MySQL表的增删改查（进阶）

本节目标：

> - 数据库约束
> - 表的关系
> - 新增：
> - 删除
> - 修改
> - 查询

1. 数据库约束

1.1 约束类型

- **NOT NULL** - 指示某列不能存储 NULL 值。
- **UNIQUE** - 保证某列的每行必须有唯一的值。
- **DEFAULT** - 规定没有给列赋值时的默认值。
- **PRIMARY KEY** - NOT NULL 和 UNIQUE 的结合。确保某列（或两个列多个列的结合）有唯一标识，有助于更容易更快速地找到表中的一个特定的记录。
- **FOREIGN KEY** - 保证一个表中的数据匹配另一个表中的值的参照完整性。
- **CHECK** - 保证列中的值符合指定的条件。对于MySQL数据库，对CHECK子句进行分析，但是忽略CHECK子句。

1.2 NULL约束

创建表时，可以指定某列不为空：

```
-- 重新设置学生表结构
DROP TABLE IF EXISTS student;
CREATE TABLE student (
    id INT NOT NULL,
    sn INT,
    name VARCHAR(20),
    qq_mail VARCHAR(20)
);
```

1.3 UNIQUE：唯一约束

指定sn列为唯一的、不重复的：

```
-- 重新设置学生表结构
DROP TABLE IF EXISTS student;
CREATE TABLE student (
    id INT NOT NULL,
    sn INT UNIQUE,
    name VARCHAR(20),
    qq_mail VARCHAR(20)
);
```

1.4 DEFAULT：默认值约束

指定插入数据时，name列为空，默认值unkown：

```
-- 重新设置学生表结构
DROP TABLE IF EXISTS student;
CREATE TABLE student (
    id INT NOT NULL,
    sn INT UNIQUE,
    name VARCHAR(20) DEFAULT 'unkown',
    qq_mail VARCHAR(20)
);
```

1.5 PRIMARY KEY：主键约束

指定id列为主键：

```
-- 重新设置学生表结构
DROP TABLE IF EXISTS student;
CREATE TABLE student (
    id INT NOT NULL PRIMARY KEY,
    sn INT UNIQUE,
    name VARCHAR(20) DEFAULT 'unkown',
    qq_mail VARCHAR(20)
);
```

对于整数类型的主键，常配搭自增长auto_increment来使用。插入数据对应字段不给值时，使用最大值+1。

```
-- 主键是 NOT NULL 和 UNIQUE 的结合，可以不用 NOT NULL
id INT PRIMARY KEY auto_increment,
```

1.6 FOREIGN KEY：外键约束

外键用于关联其他表的**主键**或**唯一键**，语法：

```
foreign key (字段名) references 主表(列)
```

案例：

- 创建班级表classes，id为主键：

  ```
  -- 创建班级表，有使用MySQL关键字作为字段时，需要使用``来标识
  DROP TABLE IF EXISTS classes;
  CREATE TABLE classes (
      id INT PRIMARY KEY auto_increment,
      name VARCHAR(20),
      `desc` VARCHAR(100)
  );
  ```

- 创建学生表student，一个学生对应一个班级，一个班级对应多个学生。使用id为主键，classes_id为外键，关联班级表id

```sql
-- 重新设置学生表结构
DROP TABLE IF EXISTS student;
CREATE TABLE student (
    id INT PRIMARY KEY auto_increment,
    sn INT UNIQUE,
    name VARCHAR(20) DEFAULT 'unkown',
    qq_mail VARCHAR(20),
    classes_id int,
    FOREIGN KEY (classes_id) REFERENCES classes(id)
);
```

1.7 CHECK约束（了解）

MySQL使用时不报错，但忽略该约束：

```sql
drop table if exists test_user;
create table test_user (
   id int,
   name varchar(20),
   sex varchar(1),
   check (sex ='男' or sex='女')
);
```

