第二讲 约束与 CRUD 操作

一、增删改表中数据

1. 插入记录（录屏 5）

INSERT [INTO] 表名 [字段名] VALUES (字段值)

表名：表示往哪张表中添加数据

(字段名 1，字段名 2, …)：要给哪些字段设置值

VALUES (值 1，值 2, …)：设置具体的值

① 插入全部字段

- 所有的字段名都写出来

INSERT INTO 表名 (字段名 1，字段名 2，字段名 3…) VALUES (值 1，值 2，值 3);

- 不写字段名

INSERT INTO 表名 VALUES (值 1，值 2，值 3…);

② 插入部分数据

INSERT INTO 表名 (字段名 1，字段名 2, ...) VALUES (值 1，值 2, ...);

注：没有添加数据的字段会使用 NULL

举例：

```
-- 插入所有的列，向学生表中
insert into student (id,name,age,sex) values (1, '孙悟空', 20, '男');
insert into student (id,name,age,sex) values (2, '孙悟天', 16, '男');

-- 插入所有列
```

```
insert into student values (3, '孙悟饭', 18, '男', '龟仙人洞中');
-- 如果只插入部分列，必须写列名
insert into student values (3, '孙悟饭', 18, '男');
```

③ insert 的注意事项

- 插入的数据应与字段的数据类型相同
- 数据的大小应在列的规定范围内，例如：不能将一个长度为 80 的字符串加入到长度为 40 的列中。
- 在 values 中列出的数据位置必须与被加入的列的排列位置相对应。在 mysql 中可以使用 value，但不建议使用，功能与 values 相同。
- 字符和日期型数据应包含在单引号中。MySQL 中也可以使用双引号做为分隔符。
- 不指定列或使用 null，表示插入空值。

④ 蠕虫复制

将一张已经存在的表中的数据复制到另一张表中

将表名 2 中的所有的列复制到表名 1 中 INSERT INTO 表名 1 SELECT * FROM 表名 2;

只复制部分列 INSERT INTO 表名 1(列 1, 列 2) SELECT 列 1, 列 2 FROM 表 2;

举例：

```
-- 创建 student2 表，student2 结构和 student 表结构一样
drop table student2;
create table student2 like student;

-- 将 student 表中的数据添加到 student2 表中
insert into student2 select * from student;

-- 如果只想复制 student 表中 name,age 字段数据到 student2 表中，两张表都写出相应的列名
insert into student2 (name,age) select name,age from student;
```

2. 更新表记录(录屏 6)

UPDATE 表名 SET 列名=值 [WHERE 条件表达式]

UPDATE: 需要更新的表名

SET: 修改的列值

WHERE: 符合条件的记录才更新

① 不带条件修改数据

UPDATE 表名 SET 字段名=值; -- 修改所有的行

② 带条件修改数据

UPDATE 表名 SET 字段名=值 WHERE 字段名=值;

举例:

```
-- 不带条件修改数据，将所有的性别改成女
update student set sex = '女';

-- 带条件修改数据，将 id 号为 2 的学生性别改成男
update student set sex='男' where id=2;

-- 一次修改多个列，把 id 为 3 的学生，年龄改成 26 岁，address 改成北京
update student set age=26, address='北京' where id=3;
```

注意:

09:44:16 update student3 set address='江苏南京' Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect. 0.00034 sec

SET SQL_SAFE_UPDATES = 0;

3. 删除表记录（录屏 7）

DELETE FROM 表名 [WHERE 条件表达式]

如果没有指定 WHERE 子句，MySQL 表中的所有记录将被删除。你可以在 WHERE 子句中指定任何条件

① 不带条件删除数据

DELETE FROM 表名;

② 带条件删除数据

DELETE FROM 表名 WHERE 字段名=值;

③ 使用 truncate 删除表中所有记录

TRUNCATE TABLE 表名

④ truncate 和 delete 的区别

truncate 相当于删除表的结构，再创建一张表。

举例：

```
-- 带条件删除数据，删除 id 为 1 的记录
delete from student where id=1;


-- 不带条件删除数据,删除表中的所有数据
delete from student;
```

