MySQL 常用语法

Note:

粗体表示变量；`强调体` 表示语法框架；大写表示关键字；括号表示可选项（alternatives）

I. 基础语句（SQL Basics）

`SELECT` DISTINCT **c**.attr1,**d**.*, AVG()/SUM()/MIN()/MAX()/COUNT() as **MR**

`FROM` **TB_1** AS **A**, **TB_2** AS **B**, **TB_3** UNION/INTERSECT/EXCEPT (ALL) **TB_4**, **TB_5** OUTER/NATURAL/LEFT/RIGHT/CROSS JOIN **TB_6** (AS) **F**

`WHERE` (EXIST/ANT/ALL 子嵌套) AND (**A**.attr1 == **B**.attr2) OR (**F**.attr1 (NOT) IN 枚举类型)

Note:

1. SELECT ... FROM ... WHERE 为MySQL最常用语句
2. FROM 语句中的AS 可以省略
3. 将子嵌套选出的数据另取一个名字可以避免 name alignment problem
4. 复杂语句构建时自底向上（Divided and conquer），先把小的部分表示出来，再通过组合处理单个部分形成复杂的语句

II. 修改 （Modification）

`INSERT INTO` **A**(attr1, attr2, attr3)

`VALUES` (attr1, attr2, attr3) / `SELECT` attr1, attr2, attr3 `FROM` ... `WHERE` ...

`CREATE TABLE` **tb_name**(

attr1 char (10) PRIMARY KEY,

attr2 int,

attr3 REFERENCES S(attr1) (ON DELETE SET NULL/ ON UPDATE SET CASCADE),

)

`DELETE FROM` **tb_name**

`WHERE` (condition, e.g. SELECT … FROM … WHERE/ attr1== v1)

`UPDATE` **tb_name**

`SET` (update operations, e.g. age=age+1, dept= 'CS')

`WHERE` (condition, e.g. SELECT … FROM … WHERE/ attr1== v1)

Note:

1. UPDATE 语句中如果属性没有显示指明，则按照默认参数顺序赋值，并自动补全其他属性
2. CREATE TABLE语句中指明属性与外键(foreign key)关联：REFERENCES 时最好指明当被引用表改属性改变（delete, update）时系统应该如何处理：ON DELETE SET NULL/ ON UPDATE SET CASCADE

III. 高级语句（Advanced SQL）

`SELECT` **c.**attr1 AS JP, **d**.attr2 as MO

`FROM` **product** P, **marker** M

`WHERE` (**c**.attr1==**d**.attr1) AND (EXIST/ALL/ANY (SELECT … FROM … WHERE **d**.attr1 (NOT) IN 二层子嵌套))

`GROUP BY` **c**.attr1, **d**.attr2

`HAVING` SUM(**c**.attr2)> v1

`ORDER BY` 1 DEC, 2 AEC, **c**.attr3

Note:

1. 后三条语句为当分组情况下（GROUP BY）使用，HAVING 指明了选择分组的条件，ORDER BY指明了分组的排列次序。
2. GROUP BY语句中若有多个分组条件则形成多级分组
3. HAVING 中的条件以组为单位，因此需要使用 SUM(), AVG(), MIN(), MAX(), COUNT() 等 aggregate的关键字，而非 c.attr1>v1
4. ORDER BY 语句中的数字（1, 2等）表示第几个属性，默认排序方式为 AEC，若要使其为降序要用DEC指明。语句中有多个排序条件时按照从左到右，当第i个条件无法分辨两条数据时，使用第i+1个条件，以此类推

IV. 约束（Constraints）

A. 局部约束 （从属于某个schema，如某个表）

`CREATE TABLE` **tb_name** (

attr1 char (10) PRIMARY KEY,

attr2 char (3),

attr3 REFERENCES S(attr1) (ON DELETE SET NULL/ ON UPDATE SET CASCADE),

attr4 INT `CHECK` (attr4<100),

`CONSTRAINT` **cons_name** `CHECK` (attr2 = 'F' OR attr1 IN 'Ms.%')

)

`SET CONSTRAINT` **cons_name** (INITIALLY) DEFERRED/ IMMEDIATE

`ALTER TABLE` **tb_name**

`ADD CONSTRAINT` **cons_name**/ `DROP CONSTRAINT` **cons_name**

B. 全局约束

`CREATE ASSERTION` **assert_name** `CHECK` (condition, e.g. **tb_name**.attr1<100)

Note:

1. 局部约束最好也要有名字，因为约束名称一旦定义无法修改，因此无名字不能通过ALTER TABLE 语句来增加、删除约束。
2. SET CONSTRAINT 语句中可以设置约束检查的时机，IMMEDIATE表示当约束相关属性改变时立即检查，而DEFERRED 表示推迟到本次事务（transaction）结束或者下一个事务开始（加INITIALLY关键字）时检查约束条件是否被违反。
3. CREATE ASSERTION语句建立全局约束。

V. 视图（View）

`CREATE VIEW` **v_name** (attr1, attr2, attr3)

`AS` SELECT attr1, attr2, attr3 FROM … WHERE …

`DROP VIEW` v_name

Note:

1. VIEW相当于一个宏定义，表明了数据的查看方式（将一条SELECT语句存储下来，而非存储SELECT出来的数据，这样大大节省了空间），每次调用VIEW语句相当于使用一次SELECT 语句。

2. VIEW语句在INSERT/UPDATE时候要格外注意，因为可能出现： INSERT/UPDATE的数据不再VIEW 观测中的属性将自动填充，而这些填充的属性有可能不满足VIEW语句的 WHERE条件，因此这些新增的语句不会显示在VIEW的返回数据中，造成逻辑上的歧义。
3. VIEW返回的数据若要频繁的使用，则适合将VIEW给实例化：

   `CREATE MATERIALIZED VIEW` CS_S
   `AS` SELECT sno,name,age FROM S WHERE dept='CS';

   这样能节约每次VIEW筛选相关数据的时间

VI. SQL环境

- schema:

  数据库管理系统（DBMS）管理的最小单元，表（table），视图（view），trigger，函数（function）都是schema中的一种。

- catalog:

  可以认为是文件夹/狭义上数据库，包含了多个schema。一个用户可以同时拥有多个catalog，即多个数据库

- clusters:

  可以认为是主文件夹，每个用户的所有相关信息的范围在一个cluster内

VII. Trigger

`CREATE TRIGGER` trg_name

`AFTER/BEFORE` `UPDATE/DELETE/INSERT`

`ON` attr1, attr2, attr3 `OF` tb_name

`REFERENCING` NEW ROW/ OLD ROW `AS` newrow_name/oldrow_name

`FOR EACH` ROW/STATEMENT

`WHEN` (condition, e.g. newrow_name.attr1=100)

`BEGIN`

operations, e.g. SELECT … FROM … WHERE …

```
END
```

Note:

1. ECA rules (event-condition-action rules):

    - Event = changes in DB, e.g., “insertion into S”
    - Condition = a test for whether or not the trigger applies
    - Action = one or more SQL statements

2. Comparison between trigger and constraint:

    Comparison:

	triggering event	cond	actions	who specify
Constraint	violation	check	reject	system
Trigger	explicitly specified	action	cond	explicitly specified programmer