MySQL マニュアル
データベース基礎 for Windows, Linux
入門編

13.MySQL 管理運用

・MySQL データのバックアップ方法
・MySQL データのリストア方法
・MySQL データの設定ファイル

1.テーブルの種類、ファイルの構成

MySQL は、全部で8種類のテーブルを扱うことができます。この8種類のテーブルでは、データをやり取りするためのストレージエンジン（テーブルハンドラ）がそれぞれ異なります。用途によって使い分けますが、MySQL 4.1.8 では「MyISAM のみ使用」という設定にしない限り InnoDB がデフォルトになっており、その他のストレージエンジンを使用する場合は、テーブル作成時に指定します。以下、MySQL で使えるテーブル一覧です。

- MyISAM
- InnoDB
- MERGE
- MEMORY (HEAP)
- BDB (BerkeleyDB)
- ARCHIVE
- CSV
- ISAM

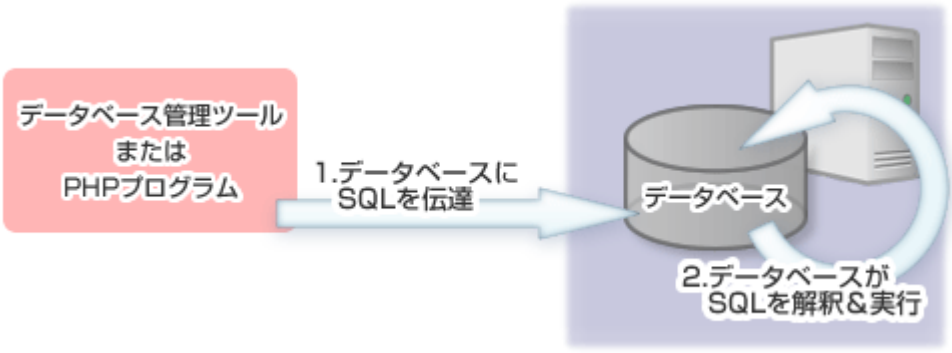
トランザクションなどの機能を使わないのであれば、MyISAM を使用するとよいでしょう。MyISAM は古いテーブルの型なので、今後サポートされなくなる予定です。

また、関係データベース（リレーショナルデータベース）の基礎として、表の行のことをレコード、表の列のことをカラムと呼びます。関係データベースでは、クエリと呼ばれる命令を出して結果を取得します。関係データベースでは、複数の表と連結する機能があり、連結するためにつかう参照元のキーを主キー、参照先のキーを外部キーと呼びます。複数の表を連結する場合、この主キー、外部キーをもとに連結を行います。

テーブル　カラム

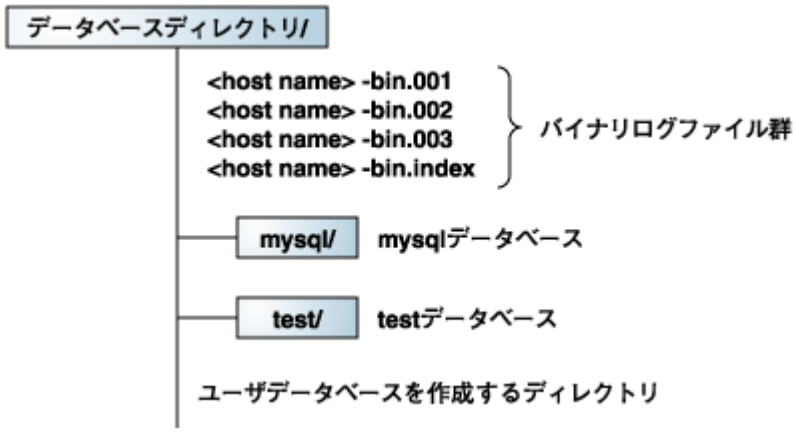
店舗No	住所	店名	予算
1	吉祥寺	うさぎや	2000
2	高円寺	パンケーキーズ	2000
3	千葉	ビストロアーシュ	3500
4	新宿	エスパーニャ	3000

レコード



2.MySQL のディレクトリ構造

MySQL のデータベースディレクトリには、バイナリログと呼ばれるデータベースの更新情報を格納するファイルと、2つのサブディレクトリが存在します。



mysql ディレクトリには権限テーブルと呼ばれる MySQL が使用するテーブル群を格納し、test ディレクトリは権限テーブルを初期化する際に作成されるデータベースのディレクトリです。ユーザが新たなデータベースの作成を行うと、MySQL はそのデータベース名のサブディレクトリをデータベースディレクトリ内に作成します。この構造は、使用するストレートエンジンに関係なく共通です。

MySQL では、テーブルに入力されたデータが、MySQL（mysql）ディレクトリの中にあるデータディレクトリに蓄積されます。MyISAM の場合のファイルの構成は、デフォルトでは次のようになります（インストール方法によっては、MySQL までの階層が異なる場合があります）。

- Windows
  C:/Program Files/MySQL/MySQL Server バージョン番号/data/データベース名/テーブル名.frm
  C:/Program Files/MySQL/MySQL Server バージョン番号/data/データベース名/テーブル名.MYD
  C:/Program Files/MySQL/MySQL Server バージョン番号/data/データベース名/テーブル名.MYI

- Linux
  /usr/local/mysql/var/データベース名/テーブル名.frm
  /usr/local/mysql/var/データベース名/テーブル名.MYD
  /usr/local/mysql/var/データベース名/テーブル名.MYI

.frm ファイルにはテーブル構造のデータ（カラム定義など）、.MYD ファイルには実際のデータ（テーブルのレコードデータ）、.MYI ファイルにはインデックスに関する情報（テーブルに対して作成された複数のインデックスデータとテーブルの統計情報）が書き込まれています。バックアップのときには、これらのファイルをコピーし、同じ data ディレクトリに復元することで、同じように使うことが可能です。

トランザクションを使用する場合は、InnoDB のテーブルを作成します。InnoDB では、各テーブルの定義は MyISAM 同様、.frm ファイルとして保存されますが、全テーブルのデータがテーブル空間として、data ディレクトリ直下にある「ibdata」で始まる名前のファイルに蓄積されます。InnoDB をファイルでバックアップする場合は、データディレクトリにあるデータファイル「ibdata 数値」とログファイル「ib_logfile 数値」と、該当のデータベースディレクトリにある.frm ファイルをコピーします。また、設定ファイル「my.ini」や「my.cnf」もバックアップしてください。

- Windows
  C:/Program Files/MySQL/MySQL Server バージョン番号/data/ibdata 数値
  C:/Program Files/MySQL/MySQL Server バージョン番号/data/ib_logfile 数値

- Linux
  /usr/local/mysql/var/ibdata 数値
  /usr/local/mysql/var/ib_logfile 数値

3.MyISAM

- MyISAM テーブルの特徴

MyISAM テーブルには固定長構造、可変長構造、圧縮テーブルの 3 種類のデータ構造があります。前者の 2 つはレコードデータのサイズの取り扱い方法で、MySQL が自動で選択します。なお、データ構造は「show table status」コマンドで確認できます。次の例では「TEST」テーブルが固定長レコードの構造（Row_format: Fixed）でできていることがわかります。

```
mysql> show table status ¥G;
******** 1. row ***************************
Name:            TEST
Engine:          MyISAM
Version:         10
Row_format:      Fixed
Rows:            0
Avg_row_length:  0
Data_length:     0
Max_data_length: 41658296553177087
Index_length:    1024
Data_free:       0
Auto_increment:  NULL
Create_time:     2006-07-12 15:09:18
Update_time:     2006-07-12 15:09:18
Check_time:      NULL
Collation:       latin1_swedish_ci
Checksum:        NULL
Create_options:
Comment:
```

- 固定長構造

  固定長構造は、テーブルを構成するカラムのデータ型に VARCHAR、TEXT、BLOB を含んでいない場合に選択されます。固定長構造の最大の利点は、レコードの削除が行われた時に削除されたデータ領域の再利用が容易なことです。よって、固定長構造のテーブルファイルは再利用できないデータ領域が残ってしまう「データのフラグメンテーション」が発生しない特長を持っています。データのフラグメンテーションとは、利用できない無駄なデータ領域が虫食いのように残ってしまうことをいいます。また固定長構造のテーブルには、レコードデータに「row number」と呼ぶレコードを一意に識別する値が付けられ、MySQL はこの値を利用して高速に該当レコードを探し当てる仕組みを持っています。

- 可変長構造

  可変長構造は、テーブルを構成するカラムのデータ型に VARCHAR、TEXT、BLOB を含んでいる場合に選択されます。可変長構造のテーブルファイルは固定長構造と異なり、レコードを削除した場合のデータ領域の再利用が難しいため、データのフラグメンテーションが発生する可能性を持っています。データのフラグメンテーションが発生するとディスクの利用効率が低下するため、検索性能が劣化してしまいます。そのため可変長構造のテーブルの場合は、一定周期でこのデータのフラグメンテーションを取り除く処理を実施する必要があります。

- 圧縮テーブル

  3 つ目の構造は、圧縮テーブルと呼ぶ読み取り専用のものです。この構造は自動で選択されるものではなく、オプションユーティリティ（myisampack）を用いてユーザが作成するものです。固定長構造／可変長構造ともに、圧縮テーブル構造に変更することが可能です。

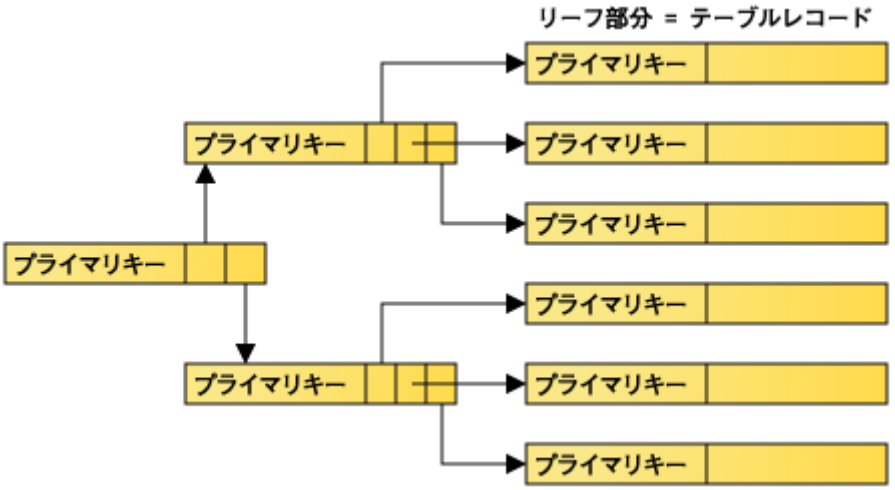
4.InnoDB

- InnoDB のディレクトリ構造とファイル

テーブルを作成する際に InnoDB を選択すると、データベース名のサブディレクトリ内には、「テーブル名.frm」ファイルのみが作成されます。InnoDB のテーブルのレコードデータやインデックスデータは、テーブルスペースと呼ばれるファイル内に格納するためです。テーブルスペースとは、標準ではデータベースディレクトリ内に「ibdata1」という名称で作成されるファイルのことです。このテーブルスペースは、コンフィグレーションの変更により、その場所やサイズを指定することができます。なおテーブル構造のデータは、テーブル名.frm ファイル内だけでなく、テーブルスペース内にも格納されます。

