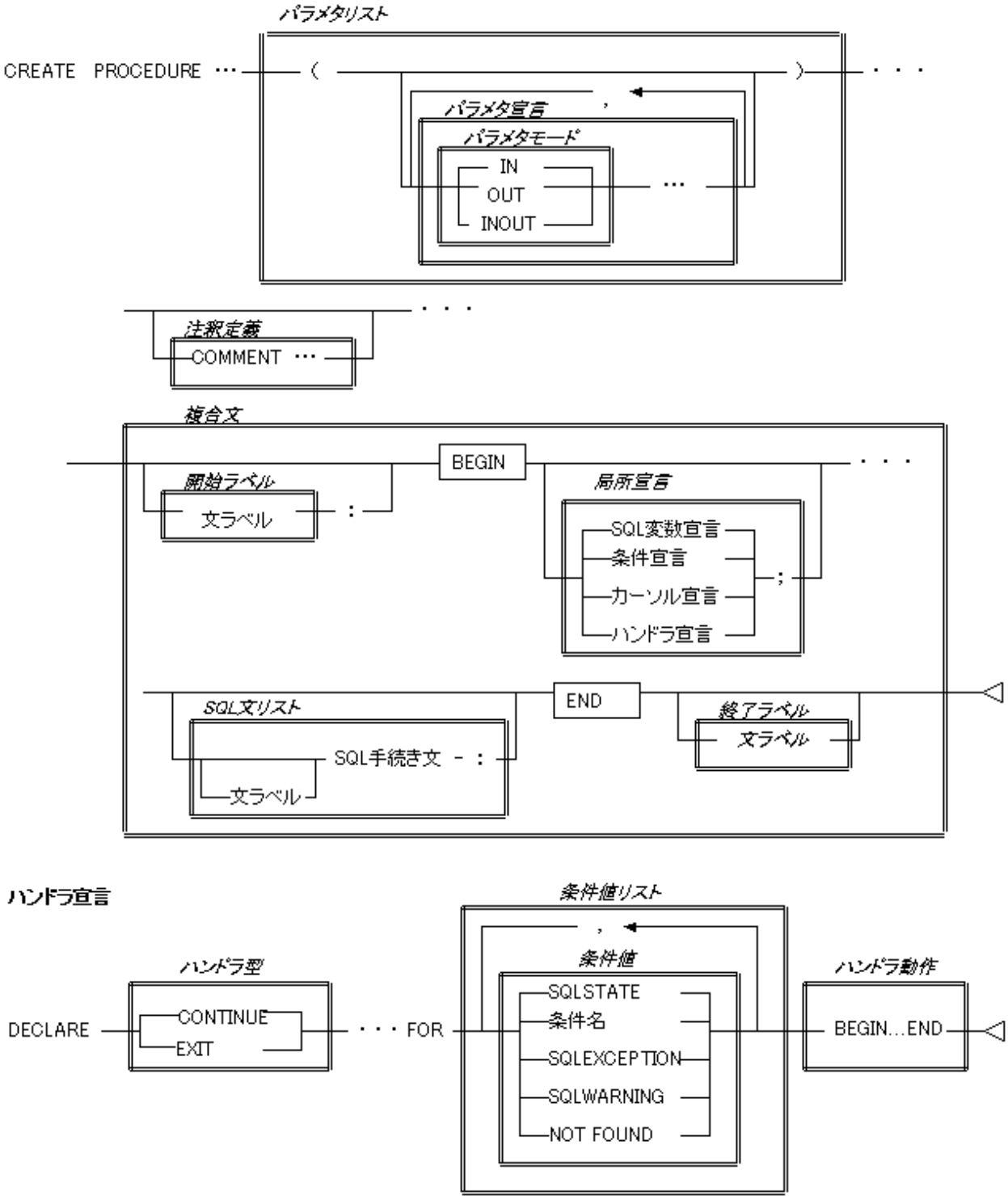
条件宣言



ハンドラ宣言


DECLARE
CONTINUE
EXIT
HANDLER
FOR
・・・
,
SQLSTATE
'
SQLSTATE値
'
条件名
SQLEXCEPTION
SQLWARNING
NOT FOUND
BEGIN
・・・
文ラベル
:
END
文ラベル
SQL手続き文
;
文ラベル
:

構文の構成



参照項番

- データ型 → “2.2 データ型”
- 定数 → “2.1.2 定数”
- 日本語文字列 → “2.1.3 トークン”

一般規則

- CALL文を指定したプロシジャルーチンを定義する場合、呼出し先のルーチンはあらかじめ定義されていることが必要です。
- 複合文中のSQL変数宣言、カーソル宣言、条件宣言およびハンドラ宣言により宣言されたSQL変数、カーソル、条件およびハンドラは、その複合文の中でのみ有効です。宣言されたSQL変数、カーソル、条件およびハンドラは複合文の終了時に破棄されます。
- パラメタリスト上のパラメタ宣言のパラメタ名とSQL変数宣言のSQL変数名には、同一名は指定できません。
- 同名のパラメタ名のパラメタ宣言は指定できません。
- 同名のSQL変数名のSQL変数宣言は指定できません。
- 同名のカーソル名のカーソル宣言は指定できません。
- 同名の条件名の条件宣言は指定できません。
- プロシジャルーチン内のSQL文で、SQLSTATE値の例外コードが40の例外事象（ロールバック例外）が発生した場合、プロシジャルーチンはそこで処理を終了し、呼出し元のアプリケーションに復帰します。このとき、トランザクションはロールバックされます。
- プロシジャルーチン内においてOBJECT型の表でサイズが32K以上のBLOB型の列は扱えません。
- プロシジャルーチン定義に行識別子“ROW_ID”は指定できません。
- BEGINとENDの間にENDが現われた場合、文法エラーとはならず複合文の処理を終了します。
- 処理手続き全体の長さは無制限です。

パラメタモード

- パラメタモードには、プロシジャの受渡しに使うパラメタのモードを指定します。パラメタモードがINの場合には、そのパラメタはプロシジャルーチンの入力に使われ、パラメタモードがOUTの場合には、そのパラメタはプロシジャルーチンの出力に使われ、パラメタモードがINOUTの場合には、そのパラメタはプロシジャルーチンの入出力に使われます。
- パラメタリスト上のパラメタ宣言のパラメタ名とSQL変数宣言のSQL変数名は、同一名を指定してはいけません。
- パラメタモードがINのパラメタは相手指定には指定できません。つまり、パラメタモードがINのパラメタはその値を参照することはできますが、値を設定することはできません。
- パラメタモードがOUTのパラメタは値指定には指定できません。つまり、パラメタモードがOUTのパラメタには値を設定することはできますが、参照することはできません。
- パラメタモードがINOUTのパラメタは値指定および相手指定に指定できます。つまり、SQL変数と同様に値の設定および参照することができます。
- パラメタモードがINおよびINOUTのパラメタは、プロシジャルーチンの実行開始時に呼出し元のCALL文の引数に指定された値を取り込みます。このとき、代入エラーが発生した場合、プロシジャルーチンは実行されずにCALL文は異常終了します。

SQL変数宣言のDEFAULT句

- SQL変数宣言にDEFAULT句が指定されている場合、プロシジャルーチンの実行開始時にDEFAULT句で指定された値がSQL変数に設定されます。
- DEFAULT句に指定する値は、SQL変数のデータ型に対して代入可能である必要があります。代入時にエラーが発生した場合、プロシジャルーチンは実行されずにCALL文は異常終了します。

SQL手続き文

- SQL手続き文とは、プロシジャルーチンに定義することができるSQL文です。プロシジャルーチンに定義できるSQL手続き文を以下に示します。

項目	文／要素
SQLデータ操作文	CLOSE文

項目	文／要素
	DECLARE CURSOR(カーソル宣言)
	DELETE文:位置づけ
	DELETE文:探索
	FETCH文
	INSERT文
	OPEN文
	Single row SELECT文(単一行SELECT文)
	UPDATE文:位置づけ
	UPDATE文:探索
トランザクション管理文	COMMIT文
	ROLLBACK文
SQL制御文	CALL文
	GOTO文
	IF文
	LOOP文
	LEAVE文
	REPEAT文
	SET文
	WHILE文
	WHENEVER文
	SIGNAL文
	RESIGNAL文

- 複合文中のSQL手続き文に指定したSQL変数およびカーソルは、複合文の局所宣言で宣言されている必要があります。
- プロシジャルーチン内の1つのSQL手続き文が1つのSQL文になります。つまり、プロシジャルーチン内のSQL文の実行でエラーが起きた場合は、エラーを起こしたSQL文が無効となります。ただし、SQLSTATE値の例外コードが40の場合は、トランザクションはロールバックされます。
- 複合文中のSQL手続き文の結果は、複合文の局所宣言で宣言されたSQLSTATE変数に設定されます。このSQLSTATE変数を使用することで以下のような条件判断を行うことができます。SQLSTATEによる例外事象に対する処理の切りわけを使用例に示します。
- プロシジャルーチン内のSQL文でエラーが発生した場合、SQLSTATE値が40003の例外事象(処理時間オーバまたは強制終了)の場合は、プロシジャルーチンはそこで処理を終了し、呼出し元のアプリケーションに復帰します。
- SQL手続き文は複数指定することができます。これを、“SQL文リスト”と呼びます。
- トリガ動作の延長で呼び出されたプロシジャルーチン内でSQLSTATE値の例外コードが40の例外が発生した場合、CALL文は異常終了し、トリガを起動する元となったSQL文を含むトランザクションが自動的にロールバックされます。
- トリガ動作の延長で呼び出されたプロシジャルーチンではトランザクション管理文を実行することはできません。トランザクション管理文を実行すると実行エラー（禁止されているSQL文の実行)になります。
- 複合文内のSQL手続き文で例外が発生したとき、複合文内にハンドラ宣言が1つ以上指定されている状態で、かつ複合文内に適切なハンドラが見つからなかった場合、プロシジャルーチンを呼び出したCALL文は異常終了します。このとき、複合文内で発生した例外事象が返却されます。

SQL制御文

― SQL制御文には以下に示すような処理の流れを制御する文があります。

- IF文は条件に応じて処理を切り分けます。
- LOOP文は無条件に処理を繰り返し、LEAVE文で繰り返し処理から抜け出します。また、LOOP文には識別用に文ラベルを指定します。これはどのLOOP文から抜け出すかをLEAVE文で指定するために使われます。
- REPEAT文とWHILE文は条件によって処理を繰り返します。REPEAT文とWHILE文は、繰り返し条件以外にLEAVE文で抜け出すことができます。この場合はREPEAT文またはWHILE文に文ラベルを指定する必要があります。
- GOTO文は無条件に処理を分岐します。分岐先は文ラベルで指定します。LOOP文、REPEAT文およびWHILE文の先頭に分岐する場合は、これらの開始ラベルをGOTO文に指定します。また、開始ラベルを持たないその他のSQL手続き文に分岐する場合は、SQL手続き文の前に文ラベルを指定しておく必要があります。
- WHENEVER文はSQL手続き文の実行時に例外事象が発生した場合にとる動作を指定します。GOTO句が指定された場合は、指定された分岐先へ分岐します。

注釈定義(COMMENT)

― 注釈は256バイト以内の文字列(日本語記述可能)を指定することができます。

ルーチン名

― 作成するプロシジャルーチンの名前を指定します。

― ルーチン名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

― ルーチン名は、スキーマ内で一意の名前であることが必要です。

― プロシジャルーチン定義がスキーマ定義に含まれる場合、ルーチン名のスキーマ名の修飾を省略した場合は、スキーマ定義で指定したスキーマ名で修飾したとみなされます。また、スキーマ名で修飾する場合は、スキーマ定義で指定したスキーマであることが必要です。

パラメタ名

― パラメタ名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

― パラメタ名にはSQL変数宣言のSQL変数名と同一名を指定することはできません。

SQL変数名

― SQL変数名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

― SQL変数名にはパラメタ宣言のパラメタ名と同一名を指定することはできません。

― SQL変数名に“SQLSTATE”と“SQLMSG”を指定した場合、プロシジャルーチン内の例外事象は“SQLSTATE”と“SQLMSG”のSQL変数に設定されます。設定された値を呼出し元に返却するには、パラメタモードがOUTもしくはINOUTのパラメタに値を設定して、呼出し元のCALL文の引数に渡す必要があります。

文ラベル(開始ラベルおよび終了ラベル)

― 複合文の先頭の文ラベルは“開始ラベル”、終端の文ラベルは“終了ラベル”と呼びます。

― 文ラベルには、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

― 開始ラベルと終了ラベルは同じラベル名を指定する必要があります。

― 開始ラベルを省略した場合、複合文の開始ラベルはルーチン名と仮定されます。したがって、終了ラベルにはルーチン名を指定します。

― ラベル名は、複合文内の他のラベル名と重複できません。

条件宣言

― 条件宣言は、特定のSQLSTATE値に条件名を定義することができます。

― 条件宣言は、有効範囲内である複合文中において、1つのSQLSTATE値には1つの条件宣言しか定義できません。

- 条件名に定義するSQLSTATE値は、0～9およびA～Zまでの任意の文字を5桁で指定します。不当な値を指定した場合、定義エラーとなります。
- 条件名に定義するSQLSTATE値に、例外コード00のSQLSTATE値を指定することはできません。

ハンドラ宣言

- 複合文中にハンドラ宣言とWHENEVER文の両方を指定することはできません。
- ハンドラ外のSQL制御文からハンドラ動作に指定された文ラベルに分岐することはできません。
- ハンドラ動作内にLEAVE文を指定してハンドラ動作を終了する場合、ハンドラ動作のBEGINからENDは文ラベルを省略することはできません。
- 条件値にSQLEXCEPTION, SQLWARNING またはNOT FOUNDを指定する場合、条件値リストにSQLSTATE値または条件名を指定することはできません。
- ハンドラ宣言の有効範囲内では、同じ条件を表現する条件値を他のハンドラ宣言に指定することはできません。
- 条件値リスト中に、同じ条件値を二度以上指定することはできません。
- 条件値リスト中に、同じ条件を表現するSQLSTATE値と条件名の両方とも指定することはできません。
- 条件値に指定するSQLSTATE値は、0～9およびA～Zまでの任意の文字を5桁で指定します。不当な値を指定した場合、定義エラーとなります。
- 条件値は、例外コード00のSQLSTATE値を指定することはできません。

条件名

- 条件名には、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

条件値

- ハンドラ宣言を記述した複合文内において、条件値リストに指定された例外条件が発生すると、ハンドラが有効化され、ハンドラ動作に指定されたSQL手続き文が実行されます。
- 条件値にSQLEXCEPTIONが指定されると、複合文内のSQL手続き文の実行結果が例外コード00(正常終了)、02(データなし)、または01(警告)のいずれでもないとき、ハンドラが関連付けられます。
- 条件値にSQLWARNINGが指定されると、複合文内のSQL手続き文の実行結果が例外コード01(警告)のとき、ハンドラが関連付けられます。
- 条件値にNOT FOUNDが指定されると、複合文内のSQL手続き文の実行結果が例外コード02(データなし)のとき、ハンドラが関連付けられます。
- 条件値に条件名が指定されると、複合文内のSQL手続き文の実行で、条件名または条件名に指定されたSQLSTATE値と同じ例外が発生したとき、ハンドラが関連付けられます。
- 条件値にSQLSTATE値が指定されると、複合文内のSQL手続き文の実行結果が同一のSQLSTATE値のとき、ハンドラが関連付けられます。
- 条件値にSQLEXCEPTION, SQLWARNING, NOT FOUNDのいずれかを指定した場合、そのハンドラ宣言は一般ハンドラ宣言と呼びます。それ以外は特有ハンドラ宣言と呼びます。複合文中に同じ例外コードの値に対する一般ハンドラ宣言と特有ハンドラ宣言がある場合、特有ハンドラ宣言がその条件値に関連付けられます。

ハンドラ型

- ハンドラ型にCONTINUEが指定された場合、以下の手順で実行されます。
  1. ハンドラ動作に指定されたSQL手続き文が実行されます。
  2. ハンドラ動作終了時、ハンドラ動作が正常終了したときは、例外条件を引き起こしたSQL手続き文の次の文から処理を再開します。ハンドラ動作で新たに例外が発生した場合、プロシジャルーチンの実行を終了し、呼出し側に例外事象が通知されます。
- ハンドラ型にEXITが指定された場合、以下の手順で実行されます。
  1. ハンドラ動作に指定されたSQL手続き文が実行されます。

2. ハンドラ動作終了後、ハンドラ動作が正常終了したときは、プロシジャルーチンの実行は正常終了します。ハンドラ動作で新たに例外が発生した場合、プロシジャルーチンの実行を終了し、呼出し側に例外事象が通知されます。

― 例外コードが40(ロールバック例外)の例外条件が発生した場合、ハンドラ型の指定に関わらず、プロシジャルーチンは処理を終了し、呼出し元のアプリケーションに復帰します。このとき、トランザクションはロールバックされます。

その他の構文要素

その他の構文要素の説明は、“3.22 CREATE TABLE文(表定義)”を参照してください。

使用例

例1

ルーチン“在庫追加”は、指定した倉庫内の製品の在庫数量を追加し、追加後の在庫数量を呼出し元に返却する処理を行います。処理の流れを以下に示します。

― 指定された製品の在庫数量に指定された数量を加えます。

― 指定製品が存在しない場合は、新たに製品情報を追加します。

― 更新後の在庫数量を取り出し、呼出し元に返却します。

```
CREATE PROCEDURE 在庫管理.在庫追加(
  IN  対象番号     SMALLINT,
  IN  対象製品名   NCHAR(10),
  IN  対象倉庫     SMALLINT,
  IN  追加数量     INTEGER,
  OUT 結果数量     INTEGER,
  OUT PRCSTATE     CHAR(5),
  OUT PRCMSG       CHAR(255)
)
BEGIN
  DECLARE SQLSTATE      CHAR(5);
  DECLARE SQLMSG        CHAR(255);

  UPDATE S1.在庫表 SET 在庫数量 = 在庫数量 + 追加数量
    WHERE 製品番号 = 対象番号 AND 製品名 = 対象製品名
      AND 倉庫番号 = 対象倉庫;
  IF SQLSTATE = '02000' THEN
     INSERT INTO S1.在庫表
       VALUES(対象番号,対象製品名,対象倉庫,追加数量);
     IF SQLSTATE <> '00000' THEN
        GOTO ERR_END;
     END IF;
  ELSEIF SQLSTATE <> '00000' THEN
       GOTO ERR_END;
  END IF;

  SELECT 在庫数量 INTO 結果数量 FROM S1.在庫表
    WHERE 製品番号 = 対象番号 AND 製品名 = 対象製品名
          AND 倉庫番号 = 対象倉庫;
  IF SQLSTATE <> '00000' THEN
     GOTO ERR_END;
  END IF;

  LEAVE 在庫追加;(注1)

ERR_END:
        SET PRCSTATE = SQLSTATE;(注2)
        SET PRCMSG   = SQLMSG;
END
```

注1) 複合文の開始ラベル省略時は、ルーチン名が文ラベルとなる。

注2) 予想外のエラーが発生した場合は、そのエラー情報を返却する。

例2

ハンドラ型CONTINUEおよび条件宣言の使用例を以下に示します。

```
CREATE PROCEDURE SCM1.PRC1(
  IN 伝票番号     INTEGER,
  IN 顧客コード   INTEGER,
  IN 顧客名       CHAR(40)
)
BEGIN
  DECLARE SQLSTATE    CHAR(5);
  DECLARE SQLMSG      CHAR(256);
  DECLARE NUM,RETRY   INTEGER;
  -- 条件宣言
  DECLARE UNIQUE_VIOLATION CONDITION FOR SQLSTATE'23000';
  -- ハンドラ1宣言
  DECLARE CONTINUE HANDLER FOR UNIQUE_VIOLATION
  BEGIN -- 動作に複合文を指定
    SET NUM   = NUM + 1;
    SET RETRY = 1;  -- retry flag
  END;
  -- ハンドラ2宣言
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN  -- その他の例外用ハンドラ
    RESIGNAL;  -- 呼出し側に例外事象を通知
  END;

  -- ルーチン本文
   SET NUM = 伝票番号;

LABEL1:
  SET RETRY = 0;
  INSERT INTO SCM1.TBL04 VALUES(NUM, 顧客コード, 顧客名);
  IF (RETRY <> 0) THEN
    GOTO LABEL1;
  END IF;
END
```

例3

ハンドラ型EXITの使用例を以下に示します。

```
CREATE PROCEDURE SCM1.PRC1(
  IN 伝票番号     INTEGER,
  IN 顧客コード   INTEGER,
  IN 顧客名       CHAR(40)
)
BEGIN
  DECLARE SQLSTATE    CHAR(5);
  DECLARE SQLMSG      CHAR(256);
  DECLARE NUM,RETRY   INTEGER;
  -- ハンドラ1宣言
  DECLARE CONTINUE HANDLER FOR SQLSTATE'23000'
  BEGIN -- 動作に複合文を指定
    SET NUM   = NUM + 1;
    SET RETRY = 1;  -- retry flag
  END;
  -- ハンドラ2宣言
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    RESIGNAL;
```

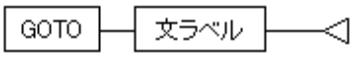
```
  END;

  -- ルーチン本文
  SET NUM = 伝票番号;
LABEL1:
  SET RETRY = 0;
  INSERT INTO SCM1.TBL04 VALUES(NUM, 顧客コード, 顧客名);
  IF (RETRY <> 0) THEN
    GOTO LABEL1;
  END IF;
END
```

5.2 GOTO文

機能

文ラベルに無条件に分岐します。

記述形式

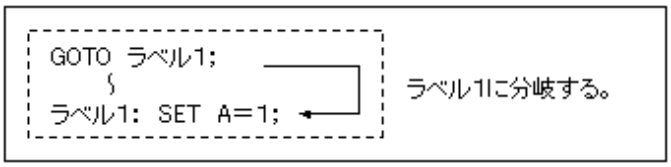


参照項番

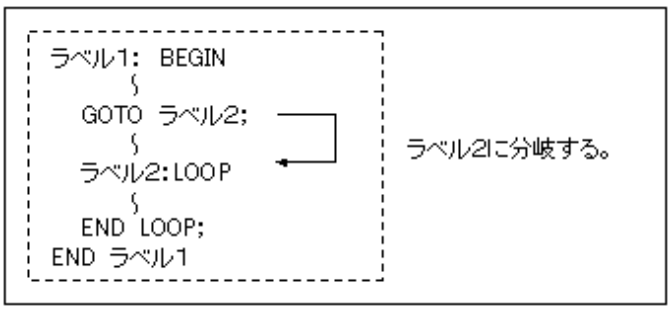
- 日本語文字列 → “2.1.3 トークン”

一般規則

- GOTO文を実行すると、指定した文ラベルに分岐します。



- GOTO文に指定した文ラベルが、SQL手続き文の開始ラベルの場合、処理ブロックの先頭に分岐します。



文ラベル

- 文ラベルは、SQL手続き文の開始ラベルまたは、分岐先文の文ラベルを指定します。
- 文ラベルには、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

使用例

例

ラベル1に分岐します。

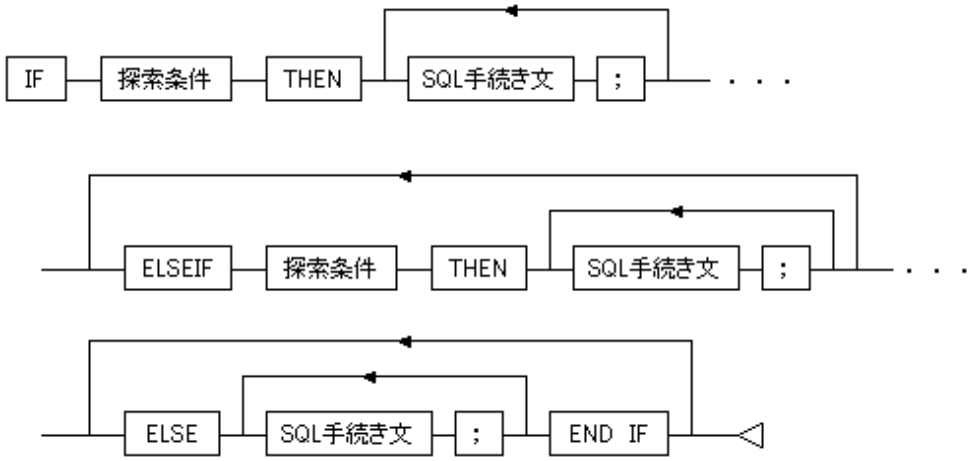
```
GOTO ラベル1

ラベル1:LOOP
    FETCH CSR1 INTO 製品番号,製品名,在庫数量,倉庫番号;
    IF SQLSTATE <> '00000' THEN
      LEAVE ラベル1;
    END IF;
    IF 在庫数量 <= 500 THEN
       SET 数量 = 500 - 在庫数量;
                :
    END IF;
END LOOP;
```

5.3 IF文

機能

条件付き実行制御を行います。

記述形式



参照項番

- 探索条件 → “2.13 探索条件”
- SQL手続き文 → “5.1 ストアドプロシジャの概要”

一般規則

- 探索条件には比較述語およびNULL述語しか記述することはできません。また、比較述語に副問合せを指定することはできません。
- 探索条件の比較述語で指定する値式のデータ型は、比較可能であることが必要です。

  比較可能なデータ型は、“2.11.1 数値式”を参照してください。
- 探索条件に指定する値式にはCASE式およびROWNUMを指定することはできません。
- 1つ以上のIF文ELSEIF句が指定されている場合、以下のようなIF文ELSEIF句を含まないように指定したIF文と等価になります。

```
IF <探索条件1> THEN
   <SQL手続き文1> ;
ELSEIF <探索条件2> THEN
   <SQL手続き文2> ;
```

```
ELSEIF 〈探索条件3〉 THEN
   〈SQL手続き文3〉 ;
ELSE
   〈SQL手続き文4〉 ;
END IF ;
```

```
IF 〈探索条件1〉 THEN
   〈SQL手続き文1〉 ;
ELSE
  IF 〈探索条件2〉 THEN
     〈SQL手続き文2〉 ;
  ELSE
    IF 〈探索条件3〉 THEN
       〈SQL手続き文3〉 ;
    ELSE
       〈SQL手続き文4〉 ;
    END IF ;
  END IF ;
END IF ;
```

- IF文の探索条件の結果は、真の場合、偽の場合、NULLの場合およびエラーの場合があります。真の場合は、THEN句の続きのSQL手続き文を実行します。他の場合、ELSE句が指定されていれば、ELSE句の続きのSQL手続き文を実行します。

SQL手続き文

— 実行するSQL文を指定します。

— SQL手続き文に複合文を指定することはできません。

使用例

例

IF文の定義をします。

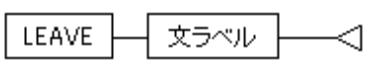
```
BEGIN
  DECLARE A INT;
  DECLARE B INT;
        :
  IF A = 1 THEN
    INSERT INTO SCM.TBL(C,D) VALUES(1,2);  ←条件が真の場合
  ELSE
    INSERT INTO SCM.TBL(C,D) VALUES(3,4);  ←条件が偽の場合
  END IF;
END
```

5.4 LEAVE文

機能

LOOP文、複合文、WHILE文、REPEAT文から脱出します。

記述形式



参照項番

- 日本語文字列 → “2.1.3 トークン”

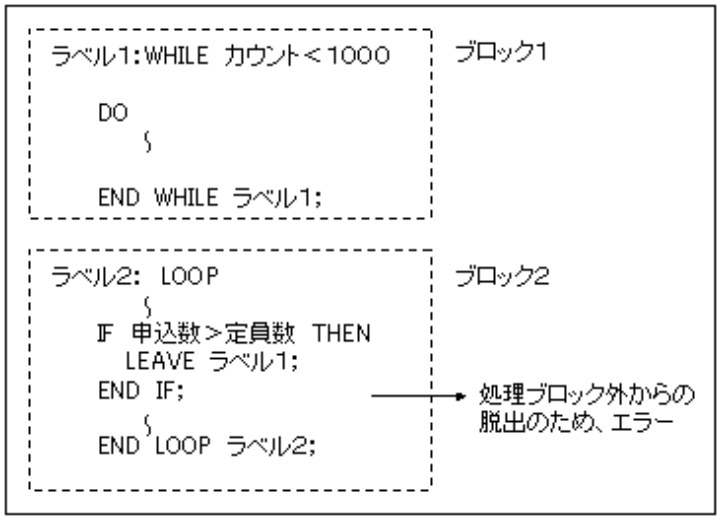
一般規則

- LEAVE文を実行すると、指定した開始ラベルの処理ブロックを終了します。
- LEAVE文は脱出する処理ブロック内に指定します。

例1: LOOP文からの脱出

```
ラベル1: LOOP
      ｛
  IF 申込数＞定員数 THEN
    LEAVE ラベル1;
  END IF;
      ｝
  END LOOP ラベル1;
```

例2: LEAVE文の誤った指定(処理ブロック外の脱出指定)



文ラベル

ー 文ラベルは、SQL手続き文の開始ラベルを指定します。

ー 文ラベルには、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

使用例

例

テーブルを全件検索して、在庫数量が500以下の場合、数量に不足分を補充します。この時、SQLSTATEが00000以外の場合、LOOP文を抜けます。

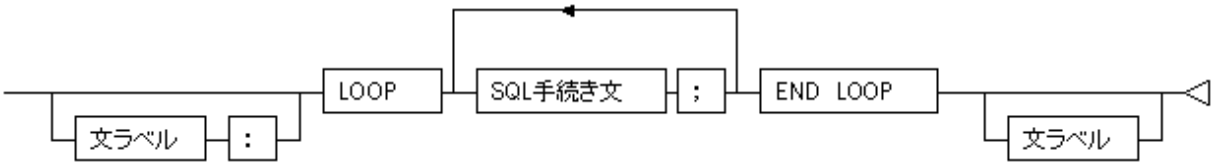
```
ラベル1:LOOP
  FETCH CSR1 INTO 製品番号,製品名,在庫数量,倉庫番号;
  IF SQLSTATE <> '00000' THEN
    LEAVE ラベル1;
  END IF;
  IF 在庫数量 <= 500 THEN
     SET 数量 = 500 - 在庫数量;
              :
  END IF;
END LOOP;
```

5.5 LOOP文

機能

文の実行を繰り返します。

記述形式



参照項番

・ SQL手続き文 → “5.1 ストアドプロシジャの概要”

一般規則

・ ループを脱出する場合には、LEAVE文を指定します。

SQL手続き文

― LOOPからEND LOOP間に記述したSQL手続き文を無条件に繰り返し実行します。

― SQL手続き文に複合文を指定してはいけません。

文ラベル(開始ラベルおよび終了ラベル)

― 先頭の文ラベルは“開始ラベル”、終端の文ラベルは“終了ラベル”と呼びます。

― 開始ラベルと終了ラベルは同じラベルを指定することが必要です。

― 終了ラベルを指定した場合、対応する開始ラベルを指定することが必要です。

― 開始ラベルを指定した場合、プロシジャルーチン内に含まれるすべての文ラベルと異なっていることが必要です。

使用例

例

テーブルを全件検索して、在庫数量が500以下の場合、数量に不足分を補充します。

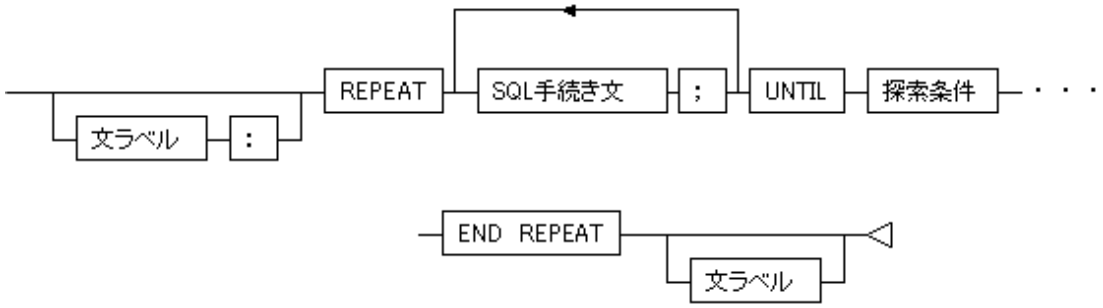
```
ラベル1:LOOP
  FETCH CSR1 INTO 製品番号,製品名,在庫数量,倉庫番号;
  IF SQLSTATE <> '00000' THEN
    LEAVE ラベル1;
  END IF;
  IF 在庫数量 <= 500 THEN
     SET 数量 = 500 - 在庫数量;
              :
  END IF;
END LOOP;
```