二、单表查询

SELECT 列名 FROM 表名 [WHERE 条件表达式]

SELECT 命令可以读取一行或者多行记录。

你可以使用星号（*）来代替其他字段，SELECT 语句会返回表的所有字段数据

你可以使用 WHERE 语句来包含任何条件。

测试用例

创建一个学生表，包含如下列：

```
CREATE TABLE student3 (
```

```
    id int,  -- 编号
    name varchar(20), -- 姓名
    age int, -- 年龄
    sex varchar(5),  -- 性别
    address varchar(100),  -- 地址
    math int, -- 数学
    english int -- 英语
);
```

准备数据

```
INSERT INTO student3(id,NAME,age,sex,address,math,english) VALUES (1,'马云',55,'男',' 杭州',66,78),(2,'马化腾',45,'女','深圳',98,87),(3,'马景涛',55,'男','香港',56,77),(4,'柳岩 ',20,'女','湖南',76,65),(5,'柳青',20,'男','湖南',86,NULL),(6,'刘德华',57,'男','香港 ',99,99),(7,'马德',22,'女','香港',99,99),(8,'德玛西亚',18,'男','南京',56,65);
```

1. 简单查询（录屏 8）

① 查询表所有行和列的数据

使用*表示所有列 **SELECT * FROM 表名;**

```
查询所有的学生：  select * from student;
```

② 查询指定列

查询指定列的数据,多个列之间以逗号分隔

SELECT 字段名 1, 字段名 2, 字段名 3, ... FROM 表名;

```
-- 查询 student 表中的 name 和 age 列
select name,age from student;
```

2. 指定列的别名进行查询

使用别名的好处： 显示的时候使用新的名字，并不修改表的结构。

语法：

- 对列指定别名

SELECT 字段名 1 AS 别名, 字段名 2 AS 别名... FROM 表名;

- 对列和表同时指定别名

SELECT 字段名 1 AS 别名, 字段名 2 AS 别名... FROM 表名 AS 表别名

举例:

```
-- 使用别名
select name as 姓名,age as 年龄 from student;
-- 表使用别名
select st.name as 姓名,age as 年龄 from student as st
```

表使用别名的原因：用于多表查询操作

3. 清除重复值

查询指定列并且结果不出现重复数据

SELECT DISTINCT 字段名 FROM 表名;

举例:

```
-- 查询学生来至于哪些地方
select address from student;


-- 去掉重复的记录
select distinct address from student;
```

4. 查询结果参与运算

- 某列数据和固定值运算

SELECT 列名 1 + 固定值 FROM 表名;

- 某列数据和其他列数据参与运算

SELECT 列名 1 + 列名 2 FROM 表名;

注意: 参与运算的必须是数值类型

举例:

```
-- 给所有的数学加 5 分
select math+5 from student;


-- 查询 math + english 的和
```

```
select *,(math+english) as 总成绩 from student; -- as 可以省略
select *,(math+english) 总成绩 from student;
```

5. 条件查询（录屏 9-1）

SELECT 字段名 FROM 表名 WHERE 条件;

流程：取出表中的每条数据，满足条件的记录就返回，不满足条件的记录不返回

① 比较运算符

>、<、<=、>=、=、<>

<>在 SQL 中表示不等于，在 mysql 中也可以使用!= 没有==

IS NULL 查询某一列为 NULL 的值，注：不能写=NULL

举例：

```
-- 查询 math 分数大于 80 分的学生
select * from student3 where math>80;

-- 查询 english 分数小于或等于 80 分的学生
select * from student3 where english <=80;

-- 查询 age 等于 20 岁的学生
select * from student3 where age = 20;

-- 查询 age 不等于 20 岁的学生，注：不等于有两种写法
select * from student3 where age <> 20;
select * from student3 where age != 20;
```

② 逻辑运算符

and 或 &&,

or 或 ||

not 或 !