- InnoDB テーブルの特徴

InnoDB のテーブルは、クラスタードインデックスと呼ばれる特別なインデックスを備えた構造にて、テーブルスペースの中に格納されます。クラスタードインデックスとは、図に示すように、リーフと呼ぶインデックスの最下位レベルの部分がテーブルのレコードそのものになっている構成のインデックスです。よって、テーブル内のレコードはインデックス値の順に並んでいるといった特長を持ちます。他の DB として、Oracle では、この構成を索引構成表と呼び、通常のテーブルと区別して提供しています。



InnoDB のクラスタードインデックスの値としては、プライマリキーが使用されるため、プライマリキーの値順にレコードが並んだ構成のテーブルとなります。プライマリキーが定義されていないテーブルの場合は、InnoDB が自動的に 6 バイトのローID と呼ぶフィールドをレコードに追加し、このローID を用いてクラスタードインデックスを構成します。クラスタードインデックスは、構造上 1 テーブルに 1 つしか作成できません。そこで、セカンダリーインデックスとして、非クラスタードインデックスが作成できます。非クラスタードインデックスのリーフ部分には、プライマリキー値もしくは、ローID が使用され、最終的にはクラスタードインデックスを通して、対象のレコードが選択されます。

- トランザクション機能について

MyISAM エンジンはトランザクション機能を持っていませんが、MyISAM テーブルは更新できないわけではありません。INSERT／UPDATE／DELETE すべて実行可能です。トランザクション機能を持っていない分、非常に軽快に動作することで有名です。また MyISAM のテーブルのロック単位はテーブル単位で、InnoDB のテーブルはレコード単位です。

- InnoDB エンジンのトランザクション機能

InnoDB のトランザクション機能の特長は以下の通りです。

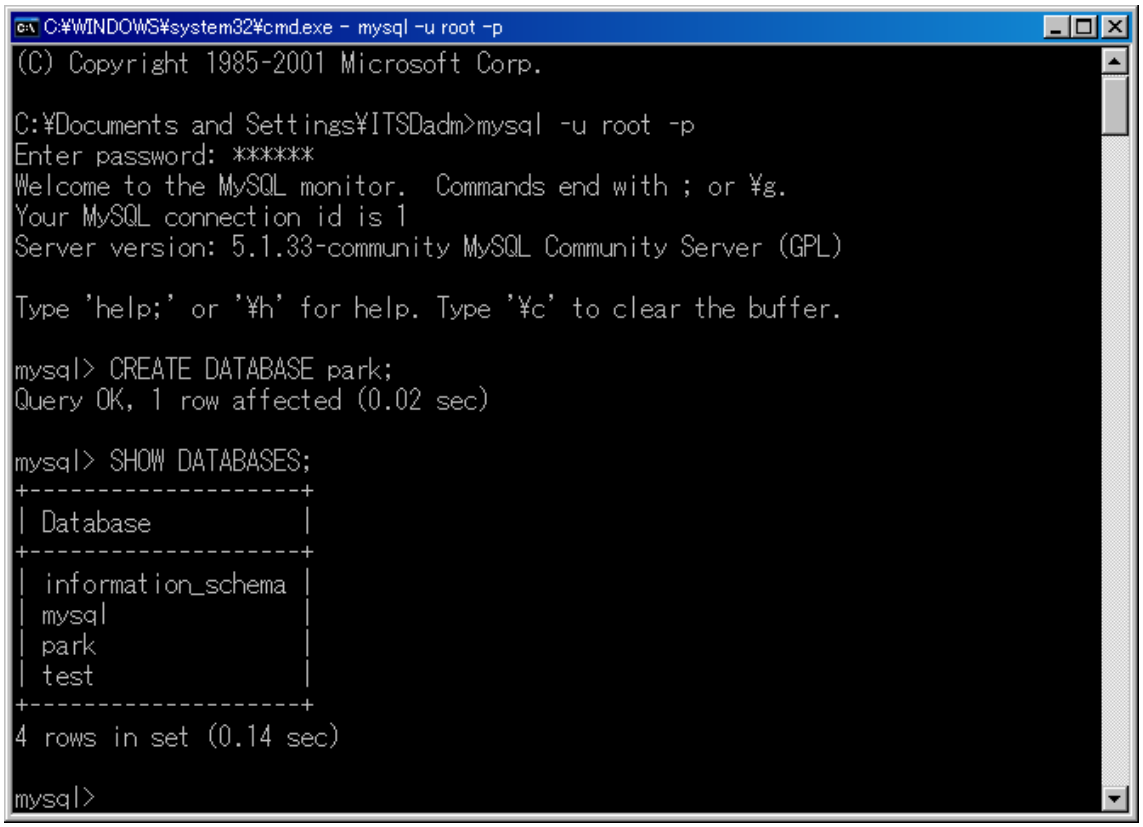
- Oracle 相当の読み込み一貫性機能を持つ
- ロック単位はレコード単位で、ロックのエスカレーションがない
- トランザクションの分離レベルは 4 つ持つ
  - ✓ リードアンコミッティド
  - ✓ リードコミッティド
  - ✓ リピータブルリード
  - ✓ シリアライザブル

InnoDB の読み込み一貫性機能の仕組みは、変更前データをロールバックセグメントに格納することによって実現しています。このロールバックセグメントは、テーブルスペース内に用意されます。機能だけでなく、実現方法も Oracle に似ています。ロックのエスカレーションが発生しない理由は、非常に少ないリソースにて個々のロックを実現しているためです。ここで紹介した特長はほんの一部ですが、InnoDB のトランザクション機能は商用の RDBMS に匹敵する機能を有しています。

- InnoDB エンジンのトランザクションログ

InnoDB は、トランザクションログのファイル群としてデータベースディレクトリ内の「ib_logfole0」ファイルと「ib_logfile1」ファイルを使用します。このファイル群のファイル数はユーザ自身で設定可能で、デフォルトで 2 つです。各ログファイルのサイズは 5M バイトで、これも設定可能です。InnoDB は「ib_logfole0」ファイルと「ib_logfile1」ファイルを循環的に使用して REDO ログと呼ぶ変更後データのログを記録します。

5.データベースの操作



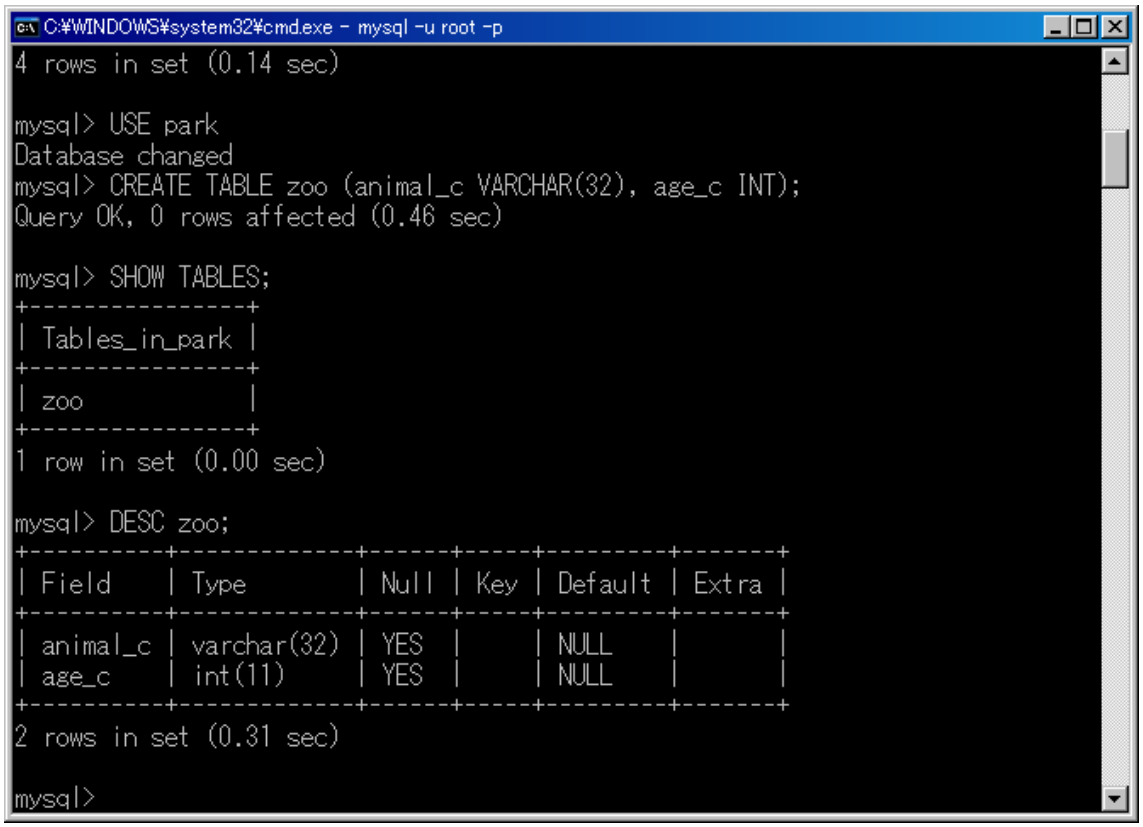
- **データベースの作成**
  CREATE DATABASE データベース名 [CHARACTER SET キャラクタセット名]
  例）CREATE DATABASE park;

- **データベースの削除**
  DROP DATABASE データベース名
  例）DROP DATABASE park;

- **データベースの一覧**
  SHOW DATABASES;

- **一覧にあるデータベースから使用するデータベースを選択**
  USE データベース名;
  例）USE park;

6.テーブルの操作



- **テーブルの作成**

  CREATE TABLE テーブル名 (テーブル定義)

  例）CREATE TABLE zoo (animal_c VARCHAR(32), age_c INT);

  ✓ **空欄(NULL)をゆるさないカラムを持つテーブルの作成**

  CREATE TABLE テーブル名 (カラム名1 データ型 NOT NULL, ...)

