

【MySQL】基礎と応用

環境編

■環境構築 (Ubuntu 編)

- ▶ MySQLをインストール
- ▶ ☆ MySQLを完全にアンインストール
- ▶ バージョンを確認

■基本

- ▶ ☆ たいていの場合において初めに行うこと
- ▶ ※ 以下の「ユーザ」とは当然、MySQL 内におけるユーザのことである。
- ▶ rootユーザとしてコンソールを起動
- ▶ " し、あるDBに接続
- ▶ 一般ユーザとしてコンソールを起動
- ▶ " し、あるDBに接続
- ▶ コンソールを終了
- ▶ ※ 外部ホストから接続を試みた場合に `_mysql_connector.MySQLInterfaceError: Can't connect to mysql server on '192.168.xxx.xxx:3306' (10061) (111)` のことも) などと出て接続に失敗するとき → my.cnf のなかで `bind-address = 127.0.0.1` という行をコメントアウトしたのち、MySQLを再起動。

基礎編

■初歩的注意

- ▶ ※ 正確には、MySQL から派生した MariaDB について学習する。
- ▶ ※ MySQLでは表形式でデータベースを管理する（関係データモデル）。
- ▶ ※ AS句は、列や表に別名をつけるために使う。（いくつかの場合を除く）
- ▶ ※ 命名規則はスネークケース (hoge_hoge_hoge) 。
- ▶ ※ 以降で「計算式」という場合、それには「値を別の値に変換する行為」すべてが含まれる。

【MySQL】基礎と応用

環境編

■環境構築 (Ubuntu 編)

- ▶ MySQLをインストール `$ sudo apt install mysql-server`
- ▶ ☆ MySQLを完全にアンインストール
- ▶ バージョンを確認 `$ mysql --version`

■基本

- ▶ ☆ たいていの場合において初めに行うこと
- ▶ ※ 以下の「ユーザ」とは当然、MySQL 内におけるユーザのことである。
- ▶ rootユーザとしてコンソールを起動 `$ sudo mysql -u root` ※Debianの場合
- ▶ " し、あるDBに接続 `$ sudo mysql -u root DB` ※ "
- ▶ 一般ユーザとしてコンソールを起動 `$ mysql -u user -p`
- ▶ " し、あるDBに接続 `$ mysql -u user -p DB`
- ▶ コンソールを終了 `exit` か `{Ctrl + D}`
- ▶ ※ 外部ホストから接続を試みた場合に `_mysql_connector.MySQLInterfaceError: Can't connect to mysql server on '192.168.xxx.xxx:3306' (10061) (111)` のことも) などと出て接続に失敗するとき → my.cnf のなかで `bind-address = 127.0.0.1` という行をコメントアウトしたのち、MySQLを再起動。

基礎編

■初歩的注意

- ▶ ※ 正確には、MySQL から派生した MariaDB について学習する。
- ▶ ※ MySQLでは表形式でデータベースを管理する（関係データモデル）。
- ▶ ※ AS句は、列や表に別名をつけるために使う。（いくつかの場合を除く）
- ▶ ※ 命名規則はスネークケース (hoge_hoge_hoge) 。
- ▶ ※ 以降で「計算式」という場合、それには「値を別の値に変換する行為」すべてが含まれる。

- ▶ ※ たくさんの句を後置する場合も、単に前から順に処理してゆくと考えれば大丈夫！
(WHERE と GROUP BY の関係だってそう)

■用語

- テーブル（ここでは表と表記してゆく）
- レコード（ここでは録と表記してゆく）：1行1行のデータのこと。
- カラム（ここでは列と表記してゆく）
- ビュー（ここでは観と表記してゆく）：既存の表（実表、基底表）を参照して作ることができ、その表の変更に伴い自らも自動更新する、表に似た存在（導出表）。
- クエリ（命令のこと）

■データ型の種類

- ▶ 整数
- ▶ 実数
- ▶ 文字列
- ▶ 真偽値
- ▶ 日時

■基本

- ▶ コメントのしかた
- ▶ DBを作る
- ▶ 接続するDBを切替
- ▶ キャッシュしたメモリを解放
- ▶ ユーザを作成
- ▶ ☆ ユーザのパスワード変更
- ▶ あるDBに対するほぼ全ての操作の権限をあるユーザに与える