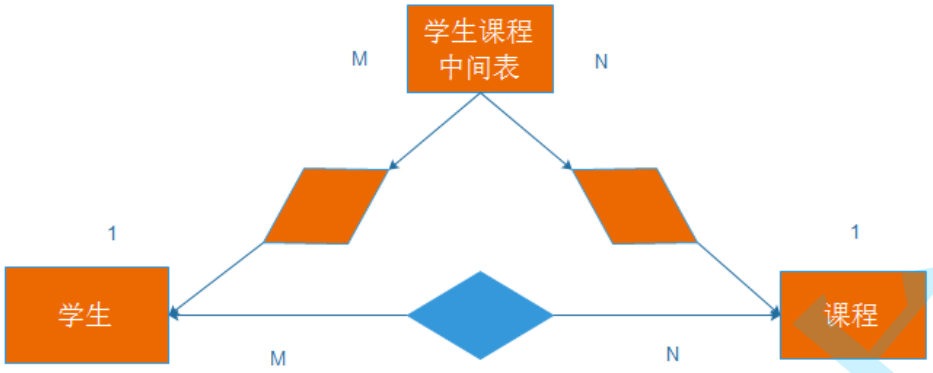
2. 表的设计

三大范式：

2.1 一对一

2.2 一对多

2.3 多对多



- 创建课程表

```
-- 创建课程表
DROP TABLE IF EXISTS course;
CREATE TABLE course (
    id INT PRIMARY KEY auto_increment,
    name VARCHAR(20)
);
```

- 创建学生课程中间表，考试成绩表

```
-- 创建课程学生中间表：考试成绩表
DROP TABLE IF EXISTS score;
CREATE TABLE score (
    id INT PRIMARY KEY auto_increment,
    score DECIMAL(3, 1),
    student_id int,
    course_id int,
    FOREIGN KEY (student_id) REFERENCES student(id),
    FOREIGN KEY (course_id) REFERENCES course(id)
);
```

3. 新增

插入查询结果

语法：

```
INSERT INTO table_name [(column [, column ...])] SELECT ...
```

案例：创建一张用户表，设计有name姓名、email邮箱、sex性别、mobile手机号字段。需要把已有的学生数据复制进来，可以复制的字段为name、qq_mail

```sql
-- 创建用户表
DROP TABLE IF EXISTS test_user;
CREATE TABLE test_user (
    id INT primary key auto_increment,
    name VARCHAR(20) comment '姓名',
    age INT comment '年龄',
    email VARCHAR(20) comment '邮箱',
    sex varchar(1) comment '性别',
    mobile varchar(20) comment '手机号'
);

-- 将学生表中的所有数据复制到用户表
insert into test_user(name, email) select name, qq_mail from student;
```

4. 查询

4.1 聚合查询

4.1.1 聚合函数

常见的统计总数、计算平局值等操作，可以使用聚合函数来实现，常见的聚合函数有：

函数	说明
COUNT([DISTINCT] expr)	返回查询到的数据的 数量
SUM([DISTINCT] expr)	返回查询到的数据的 总和，不是数字没有意义
AVG([DISTINCT] expr)	返回查询到的数据的 平均值，不是数字没有意义
MAX([DISTINCT] expr)	返回查询到的数据的 最大值，不是数字没有意义
MIN([DISTINCT] expr)	返回查询到的数据的 最小值，不是数字没有意义

案例：

- COUNT

```sql
-- 统计班级共有多少同学
SELECT COUNT(*) FROM student;
SELECT COUNT(0) FROM student;

-- 统计班级收集的 qq_mail 有多少个，qq_mail 为 NULL 的数据不会计入结果
SELECT COUNT(qq_mail) FROM student;
```

- SUM

```sql
-- 统计数学成绩总分
SELECT SUM(math) FROM exam_result;

-- 不及格 < 60 的总分，没有结果，返回 NULL
SELECT SUM(math) FROM exam_result WHERE math < 60;
```

- AVG

```
-- 统计平均总分
SELECT AVG(chinese + math + english) 平均总分 FROM exam_result;
```

- MAX

```
-- 返回英语最高分
SELECT MAX(english) FROM exam_result;
```

- MIN

```
-- 返回 > 70 分以上的数学最低分
SELECT MIN(math) FROM exam_result WHERE math > 70;
```

4.1.2 GROUP BY子句

SELECT 中使用 GROUP BY 子句可以对指定列进行分组查询。需要满足：使用 GROUP BY 进行分组查询时，SELECT 指定的字段必须是“分组依据字段”，其他字段若想出现在SELECT 中则必须包含在聚合函数中。

```
select column1, sum(column2), .. from table group by column1,column3;
```

案例：

- 准备测试表及数据：职员表，有id（主键）、name（姓名）、role（角色）、salary（薪水）

```
create table emp(
    id int primary key auto_increment,
    name varchar(20) not null,
    role varchar(20) not null,
    salary numeric(11,2)
);

insert into emp(name, role, salary) values
('马云','服务员', 1000.20),
('马化腾','游戏陪玩', 2000.99),
('孙悟空','游戏角色', 999.11),
('猪无能','游戏角色', 333.5),
('沙和尚','游戏角色', 700.33),
('隔壁老王','董事长', 12000.66);
```