  例）CREATE TABLE zoo_null (animal_c VARCHAR(32) NOT NULL, age_c INT);

  ✓ **負の値が入らないようにカラムを作成**

  CREATE TABLE テーブル名 (カラム名1 数値データ型 UNSIGNED, ...)

  例）CREATE TABLE zoo_unsigned (animal_c VARCHAR(32), age_c INT UNSIGNED);

- **テーブルの一覧**

  SHOW TABLES;

- **テーブルの構成一覧**

  DESC テーブル名;

  [SHOW FIELDS FROM テーブル名;]

  例）DESC zoo;

- **テーブルの削除**

  DROP TABLE データベース名

  例）DROP TABLE zoo;

- **テーブルの構成一覧について**

Field, Type は文字通りそのカラム名と対応するデータ型を意味しています。CREATE 文を使用した際に指定したフィールド名およびデータ型そのものです。その後の意味はこうなっています。

Null	Null キャラクタの挿入が可能かどうか。"YES"なら挿入が可能
Key	データの抽出を速くするために使われる"index"（見出し）を作成する場合に設定されるキーの型
Default	データ挿入時に何も指定されなかった場合に挿入されるもの
Extra	そのほかの説明

- **カラムのデータ型について**

MySQL で使用可能なデータの型を次に挙げます。テーブル作成時にカラムの型として指定します。M は表示桁数、D は小数点以下の桁数を指定します。

数値型

型	内容
TINYINT [(M)] [UNSIGNED] [ZEROFILL]	1 バイト、符号あり -128～127, 符号なし 0～255
SMALLINT [(M)] [UNSIGNED] [ZEROFILL]	2 バイト、符号あり -32768～32767, 符号なし 0～65535
MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]	3 バイト、符号あり -8388608～8388607, 符号なし 0～16777215
INT [(M)] [UNSIGNED] [ZEROFILL]	4 バイト、符号あり -2147483648～2147483647, 符号なし 0～4294967295
BIGINT [(M)] [UNSIGNED] [ZEROFILL]	8 バイト、符号あり -9223372036854775808～ 9223372036854775807, 符号なし 0～18446744073709551615
FLOAT (p) [UNSIGNED] [ZEROFILL]	浮動小数点数。p は、単精度では 0～24, 倍精度では 25～53
FLOAT [(M, D)] [UNSIGNED] [ZEROFILL]	浮動小数点数。-3.402823466E+38～ -1.175494351E-38, 0, 1.175494351E-38～ 3.402823466E+38
DOUBLE [(M, D)] [UNSIGNED] [ZEROFILL]	浮動小数点数。-1.7976931348623157E+308～ -2.2250738585072014E-308, 0, 2.2250738585072014E-308～ 1.7976931348623157E+308
DOUBLE PRECISION [(M, D)] [UNSIGNED] [ZEROFILL]	DOUBLE と同じ
REAL [(M, D)] [UNSIGNED] [ZEROFILL]	DOUBLE と同じ
DECIMAL [(M, [D])] [UNSIGNED] [ZEROFILL]	文字列として格納される数値
DEC [(M, [D])] [UNSIGNED] [ZEROFILL]	DECIMAL と同じ
NUMERIC [(M, [D])] [UNSIGNED] [ZEROFILL]	DECIMAL と同じ
FIXED [(M, [D])] [UNSIGNED] [ZEROFILL]	DECIMAL と同じ
BIT	TINYINT(1)と同じ
BOOL	TINYINT(1)と同じ
BOOLEAN	TINYINT(1)と同じ

数値型データは、BIT, BOOL, BOOLEAN 以外であれば、UNSIGNED, ZEROFILL の指定ができます。
ZEROFILL を指定したカラムでは、指定のデータ型の表示幅に満たない場合、整数部分の左が 0 で埋められます。正の数に使用でき、たとえば、INT(8) と指定したカラムに 123 と入れると、左を 5 個 0 で埋めて表示されます。

日付型

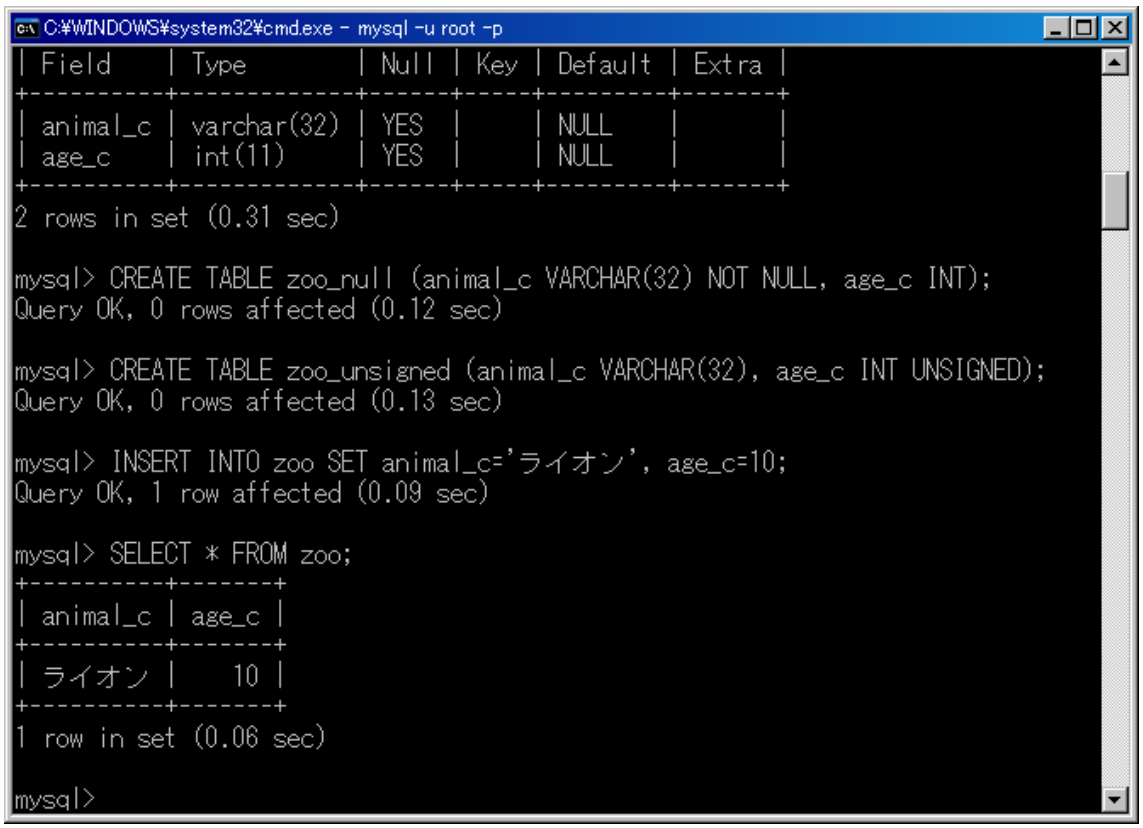
型	説明
DATETIME	YYYY-MM-DD HH:MM
DATE	YYYY-MM-DD で、1000-01-01～9999-12-31 まで
TIMESTAMP	YYYY-MM-DD HH:MM:SS で、1970-01-01 00:00:01～2037-12-31 23:59:59 まで
TIME	HH:MM:SS で、-838:59:59～838:59:59 まで
YEAR[(214)]	YYYYで、1901～2155 まで。YY で、2001～2069 まで(01-69)と、19701～1999 まで(70-99)。2000 年は'00'と文字列として指定する必要があり、00 では 0000 と解釈される

文字型

型	説明
[NATIONAL]CHAR(L)[BINARY | ASCII | UNICODE]	固定文字列。0～255 バイト
[NATIONAL]VARCHAR(L)[BINARY]	可変長文字列。0～255 バイト
BINARY(L)	CHAR とほぼ同じだが、文字コードの設定ができず、英字は大文字・小文字を区別する
VARBINARY(L)	VARCHAR とほぼ同じだが、文字コードの設定ができず、英字は大文字・小文字を区別する
TINYBLOB, TINYTEXT	最大長 255 バイト。TITYBLOB は大文字・小文字の区別をし、TINYTEXT は区別しない
BLOB, TEXT	最大長 65533 バイト。BLOB は大文字小文字の区別をし、TEXT は区別しない
MEDIUMBLOB, MEDIUMTEXT	最大長 16777215 バイト。MEDIUMBLOB は大文字・小文字の区別をし、MEDIUMTEXT は区別しない
LONGBLOB, LONGTEXT	最大長 4294967295 バイト。LONGBLOB は大文字小文字の区別をし、LONGTEXT は区別しない
ENUM(値 1, 値 2, 値 3,...)	カラムに入力可能な値のリスト。リストの値は 65535 個まで。リストの各値は、内部ではビットに対応する整数になっている
SET(値 1, 値 2, 値 3,...)	カラムに入力可能な値のリスト。リストの値は 64 個まで。リストの各値は、内部では整数になっている

※Lは、MySQL　4.1 からは、サイズではなくサーバの文字コードにあわせた文字数として指定します。

7.レコードの登録



- **レコードを登録する**

INSERT INTO テーブル名 SET カラム名='値' [,カラム名='値'....]
INSERT INTO テーブル名 (カラム名 [, カラム名...]) VALUES ('値' [, '値'...])
例)INSERT INTO zoo SET animal_c='ライオン', age_c=10;

- **一度にまとめて登録する**

INSERT INTO テーブル名 (カラム名 [, カラム名...])
  VALUES ('値' [, '値'...]) [, ('値'[,'値'...])...]
例)INSERT INTO zoo (animal_c, age_c)
    VALUES
      ('ペンギン', 5),
      ('カモノハシ', 15),
      ('ペリカン', 20);

- **テーブルのデータを見る(カラムを＊にすると全カラムを意味します)**

SELECT [DISTINCT] カラム名 FROM テーブル名
[WHERE [条件式] | [NOT 条件式] | [カラム名 LIKE '検索文字列']]
[ORDER BY カラム名 [ASC | DESC]]
例)SELECT * FROM zoo;

※DISTINCT を指定すると重複するカラム名をひとつにまとめて表示します。
※ORDER BY を指定するとカラム名の昇順(ASC)降順(DESC)で並び替えを行えます。順序の指定を省略すると昇順(ASC)で出力されます。
※LIKE を指定すると文字列検索を実行できます。ワイルドカードとして以下が利用できます。
% (0または複数文字を表す)
_ (1文字を表す)