SQL 中建议使用前者，后者并不通用。

举例：

```
-- 查询 age 大于 35 且性别为男的学生(两个条件同时满足)
select * from student3 where age>35 and sex='男';

-- 查询 age 大于 35 或性别为男的学生(两个条件其中一个满足)
select * from student3 where age>35 or sex='男';

-- 查询 id 是 1 或 3 或 5 的学生
select * from student3 where id=1 or id=3 or id=5;
```

③ in 关键字 （录屏 9-2）

SELECT 字段名 FROM 表名 WHERE 字段 in (数据 1, 数据 2...);
IN(集合) 集合表示多个值，使用逗号分隔
in 里面的每个数据都会作为一次条件，只要满足条件的就会显示
举例：

```
-- 查询 id 是 1 或 3 或 5 的学生
select * from student3 where id in(1,3,5);

-- 查询 id 不是 1 或 3 或 5 的学生
select * from student3 where id not in(1,3,5);
```

④ 范围查询

BETWEEN 值 1 AND 值 2 表示从值 1 到值 2 范围，包头又包尾
比如：age BETWEEN 80 AND 100 相当于： age<=100 and age>=80
举例：

```
查询 english 成绩大于等于 75，且小于等于 90 的学生
select * from student3 where english between 75 and 90;
```

⑤ like 关键字 （录屏 9-3）

LIKE 表示模糊查询
SELECT * FROM 表名 WHERE 字段名 LIKE '通配符字符串';

MySQL 通配符

通配符	说明
%	匹配任意多个字符串
_	匹配一个字符

举例：

```
-- 查询姓马的学生
select * from student3 where name like '马%';
select * from student3 where name like '马';

-- 查询姓名中包含'德'字的学生
select * from student3 where name like '%德%';

-- 查询姓马，且姓名有两个字的学生
select * from student3 where name like '马_';
```

6. 排序（录屏 10）

通过 ORDER BY 子句，可以将查询出的结果进行排序(排序只是显示方式，不会影响数据库中数据的顺序)

SELECT 字段名 FROM 表名 WHERE 字段=值 ORDER BY 字段名 [ASC|DESC];

ASC: 升序，默认值 DESC: 降序

① 单列排序

只按某一个字段进行排序

```
-- 查询所有数据,使用年龄降序排序
select * from student order by age desc;
```

② 组合排序

同时对多个字段进行排序，如果第 1 个字段相等，则按第 2 个字段排序，依次类推。

语法： **SELECT 字段名 FROM 表名 WHERE 字段=值 ORDER BY 字段名 1 [ASC|DESC], 字段名 2 [ASC|DESC];**

```
-- 查询所有数据,在年龄降序排序的基础上，如果年龄相同再以数学成绩升序排序
```

```
select * from student order by age desc, math asc;
```

7. 聚合函数（录屏 11）

之前我们做的查询都是横向查询，它们都是根据条件一行一行的进行判断，而使用聚合函数查询是纵向查询， 它是对一列的值进行计算，然后返回一个结果值。聚合函数会忽略空值 NULL。

五个聚合函数

SQL 中的聚合函数	作用
max(列名)	求这一列的最大值
min(列名)	求这一列的最小值
avg(列名)	求这一列的平均值
count(列名)	统计这一列有多少条记录
sum(列名)	对这一列求总和

语法： SELECT 聚合函数(列名) FROM 表名;

```
-- 查询学生总数
select count(id) as 总人数 from student;
select count(*) as 总人数 from student;
```

我们发现对于 NULL 的记录不会统计，建议如果统计个数则不要使用有可能为 null 的列，但如果需要把 NULL 也统计进去呢？

IFNULL(列名，默认值)

如果列名不为空，返回这列的值。如果为 NULL，则返回默认值。

```
-- 查询 id 字段，如果为 null，则使用 0 代替
select ifnull(id,0) from student;
```

我们可以利用 IFNULL()函数，如果记录为 NULL，给个默认值，这样统计的数据就不会遗漏

```
select count(ifnull(id,0)) from student;
```