5.6 REPEAT文

機能

条件が真になるまで文の実行を繰り返します。

記述形式



参照項番

- SQL手続き文 → “5.1 ストアドプロシジャの概要”
- 探索条件 → “2.13 探索条件”

一般規則

- REPEAT文は、以下に示すようなLOOP文と同等の機能です。

```
ラベル:
REPEAT
      SQL手続き文  ;
UNTIL    探索条件
END REPEAT ラベル  ;
```

```
ラベル:
LOOP
      SQL手続き文  ;
  IF 探索条件 THEN
    LEAVE ラベル  ;
  END IF  ;
END LOOP ラベル  ;
```

SQL手続き文

- SQL手続き文に複合文を指定してはいけません。

探索条件

- 探索条件には比較述語およびNULL述語しか記述することはできません。また、比較述語には副問合せを指定することはできません。
- 探索条件に指定する値式にはCASE式およびROWNUMを指定することはできません。
- 探索条件が偽の間、SQL手続き文を繰り返し実行します。
- 探索条件が真になるとループ処理を終了します。

文ラベル(開始ラベルおよび終了ラベル)

- 先頭の文ラベルは“開始ラベル”、終端の文ラベルは“終了ラベル”と呼びます。
- 開始ラベルと終了ラベルは同じラベルを指定することが必要です。
- 終了ラベルを指定した場合、対応する開始ラベルを指定することが必要です。
- 開始ラベルを指定した場合、プロシジャルーチン内に含まれるすべての文ラベルと異なっていることが必要です。

使用例

例

取引先が62になるまで、検索を続けます。

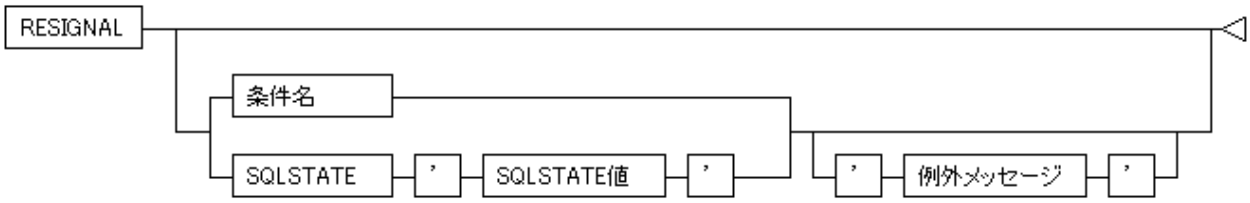
```
ラベル1: REPEAT
    FETCH CSR1 INTO 取引先,取引製品,仕入価格,発注数量;
        UNTIL 取引先 = 62
END REPEAT ラベル1;
```

5.7 RESIGNAL文

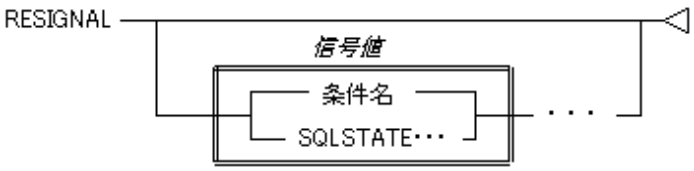
機能

例外条件を再送します。

記述形式



構文の構成



一般規則

- RESIGNAL文はハンドラ宣言のハンドラ動作内にのみ指定できます。
- 信号値を省略した場合、RESIGNAL文を含むハンドラが関連付けられた例外事象をハンドラ内で再度発生させます。
- RESIGNAL文に信号値を指定した場合、指定された例外事象を発生させます。
- SQLSTATE値を指定する場合、SQLSTATE値は、例外コードに60を指定する必要があります。副例外コードは0～9およびA～Zまでの任意の文字を3桁で指定します。
- RESIGNAL文に条件名を指定する場合、例外コードが60および副例外コードが0～9およびA～Zまでの任意の文字3桁で表現されたSQLSTATE値を定義した条件名を指定する必要があります。
- 例外メッセージは文字列型で指定します。長さは最大256バイトまでが有効となります。それ以上の長さを指定した場合、切り捨てられます。

使用例

例

RESIGNAL文を利用して、呼出し側に例外事象を通知する例を以下に示します。

```
CREATE PROCEDURE S1.在庫チェック(IN 在庫数パラ INT)
BEGIN
  DECLARE SQLSTATE  CHAR(5);
```

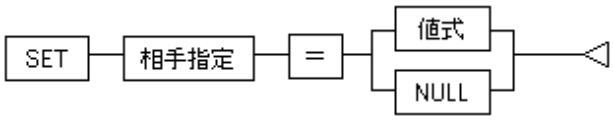
```
  DECLARE SQLMSG    CHAR(256);
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    RESIGNAL; -- 呼出し元に発生例外を通知する
END;
  -- ルーチン本体の処理
  IF (在庫数パラ < 0) THEN
  SIGNAL SQLSTATE'60001' '在庫数が不当です';
  END IF;
END
```

5.8 SET文(代入文)

機能

値を変数に設定します。

記述形式



参照項番

- 相手指定 → “2.3 値指定と相手指定”
- 値式 → “2.11 値式”

一般規則

相手指定

ー 相手指定には、SQL変数またはパラメタを指定します。

ー 相手指定(代入先)に、入力モードのプロシジャルーチンのパラメタを指定してはいけません。

値式

ー 値式(代入元)に、出力モードのプロシジャルーチンのパラメタを指定してはいけません。

ー 値式には、列指定、集合関数指定、CASE式およびROWNUMを指定することはできません。

使用例

例

値を変数Aに設定します。

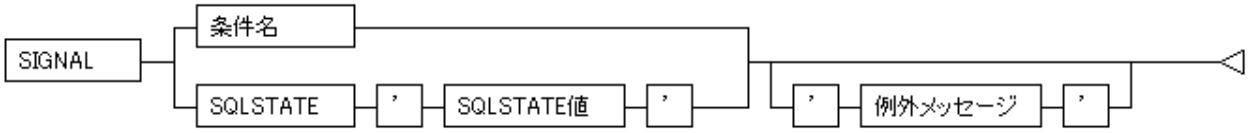
```
SET A = 10;
```

5.9 SIGNAL文

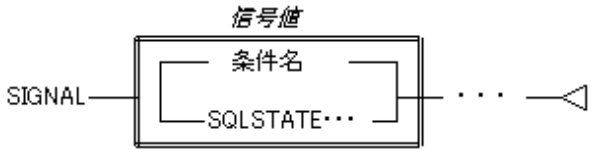
機能

例外条件を送信します。

記述形式



構文の構成



一般規則

- SIGNAL文を実行すると、例外条件が発生し、指定されたSQLSTATE値や例外メッセージがプロシジャルーチン内のSQL変数SQLSTATEおよびSQLMSGに設定されます。
- SIGNAL文に指定するSQLSTATE値は、例外コードに60を指定する必要があります。副例外コードは0～9およびA～Zまでの任意の文字を3桁で指定します。不当な値を指定した場合、定義エラーになります。
- SIGNAL文に条件名を指定する場合、例外コードが60および副例外コードが0～9およびA～Zまでの任意の文字3桁で表現されたSQLSTATE値を定義した条件名を指定する必要があります。その他のSQLSTATE値を定義した条件名を指定した場合、定義エラーになります。
- 例外メッセージは文字列型で指定します。長さは最大256バイトまでが有効となります。それ以上の長さを指定した場合、切り捨てられます。

使用例

例

SIGNAL文を利用して、ユーザが任意に例外を発生する例を以下に示します。

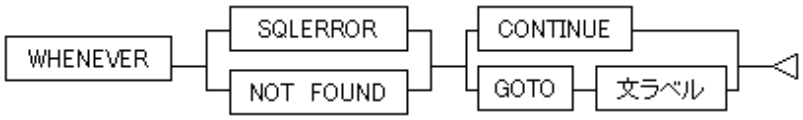
```
CREATE PROCEDURE S1.在庫チェック(IN 在庫数パラ INT)
BEGIN
  DECLARE SQLSTATE  CHAR(5);
  DECLARE SQLMSG    CHAR(256);
  DECLARE ユーザ定義例外１ CONDITION FOR SQLSTATE'60001';
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    RESIGNAL;  -- 呼出し元に発生例外を通知する
  END;
  -- ルーチン本体の処理
  IF (在庫数パラ < 0) THEN
    SIGNAL ユーザ定義例外１ '在庫数が不当です';
  END IF;
END
```

5.10 WHENEVER文

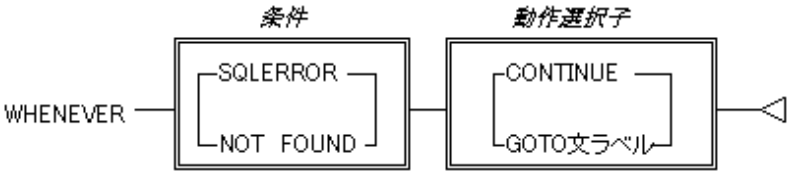
機能

プロシジャルーチン内のSQL手続き文実行時に、例外事象が発生した場合にとる動作を指定します。

記述形式



構文の構成



参照項番

- 日本語文字列 → “2.1.3 トークン”

一般規則

- WHENEVER文は、SQL手続き文中に複数回指定することができます。WHENEVER文を指定すると、以降に同じ条件のWHENEVER文が出現するまで、有効です。後続のWHENEVER文が出現すると以前の宣言は無効となります。
- WHENEVER文は、複合文、LOOP文、WHILE文、REPEAT文、IF文の階層に関係なく、指定された行から後のSQL手続き文に対して有効になります。

条件(SQLERRORおよびNOT FOUND)

- 条件にSQLERRORが指定されると、SQL手続き文の実行結果が正常終了、データなし、または警告のいずれでもない場合に、動作選択子が実行されます。
- 条件にNOT FOUNDが指定されると、SQL手続き文の実行結果がデータなしの場合に、動作選択子が実行されます。

動作選択子(CONTINUEおよびGOTO句)

- 動作選択子にCONTINUEが指定されると、条件に指定された事象の発生したSQL手続き文の次を実行します。
- 動作選択子にGOTO句が指定されると、指定した文ラベルに分岐します。
- WHENEVER文が出現する以前に例外事象が発生した場合、CONTINUEが指定されたものとみなされます。

文ラベル

- 文ラベルは、SQL手続き文の開始ラベルまたは、分岐先文の文ラベルを指定します。
- 文ラベルには、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。

使用例

例

WHENEVER文を指定します。

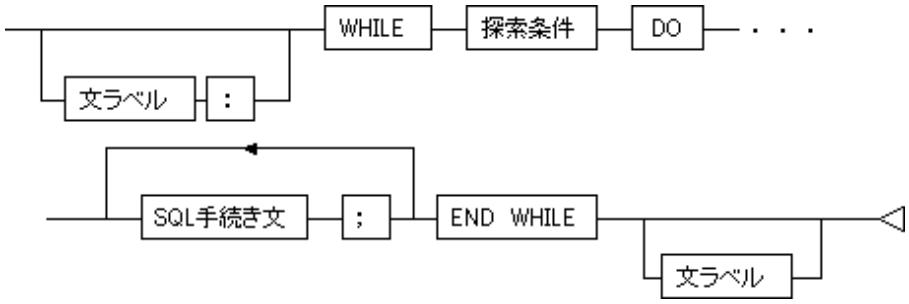
```
    WHENEVER SQLERROR GOTO エラー1;
ラベル1: LOOP
    FETCH CSR1 INTO 製品番号,製品名,在庫数量,倉庫番号;
    IF 在庫数量 <= 500 THEN
       SET 数量 = 500 - 在庫数量;
                :
    END IF;
END LOOP;
エラー1:
```

5.11 WHILE文

機能

条件が真の間、実行文を繰り返します。

記述形式



参照項番

- 探索条件 → “2.13 探索条件”
- SQL手続き文 → “5.1 ストアドプロシジャの概要”
- 日本語文字列 → “2.1.3 トークン”

一般規則

- WHILE文は、以下に示すようなLOOP文と同等の機能です。

```
ラベル:
WHILE 探索条件 DO
    SQL手続き文  ;
END WHILE ラベル  ;
```

```
ラベル:
LOOP
  IF 探索条件 THEN
    SQL手続き文  ;
  ELSE
    LEAVE ラベル  ;
  END IF  ;
END LOOP ラベル  ;
```

探索条件

- 探索条件には比較述語およびNULL述語しか記述することはできません。また、比較述語には副問合せを指定することはできません。
- 探索条件に指定する値式にはCASE式およびROWNUMを指定することはできません。
- 探索条件が真の間、SQL手続き文を繰り返し実行します。
- 探索条件が偽、NULLまたはエラーになるとループ処理を終了します。

SQL手続き文

- 実行するSQL文を指定します。
- SQL手続き文に複合文を指定してはいけません。

文ラベル(開始ラベルおよび終了ラベル)

- 先頭の文ラベルは“開始ラベル”、終端の文ラベルは“終了ラベル”と呼びます。

- 文ラベルには、36文字以内の先頭が英字で始まる英数字、または18文字以内の日本語文字列を指定します。
- 開始ラベルと終了ラベルは同じラベルを指定することが必要です。
- 終了ラベルを指定した場合、対応する開始ラベルを指定することが必要です。
- 開始ラベルを指定した場合、プロシジャルーチン内に含まれるすべての文ラベルと異なっていることが必要です。

使用例

例

取引先が61の間、発注数量を追加します。

```
ラベル1: WHILE 取引先 = 61
         DO
           SET 発注数量 = 発注数量 + 10 ;
           FETCH CSR2 INTO 取引先 ;
         END WHILE ラベル1;
```

第6章 埋込みSQL

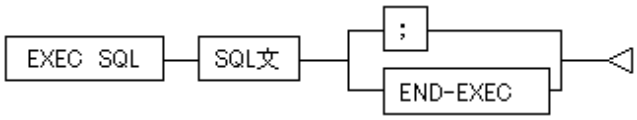
データベースに対する各種操作を行うアプリケーションを作成する場合、データベース処理部分はSQL文で記述し、ほかの処理は既存のプログラミング言語で記述するのが一般的です。このように、アプリケーション(ホストプログラム)に埋め込まれたSQL文のテキストのことを、“埋込みSQL文”と呼びます。また、SQL文が埋め込まれたホストプログラムを、“SQL埋込みホストプログラム”と呼びます。

6.1 Embedded SQL文(埋込みSQL文)

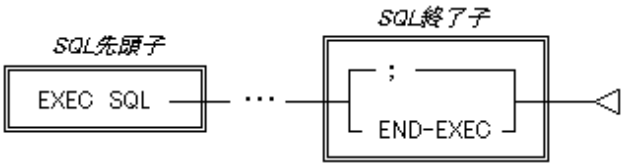
機能

埋込みSQL文は、アプリケーション(ホストプログラム)に埋め込まれたSQL文のテキストです。

記述形式



構文の構成



一般規則

- データベースに対する各種操作を行うアプリケーションを作成する場合、データベース処理部分はSQL文で記述し、ほかの処理は既存のプログラミング言語で記述するのが一般的です。このように、アプリケーション(ホストプログラム)に埋め込まれたSQL文のテキストのことを、“埋込みSQL文”と呼びます。以下に埋込みSQL文を示します。
  - DECLARE CURSOR(カーソル宣言)
  - DECLARE CURSOR(動的カーソル宣言)
  - DECLARE TABLE(表宣言)
  - WHENEVER文(埋込み例外宣言)
  - INCLUDE文
  - Single row SELECT文(単一行SELECT文)
  - DELETE文:探索
  - INSERT文
  - UPDATE文:探索
  - OPEN文
  - CLOSE文
  - FETCH文
  - DELETE文:位置づけ
  - UPDATE文:位置づけ
  - RELEASE TABLE文

- SET TRANSACTION文
- COMMIT文
- ROLLBACK文
- CONNECT文
- SET CONNECTION文
- DISCONNECT文
- SET CATALOG文
- SET SCHEMA文
- SET SESSION AUTHORIZATION文
- SET ROLE文
- SET USER PASSWORD文
- ALLOCATE DESCRIPTOR文
- DEALLOCATE DESCRIPTOR文
- GET DESCRIPTOR文(DESCRIPTOR取得文)
- SET DESCRIPTOR文(DESCRIPTOR設定文)
- PREPARE文
- DEALLOCATE PREPARE文
- DESCRIBE文
- EXECUTE文
- EXECUTE IMMEDIATE文
- OPEN文(動的OPEN文)
- FETCH文(動的FETCH文)
- CLOSE文(動的CLOSE文)
- DELETE文(動的DELETE文:位置づけ)
- UPDATE文(動的UPDATE文:位置づけ)
- CALL文

・ 埋込みSQL文は、SQL埋込みホストプログラムに少なくとも1つ以上記述されていることが必要です。

・ SQL先頭子の“EXEC”と“SQL”は同一行に記述することが必要です。

・ SQL埋込みCプログラムの中に記述される埋込みSQL文および埋込みSQL宣言節には、SQL終了子に“;(セミコロン)”を指定することが必要です。

・ SQL埋込みCOBOLプログラムの中に記述される埋込みSQL文および埋込みSQL宣言節には、SQL終了子に“END-EXEC”を指定することが必要です。

6.2 INCLUDE文

機能

別ファイルの、SQL文や対象となる言語テキストを展開します。

記述形式

一般規則

- 展開するファイルの中には、SQL文、CおよびCOBOLの言語テキストを記述できます。また、INCLUDE文は記述できません。
- INCLUDE文はSQL埋込みホストプログラム中の任意の位置に記述することができます。
- INCLUDE文はSQL埋込みホストプログラム中に複数記述することができます。

ファイル名

ー ファイル名には、展開するファイルのファイル名を指定します。

ー ファイル名は、ディレクトリつきで指定することはできません。

ー ファイル名は、各オペレーティングシステムの文法に従って記述しますが、それ以外に以下の文字を指定することはできません。

- 空白
- 改行
- シングルクォーテーション
- ダブルクォーテーション
- セミコロン

使用例

例

ホスト変数宣言節中に、ファイル“var1.h”を展開する例を示します。

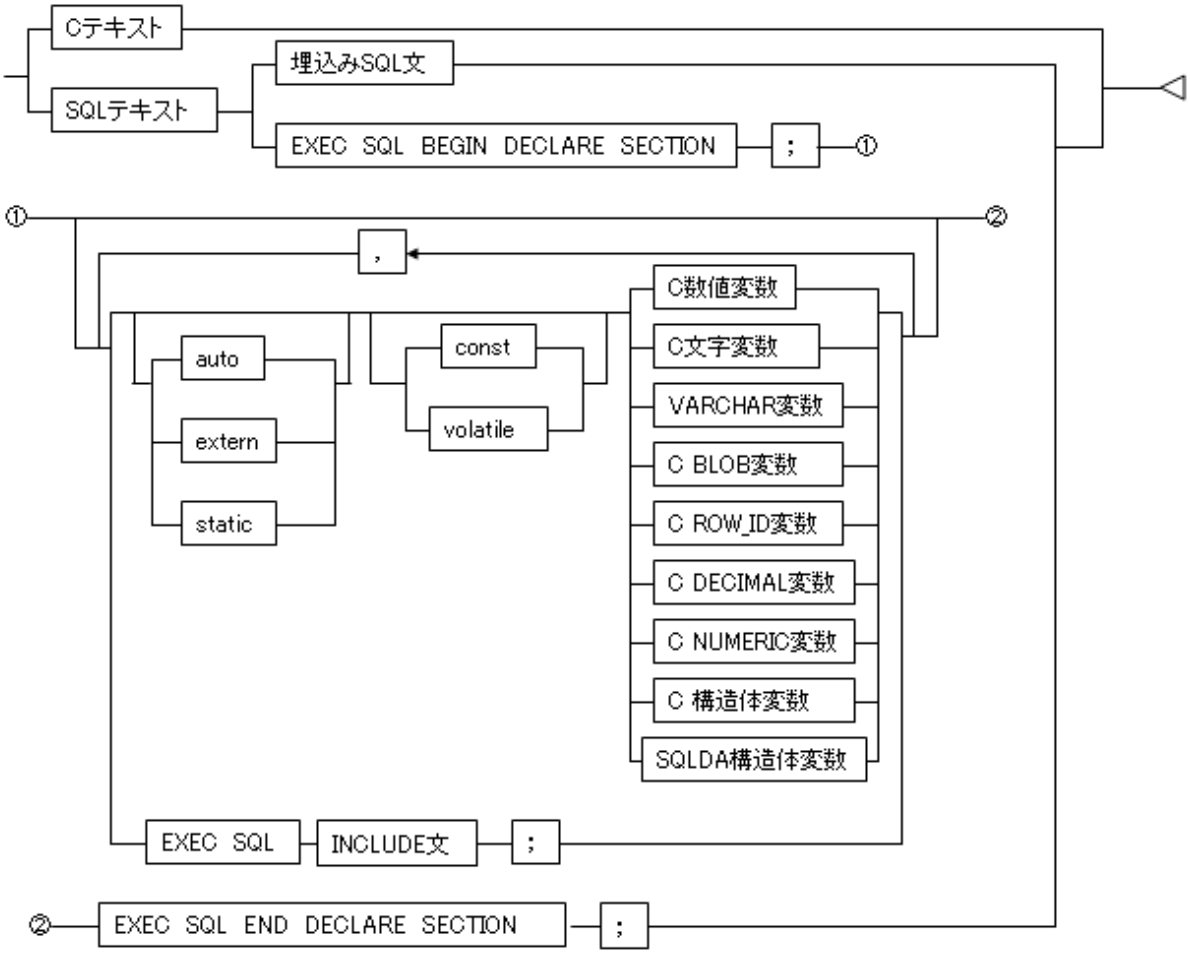
```
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL INCLUDE var1.h;
EXEC SQL END DECLARE SECTION;
```

6.3 SQL埋込みCプログラム

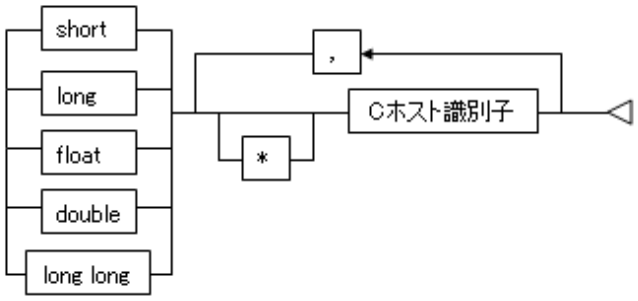
機能

ホストプログラムとしてC言語を使用するために必要な各種の定義を行います。

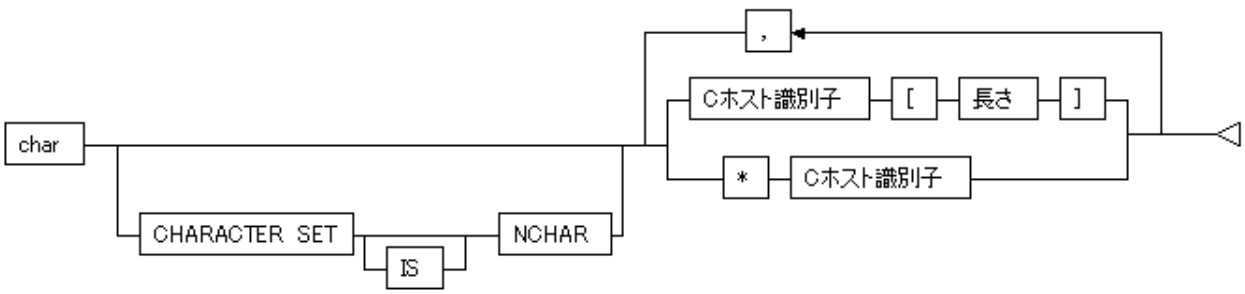
記述形式



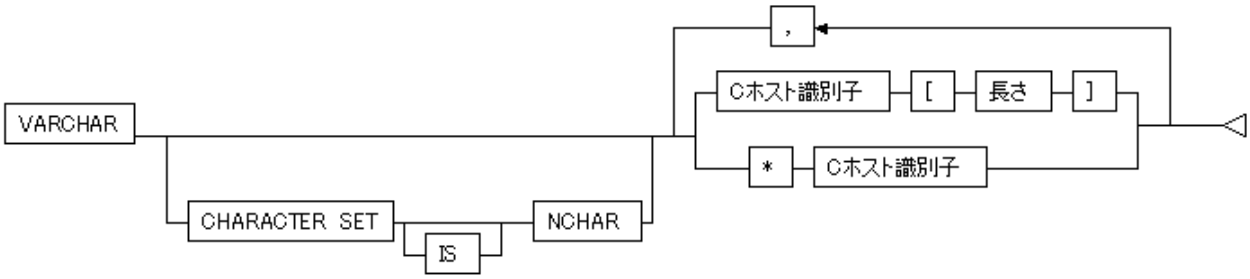
C数値変数



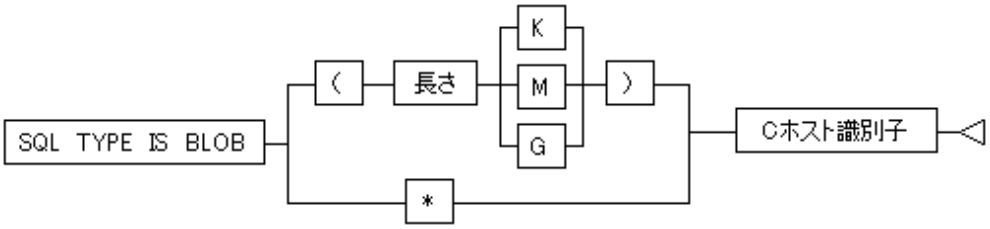
C文字変数



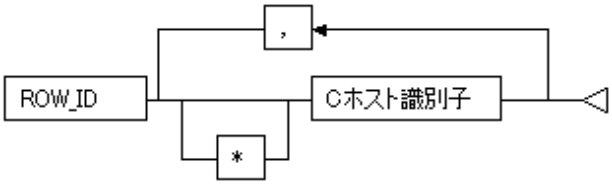
VARCHAR変数



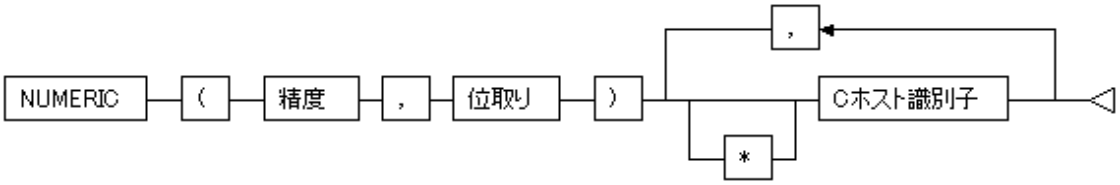
C BLOB変数



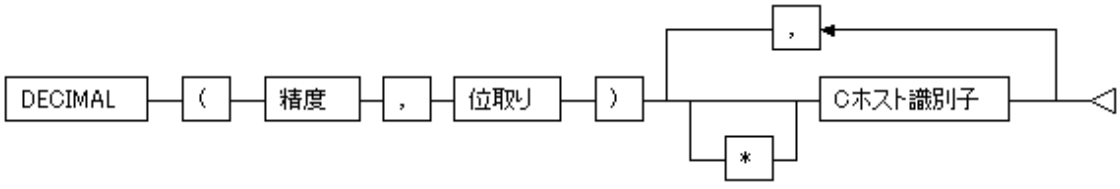
C ROW_ID変数



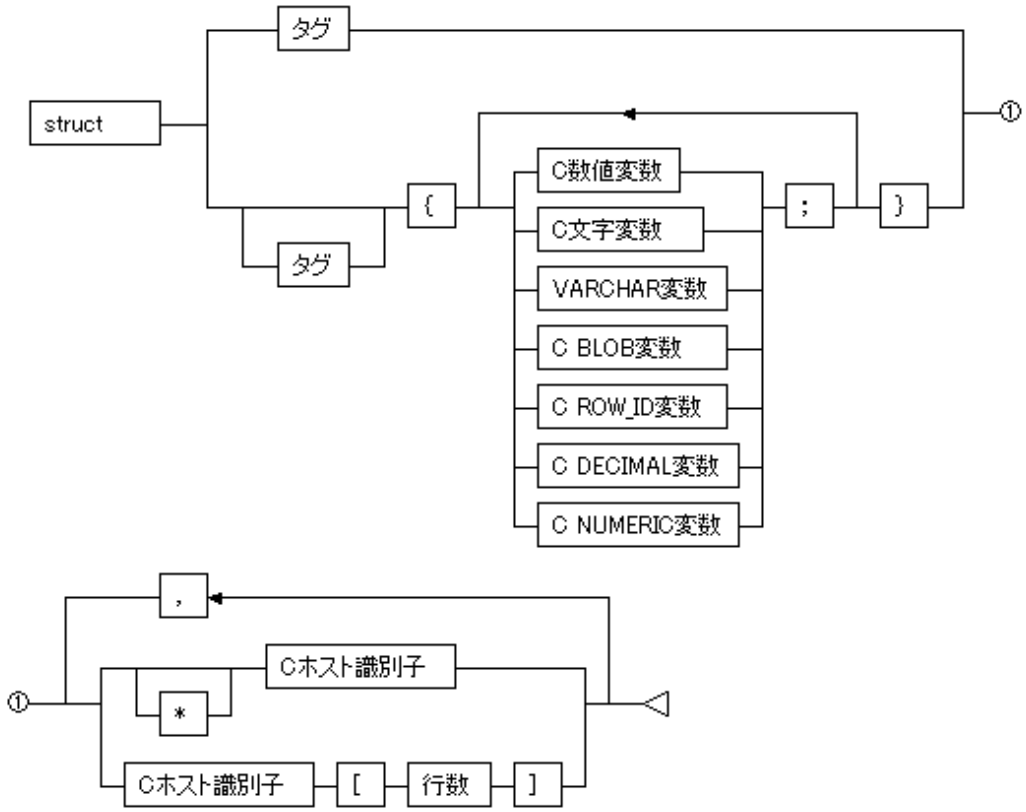
C NUMERIC変数



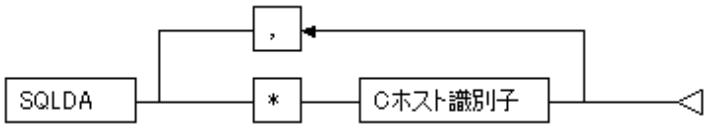
C DECIMAL変数



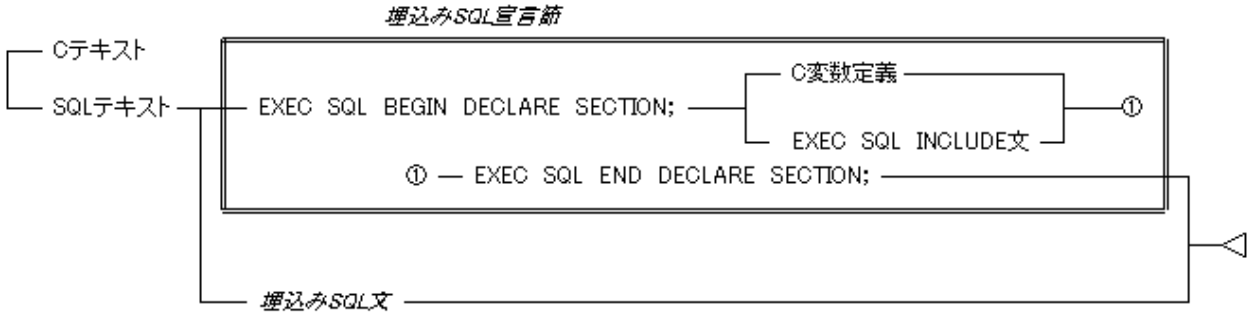
C 構造体変数



SQLDA構造体変数



構文の構成



参照項番

- 埋込みSQL文 → “6.1 Embedded SQL文(埋込みSQL文)”
- INCLUDE文 → “6.2 INCLUDE文”

一般規則

- SQL埋込みCプログラムの関数名は、“SQL”以外で始まる名前とします。これらは、英大文字を等価な英小文字に置き換えた場合にも同じです。

・ 埋込みSQL文は、関数内でC言語の実行文が記述可能な箇所に記述します。

・ C文字変数およびVARCHAR変数の長さは、2以上を指定することが必要です。

・ C文字変数およびVARCHAR変数の長さは、“データ長+1”と記述することもできます。

・ C文字型のポインタ宣言が参照する領域の長さは、NULL文字までの長さになります。

・ VARCHAR型のポインタ変数が参照する領域の長さは、コンパイル時に展開された構造体のメンバ“sqllen”に設定してください。

・ BLOB型のポインタ変数が参照する領域の長さは、コンパイル時に展開された構造体のメンバ“変数名_length” に設定してください。

・ C数値型のポインタ変数が参照する領域の長さは、ポインタ変数の型の長さになります。

・ C構造体変数を定義する場合に、メンバとして構造体を指定することはできません。

埋込みSQL宣言節

ー 埋込みSQL宣言節は、C言語の変数宣言が記述可能な箇所に記述します。

ー 埋込みSQL宣言節で宣言される変数名は、コンパイル単位内で一意であることが必要です。

CテキストおよびSQLテキスト

ー Cテキストは、C言語の文法規則に準拠します。

ー SQL埋込みCプログラムは、CテキストおよびSQLテキストからなるコンパイル単位とします。Cテキストは、C言語の文法規則に準拠します。SQLテキストは、1つ以上の省略可能な埋込みSQL宣言節と、1つ以上の埋込みSQL文から構成されます。