8.レコードの操作

➢ レコードを削除する

レコードを削除するには、DELETE 文を使用します。WHERE 句をつけることで指定したレコードを削除できますが、つけわすれるとテーブルごと削除されてしまいますので注意してください。また、複数のテーブルから任意のレコードを削除することも可能です。

〈1つのテーブルから削除〉
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM テーブル名
　[WHERE 条件式] [ORDER BY ...] [LIMIT 行数]

〈複数のテーブルから削除〉
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] テーブル名[. *] [, テーブル名[. *]...]
　FROM テーブル参照方法
　[WHERE 条件式]

あるいは、
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM テーブル名[. *] [, テーブル名[. *]...]
　USING テーブル参照方法 [WHERE 条件式]

指定	説明
LOW_PRIORITY	ほかのクライアントによるテーブルからの読み取りが終了した後で削除を実行する
QUICK	MyISAM のみ。インデックスのリーフをマージせずに削除を実行する
テーブル参照方法	JOIN を含むテーブルリスト（SELECT の JOIN の説明を参照）

- **レコードを更新する**

〈テーブルが1つの場合〉
UPDATE [LOW_PRIORITY] [IGNORE] テーブル名
　　SET カラム名１＝値1 ［, カラム名２＝値2....］
　　［WHERE 条件式］［ORDER BY ...］［LIMIT 行数］

〈テーブルが複数の場合〉
UPDATE [LOW_PRIORITY] [IGNORE] テーブル名１［, テーブル名２....］
　　SET テーブル名. カラム名１＝値１［, テーブル名２. カラム名２＝値2....］
　　［WHERE 条件式］

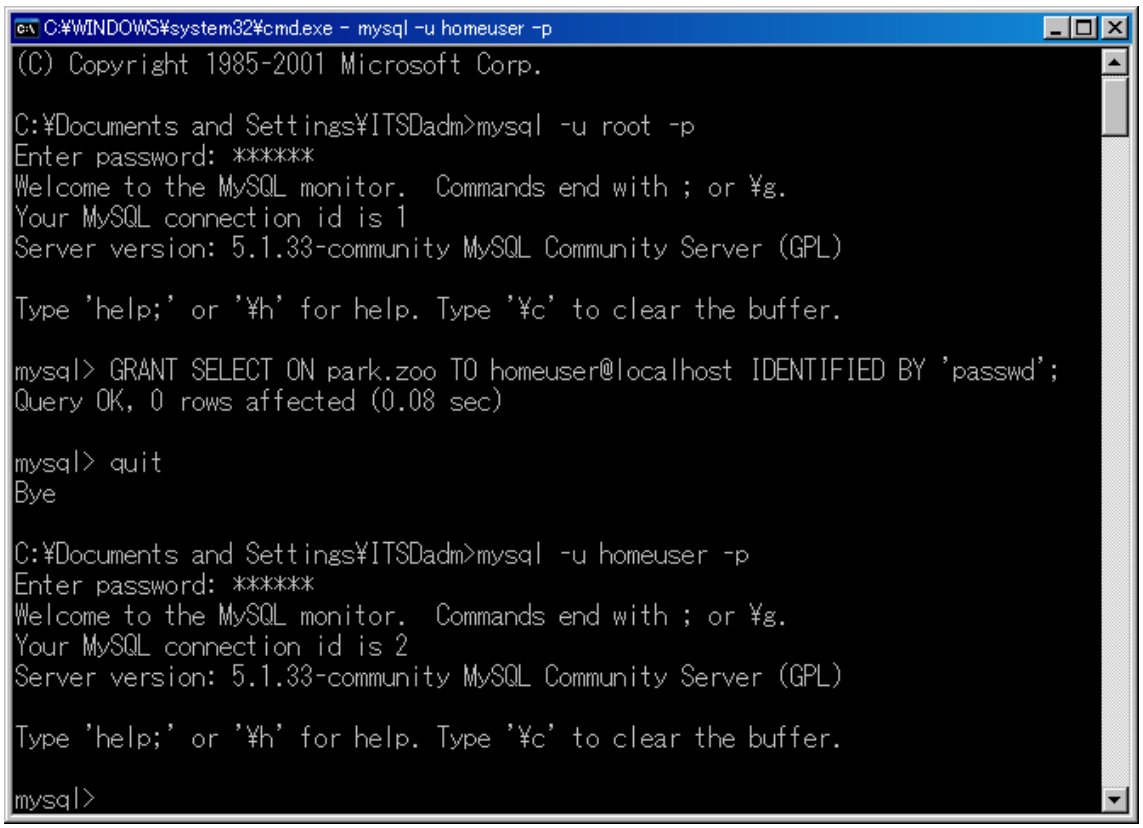
指定	説明
LOW_PRIORITY	他のクライアントによるテーブルからの読み取りが終了した後に更新を実行する
IGNORE	主キーまたはユニークキーのカラムに重複データで更新する場合、IGNORE の指定なしではエラーになり、IGNORE があると重複キーでも中断されず、重複を発生させるレコードは更新されない

- **レコードを入れ替える**

REPLACE INTO テーブル名 SET カラム名＝’値’［, カラム名＝’値’....］

主キーに設定されたカラムを持つテーブルを更新する場合に、REPLACE というコマンドが使用できます。ただし、既存のレコードに主キーの数値が一致するものがなければ、通常のレコード追加と同じ動作になります。

9.ユーザの作成削除、権限の設定



- **ユーザの作成**
  GRANT 権限 ON データベース名.テーブル名 TO ユーザ名@ホスト名 IDENTIFIED BY 'パス'

- **すべての権限を与える**
  GRANT ALL ON データベース名.テーブル名 TO ユーザ名@ホスト名 IDENTIFIED BY 'パス'

- **権限の反映(直接 mysql データベースに追加・変更を行った場合)**
  FLUSH PRIVILEGES

- **権限を無効にする**
  REVOKE 権限 ON テーブル名 FROM ユーザ名@ホスト名

- **ユーザを削除する**
  DROP USER ユーザ名

- MySQL のユーザ権限について

MySQL のユーザ権限には、以下のものがあります。「ALL」はすべての権限を与えることになるので、使用には注意が必要です。

権限	内容
ALL [PRIVILEGES]	すべての権限。管理者のみが持つことが望ましい。
ALTER	ALTER コマンドを発行する権限
CREATE	データベースやテーブル作成の権限
CREATE TEMPORARY TABLES	一時テーブル作成の権限。MySQL4.0.2 から
DELETE	DELETE コマンドを実行する権限
DROP	データベースやテーブル削除の権限
FILE	LOAD DATA INFILE および SELECT … INTO OUTFILE 使用の権限
INDEX	インデックス作成・削除の権限
INSERT	INSERT コマンドを発行する権限
LOCK TABLES	LOCK TABLES コマンドを発行する権限。MySQL4.0.2 から
PROCESS	SHOW FULL PROCESSLIST の使用権限
RELOAD	FLUSH を使用する権限
REPLICATION SLAVE	レプリケーションのスレーブサーバがマスターサーバへ接続する権限。MySQL4.0.2 から
REPLICATION CLIENT	SHOW MASTER STATUS および SHOW SLAVE STATUS を発行する権限。MySQL4.0.2 から
SELECT	テーブルに SELECT コマンドを発行する権限
SHOW DATABASES	SHOW DATABASES コマンドを発行できる権限。MySQL4.0.2 から
SHUTDOWN	mysqladmin プログラムで shutdown を使用する権限
SUPER	CHANGE MASTER,KILL,PURGE MASTER LOGS,SET GLOBAL を発行する権限。MySQL4.0.2 から
UPDATE	UPDATE コマンドを発行できる権限
USAGE	「権限なし」と同義
GRANT OPTION	GRANT コマンドを発行できる権限。ユーザの追加・削除やユーザに権限を与えたりすることができる。管理者のみが持つことが望ましい。

10.テーブル同士の操作

データベースでは、複数のテーブルを作成することでデータ変更時の手間を最小限にしています。しかし、複数のテーブルに分けたことにより、目的としているデータを取得するため複数に分けたテーブル同士を結合する作業が出てきました。テーブル同士の結合には任意のカラムをもとにテーブル同士を結合します。このとき、同じ値の同じ意味のあるカラム同士をもとに結合させます。結合方法には、全てのパターンを抽出する交差結合、記述したSQL分の左テーブルを元に結合を行う左結合、記述したSQL分の右テーブルを元に結合を行う右結合、同じ値があるレコード同士を抽出する内部結合とあります。結合された情報から WHERE 句などをつかい必要なレコード情報を取得することができます。

- **交差結合**

  SELECT * FROM テーブル名, テーブル名 [, テーブル名 …]

