MySQL关系型数据库

（三）高级查询、约束

作者：Daniel.Wang

主要内容

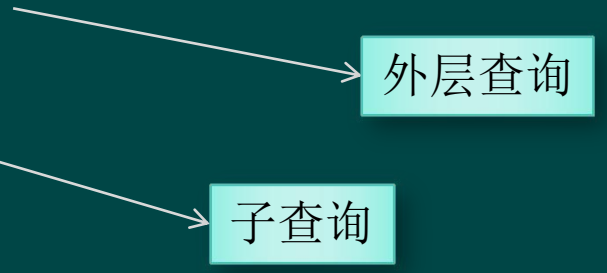
1. 子查询
2. 联合查询
3. 约束
4. 数据导入导出
5. 表的复制和重命名

（一）高级查询

1. 子查询

1）什么是子查询：查询语句中嵌套了另一个查询，也叫嵌套查询。例如：

```
select * from orders where amt >
(select avg(amt) from orders)
```



说明：

- 括号中的称为子查询
- 子查询返回一个结果集，可以多行多列，也可以一行一列
- 子查询返回的结果，要和条件要求的结果相匹配
- 先执行子查询，将子查询执行的结果作为外层查询条件，再执行外层查询
- 子查询只执行一遍

2）什么情况下使用子查询：当一个查询语句无法实现（或不方便实现）查询

3）单表子查询：子查询和外层查询查同一个表

- 语法

  select 字段列表 from 表A where 条件（select 字段列表 from 表A)

- 示例

  查询订单表中，余额大于所有订单金额平均值的账户

  select * from orders where amt >

  (select avg(amt) from orders)

  上面的查询等价于：

  select * from acct where balance > 514.44 -- 514.44是子查询得到的结果

4）多表子查询：子查询和外层查询不是同一个表

- 语法

  select 字段列表 from 表A where 条件

  (select 字段列表 from 表B [where 条件])

- 示例

  首先向customer表中插入更多数据

  insert into customer values

  ('C0004', 'Tom','13511111111'),

  ('C0005', 'Jack','13522222222'),

  ('C0006', 'Emma','13533333333'),

  ('C0007', 'Irris','13544444444'),

  ('C0008', 'Steven','13555555555'),

  ('C0009', 'Michile','13566666666'),

  ('C0010', 'Daniel','13577777777');

示例1：查询所有下过订单的客户编号、姓名、电话

```
select cust_id, cust_name, tel_no
from customer where cust_id in
(select distinct cust_id from orders);
```

示例2：查询所有状态为2的订单对应客户姓名、号码

```
select cust_id, cust_name, tel_no
from customer where cust_id in
(select distinct cust_id from orders where status = 2);
```

2. 联合查询

1）什么是联合查询：也叫连接查询，将两个（或以上）的表连接起来，得到一个查询结果

2）什么情况下使用联合查询：当从一个表中无法获得全部想要的数据时使用（前提是连接的表有数据关联）。例如：

```
select a.order_id, a.order_date,  b.cust_name, b.tel_no
from orders a, customer b
where a.cust_id = b.cust_id;
```

来自orders表

来自customer表

order_id	order_date	cust_name	tel_no
201801010001	2019-02-19 15:48:38	Jerry	13511223344
201801010002	NULL	Dekie	13844445555
201801010003	2019-02-19 17:59:00	Dokas	15822223333
201801010004	2019-02-19 17:59:00	Tom	13511111111

3）联合查询理论基础：笛卡尔积

- 什么是笛卡尔积：两个集合的乘积，表示用集合中的元素两两组合，产生的新集合
- 笛卡尔积的意义：表示集合中元素所有可能的组合情况。例如，A表示男同学集合，B表示女同学集合，体育课上一个男同学和一个女同学为一组做游戏，那么笛卡尔积表示所有男女同学可能的组合

A

姓名	性别
Tom	男
Jack	男
Steven	男

B

姓名	性别
Jerry	女
Lily	女

A × B

姓名	性别	姓名	性别
Tom	男	Jerry	女
Tom	男	Lily	女
Jack	男	Jerry	女
Jack	男	Lily	女
Steven	男	Jerry	女
Steven	男	Lily	女

- 笛卡尔积和关系：笛卡尔集中，去掉没有意义（或不存在的）组合，就是关系。例如，在上面的示例中，最终确定做游戏分组为如下二维表（关系）所示：

分组	姓名	性别
第一组	Tom	男
第二组	Jack	男
第三组	Steven	男

分组	姓名	性别
第一组	Jerry	女
第二组	Lily	女

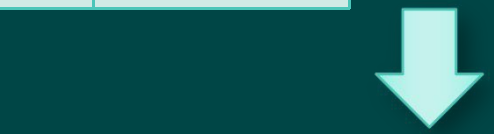
分组	姓名	性别	姓名	性别
第一组	Tom	男	Jerry	女
第二组	Jack	男	Lily	女
第三组	Steven	男		