Cホスト識別子

ー Cホスト識別子は、C言語で規定される変数名です。ただし、変数名が日本語文字でない場合以下を満たすことが必要です。

- “SQL”以外で始まる変数名とします。これらは、英大文字を等価な英小文字に置き換えた場合にも同じです。

C変数定義

ー SQLのデータ型と対応するC変数定義を以下に示します。

表6.1 SQLのデータ型と対応するC変数定義

SQLデータ型		C変数定義
文字列型	CHARACTER(n)	char 変数名［n+1］ char *変数名(注1)
	CHARACTER VARYING(n)	VARCHAR 変数名［n+1］(注2) VARCHAR *変数名(注1)(注3)
各国語文字列型	NATIONAL CHARACTER(n)	char CHARACTER SET［IS］NCHAR 変数名［n*2+1］ char CHARACTER SET [IS] NCHAR *変数名(注1)
	NATIONAL CHARACTER VARYING(n)	VARCHAR CHARACTER SET［IS］NCHAR 変数名［n*2+1］ (注1) VARCHAR CHARACTER SET［IS］NCHAR *変数名 (注1)(注3)
真数型	NUMERIC(p,q)	NUMERIC(p,q) 変数名(注4) NUMERIC(p,q) *変数名(注1)
	DECIMAL(p,q)	DECIMAL(p,q) 変数名(注5) DECIMAL(p,q) *変数名(注1)

SQLデータ型		C変数定義
	SMALLINT	short 変数名 short *変数名(注1)
	INTEGER	long 変数名 (注6) long *変数名(注1)
概数型	REAL	float 変数名 float *変数名(注1)
	DOUBLE PRECISION	double 変数名 double *変数名(注1)
日時型	DATE	char 変数名[11]
	TIME	char 変数名[9]
	TIMESTAMP	char 変数名[20]
時間隔型	INTERVAL	“表6.2 SQLの時間隔型と対応するC変数定義”参照
BLOB型	BINARY LARGE OBJECT (n単位)	SQL TYPE IS BLOB(n単位) 変数名(注7) SQL TYPE IS BLOB *変数名(注1)(注8)
ROW_ID	ROW_ID	ROW_ID 変数名 ROW_ID *変数名(注1)

n: 長さ、p: 精度、q: 位取り、単位: K(キロバイト)、M(メガバイト)、G(ギガバイト)

注1) 各型のポインタ変数を宣言する場合は、変数名の前に*を指定します。

注2) 可変長文字列の場合、コンパイル時に以下の構造体に展開されます。

```
struct {
  short sqllen;
  char sqlvar[n+1];
} 変数名
```

注3) 可変長文字列のポインタ変数の場合、コンパイル時に以下の構造体に展開されます。

```
struct 変数名_SQLVAR{
  short sqllen;
  char sqlvar[1];
} *変数名
```

注4) NUMERIC型の場合、コンパイル時に以下のように展開されます。

```
char 変数名 [p+1];
```

注5) DECIMAL型の場合、コンパイル時に以下のように展開されます。

```
char 変数名 [(p÷2)+1];
```

注6) 64ビットのSQL埋込みCプログラムの場合、対応するC変数定義はintです。

注7) BLOB型の場合、コンパイル時に以下の構造体に展開されます。

```
struct {
  long  変数名_reserved; (注9)
  unsigned long 変数名_length; (注9)
  char 変数名_data[n*1024];
} 変数名
```

注8) BLOB型のポインタ変数の場合、コンパイル時に以下の構造体に展開されます。

```
struct 変数名_SQLBLOB{
  long  変数名_reserved;  (注9)
```

```
  unsigned long 変数名_length;  (注9)
  char 変数名_data[1];
} *変数名
```

注9) 64ビットのSQL埋込みCプログラムの場合、longはintに展開されます。

－ SQLの時間隔型と対応するC変数定義を以下に示します。

表6.2 SQLの時間隔型と対応するC変数定義

時間隔型		C変数定義
年月型	INTERVAL YEAR[(n)] TO MONTH	char 変数名［n+5］
	INTERVAL YEAR[(n)]	char 変数名［n+2］ long 変数名 (注10)(注11)
	INTERVAL MONTH[(n)]	char 変数名［n+2］ long 変数名 (注10)(注11)
日時型	INTERVAL DAY[(n)] TO HOUR	char 変数名［n+5］
	INTERVAL DAY[(n)] TO MINUTE	char 変数名［n+8］
	INTERVAL DAY[(n)] TO SECOND	char 変数名［n+11］
	INTERVAL DAY[(n)]	char 変数名［n+2］ long 変数名 (注10)(注11)
	INTERVAL HOUR[(n)] TO MINUTE	char 変数名［n+5］
	INTERVAL HOUR[(n)] TO SECOND	char 変数名［n+8］
	INTERVAL HOUR[(n)]	char 変数名［n+2］ long 変数名 (注10)(注11)
	INTERVAL MINUTE[(n)] TO SECOND	char 変数名［n+5］
	INTERVAL MINUTE[(n)]	char 変数名［n+2］ long 変数名 (注10)(注11)
	INTERVAL SECOND[(n)]	char 変数名［n+2］ long 変数名 (注10)(注11)

n: 桁数（1～9）

注10) 位取り0の真数型が指定可能なので、以下のデータ型も指定できます。

```
short  変数名   (n< 5の場合)
```

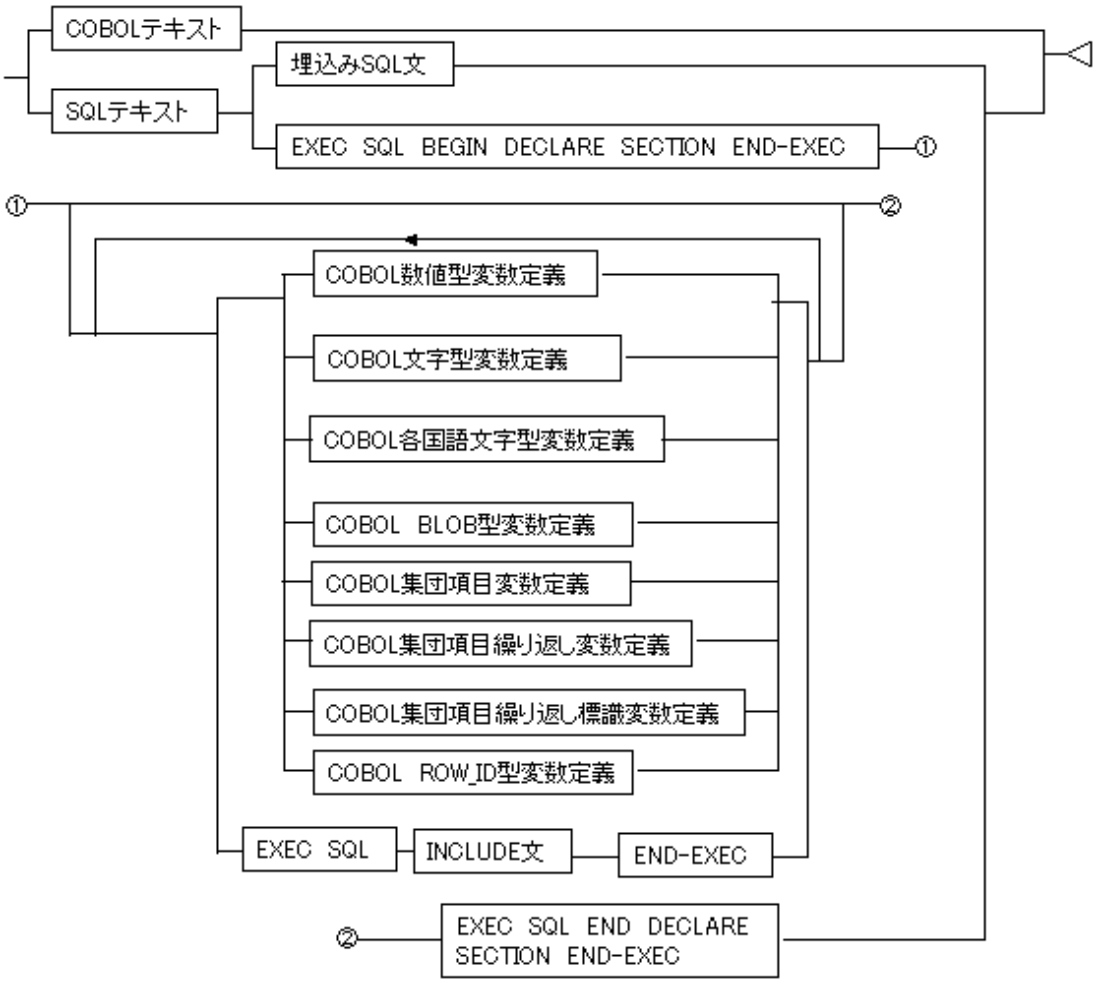
注11) 64ビットのSQL埋込みCプログラムの場合、対応するC変数定義はintです。