- **左結合**

  SELECT * FROM テーブル名１ LEFT JOIN テーブル名２
  ON テーブル名１. カラム名 = テーブル名２. カラム名

- **右結合**

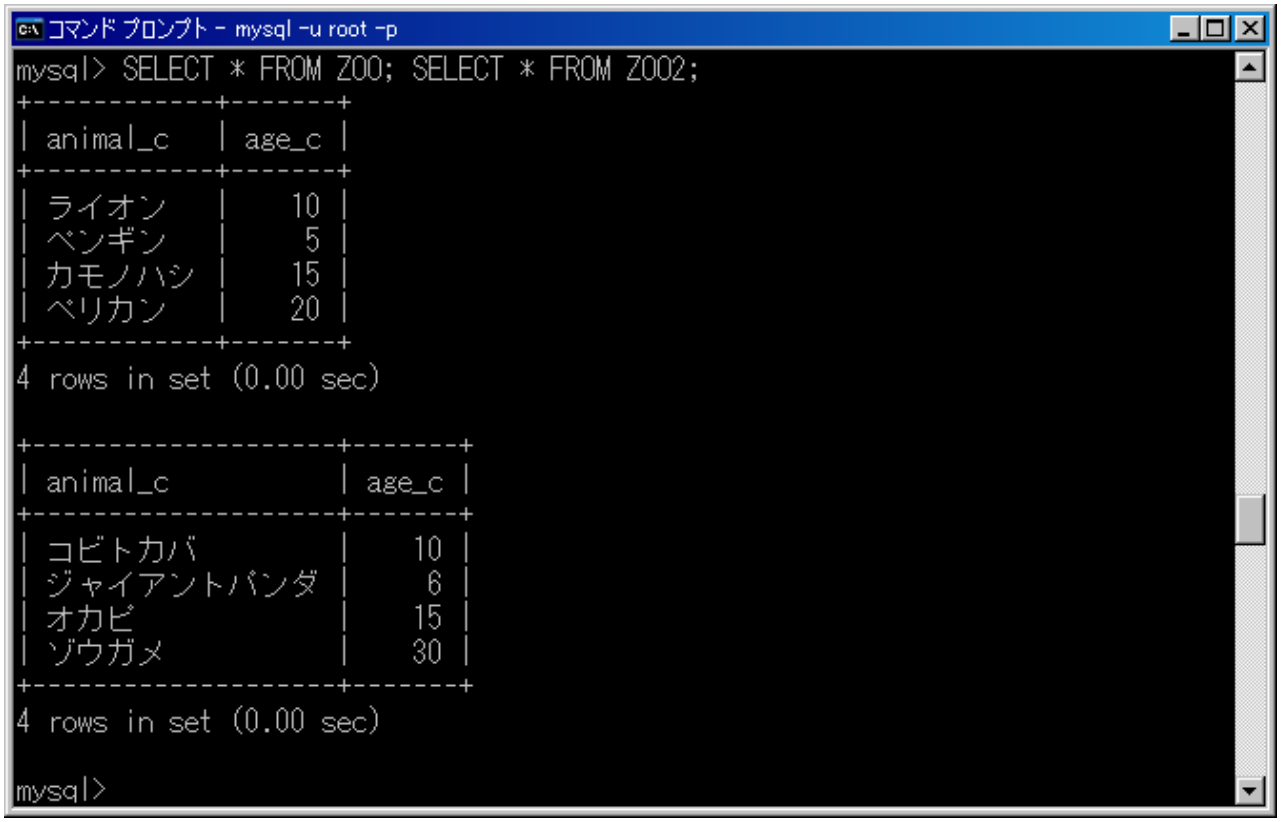
  SELECT * FROM テーブル名１ RIGHT JOIN テーブル名２
  ON テーブル名１. カラム名 = テーブル名２. カラム名

- **内部結合**

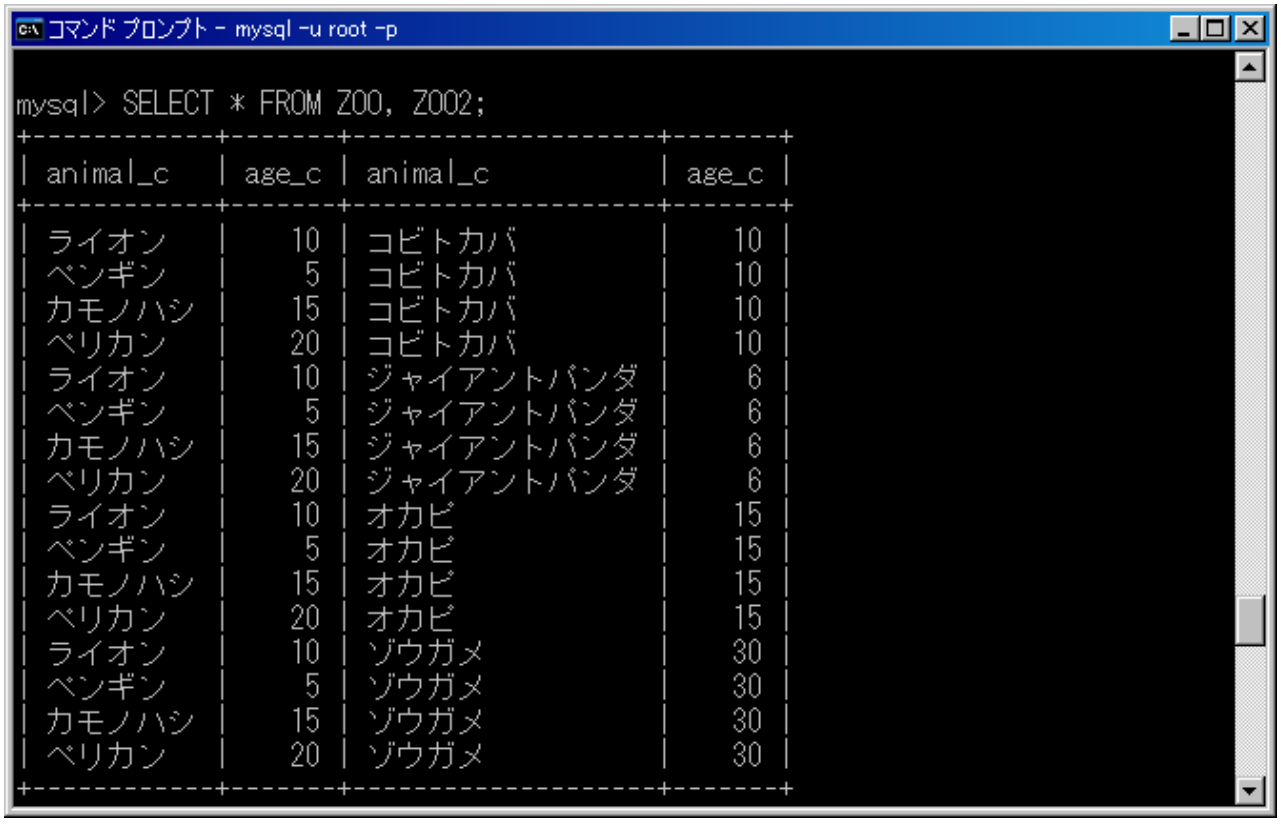
  SELECT * FROM テーブル名１ INNER JOIN テーブル名２
  ON テーブル名１. カラム名 = テーブル名２. カラム名

実行例

データベースに ZOO と ZOO2 のテーブルがあるとします。それぞれ結合パターンを実行してみます。

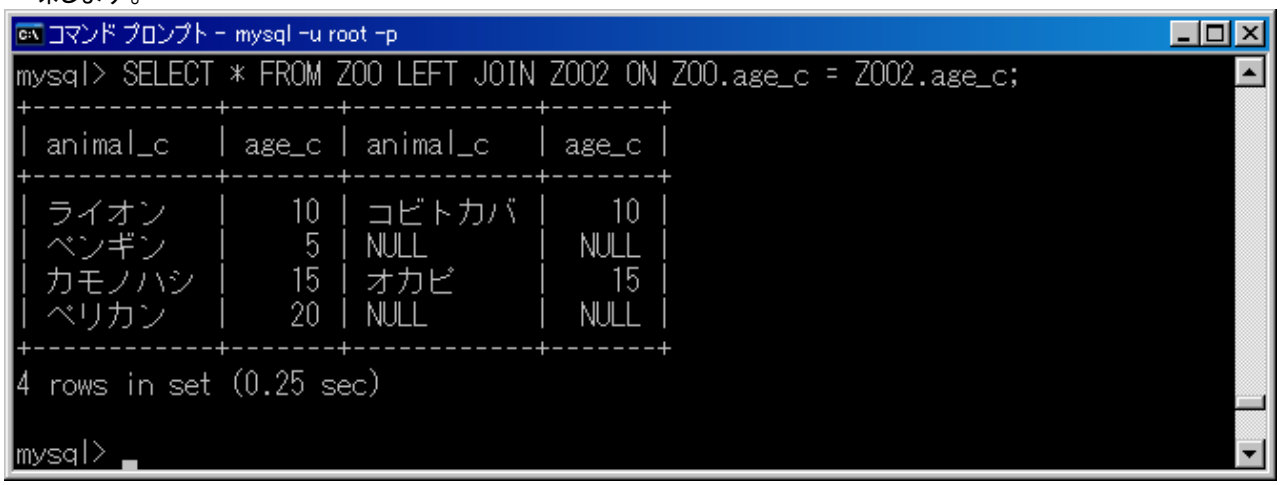


✓ 交差結合



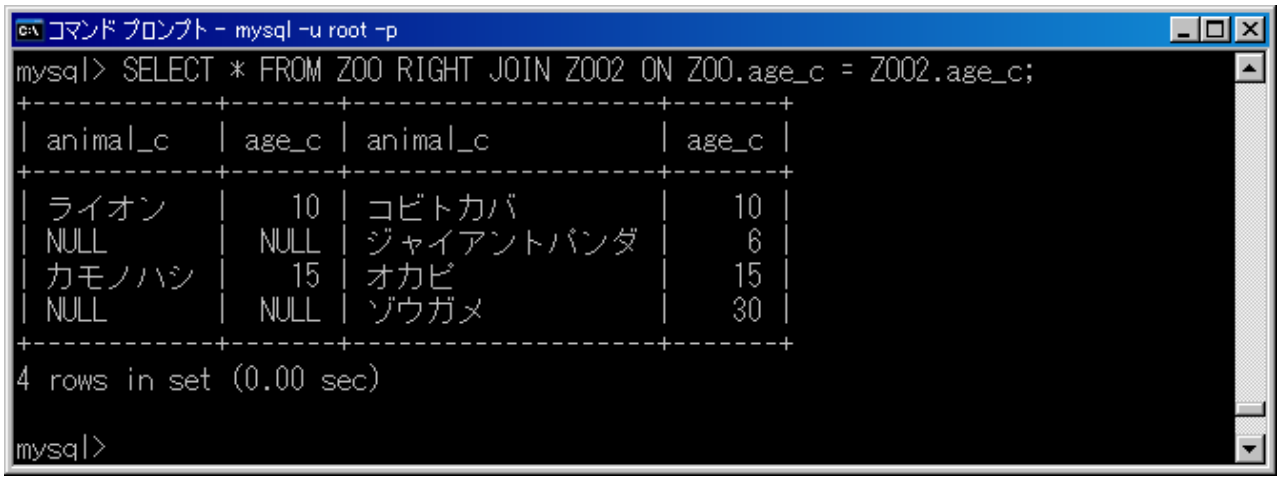
✓　左結合

年齢をもとに結合してみました。JOIN 句からみて左の ZOO のテーブルがそのまま残り、右の ZOO2 から同じ年齢のレコードを索引して結合されています。NULL は ZOO2 のテーブルに同じ年齢のレコードがなかったことを意味します。