举例：

```
-- 查询年龄大于 20 的总数
select count(*) from student where age>20;
-- 查询数学成绩总分
```

```
select sum(math) 总分 from student;
-- 查询数学成绩平均分
select avg(math) 平均分 from student;
-- 查询数学成绩最高分
select max(math) 最高分 from student;
-- 查询数学成绩最低分
select min(math) 最低分 from student;
```

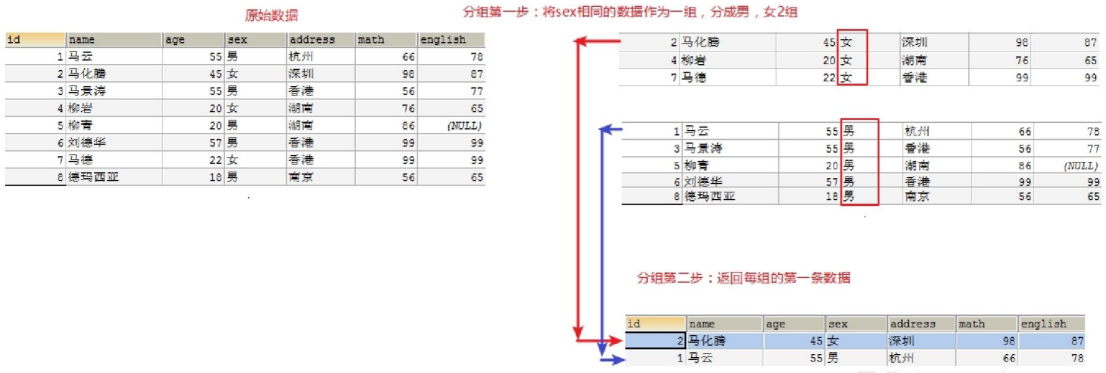
8. 分组（录屏 12）

分组查询是指使用 GROUP BY 语句对查询信息进行分组，相同数据作为一组

SELECT 字段 1,字段 2... FROM 表名 GROUP BY 分组字段 [HAVING 条件];

① GROUP BY 怎么分组的?