6.4 SQL埋込みCOBOLプログラム

機能

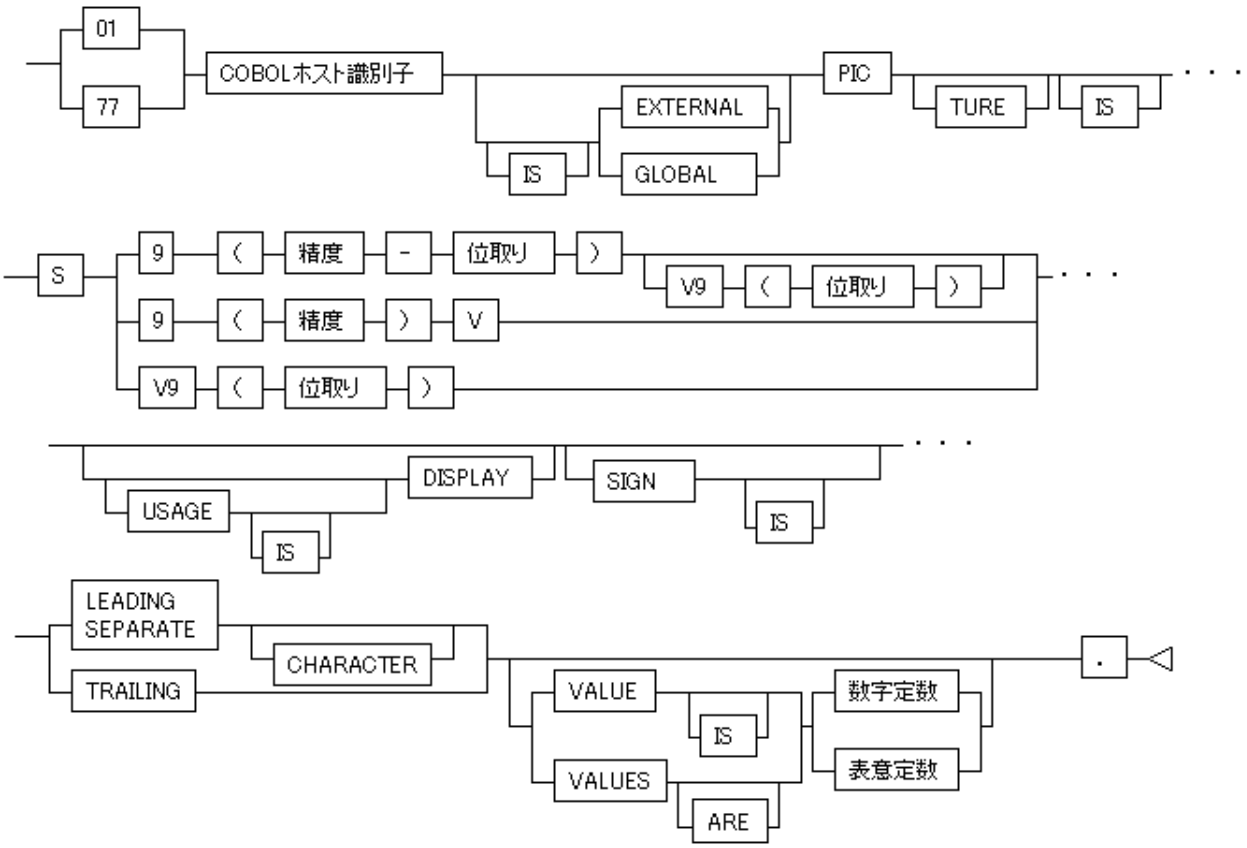
ホストプログラムとしてCOBOLを使用するために必要な各種の定義を行います。

記述形式

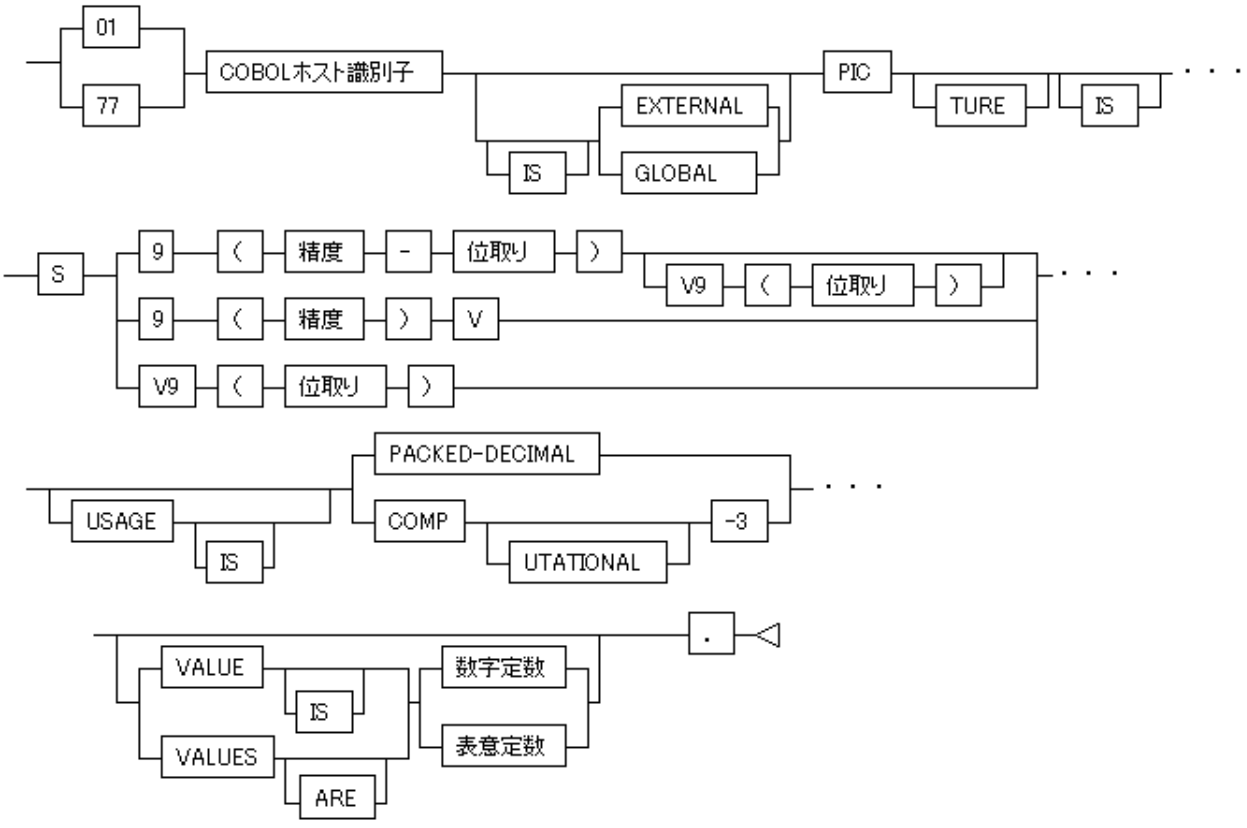


COBOL数値型変数定義

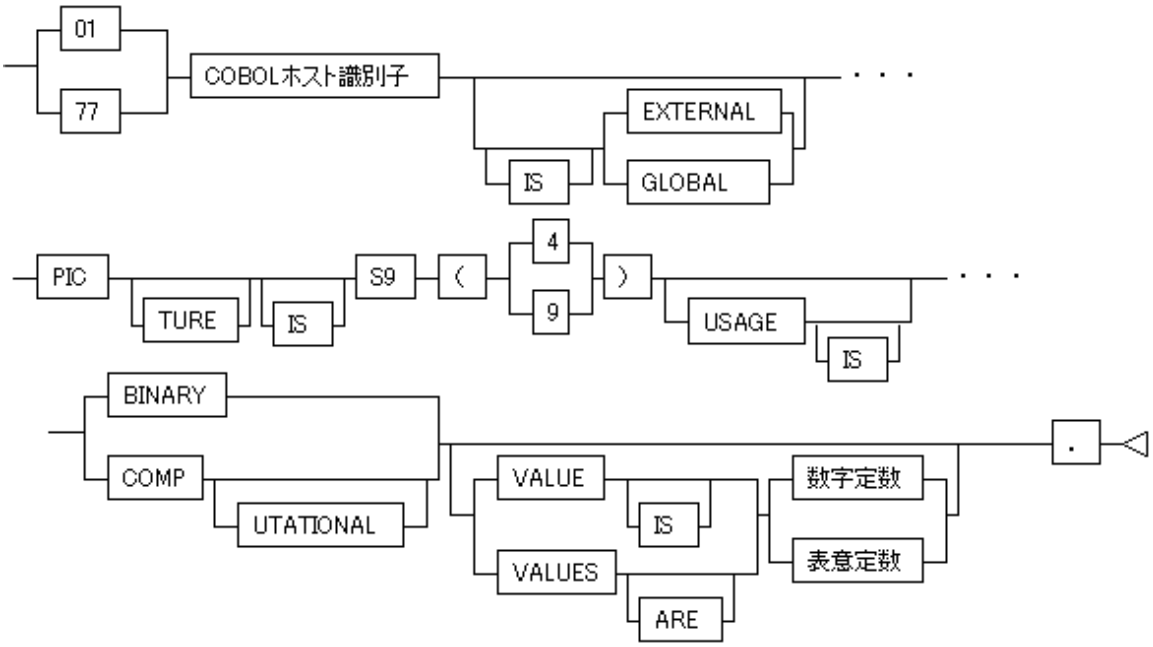
COBOL外部10進型変数定義



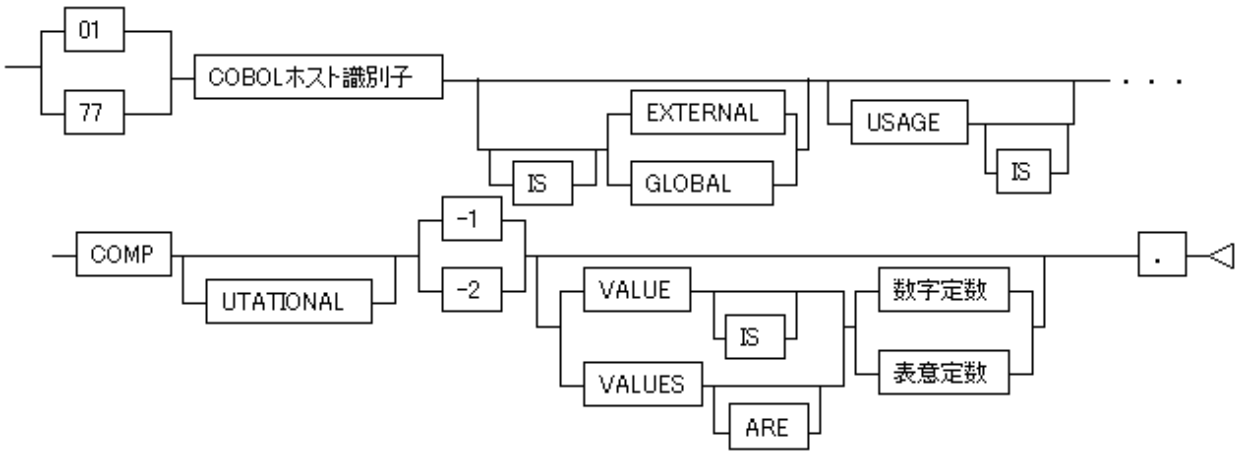
COBOL内部10進型変数定義



COBOL2進型変数定義

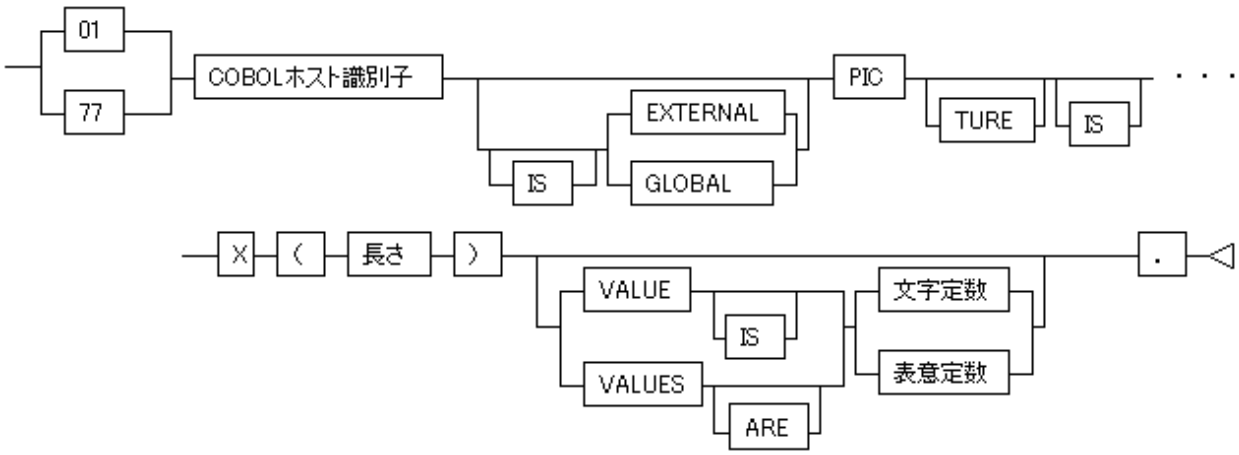


COBOL内部浮動小数点型変数定義

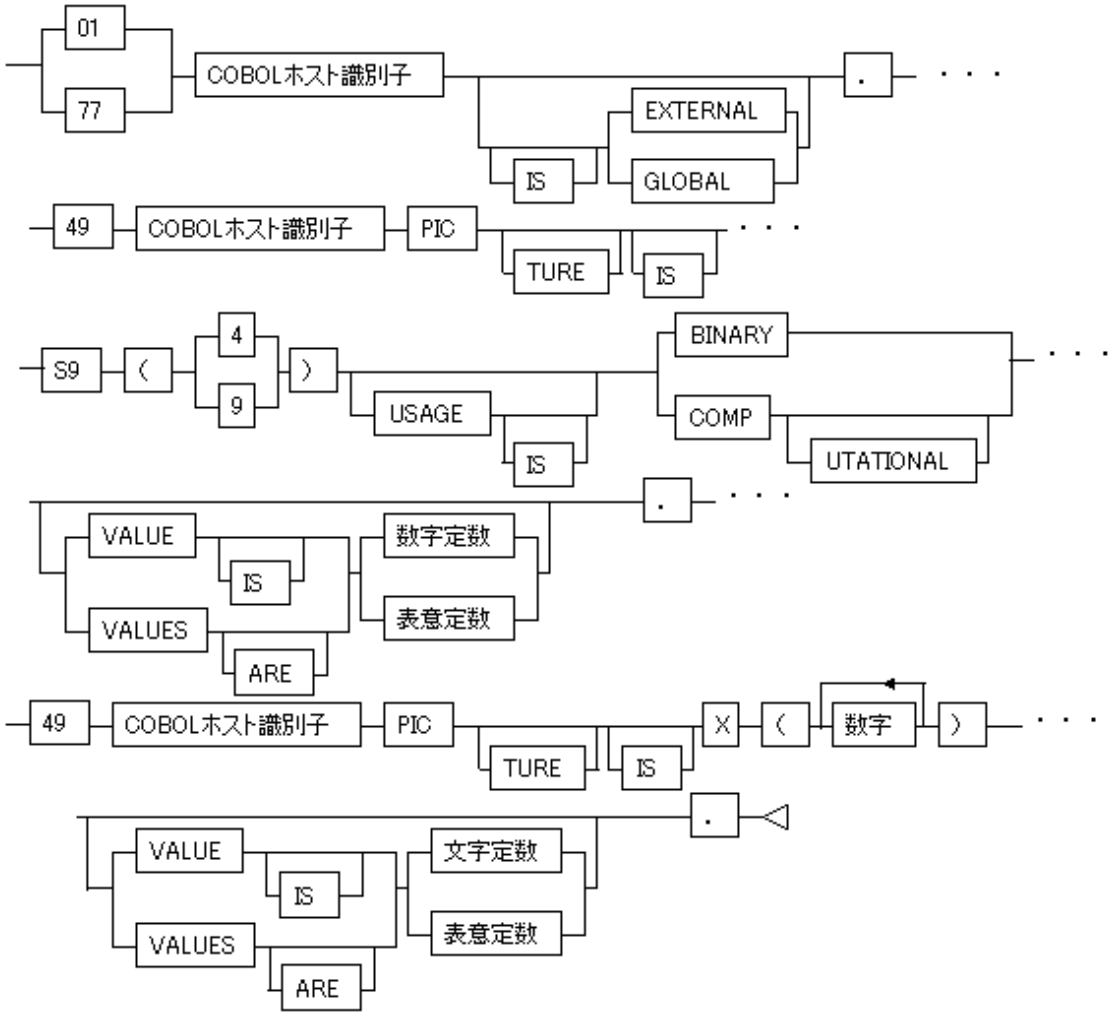


COBOL文字型変数定義

COBOL固定長文字型変数定義

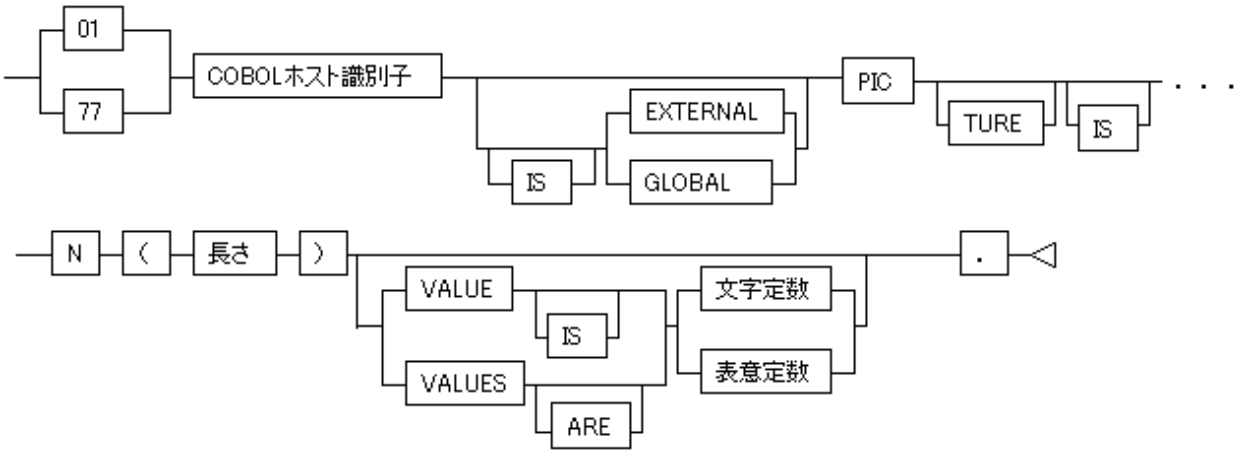


COBOL可変長文字型変数定義

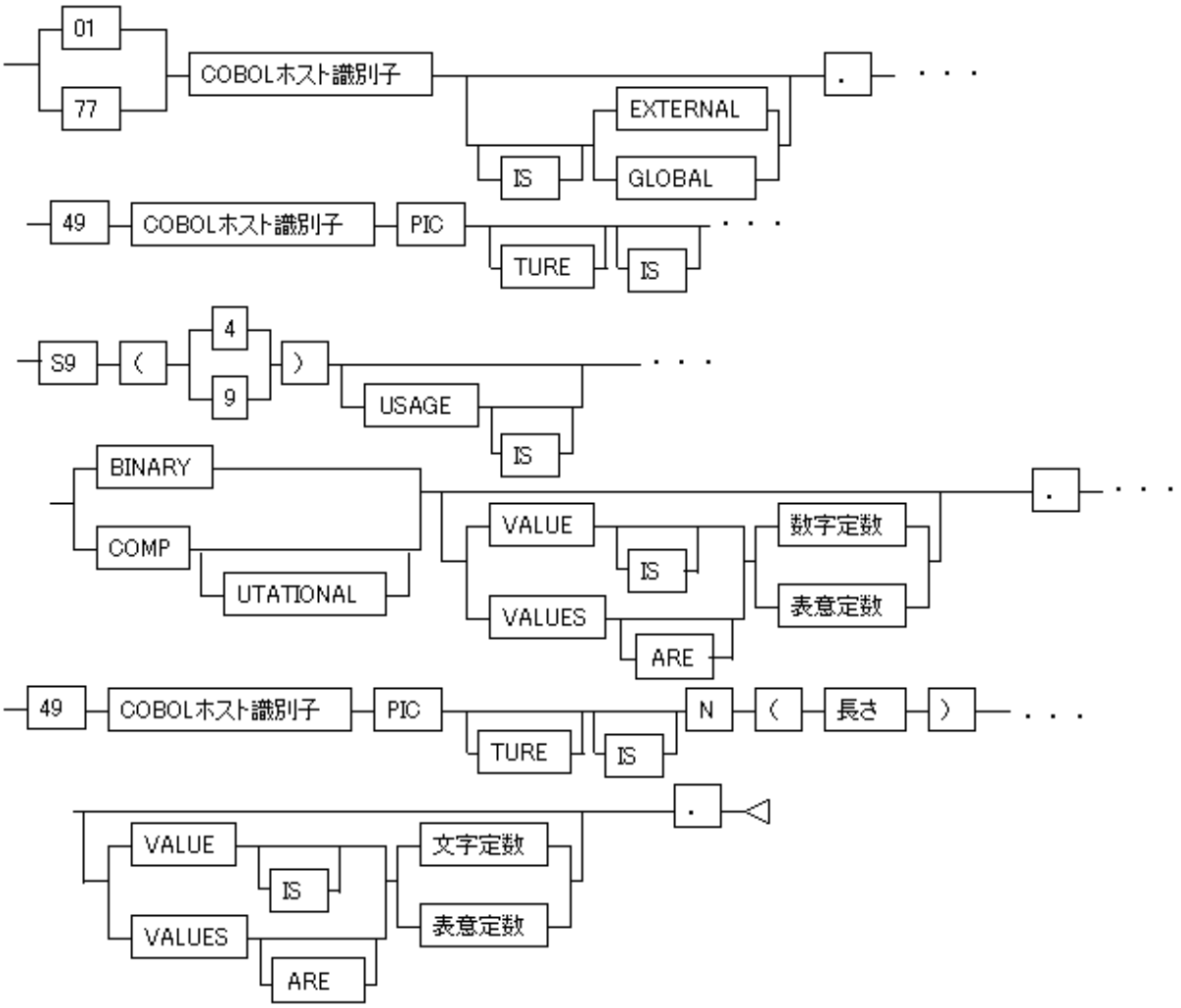


COBOL各国語文字型変数定義

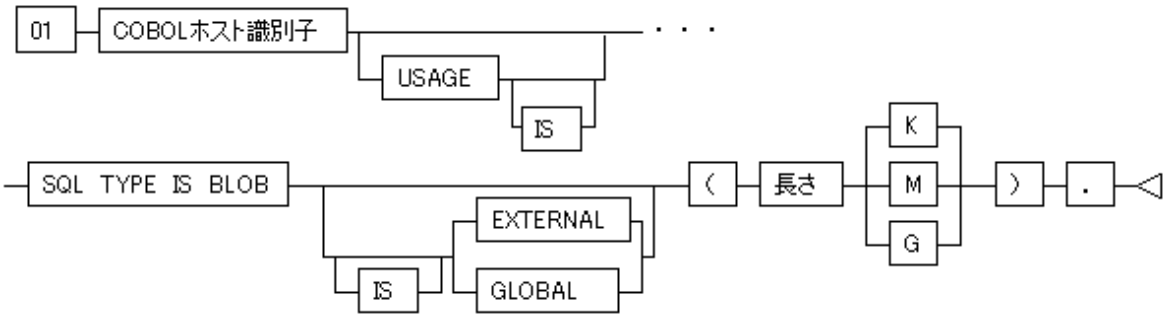
COBOL固定長各国語文字型変数定義



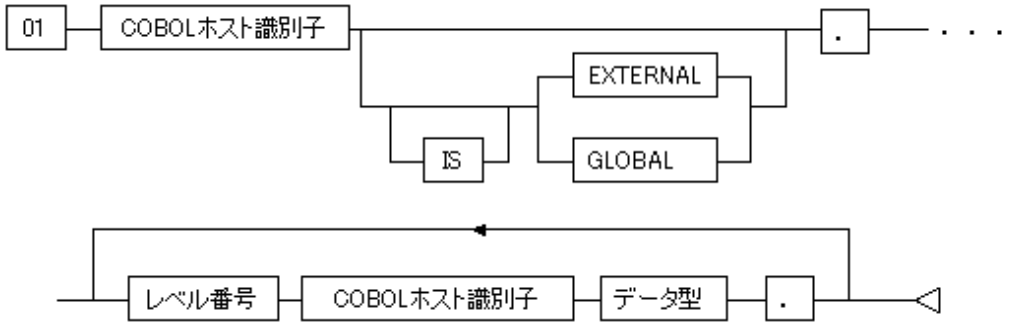
COBOL可変長各国語文字型変数定義



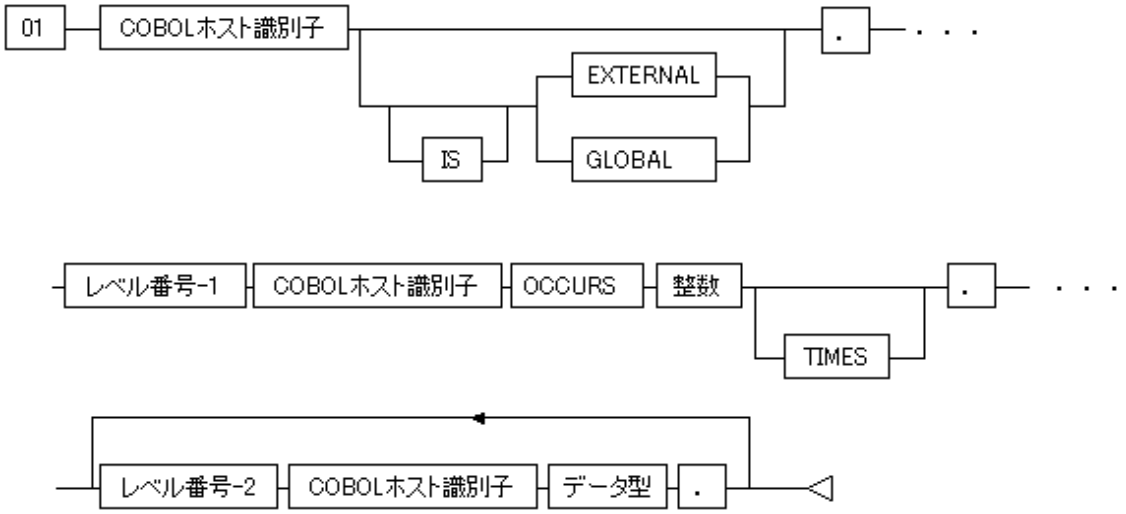
COBOL BLOB型変数定義



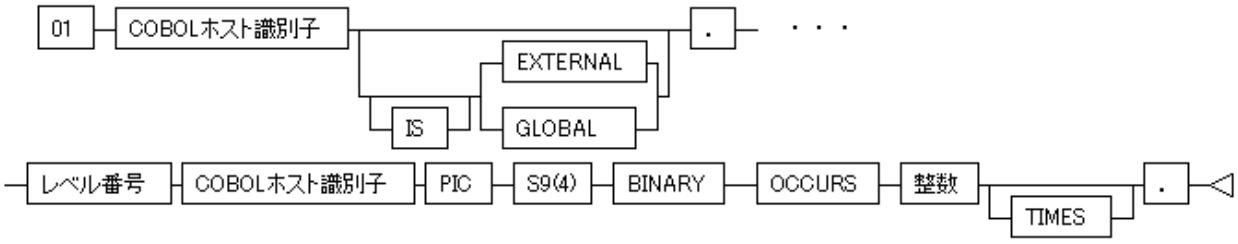
COBOL集団項目変数定義



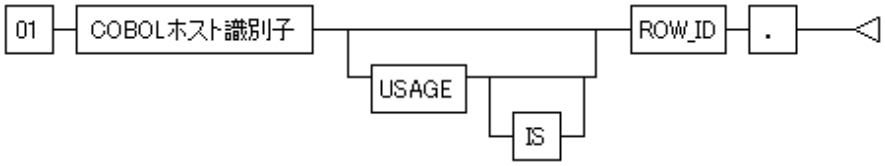
COBOL集団項目繰り返し変数定義



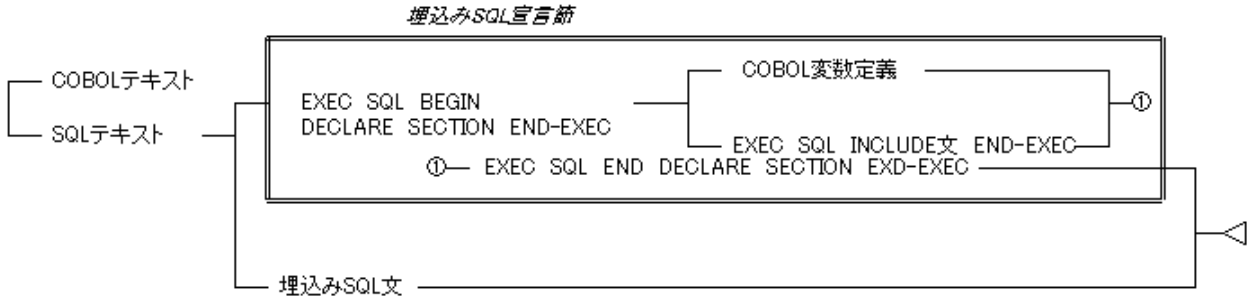
COBOL集団項目繰り返し標識変数定義



COBOL ROW_ID型変数定義



構文の構成



参照項番

- 埋込みSQL文 → “6.1 Embedded SQL文(埋込みSQL文)”
- INCLUDE文 → “6.2 INCLUDE文”
- データ型 → “2.2 データ型”

一般規則

- 埋込みSQL文は、手続き部に記述します。

埋込みSQL宣言節

— 埋込みSQL宣言節は、データ部の作業場所節または連絡節に記述します。

COBOLテキストおよびSQLテキスト

— COBOLテキストは、COBOL97、PowerCOBOL97またはNetCOBOLの文法規則に準拠します。

— SQL埋込みCOBOLプログラムは、COBOLテキストおよびSQLテキストからなるコンパイル単位とします。COBOLテキストは、COBOL97、PowerCOBOL97またはNetCOBOLの文法規則に準拠します。SQLテキストは、1つ以上の省略可能な埋込みSQL宣言節と、1つ以上の埋込みSQL文から構成されます。

COBOLホスト識別子

— COBOLホスト識別子は、COBOL97、PowerCOBOL97またはNetCOBOLで規定される変数名です。ただし、変数名が日本語文字でない場合以下を満たすことが必要です。

- “SQL”以外で始まる変数名とします。これらは、英大文字を等価な英小文字に置き換えた場合にも同じです。

— COBOLホスト識別子に集団項目ホスト変数を指定する場合の一般規則を以下に示します。

- 集団項目に属する基本項目はそれぞれ、COBOL変数定義で規定されている記述しかできません。
- 集団項目に属する基本項目のレベル番号は、02から48のいずれかであることが必要です。なお、可変長文字列の従属項目はレベル49です。
- 集団項目に属する基本項目のレベル番号は、すべて同一であることが必要です。
- コンパイル単位で集団項目名は一意であることが必要です。
- 集団項目に属する基本項目名は、同一の集団項目中では一意であることが必要です。
- 集団項目ホスト変数や集団項目繰り返しホスト変数の宣言中に、別の集団項目ホスト変数や集団項目繰り返しホスト変数を入れ子で宣言することはできません。
- 集団項目ホスト変数のSQL文中での記述形式を以下に示します。

参照

指定例については、“アプリケーション開発ガイド(埋込みSQL編)”を参照してください。

記述形式

```
  :〈集団項目名〉
| :〈集団項目名〉.〈集団項目に属する基本項目名〉
```

注意

- 集団項目ホスト変数の並び順は、選択リストの並び順に対応します。
- 集団項目に属する基本項目名は、集団項目名の修飾なしでSQL文中に記述できません。
- 集団項目名のみの指定は、INTO句およびVALUES句以外には記述できません。

例

COBOL集団項目変数定義

```
01 G1.
   02 HOST1 PIC S9(4) BINARY.
   02 HOST2 PIC X(10).
   02 HOST3 PIC S9(4) BINARY.
   02 HOST4 PIC S9(4) BINARY.
```

ー COBOLホスト識別子に集団項目繰り返しホスト変数を指定する場合の一般規則を以下に示します。

- 集団項目に属する基本項目はそれぞれ、COBOL変数定義で規定されている記述しかできません。
- 集団項目に属する基本項目のレベル番号-1は、02から47のいずれかであることが必要です。
- 集団項目に属する基本項目のレベル番号-2は、03から48のいずれかであることが必要です。なお、可変長文字列の従属項目はレベル49です。
- 集団項目に属する基本項目のレベル番号-2は、レベル番号-1より大きくなければなりません。
- 集団項目に属する基本項目のレベル番号-2は、すべて同一であることが必要です。
- 集団項目に属する基本項目のレベル番号-1は、複数指定することはできません。
- コンパイル単位で集団項目名は一意であることが必要です。
- 集団項目に属する基本項目名は、同一の集団項目中では一意であることが必要です。
- 集団項目ホスト変数や集団項目繰り返しホスト変数の宣言中に、別の集団項目ホスト変数や集団項目繰り返しホスト変数を入れ子で宣言することはできません。
- 集団項目繰り返しホスト変数のSQL文中での記述形式を以下に示します。

記述形式

```
:〈集団項目名〉.〈集団項目に属する基本項目名〉
```

注意

- 集団項目繰り返しホスト変数の並び順は、選択リストの並び順に対応します。
- 集団項目に属する基本項目名は、レベル番号-1の基本項目名を指定します。
- 基本項目繰り返しホスト変数は、INSERT文(一括指定)のVALUES句以外には記述できません。INSERT文の詳細については“3.48 INSERT文”を参照してください。

例

COBOL集団項目繰り返し変数定義

```
01 G1.
   02 G2 OCCURS 10 TIMES.
```

```
    03 HOST1 PIC S9(4) BINARY.
    03 HOST2 PIC X(10).
    03 HOST3 PIC S9(4) BINARY.
    03 HOST4 PIC S9(4) BINARY.
```

ー COBOLホスト識別子に集団項目繰り返し標識変数を指定する場合の一般規則を以下に示します。

- 集団項目繰り返し標識変数は、標識変数を集団項目とする変数です。
- 集団項目繰り返し標識変数は集団項目ホスト変数に対応した標識変数定義を繰り返し記述することができます。
- 集団項目に属する基本項目のレベル番号は、02から48のいずれかであることが必要です。
- 集団項目繰り返し標識変数に指定する整数の範囲は、1～2147483647であることが必要です。
- 集団項目繰り返し標識変数のSQL文中での記述形式を以下に示します。

記述形式

```
 :<集団項目名>
|INDICATOR:<集団項目名>
```

注意

- 集団項目繰り返し標識変数の繰り返し順は、選択リストの並び順および集団項目ホスト変数の並び順に対応します。
- 集団項目に属する基本項目名は、SQL文中に記述できません。
- 集団項目繰り返し標識変数の繰り返し数が、集団項目ホスト変数に属する基本項目の数より少ない場合、不足分は標識変数が指定されなかったものと解釈します。また、集団項目繰り返し標識変数の繰り返し数が、集団項目ホスト変数に属する基本項目の数より多い場合、余りの標識変数は無視されます。

COBOL変数定義

ー SQLのデータ型と対応するCOBOL変数定義を以下に示します。

表6.3 SQLのデータ型と対応するCOBOL変数定義

SQLデータ型		COBOL変数定義
文字列型	CHARACTER(n)	01 変数名 PIC X(n).
	CHARACTER VARYING(n)	01 変数名-1. 49 変数名-2 PIC S9({4|9}) {BINARY|COMP}. 49 変数名-3 PIC X(n).
各国語文字列型	NATIONAL CHARACTER(n)	01 変数名 PIC N(n).
	NATIONAL CHARACTER VARYING(n)	01 変数名-1. 49 変数名-2 PIC S9({4|9}) {BINARY|COMP}. 49 変数名-3 PIC N(n).
真数型	NUMERIC(p,q)	01 変数名 PIC S9(p-q)V9(q) DISPLAY [SIGN] {LEADING SEPARATE [CHARACTER]|TRAILING}.
	DECIMAL(p,q)	01 変数名 PIC S9(p-q)V9(q) {PACKED-DECIMAL|COMP-3}.
	SMALLINT	01 変数名 PIC S9(4) {BINARY|COMP}.
	INTEGER	01 変数名 PIC S9(9) {BINARY|COMP}.
概数型	REAL	01 変数名 COMP-1.

SQLデータ型		COBOL変数定義
	DOUBLE PRECISION	01 変数名 COMP-2.
日時型	DATE	01 変数名 PIC X(10).
	TIME	01 変数名 PIC X(8).
	TIMESTAMP	01 変数名 PIC X(19).
時間隔型	INTERVAL	“表:SQLの時間隔型と対応するCOBOL変数定義”参照
BLOB型	BINARY LARGE OBJECT(n単位)	01 変数名 SQL TYPE IS BLOB(n単位). (注)
ROW_ID	ROW_ID	01 変数名 ROW_ID.

n: 長さ、p: 精度、q: 位取り、単位: K(キロバイト)、M(メガバイト)、G(ギガバイト)

注) BLOB型の場合、コンパイル時に以下の構造体に展開されます(単位がKの場合)

```
01 変数名.
    49 変数名-RESERVED PIC S9(9) USAGE IS BINARY.
    49 変数名-LENGTH   PIC S9(9) USAGE IS BINARY.
    49 変数名-DATA     PIC X(n*1024).
```

ー SQLの時間隔型と対応するCOBOL変数定義を以下に示します。

表6.4 SQLの時間隔型と対応するCOBOL変数定義

時間隔型		COBOL変数定義
年月型	INTERVAL YEAR[(n)] TO MONTH	01 変数名 PIC X(n+4).
	INTERVAL YEAR[(n)]	01 変数名 PIC X(n+1). 01 変数名 PIC S9(9) {BINARY|COMP}. (注)
	INTERVAL MONTH[(n)]	01 変数名 PIC X(n+1). 01 変数名 PIC S9(9) {BINARY|COMP}. (注)
日時型	INTERVAL DAY[(n)] TO HOUR	01 変数名 PIC X(n+4).
	INTERVAL DAY[(n)] TO MINUTE	01 変数名 PIC X(n+7).
	INTERVAL DAY[(n)] TO SECOND	01 変数名 PIC X(n+10).
	INTERVAL DAY[(n)]	01 変数名 PIC X(n+1). 01 変数名 PIC S9(9) {BINARY|COMP}. (注)
	INTERVAL HOUR[(n)] TO MINUTE	01 変数名 PIC X(n+4).
	INTERVAL HOUR[(n)] TO SECOND	01 変数名 PIC X(n+7).
	INTERVAL HOUR[(n)]	01 変数名 PIC X(n+1). 01 変数名 PIC S9(9) {BINARY|COMP}. (注)
	INTERVAL MINUTE[(n)] TO SECOND	01 変数名 PIC X(n+4).

時間隔型		COBOL変数定義
	INTERVAL MINUTE[(n)]	01 変数名 PIC X(n+1). 01 変数名 PIC S9(9) {BINARY|COMP}. (注)
	INTERVAL SECOND[(n)]	01 変数名 PIC X(n+1). 01 変数名 PIC S9(9) {BINARY|COMP}. (注)

n: 桁数(1～9)

注) 位取り0の真数型が指定可能なので、以下のデータ型も指定できます。

```
01 変数名  PIC S9(n) DISPLAY [SIGN] {LEADING SEPARATE [CHARACTER]|TRAILING}
01 変数名  PIC S9(n) {PACKED-DECIMAL|COMP-3}
01 変数名  PIC S9(4) {BINARY|COMP}.  (n<5の場合)
```

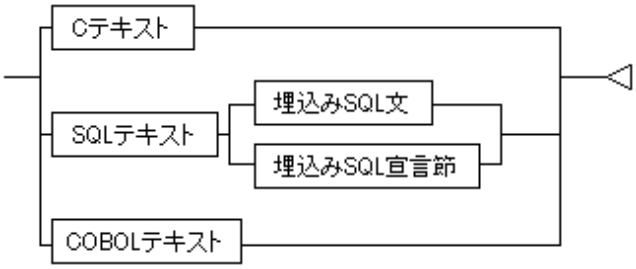
6.5 SQL埋込みホストプログラム

機能

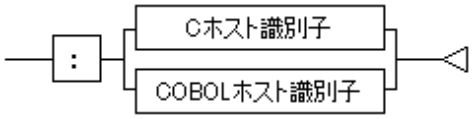
SQL埋込みホストプログラムは、特定の標準プログラム言語の機能を拡張したアプリケーションで、プログラミング言語テキストとSQLテキストからなっています。プログラミング言語テキストは、特定の標準プログラム言語です。SQLテキストは、埋込みSQL文と埋込みSQL宣言節からなっています。

記述形式

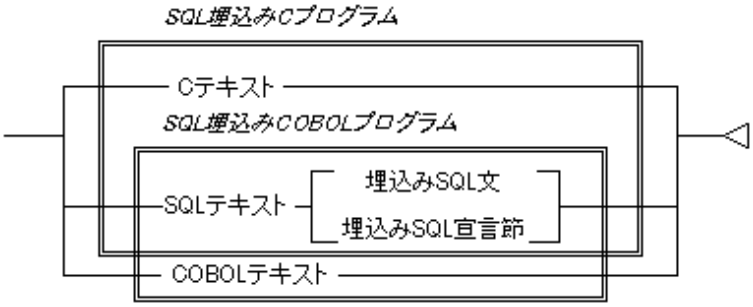
SQL埋込みホストプログラム



埋込み変数名



構文の構成



参照項番

- SQL埋込みCプログラム → “6.3 SQL埋込みCプログラム”
- SQL埋込みCOBOLプログラム → “6.4 SQL埋込みCOBOLプログラム”
- Cテキスト → “6.3 SQL埋込みCプログラム”
- SQLテキスト → “6.3 SQL埋込みCプログラム”
- COBOLテキスト → “6.4 SQL埋込みCOBOLプログラム”
- 埋込みSQL文 → “6.1 Embedded SQL文(埋込みSQL文)”
- 埋込みSQL宣言節 → “6.3 SQL埋込みCプログラム”
- Cホスト識別子 → “6.3 SQL埋込みCプログラム”
- COBOLホスト識別子 → “6.4 SQL埋込みCOBOLプログラム”

一般規則

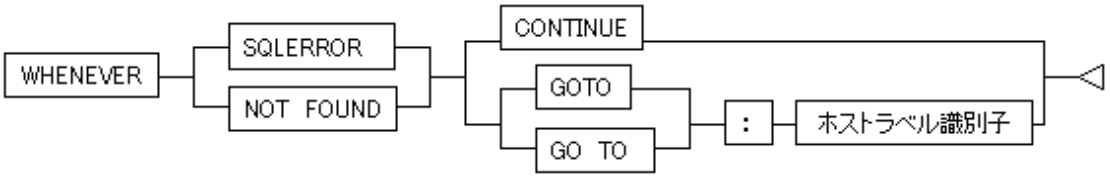
- 埋込みSQL文は、SQL埋込みホストプログラムに少なくとも1つ以上記述されていることが必要です。
- SQL先頭子の“EXEC”と“SQL”は同一行に記述することが必要です。
- SQL埋込みCプログラムの中に記述される埋込みSQL文および埋込みSQL宣言節には、SQL終了子に“;(セミコロン)”を指定することが必要です。
- SQL埋込みCOBOLプログラムの中に記述される埋込みSQL文および埋込みSQL宣言節には、SQL終了子に“END-EXEC”を指定することが必要です。

6.6 WHENEVER文(埋込み例外宣言)

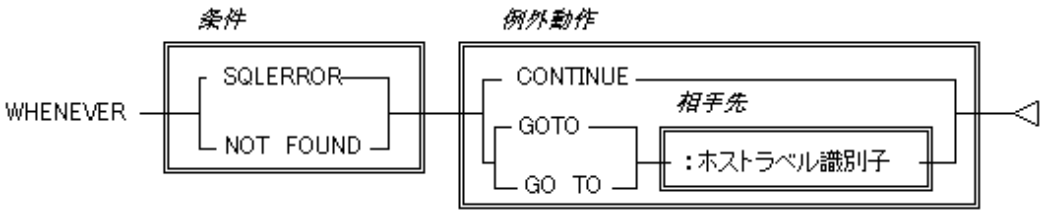
機能

埋込みSQLの実行時、例外条件が発生した場合にSQL埋込みホストプログラムがとる動作を指定します。

記述形式



構文の構成



一般規則

- SQL埋込みホストプログラムでWHENEVER文が指定されていない場合、またはWHENEVER文が出現する以前に例外条件が発生した場合、CONTINUEが指定されたとみなされます。

・ WHENEVER文は、SQL埋込みホストプログラム中に複数回宣言することができます。宣言されたWHENEVER文は、宣言された時点以降のテキストの並びに、同じ例外条件を指定したWHENEVER文が出現するまで有効です。後続のWHENEVER文が出現すると以前の宣言は無効となります。

  埋込み例外宣言の有効範囲をSQL埋込みCOBOLプログラムを例に“図6.1 埋込み例外宣言の有効範囲”に示します。SQL埋込みCプログラムについても同様です。

条件(SQLERRORおよびNOT FOUND)

ー 条件にSQLERRORが指定されると、埋込みSQL文の実行結果が正常終了、データなし、または警告のいずれでもないときに、例外動作が実行されます。

ー 条件にNOT FOUNDが指定されると、埋込みSQL文の実行結果がデータなしのときに、例外動作が実行されます。

例外動作(CONTINUEおよびGOTO句)

ー 例外動作にCONTINUEが指定されると、条件に指定された事象の発生した埋込みSQL文の次の文が実行されます。

ー 例外動作にGOTOまたはGO TOが指定されると、相手先で指定された相手先へ分岐します。

ホストラベル識別子

ー ホストラベル識別子は、以下のように指定します。

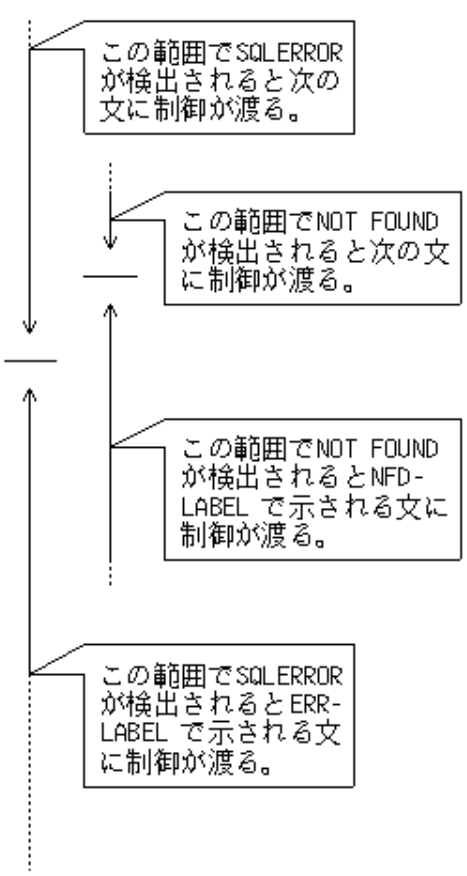
- SQL埋込みCプログラムの中に指定する場合は、分岐先の文に付けられた名札(ラベル)を指定します。
- SQL埋込みCOBOLプログラムの中に指定する場合は、分岐先の節名(セクション名)または段落名(パラグラフ名)を指定します。なお、節名および段落名が日本語文字でない場合、先頭は英字とします。段落名には、節名による修飾を付けてはいけません。

ー ホストラベル識別子は、“SQL”以外で始まることが必要です。これらは、英大文字を等価な英小文字に置き換えた場合にも同じです。

使用例

図6.1 埋込み例外宣言の有効範囲

```
    EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.

    EXEC SQL DECLARE D CURSOR FOR ...   END-EXEC.

    EXEC SQL OPEN D  END-EXEC.

    EXEC SQL FETCH D INTO ...  END-EXEC.

    EXEC SQL WHENEVER NOT FOUND
                        GO TO :NFD-LABEL END-EXEC.
    EXEC SQL DELETE FROM ... END-EXEC.

    EXEC SQL WHENEVER SQLERROR
                        GO TO :ERR-LABEL END-EXEC.

    EXEC SQL FETCH D INTO ...  END-EXEC.

    EXEC SQL DELETE FROM ... END-EXEC.

RTN-LABEL.
    EXEC SQL CLOSE D END-EXEC.
        •
        •


NFD-LABEL.
    DISPLAY "NOT FOUND".
    GO TO RTN-LABEL.

ERR-LABEL.
    DISPLAY "ERROR OCCURRED".
```



第7章 SQL拡張インタフェース

SQL拡張インタフェースは、以下の機能で使用するC言語の関数群です。

7.1 セションの操作

Symfoware/RDBでは、SQL拡張インタフェースを使用することにより、複数のセションを作成することができます。

これにより、1つのアプリケーションプロセスにおいて、セションごとに異なった環境・トランザクションで並行してデータ操作を行うことができます。

セションの操作には、以下の関数を使用します。

SQLThrAllocID:

セションの作成

SQLThrStartID:

セションの開始

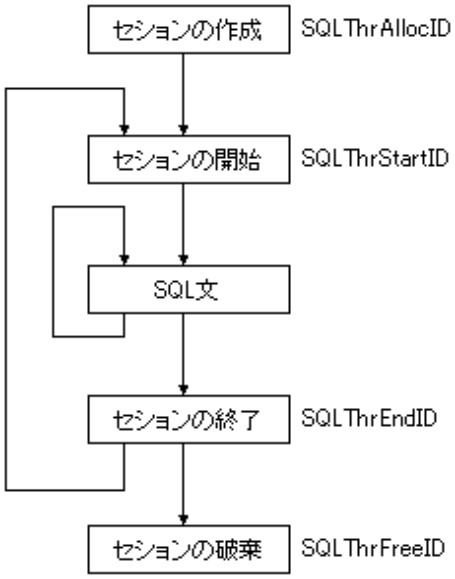
SQLThrEndID:

セションの終了

SQLThrFreeID:

セションの破棄

セションの操作の各関数は、以下のシーケンスで発行します。



7.1.1 SQLThrAllocID

機能

プロセス内にセション(セションを実行する環境)を作成します。環境が作成された場合にはセションIDを返却します。セションIDとは、プロセス内でセションを識別するための一意なIDです。

記述形式

```
SQLRETURN SQLThrAllocID(ses_id_p)
          SQLHDBS *ses_id_p;
```

一般規則

- SQLRETURNは、2バイトの整数型です。
- ses_id_pには、セションIDを返却するアドレスを指定します。
- セションIDは、セションを実行する環境を破棄するまで必要です。必ず破棄するまで保持してください。
- 異常終了した場合、状態変数(SQLSTATE)などには影響を与えません。必ずSQLRETURNで実行状態を判別してください。
- sqlrdbei.hを組み込む必要があります。
- SQLHDBSの型のサイズはsqlrdbei.hを確認してください。

異常時の対処

復帰コード	意味	対処
SQLRDB_NORMAL	正常終了	－
SQLRDB_ENOMEM	メモリ不足	メモリを増やすか多重度を減らしてください。
SQLRDB_EINVENV	環境異常	実行環境に誤りがあります。 動作環境ファイルの実行パラメタまたは環境変数を確認してください。

7.1.2 SQLThrEndID

機能

セションとスレッドの関係付けを終了します。本関数により、そのスレッド上でSQL文が実行不可能な状態になります。

記述形式

```
SQLRETURN SQLThrEndID(ses_id)
          SQLHDBS ses_id;
```

一般規則

- SQLRETURNは、2バイトの整数型です。
- ses_idには、スレッドと関係付けを終了したいセションIDを指定します。
- 本関数の発行の前にSQLThrStartIDの発行が必要です。
- 本関数を発行したスレッド上では、別のセションのSQL文が実行可能になります。また、現在のセションを再度、開始してSQL文を実行してもかまいません。
- 異常終了した場合、状態変数(SQLSTATE)などには影響を与えません。必ずSQLRETURNで実行状態を判別してください。
- sqlrdbei.hを組み込む必要があります。

異常時の対処

復帰コード	意味	対処
SQLRDB_NORMAL	正常終了	－

復帰コード	意味	対処
SQLRDB_EINVAL	パラメタ不当	セションは開始されていません。SQLThrStartIDの発行を確認してください。
SQLRDB_ENOMEM	メモリ不足	メモリを増やすか多重度を減らしてください。

7.1.3 SQLThrFreeID

機能

セションIDで指定したセション(セションを実行する環境)を破棄します。

記述形式

```
SQLRETURN SQLThrFreeID(ses_id)
          SQLHDBS ses_id;
```

一般規則

- SQLRETURNは、2バイトの整数型です。
- ses_idには、破棄したいセションIDを指定します。
- 本関数の発行の前にSQLThrAllocIDまたはSQLThrEndIDの発行が必要です。
- 作成したセションが不要になった場合には破棄してください。破棄しない場合は、プロセスが終了するまで環境が保存されます。
- 異常終了した場合、状態変数(SQLSTATE)などには影響を与えません。必ずSQLRETURNで実行状態を判別してください。
- sqlrdbei.hを組み込む必要があります。

異常時の対処

復帰コード	意味	対処
SQLRDB_NORMAL	正常終了	－
SQLRDB_EINVAL	パラメタ不当	破棄できるセション環境が存在しません。セションIDまたはSQLThrAllocIDとSQLThrEndIDの発行を確認してください。

7.1.4 SQLThrStartID

機能

セションとスレッドの関係付けを行います。本関数により、そのスレッド上でSQL文が実行可能な状態になります。

記述形式

```
SQLRETURN SQLThrStartID(ses_id)
          SQLHDBS ses_id;
```

一般規則

- SQLRETURNは、2バイトの整数型です。
- ses_idには、SQL文を実行したいセションIDを指定します。
- 本関数の発行の前にSQLThrAllocIDの発行が必要です。
- SQLThrAllocIDと異なるスレッド上から発行可能です。
- 同時に1つのスレッド上に複数のセションを関係付けることはできません。
- 同時に複数のスレッド上に1つのセションを関係付けることはできません。
- 異常終了した場合、状態変数(SQLSTATE)などには影響を与えません。必ずSQLRETURNで実行状態を判別してください。
- sqlrdbei.hを組み込む必要があります。

異常時の対処

復帰コード	意味	対処
SQLRDB_NORMAL	正常終了	ー
SQLRDB_EINVAL	パラメタ不当	セション環境が作成されていません。 セションIdまたはSQLThrAllocIDの発行を確認してください。
SQLRDB_EINVENV	環境異常	セションが開始中です。 セション実行の排他制御を確認してください。
SQLRDB_ENOMEM	メモリ不足	メモリを増やすか多重度を減らしてください。

7.2 ルーチンの操作

ファンクションルーチンで使用できるSQLSignalMSG関数の文法を説明します。

7.2.1 SQLSignalMSG

機能

プログラムの処理で異常を検出して、Symfoware/RDBに異常復帰する場合は、SQLSignalMSG関数を使用します。

記述形式

```
void  SQLSignalMSG(char * msg_p)
```

一般規則

- msg_pには、メッセージ格納先アドレスを指定します。
- メッセージの長さは、256バイト以内です。超えた場合は切り捨てられます。
- メッセージは、サーバシステムの文字コード系で作成します。
- 検出した事象を設定したSQLSignalMSG関数を実行後、プログラムからSymfoware/RDBに復帰するとSQL文の処理が異常として扱われSQL文を実行したアプリケーションにメッセージが通知されます。