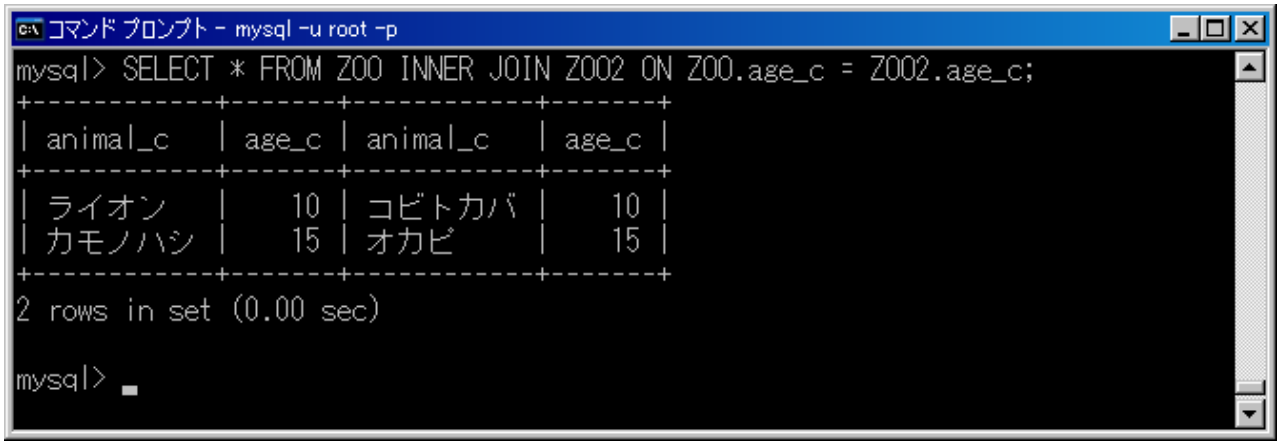
✓　右結合

JOIN 句からみて右に書かれてある ZOO2 テーブルをもとに ZOO テーブルから索引して結合しています。



✓　内部結合

結合するテーブルをみて双方にあるレコードだけを抽出します。



➢　縦方向に結合する（和結合）

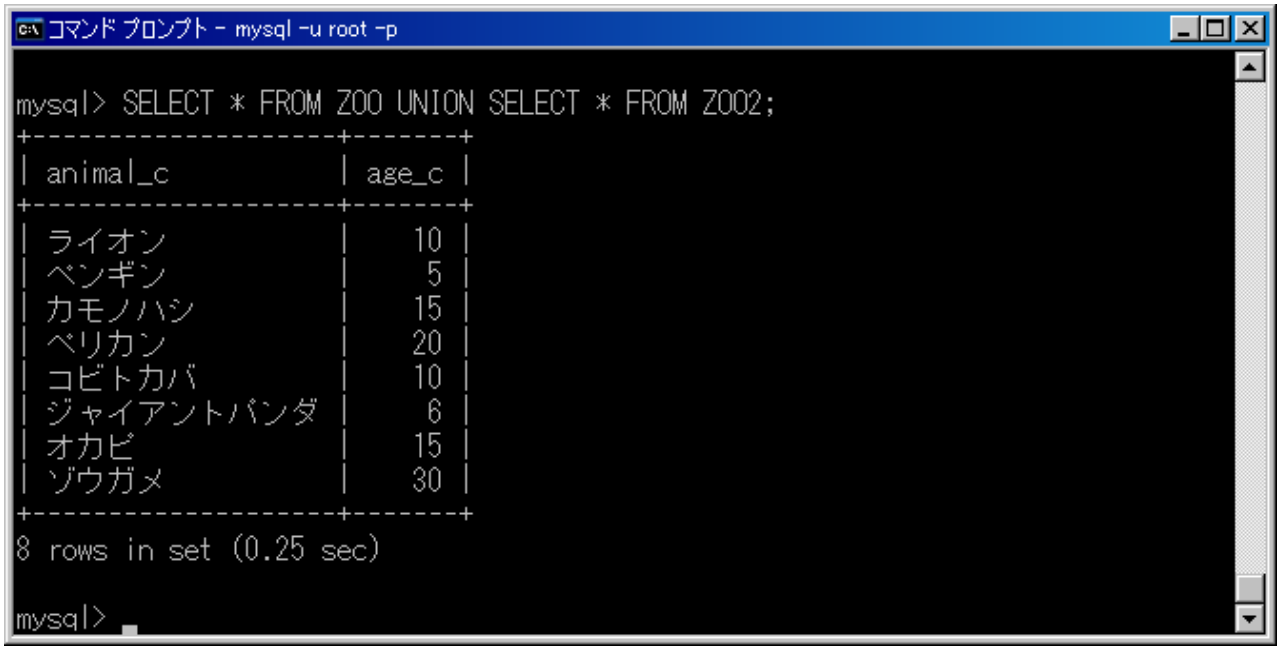
いままでのものは、テーブルを右に結合していく JOIN 句をやってみましたがテーブルを縦方向に結合するには UNION 句を使用します。UNION 句は同じテーブル構成でなければ結合できません。また、MySQL4.0.0 以降から使用可能となっています。3つ以上のテーブルを結合する場合も、そのまま UNION SELECT を続けて記述することで、全テーブルが結合されます。

構文

SELECT * FROM テーブル名１ UNION SELECT * FROM テーブル名２

SELECT * FROM テーブル名１ UNION SELECT * FROM テーブル名２ [UNION SELECT * ....]

※ *（ワイルドカード）の代わりに必要なカラム名を列記することも可能。いずれの場合もカラム数が同じになる必要があります。



11.サブクエリ

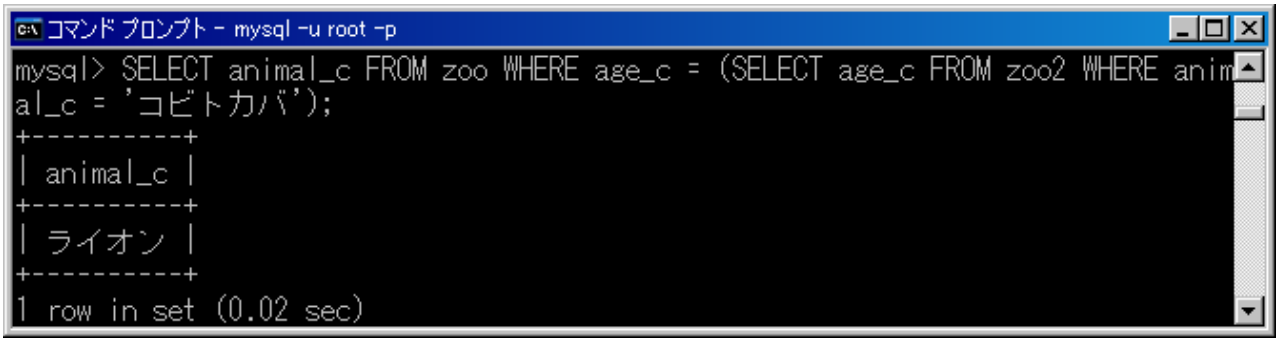
サブクエリの意味はひとことでいえば、構造化問い合わせ(副問い合わせ)といえばわかりやすいと思います。SQL 文の中に SELECT 文を埋め込み、抽出条件として利用することを可能としています。これまで、テーブルからデータを抽出するために WHERE 句による絞り込みや JOIN 句による結合を利用してきましたが、サブクエリはその応用といえる存在です。

構文
SELECT カラム名 FROM テーブル名
　　WHERE カラム 演算子 (サブクエリ);

WHERE 句にて、カラムに対する条件式としてサブクエリを使用しています。通常、WHERE 句に使う条件としては、数値や文字列といった何らかの値を利用しますが、サブクエリを利用することにより、SELECT 文の結果を条件として利用できます。サブクエリとして SELECT 文を記述する際、単一の結果を返す SELECT 文と複数行にわたる結果を返す SELECT 文が存在します。単一の結果であれば、イコール、大小といった条件式を利用できます。しかし、サブクエリの結果が複数行となる場合はこのような条件式は利用できず、「条件のいずれかと一致する」を意味する「IN」句を利用する必要があります。

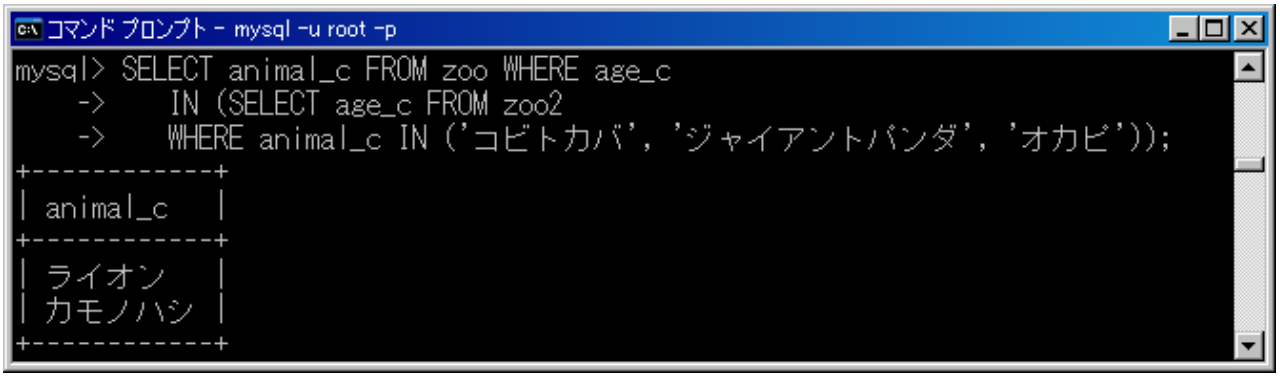
➢ WHERE 句でのサブクエリ

例)
SELECT animal_c FROM zoo WHERE age_c = (SELECT age_c FROM zoo2 WHERE animal_c = 'コビトカバ');



この SQL 文では、まず、zoo2 のテーブルから animal_c がコビトカバのレコードを探しその age_c を返します。次に返ってきた zoo2 の age_c のデータが zoo テーブルの age_c と同じものがあるか探し、該当するレコードの animal_c カラムを出力します。ちなみにこの場合、サブクエリで抽出される値はひとつだけなので比較演算子としてイコール(=)以外にも、大なり(>)、小なり(<)、以上(>=)、以下(<=)、等しくない(<>)の指定ができます。

例）
SELECT animal_c FROM zoo WHERE age_c
　IN (SELECT age_c FROM zoo2 WHERE animal_c IN ('コビトカバ', 'ジャイアントパンダ', 'オカピ'));



　サブクエリの返り値が1つ以上の場合（1つでも利用可能）、IN を指定して抽出条件を絞ります。この例では、zoo2 テーブルから animal_c の値がコビトカバ、ジャイアントパンダ、オカピの age_c である、10、6、15 がサブクエリの値として返ります。次にその値のいずれかに該当するレコードが zoo テーブルにあるか検索し該当するレコードの animal_c の値を出力します。他に比較演算子としては以下のようなものがあります。