- 查询每个角色的最高工资、最低工资和平均工资

```
select role,max(salary),min(salary),avg(salary) from emp group by role;
```

4.1.3 HAVING

GROUP BY 子句进行分组以后，需要对分组结果再进行条件过滤时，不能使用 WHERE 语句，而需要用 HAVING

- 显示平均工资低于1500的角色和它的平均工资

```
select role,max(salary),min(salary),avg(salary) from emp group by role
having avg(salary)<1500;
```

4.2 联合查询

实际开发中往往数据来自不同的表，所以需要多表联合查询。多表查询是对多张表的数据取笛卡尔积：

aaa

a1	a2
a	aa
b	bb

bbb

b1	b2
1	11
2	22

a	aa	1	11
a	aa	2	22
b	bb	1	11
b	bb	2	22

注意：关联查询可以对关联表使用别名。

初始化测试数据：

```
insert into classes(name, `desc`) values
('计算机系2019级1班', '学习了计算机原理、C和Java语言、数据结构和算法'),
('中文系2019级3班','学习了中国传统文学'),
('自动化2019级5班','学习了机械自动化');

insert into student(sn, name, qq_mail, classes_id) values
('09982','黑旋风李逵','xuanfeng@qq.com',1),
('00835','菩提老祖',null,1),
('00391','白素贞',null,1),
('00031','许仙','xuxian@qq.com',1),
('00054','不想毕业',null,1),
('51234','好好说话','say@qq.com',2),
('83223','tellme',null,2),
('09527','老外学中文','foreigner@qq.com',2);

insert into course(name) values
('Java'),('中国传统文化'),('计算机原理'),('语文'),('高阶数学'),('英文');

insert into score(score, student_id, course_id) values
-- 黑旋风李逵
(70.5, 1, 1),(98.5, 1, 3),(33, 1, 5),(98, 1, 6),
-- 菩提老祖
(60, 2, 1),(59.5, 2, 5),
-- 白素贞
(33, 3, 1),(68, 3, 3),(99, 3, 5),
-- 许仙
(67, 4, 1),(23, 4, 3),(56, 4, 5),(72, 4, 6),
-- 不想毕业
(81, 5, 1),(37, 5, 5),
```

```
-- 好好说话
(56, 6, 2),(43, 6, 4),(79, 6, 6),
-- tellme
(80, 7, 2),(92, 7, 6);
```

4.2.1 内连接

语法：

```
select 字段 from 表1 别名1 [inner] join 表2 别名2 on 连接条件 and 其他条件;
select 字段 from 表1 别名1,表2 别名2 where 连接条件 and 其他条件;
```

案例：

（1）查询“许仙”同学的 成绩

```
select sco.score from student stu inner join score sco on stu.id=sco.student_id
and stu.name='许仙';
-- 或者
select sco.score from student stu, score sco where stu.id=sco.student_id and
stu.name='许仙';
```

（2）查询所有同学的总成绩，及同学的个人信息：

```
-- 成绩表对学生表是多对1关系，查询总成绩是根据成绩表的同学id来进行分组的
SELECT
    stu.sn,
    stu.NAME,
    stu.qq_mail,
    sum( sco.score )
FROM
    student stu
    JOIN score sco ON stu.id = sco.student_id
GROUP BY
    sco.student_id;
```

（3）查询所有同学的成绩，及同学的个人信息：

```
-- 查询出来的都是有成绩的同学，“老外学中文”同学 没有显示
select * from student stu join score sco on stu.id=sco.student_id;

-- 学生表、成绩表、课程表3张表关联查询
SELECT
    stu.id,
    stu.sn,
    stu.NAME,
    stu.qq_mail,
    sco.score,
    sco.course_id,
    cou.NAME
FROM
    student stu
    JOIN score sco ON stu.id = sco.student_id
    JOIN course cou ON sco.course_id = cou.id
ORDER BY
```

```
    stu.id;
```

4.2.2 外连接

外连接分为左外连接和右外连接。如果联合查询，左侧的表完全显示我们就说是左外连接；右侧的表完全显示我们就说是右外连接。

语法：

```
-- 左外连接，表1完全显示
select 字段名  from 表名1 left join 表名2 on 连接条件;

-- 右外连接，表2完全显示
select 字段 from 表名1 right join 表名2 on  连接条件;
```