7.3 コールバックの操作

コールバック機能で使用できる関数の文法を説明します。

7.3.1 SQLDynSetCallback

機能

コールバック関数を動的に登録します。このコールバック関数は、利用者が作成します。

記述形式

```
SQLRETURN SQLDynSetCallback ( SQLHDBS ses_id )
```

一般規則

- SQLRETURNは、2バイトの整数型です。
- マルチスレッドで動作するアプリケーションの場合は、ses_idにはセションIDが設定されます。シングルスレッドで動作するアプリケーションの場合は、0が設定されます。
- SQLDynSetCallback関数は、以下のタイミングでロードされ実行されます。
  - ー マルチスレッドで動作するアプリケーションの場合は、SQLThrAllocID関数の処理でコールされます。
  - ー シングルスレッドで動作するアプリケーションの場合は、プロセス内で最初に実行されたSQL文の処理でコールされます。
- SQLDynSetCallback関数の内部では、SQLSetCallback関数、SQLGetCallback関数以外のSQL拡張インタフェースおよび埋込みSQL文は実行できません。
- 正常復帰する場合は、SQLRDB_NORMALを返してください。異常終了する場合は、SQLRDB_NORMAL以外を返してください。
  - ー マルチスレッドで動作するアプリケーションの場合は、SQLThrAllocID関数でエラー復帰します。
  - ー シングルスレッドで動作するアプリケーションの場合は、SQLDynSetCallback関数がコールされたSQL文の実行結果として、SQLSTATEに74600を設定します。またホスト変数SQLMSGが定義されている場合は、返却された値を詳細種別に埋め込んだ以下のエラーメッセージを出力します。
    - JYP1085E:コールバック関数の登録処理でエラーが発生しました. 詳細種別=“@1@”
- sqlrdbei.hを組み込む必要があります。

異常時の対処

復帰コード	意味	対処
SQLRDB_NORMAL	正常終了	ー
SQLRDB_NORMAL以外の任意の値	異常終了	コールバック関数の登録に失敗しました。

使用例

INSERT文、UPDATE文、DELETE文および単一行SELECT文に対するコールバック関数を実行時に登録します。

登録用DLLのプログラム

```
#include <stdio.h>
#include "sqlrdbei.h"

/* コールバック関数の動的登録関数定義 */
/* コールバック関数のプロトタイプ */

/* INSERT文用 */
SQLRETURN CFInsert( SQLHDBS, char *, char *, void *, SQLCALL_T * );
```

```
/* UPDATE文用 */
SQLRETURN CFUpdate( SQLHDBS, char *, char *, void *, SQLCALL_T * );

/* DELETE文用 */
SQLRETURN CFDelete( SQLHDBS, char *, char *, void *, SQLCALL_T * );

/* 単一行SELECT文用 */
SQLRETURN CFSelect( SQLHDBS, char *, char *, void *, SQLCALL_T * );

SQLRETURN SQLDynSetCallback( SQLHDBS  sid )
{
    SQLRETURN    ret;
    short        kind_list[2];

    /* INSERT文の入口コールバック関数 */
    kind_list[0] = SQLRDB_CF_INSERT;
    kind_list[1] = 0;
    ret = SQLSetCallback( sid, CFInsert, NULL, kind_list, SQLRDB_CALL_IN );
    if ( ret != SQLRDB_NORMAL ) {
        return( ret );
    }

    /* UPDATE文の入口コールバック関数 */
    kind_list[0] = SQLRDB_CF_UPDATE_SEARCH;
    kind_list[1] = 0;
    ret = SQLSetCallback( sid, CFUpdate, NULL, kind_list, SQLRDB_CALL_IN );
    if ( ret != SQLRDB_NORMAL ) {
        return( ret );
    }

    /* DELETE文の入口コールバック関数 */
    kind_list[0] = SQLRDB_CF_DELETE_SEARCH;
    kind_list[1] = 0;
    ret = SQLSetCallback( sid, CFDelete, NULL, kind_list, SQLRDB_CALL_IN );
    if ( ret != SQLRDB_NORMAL ) {
        return( ret );
    }

    /* 単一行SELECT文の入口コールバック関数 */
    kind_list[0] = SQLRDB_CF_SELECT;
    kind_list[1] = 0;
    ret = SQLSetCallback( sid, CFSelect, NULL, kind_list, SQLRDB_CALL_IN );
    if ( ret != SQLRDB_NORMAL ) {
        return( ret );
    }

    return( SQLRDB_NORMAL );
}

SQLRETURN CFInsert(    SQLHDBS        sid,
                       char        *sqlstate,
                       char        *sqlmsg,
                       void        *area,
                       SQLCALL_T    *sql_dat)
{
    SQLDATA_T        param;
    short            roop;

    char            sqlstr[1024];

    printf( "ファイル名 : %s¥n", sql_dat->application );
    printf( "SQL文      : %s¥n", sql_dat->sql_stmt );
```

```
    return ( SQLRDB_CONTINUE );
}

SQLRETURN CFUpdate(    SQLHDBS        sid,
                       char        *sqlstate,
                       char        *sqlmsg,
                       void        *area,
                       SQLCALL_T    *sql_dat)
{
    SQLDATA_T        param;
    short            roop;

    char            sqlstr[1024];

    printf( "ファイル名 : %s¥n", sql_dat->application );
    printf( "SQL文      : %s¥n", sql_dat->sql_stmt );

    return ( SQLRDB_CONTINUE );
}

SQLRETURN CFDelete( SQLHDBS     sid,
                    char        *sqlstate,
                    char        *sqlmsg,
                    void        *area,
                    SQLCALL_T   *sql_dat)
{
    SQLDATA_T        param;
    short            roop;

    char            sqlstr[1024];

    printf( "ファイル名 : %s¥n", sql_dat->application );
    printf( "SQL文      : %s¥n", sql_dat->sql_stmt );

    return ( SQLRDB_CONTINUE );
}

SQLRETURN CFSelect( SQLHDBS     sid,
                    char        *sqlstate,
                    char        *sqlmsg,
                    void        *area,
                    SQLCALL_T   *sql_dat)
{
    SQLDATA_T        param;
    short            roop;

    char            sqlstr[1024];

    printf( "ファイル名 : %s¥n", sql_dat->application );
    printf( "SQL文      : %s¥n", sql_dat->sql_stmt );

    return ( SQLRDB_CONTINUE );
}
```

7.3.2 SQLGetCallback

機能

指定されたSQL文種別に対し、コールバック関数の登録状況を取得します。

記述形式

```
SQLRETURN SQLGetCallback (  SQLHDBS    ses_id,
                            SQLRETURN (**function)( SQLHDBS,
                                                    char *,
                                                    char *,
                                                    void *,
                                                    struct SQLCALL_T * ),
                            short      sql_kind,
                            short      when )
```

一般規則

- SQLRETURNは、2バイトの整数型です。
- マルチスレッドで動作するアプリケーションの場合は、ses_idにセションIDを指定します。シングルスレッドで動作するアプリケーションの場合は、ses_idに0を指定します。
- 現在、コールバック関数が登録されている場合はfunctionに関数アドレスが設定されます。登録されていない場合は、NULLが設定されます。
- functionにNULLを指定した場合でもエラーとはなりません。ただし、コールバック関数が登録されていても関数アドレスは取得できません。
- sql_kindには、コールバック関数の登録状況を取得したいSQL文種別を指定します。
- SQL文種別に指定できる値は、“7.3.3 SQLSetCallback”を参照してください。
- whenに指定できる値は、“7.3.3 SQLSetCallback”を参照してください。
- sqlrdbei.hを組み込む必要があります。

異常時の対処

復帰コード	意味	対処
SQLRDB_ENTRY	登録済み	－
SQLRDB_NOENTRY	未登録	－
SQLRDB_EINVAL	パラメタ不当	ses_id、sql_kindまたはwhenに設定した値を見直してください。
SQLRDB_ENOENV	環境未開設	セション環境が作成されていません。セションIDまたはSQLThrAllocIDの発行を確認してください。

使用例

INSERT文に対するコールバック関数の登録状況を取得し、未登録ならコールバック関数を登録します。

```
#include <stdio.h>
#include "sqlrdbei.h"

 /* コールバック関数のプロトタイプ */
SQLRETURN CFInsertEntry( SQLHDBS, char *, char *, void *, SQLCALL_T * );
SQLRETURN CFInsertExit( SQLHDBS, char *, char *, void *, SQLCALL_T * );

void main()
{
    EXEC SQL BEGIN DECLARE SECTION;
        char        SQLSTATE[6];
        char        SQLMSG[256];
    EXEC SQL END DECLARE SECTION;

    SQLRETURN       ret;
```

```
    short           kind_list[2];
    SQLRETURN       (* pfunc_entry)(SQLHDBS, char *,
                                   char *, void *, SQLCALL_T * );
    SQLRETURN       (* pfunc_exit)(SQLHDBS, char *,
                                   char *, void *, SQLCALL_T * );

    /* INSERT文に対する入口コールバック関数の登録状況を取得する。 */
    /* 現在、コールバック関数が登録されている場合は、そのアドレスを取得する。*/
    kind_list[0] = SQLRDB_CF_INSERT;
    kind_list[1] = 0;
    ret = SQLGetCallback( 0, &pfunc_entry, SQLRDB_CF_INSERT, SQLRDB_CALL_IN );
    if ( ret == SQLRDB_NOENTRY ) {
        kind_list[0] = SQLRDB_CF_INSERT;
        kind_list[1] = 0;
        SQLSetCallback( 0, CFInsertEntry, NULL, kind_list, SQLRDB_CALL_IN );
    }

    /* INSERT文に対する出口コールバック関数の登録状況を取得する。 */
    /* 現在、コールバック関数が登録されている場合は、そのアドレスを取得する。*/
    ret = SQLGetCallback( 0, &pfunc_exit, SQLRDB_CF_INSERT, SQLRDB_CALL_OUT );
    if ( ret == SQLRDB_NOENTRY ) {
        kind_list[0] = SQLRDB_CF_INSERT;
        kind_list[1] = 0;
        SQLSetCallback( 0, CFInsertExit, NULL, kind_list, SQLRDB_CALL_OUT );
    }

    EXEC SQL INSERT INTO SCHEMA1.TABLE1( COL01, COL02, COL_KEY )
                                                    VALUES ( 1, 2, 3 );

    /* 変更前のコールバック関数を復元する。 */
    if ( pfunc_entry != NULL ) {
        SQLSetCallback( 0, pfunc_entry, NULL, kind_list, SQLRDB_CALL_IN );
    }
    if ( pfunc_exit != NULL ) {
        SQLSetCallback( 0, pfunc_exit, NULL, kind_list, SQLRDB_CALL_OUT );
    }

    return;
}

SQLRETURN CFInsertEntry( SQLHDBS     sid,
                         char        *sqlstate,
                         char        *sqlmsg,
                         void        *area,
                         SQLCALL_T   *sql_dat)
{
    SQLDATA_T       param;
    short           roop;

    char           sqlstr[1024];

    printf( "ファイル名 : %s¥n", sql_dat->application );
    printf( "SQL文      : %s¥n", sql_dat->sql_stmt );

    return ( SQLRDB_CONTINUE );
}

SQLRETURN CFInsertExit( SQLHDBS     sid,
                        char        *sqlstate,
                        char        *sqlmsg,
                        void        *area,
                        SQLCALL_T   *sql_dat)
{
```

```
    SQLDATA_T        param;
    short            roop;

    char            sqlstr[1024];

    printf( "ファイル名 : %s¥n", sql_dat->application );
    printf( "SQL文      : %s¥n", sql_dat->sql_stmt );
    printf( "SQLSTATE   : %s¥n", sqlstate );
    printf( "SQLMSG     : %s¥n", sqlmsg );

    return ( SQLRDB_CONTINUE );
}
```

7.3.3 SQLSetCallback

機能

指定されたSQL文種別に対し、コールバック関数を登録します。

記述形式

```
SQLRETURN SQLSetCallback (  SQLHDBS  ses_id,
                            SQLRETURN (*コールバック関数名)( SQLHDBS,
                                                             char *,
                                                             char *,
                                                             void *,
                                                         struct SQLCALL_T * ),
                            void     *user_area,
                            short     sql_kind[],
                            short     when )
```

一般規則

- SQLRETURNは、2バイトの整数型です。
- マルチスレッドで動作するアプリケーションの場合は、ses_idにセションIDを指定します。シングルスレッドで動作するアプリケーションの場合は、ses_idに0を指定します。
- コールバック関数名には、登録するコールバック関数の関数アドレスを指定します。コールバック関数の詳細は“7.3.4 コールバック関数”を参照してください。
- SQLCallbackFunctionにNULLを指定すると、コールバック関数の登録を解除することができます。
- user_areaには、ユーザが定義する領域のアドレスを指定します。このアドレスは、コールバック関数がコールされるとき第4パラメタに設定されます。
- sql_kindには、コールバック関数を登録するSQL文種別のリストを指定します。SQL文種別はshort型の配列で指定し、配列の最後は0を指定します。
- SQL文種別には以下の定数を指定します。

SQL文種別	指定値
ALLOCATE DESCRIPTOR文	SQLRDB_CF_ALLOC_DESC
CALL文	SQLRDB_CF_CALL
CLOSE文	SQLRDB_CF_CLOSE
COMMIT文	SQLRDB_CF_COMMIT
CONNECT文	SQLRDB_CF_CONNECT
DEALLOCATE DESCRIPTOR文	SQLRDB_CF_DEALLOC_DESC

SQL文種別	指定値
DEALLOCATE PREPARE文	SQLRDB_CF_DEALLOC_PREPARE
DELETE文:位置づけ	SQLRDB_CF_DELETE_POSITION
DELETE文:探索	SQLRDB_CF_DELETE_SEARCH
DESCRIBE文	SQLRDB_CF_DESCRIBE
DISCONNECT文	SQLRDB_CF_DISCONNECT
EXECUTE IMMEDIATE文	SQLRDB_CF_EXECUTE_IMME
EXECUTE文	SQLRDB_CF_EXECUTE
FETCH文	SQLRDB_CF_FETCH
GET DESCRIPTOR文	SQLRDB_CF_GET_DESC
INSERT文	SQLRDB_CF_INSERT
OPEN文	SQLRDB_CF_OPEN
PREPARE文	SQLRDB_CF_PREPARE
RELEASE文	SQLRDB_CF_RELEASE
ROLLBACK文	SQLRDB_CF_ROLLBACK
SET CATALOG文	SQLRDB_CF_SET_CATALOG
SET CONNECTION文	SQLRDB_CF_SET_CONNECT
SET DESCRIPTOR文	SQLRDB_CF_SET_DESC
SET ROLE文	SQLRDB_CF_SET_ROLE
SET SCHEMA文	SQLRDB_CF_SET_SCHEMA
SET SESSION AUTHORIZATION文	SQLRDB_CF_SET_SESSION
SET TRANSACTION文	SQLRDB_CF_SET_TRANSACTION
SET USER PASSWORD文	SQLRDB_CF_SET_USER
UPDATE文:位置づけ	SQLRDB_CF_UPDATE_POSITION
UPDATE文:探索	SQLRDB_CF_UPDATE_SEARCH
単一行SELECT文	SQLRDB_CF_SELECT

- コールバック関数は、プロシジャルーチン内のSQL文は対象となりません。
- whenには目的に応じて以下の定数を指定します。

when	意味
SQLRDB_CALL_IN	入口コールバック関数として登録します。
SQLRDB_CALL_OUT	出口コールバック関数として登録します。

- コールバック関数は、マルチスレッドで動作するアプリケーションの場合は、SQLThrAllocID関数によりセションを作成した後はいつでも登録できます。シングルスレッドで動作するアプリケーションの場合は、いつでも登録することができます。
- sqlrdbei.hを組み込む必要があります。

異常時の対処

復帰コード	意味	対処
SQLRDB_NORMAL	正常終了	－
SQLRDB_EINVAL	パラメタ不当	ses_id、sql_kindまたはwhenに設定した値を見直してください。
SQLRDB_ENOENV	環境未開設	セション環境が作成されていません。セションIDまたはSQLThrAllocIDの発行を確認してください。

使用例

INSERT文、DELETE文、UPDATE文および単一行SELECT文にコールバック関数を登録します。

```
#include <stdio.h>
#include "sqlrdbei.h"

/* コールバック関数のプロトタイプ */
SQLRETURN CFEntry1( SQLHDBS, char *, char *, void *, SQLCALL_T * );
SQLRETURN CFEntry2( SQLHDBS, char *, char *, void *, SQLCALL_T * );

void main()
{
    EXEC SQL BEGIN DECLARE SECTION;
        char    SQLSTATE[6];
        char    SQLMSG[256];
        short   col01;
        short   col02;
        short   key_dat;
    EXEC SQL END DECLARE SECTION;

    SQLRETURN       ret;
    short           kind_list[5];

    /* CBEntry1をINSERT文、DELETE文、UPDATE文、単一行SELECT文 */
    /* の入口コールバック関数に登録する。*/

    kind_list[0] = SQLRDB_CF_INSERT;
    kind_list[1] = SQLRDB_CF_DELETE_SEARCH;
    kind_list[2] = SQLRDB_CF_UPDATE_SEARCH;
    kind_list[3] = SQLRDB_CF_SELECT;
    kind_list[4] = 0;
    ret = SQLSetCallback( 0, CFEntry1, NULL, kind_list, SQLRDB_CALL_IN );

    EXEC SQL INSERT INTO SCHEMA1.TABLE1( COL01, COL02, COL_KEY )
            VALUES ( 1, 2, 100 );                                    ・・・(1)
    EXEC SQL UPDATE SCHEMA1.TABLE1 SET COL01 = 10, COL02 = 20
            WHERE COL_KEY = 100;                                     ・・・(1)
    EXEC SQL SELECT COL01, COL02, COL_KEY
            INTO :col01, :col02
            FROM SCHEMA1.TABLE1
            WHERE COL_KEY = 100;                                     ・・・(1)
    EXEC SQL DELETE FROM SCHEMA1.TABLE1 WHERE COL_KEY = 100;         ・・・(1)

    /* INSERT文、DELETE文の入口コールバック関数をCBEntry2に変更する。 */
    kind_list[0] = SQLRDB_CF_INSERT;
    kind_list[1] = SQLRDB_CF_DELETE_SEARCH;
    kind_list[2] = 0;
    ret = SQLSetCallback( 0, CFEntry2, NULL, kind_list, SQLRDB_CALL_IN );
```

```
    /* 単一行SELECT文の入口コールバック関数を解除する。 */
    kind_list[0] = SQLRDB_CF_SELECT;
    kind_list[1] = 0;
    ret = SQLSetCallback( 0, NULL, NULL, kind_list, SQLRDB_CALL_IN );

    EXEC SQL INSERT INTO SCHEMA1.TABLE1( COL01, COL02, COL_KEY )
            VALUES ( 1,2,101 );                                          ・・・(2)
    EXEC SQL UPDATE SCHEMA1.TABLE1 SET COL01 = 5, COL02 = 6
            WHERE COL_KEY = 101;                                         ・・・(1)
    EXEC SQL SELECT COL01, COL02, COL_KEY
            INTO :col01, :col02
            FROM SCHEMA1.TABLE1
            WHERE COL_KEY = 101;                                         ・・・(3)
    EXEC SQL DELETE FROM SCHEMA1.TABLE1 WHERE COL_KEY = 101;             ・・・(2)

    return;
}

SQLRETURN CFEntry1( SQLHDBS     sid,
                    char        *sqlstate,
                    char        *sqlmsg,
                    void        *area,
                    SQLCALL_T   *sql_dat)
{
    SQLDATA_T        param;
    short            roop;

    char            sqlstr[1024];

    printf( "ファイル名 : %s¥n", sql_dat->application );
    printf( "SQL文      : %s¥n", sql_dat->sql_stmt );

    return ( SQLRDB_CONTINUE );
}

SQLRETURN CFEntry2( SQLHDBS     sid,
                    char        *sqlstate,
                    char        *sqlmsg,
                    void        *area,
                    SQLCALL_T   *sql_dat)
{
    SQLDATA_T        param;
    short            roop;

    char            sqlstr[1024];

    printf( "ファイル名 : %s¥n", sql_dat->application );
    printf( "SQL文      : %s¥n", sql_dat->sql_stmt );

    return ( SQLRDB_CONTINUE );
}
```

(1) 入口コールバック関数としてCBEntry1がコールされる。

(2) 入口コールバック関数としてCBEntry2がコールされる。

(3) 入口コールバック関数はコールされない。

7.3.4 コールバック関数

機能

コールバック関数の関数定義です。コールバック関数は以下の書式に従って利用者が作成します。

記述形式

```
SQLRETURN コールバック関数名 （  SQLHDBS          ses_id,
                                 char      *SQLSTATE,
                                 char      *SQLMSG,
                                 void      *user_area,
                                 SQLCALL_T  *SQLData ）
```

一般規則

- コールバック関数名は、ユーザが任意に指定できます。
- SQLRETURNは、2バイトの整数型です。
- 入口コールバック関数がSQLRDB_NOCONTINUEを返した場合は、SQL文の処理は実行されません。
- 入口コールバック関数がSQLRDB_NOCONTINUE以外を返した場合は、SQLRDB_CONTINUEが返されたものと見なします。
- マルチスレッドで動作するアプリケーションの場合は、ses_idにセションIDが設定されます。シングルスレッドで動作するアプリケーションの場合は、ses_idに0が設定されます。
- SQLSTATEおよびSQLMSGには、以下の値が設定されます。
  - 入口コールバック関数の場合は、NULLが設定されます。
  - 出口コールバック関数の場合は、SQL文の実行結果が設定されます。ただし、入口コールバック関数がSQLRDB_NOCONTINUEを返した場合は、NULLが設定されます。
- アプリケーションにホスト変数SQLMSGが定義されていない場合は、SQLMSGにはNULLが設定されます。
- SQLMSGには、アプリケーションの埋込みSQL宣言節で指定した領域長の後にNULL終端文字を付加したものを設定します。
- user_areaには、SQLSetCallback関数でユーザが指定したアドレスが設定されます。
- sqlrdbei.hを組み込む必要があります。
- SQLDataが指す領域は以下の形式です。

```
struct SQLCALL_T {
    long                    total_size;
    char                    *application;
    long                    sql_line;
    long                    sql_kind;
    long                    sql_length;
    char                    *sql_stmt;
    long                    sql_variable_length;
    char                    *sql_variable_stmt;
    void                    *reserve;
    long                    data_num;
    struct SQLDATA_T        data[1];
};

struct SQLDATA_T {
    long                    length;
    short                   type;
    short                   precision;
    short                   scale;
    short                   rsv;
    void                    *area;
    short                   *indicator;
    void                    *reserve;
}
```

SQLCALL_T 構造体のパラメタの意味

パラメタ名	意味
total_size	構造体の全体の長さ
application	SQL埋込みプログラムのソースファイル名のアドレス
sql_line	SQL埋込みプログラムのソースファイルの行番号
sql_kind	SQL文の種別
sql_length	SQL文の長さ
sql_stmt	SQL文
sql_variable_length	被準備文の長さ
sql_variable_stmt	被準備文
reserve	将来のために予約されています。
data_num	入出力データの個数
data	データ構造体のポインタ

- applicationは以下のようになります。
  - NULL終端文字を付加した、SQL埋込みプログラムのソースファイル名のアドレスが設定されます。
  - SQL埋込みプログラムのソースファイル名は、拡張子を含みません。
  - SQL文がINCLUDE文で指定したファイルに記述されている場合、applicationには、NULL終端文字を付加した、INCLUDE文で指定したソースファイル名のアドレスが設定されます。
- sql_lineは以下のようになります。
  - SQL文が記述されているSQL埋込みプログラムのソースファイルの行番号が設定されます。
  - SQL文が複数の行で構成されている場合は、先頭の行番号が設定されます。
  - SQL文がINCLUDE文で指定したファイルに記述されている場合、sql_lineには、INCLUDE文で指定したソースファイルの行番号が設定されます。
- sql_kindに設定される値は、“7.3.3 SQLSetCallback”を参照してください。
- sql_lengthはNULL終端文字を含みます。
- sql_stmtは、実行したSQL文のアドレスが設定されます。
- SQL文の値指定にホスト変数または標識変数が指定されている場合は、コールバック関数に通知されるSQL文では動的パラメタ指定の表記に変更されます。
- SQL埋込みプログラムソースでSQL文の途中にタブ文字、改行またはコメントが存在する場合は、これらは空白文字に置き換えられます。
- sql_variable_lengthは以下のようになります。
  - EXECUTE文、EXECUTE IMMEDIATE文およびPREPARE文の場合は、NULL終端文字を含む被準備文の長さが設定されます。
  - OPEN文の場合は、カーソル宣言に指定した問合せ式のNULL終端文字を含む長さが設定されます。
- sql_variable_stmtは以下のようになります。
  - EXECUTE文、EXECUTE IMMEDIATE文およびPREPARE文の場合は、NULL終端文字を付加した被準備文のアドレスが設定されます。
  - OPEN文の場合は、カーソル宣言に指定した問合せ式にNULL終端文字を付加した問合せ式のアドレスが設定されます。
- reserveは将来のために予約されています。
- data_numには、入出力データの個数が設定されます。

ー dataは以下のようになります。

- 入口コールバック関数の場合は、入力データの情報が設定されます。また、出口コールバック関数の場合は、出力データの情報になります。
- SQLDATA_T構造体のアドレスが設定されます。SQLDATA_T構造体は以下のパラメタが設定されます。

SQLDATA_T 構造体のパラメタの意味

パラメタ名	意味
length	データの領域長
type	データの型
precision	データの精度
scale	データの位取り
rsv	将来のために予約されています。
area	データ領域のアドレス
indicator	標識変数のアドレス
reserve	将来のために予約されています。

ー 入力データが入力ホスト変数の場合の順番は、SQL文の中に出てくる動的パラメタ指定の順番になります。また、出力データが出力ホスト変数の場合は、SQL文の中の相手指定の順番になります。

各データ型に対応する個々の値は、以下の通りです。

データの型	type	length	precision	scale
文字列型	SQLRDB_CHAR	1～32000 (注1)	0	0
可変長文字列型	SQLRDB_VARCHAR	3～32002 (注1)	0	0
拡張可変長文字列型	SQLRDB_EXTVARCHAR	5～32004 (注1)	0	0
各国語文字列型	SQLRDB_NCHAR	1～32000 (注1)	0	0
可変長各国語文字列型	SQLRDB_NVARCHAR	4～32002 (注1)	0	0
拡張可変長各国語文字列型	SQLRDB_EXTNVARCHAR	6～32004 (注1)	0	0
short	SQLRDB_SHORT	2	15	0
int	SQLRDB_INT	4	31	0
float	SQLRDB_FLOAT	4	23	0
double	SQLRDB_DOUBLE	8	52	0
NUMERIC	SQLRDB_NUMERIC	1～19	1～18	0～18
DECIMAL	SQLRDB_DECIMAL	1～10	1～18	0～18
BLOB	SQLRDB_BLOB	n*1024 (注2)	0	0
ROW_ID	SQLRDB_LOW_ID	24	0	0

注1) 単位はバイトです。
注2) nは1～2097150を示します。

各SQL文ごとにdataに設定される値は、以下の通りです。

SQL文種別	SQL文領域	被準備文	入力データ	出力データ
ALLOCATE DESCRIPTOR文 (注1)	SQL文	－	－	－
CALL文	SQL文	－	データ	データ
CLOSE文	SQL文	－	－	－
COMMIT文	SQL文	－	－	－
CONNECT文 (注2)、(注3)	SQL文	－	－	－
DEALLOCATE DESCRIPTOR文 (注4)	SQL文	－	－	－
DEALLOCATE PREPARE文	SQL文	－	－	－
DELETE文:位置づけ	SQL文	－	データ	－
DELETE文:探索	SQL文	－	データ	－
DESCRIBE文 (注4)	SQL文	－	－	－
DISCONNECT文 (注2)	SQL文	－	－	－
EXECUTE IMMEDIATE文	SQL文	被準備文	－	－
EXECUTE文	SQL文	被準備文	データ	データ
FETCH文	SQL文	－	－	データ
GET DESCRIPTOR文 (注5)	SQL文	－	－	取得識別子のデータ
INSERT文	SQL文	－	データ	－
OPEN文	SQL文	問合せ指定	データ	－
PREPARE文	SQL文	被準備文	－	－
RELEASE文	SQL文	－	－	－
ROLLBACK文	SQL文	－	－	－
SET CATALOG文 (注6)	SQL文	－	－	－
SET CONNECTION文 (注2)	SQL文	－	－	－
SET DESCRIPTOR文 (注7)	SQL文	－	設定識別子のデータ	－
SET ROLE文	SQL文	－	－	－
SET SCHEMA文 (注8)	SQL文	－	－	－
SET SESSION AUTHORIZATION文 (注2)、(注3)	SQL文	－	－	－
SET TRANSACTION文	SQL文	－	－	－
SET USER PASSWORD文 (注3)	SQL文	－	－	－
UPDATE文:位置づけ	SQL文	－	データ	－
UPDATE文:探索	SQL文	－	データ	－

SQL文種別	SQL文領域	被準備文	入力データ	出力データ
単一行SELECT文	SQL文	－	データ	データ

注1) 記述子名または実現値をホスト変数で指定した場合は、設定されるSQL文の中にはホスト変数の値が埋め込まれます。

注2) SQLサーバ名、コネクション名またはユーザ指定をホスト変数で指定した場合は、設定されるSQL文の中にはホスト変数の値が埋め込まれます。

注3) セキュリティ上、指定されたパスワードはSQL文中には表示しません。

注4) 記述子名をホスト変数で指定した場合は、設定されるSQL文の中にはホスト変数の値が埋め込まれます。

注5) 記述子名またはSQLVAR取得番号をホスト変数で指定した場合は、設定されるSQL文の中にはホスト変数の値が埋め込まれます。また、SQLVAR取得情報は動的パラメタ指定の表記に変更されます。

注6) データベース名をホスト変数で指定した場合は、設定されるSQL文の中にはホスト変数の値が埋め込まれます。

注7) 記述子名またはSQLVAR設定番号をホスト変数で指定した場合は、設定されるSQL文の中にはホスト変数の値が埋め込まれます。また、SQLVAR設定情報は動的パラメタ指定の表記に変更されます。

注8) スキーマ名をホスト変数で指定した場合は、設定されるSQL文の中にはホスト変数の値が埋め込まれます。

異常時の対処

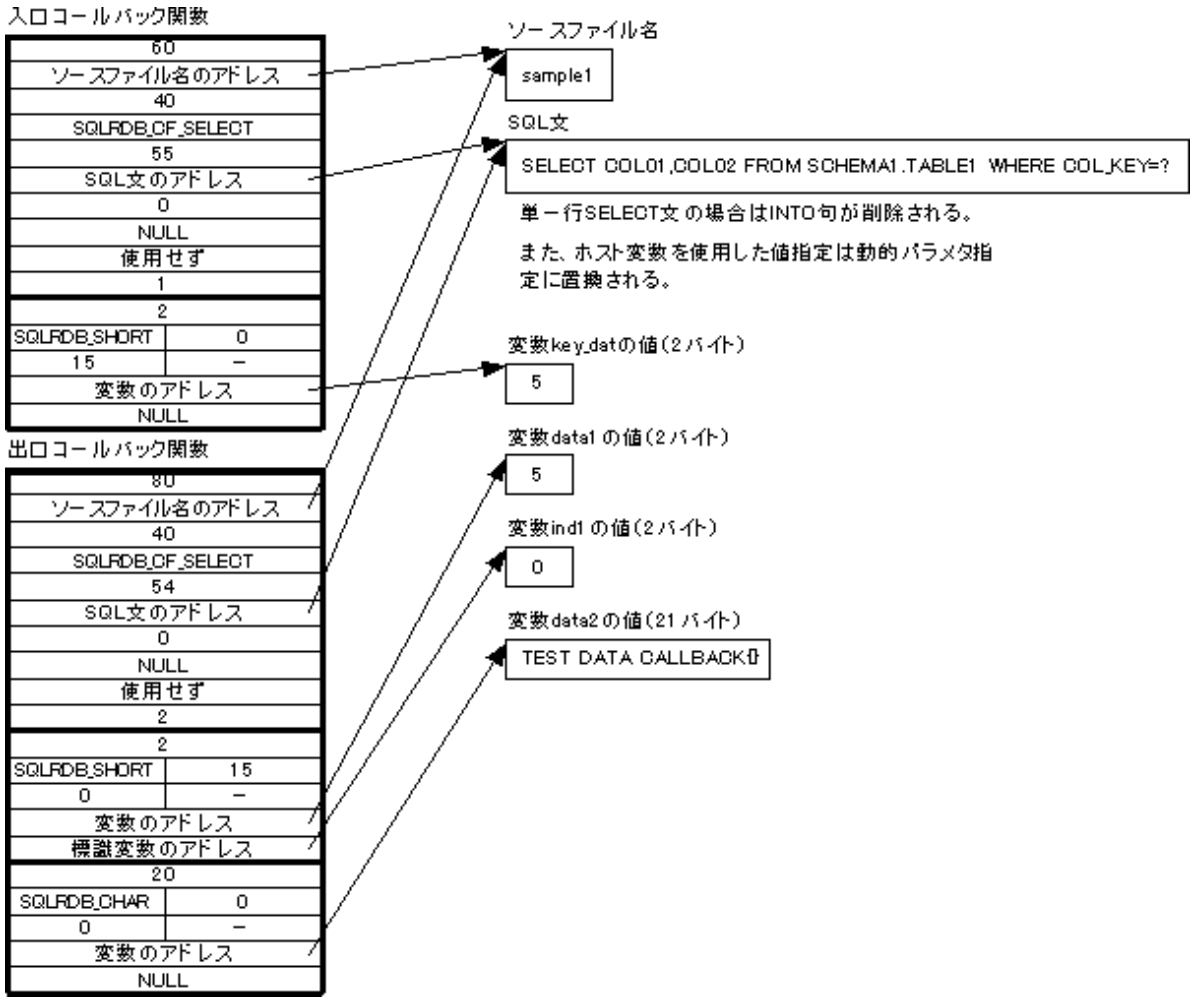
復帰コード	意味	対処
SQLRDB_CONTINUE	正常終了	－
SQLRDB_NOCONTINUE	異常終了	SQL文の処理を実行しません。