演算子	説明
=	等しい（左側と右側が等しい）
>	より大きい（左側が右側より大きい）
>=	以上（左側と右側が等しいか、左側が右側より大きい）
<	より小さい（左側が右側より小さい）
<=	以下（左側と右側が等しいか、左側が右側より小さい）
<>、!=、^=	等しくない（左側と右側が等しくない）
BETWEEN .... AND	2つの値の間（2つの値は最小値、最大値の順に記述し、その値も含む）
IN (値リスト)	値リストのいずれかと等しい
LIKE パターン	パターンと等しい
IS NULL	NULLと等しい

　複数の条件があるときは、条件の優先順位に従って処理が行われます。条件式内での優先順位は次のとおりです。なお、算術演算子は、「*（乗算）、/（除算）」が先に計算され、「+（加算）、-（減算）」が後に計算されます。これらの優先順位を変更する場合は、「()」を使用します。()で囲まれた条件式は先に処理が行われます。

優先順位	条件式
1	SQL演算子（算術演算子（+、-、*、/）、連結演算子（||）など）
2	比較条件（=、<、<=、>、>=、<>、!=、^=）
3	IS NULL、IS NOT NULL、LIKE、IN、NOT IN
4	BETWEEN、NOT BETWEEN
5	NOT論理条件
6	AND論理条件
7	OR論理条件

- SELECT 句でのサブクエリ

ここまで取り上げたサブクエリは WHERE 句で利用してきましたが、サブクエリは WHERE 句だけでなくさまざまな場所でも利用可能です。ここでは、代表的な利用方法である SELECT 句と FROM 句での利用を取り上げます。

WHERE 句でサブクエリを利用する場合、サブクエリの結果を出力することはできず、WHERE 句にて条件式として利用するだけでした。一方、SELECT 句でサブクエリを利用する場合、メインクエリの結果としてサブクエリの結果を出力できます。代表的な例として集約関数などで使われます。

- FROM 句でのサブクエリ

FROM 句でのサブクエリの利用は、これまでのサブクエリと性格が異なります。WHERE 句、SELECT 句でのサブクエリはあくまで結果として値を利用していました。これが FROM 句での利用では、サブクエリの結果を 1 つのテーブルとして扱うことが可能となります。FROM 句でのサブクエリの利用は非常に応用性が高く、サブクエリの主だった利用目的はこの FROM 句での利用にあるといえます。FROM 句で利用することにより集約関数を組み込んだクエリを抽出対象に指定することが可能となります。サブクエリを FROM 句にて指定することで結果をあたかも1つのテーブルとして扱うことが可能となります。

12.集約関数

複雑で巨大なクエリを行った結果を端的に示すため、通常、平均値やデータの個数の合計値など文字通りデータを集約するための関数が SQL には標準で利用できます。そのための関数が集約関数です。基本的な集約関数は以下の通りです。集約関数は GROUP BY 句で集約したいカラムを指定できます。

集約関数	意味
SUM	合計
COUNT	行数
AVG	平均値
MIN, MAX	最大値、最小値
STD, STDDEV	標準偏差

- SUM 関数

構文
SELECT [カラム名,] SUM(カラム名) [AS 別名]
FROM テーブル
[GROUP BY カラム名];

グループ内の合計を計算します。NULL 値は無視されます。

- COUNT 関数

構文
SELECT [カラム名,] COUNT([DISTINCT] カラム名 | *) [AS 別名]
FROM テーブル
[GROUP BY カラム名];

グループ内の行数を求めます。単純に行数を得る場合、*を指定します。カラム名を指定した場合、指定されたカラム名の内容が **NULL 値であるものを除いて**カウントします。さらに DISTINCT を指定することにより重複する値を取り除いた状態でカウントします。

- AVG 関数

構文
SELECT [カラム名,] AVG([DISTINCT] カラム名) [AS 別名]
FROM テーブル
[GROUP BY カラム名];

グループ内の平均値を求めます。DISTINCT を指定すると、重複する値を取り除いた状態で平均値を計算します。

➢ MIN、MAX 関数

構文
SELECT [カラム名,] MIN | MAX(カラム名) [AS 別名]
FROM テーブル
[GROUP BY カラム名];

GROUP BY 句を指定しない SELECT では全体が1グループとなります。数値型、文字列型、日付型について抽出することが可能です。

➢ STD、STDDEV 関数

構文
SELECT [カラム名,] STDDEV(カラム名) [AS 別名]
FROM テーブル
[GROUP BY カラム名];

グループ内の標準偏差を計算します（DB2、MySQL では母集団標準偏差）。GROUP BY 句を指定しない SELECT では全体が1グループとなります。数値型についてのみ、標準偏差を求めることが可能です。Oracle、DB2、PostgreSQL、MySQL では、STDDEV を、SQL Server、Access では、STDEV を使います。

13.MySQL 管理運用

➢ MySQL データのバックアップ方法

ファイルやディレクトリのバックアップは比較的簡単ですが、データベースのバックアップとなると、いくつか特別な工夫を施す必要があります。ここでは、MySQL を取り上げていますが、原理としては PostgresSQL やその他のリレーショナルデータベースにもあてはまります。

MySQL サーバを休みなく稼動させる必要がないなら、以下に示すような圧縮なしのオフラインバックアップ手法が手っとり早いです。

1.MySQL サーバを停止させる。

```
# /etc/init.d/mysqld stop
```

2.MySQL のデータファイルおよびディレクトリをコピーする。例えば、MySQL のデータディレクトリ /var/lib/mysql を /tmp/mysql-backup に保存する場合は、次のようにします。

```
# cp -r /var/lib/mysql /tmp/mysql-backup
```

3.MySQL サーバを起動し直す。

```
# /etc/init.d/mysqld start
```

これに対し、オンラインバックアップは一筋縄ではいきません。相互に依存する MyISAM テーブルがある（外部キーやトランザクションは存在しない）場合は、各テーブルを順にロックし、そのファイルをコピーしてからロック解除を行うことでできます。しかし、InnoDB テーブルが存在する場合や、誰かが複数のテーブルを必要とするトランザクションを書く可能性がある場合は mysqlhotcopy, mysqlsnapshot, レプリケーション（replication）, mysqldump といった商用でない手頃なソリューションを利用することになります。

mysqlhotcopy は、ISAM テーブルや MyISAM テーブルをそのままの形でオンラインバックアップすることができる Perl スクリプトです。man ページには多数のオプションが記されていますが、以下では drupal と名付けられた単体のデータベースのバックアップ方法を示します。

```
# mysqlhotcopy -u user -p password drupal /tmp
  Locked 57 tables in 0 seconds.
  Flushed tables ('drupal'.'access', 'drupal'.'accesslog', 'drupal'.'aggregator_
  category', 'drupal'.'aggregator_category_feed', 'drupal'.'aggregator_category_item',
  'drupal'.'aggregator_feed', 'drupal'.'aggregator_item', 'drupal'.'authmap', 'drupal'.
  'blocks', 'drupal'.'book', 'drupal'.'boxes', 'drupal'.'cache', 'drupal'.'client',
  'drupal'.'client_system', 'drupal'.'comments', 'drupal'.'contact', 'drupal'.'file_
  revisions', 'drupal'.'files', 'drupal'.'filter_formats', 'drupal'.'filters',
  'drupal'.'flood', 'drupal'.'forum', 'drupal'.'history', 'drupal'.'locales_meta',
  'drupal'.'locales_source', 'drupal'.'locales_target', 'drupal'.'menu', 'drupal'.
  'node', 'drupal'.'node_access', 'drupal'.'node_comment_statistics', 'drupal'.'node_
  counter', 'drupal'.'node_revisions', 'drupal'.'permission', 'drupal'.'poll',
  'drupal'.'poll_choices', 'drupal'.'poll_votes', 'drupal'.'profile_fields', 'drupal'.
  'profile_values', 'drupal'.'role', 'drupal'.'search_dataset', 'drupal'.'search_
  index', 'drupal'.'search_total', 'drupal'.'sequences', 'drupal'.'sessions', 'drupal'.
  'system', 'drupal'.'term_data', 'drupal'.'term_hierarchy', 'drupal'.'term_node',
  'drupal'.'term_relation', 'drupal'.'term_synonym', 'drupal'.'url_alias', 'drupal'.
  'users', 'drupal'.'users_roles', 'drupal'.'variable', 'drupal'.'vocabulary',
  'drupal'.'vocabulary_node_types', 'drupal'.'watchdog') in 0 seconds.
  Copying 171 files...
  Copying indices for 0 files...
  Unlocked tables.
  mysqlhotcopy copied 57 tables (171 files) in 1 second (1 seconds overall).
```

mysqlsnapshot はさらに簡単に使えます。こちらは、サーバ上のすべての ISAM テーブルまたは MyISAM テーブルを、データベースごとに1つの tar ファイルにまとめてバックアップしてくれます。

```
# ./mysqlsnapshot -u user -p password -s /tmp --split -n
checking for binary logging... ok
backing up db drupal... done
backing up db mysql... done
backing up db test... done
snapshot completed in /tmp
```

MySQL のレプリケーション機能を毎日 24 時間休みなく利用できる設定にしていれば、上記各方法の1つを使ってスレーブサーバからバックアップを行うことができます。レプリケーション情報（ログや設定ファイルなど）の保存も必要になります。

ハードウェア障害（人的エラーは除く）に対するデータの保護を強化するには、レプリケーションを設定してスレーブサーバ（またはマスタサーバ、あるいはその両方）に RAID1（ミラーリング）ディスクを用意する必要があります。

多くの MySQL サイトは、本来のデータベーストランザクションとより優れた書き込みパフォーマンスを得るために、MyISAM テーブルから InnoDB テーブルにデータを移行しています。InnoDB のオンラインバックアップ用には、InnoDB モジュールの開発者によって InnoDB Hot Backup という商用プロダクトが提供されています。

最後に紹介するのが mysqldump で、ほとんどのマニュアルでは最初に紹介されていることが多いものです。mysqldump は、処理を加えずに(一字一句そのままに)コピーするのではなく、指定されたデータベースおよびテーブルの ASCII ダンプを生成します。この方法は、InnoDB などすべての種類の MySQL テーブルで使えます。比較的時間がかかり、巨大なテキストファイルが生成されるが、圧縮効率はかなり高いものとなります。ときどきこうしたダンプを作成しておくと役に立ちます。データベースやテーブルをスクラッチから再作成するためのわかりやすいスクリプトが含まれているためです。エディタや grep をはじめとするテキストツールを使用することにより、ダンプファイルに対して検索をかけたり、変更を加えることができます。