案例：查询所有同学的成绩，及同学的个人信息，如果该同学没有成绩，也需要显示

```
-- “老外学中文”同学 没有考试成绩，也显示出来了
select * from student stu left join score sco on stu.id=sco.student_id;
-- 对应的右外连接为：
select * from  score sco right join student stu on stu.id=sco.student_id;

-- 学生表、成绩表、课程表3张表关联查询
SELECT
    stu.id,
    stu.sn,
    stu.NAME,
    stu.qq_mail,
    sco.score,
    sco.course_id,
    cou.NAME
FROM
    student stu
    LEFT JOIN score sco ON stu.id = sco.student_id
    LEFT JOIN course cou ON sco.course_id = cou.id
ORDER BY
    stu.id;
```

4.2.3 自连接

自连接是指在同一张表连接自身进行查询。

案例：

显示所有“计算机原理”成绩比“Java”成绩高的成绩信息

```
-- 先查询“计算机原理”和“Java”课程的id
select id,name from course where name='Java' or name='计算机原理';

-- 再查询成绩表中，“计算机原理”成绩比“Java”成绩 好的信息
SELECT
    s1.*
FROM
    score s1,
    score s2
WHERE
    s1.student_id = s2.student_id
```

```
    AND s1.score < s2.score
    AND s1.course_id = 1
    AND s2.course_id = 3;

-- 也可以使用join on 语句来进行自连接查询
SELECT
    s1.*
FROM
    score s1
    JOIN score s2 ON s1.student_id = s2.student_id
    AND s1.score < s2.score
    AND s1.course_id = 1
    AND s2.course_id = 3;
```

以上查询只显示了成绩信息，并且是分布执行的。要显示学生及成绩信息，并在一条语句显示：

```
SELECT
    stu.*,
    s1.score Java,
    s2.score 计算机原理
FROM
    score s1
    JOIN score s2 ON s1.student_id = s2.student_id
    JOIN student stu ON s1.student_id = stu.id
    JOIN course c1 ON s1.course_id = c1.id
    JOIN course c2 ON s2.course_id = c2.id
    AND s1.score < s2.score
    AND c1.NAME = 'Java'
    AND c2.NAME = '计算机原理';
```

4.2.4 子查询

子查询是指嵌入在其他sql语句中的select语句，也叫嵌套查询

- 单行子查询：返回一行记录的子查询

  查询与“不想毕业” 同学的同班同学：

  ```
  select * from student where classes_id=(select classes_id from student where
  name='不想毕业');
  ```

- 多行子查询：返回多行记录的子查询

  案例：查询“语文”或“英文”课程的成绩信息

  1. **[NOT] IN**关键字：

     ```
     -- 使用IN
     select * from score where course_id in (select id from course where
     name='语文' or name='英文');

     -- 使用 NOT IN
     select * from score where course_id not in (select id from course where
     name!='语文' and name!='英文');
     ```

     可以使用多列包含：

```
-- 插入重复的分数：score，student_id，course_id列重复
insert into score(score, student_id, course_id) values
-- 黑旋风李逵
(70.5, 1, 1),(98.5, 1, 3),
-- 菩提老祖
(60, 2, 1);

-- 查询重复的分数
SELECT
    *
FROM
    score
WHERE
    ( score, student_id, course_id ) IN ( SELECT score, student_id,
course_id FROM score GROUP BY score, student_id, course_id HAVING
count( 0 ) > 1 );
```

2. **[NOT] EXISTS**关键字：

```
 -- 使用 EXISTS
select * from score sco where exists (select sco.id from course cou
where (name='语文' or name='英文') and cou.id = sco.course_id);

-- 使用 NOT EXISTS
select * from score sco where not exists (select sco.id from course cou
where (name!='语文' and name!='英文') and cou.id = sco.course_id);
```

- 在from子句中使用子查询：子查询语句出现在from子句中。这里要用到数据查询的技巧，把一个子查询当做一个临时表使用。

  查询所有比“中文系2019级3班”平均分高的成绩信息：

```
-- 获取“中文系2019级3班”的平均分，将其看作临时表
SELECT
    avg( sco.score ) score
FROM
    score sco
    JOIN student stu ON sco.student_id = stu.id
    JOIN classes cls ON stu.classes_id = cls.id
WHERE
    cls.NAME = '中文系2019级3班';
```