使用例

単一行SELECT文を実行した場合のデータ構造

```
EXEC SQL BEGIN DECLARE SECTION;
    char    SQLSTATE[6];
    char    SQLMSG[256];
    short   data1;
    short   ind1;
    char    data2[21];
EXEC SQL END DECLARE SECTION;
                    ▪
                    ▪
EXEC SQL SELECT COL01, COL02 INTO :data1 :ind1, :data2
            FROM SCHEMA1.TABLE1
            WHERE COL_KEY = 5;
                    ▪
                    ▪
```

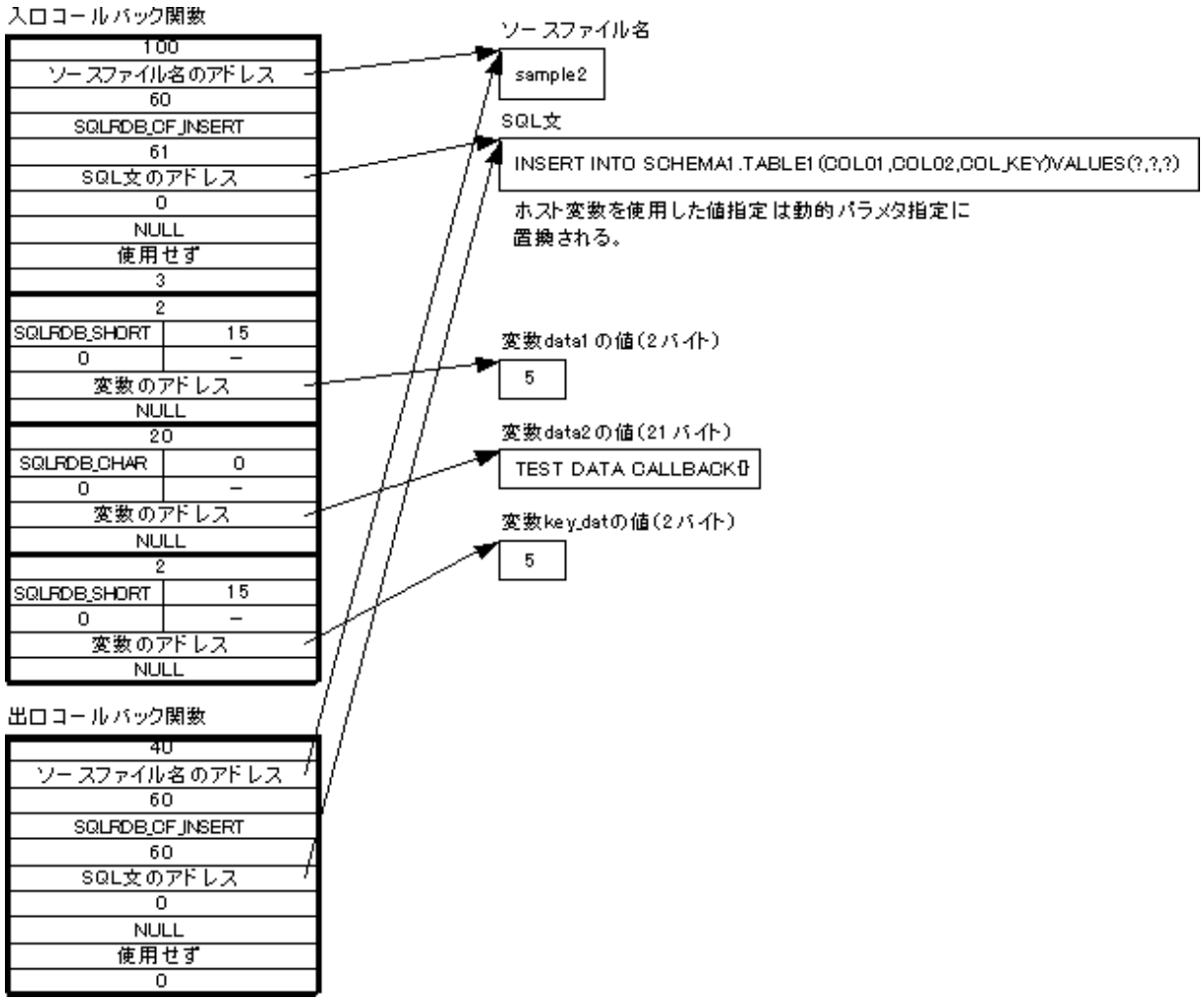
上記SQL文を実行した場合は、コールバック関数に通知されるSQL文情報は以下のようになります。



INSERT文を実行した場合のデータ構造

```
EXEC SQL BEGIN DECLARE SECTION;
    char    SQLSTATE[6];
    char    SQLMSG[256];
    short   data1;
    char    data2[21];
    short   key_dat;
EXEC SQL END DECLARE SECTION;
                    ▪
                    ▪
data1 = 5;
strcpy( data2, “TEST DATA CALLBACK()” );
key_dat = 5;
EXEC SQL INSERT INTO SCHEMA1.TABLE1( COL01, COL02, COL_KEY )
            VALUES ( :data1, :data2, :key_dat );
                    ▪
                    ▪
```

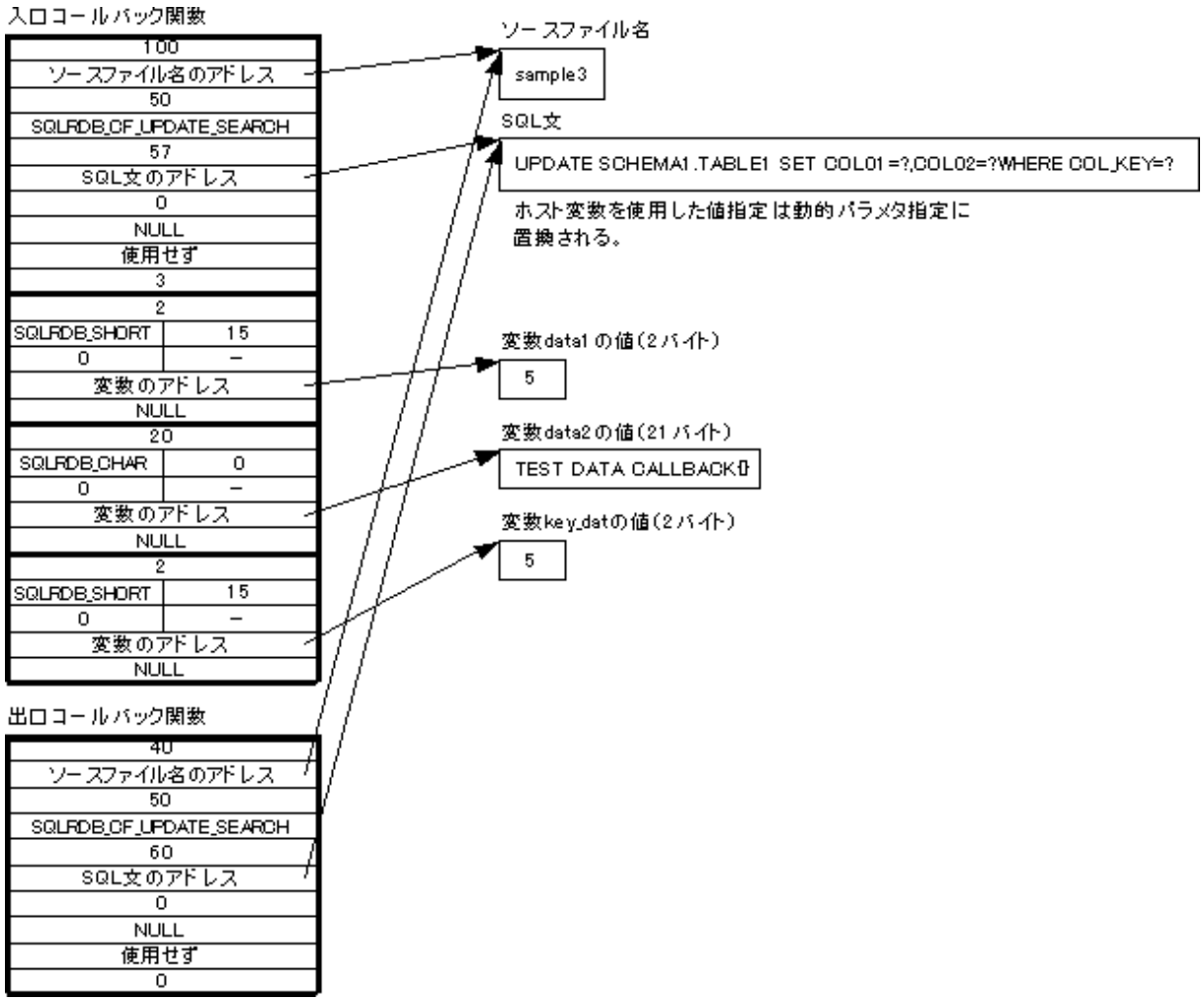
上記SQL文を実行した場合は、コールバック関数に通知されるSQL文情報は以下のようになります。



UPDATE文を実行した場合のデータ構造

```
EXEC SQL BEGIN DECLARE SECTION;
    char    SQLSTATE[6];
    char    SQLMSG[256];
    short   data1;
    char    data2[21];
    short   key_dat;
EXEC SQL END DECLARE SECTION;
                    ▪
                    ▪
data1 = 5;
strcpy( data2,  “TEST DATA CALLBACK()” );
key_dat = 5;
EXEC SQL UPDATE SCHEMA1.TABLE1 SET COL01 = :data1, COL02 = :data2
            WHERE COL_KEY = :key_dat;
                    ▪
                    ▪
```

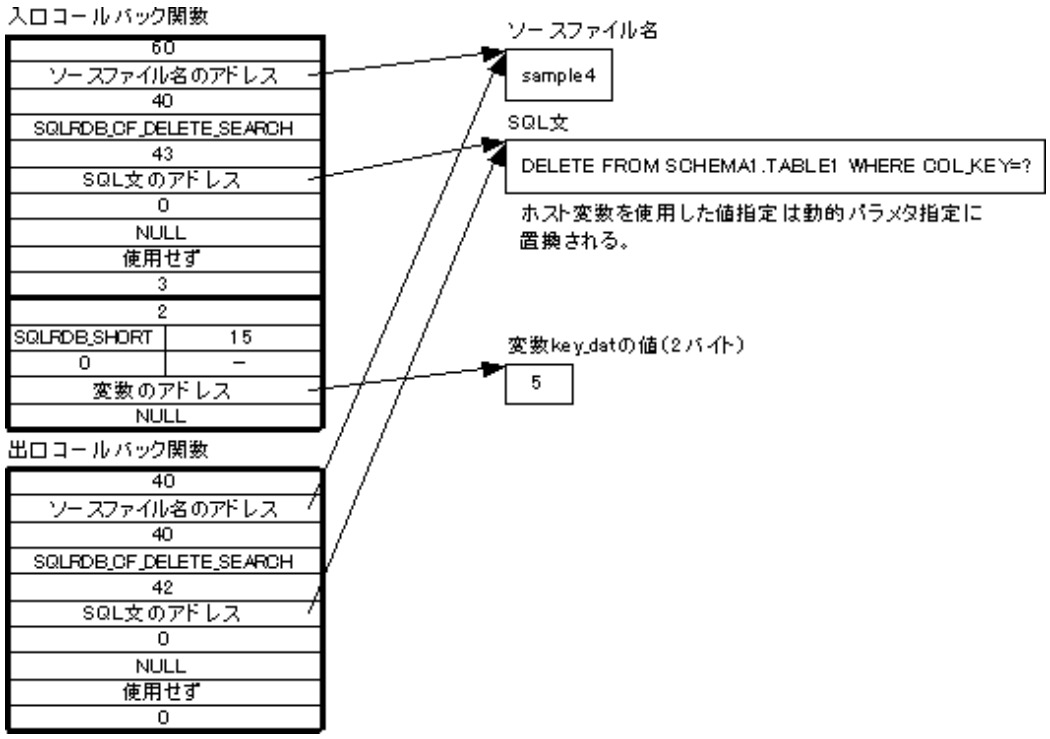
上記SQL文を実行した場合は、コールバック関数に通知されるSQL文情報は以下のようになります。



DELETE文を実行した場合のデータ構造

```
EXEC SQL BEGIN DECLARE SECTION;
    char    SQLSTATE[6];
    char    SQLMSG[256];
    short   data1;
    char    data2[21];
    short   key_dat;
EXEC SQL END DECLARE SECTION;
                    ▪
                    ▪
EXEC SQL DELETE FROM SCHEMA1.TABLE1 WHERE COL_KEY = 5;
                    ▪
```

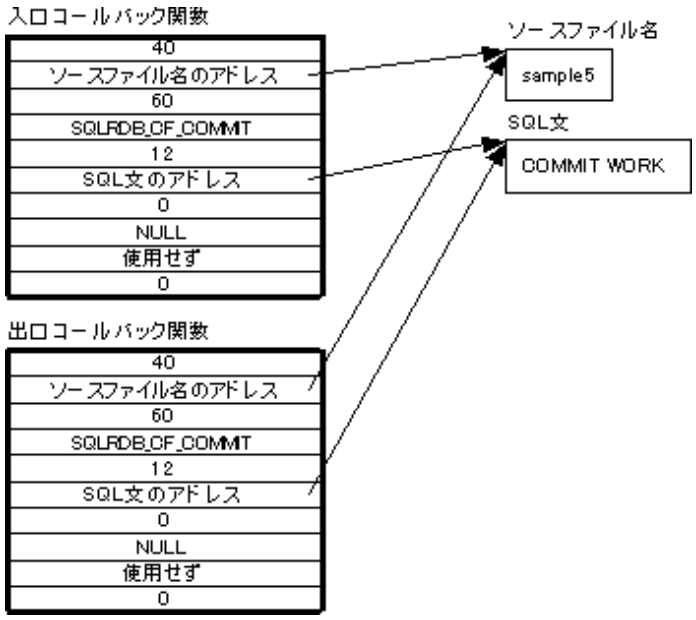
上記SQL文を実行した場合は、コールバック関数に通知されるSQL文情報は以下のようになります。



COMMIT文を実行した場合のデータ構造

```
EXEC SQL BEGIN DECLARE SECTION;
    char    SQLSTATE[6];
    char    SQLMSG[256];
EXEC SQL END DECLARE SECTION;
                    .
                    .
EXEC SQL COMMIT WORK;
                    .
                    .
```

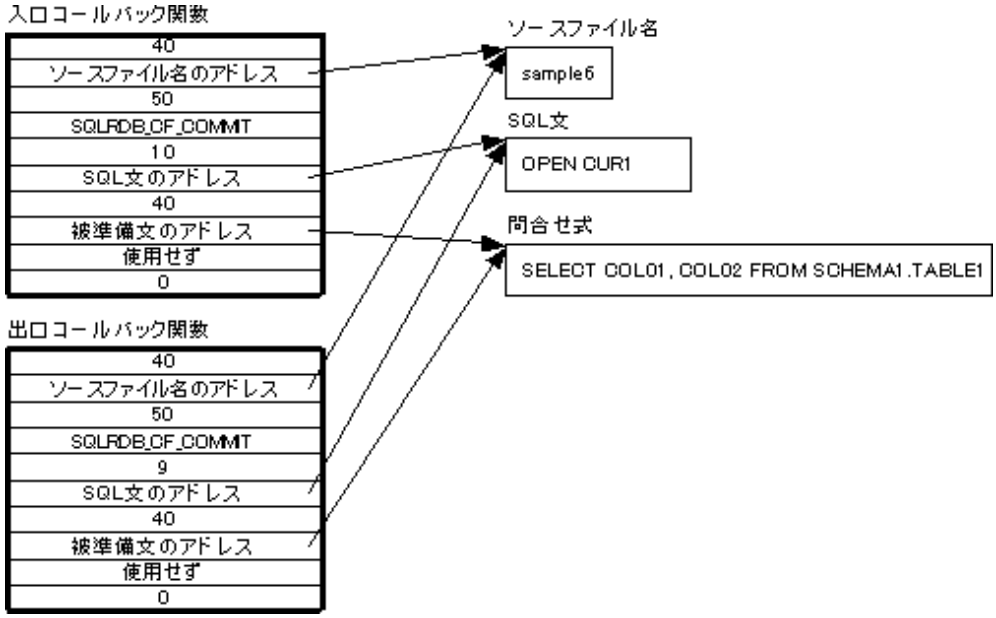
上記SQL文を実行した場合は、コールバック関数に通知されるSQL文情報は以下のようになります。



OPEN文を実行した場合のデータ構造

```
EXEC SQL BEGIN DECLARE SECTION;
    char    SQLSTATE[6];
    char    SQLMSG[256];
EXEC SQL END DECLARE SECTION;
                    .
                    .
EXEC SQL DECLARE CUR1 CURSOR FOR
        SELECT COL01, COL02 FROM SCHEMA1.TABLE1;
                    .
                    .
EXEC SQL OPEN CUR1;
                    .
                    .
```

上記SQL文を実行した場合は、コールバック関数に通知されるSQL文情報は以下のようになります。



付録A　SQLSTATE値

アプリケーション中で、SQL文の実行結果は状態変数のSQLSTATEに通知されます。したがって、アプリケーションは、SQL文を実行するたびにSQLSTATEをチェックして、Symfoware/RDBの処理結果を確認しながら処理を進めることができます。

SQLSTATE

状態変数のSQLSTATEは、5桁の文字列型の変数です。SQL文を実行した結果、SQLSTATEに設定される値を“状態コード”と呼びます。状態コードの先頭の2桁を“例外コード”と呼び、うしろの3桁を“副例外コード”と呼びます。

SQL文の実行結果に対するコードの値を以下に示します。

参照

クラスタシステム利用時のクラスタ切替え時に返却されるSQLSTATEについては、“クラスタ導入運用ガイド”の“SQLSTATE値”を参照してください。

表A.1 SQLSTATE値

例外コード	例外条件	副例外コード	副例外コードの意味
00	正常終了	000	なし
01	警告	005	なし
02	データなし	000	なし
07	動的SQLエラー	000	なし
		001	USING句が被準備文内の動的パラメタと対応しない
		002	USING句が被準備文内の選択リスト(相手指定)と一致しない
		003	被準備文が＜カーソル指定＞である
		004	動的パラメタ指定に対するUSING句が指定されていない
		005	被準備文が＜カーソル指定＞でない
		006	埋め込みホスト変数属性違反(制限されたデータ型属性を指定)
		007	(相手指定に対する)結果USINGが指定されていない
		008	SQL記述子域の個数不当
		009	SQL記述子域のインデックス不当
08	コネクション違反	000	コネクションの利用方法の誤り
		002	使用中のコネクション
		003	コネクションが存在しない
		004	SQLサーバがSQLコネクションの接続を拒絶した
09	被トリガ動作例外	000	なし
0A	未サポート機能	000	なし
0P	ロール名不当	000	なし
10	XQuery式エラー	000	なし

例外コード	例外条件	副例外コード	副例外コードの意味
21	基数違反	000	単一行SELECT文の実行で2行以上が検索された、または、比較述語に直接含まれる副問合せの結果が2行以上検索された
22	データ例外	001	文字データの右側を切り捨てた(トランケート)
		002	NULL値を設定する標識変数の指定がない
		003	扱える数値の範囲を超えた
		005	代入規則違反
		008	扱える日時の範囲を超えた
		00M	正しくないXML形式のデータ
		011	部分文字列の指定に誤り
		012	除数0による除算
		015	扱える時間隔の範囲を超えた
		018	型変換を行う文字列の形式に誤り
		019	エスケープ文字の不当
		024	正しくない文字型ホスト変数(入力ホスト変数の定義サイズ内にNULL文字が存在していない)
		027	TRIM文字の指定に誤り
		701	正しくない可変長文字列(長さ部の値が0未満、または、定義サイズを超えている)
		702	DSI範囲外アクセス
		703	正しくない数値(固定小数点の符号部の値が不当)
		704	正しくない文字(各国語文字列の値が不当)
23	NOT NULL制約違反、一意性制約違反	000	なし
24	カーソル状態不当	000	なし
25	トランザクション状態不当	000	なし
26	SQL文識別子不当	000	なし
28	パスワードが不当	000	なし
2B	依存する権限記述子がまだ存在する(注1)	000	なし
2E	コネクション名が不当	000	なし
2F	ルーチン呼び出しエラー	000	なし
		003	禁止されているSQL文の実行
33	記述子名不当	000	なし
34	カーソル名不当	000	なし
37	SQL文構文エラー	000	なし

例外コード	例外条件	副例外コード	副例外コードの意味
3C	あいまいなカーソル名	000	なし
3D	カタログ名不当	000	なし
3F	スキーマ名不当	000	なし
40	トランザクションロールバック(注2)	001	トランザクション直列化に失敗または実時間オーバ
		003	文終了不明(処理時間オーバまたは強制終了)
		701	テンポラリログ領域不足
		702	COMMIT実行時エラー
		703	アクセス中のノードダウン
		704	アーカイブログファイル不足
		705	Connection Managerの強制終了
		801	定義系処理異常
42	アクセス規則違反	000	権限なし
44	ビューに対する更新でWITH CHECK OPTIONに違反	000	なし
60	利用者送信例外	XXX(注3)	利用者が送信した例外
61	SQL文実行キャンセル	000	なし
70	定量制限	000	なし
71	環境不整合または運用矛盾	000	OSの制限値を超えた
		001	Symfoware/RDBが未起動
		002	サーバシステムが未起動
		003	ダウン中ノードへのアクセス要求
		004	外部プロセスダウン
		010	環境変数または、動作環境ファイルの形式が不当
		020	Symfoware/RDBの制限値を超えた
		030	Symfoware/RDB環境矛盾
		031	コード変換ライブラリロード失敗
		210	DSIが未定義またはDSIが未選択
		220	DSIが未フォーマット
		300	カーソル矛盾
		400	利用規定(注4)
		410	アクセス禁止
		500	データベース未定義またはDSO未定義
		510	検索パス未創成
		600	セションロック不可状態
		700	トランザクションの実行多重度オーバ

例外コード	例外条件	副例外コード	副例外コードの意味
		710	リカバリ要状態
		720	データベースまたは、RDB ディクショナリの矛盾
		730	定義系処理異常
		740	分割転送での矛盾
		750	一時表の制限値オーバ
		800	DSI自動容量拡張での異常
		900	待機系のRDBシステムでのSQL文の実行
		A00	順序の利用順違反
72	外部記憶領域などの不足	000	DSIの格納領域不足
		010	データベーススペース不足
		100	ディレクトリ領域不足
		200	作業用テーブル領域不足
73	メモリの領域不足	100	ローカルメモリのプール数が設定上限値を超えた
		110	ローカルメモリの不足
		200	共用メモリのプール数が設定上限値を超えた
		210	共用メモリの不足
		300	共用バッファの不足
74	外部記憶領域への入出力異常	000	データベーススペースの入出力異常
		010	データベーススペースのアクセス権なし
		100	ディレクトリ領域への入出力異常
		110	ディレクトリ領域のアクセス権なし
		200	作業用ファイルの入出力異常
		210	作業用ファイルのアクセス権なし
		400	テンポラリログ領域の入出力異常
		410	テンポラリログ領域のアクセス権なし
		500	スナップファイルへの入出力異常
		600	その他のファイルの入出力異常
		610	アクセス権なし
		720	RDBディクショナリの入出力異常
75	コネクション環境異常	000	コネクション実行環境の異常
76	システム環境異常	000	システム異常
77	他製品の動作異常	000	PowerAIM/TJNLで異常検出
		010	TJNLが未起動
		100	Accela BizSearchで異常検出

注1) 権限記述子は、以下の項目からなります。

・ 権限受領者が所有する資源(表、ビューなど)

・ 権限受領者が他の認可識別子に付与した権限

注2)トランザクション制御用のSQL文(COMMIT文、ROLLBACK文)の実行で、状態コード 40003 または40705が通知された場合、トランザクションの状態が不定となります。直前に行っていた操作からトランザクションの状態を確認してください。また、JDBCドライバを使用するアプリケーションでは、SQL文の実行ごとに自動的にコミット処理を行う自動コミット機能があります。この場合、SQL文の実行で、状態コード 40003または40705が通知される場合があります。直前に行っていた操作から状態を確認してください。

注3) SIGNAL文の副例外コードで設定する0～9およびA～Zまでの任意の文字を3桁で表現します。

注4) 処理対象のDSI に対するrdbfmtコマンドを実行中の場合にも設定されることがあります。

参照

ODBCに対応したパソコンツールとの連携で対応するコードが表に存在しない場合には、“MSDNライブラリ Visual Studio” で参照してください。

SQLSTATE値とトランザクションの関係

SQL文を実行した結果により、トランザクションは開始された状態が継続される場合と、終了してアプリケーションに復帰する場合があります。また、SQL文でエラーが発生した場合のトランザクションの扱いを、動作環境ファイルのパラメタTRAN_SPECで指定することができます。

参照

動作環境ファイルについては、“アプリケーション開発ガイド(共通編)”の“動作環境ファイルのパラメタ一覧”を参照してください。

これらの関係を以下に示します。

表A.2 SQLSTATE値とトランザクションの関係

TRAN_SPEC（注1）	SQLSTATEの例外コード	復帰時のトランザクションの状態	アプリケーションでの対処
NONE	00、02	トランザクションは開始された状態が継続しています。	そのまま次の処理に進みます。
	40	トランザクションはロールバックされ終了しています。(注2)	無効となったトランザクションを再実行(リトライ)するかどうかを決定します。
	上記以外	トランザクションは開始された状態が継続しています。該当のSQL 文によるデータベースの更新は、実行直前の状態に戻されています。ただし、それまでに獲得されたデータベース資源のロックとカーソルの位置はそのまま保持されます。	エラーの原因によりトランザクションを継続させるか終了させるかを決定します。
TRANSACTION_ROLLBACK	00、02	トランザクションは開始された状態が継続しています。	そのまま次の処理に進みます。
	40	トランザクションはロールバックされ終了しています。(注2)	無効となったトランザクションを再実行(リトライ)するかどうかを決定します。

TRAN_SPEC（注1）	SQLSTATEの例外コード	復帰時のトランザクションの状態	アプリケーションでの対処
	上記以外	トランザクションはロールバックされ終了しています。	エラーの原因を取り除いて、トランザクションの処理を最初からやり直す必要があります。このためSQL 文の実行でエラーを起こさないように、アプリケーションで入力データのチェックなどを行います。

注1) 動作環境ファイルのパラメタ“TRAN_SPEC”に設定された値を示します。

注2) トランザクション制御用のSQL文（COMMIT文、ROLLBACK文）の実行で、状態コード 40003 または40705が通知された場合、トランザクションの状態が不定となります。直前に行っていた操作からトランザクションの状態を確認してください。また、JDBCドライバを使用するアプリケーションでは、SQL文の実行ごとに自動的にコミット処理を行う自動コミット機能があります。この場合、SQL文の実行で、状態コード 40003または40705が通知される場合があります。直前に行っていた操作から状態を確認してください。

被トリガSQL文の実行でエラーの場合の注意事項

アプリケーションの実行時に、被トリガSQL文の実行がエラーとなった場合、トリガ契機となったSQL文のSQLSTATEの値は、以下に示す値になります。

表A.3 トリガ契機となったSQL文のSQLSTATE

被トリガSQL文のSQLSTATE	トリガ契機となったSQL文のSQLSTATE	備考
4xxxx	4xxxx	SQLSTATEとエラーメッセージが、そのままトリガ契機となったSQL文の実行結果に引き継がれます。
7xxxx	7xxxx	上記と同じです。
その他	09000	トリガ契機となったSQL文のSQLSTATEは、09000となり、そのエラーメッセージは、被トリガのSQL文のエラーメッセージが埋め込まれます。

付録B　キーワード一覧

“2.1.3トークン”で説明されているキーワードの一覧を示します。なお、*印、**印、***印および****印の付いていないものはSQL92のキーワード、*印はSQL95で追加されたキーワード、**印はSQL96で追加されたキーワードです。***印はSQL2000で追加されたキーワード、****印はSQL2007で追加されたキーワードです。

【A】
ABSOLUTE
ACTION
ADD
ADMIN(***)
AFTER(**)
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG

【B】
BEFORE(**)
BEGIN
BETWEEN
BINARY
BIT
BIT_LENGTH
BLOB
BOTH
BY

【C】
CALL(*)
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHARACTER
CHARACTER_LENGTH
CHAR_LENGTH
CHECK
CLOSE
COALESCE
COLLATE
COLLATION
COLUMN
COMMIT
CONDITION(***)

【G】
GET
GLOBAL
GO
GOTO
GRANT
GROUP

【H】
HANDLER(***)
HAVING
HOUR

【I】
IDENTITY
IF(*)
IMMEDIATE
IN
INDICATOR
INITIALLY
INNER
INOUT(*)
INPUT
INSENSITIVE
INSERT
INSTEAD(**)
INT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION

【J】
JOIN

【K】
KEY

【L】
LANGUAGE
LARGE
LAST
LEADING
LEAVE(*)
LEFT
LEVEL
LIKE
LOCAL

RESIGNAL(***)
RESTRICT
RETURN(***)
RETURNS(***)
REVOKE
RIGHT
ROLE(***)
ROLLBACK
ROUTINE(*)
ROW(**)
ROW_ID(**)
ROWNUM(****)
ROWS

【S】
SCHEMA
SCROLL
SECOND
SECTION
SELECT
SEQUENCE
SESSION
SESSION_USER
SET
SIGNAL(***)
SIZE
SMALLINT
SOME
SPACE
SPECIFIC(***)
SQL
SQLCODE
SQLERROR
SQLEXCEPTION(***)
SQLSTATE
SQLWARNING(***)
SUBSTRING
SUM
SYSTEM_USER

【T】
TABLE
TEMPORARY
THEN
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO

CONNECT
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR

【D】
DATE
DAY
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFAULT
DEFERRABLE
DEFERRED
DELETE
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DISCONNECT
DISTINCT
DO(*)
DOMAIN
DOUBLE
DROP

【E】
EACH(**)
ELSE
ELSEIF(*)
END
END-EXEC
ESCAPE
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXIT(***)
EXTERNAL
EXTRACT

【F】
FALSE

LOOP(*)
LOWER

【M】
MATCH
MAX
MIN
MINUTE
MODULE
MONTH

【N】
NAMES
NATIONAL
NATURAL
NCHAR
NEW(**)
NEW_TABLE(**)
NEXT
NO
NONE(***)
NOT
NULL
NULLIF
NUMERIC

【O】
OBJECT
OCTET_LENGTH
OF
OLD(**)
OLD_TABLE(**)
ON
ONLY
OPEN
OPTION
OR
ORDER
OUT(*)
OUTER
OUTPUT
OVERLAPS

【P】
PAD
PARALLEL(**)
PARAMETER(***)
PARTIAL
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC

TRAILING
TRANSACTION
TRANSLATE
TRANSLATION
TRIGGER(**)
TRIM
TRUE

【U】
UNDO(***)
UNION
UNIQUE
UNKNOWN
UNTIL(*)
UPDATE
UPPER
USAGE
USER
USING

【V】
VALUE
VALUES
VARCHAR
VARYING
VIEW

【W】
WHEN
WHENEVER
WHERE
WHILE(*)
WITH
WORK
WRITE

【Y】
YEAR

【Z】
ZONE

FETCH	【R】	
FIRST	READ	
FLOAT	REAL	
FOR	REDO(***)	
FOREIGN	REFERENCES	
FOUND	REFERENCING(**)	
FROM	RELATIVE	
FULL	REPEAT(*)	
FUNCTION(*)		

付録C　定量制限

Symfoware Serverの定量制限について説明します。

C.1 Symfoware/RDBの定量制限

Symfoware/RDBの定量制限を以下に示します。

表C.1 定量制限

項目			定量制限
データベース数			無制限
名標の長さ	英字	データベース名	36文字以内 (注1)
		スキーマ名	36文字以内 (注1) (注2)
		表名	36文字以内 (注1) (注2)
		列名	36文字以内 (注1)
		インデックス名	36文字以内 (注1) (注2)
		DSO名	36文字以内 (注1)
		DSI名	36文字以内 (注1)
		データベーススペース名	36文字以内 (注1)
		カーソル名	36文字以内 (注1)
		相関名	36文字以内 (注1)
		ルーチン名	36文字以内 (注1)
		スコープ名	36文字以内 (注1)
		トリガ名	36文字以内 (注1)
		SQL文識別子	36文字以内 (注1)
		SQL変数名	36文字以内 (注1)
		文ラベル	36文字以内 (注1)
		パラメタ名	36文字以内 (注1)
		条件名	36文字以内 (注1)
		認可識別子	18文字以内
		記述子名	36文字以内
		コネクション名	36文字以内
		SQLサーバ名	36文字以内
		順序名	36文字以内 (注1)
		ロール名	36文字以内 (注1)
		ロググループ名	18文字以内
		共用バッファ識別子	18文字以内
	日本語	データベース名	18文字以内
		スキーマ名	18文字以内 (注2)
		表名	18文字以内 (注2)
		列名	18文字以内
		インデックス名	18文字以内 (注2)

項目			定量制限
		DSO名	18文字以内
		DSI名	18文字以内
		データベーススペース名	18文字以内
		カーソル名	18文字以内
		相関名	18文字以内
		ルーチン名	18文字以内
		スコープ名	18文字以内
		トリガ名	18文字以内
		SQL文識別子	18文字以内
		SQL変数名	18文字以内
		文ラベル	18文字以内
		パラメタ名	18文字以内
		条件名	18文字以内
		認可識別子	9文字以内
		記述子名	18文字以内
		コネクション名	18文字以内
		SQLサーバ名	18文字以内
		順序名	18文字以内
		ロール名	18文字以内
		ロググループ名	18文字以内または12文字以内 (注3)
		共用バッファ識別子	18文字以内
スキーマ要素	実表数／スキーマ		無制限
	列数／表		32,000個
	表の行長	SEQUENTIAL構造(BLOB型を含む場合)	2,000メガバイト
		SEQUENTIAL構造(BLOB型を含まない場合)またはRANDOM構造	32,000バイト
		OBJECT構造	2,047メガバイト＋32,000バイト
	一意性制約を構成する列数		64列
	一意性制約を構成するデータ長		1,000バイト
	既定値オプションのデータ長		3,000バイト (注4)
	容量／表		ペタバイトオーダ
	注釈定義		256バイト
	トリガ定義文の探索条件の文字列長		15,000バイト
	トリガ定義文の被トリガSQL文の文字列長		15,000バイト
	ビュー定義文の問合せ指定の文字列長		32,000バイト (注5)
	プロシジャルーチン定義の文字列長		無制限 (注5)
格納構造要素	実表	DSO数／スキーマ	無制限
		DSI数	無制限

項目				定量制限
		DSI数／DSO		無制限
		分割条件を指定する固定長文字列型の1つあたりのデータ長	CHARACTER	254文字
			NATIONAL CHARACTER	127文字
		キー構成列数		64列
		クラスタのキーの長さ		1,000バイト
		ページ長(BLOBが31キロバイト以内)		1、2、4、8、16、32キロバイト
		ページ長(BLOBが32キロバイト以上)		32キロバイト
		容量／DSI		2テラバイト未満
		DSIの静的容量拡張／拡張量		2,097,151キロバイト
		DSIの動的容量拡張／1回の拡張量		2,097,151キロバイト
		DSIの動的容量拡張／拡張契機		2テラバイト未満
		DSI領域のアラームポイント		2テラバイト未満
	インデックス	DSO数		無制限
		DSI数		無制限
		DSI数／DSO		無制限
		キー構成列数		64列
		キーの長さ		1,000バイト
		ページ長		1、2、4、8、16、32キロバイト
		容量／DSI		2テラバイト未満
		DSIの静的容量拡張／拡張量		2,097,151キロバイト
		DSIの動的容量拡張／1回の拡張量		2,097,151キロバイト
		DSIの動的容量拡張／拡張契機		2テラバイト未満
		DSI領域のアラームポイント		2テラバイト未満
扱えるデータ種と属性(C)	文字		固定長	32,000バイト
			可変長	32,000バイト
	日本語文字		固定長	32,000バイト
			可変長	32,000バイト
	数字	外部10進表現		18桁
		内部2進表現	2バイト	-32,768 から 32,767
			4バイト	-2,147,483,648 から 2,147,483,647
			8バイト	-999,999,999,999,999,999 から 999,999,999,999,999,999
		内部10進表現		18桁
		浮動小数点表現	4バイト	3.4E +/- 38 (7 桁)
			8バイト	1.7E +/- 308 (15 桁)
	BLOB			2ギガバイト

項目				定量制限
扱えるデータ種と属性(COBOL)	文字		固定長	32,000文字
			可変長	32,000文字
	日本語文字(UTF-32形式以外)		固定長	16,000文字
			可変長	16,000文字
	日本語文字(UTF-32形式)		固定長	8,000文字
			可変長	8,000文字
	数字	外部10進表現		18桁
		内部2進表現	2バイト	4桁
			4バイト	9桁
		内部10進表現		18桁
		浮動小数点表現	4バイト	仮数桁数6以下
			8バイト	仮数桁数7以上
	BLOB			2ギガバイト
プロシジャルーチン定義	SQLパラメタ宣言リストに指定できるSQLパラメタ宣言の数			32,767個
	局所宣言リストに指定できる局所宣言の数			32,767個
	SQL変数宣言に指定できるSQL変数名の数			32,767個
	ハンドラ宣言に指定できる条件値の数			32,767個
	SQL文リストに指定できるSQL手続き文の数			32,767個
GRANT文/REVOKE文	権限受領者に指定できる認可識別子またはロール名の数			32,767個
	ロール付与またはロール剥奪に指定できるロール名の数			32,767個
データ操作文	アプリケーションの1プロセスあたりの最大コネクション数		ローカルアクセス	512
			リモートアクセス	無制限
	選択リストに指定可能な値式数			32,000個
	選択リストに指定可能な列の合計長			2ギガバイト－1(注6)
	表式のFROM句に指定可能な表参照の数			128個
	GROUP BY句に指定可能な値式数			32,000個
	ORDER BY句に指定可能な値式数			32,000個
	1つの問合せ式に指定できる問合せ項の数			64個 (注7)
	1つの参照表に指定できる結合表の入れ子の数			128個 (注8)
	1つの値式に指定できる関数または演算式の数			128個 (注9)
	1つの行値構成子に指定できる値式の数			127個
	1つのファンクションルーチン指定の引数リストに指定可能な値式の合計長			2ギガバイト-1 (注10)
	1つのセションで同時に処理可能なカーソル数			無制限
	1つのセションで同時に接続できるコネクション			16
	1つのSQL文の文字列長		静的SQL	無制限
			動的SQL	32キロバイト(注11)

項目		定量制限
	1つのSQL文に指定できる動的パラメタ指定の数	32,767個
	IN述語の限定値リストに指定できる値指定の数	32,767個
	USING句に指定できる引数の数	32,767個
	結合表に指定できる結合演算項の数 (注12)	128個
	表宣言に指定できる表名の数	32,767個
	COALESCEに指定できる値式の数	32,767個
	CASE式に指定できる単純WHEN句または探索WHEN句の数	32,767個
	作業用テーブルおよび作業用ソート領域に格納可能な列の合計長 (注13)	64キロバイト (注14)
	作業用テーブルおよび作業用ソート領域に格納可能なレコード数	2147483647件
ASSIST指定	ASSIST要素に指定できる制約ASSISTの数	128個
	ジョインASSISTに指定できるASSIST表要素の数	128個
RDBシステムで管理可能なデータベース規模		128ペタバイト
ロググループ数 (注15)		100個 (注16)
データベーススペースの割付けサイズ		1キロバイト以上2テラバイト未満(注17)
RDBディクショナリの割付けサイズ		2テラバイト未満(注17)
DSIの割付けサイズ		2テラバイト未満
RDBディレクトリファイルの割付けサイズ		4テラバイト未満(注17)
テンポラリログファイル	BIログ域サイズ	16ギガバイト未満
	AIログ域サイズ	16ギガバイト未満
	インダウトログファイルのサイズ	16ギガバイト未満
アーカイブログファイルのサイズ		制限なし(注17)
アンロードファイル(rdbunl、rdbunlsql、rdbunlx、rdbunladt)のサイズ		制限なし
入力データファイル(rdbsloader、rdbsaloader、rdbsuloader)のサイズ		制限なし
入力データファイル(rdbupt)のサイズ		2ギガバイト未満
DSI退避ファイル(rdbdmp)のサイズ		制限なし
RDBディクショナリ退避ファイル(rdbdmpdic)のサイズ		制限なし
アーカイブログ退避ファイル(rdblog -B)のサイズ		制限なし
アプリケーションの1プロセスあたりの最大コネクション数	ローカルアクセス	512
	リモートアクセス	制限なし
ソート作業域のサイズ (rdbgcdsi、rdbsaloader、rdbsloader、rdbsuloader、rdbrcv、rdbups)		制限なし
1つのデータベーススペース内に作成可能なサブエクステント数 (注18)		262,138個
パフォーマンスデータ格納ファイルのサイズ		制限なし
コアファイルのサイズ		制限なし

注1) Symfoware/RDBと組み合わせて使用する製品で18文字を超える識別子を扱えない場合は、システム用の動作環境ファイルでNAME_SIZE_CHECK=YESを指定することにより、19文字以上の資源名を定義できないようにチェックすることができます。19文字以上の資源名の定義に対しては、エラーが返却されます。

注2) 格納構造定義を行わない場合は8文字以内です。ただし、システム用の動作環境ファイルでDEFAULT_DSI_NAME=CODEを指定した場合、または、デフォルトデータベーススペースに表またはインデックスを作成する場合は、定量制限どおりとなります。

注3) Symfoware/RDBの文字コード系がEUCコードまたはシフトJISコードの場合は18文字以内となります。Symfoware/RDBの文字コード系がUNICODEの場合は12文字以内となります。

注4) 既定値オプションを文字列定義で指定する場合、データ中の引用符は2バイトとして数えます。

注5) 30,000バイトを超える問合せ指定、または30,000バイトを超えるプロシジャルーチンを定義した場合、システム表のテーブルRDBII_SYSTEM.RDBII_DESCRIPTIONの、列DESC_VALUEには、30,000バイトまでしか格納されません。

注6) 以下の見積り式で2ギガバイトー1まで指定可能です。

```
列の合計長 ＝選択リストに指定した列の合計長
             ＋（選択リストに指定したBLOB列数 × 8）
             ＋ NULL値を許可する列数（表定義でNOT NULLを指定していない列数）
             ＋（可変長の列数 × 2）
```

なお、選択リストに定数を指定した場合、返却データのデータ型は指定する定数に依存します。
定数を指定した場合のデータ型については、“2.1.2 定数”を参照してください。

注7) 1つの問合せ式として、問合せ項UNION(またはUNION ALL)を複数指定した場合のUNION(またはUNION ALL)の数の定量制限値です。

以下にこの定量制限値でのUNION(またはUNION ALL)の数え方の例を示します。

例

以下の例の場合、UNIONの数は2個です。

```
SELECT 倉庫番号 FROM 在庫管理.在庫表
    UNION SELECT 取引先 FROM 在庫管理.発注表
    UNION SELECT 会社番号 FROM 在庫管理.会社表
```

上記例の数え方については、以下のとおりです。

```
SELECT 倉庫番号 FROM 在庫管理.在庫表
  UNION SELECT 取引先 FROM 在庫管理.発注表      1個目
  UNION SELECT 会社番号 FROM 在庫管理.会社表    2個目
```

注8) FROM句に指定できる1つの表参照として、結合表を入れ子で指定した場合の結合表の数の定量制限値です。

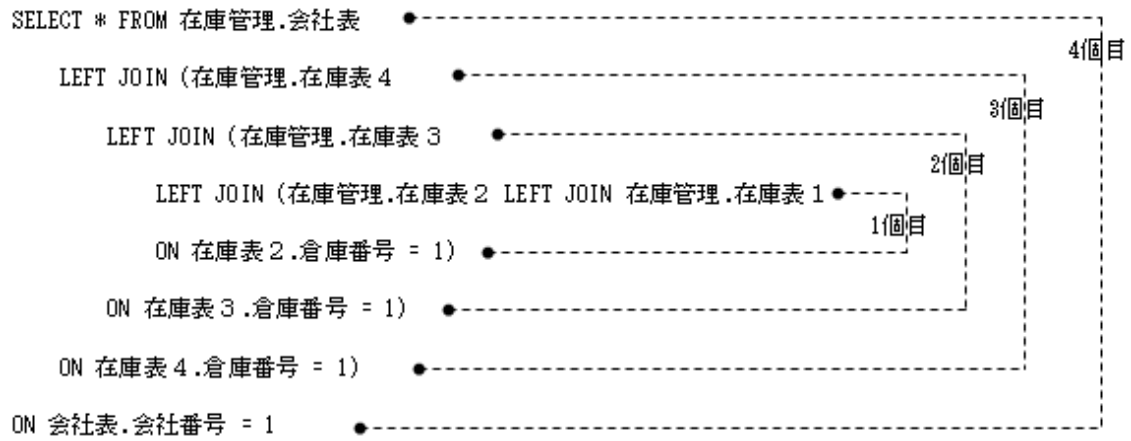
以下にこの定量制限値での結合表の数え方の例を示します。

例

以下の例の場合、結合表の数は4個です。

```
SELECT * FROM 在庫管理.会社表
    LEFT JOIN（在庫管理.在庫表４
        LEFT JOIN（在庫管理.在庫表３
            LEFT JOIN（在庫管理.在庫表２ LEFT JOIN 在庫管理.在庫表１
            ON 在庫表２.倉庫番号 = 1）
        ON 在庫表３.倉庫番号 = 1）
    ON 在庫表４.倉庫番号 = 1）
ON 会社表.会社番号 = 1
```

上記例の数え方については、以下のとおりです。



注9) 1つの値式に指定した関数または演算式の合計数の定量制限値です。

関数の場合、1つの値式として、関数を入れ子で指定した場合の入れ子の数です。

演算式の場合、1つの値式として、演算式を複数指定した場合の演算式の数です。

なお対象となる値式は、以下のとおりです。

- 列指定
- 数値関数
- データ列値関数
- 日時値関数
- ファンクションルーチン指定
- CASE式
- CAST指定
- 数値式(単項演算子または二項演算子)
- データ列値値式(連結演算子)
- 日時値式(日時演算)
- 時間隔値式(時間隔演算)

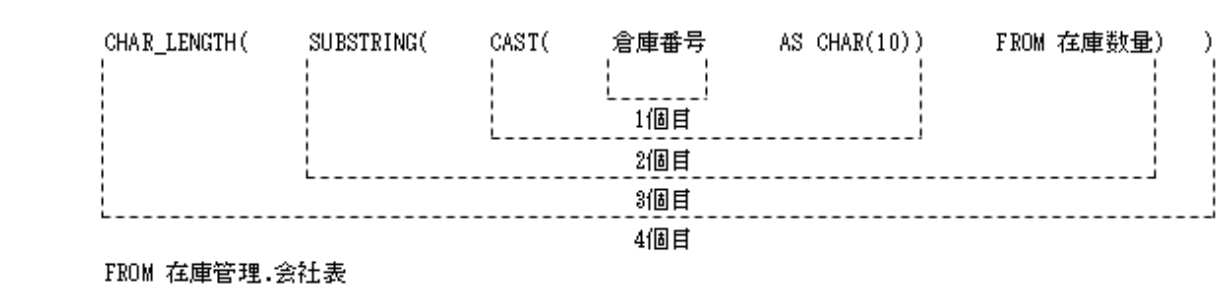
以下にこの定量制限値での関数および演算式の数え方の例を示します。

例1:関数の場合

以下の例の場合、関数の数は4個です。

```
SELECT
    CHAR_LENGTH(SUBSTRING(CAST(倉庫番号 AS CHAR(10)) FROM 在庫数量))
    FROM 在庫管理.会社表
```

上記例の関数の数え方については、以下のとおりです。

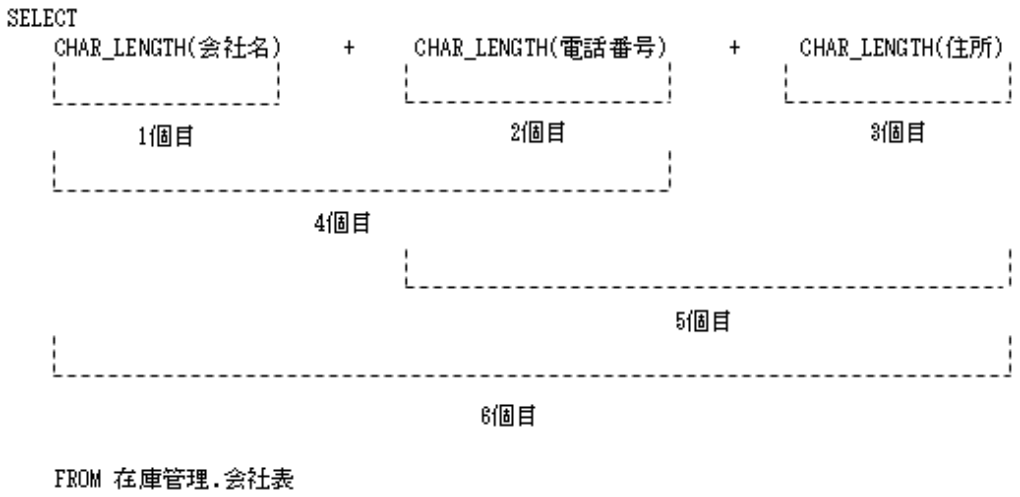


例2:演算式の場合

以下の例の場合、演算式の数は6個です。

```
SELECT
    CHAR_LENGTH(会社名)+CHAR_LENGTH(電話番号)+CHAR_LENGTH(住所)
    FROM 在庫管理.会社表
```

上記例の演算式の数え方については、以下のとおりです。



注10) 以下の見積り式で2ギガバイト-1まで指定可能です。

```
引数リストの合計長 ＝引数リストに指定した値式の合計長
                  ＋ 戻りデータ型の長さ
                  ＋ 2MB

引数リストに指定した値式の合計長 :
SQL文に複数のファンクションルーチンを指定している場合、各ファンクションルーチンについて引数リストに指定した値式の合計長を求め、その合計長のうち、最長の値を指定してください。
引数リストに指定した値式がBLOB型を含む、かつ、値式の合計長が128K以上の場合、BLOB型の値式については実データ長で算出してください。BLOB型以外の値式については定義長で算出してください。

戻りデータ型の長さ:
SQL文に複数のファンクションルーチンを指定している場合、各ファンクションルーチンの戻りデータ型の定義長のうち、最長の値を指定してください。
```

注11) 埋込みSQL連携の場合の定量制限値です。Java連携、ODBC連携および.NET Framework連携の場合は、無制限です。

注12) FROM句に指定できる1つの結合表として、結合演算項の数の定量制限値です。

以下にこの定量制限値での結合演算項の数え方の例を示します。

例

以下の例の場合、結合演算項の数は4個です。

```
SELECT * FROM 在庫管理.会社表
    LEFT JOIN 在庫管理.在庫表４ ON 会社表.会社番号 = 1
    LEFT JOIN 在庫管理.在庫表３ ON 在庫表４.倉庫番号 = 1
    LEFT JOIN 在庫管理.在庫表２ ON 在庫表３.倉庫番号 = 1
    LEFT JOIN 在庫管理.在庫表１ ON 在庫表２.倉庫番号 = 1
```

上記例の数え方については、以下のとおりです。

```
SELECT * FROM 在庫管理.会社表
    LEFT JOIN 在庫管理.在庫表４ ON 会社表.会社番号 = 1    ●---------- 1個目
    LEFT JOIN 在庫管理.在庫表３ ON 在庫表４.倉庫番号 = 1  ●---------- 2個目
    LEFT JOIN 在庫管理.在庫表２ ON 在庫表３.倉庫番号 = 1  ●---------- 3個目
    LEFT JOIN 在庫管理.在庫表１ ON 在庫表２.倉庫番号 = 1  ●---------- 4個目
```

注13) 選択リストに記述した項目に列が含まれる場合、列のデータ長は定義長です。ただし、32キロバイトを超えるBLOB列については、格納構造がSEQENTIAL構造の場合は24バイト、OBJECT構造の場合は18バイトです。

以下のいずれかに該当する場合、標識変数の1バイトを含めてください。

- NOT NULL制約を指定していない列
- LEFTを指定した結合表(OUTER JOIN)の右側の表の列
- RIGHTを指定した結合表(OUTER JOIN)の左側の表の列

データ長は、以下で確認できます。

- アクセスプラン:エレメント情報の詳細情報 insert record length
- SQL性能情報 :実行エレメント情報の詳細情報 record length(byte)

注14) カーソル宣言のSCROLLの指定の有無により、しきい値は異なります。カーソル宣言にSCROLLを指定した場合のしきい値は65516バイト、カーソル宣言にSCROLLを指定しない場合のしきい値は65519バイトです。

注15) ロググループ数とは、システムロググループ数(1個)とユーザロググループ数の合計値です。ユーザロググループには、監査ログ用ロググループも含まれます。

注16) ロードシェア運用の場合は、ロードシェアシステム全体で最大1600個のロググループが定義できます。

注17) Solaris環境でローデバイスに配置する場合、1テラバイト未満です。

注18) rdbprtコマンドでデータベーススペースの定義情報を出力した時に、“DSI information”に表示されるDSI情報の数が、データベーススペース内のサブエクステント数です。

コンパイル時の定量制限

1つのコンパイル単位における定量制限を以下に示します。

表C.2 コンパイル時の定量制限

項目	定量制限	備考
ホストラベル識別子	32個	
日本語名標	－	ホスト変数名など、C言語では使用不可

C.2 Textアダプタの定量制限(Solarisの場合)

Textアダプタの定量制限を以下に示します。

表C.3 Textアダプタの定量制限

項目		定量制限
Textアダプタ定義	Textアダプタ定義名長	32バイト
	Textアダプタ定義数	無制限
	Columnタグ数	64個
	フィールド名長	15バイト

項目		定量制限
TEXT検索式	TEXT検索式	8192バイト

C.3 XMLアダプタの定量制限

XMLアダプタの定量制限を以下に示します。

表C.4 XMLアダプタの定量制限

項目			定量制限
名標の長さ	英字	XMLグループ名	36文字以内
	日本語	XMLグループ名	18文字以内
格納構造要素	XML構造のインデックス	DSO数	無制限
		DSI数	無制限
		DSOの中に存在するDSI数	無制限
		キー構成列数	1個(BLOB型)
		ページ長	1、2、4、8、16、32キロバイト
		実表のDSOに対するXML構造のインデックスのDSO数	1個
XMLグループ定義	XMLグループ定義数		無制限
	パス定義数		1024個
	パスの段数		50段
	パスの文字列長		500バイト
	型指定	STRING	1000バイト
		NUMBER	18桁
	選択条件	列名リスト	64個

付録D　SQL文の省略値に関する注意事項

SQL文の省略値に関する注意事項について説明します。

D.1　SQL文の一般規則に関するチューニング

SQL文の一般規則を、クライアント用の動作環境ファイルの実行パラメタで変更することができます。チューニング可能な一般規則を以下に示します。

表D.1 チューニング可能な一般規則

項目	クライアント用の動作環境ファイルの実行パラメタ
代入処理でオーバフローが発生した場合	記述形式 CAL_ERROR = ({ REJECT | NULL })
	実行パラメタの意味 代入処理でオーバフローが発生した場合の処理を指定します。省略した場合は、REJECTが指定されたとみなします。
	パラメタの意味 REJECT: 例外事象とします。 NULL: 代入結果をNULLとします。
動的SQLのSQL記述子の情報	記述形式 DESCRIPTOR_SPEC = (WITH MAX省略値)
	実行パラメタの意味 動的SQLのSQL記述子の情報を指定します。
	パラメタの意味 WITH MAX省略値: ALLOCATE DESCRIPTOR文でWITH MAXを省略した場合の値を指定します。指定できる値は、1～32767までの値です。省略した場合は、100が指定されたとみなされます。

D.2　名前に関する注意事項

SQLで使用する名前の省略値および注意事項について説明します。

データベース名の省略値

動的SQL文

動的SQLによって実行される被準備文の対象となるデータベース名は省略することができます。データベース名の省略値の優先順位は以下のとおりです。

1. SET CATALOG文で設定しているデータベース名
2. CONNECT文に指定したデータベース名
3. クライアント用の動作環境ファイルのDEFAULT_CONNECTIONに指定したデータベース名
4. アプリケーションのコンパイル時オプションで指定したデータベース名

スキーマ名の省略値

動的SQL文

表名、ルーチン名およびトリガ名を修飾するスキーマ名は省略することができます。スキーマ名の省略値の優先順位は以下のとおりです。

1. SET SCHEMA文で設定しているスキーマ名

2. CONNECT文のユーザ指定に指定した認可識別子
3. クライアント用の動作環境ファイルのDEFAULT_CONNECTIONに指定した認可識別子
4. アプリケーションを実行しているユーザID

データ操作文

表名を修飾するスキーマ名は省略することができます。スキーマ名の省略値の優先順位は以下のとおりです。

1. 表宣言文により宣言しているスキーマ名
2. アプリケーションのコンパイル時のログイン名
3. SQL利用者セションの現行利用者認可識別子
4. ルーチン実行時の利用者認可識別子

rdbddlexコマンドで実行するSQL文

表名を修飾するスキーマ名は省略することができます。スキーマ名の省略値は以下のとおりです。

- rdbddlexコマンドを実行するユーザID

利用者名について

アプリケーションでデータベースをアクセスする場合の認可識別子を決定する優先順位は以下のとおりです。

1. SET SESSION AUTHORIZATION文で設定している認可識別子
2. CONNECT文のユーザ指定に指定した認可識別子
3. クライアント用の動作環境ファイルのDEFAULT_CONNECTIONに指定した認可識別子
4. アプリケーションを実行しているユーザID

参照

クライアント用の動作環境ファイルについては、“アプリケーション開発ガイド(埋込みSQL編)”を参照してください。

D.3 格納領域指定時の格納構造

表定義またはインデックス定義で格納領域指定を指定する、または、デフォルトデータベーススペースに表を作成して、格納構造定義を行わない場合の格納構造について説明します。

格納構造定義を行わない場合、表およびインデックスのページ長や割付け量などの初期値は、システム用の動作環境ファイルのパラメタ“DEFAULT_TABLE_SIZE”および“DEFAULT_INDEX_SIZE”で設定できます。このパラメタの指定を省略した場合、以下のようになります。

注意

データベース簡単運用の場合、システム用の動作環境ファイルのパラメタ“DEFAULT_TABLE_SIZE”および“DEFAULT_INDEX_SIZE”は指定できません。

参照

システム用の動作環境ファイルについての詳細は、“セットアップガイド”を参照してください。

表定義

表の格納構造

表の格納構造は以下のようになります。

格納構造:

SEQUENTIAL構造

データ部のページ長:

32K

データ部の割付け量:

30720K

拡張量

動作環境ファイルのDEFAULT_TABLE_SIZEで指定した値になります。

DEFAULT_TABLE_SIZEの指定を省略した場合は、10240Kになります。

ORDER(n)の指定:

ORDER(1)

PRECEDENCE(n)の指定:

データベーススペースに表を定義する場合

なし

デフォルトデータベーススペースに表を定義する場合

PRECEDENCE(1)

ただし、表定義が以下の場合、表の格納構造はOBJECT構造となります。なお、以下の表定義の場合でも、動作環境ファイルのDEFAULT_DSI_TYPE でSEQUENTIALを指定している場合またはデフォルトデータベーススペースに表を定義する場合は、格納構造はSEQUENTIAL構造となります。

表定義

- 表の最後に1つだけBLOB型でサイズが32Kバイト以上の列を指定している。
- BLOB型以外の列は固定長属性である。
- BLOB型の列にNOT NULL制約を指定している。

格納構造

OBJECT構造

データ部のページ長

32K

データ部の割付け量

32768K

拡張量

動作環境ファイルのDEFAULT_OBJECT_TABLE_SIZEで指定した値になります。

DEFAULT_OBJECT_TABLE_SIZEの指定を省略した場合は、32768Kになります。

表のDSO名およびDSI名

表のDSO名および表のDSI名は、表定義時のスキーマ名と表名を組み合せて命名します。DSO名とDSI名は同じ名前になります。

```
“＃ スキーマ名 {＃ | @} 表名”
 (1)    (2)   (3) (4)  (5)
```

(1) 固定プレフィックス

(2) CREATE TABLE文で指定したスキーマ名

(3) 表名が文字列の場合

(4) 表名が各国語文字列の場合

(5) CERATE TABLE文で指定した表名

ただし、スキーマ名および表名のいずれか、または両方が各国語文字列の場合には、DSO名およびDSI名は、全体が各国語文字列となります。

システム用の動作環境ファイルでDEFAULT_DSI_NAME=CODEを指定した場合またはデフォルトデータベーススペースに表を定義する場合、表のDSO名およびDSI名は、システムで採番する10桁の数字を使用して命名します。

インデックス定義

インデックスの格納構造

インデックスの格納構造は以下のようになります。

格納構造:

BTREE構造

ベース部のページ長:

8K

インデックス部のページ長:

8K

ベース部の割付け量:

30720K

縮退指定:

データベーススペースに作成した表にインデックスを定義する場合

あり

デフォルトデータベーススペースに作成した表にインデックスを定義する場合

なし

インデックス部の割付け量:

10240K

拡張量

動作環境ファイルのDEFAULT_INDEX_SIZEで指定した値になります。

DEFAULT_INDEX_SIZEの指定を省略した場合は、10240Kになります。

付録E　SQL文の使用範囲

以下に、Symfoware/RDBがコンパイル・実行時にサポートするSQL文の使用範囲を示します。

データベース簡単運用の場合には、使用可能なSQL文の範囲が異なります。データベース簡単運用ガイドを参照し、使用可能なSQL文の範囲を確認してください。

表E.1 アプリケーションに記述できるSQL文の範囲

分類		対象OS		コンパイル時のサポート範囲	実行時のサポート範囲
			クライアント	Solaris Linux Windows	Solaris Linux Windows
			サーバ	ー	Solaris Linux Windows
共通要素	データ型(BLOB)			○	○
	データ型(DATE、TIME、TIMESTAMP)			○	○
	データ型(INTERVAL)			○	○
	データ型（上記以外)			○	○
	値指定と相手指定			○	○
	項目参照			○	○
	列指定			○	○
	関数			○	○
	集合関数指定			○	○
	数値関数			○	○
	データ列値関数			○	○
	日時値関数			○	○
	XMLQUERY関数			○	○
	CAST指定			○	○
	行識別子(ROW_ID)			○	○
	ROWNUM			○	○
	値式			○	○
	行値構成子			○	○
	数値式			○	○
	データ列値式			○	○
	日時値式			○	○
	時間隔値式			○	○
	CASE式			○	○
	述語			○	○
	比較述語			○	○
	BETWEEN述語			○	○
	IN述語			○	○
	LIKE述語			○	○