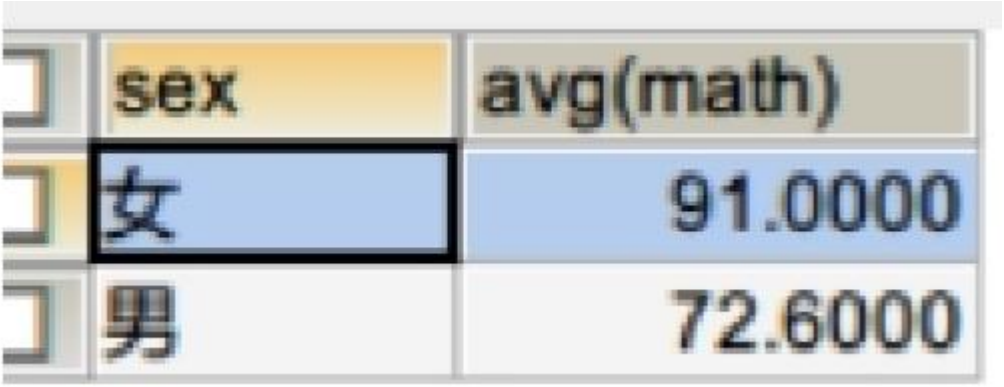
将分组字段结果中相同内容作为一组，如按性别将学生分成 2 组。



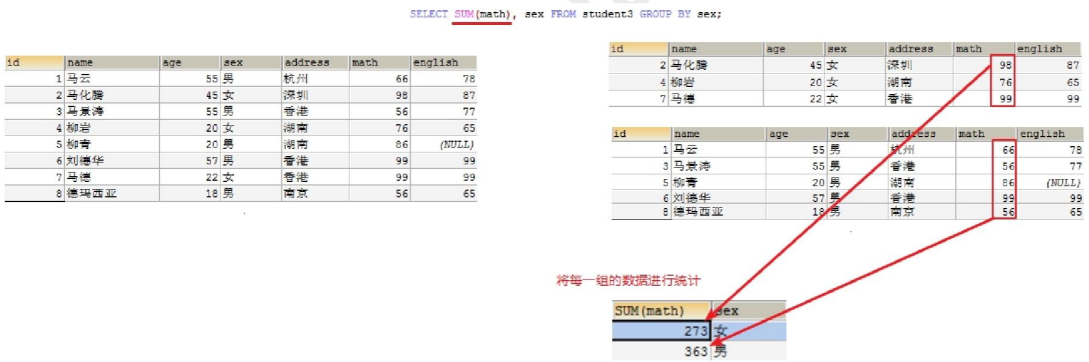
GROUP BY 将分组字段结果中相同内容作为一组，并且返回每组的第一条数据，所以单独分组没什么用处。 分组的目的就是为了统计，一般分组会跟聚合函数一起使用。

```
-- 按性别进行分组，求男生和女生数学的平均分
select sex, avg(math) from student3 group by sex;
```

分组查询结果：



实际上是将每组的 math 求了平均,返回每组统计的结果



注意：当我们使用某个字段分组,在查询的时候也需要将这个字段查询出来,否则看不到数据属于哪组的

举例：

- 查询男女各多少人 1) 查询所有数据,按性别分组。 2) 统计每组人数

```
select sex, count(*) from student3 group by sex;
```

- 查询年龄大于 25 岁的人,按性别分组,统计每组的人数

1) 先过滤掉年龄小于 25 岁的人。 2) 再分组。 3) 最后统计每组的人数

```
select sex, count(*) from student3 where age > 25 group by sex ;
```

② having 与 where 的区别

- 查询年龄大于 25 岁的人，按性别分组，统计每组的人数，并只显示性别人数大于 2 的数据 以下代码是否正确？

```
SELECT sex, COUNT(*) FROM student3 WHERE age > 25 GROUP BY sex WHERE COUNT(*) >2;
```

正确写法：

```
-- 对分组查询的结果再进行过滤
SELECT sex, COUNT(*) FROM student3 WHERE age > 25 GROUP BY sex having COUNT(*) >2;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

1.先过滤掉年龄小于25岁的人。

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
6	刘德华	57	男	香港	99	99

2.再分组。

id	name	age	sex	address	math	english
2	马化腾	45	女	深圳	98	87

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
3	马景涛	55	男	香港	56	77
6	刘德华	57	男	香港	99	99

3.统计每组的人数

sex	COUNT(*)
女	1
男	3

4.显示性别人数大于2的数据

sex	COUNT(*)
男	3

子句	作用
where 子句	1) 对查询结果进行分组前，将不符合 where 条件的行去掉，即在分组之前过滤数据，即先过滤 再分组。 2) where 后面不可以使用聚合函数
having 子句	1) having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，即先分组再过滤。 2) having 后面可以使用聚合函数

9. limit 语句(录屏 13)

limit 的作用： LIMIT 是限制的意思，所以 LIMIT 的作用就是限制查询记录的条数。

SELECT *|字段列表 [as 别名] FROM 表名 [WHERE 子句] [GROUP BY 子句][HAVING 子句][ORDER BY 子 句][LIMIT 子句];

LIMIT 语法格式： **LIMIT offset,length;**

offset：起始行数，从 0 开始计数，如果省略，默认就是 0

length： 返回的行数

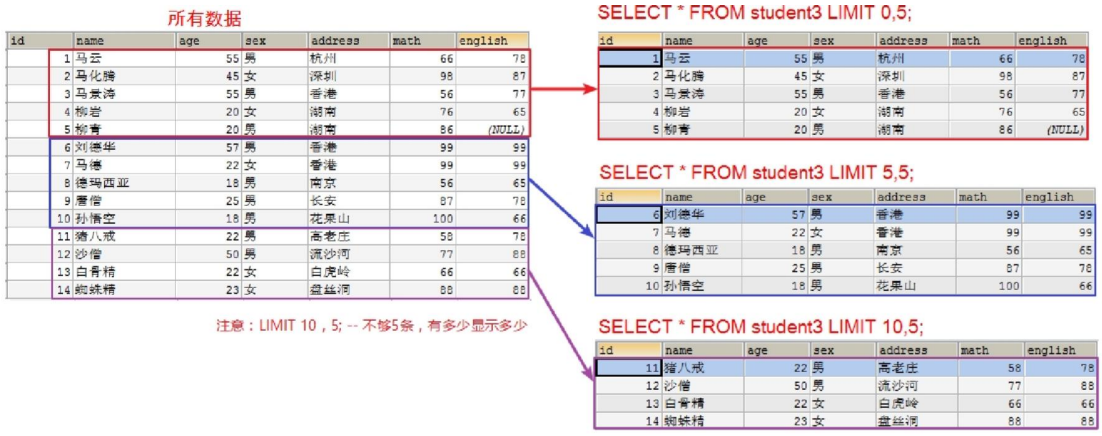
举例：