■標準出力・表の読み取り

- ▶ バージョンを出力
- ▶ 出力
- ▶ DB一覧
- ▶ 今どのユーザで接続中かを出力
- ▶ 今どのDBに接続中かを出力

- ▶ ※ たくさんの句を後置する場合も、単に前から順に処理してゆくと考えれば大丈夫！
(WHERE と GROUP BY の関係だってそう)

■用語

- テーブル（ここでは表と表記してゆく）
- レコード（ここでは録と表記してゆく）：1行1行のデータのこと。
- カラム（ここでは列と表記してゆく）
- ビュー（ここでは観と表記してゆく）：既存の表（実表、基底表）を参照して作ることができ、その表の変更に伴い自らも自動更新する、表に似た存在（導出表）。
- クエリ（命令のこと）

■データ型の種類

- ▶ 整数 TINYINT INT BIGINT ※後に UNSIGNED をつければ0以上の数だけ
- ▶ 実数 DECIMAL FLOAT DOUBLE ※後に UNSIGNED をつければ0以上の数だけ
- ▶ 文字列 CHAR VARCHAR TEXT ENUM SET
- ▶ 真偽値 BOOL
- ▶ 日時 DATE TIME DATETIME

■基本

- ▶ コメントのしかた -- や # で行末まで、あるいは /* */ で囲めば改行可能。
- ▶ DBを作る CREATE DATABASE *DBName*;
- ▶ 接続するDBを切替 USE *DB*;
- ▶ キャッシュしたメモリを解放 FLUSH PRIVILEGES;
- ▶ ユーザを作成 CREATE USER '*userName*'@'localhost' IDENTIFIED BY '*password*';
 (事後いつか FLUSH PRIVILEGES;)

▶ ☆ ユーザのパスワード変更

- ▶ あるDBに対するほぼ全ての操作 GRANT ALL ON *DB.** TO '*user*'@'localhost';
 の権限のあるユーザに与える (事後いつか FLUSH PRIVILEGES;)

■標準出力・表の読み取り

- ▶ バージョンを出力 `select version();`
- ▶ 出力 `SELECT 何か;`
- ▶ DB一覧 `SHOW DATABASES;`
- ▶ 今このユーザで接続中かを出力 `select current_user();`
- ▶ 今このDBに接続中かを出力 `select database();`

▶ ユーザとそのホスト一覧	
▶ あるユーザがもつ権限を出力	
▶ すべての表の名前一覧	
▶ 表の構造を出力	
▶ 直近に挿入した録の主キー値を出力	

▶ すべての列を出力	
▶ 特定の列を出力	
▶ 条件に合う録のみに	
▶ ある列で並び替え	
▶ 最大 n 件の録のみに	
▶ m 件目以降で "	
▶ 計算して出力	
▶ UNIQUE(グループ化)	
▶ " して出力	
▶ グル化し集計して出力	
▶ 条件に合うグルのみに	
▶ 出力を上下で連結	
▶ 内部結合して出力	
▶ 左外部結合して出力	
▶ 右外部結合して出力	

■表の作成

▶ 表を作る	
▶ ある列で空値を禁止	
▶ ある列で値の重複を禁止する	
▶ ある列に初期値を設定	
▶ ある列に値の制限をかける	
▶ ある列を主キーに	
▶ ある列を主キー (自動) に	
▶ ある列を作成日時 (自動) に	

▶ ユーザとそのホスト一覧	SELECT user, host FROM mysql.user;
▶ あるユーザがもつ権限を出力	SHOW GRANTS FOR 'user'@'localhost';
▶ すべての表の名前一覧	SHOW TABLES;
▶ 表の構造を出力	DESC 表;
▶ 直近に挿入した録の主キー値を出力	select last_insert_id();