4）内连接

- 什么是内连接：在表间利用某个列的值进行比较，将这些符合条件的数据组成新的记录，只有满足条件的数据才出现在查询结果中，不满足条件的数据丢弃不显示。如下面的图所示：

分组	姓名	性别
第一组	Tom	男
第二组	Jack	男
第三组	Steven	男

分组	姓名	性别
第一组	Jerry	女
第二组	Lily	女



分组	姓名	性别	姓名	性别
第一组	Tom	男	Jerry	女
第二组	Jack	男	Lily	女

- 如何实现内连接查询
  - 方式一：利用where条件连接

    语法：select 字段列表 from 表A, 表B where 关联条件

    示例：查询订单编号、下单日期、下单客户名称、下单客户电话

```
select a.order_id, a.order_date, b.cust_name, b.tel_no
from orders a, customer b
where a.cust_id = b.cust_id;
```

  - 方式二：利用inner join关键字

    语法：select 字段列表 from 表A inner join 表B on 关联条件

    示例：查询订单编号、下单日期、下单客户名称、下单客户电话

```
select a.order_id, a.order_date,  b.cust_name, b.tel_no
from orders a INNER JOIN customer b
on a.cust_id = b.cust_id;
```

5）外连接

- 什么是外连接：联合查询时，没有关联到的数据也显示称之为外连接
  - 左连接：左表数据全部显示，右表没匹配到则显示NULL
  - 右连接：右表数据全部显示，左表没匹配到则显示NULL

下图是一个左连接示意

分组	姓名	性别
第一组	Tom	男
第二组	Jack	男
第三组	Steven	男

分组	姓名	性别
第一组	Jerry	女
第二组	Lily	女

分组	姓名	性别	姓名	性别
第一组	Tom	男	Jerry	女
第二组	Jack	男	Lily	女
第三组	Steven	男	NULL	NULL

- 如何实现左连接查询
- 语法格式：将内连接中inner join改为left join即可

```
select 字段列表 from 表A
left join 表B
on 关联条件
```

- 示例：查询订单编号、下单日期、下单客户名称、下单客户电话，左表数据全部显示，如果订单没有匹配到客户数据，则显示NULL（查询语句和结果如下所示）：

```
select a.order_id, a.order_date,
       b.cust_name, b.tel_no
from orders a LEFT JOIN customer b
on a.cust_id = b.cust_id;
```

order_id	order_date	cust_name	tel_no
201801010010	2019-02-22 16:08:00	Emma	13533333333
201801010007	2019-02-22 16:08:00	Irris	13544444444
201801010008	2019-02-22 16:08:00	Steven	13555555555
201801010009	2019-02-22 16:08:00	Steven	13555555555
201801010011	2019-02-25 15:20:07	(Null)	(Null)

注意：orders表中，必须包含一笔订单，cust_id在customer表中找不到，才能看出效果

- 如何实现右连接查询
- 语法格式：只需要将左连接中left改为right即可

```
select 字段列表 from 表A
right join 表B
on 关联条件
```

- 示例：查询订单编号、下单日期、下单客户名称、下单客户电话，右表数据全部显示，如果客户没有匹配到订单，则显示NULL（查询语句和结果如下所示）：

```
select a.order_id, a.order_date,
       b.cust_name, b.tel_no
from orders a RIGHT JOIN customer b
on a.cust_id = b.cust_id;
```

order_id	order_date	cust_name	tel_no
201801010008	2019-02-22 16:08:00	Steven	13555555555
201801010009	2019-02-22 16:08:00	Steven	13555555555
201801010010	2019-02-22 16:08:00	Emma	13533333333
(Null)	(Null)	Michile	13566666666
(Null)	(Null)	Daniel	13577777777

注意：customer表中，必须包含一笔客户信息，在orders表中没有订单信息，才能看出效果

（二）约束

1. 约束概述

1）什么是约束（constraint）：保证数据完整性、一致性、有效性的规则

2）约束的作用：可以限制无效的数据进入数据库中，从数据库层面上提供了"安检"

3）约束的分类

- 非空约束：字段的值不能为空
- 唯一约束：字段的值必须唯一
- 主键约束：字段作为主键，非空、唯一
- 默认约束：未填写值的情况下，自动填写默认值
- 自动增加：字段值自动增加
- 外键约束