すべてのテーブルをロックして、1つのファイルにダンプするには、次のように入力します。

```
# mysqldump -u user -p password -x --all-databases > /tmp/mysql.dump
```

次のように、その出力をパイプして gzip をかけると、多少は処理時間とサイズを減らすことができます。

```
# mysqldump -u user -p password -x --all-databases | gzip > / tmp/mysql.dump.gz
```

こうしたバックアップツールのフロントエンドとしては、Zmanda Recovery Manager for MySQL というオープンソースツール(ダウンロードは無償、サポートは有償)があります。

用語	解説
mysqlhotcopy	mysqlhotcopy は、LOCK TABLES、FLUSH TABLES、および cp(または scp)を使用して、すばやくデータベースのバックアップを行う Perl スクリプトです。これは、データベースや単一のテーブルのバックアップを行う最速の方法ですが、データベースディレクトリのある同一マシンだけでしか実行できません。 mysqlhotcopy は、Unix のみ、および MyISAM テーブルと ISAM テーブルでのみ使用できます。サーバ上のすべての MyISAM テーブルを、DB ごとに 1 つの tar にまとめてバックアップしてくれます。
mysqlsnapshot	mysqlhotcopy とは、mysql の標準機能であり、MyISAM テーブルをそのままの形でオンラインバックアップできるスクリプト。mysqldump で sql 形式で出力するより効率がいいとされています。データベースをひとつずつ指定してバックアップしなくてはいけない。
レプリケーション(replication)	レプリケーションとは、データベース管理システムが持つ負荷分散機能の一つ。あるデータベースとまったく同じ内容の複製(レプリカ)をネットワーク上に複数配置し、通信回線や 1 台 1 台のコンピュータにかかる負荷を軽減する仕組みのこと。マスターデータベースとレプリカは、ネットワークを通じて互いにデータを交換しあい、常に内容が一致するようにできているため、一ヶ所でデータを更新すると、マスターとすべてのレプリカに自動的に更新内容が伝播する。
mysqldump	mysqldump とは、バックアップ用のデータベースまたはデータベースの集合をダンプしたり、他の SQL サーバ(MySQL サーバである必要はない)にデータを移動するためのユーティリティです。ダンプには、テーブル作成や入力のための SQL ステートメントが含まれます。

➢ MySQL データのリストア方法

mysql のデータベースはそもそもファイルで構成されているので、ファイルをそのまま別の場所へ移動するだけでもバックアップになります。ただし、書き込みなどが頻発しているデータベースをコピーするのはデータの不整合などを起こすことがあるので、データベースへの書き込みを禁止してバックアップしてくれるツールを使用することでそれが可能となります。基本的にバックアップファイルの復元の方法は、mysqlhotcopy, mysqlsnapshot などを使用した場合はバックアップを展開した*.MYD ファイル　MySQL のデータファイル, *.MYI ファイル　MySQL のインデックスファイル, *.frm ファイル　MySQL のテーブル定義ファイルの、つまり、1 つのテーブルにつき 3 ファイルの構成でできているので、これらのファイルを MySQL を停止し、MySQL のデータディレクトリへコピーまたは移動することで復元が完了します。しかしながら、mysqlhotcopy, mysqlsnapshot は MySQL データベース同士のデータ移行を前提としており、MySQL から他データベース（PostgreSQL や Oracle など）へのデータの移行に関しては前項のとおり、mysqldump を使用して行います。この場合、データは SQL クエリデータへ逆変換されダンプ（吐き出される）されます。他データベースへ移行する場合はこのダンプされた SQL クエリデータを入力データとして他のデータベースで読み込むことによりリストアが完了となります。

✓ mysqldump

<ダンプ>
C:¥>mysqldump データベース名 [テーブル名] > ″ディレクトリ名¥ファイル名″ -u ユーザ名 -p パスワード
C:¥>mysqldump --all-databases > ″ディレクトリ名¥ファイル名″ -u ユーザ名 -p パスワード
C:¥>mysqldump --databases データベース名 1 [データベース名 2 データベース名 3 ...] > ″ディレクトリ名¥ファイル名″ -u ユーザ名 -p パスワード

<復元>
C:¥>mysql データベース名 < ″ディレクトリ名¥ファイル名″ -u ユーザ名 -p パスワード

このコマンドは、mysql モニタからではなく、Linux のシェルや Windows のコマンドプロンプトから発行します。指定のデータベースの中の、すべてのテーブルとレコードを作成するための SQL 文を、ダンプファイルとして取り出す方法です。

C:¥>mysqldump park > ″c:¥park_dump.sql″ -u root -p
Enter password: ******

復元させるときは、あらかじめダンプファイルを展開する先のデータベースを作成しておきます。以下は、park2 データベースを指定して展開しています。

C:¥>mysql park2 < ″c:¥park_dump.sql″ -u root -p
Enter password: ******

「コマンドがない」というようなメッセージが表示されたら、mysql の bin ディレクトリ（例：Windows の場合は C:¥Program Files¥MySQL¥MySQL Server 5.1¥bin, Linux の場合は/usr/local/mysql/bin など）に移動してから発行するとよいでしょう。

ダンプファイル保存先のディレクトリには、コマンドを実行しているユーザの書き込み権限が必要です。データベース名の後ろにテーブル名をつけることで、特定のテーブルのダンプを取ることもできます。また、「mysqldump」に続けて、すべてのデータベースのダンプを取る場合は「--all-databases」というオプションを、複数のデータベースをダンプする場合は「--databases データベース名 1[データベース名 2 データベース名 3 ...] 」とカンマで区切らずにデータベース名を記述します。

➢ MySQL データの設定ファイル

MySQL サーバ起動時に、設定ファイル（my.ini、my.cnf）が読み込まれます。設定ファイルでは、サーバやクライアントのキャラクタセット、データファイルの保存場所、接続ポートなどを設定できます。

✓ Windows

設定ファイルは、インストーラを使用して、サービスを自動的に登録した場合、「mysqld-nt --defaults-file="C:¥Program Files¥MySQL¥MySQL Server 5.1¥my.ini"」と設定され、このファイルに記述された内容が有効になります。この指定がない場合、以下のファイルが以下の順番で読み込まれます。いずれもグローバルオプションです。最後に読み込まれた設定が有効になります。「WINDIR」はオペレーティングシステムによって異なり、Windows XP の場合は、「C:¥WINDOWS」です。

WINDIR¥my.ini
WINDIR¥my.cnf
C:¥my.ini
C:¥my.cnf

✓ Linux

設定ファイルは、configure 時に「--defaults-file」が指定されていない場合、以下の場所に保存された設定ファイルが、以下の順番で読み込まれます。ファイルがなければ無視して次のファイルを探し、最後に読み込まれた設定が有効になります。

/etc/my.cnf
DATADIR/my.cnf
defaults-extra-file
~/.my.cnf

Linux では、.cnf ファイルは作成されないので、自分で作成するか、MySQL の一部のバージョンでテンプレートとして作成される.cnf ファイル（例：/usr/local/mysql/share/mysql/my-small.cnf）をコピーし、適宜内容を書き換えて、ファイル名を変更して利用します。

✓ 設定ファイルの記述

日本語のレコードを扱い、mysql モニタで表示する場合は、設定ファイルの「mysqld」、「mysql」、「mysqldump」のそれぞれの欄に、「default-character-set=sjis」（Windows）あるいは「default-character-set=ujis」（Linux）と記述します。

```
[mysqld]
default-character-set=使用するキャラクタセット名
```

その他、必要に応じて変数を設定することが可能です。以下はデフォルトのテーブル型を MyISAM に指定しています。

```
[mysqld]
default-table-type=MyISAM
```

Windows で、デフォルトの設定ファイルにキャラクタセットの記述をしても表示がうまくいかない場合、「WINDIR」（Windows XP の場合は C:¥WINDOWS）に設定ファイルを作成し、[mysql]の欄を作成して、「default-character-set=sjis」を記述してください。それでもうまくいかない場合は、「my.ini」あるいは「my.cnf」ファイルを作成し、以下を同様に記述して、C:¥ドライブ直下に保存してください。あるいは、サンプル設定ファイル「my-small.cnf」「my-huge.cnf」「my-large.cnf」「my-medium.cnf」を利用してもよいでしょう。

```
[mysql]
default-character-set=sjis
```

以下は、「my.ini」あるいは「my.cnf」ファイルの設定例です。「MySQL Server Instance Configuration Wizard」で生成された「my.ini」ファイルを基に作成しています。設定値は、MySQL のクライアントから「SHOW VARIABLES」コマンドを発行して表示させることが可能です。

```
[client]
port=3308

[mysqld]
# ポート番号
port=3306

# ユーザパスワードを古いパスワード形式で保存します
old-passwords

# MySQL のディレクトリ
basedir="C:¥Program Files¥MySQL¥MySQL Server 5.1¥"

# データディレクトリ
datadir="C:¥Program Files¥MySQL¥MySQL Server 5.1¥Data¥"

# デフォルトのキャラクタセット
default-character-set=sjis

# デフォルトのテーブル型。トランザクションをあまり使用しない場合は MyISAM を指定
default-storage-engine=INNODB

# 最大同時接続数
max_connections=100

# クエリキャッシュのサイズ
query_cache_size=0
```

```
# テーブルキャッシュ
table_cache=256

# メモリ上の一時テーブルの最大サイズ
tmp_table_size=5M

# バイナリログが必要な場合、コメントアウトを外します
# log-bin

# 一般クエリログが必要な場合、コメントアウトを外します
# log

# 再利用できるようにキャッシュするスレッド数
thread_cache=8

#*** 以下、MyISAM 用の設定
myisam_max_sort_file_size=100G
myisam_max_extra_sort_file_size=100G
myisam_sort_buffer_size=8M
key_buffer_size=4M
read_buffer_size=64K
read_rnd_buffer_size=256K
sort_buffer_size=212K

#*** 以下、INNODB 用設定
innodb_additional_mem_pool_size=2M
innodb_flush_log_at_trx_commit=1
innodb_log_buffer_size=1M
innodb_buffer_pool_size=8M
innodb_log_file_size=10M
innodb_thread_concurrency=8

[mysql]
# SJIS 日本語キャラクタセットにします。EUC の場合は、「ujis」と記述
default-character-set=sjis
```

記述を間違えると MySQL サーバが起動しなくなるので、設定ファイルはバックアップを取ってから変更するとようでしょう。MySQL サーバ起動時の設定については、基本的に「mysqld-safe」や「mysqld-nt」のオプションを記述することが可能です。