```
-- 查询学生表中数据，从第 3 条开始显示，显示 6 条。
select * from student3 limit 2,6;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65
9	唐僧	25	男	长安	87	78
10	孙悟空	18	男	花果山	100	66
11	猪八戒	22	男	高老庄	58	78
12	沙僧	50	男	流沙河	77	88
13	白骨精	22	女	白虎岭	66	66
14	蜘蛛精	23	女	盘丝洞	88	88

id	name	age	sex	address	math	english
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

LIMIT 的使用场景： 分页：比如我们登录京东，淘宝，返回的商品信息可能有几万条，不是一次全部显示出来。是一页显示固定的 条数。 假设我们每页显示 5 条记录的方式来分页。



id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)

id	name	age	sex	address	math	english
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65
9	唐僧	25	男	长安	87	78
10	孙悟空	18	男	花果山	100	66

id	name	age	sex	address	math	english
11	猪八戒	22	男	高老庄	58	78
12	沙僧	50	男	流沙河	77	88
13	白骨精	22	女	白虎岭	66	66
14	蜘蛛精	23	女	盘丝洞	88	88

特例：

```
-- 如果第一个参数是 0 可以省略写：
select * from student3 limit 5;
-- 最后如果不够 5 条，有多少显示多少
select * from student3 limit 10,5;
```

三、数据库表的约束

1. 约束的作用

对表中的数据进行限制，保证数据的正确性、有效性和完整性。
一个表如果添加了约束，不正确的数据将无法插入到表中。约束在创建表的时候添加比较合适。

2. 约束种类

约束名	约束关键字
主键	primary key
唯一	unique
非空	not null
外键	foreign key
检查约束	check 注：mysql 不支持

3. 主键约束（录屏 14）

① 主键的作用

用来唯一标识数据库中的每一条记录

② 哪个字段应该作为表的主键?

通常不用业务字段作为主键，单独给每张表设计一个 id 的字段，把 id 作为主键。主键是给数据库和程序使用 的，不是给最终的客户使用的。所以主键有没有含义没有关系，只要不重复，非空就行。
如：身份证，学号不建议做成主键

③ 创建主键

主键关键字： primary key

主键的特点：

1) 非空 not null 2) 唯一

创建主键方式：

- 在创建表的时候给字段添加主键

  字段名 字段类型 PRIMARY KEY

- 在已有表中添加主键

  ALTER TABLE 表名 ADD PRIMARY KEY(字段名);

举例：

```
-- 创建表学生表 st5, 包含字段(id, name, age)将 id 做为主键
create table st5 (
    id int primary key,   -- id 为主键
    name varchar(20),
    age int
 )
 desc st5;
```

```
-- 插入重复的主键值
insert into st5 values (1, '关羽', 30);
 -- 错误代码： 1062 Duplicate entry '1' for key 'PRIMARY'
insert into st5 values (1, '关云长', 20);
 select * from st5;

-- 插入 NULL 的主键值, Column 'id' cannot be null
insert into st5 values (null, '关云长', 20);
```

④ 删除主键

```
-- 删除 st5 表的主键
alter table st5 drop primary key;
-- 添加主键
```

```
alter table st5 add primary key(id);
```