▶ すべての列を出力	SELECT * FROM 表;
▶ 特定の列を出力	SELECT 列1, 列2, ... FROM 表;
▶ 条件に合う録のみに	WHERE 条件式
▶ ある列で並び替え	ORDER BY 列1 DESC, 列2, ... ※ DESC つけると降順に
▶ 最大 n 件の録のみに	LIMIT n
▶ m 件目以降で "	LIMIT n OFFSET m または LIMIT m, n
▶ 計算して出力	SELECT 計算式 AS 好きな見出し名 FROM 表;
▶ UNIQUE(グループ化)	GROUP BY 列 ※この列は重複した値を持つ想定
▶ " して出力	SELECT 列 FROM 表 " ; か SELECT DISTINCT 列 FROM 表;
▶ グル化し集計して出力	SELECT 列1, 列2への集計処理 FROM 表 GROUP BY 列1;
▶ 条件に合うグルのみに	GROUP BY 列1 HAVING 条件式
▶ 出力を上下で連結	(SELECT ...) UNION ALL (SELECT ...);
▶ 内部結合して出力	SELECT ... FROM 表1 JOIN 表2 ON 表1.列1 = 表2.列2;
▶ 左外部結合して出力	SELECT ... FROM 表1 LEFT JOIN 表2 ON 表1.列1 = 表2.列2;
▶ 右外部結合して出力	SELECT ... FROM 表1 RIGHT JOIN 表2 ON 表1.列1 = 表2.列2;

■表の作成

▶ 表を作る	CREATE TABLE 表名 (列名1 型1, 列名2 型2, ...); ※既存ならI-
▶ ある列で空値を禁止	列名 型 NOT NULL
▶ ある列で値の重複を禁止する	列名 型 UNIQUE
▶ ある列に初期値を設定	列名 型 DEFAULT 値
▶ ある列に値の制限をかける	列名 型 CHECK (列名 >= 0 AND 列名 <= 200) など
▶ ある列を主キーに	..., 列名 n INT NOT NULL, ..., PRIMARY KEY (列名 n)
▶ ある列を主キー (自動) に	..., 列名 n INT NOT NULL AUTO_INCREMENT, ..., "
▶ ある列を作成日時 (自動) に	列名 DATETIME DEFAULT NOW()

▶ ある列を更新日時 (自動) に	
▶ 表を複製	
▶ 表の構造のみ複製	
▶ 出力を新表として作成	
▶ 出力を新視として作成	

■表の更新

▶ 表を削除	
▶ 表を削除し再作成	
▶ 表名を変更	
▶ 列を追加	
▶ 列を削除	
▶ 列名を変更	

▶ 録を挿入	
▶ 録に計算を加える	
▶ 録を削除	

■条件式・条件分岐

▶ 比較演算子	
▶ 論理演算子	
▶ m 以上 n 以下	
▶ a, b, c どれかに一致	
▶ ワイルドカード	
▶ パターンマッチング	
▶ NULL かどうか	
▶ ※ <code>!=</code> を含む条件の結果として NULL は含まれない	

▶ IF関数的な	
▶ SWITCH関数的な	

■数値

▶ 算術演算子	
---------	--

▶ ある列を更新日時 (自動) に	列名 DATETIME DEFAULT NOW() ON UPDATE NOW()
▶ 表を複製	CREATE TABLE 表名 AS SELECT * FROM 既存の表;
▶ 表の構造のみ複製	CREATE TABLE 表名 LIKE 既存の表;
▶ 出力を新表として作成	CREATE TABLE 表名 AS 表の形で出力させるクエリ;
▶ 出力を新視として作成	CREATE VIEW 視名 AS " ;

■表の更新

▶ 表を削除	DROP TABLE IF EXISTS 表;
▶ 表を削除し再作成	TRUNCATE TABLE 表; ※未存ならエラー
▶ 表名を変更	ALTER TABLE 表 RENAME 表名; ※表名の表が既存ならエラー
▶ 列を追加	ALTER TABLE 表 ADD 列名 型 ·· <u>AFTER 列</u> ※; ※略可。 <u>FIRST</u> も可
▶ 列を削除	ALTER TABLE 表 DROP 列名;
▶ 列名を変更	ALTER TABLE 表 CHANGE 列 列名 型; ※ <code>CHANGE</code> は注意が必要

▶ 録を挿入	INSERT INTO 表 (列1, 列2, ...) VALUES (値11, 値12, ...), (値21, ...), ...;
▶ 録に計算を加える	UPDATE 表 SET 列 = 計算式 WHERE 条件式;
▶ 録を削除	DELETE FROM 表 WHERE 条件式;