分類		対象OS		コンパイル時のサポート範囲	実行時のサポート範囲
			クライアント	Solaris Linux Windows	Solaris Linux Windows
			サーバ	ー	Solaris Linux Windows
	NULL述語			○	○
	限定述語			○	○
	EXISTS述語			○	○
	XMLEXISTS述語			○	○
	探索条件			○	○
	表式			○	○
	FROM句			○	○
	WHERE句			○	○
	GROUP BY句			○	○
	HAVING句			○	○
	問合せ指定			○	○
	問合せ式			○	○
	副問合せ			○	○
	ASSIST指定			○	○
システム制御文	SET SYSTEM PARAMETER文			○（注1）	○（注1）
利用者制御文	利用者定義文(CREATE USER)			○（注1）	○（注1）
	利用者変更文(ALTER USER)			○（注1）	○（注1）
	利用者削除文(DROP USER)			○（注1）	○（注1）
	SET USER PASSWORD文			○	○（注2）
データベース定義文	データベース定義 (CREATE DATABASE)			○（注1）	○（注1）
	データベーススペース定義 (CREATE DBSPACE)			○（注1）	○（注1）
データベース操作文	データベース削除文 (DROP DATABASE)			○（注1）	○（注1）
	データベーススペース削除文 (DROP DBSPACE)			○（注1）	○（注1）

分類		対象OS		コンパイル時のサポート範囲	実行時のサポート範囲
			クライアント	Solaris Linux Windows	Solaris Linux Windows
			サーバ	ー	Solaris Linux Windows
スキーマ定義文	スキーマ定義(CREATE SCHEMA)			○(注1)	○(注1)
	順序定義(CREATE SEQUENCE)			○(注1)	○(注1)
	表定義(CREATE TABLE)			○(注1)	○(注1)
	ビュー定義(CREATE VIEW)			○(注1)	○(注1)
	プロシジャルーチン定義 (CREATE PROCEDURE)			○(注1)	○(注1)
	ファンクションルーチン定義 (CREATE FUNCTION)			○(注1)	○(注1)
	インデックス定義(CREATE INDEX)			○(注1)	○(注1)
	トリガ定義(CREATE TRIGGER)			○(注1)	○(注1)
スキーマ操作文	スキーマ削除文(DROP SCHEMA)			○(注1)	○(注1)
	順序削除文(DROP SEQUENCE)			○(注1)	○(注1)
	表削除文(DROP TABLE)			○(注1)	○(注1)
	ビュー削除文(DROP VIEW)			○(注1)	○(注1)
	表定義変更文(ALTER TABLE)			○(注1)	○(注1)
	プロシジャルーチン削除文 (DROP PROCEDURE)			○(注1)	○(注1)
	ファンクションルーチン削除文 (DROP FUNCTION)			○(注1)	○(注1)
	インデックス削除文(DROP INDEX)			○(注1)	○(注1)
	トリガ削除文(DROP TRIGGER)			○(注1)	○(注1)
	表交換文(SWAP TABLE)			○(注1)	○(注1)
格納構造定義文	表のDSO定義文(CREATE DSO)			○(注1)	○(注1)
	インデックスのDSO定義文(CREATE DSO)			○(注1)	○(注1)
	表のDSI定義文(CREATE DSI)			○(注1)	○(注1)
	インデックスのDSI定義文 (CREATE DSI)			○(注1)	○(注1)
	スコープ定義文(CREATE SCOPE)			○(注1)	○(注1)
格納構造操作文	DSO削除文(DROP DSO)			○(注1)	○(注1)
	DSI削除文(DROP DSI)			○(注1)	○(注1)
	DSI変更文(ALTER DSI)			○(注1)	○(注1)
	スコープ削除文(DROP SCOPE)			○(注1)	○(注1)
	スコープ適用文(APPLY SCOPE)			○(注1)	○(注1)
	スコープ解除文(RELEASE SCOPE)			○(注1)	○(注1)
	最適化情報設定文(SET STATISTICS)			○(注1)	○(注1)

分類			対象OS	コンパイル時のサポート範囲	実行時のサポート範囲
			クライアント	Solaris Linux Windows	Solaris Linux Windows
			サーバ	ー	Solaris Linux Windows
	最適化情報表示文(PRINT STATISTICS)			○(注1)	○(注1)
アクセス制御文	ロール定義文 (CREATE ROLE)			○(注1)	○(注1)
	ロール削除文 (DROP ROLE)			○(注1)	○(注1)
	SET ROLE文			○	○(注2)
	GRANT文			○(注1)	○(注1)
	REVOKE文			○(注1)	○(注1)
データ操作文	単一行SELECT文			○	○
		並列指定		○	○
		占有モード指定		○	○
		最大件数指定		○	○
	DELETE文:探索			○	○
		占有モード指定		○	○
	INSERT文			○	○
		占有モード指定		○	○
	UPDATE文:探索			○	○
		占有モード指定		○	○
	カーソル宣言			○	○
		更新可能性句		○	○
		SCROLL		○	○
		並列指定		○	○
		占有モード指定		○	○
		カーソルモード指定		○	○
		最大件数指定		○	○
	OPEN文			○	○
	CLOSE文			○	○
	FETCH文			○	○
	DELETE文:位置づけ			○	○
	UPDATE文:位置づけ			○	○
	表宣言			○	○
資源操作文	RELEASE TABLE文			○	○(注2)
トランザクション管理文	SET TRANSACTION文			○	○
	COMMIT文			○	○

分類		対象OS		コンパイル時のサポート範囲	実行時のサポート範囲
		クライアント		Solaris Linux Windows	Solaris Linux Windows
		サーバ		ー	Solaris Linux Windows
	ROLLBACK文			○	○
コネクション管理文	CONNECT文			○	○
	SET CONNECTION文			○	○
	DISCONNECT文			○	○
セション管理文	SET CATALOG文			○	○
	SET SCHEMA文			○	○
	SET SESSION AUTHORIZATION文			○	○
動的ＳＱＬ文	USING句			○	○
	ALLOCATE DESCRIPTOR文			○	○
	DEALLOCATE DESCRIPTOR文			○	○
	DESCRIPTOR 取得文			○	○
	DESCRIPTOR 設定文			○	○
	PREPARE文			○	○
	DEALLOCATE PREPARE文			○	○
	DESCRIBE文			○	○
	EXECUTE文			○	○
	EXECUTE IMMEDIATE文			○	○
	動的カーソル宣言			○	○
	動的OPEN文			○	○
	動的FETCH文			○	○
	動的CLOSE文			○	○
	動的DELETE文:位置づけ			○	○
	動的UPDATE文:位置づけ			○	○
	準備可能動的DELETE文:位置づけ			○	○
	準備可能動的UPDATE文:位置づけ			○	○
プロシジャ実行	CALL文			○	○
埋込みＳＱＬ	埋込み例外宣言			○	○
	INCLUDE文			○	○

○:サポートする

注1)動的SQLの準備可能文の場合のみ指定できます。

注2)動的SQL文では利用できません。

付録F　定義文の実行時間に関する注意事項

定義文の実行時間に関する注意事項を説明します。

・ 下記の表が示す資源種別Aの同一資源を指定した資源種別Bの定義文(定義削除文)を大量に実行する場合に、資源種別Bの定義文(定義削除文)を実行する時間が、資源種別Bの資源の数の増加により2次曲線的に増加する場合があります。資源種別Bの定義文を資源種別Aの複数資源に分割するように対処してください。

資源種別A	資源種別B	資源種別Bの定義文	資源種別Bの定義削除文
データベーススペース	DSI	CREATE DSI文	DROP DSI文
権限受領者(ロール)	権限	GRANT文	REVOKE

例

同じデータベーススペースを指定したCREATE DSI文を大量に実行した場合に、CREATE DSI文を実行する時間が、DSIの数の増加により2次曲線的に増加します。CREATE DSI文の定義文を資源種別データベーススペースの複数資源に分割するように対処してください。

・ 下記の表が示す資源種別Aの同一資源を指定した資源種別Bの定義文(定義削除文)を大量に実行する場合に、資源種別Bの定義文(定義削除文) を実行する時間が、資源種別Bの資源の数の増加により2次曲線的に増加する場合があります。複数の資源種別Bの資源を1つの定義文(定義削除文)で一括処理するように対処してください。

資源種別A	資源種別B	資源種別Bの定義文	資源種別Bの定義削除文
権限	権限受領者(認可識別子)	GRANT文	REVOKE文
スコープ	スコープを適用する利用者(認可識別子)	APPLY SCOPE文	RELEASE SCOPE文
スコープを適用する利用者(認可識別子)	スコープ	APPLY SCOPE文	RELEASE SCOPE文
権限	権限受領者(ロール)	GRANT文	REVOKE

例1

同じ権限を指定したGRANT文を大量に実行して大量のロールに権限付与する場合に、GRANT文を実行する時間が、ロールの数の増加により2次曲線的に増加します。複数のGRANT文を1つのGRANT文で一括処理するように対処してください。

例2

同じスコープを指定したAPPLY SCOPE文を大量に実行して大量の利用者にスコープを適用する場合に、APPLY SCOPE文を実行する時間が、利用者の数の増加により2次曲線的に増加します。複数のAPPLY SCOPE文を1つのAPPLY SCOPE文で一括処理するように対処してください。

・ 下記の表が示す資源種別Aの同一資源を指定した資源種別Bの定義文を大量に実行する場合に、資源種別Bの定義文・変更文(あるいは定義削除文)を実行する時間が、資源種別Bの資源の数の増加により2次曲線的に増加する場合があります。定義文・変更文(あるいは定義削除文)に十分な実行時間を用意してください。

資源種別A	資源種別B	資源種別Bの 定義文・変更文	資源種別Bの 定義削除文
表	列定義	ALTER TABLE文 (ADD 列名)	ALTER TABLE文 (DROP 列名)
表	DSO	CREATE DSO文	DROP DSO文
表	ビュー	CREATE VIEW文	DROP VIEW文
表	トリガ	CREATE TRIGGER 文	DROP TRIGGER文
DSO	DSI	対象外（注1）	DROP DSI文
DSI	割付け先	ALTER DSI文(ADD ALLOCATE)	対象外（注2）
プロシジャルーチン	トリガ	CREATE TRIGGER 文	DROP TRIGGER文
ファンクションルーチン	ビュー	CREATE VIEW文	DROP VIEW文
順序	表	CREATE TABLE文	DROP TABLE文
順序	プロシジャルーチン	CREATE PROCEDURE文	DROP PROCEDURE文
順序	トリガ	CREATE TRIGGER 文	DROP TRIGGER文

注1) 同一DSOを指定したCREATE DSI文の実行時間はDSIの数の増加により、2次曲線的に増加しません。

注2) ALTER DSI文(ADD ALLOCATE)に対応する定義削除文がありません。

例

同一表を指定したCREATE VIEW文を大量に実行する場合に、CREATE VIEW文(あるいはDROP VIEW文)を実行する時間が、ビューの数の増加により2次曲線的に増加する場合があります。CREATE VIEW文(あるいはDROP VIEW文)に十分な実行時間を用意してください。

付録G　SQL規格に対するSymfoware Serverの準拠性

Symfoware Serverは、SQL:2003規格に準拠しています。

SQL:2003規格は、以下から構成されています。

- ISO/IEC 9075-1 Framework (SQL/Framework)
- ISO/IEC 9075-2 Foundation (SQL/Foundation)
- ISO/IEC 9075-3 Call Level Interface (SQL/CLI)
- ISO/IEC 9075-4 Persistent Stored Modules (SQL/PSM)
- ISO/IEC 9075-9 Management of External Data (SQL/MED)
- ISO/IEC 9075-10 Object Language Bindings (SQL/OLB)
- ISO/IEC 9075-11 Information and Definition Schemas (SQL/Schemata)
- ISO/IEC 9075-13 Routines and Types using the Java Language (SQL/JRT)
- ISO/IEC 9075-14 XML-related specifications (SQL/XML)

本付録では、SQL:2003規格のコア機能とPart 2“Foundation(SQL/Foundation)”のオプション機能に対する、Symfoware Serverの準拠性を説明します。

ここで説明するコア機能とオプション機能とは、それぞれ以下を指します。

- コア機能
  SQL:2003規格のPart 2“Foundation(SQL/Foundation)”およびPart 11“Schemata(SQL/Schemata)”にコア機能として定義されている機能です。
- オプション機能
  SQL:2003規格のPart 2“Foundation(SQL/Foundation)”に定義されているコア機能以外の機能です。

G.1　コア機能に対する準拠性

SQL:2003規格のコア機能に対するSymfowareの準拠性を示します。準拠性は、完全または部分的に準拠しているもののみ掲載しています。

詳細は機能識別子ごとに後述していますので、そちらを参照してください。

機能識別子	規格	
	和文表記	英文表記
E011	数値データ型	Numeric data types
E021	文字データ型	Character data types
E031	識別子	Identifiers
E051	基本的な問合せ指定	Basic query specification
E061	基本述部および検索条件	Basic predicates and search conditions
E071	基本的な問合せ式	Basic query expressions
E081	基本的な権限	Basic Privileges
E091	集合関数	Set functions
E101	基本的なデータ操作	Basic data manipulation
E111	単一行のSELECT文	Single row SELECT statement
E121	基本的なカーソルのサポート	Basic cursor support
E131	NULL値のサポート(値のかわりのNULL)	Null value support (nulls in lieu of values)

機能識別子	規格	
	和文表記	英文表記
E141	基本的な整合性制約	Basic integrity constraints
E151	トランザクションのサポート	Transaction support
E152	基本的なSET TRANSACTION文	Basic SET TRANSACTION statement
E161	先頭に2つの負の符号(-)を付けたSQL文のコメント	SQL comments using leading double minus
E171	SQLSTATEのサポート	SQLSTATE support
F021	基本的な情報スキーマ	Basic information schema
F031	基本的なスキーマ操作	Basic schema manipulation
F041	基本的な結合表	Basic joined table
F051	基本的な日付および時刻	Basic date and time
F131	グループ操作	Grouped operations
F201	CASTファンクション	CAST function NOTE 364 - This means the support of CAST, where relevant, among all supported data types.
F221	明示的なデフォルト	Explicit defaults
F261	CASE式	CASE expression
F311	スキーマ定義文	Schema definition statement
F481	拡張NULL述語	Expanded NULL predicate
T321	基本的なSQL起動ルーチン	Basic SQL-invoked routines

E011:数値データ型(Numeric data types)

No	規格	準拠性	備考
01	INTEGERおよびSMALLINTデータ型	○	
02	REAL、DOUBLE PRECISION、FLOATデータ型	○	
03	DECIMALおよびNUMERICデータ型	○	
04	算術演算子	○	
05	数値比較	○	
06	数値データ型間の暗黙キャスト	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E021:文字データ型(Character data types)

No	規格	準拠性	備考
01	CHARACTERデータ型	○	
02	CHARACTER VARYINGデータ型	○	
03	文字列定数	○	
04	CHARACTER_LENGTH関数	△	USING指定はできません。
05	OCTET_LENGTH関数	○	

No	規格	準拠性	備考
06	SUBSTRING関数	△	USING指定はできません。 また、<regular expression substring function>は未サポートです。
07	文字の連結	△	BLOB値式は未サポートです。
08	UPPERおよびLOWER関数	○	
09	TRIM関数	○	
10	文字列型間の暗黙的キャスト	○	
11	POSITION関数	△	<blob position expression>は未サポートです。
12	文字の比較	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E031:識別子(Identifiers)

No	規格	準拠性	備考
01	制限付き識別子	○	
02	小文字の識別子	○	
03	末尾のアンダースコア	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E051:基本的な問合せ指定(Basic query specification)

No	規格	準拠性	備考
01	SELECT DISTINCT	○	
02	GROUP BY句	○	
04	選択リストにない句を持つGROUP BYが可能	○	
05	選択リスト項目の名前変更が可能	○	
06	HAVING句	○	
07	選択リスト内の修飾付き*	○	
08	FROM句内の相関名	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E061:基本述部および検索条件(Basic predicates and search conditions)

No	規格	準拠性	備考
01	比較述語	○	
02	BETWEEN述部	○	
03	値リスト指定のIN述語	○	
04	LIKE述語	○	
05	ESCAPE句付きのLIKE述語	○	
06	NULL述語	○	

No	規格	準拠性	備考
07	限定述語	○	
08	EXISTS述部	○	
09	比較述語内の副問合せ	○	
11	IN述語内の副問合せ	○	
12	限定述語内の副問合せ	○	
13	相関副問合せ	○	
14	探索条件	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E071:基本的な問合せ式(Basic query expressions)

No	規格	準拠性	備考
01	UNION DISTINCTテーブル演算子	○	
02	UNION ALL テーブル演算子	○	
05	テーブル演算子によって結合された列の型は同一データ型である必要はない	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E081:基本的な権限(Basic Privileges)

No	規格	準拠性	備考
01	テーブルレベルのSELECT権限	○	
02	DELETE権限	○	
03	テーブルレベルのINSERT権限	○	
04	テーブルレベルのUPDATE権限	○	
08	WITH GRANT OPTION	○	
10	EXECUTE権限	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E091:集合関数(Set functions)

No	規格	準拠性	備考
01	AVG	○	
02	COUNT	○	
03	MAX	○	
04	MIN	○	
05	SUM	○	
06	ALL修飾子	○	
07	DISTINCT修飾子	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E101：基本的なデータ操作（Basic data manipulation）

No	規格	準拠性	備考
01	INSERT文	○	
03	探索UPDATE文	○	
04	探索DELETE文	○	

○：完全準拠　△：部分準拠または代替となる同等機能あり

E111：単一行のSELECT文（Single row SELECT statement）

単一行のSELECT文は完全準拠しています。

E121：基本的なカーソルのサポート（Basic cursor support）

No	規格	準拠性	備考
01	カーソル宣言	○	
02	ORDER BYの列は選択リスト内に不要	○	
03	ORDER BY句内の値式	○	
04	OPEN文	○	
06	位置づけUPDATE文	○	
07	位置づけDELETE文	○	
08	CLOSE文	○	
10	FETCH文（暗黙NEXT指定）	○	

○：完全準拠　△：部分準拠または代替となる同等機能あり

E131：NULL値のサポート（値のかわりのNULL）（Null value support (nulls in lieu of values)）

NULL値のサポートは完全準拠しています。

E141：基本的な整合性制約（Basic integrity constraints）

No	規格	準拠性	備考
01	NOT NULL制約	○	
02	NOT NULL列で構成されるUNIQUE制約	○	
03	主キー制約	○	
07	列の既定値	○	
08	PRIMARY KEY列へのNOT NULLの仮定	○	

○：完全準拠　△：部分準拠または代替となる同等機能あり

E151：トランザクションのサポート（Transaction support）

No	規格	準拠性	備考
01	COMMIT文	○	
02	ROLLBACK文	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E152:基本的なSET TRANSACTION文(Basic SET TRANSACTION statement)

No	規格	準拠性	備考
01	SET TRANSACTION 文: ISOLATION LEVEL SERIALIZABLE句	○	
02	SET TRANSACTION 文: READ ONLY and READ WRITE句	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

E161:先頭に2つの負の符号(-)を付けたSQL文のコメント(SQL comments using leading double minus)

先頭に2つの負の符号(-)を付けたSQL文のコメントは完全準拠しています。

E171:SQLSTATEのサポート(SQLSTATE support)

SQLSTATEのサポートは完全準拠しています。

F021:基本的な情報スキーマ(Basic information schema)

No	規格	準拠性	備考
01	COLUMNSビュー	△	以下は未サポートです。 ・ CHARACTER_SET_CATALOG ・ CHARACTER_SET_SCHEMA ・ CHARACTER_SET_NAME ・ COLLATION_CATALOG ・ COLLATION_SCHEMA ・ COLLATION_NAME ・ UDT_CATALOG ・ UDT_SCHEMA ・ UDT_NAME ・ SCOPE_CATALOG ・ SCOPE_SCHEMA ・ SCOPE_NAME ・ MAXIMUM_CARDINALITY ・ DTD_IDENTIFIER ・ IS_SELF_REFERENCING ・ IS_IDENTITY

No	規格	準拠性	備考
			・ IDENTITY_GENERATION ・ IDENTITY_START ・ IDENTITY_INCREMENT ・ IDENTITY_MAXIMUM ・ IDENTITY_MINIMUM ・ IDENTITY_CYCLE ・ IS_GENERATED ・ GENERATION_EXPRESSION ・ IS_UPDATABLE
02	TABLESビュー	△	以下は未サポートです。 ・ SELF_REFERENCING_COLUMN_NAME ・ REFERENCE_GENERATION ・ USER_DEFINED_TYPE_CATALOG ・ USER_DEFINED_TYPE_SCHEMA ・ USER_DEFINED_TYPE_NAME ・ IS_INSERTABLE_INTO ・ IS_TYPED ・ COMMIT_ACTION
03	VIEWSビュー	△	TABLESとRDBII_DESCRIPTIONで使用可能です。 以下はRDBII_DESCRIPTIONを使用します。 ・ VIEW_DEFINITION 以下は未サポートです。 ・ CHECK_OPTION ・ IS_UPDATABLE ・ INSERTABLE_INTO ただし、CHECK_OPTIONとIS_UPDATABLEは、RDBII_TABLEを使用することで使用可能となります。
04	TABLE_CONSTRAINTSビュー	△	以下はテーブルコードで表現しています。 ・ TABLE_CATALOG ・ TABLE_SCHEMA ・ TABLE_NAME 以下は未サポートです。 ・ IS_DEFERRABLE ・ INITIALLY_DEFERRED

○:完全準拠 △:部分準拠または代替となる同等機能あり

F031:基本的なスキーマ操作(Basic schema manipulation)

No	規格	準拠性	備考
01	永続実表のCREATE TABLE文	○	
02	CREATE VIEW文	○	
03	GRANT文	○	
04	ALTER TABLE文: ADD COLUMN句	○	
13	DROP TABLE文: RESTRICT句	○	
16	DROP VIEW文: RESTRICT句	○	
19	REVOKE文: RESTRICT句	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

F041:基本的な結合表(Basic joined table)

No	規格	準拠性	備考
01	内部結合(INNERキーワードは不要)	○	
03	LEFT OUTER JOIN	○	
04	RIGHT OUTER JOIN	○	
05	入れ子にできる外部結合	○	
07	左または右外部結合の内部テーブルは内部結合内でも使用可能	○	
08	全ての比較演算子のサポート(=以外も)	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

F051:基本的な日付および時刻(Basic date and time)

No	規格	準拠性	備考
01	DATEデータ型(日付定数のサポートを含む)	○	
02	0以上の小数秒精度を持つTIMEデータ型(時刻定数のサポートを含む)	○	
03	0と6の秒精度を持つTIMESTAMPデータ型(時刻印定数のサポートを含む)	○	
04	DATE、TIME、TIMESTAMPデータ型に対する比較述部	△	以下の比較は未サポートです。 ・ DATEとTIMESTAMP ・ TIMEとTIMESTAMP ただし、CASTを使用することで対応可能です。
05	日付時刻型と文字列型間の明示的なキャスト	○	
06	CURRENT_DATE	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

F131:グループ操作(Grouped operations)

No	規格	準拠性	備考
01	グループ化されたビューを使用する問合せにおけるWHERE、GROUP BY、HAVING句のサポート	○	
02	グループ化されたビューを使用する問合せにおける複数テーブルのサポート	○	
03	グループ化されたビューを使用する問合せにおける集合関数のサポート	○	
04	GROUP BY、HAVING句、グループ化されたビューを持つ副問合せ	○	
05	GROUP BY、HAVING句、グループ化されたビューを持つ単一行SELECT	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

F201:CASTファンクション(CAST function NOTE 364 - This means the support of CAST, where relevant, among all supported data types.)

CASTファンクションは完全準拠しています。

F221:明示的なデフォルト(Explicit defaults)

明示的なデフォルトは完全準拠しています。

F261:CASE式(CASE expression)

No	規格	準拠性	備考
01	単純なCASE	○	
02	探索CASE	○	
03	NULLIF	○	
04	COALESCE	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

F311:スキーマ定義文(Schema definition statement)

No	規格	準拠性	備考
01	CREATE SCHEMA	○	
02	永続実表に対するCREATE TABLE	○	
03	CREATE VIEW	○	
04	CREATE VIEW: WITH CHECK OPTION	○	
05	GRANT文	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

F481:拡張NULL述語(Expanded NULL predicate)

拡張NULL述語は完全準拠しています。

T321：基本的なSQL起動ルーチン（Basic SQL-invoked routines）

No	規格	準拠性	備考
01	多重定義なしユーザ定義関数	○	
02	多重定義なしユーザ定義ストアドプロシージャ	○	
03	関数呼び出し	○	
04	CALL文	○	
06	ROUTINESビュー	△	
07	PARAMETERSビュー	△	

○：完全準拠　△：部分準拠または代替となる同等機能あり

G.2 オプション機能に対する準拠性

ここでは、SQL:2003規格のオプション機能に対するSymfowareの準拠性を示します。準拠性は、完全または部分的に準拠しているもののみ掲載しています。

詳細は機能識別子ごとに後述していますので、そちらを参照してください。

機能識別子	規格	
	和文表記	英文表記
B012	埋め込みC	Embedded C
B013	埋め込みCOBOL	Embedded COBOL
B021	直接SQL文	Direct SQL
B031	基本的な動的SQL	Basic dynamic SQL
B032	拡張動的SQL	Extended dynamic SQL
B033	型なし引数を持つ関数呼出し	Untyped SQL-invoked function arguments
B034	PREPARE文でのカーソル属性の指定	Dynamic specification of cursor attributes
B122	ルーチン言語C	Routine language C
B123	ルーチン言語COBOL	Routine language COBOL
F032	波及削除動作	CASCADE drop behavior
F033	列削除	ALTER TABLE statement: DROP COLUMN clause
F034	拡張REVOKE文	Extended REVOKE statement
F052	時間隔型および日時演算、時間隔演算	Intervals and datetime arithmetic
F111	SERIALIZABLE以外の分離レベル	Isolation levels other than SERIALIZABLE
F171	単一ユーザによる複数スキーマ所有	Multiple schemas per user
F222	INSERT文の挿入元DEFAULT VALUES	INSERT statement: DEFAULT VALUES clause
F281	LIKE述語拡張	LIKE enhancements
F321	現行認可識別子の参照	User authorization
F421	各国語文字	National character
F431	読み取り専用のスクロールカーソル	Read-only scrollable cursors

機能識別子	規格	
	和文表記	英文表記
F441	拡張集合関数のサポート	Extended set function support
F531	一時表	Temporary tables
F591	導出表	Derived tables
F771	コネクション管理	Connection management
T041	基本的なLOBデータ型サポート	Basic LOB data type support
T176	シーケンス生成子	Sequence generator support
T211	基本的なトリガー	Basic trigger capability
T325	被修飾SQLパラメタ参照	Qualified SQL parameter references
T331	基本的なロール	Basic roles
T332	拡張ロール	Extended roles
T501	拡張EXISTS述語	Enhanced EXISTS predicate

B012:埋め込みC(Embedded C)

埋め込みCは完全準拠しています。

B013:埋め込みCOBOL(Embedded COBOL)

埋め込みCOBOLは完全準拠しています。

B021:直接SQL文(Direct SQL)

直接SQL文は完全準拠しています。

B031:基本的な動的SQL(Basic dynamic SQL)

基本的な動的SQL文は完全準拠しています。

B032:拡張動的SQL(Extended dynamic SQL)

No	規格	準拠性	備考
01	入力記述文	○	

○:完全準拠　△:部分準拠または代替となる同等機能あり

B033:型なし引数を持つ関数呼出し(Untyped SQL-invoked function arguments)

型なし引数を持つ関数呼出しは完全準拠しています。

B034:PREPARE文でのカーソル属性の指定(Dynamic specification of cursor attributes)

PREPARE文でのカーソル属性の指定は完全準拠しています。

B122：ルーチン言語C（Routine language C）

ルーチン言語Cは完全準拠しています。

B123：ルーチン言語COBOL（Routine language COBOL）

ルーチン言語COBOLは完全準拠しています。

関連するコア機能

T321：基本的なSQL起動ルーチン

F032：波及削除動作（CASCADE drop behavior）

波及削除動作は完全準拠しています。

関連するコア機能

F031-13：DROP TABLE文:RESTRICT句

F031-16：DROP VIEW文:RESTRICT句

F031-19：REVOKE文:RESTRICT句

F033：列削除（ALTER TABLE statement: DROP COLUMN clause）

列削除は完全準拠しています。

関連するコア機能

F031-04：ALTER TABLE文: ADD COLUMN句

F034：拡張REVOKE文（Extended REVOKE statement）

No	規格	準拠性	備考
01	スキーマ要素の所有者以外によるREVOKE文	○	
02	REVOKE文のGRANT OPTION FOR句	○	
03	WITH GRANT OPTION付き権限を持つユーザからの権限を剥奪するREVOKE文	○	

○：完全準拠　△：部分準拠または代替となる同等機能あり

関連するコア機能

E081：基本的な権限

F052：時間隔型および日時演算、時間隔演算（Intervals and datetime arithmetic）

ローカルタイム、タイムゾーンは、未サポートです。

また、時間隔演算、日時演算はサポートしますが、年月と日時の演算は未サポートです。

関連するコア機能

F051:基本的な日付および時刻

F111:SERIALIZABLE以外の分離レベル(Isolation levels other than SERIALIZABLE)

No	規格	準拠性	備考
01	READ UNCOMMITTED	○	
02	READ COMMITTED	○	
03	REPEATABLE READ	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

関連するコア機能

E152:基本的なSET TRANSACTION文

F171:単一ユーザによる複数スキーマ所有(Multiple schemas per user)

単一ユーザによる複数スキーマ所有は完全準拠しています。

関連するコア機能

F311:スキーマ定義文

F222:INSERT文の挿入元DEFAULT VALUES(INSERT statement: DEFAULT VALUES clause)

INSERT文の挿入元DEFAULT VALUESは完全準拠しています。

F281:LIKE述語拡張(LIKE enhancements)

LIKE述語拡張は完全準拠しています。

関連するコア機能

E061-04:LIKE述語

F321:現行認可識別子の参照(User authorization)

現行認可識別子の参照は完全準拠しています。

F421:各国語文字(National character)

各国語文字は完全準拠しています。

関連するコア機能

E021-04:CHARACTER_LENGTH関数

E061-04:LIKE述語

F201:CASTファンクション

F431:読み取り専用のスクロールカーソル(Read-only scrollable cursors)

No	規格	準拠性	備考
01	明示的NEXT指定のFETCH	○	
02	FIRST指定のFETCH	○	
03	LAST指定のFETCH	○	
04	PRIOR指定のFETCH	○	
05	ABSOLUTE指定のFETCH	○	
06	RELATIVE指定のFETCH	○	

○:完全準拠 △:部分準拠または代替となる同等機能あり

F441:拡張集合関数のサポート(Extended set function support)

以下をサポートします。

- DISTINCTを指定しないCOUNT関数
- 外への参照列を含む一般集合関数引数へのDISTINCT指定
- 単一の列参照でない一般集合関数引数中での外への参照列の指定(ただし、値式に演算子は指定できません)

F531:一時表(Temporary tables)

一時表はサポートしますが、一時表宣言は、未サポートです。

F591:導出表(Derived tables)

導出表には、問い合わせ指定のみ指定できます。

F771:コネクション管理(Connection management)

コネクション管理は完全準拠しています。

T041:基本的なLOBデータ型サポート(Basic LOB data type support)

BLOB型のみサポートしています。

T176:シーケンス生成子(Sequence generator support)

後続値式(<next value expression>)は、<順序>として形式を変えてサポートしています。

シーケンス生成子削除文は、未サポートです。

T211:基本的なトリガー(Basic trigger capability)

以下の機能に準拠しますが、規格と記述形式が異なる部分があります。

また、被トリガSQL 文には、INSERT文とCALL文のみ指定可能です。他のSQL文についてはCALL文から呼び出すことで代替可能です。

No	規格	準拠性	備考
01	実表に対するUPDATE, INSERT, DELETEにより起動されるトリガ	○	
02	BEFOREトリガ	○	
03	AFTERトリガ	○	
04	FOR EACH ROWトリガ	○	
05	トリガ起動条件を指定できる機能	○	
07	トリガ権限	○	
08	同じ事象に対する複数トリガのカタログでの作成順実行	△	CALL文での指定により、複数のSQL文の操作が可能です。

○:完全準拠　△:部分準拠または代替となる同等機能あり

T325:被修飾SQLパラメタ参照(Qualified SQL parameter references)

被修飾SQLパラメタ参照は完全準拠しています。

関連するコア機能

T321:基本的なSQL起動ルーチン

T331:基本的なロール(Basic roles)

SET ROLEは、未サポートです。

T332:拡張ロール(Extended roles)

ロールを権限受領者とする指定のみサポートしています。

T501:拡張EXISTS述語(Enhanced EXISTS predicate)

拡張EXISTS述語は完全準拠しています。

関連するコア機能

E061-08:EXISTS述部

索　引

[D]

[E]

[F]

[さ]

[た]

[な]

[は]

[ま]

[や]

[ら]

[わ]