SQL Cheat Sheet

By GeekHour @ bilibili

Structured
Query
Language

SCAN ME

分类

DDL（Data Definition Language） 数据定义语言

用来定义数据库对象，包括数据库、表、列等。
相关关键字包括：CREATE、DROP、ALTER等。

DML（Data Manipulation Language） 数据操作语言

用来对数据库中的数据进行增删改操作。
相关关键字包括：INSERT、DELETE、UPDATE等。

DQL（Data Query Language） 数据查询语言

用来查询数据库表中的记录/数据。
相关关键字包括：SELECT、WHERE等。

DCL（Data Control Language） 数据控制语言

用来定义数据库的访问权限和安全级别。
相关关键字包括：GRANT、REVOKE等。

TCL（Transaction Control Language） 事务控制语言

用来管理事务。
相关关键字包括：COMMIT、ROLLBACK等。

数据库相关

创建数据库

```
CREATE DATABASE IF NOT EXISTS game        -- 指定数据库名为 game
DEFAULT CHARACTER SET utf8mb4             -- 指定数据库字符集为 utf8mb4
COLLATE utf8mb4_0900_ai_ci;               -- 指定排序规则
```

- uft8mb4 表示使用UTF-8编码，每个字符最多占4个Byte。
- 0900 是UNICODE 校对算法版本。
- ai（Accent Insensitive）表示口音不敏感，排序时不区分e，è，é，ê和ë，不区分重音。
- ci（Case Insensitive）表示不区分大小写，排序时不区分a和A。

上面的语句等价于：

```
CREATE DATABASE  game;
```

MySQL8.0以后默认字符集为utf8mb4，默认排序规则为utf8mb4_0900_ai_ci。

删除数据库

```
DROP DATABASE game;
```

MySQL服务及配置

常见配置文件位置：

【Linux】
- `/etc/my.cnf`
- `/etc/mysql/my.cnf`
- `/etc/mysql/debian.cnf`
- `/etc/mysql/mysql.conf.d/mysqld.cnf`

【Windows】
- `C:\ProgramData\MySQL\MySQL Server X.X\my.ini`
- `C:\Program Files\MySQL\MySQL Server X.X\my.ini`

【Mac】
- `/opt/homebrew/etc/my.cnf`

服务启停（仅列出部分方式）：

【Linux】
- 启动：`mysqld_safe &`
- 停止：`./mysqladmin -u root -p shutdown`

【Windows】
- 启动：`mysqld --console`
- 停止：`./mysqladmin -u root shutdown`

【Mac】

```
· mysql.server start/stop/restart               # 启动服务
· brew service start/stop/restart mysql         # 自动启动
```

表相关

创建表

```
CREATE [TEMPORARY] TABLE[ IF NOT EXISTS] [库名.]表名 ( 表的结构定义 )[ 表选项]
```

```
CREATE TABLE player (
 id int DEFAULT NULL,                 -- 数据类型相关内容请参考右侧
 name varchar(45) DEFAULT NULL,
 level int DEFAULT NULL,
 exp int DEFAULT NULL,
 gold decimal(10,2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

- Uft8mb4 表示使用UTF-8编码，每个字符最多占4个Byte。
- 0900 是UNICODE 校对算法版本。
- ai（Accent Insensitive）表示口音不敏感，排序时不区分e，è，é，ê和ë，不区分重音。
- ci（Case Insensitive）表示不区分大小写，排序时不区分a和A。

删除表

```
DROP TABLE player;
```

清空表

```
TRUNCATE TABLE player;
```

SQL基础

```
USE database_name;

SELECT *
FROM player
WHERE level = 1
ORDER BY gold
LIMIT 3;
```

SQL关键字不区分大小写，但是建议关键字大写，方便阅读。
每个SQL语句需要以分号；结尾。
-- 两个中横线表示注释

SELECT 子句

```
SELECT name, level
FROM player
WHERE level = 1
ORDER BY gold
LIMIT 3;
```

SELECT子句中可以使用*表示所有字段，也可以使用各种表达式或者函数。

```
SELECT DISTINCT column FROM player;
```

DISTINCT

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

DISTINCT：去掉重复值。
e.g. `SELECT DISTINCT sex FROM player;`

IN

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

IN可以指定WHERE子句中的多个值。
e.g. `SELECT * FROM player WHERE level IN (1,3,5);`

LIKE

```
SELECT column1, column2, ...
FROM table_name
WHERE column LIKE pattern;
```

LIKE：用于在WHERE子句中搜索满足指定pattern模式的值。
e.g. `SELECT * FROM player WHERE name LIKE '张%';`

GROUP BY

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE condition
GROUP BY column_name
HAVING aggregate_function(column_name) operator value;
```