■条件式・条件分岐

▶ 比較演算子	> < >= <= = <> != ※ <code>列 > 値</code> のように使う
▶ 論理演算子	AND && OR NOT !
▶ m 以上 n 以下	列 BETWEEN m AND n
▶ a, b, c どれかに一致	列 IN (a, b, c)
▶ ワイルドカード	% : 0文字以上の任意の文字列 _ : 任意の1文字
▶ パターンマッチング	大小文字を区別しないなら 列 LIKE パターン文字列 " " を区別するなら 列 LIKE BINARY パターン文字列
▶ NULL かどうか	列 IS NULL 列 IS NOT NULL
▶ ※ <code>!=</code> を含む条件の結果として NULL は含まれない	

▶ IF関数的な	IF(条件式, 正の場合の値, 負の ")
▶ SWITCH関数的な	CASE <u>WHEN 条件式1 THEN 値1</u> ... ELSE 値n END

■数値

▶ 算術演算子	+ - * / %
---------	-----------

▶ 端数処理	
▶ 集計処理	
■文字列	
▶ 文字列を表現	
▶ 特殊な文字を表現	
▶ 連結	
▶ MID	
▶ 文字数	
▶ 大文字にする	
■日時	
▶ 現在日時	
▶ 年、月、日	
時、分、秒を抽出	
▶ 日付、時間を抽出	
▶ 別フォーマットに	
▶ 加算	
減算	
▶ 差	
■その他	
▶ ある時間だけ待つ	
▶ 外部ファをImport	

応用編

■副問合せ

- ▶ ※ 副問合せには2つある。
 - ・サブクエリ：あるクエリ内のSELECT句やWEHRE句の中身に使われるクエリ
 - ・インラインビュー：あるクエリ内のFROM句の中身に使われるクエリ
- ▶ ※ サブクエリについて、内側のSELECTによる結果は、1行1列になる必要がある。
- ▶ ☆ 相関サブクエリ：WHERE句を使用し、その中身でサブクエリ外の値を参照するもの

▶ 端数処理	FLOOR(列) CEIL(列) ROUND(列) ROUND(列, 桁数)
▶ 集計処理	COUNT(列) SUM(列) AVG(列) MAX(列) MIN(列)
■文字列	
▶ 文字列を表現	'文字列' か "文字列"
▶ 特殊な文字を表現	\n \t \' \"
▶ 連結	CONCAT(列1, 列2, ...)
▶ MID	SUBSTRING(列, <i>start</i> ※ ¹ , <i>len</i> ※ ²) ※ ¹ 負列 ※ ² 略すと最後まで
▶ 文字数	英：LENGTH(列) 日：CHAR_LENGTH(列)
▶ 大文字にする	UPPER(列)
■日時	
▶ 現在日時	NOW() ※DATE型なら今日になる
▶ 年、月、日	YEAR(列) MONTH(列) DAY(列)
時、分、秒を抽出	HOUR(列) MINUTE(列) SECONT(列)
▶ 日付、時間を抽出	DATE(列) TIME(列)
▶ 別フォーマットに	DATE_FORMAT(列, フォーマット)
▶ 加算	ADDTIME(列, '01:10:10') DATE_ADD(列, INTERVAL 1 SECOND)
減算	SUBTIME(列, '01:01') DATE_SUB(列, INTERVAL 1 DAY) など
▶ 差	DATEDIFF(列1, 列2) ※ 列1 - 列2 が得られる
■その他	
▶ ある時間だけ待つ	SELECT SLEEP(秒);
▶ 外部ファをImport	LOAD DATA LOCAL INFILE 'ファイルパス' INTO TABLE 表 … (表の列1, 表の列2, ...); ※ ... には各オプションを

応用編

■副問合せ

- ▶ ※ 副問合せには2つある。
 - ・サブクエリ：あるクエリ内のSELECT句やWEHRE句の中身に使われるクエリ
 - ・インラインビュー：あるクエリ内のFROM句の中身に使われるクエリ
- ▶ ※ サブクエリについて、内側のSELECTによる結果は、1行1列になる必要がある。
- ▶ ☆ 相関サブクエリ：WHERE句を使用し、その中身でサブクエリ外の値を参照するもの

- ▶ ☆ 抽出条件に使う
- ▶ ☆ 抽出元に使う（インラインビュー）

■ウィンドウ関数

- ▶ ※ ウィンドウ関数によって、グループ分けができる。パーティションという分け方や、さらにその中で利率的にグループ化するフレームという分け方がある。OVER句を用いて定義する。
- ▶ パーテを設定
- ▶ パーテ内で並び替え
- ▶ OVER句の中身に別名
- ▶ フレームを設定
- ▶ パーテ内で順位をつける
パーティ内で連番をふる
- ▶ パーテ内の n 個前の録
パーティ内の n 個後の録