2. 定义和使用约束

1）非空约束（Not Null Constraint）：字段的值不能为空，如果插入数据时没有指定值，则报错

- 语法：字段名称 数据类型 not null
- 示例：

```
create table t1(
        id varchar(4) not null,
        name varchar(32)
);
insert into t1 values('0001', 'Jerry');     -- OK
insert into t1 values(NULL, 'Jerry');       -- 违反id非空约束
insert into t1 values('0002', NULL);        -- OK, name字段没有非空约束
```

2）唯一约束（Unique Constraint）：该字段的值唯一、不重复，如果插入或修改的值，已经存在则报错

- 语法：字段名称 数据类型 unique
- 示例：

```
create table t2(
        id varchar(4) unique,
        name varchar(32)
);
insert into t2 values('0001', 'Jerry');     -- OK
insert into t2 values('0001', 'Jerry');     -- 违反id唯一约束
insert into t2 values('0002', 'Jerry');     -- OK
```

3）主键约束(Primary Key，简写PK)：主键用来唯一标识表中的一笔记录，要求非空、唯一

- 主键特性：
  - ✓ 主键和一笔记录有唯一的对应关系
  - ✓ 一个表最多只能有一个主键
  - ✓ 可以单个字段、可以多个字段共同构成主键（注意：不是多个主键）
- 语法：字段名称 数据类型 Primary Key
- 示例：

```
create table t3(
        id varchar(4) primary key,
        name varchar(32)
);
insert into t3 values('0001', 'Jerry'); -- OK
insert into t3 values('0001', 'Tom'); -- 报错，违反唯一约束
insert into t3 values(NULL, 'Tom'); -- 报错，违反非空约束
```

4）默认值约束(Default Constraint)：指定某列的默认值，如果新插入一笔记录没有对该字段赋值，系统会自动为该字段填写一个默认值

- 语法：字段名称 数据类型 default 默认值
- 示例：

```
create table t4(
        id varchar(4) primary key,
        name varchar(32),
        status int default 0
);
insert into t4 values('0001', 'Jerry', 1); -- status字段有值
insert into t4(id,name) values('0002', 'Tom');  -- status字段未填值，设置默认值0
```

5）自动增长(auto_increment)：指定字段的值自动增长，设置为自动增长的字段，插入式不需要指定值（也可以指定值，但保证不重复），系统自动在最大值基础上增加1，字段添加自增长属性时，必须添加唯一约束或设为主键。

- 语法：字段名称 数据类型 auto_increment
- 示例：

```
create table t5(
        id int primary key auto_increment,
        name varchar(32)
);
insert into t5 values(NULL, 'Jerry');
insert into t5 values(NULL, 'Tom');
insert into t5 values(NULL, 'Hennry');
```

6）外键约束(Foreign Key，简写作FK)

- 什么是外键：某个字段在当前表不是PK，在另一个表中是PK
- 外键约束的作用：当一个字段被设置成外键时，另一个表中通过该外键关联的数据必须存在，这个特性被称为“参照的完整性”
- 示例：有两个表（“订单信息表”和“客户信息表”），“订单信息表”中的“客户编号”，是“客户信息表”的主键，可以在该字段上添加外键约束

订单编号	下单时间	客户编号（FK）	订单状态	商品数量	总金额
201801010001	2019-02-19 15:48:38	C0001	1	1	100.00
201801010002	2019-02-19 17:59:00	C0002	1	2	240.00

客户编号（PK）	客户姓名	客户电话
C0001	Jerry	13811111111
C0002	Tom	13822222222

- 添加外键后，产生以下影响：
  - “客户信息表”中“客户编号”为“C0001”的记录不能被删除，因为“订单信息表”参照了该实体；
  - 当在“订单信息表”中插入一笔“客户编号”为“C0003”订单，不能被插入，因为参照的“C0003”实体在“客户信息表”中不存在；
  - 当修改“客户信息表”中“C0002”客户“客户编号”时，不能被修改，因为“订单信息表”参照了该实体；

- 使用外键的条件
  - 表的存储引擎类型必须为innodb
  - 被参照字段在另外的表中必须是主键
  - 当前表中类型和另外表中类型必须一致（如果字段类型为字符串，字段编码格式要一致）
- 语法
  - constraint 外键名称 foreign key(当前表字段名) references 参照表(参照字段)

- 外键示例：创建课程信息表（course），教师信息表（teacher），在教师信息表course_id上添加外键约束，并插入数据

第一步：创建表

```
create table course (
    course_id varchar(4) primary key,
    name varchar(32)
);
create table teacher (
    id int auto_increment primary key,
    name varchar(32),
    course_id varchar(4),
  CONSTRAINT fk_course FOREIGN KEY(course_id) REFERENCES course(course_id)
);
```

第二步：插入课程信息

```
insert into course values
('0001', 'Python编程基础'),
('0002', '数据库原理与应用');
```