  查询成绩表中，比以上临时表平均分高的成绩：

```
SELECT
    *
FROM
    score sco,
    (
    SELECT
        avg( sco.score ) score
    FROM
        score sco
        JOIN student stu ON sco.student_id = stu.id
        JOIN classes cls ON stu.classes_id = cls.id
    WHERE
```

```
        cls.NAME = '中文系2019级3班'
    ) tmp
WHERE
    sco.score > tmp.score;
```

4.2.5 合并查询

在实际应用中，为了合并多个select的执行结果，可以使用集合操作符 union，union all。使用UNION和UNION ALL时，前后查询的结果集中，字段需要一致。

- union

该操作符用于取得两个结果集的并集。当使用该操作符时，会自动去掉结果集中的重复行。

案例：查询id小于3，或者名字为“英文”的课程：

```
select * from course where id<3
union
select * from course where name='英文';

-- 或者使用or来实现
select * from course where id<3 or name='英文';
```

- union all

该操作符用于取得两个结果集的并集。当使用该操作符时，不会去掉结果集中的重复行。

案例：查询id小于3，或者名字为“Java”的课程

```
-- 可以看到结果集中出现重复数据Java
select * from course where id<3
union all
select * from course where name='英文';
```

5. 内容重点总结

- 数据库约束

约束类型	说明	示例

约束类型	说明	示例
NULL约束	使用NOT NULL指定列不为空	name varchar(20) not null,
UNIQUE唯一约束	指定列为唯一的、不重复的	name varchar(20) unique,
DEFAULT默认值约束	指定列为空时的默认值	age int default 20,
主键约束	NOT NULL 和 UNIQUE 的结合	id int primary key,
外键约束	关联其他表的**主键**或**唯一键**	foreign key (字段名) references 主表(列)
CHECK约束（了解）	保证列中的值符合指定的条件	check (sex ='男' or sex='女')

- 表的关系
  1. 一对一：
  2. 一对多：
  3. 多对多：需要创建中间表来映射两张表的关系
- 新增：

```
INSERT INTO table_name [(column [, column ...])] SELECT ...
```

- 查询
  1. 聚合函数：MAX、MIN、AVG、COUNT、SUM
  2. 分组查询：GROUP BY... HAVING ...
  3. 内连接：

```
select ... from 表1，表2 where 条件
-- inner可以缺省
select ... from 表1 join 表2 on 条件 where 其他条件
```

  4. 外连接：

```
 select ... from 表1 left/right join 表2 on 条件 where 其他条件
```

  5. 自连接：

```
 select ... from 表1，表1 where 条件
select ... from 表1 join 表1 on 条件
```

```
6. 子查询：

   ```sql
 -- 单行子查询
   select ... from 表1 where 字段1 = (select ... from ...);
```
```

```
  -- [NOT] IN
select ... from 表1 where 字段1 in (select ... from ...);

  -- [NOT] EXISTS
  select ... from 表1 where exists (select ... from ... where 条件);

  -- 临时表：form子句中的子查询
  select ... from 表1,  (select ... from ...) as tmp where 条件
```

7. 合并查询：

```
-- UNION：去除重复数据
select ... from ... where 条件
 union
select ... from ... where 条件

-- UNION ALL：不去重
 select ... from ... where 条件
union all
select ... from ... where 条件

-- 使用UNION和UNION ALL时，前后查询的结果集中，字段需要一致
```

SQL查询中各个关键字的执行先后顺序： from > on> join > where > group by > with > having > select > distinct > order by > limit

6. 课堂作业

- 设计图书管理系统，包含学生和图书信息，且图书可以进行分类，学生可以在一个时间范围内借阅图书，并在这个时间范围内归还图书。

  要求：

  1. 涉及以上场景的数据库表，并建立表关系。
  2. 查询某个分类下的图书借阅信息。
  3. 查询在某个时间之后的图书借阅信息。
  4. 查询图书借阅周期在某个时间范围内的图书借阅信息（图书借阅周期与查询时间范围有交集）。

- 设计网上商城的商品模块：包含商品SPU、SKU、动态属性。

  说明：SPU是指产品的某个型号，不涉及具体规格，如iPhone 11，包含一些共有属性，如CPU、屏幕材质等。而SKU是具体到规格的具体某个商品，如128G，黑色，无需合约版的iPhone 11。

  详情可参见以下商品：

  Apple iPhone11

  Apple iPhone11 Pro

  华为P30 Pro

  华为Mate20 Pro

  要求：

  1. 涉及以上场景的数据库表，并建立表关系。
  2. 查询以iPhone开头的黑色手机。