■トランザクション

- ▶ トランザクションを設定
- ▶ トラに入る前の状態を回復

■正規化

- ▶ ※ サブの表では、「もとの表のid」を列として設けるのが一般的である。必要がある。
- ▶ ☆ 外部キー制約：副表における設値を元表の状況におうじて制限する。また、元表の録の削除や更新に応じて副表も対応する録が削除、更新されるようにする。
- ▶ ※ ツリー構造をもつデータを管理するには → 親のIDを記す列を追加

■CTE（Common Table Expression＝共通テーブル式）

- ▶ ※ CTEとは、1つの問合せのために一時的に別の表を定義する方法のことである。WITH句を用いる。
- ▶ ☆ 再帰的でないCTE：副問合せに別名をつけ、簡潔に書くために用いる。
- ▶ ☆ 再帰的なCTE：該当する録がなくなるまで同じ抽出作業を繰り返すために用いる。

■トリガー

- ▶ トリガーを設定

- ▶ ☆ 抽出条件に使う
- ▶ ☆ 抽出元に使う（インラインビュー）

■ウィンドウ関数

- ▶ ※ ウィンドウ関数によって、グループ分けができる。パーティションという分け方や、さらにその中で利率的にグループ化するフレームという分け方がある。OVER句を用いて定義する。
- ▶ パーテを設定 列1に対する計算式 OVER (PARTITION BY 列2) AS 好見出名
- ▶ パーテ内で並び替え OVER (・・・ ORDER BY 列・・・)
- ▶ OVER句の中身に別名 WINDOW 別名 AS OVER句の中身
- ▶ フレームを設定 ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING など
※デフォルトで設定されていることもある。
- ▶ パーテ内で順位をつける RANK() DENSE_RANK()
パーティ内で連番をふる ROW_NUMBER()
- ▶ パーテ内の n 個前の録 LAG(列, n) ※該当する録が存在しないならNULLに。
パーティ内の n 個後の録 LEAD(列, n) ※ n を省略すると $n=1$ と同義。

■トランザクション

- ▶ トランザクションを設定 START TRANSACTION; 一連処理 COMMIT;
- ▶ トラに入る前の状態を回復 ROLLBACK; ※PHPなどと組み合わせて使う

■正規化

- ▶ ※ サブの表では、「もとの表のid」を列として設けるのが一般的である。必要がある。
- ▶ ☆ 外部キー制約：副表における設値を元表の状況におうじて制限する。また、元表の録の削除や更新に応じて副表も対応する録が削除、更新されるようにする。
- ▶ ※ ツリー構造をもつデータを管理するには → 親のIDを記す列を追加

■CTE（Common Table Expression＝共通テーブル式）

- ▶ ※ CTEとは、1つの問合せのために一時的に別の表を定義する方法のことである。WITH句を用いる。
- ▶ ☆ 再帰的でないCTE：副問合せに別名をつけ、簡潔に書くために用いる。
- ▶ ☆ 再帰的なCTE：該当する録がなくなるまで同じ抽出作業を繰り返すために用いる。

■トリガー

- ▶ トリガーを設定 DROP TRIGGER IF EXISTS トリ名;
CREATE TRIGGER トリ名 AFTER※¹ 変更内容※² ON 表 FOR EACH ROW 処理;
※¹ BEFORE でも ※² INSERT か UPDATE か DELETE

▶ 既存のトリの一覧を出力	
■インデックス	
▶ ある列に索引を設定	
▶ ある列から索引を削除	
▶ ある表の索引の一覧を出力	
▶ どの索引が使われるか検証	

▶ 既存のトリの一覧を出力	SHOW TRIGGERS\G
■インデックス	
▶ ある列に索引を設定	ALTER TABLE 表 ADD INDEX 索引名(列);
▶ ある列から索引を削除	ALTER TABLE 表 DROP INDEX 索引;
▶ ある表の索引の一覧を出力	SHOW INDEX FROM 表\G
▶ どの索引が使われるか検証	EXPLAIN WHERE句などを用いた検索クエリ\G