第三步：插入教师信息验证

```
insert into teacher values(NULL, '张大大', '0001'); -- OK
insert into teacher values(NULL, '张大大', '0003');-- 错误，0003课程实体不存在
delete from course where course_id = '0001';-- 错误，0001课程实体被参照，不能删除
```

7）通过修改字段方式添加约束

- 首先，创建测试表t6

```
create table t6(
        id int,
        name varchar(32),
        status int,
        course_id varchar(4),
        tel_no varchar(32)
 );
```

- 通过修改表定义语句添加约束

```
alter table t6 add primary key(id);      -- 添加主键
alter table t6 modify id int auto_increment;       -- 添加自增长
alter table t6 modify status int default 0;          -- 添加默认值
alter table t6 modify tel_no varchar(32) unique;   -- 添加唯一约束
alter table t6 add CONSTRAINT fk_course_id    -- 添加外键约束
FOREIGN KEY(course_id)
REFERENCES course(course_id);
```

（三）数据导入导出

1. 概述

1）导出：讲数据库中的数据导出到文件中

2）导入：将文件中的数据导入到数据库表中

3）如何查看导入导出路径：show variables like 'secure_file%';

```
mysql> show variables like 'secure_file%';
+------------------+-----------------------+
| Variable_name    | Value                 |
+------------------+-----------------------+
| secure_file_priv | /var/lib/mysql-files/ |
+------------------+-----------------------+
```

2. 导出

1）语法格式

select查询语句

into outfile '文件名称'

fields terminated by '字段分隔符'

lines terminated by '行分隔符'

2）导出示例

第一步：首先找到数据库允许导出的路径

```
show variables like 'secure_file%';
```

第二步：执行导出

```
select * from orders
into outfile '/var/lib/mysql-files/orders.csv'
fields terminated by ','
lines terminated by '\n';
```

第三步：查看导出结果（Linux命令行中执行）

```
sudo cat /var/lib/mysql-files/orders.csv
```

* 如果该路径为空，可以打开my.cnf 或 my.ini，加入以下语句后重启mysql

```
secure_file_priv=''
```

3. 导入

1）语法格式

```
load data infile '备份文件路径'
into table 表名
fields terminated by '字段分隔符'
lines terminated by '行分割符'
```

2）示例

```
load data infile '/var/lib/mysql-files/orders.csv'
into table orders
fields terminated by ','
lines terminated by '\n';
```

（四）表的复制和重命名

1）复制

- 将源表完全复制为新表

  create table orders_new select * from orders;

- 将源表部分复制到新表

  create table orders_new select * from orders where amt<= 200;

- 只复制表结构，不复制数据

  create table orders_new select * from orders where 1=0;

  * 注意，这种方式复制不会把键的属性复制过来

2）重命名

- 格式：alter table 表名 rename to 新表名
- 示例：alter table orders rename to orders_new;

（五）总结与回顾

1. 子查询
   - 一个查询中嵌套另一个查询
   - 当一个查询不方便或无法实现时使用子查询
   - 示例：
     - ✓ select * from orders where amt > (select avg(amt) from orders)
     - ✓ select * from customer where cust_id in
       (select distinct(cust_id) from orders)

2. 笛卡尔积
   - 集合的乘积，去掉无意义、不存在的组合即为关系

3. 联合查询

- 从两个或多个表中查询，返回一个查询结果集
- 当从一个表中无法查询到所有数据时使用联合查询
- 分类：内连接，外连接（左连接、右连接）
  - 内连接：关联不到的数据不显示
  - 外连接：关联不到的数据也显示，以左表为基准，数据全部显示，用右表去匹配为左连接；以右表为基准，数据全部显示，左表去匹配称为右连接

4. 约束：数据库层面的规则检测

- 非空约束：字段的值不能为空

  字段名称 数据类型 not null

- 唯一约束：字段的值必须唯一

  字段名称 数据类型 unique

- 主键约束：字段作为主键，非空、唯一

  字段名称 数据类型 Primary Key

- 默认约束：未填写值的情况下，自动填写默认值

  字段名称 数据类型 default 默认值

- 自动增加：字段值自动增加

  字段名称 数据类型 auto_increment

- 外键约束：某个属性在当前表中不是PK，在另外的表中是PK，为了保证参照完整性

  constraint 外键名称 foreign key(当前表字段名) references 参照表(参照字段)

5. 导出

select查询语句
into outfile '文件名称'
fields terminated by '字段分隔符'
lines terminated by '行分隔符'

6. 导入

load data infile '备份文件路径'
into table 表名
fields terminated by '字段分隔符'
lines terminated by '行分割符'

7. 表复制和重命名

```
create table orders_new select * from orders;
create table orders_new select * from orders where amt<= 200;
create table orders_new select * from orders where 1=0;
alter table orders rename to orders_new;
```