GROUP BY：结合聚合函数aggregate_function，根据一个或者多个列队结果集进行分组。常用的聚合函数包括COUNT、SUM、AVG、MIN、MAX等等。
HAVING：用于筛选分组后的数据。
e.g.

```
SELECT * FROM player
GROUP BY SUBSTR(name,1,1)
HAVING COUNT(*) >=5;
```

UNION

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION：用于合并两个或多个SELECT子句的结果（并集），默认去重复，如果允许重复值，请使用UNION ALL。
e.g.

```
SELECT * FROM player WHERE level BETWEEN 1 AND 5
UNION
SELECT * FROM player WHERE gold BETWEEN 1 AND 5;
```

INTERSECT

```
SELECT column_name(s) FROM table1
INTERSECT
SELECT column_name(s) FROM table2;
```

INTERSECT：用于合并两个或多个SELECT子句的结果（交集）
e.g.

```
SELECT * FROM player WHERE level BETWEEN 1 AND 5
INTERSECT
SELECT * FROM player WHERE gold BETWEEN 1 AND 5;
```

EXCEPT

```
SELECT column_name(s) FROM table1
EXCEPT
SELECT column_name(s) FROM table2;
```

EXCEPT：用于合并两个或多个SELECT子句的结果（差集）。
e.g.

```
SELECT * FROM player WHERE level BETWEEN 1 AND 5
EXCEPT
SELECT * FROM player WHERE gold BETWEEN 1 AND 5;
```

索引 （INDEX）

【创建】：

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
ON table_name (column_name(s));
```

【查询】：

```
SHOW INDEX FROM table_name;
```

【删除】：

```
DROP INDEX index_name ON table_name;
```

终端操作

登录

```
-- 登录本地MySQL服务， -u 指定用户名， -p： 使用密码登录
$ mysql -u root -p

-- 登录远程MySQL服务， -h 指定主机IP
$ mysql -h 10.211.55.5 -u root -p
```

切换数据库

```
mysql> USE database_name;
```

显示所有表

```
mysql> SHOW TABLES;
```

增删改查等操作

```
mysql> SELECT/UPDATE/DELETE/ALTER ……
```

退出

```
mysql> exit;
```

SQL基础

```
-- 这是一行注释，不会被执行
SELECT * FROM player          -- SQL语句可以写在一行，也可以分成多行
WHERE age = 20                -- SQL并不是大小写敏感的，
  AND money > 1000000         -- 但是一般建议关键字大写，其余小写
ORDER BY level DESC;          -- 语句使用分号结尾
```

修改密码及允许远程登录

修改root用户密码

```
-- 【5.7.9以下版本】修改root用户密码为123456：
UPDATE user SET password=PASSWORD("123456")
WHERE user=root;
-- or
UPDATE user SET authentication_string=PASSWORD("123456")
WHERE user=‘root’;
FLUSH PRIVILEGES;  -- 刷新权限

-- 【5.7.9以上版本】修改root用户密码为123456：
ALTER user 'root'@'localhost'
IDENTIFIED WITH mysql_native_password BY '123456';
FLUSH PRIVILEGES;  -- 刷新权限
```

修改root用户允许远程登录

```
UPDATE user SET host='%'
WHERE user='root' AND host='localhost';
FLUSH PRIVILEGES;  -- 刷新权限
```

配置文件my.cnf中的 bind-address 也需要改成 0.0.0.0

数据类型

数值类型

整型类型： INTEGER/INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT
定点类型： DECIMAL, NUMERIC
浮点类型： FLOAT, DOUBLE

日期类型

DATE, DATETIME, TIMESTAMP， TIME， YEAR

字符串类型

CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, SET

DELETE、TRUNCATE、DROP都有删除表或者数据的作用，区别是：
- DELETE只删除满足指定条件的数据；
- TRUNCATE是清空整张表的所有数据，但是表结构还在；
- DROP则是删除表结构和所有的数据了。

举个栗子：DELETE是单杀、TRUNCATE是团灭、DROP是电脑炸了。

MySQL-Shell

常用命令：

```
mysqlsh                                 # 登录
> \connect root@localhost.              # 连接服务
> \js     or  \py  or   \sql            # 切换语言
> \status or  \s                        # 显示当前MySQL Shell的状态
> \help   or  \?   or                   # 帮助
> \source or  \.                        # 使用活动语言执行脚本文件
> \quit   or  \q   or   \exit           # 退出
```

WHERE

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

WHERE：用来按照condition指定的条件过滤数据。
e.g. `SELECT * FROM player WHERE level >= 10;`

AND OR NOT

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND/OR condition2 AND/OR condition3;

SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

用于组合WHERE子句中的多个条件。优先级：NOT > AND > OR

BETWEEN

```
SELECT column1, column2, ...
FROM table_name
WHERE column BETWEEN value1 AND value2;
```

BETWEEN：用来选取介于value1和value2之间的数据范围内的值。
e.g. `SELECT * FROM player WHERE level BETWEEN 1 AND 10;`

REGEXP

```
SELECT column1, column2, ...
FROM table_name
WHERE column REGEXP pattern;
```

REGEXP：用于在WHERE子句中搜索满足指定pattern模式的值。
e.g. `SELECT * FROM player WHERE name LIKE '^张.?';`

ORDER BY

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2 ... ASC|DESC;
```

ORDER BY：用于按照column1，column2 ...来升序或者降序排序结果集。
e.g. `SELECT * FROM player ORDER BY level DESC;`

LIMIT

```
SELECT column1, column2, ...
FROM table_name
LIMIT number;
```

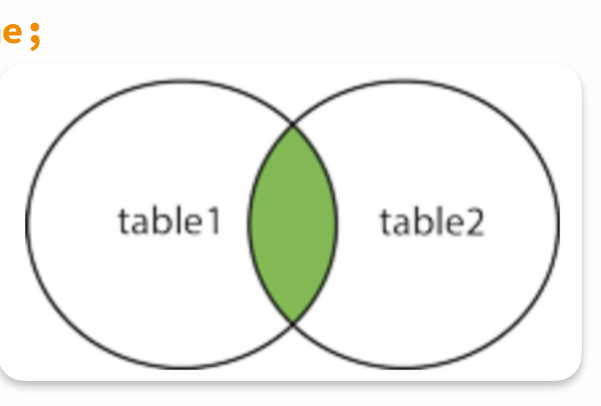
LIMIT：用于限定返回记录的条数。
e.g. `SELECT * FROM player LIMIT 3;`

INNER JOIN

```
SELECT column_name(s) FROM table1
(INNER) JOIN table2
ON table1.column_name = table2.column_name;
```

INNER JOIN：内连接
e.g.

```
SELECT * FROM player
INNER JOIN equip
ON player.id = equip.player_id;
```

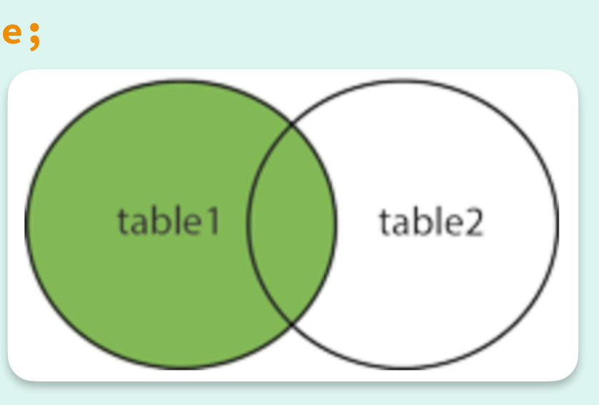


LEFT JOIN

```
SELECT column_name(s) FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

LEFT JOIN：左连接
e.g.

```
SELECT * FROM player
LEFT JOIN equip
ON player.id = equip.player_id;
```

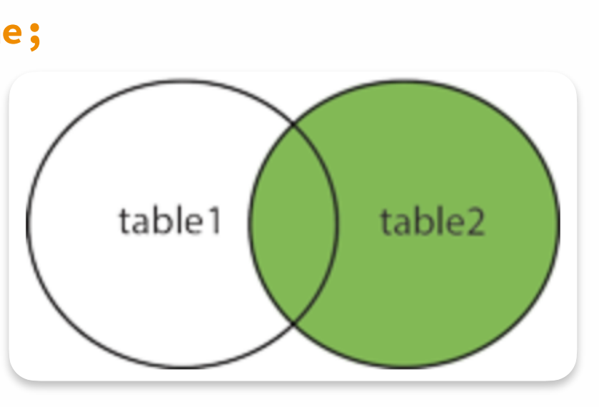


RIGHT JOIN

```
SELECT column_name(s) FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

RIGHT JOIN：右连接
e.g.

```
SELECT * FROM player
RIGHT JOIN equip
ON player.id = equip.player_id;
```



FULL JOIN

```
SELECT column_name(s) FROM table1
FULL JOIN table2
ON table1.column_name = table2.column_name;
```

FULL JOIN：全连接
e.g.

```
SELECT * FROM player
FULL JOIN equip
ON player.id = equip.player_id;
```