⑤ 主键自增

主键如果让我们自己添加很有可能重复,我们通常希望在每次插入新记录时,数据库自动生成主键字段的值

AUTO_INCREMENT 表示自动增长(字段类型必须是整数类型)

- 创建表时指定自增长列

CREATE TABLE 表名(
列名 int primary key AUTO_INCREMENT
) AUTO_INCREMENT=起始值;

默认地 AUTO_INCREMENT 的开始值是 1，如果希望修改起始值,请使用下列 SQL 语法

举例:

```
-- 指定起始值为 1000
create table st4 (
    id int primary key auto_increment,
    name varchar(20)
 ) auto_increment = 1000;

insert into st4 values (null, '孔明');
select * from st4;
```

- 创建好以后修改起始值

ALTER TABLE 表名 AUTO_INCREMENT=起始值;

举例:

```
alter table st4 auto_increment = 2000;
insert into st4 values (null, '刘备');
```

⑥ DELETE 和 TRUNCATE 对自增长的影响

- DELETE：删除所有的记录之后，自增长没有影响。

id	name	age
4	小乔	18
5	大乔	20
6	周瑜	35

- TRUNCATE：删除以后，自增长又重新开始。

id	name	age
1	小乔	18
2	大乔	20
3	周瑜	35

4. 唯一约束（录屏 15）

表中某一列不能出现重复的值

① 基本格式

字段名 字段类型 UNIQUE

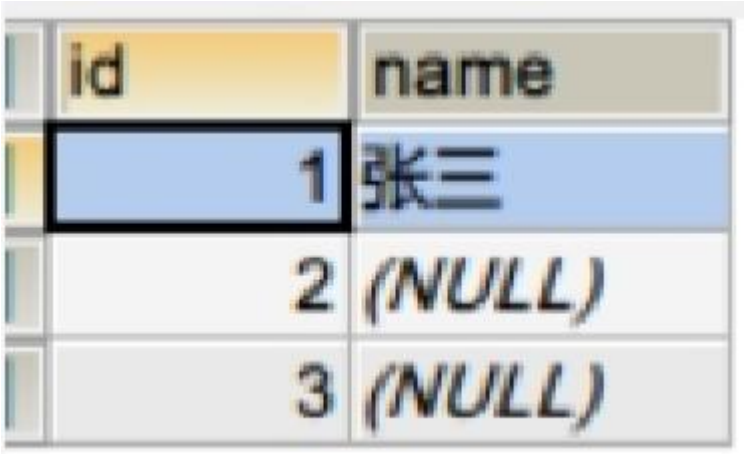
字段名 字段类型 UNIQUE

举例：

```
-- 创建学生表 st7，包含字段(id, name),name 这一列设置唯一约束,不能出现同名的学生
create table st7 (
    id int,
    name varchar(20) unique
 )
```

```
-- 添加一个同名的学生
insert into st7 values (1, '张三');
 select * from st7;
-- Duplicate entry '张三' for key 'name'
insert into st7 values (2, '张三');

-- 重复插入多个 null 会怎样?
 insert into st7 values (2, null);
 insert into st7 values (3, null);
```



id	name
1	张三
2	(NULL)
3	(NULL)

null 没有数据，不存在重复的问题

5. 非空约束 （录屏 16）

某一列不能为 null

① 基本语法格式

字段名 字段类型 NOT NULL

举例：

```
-- 创建表学生表 st8，包含字段(id,name,gender)其中 name 不能为 NULL
create table st8 (
   id int,
   name varchar(20) not null,
   gender char(1)
 )
```

```
-- 添加一条记录其中姓名不赋值
insert into st8 values (1,'张三疯','男');
  select * from st8;

-- Column 'name' cannot be null
  insert into st8 values (2,null,'男');
```

② 默认值

字段名 字段类型 DEFAULT 默认值

举例：

```
-- 创建一个学生表 st9，包含字段(id,name,address)，  地址默认值是广州
create table st9 (
    id int,
    name varchar(20),
    address varchar(20)   default '广州'
)

-- 添加一条记录,使用默认地址
insert into st9 values (1, '李四', default);
select * from st9;

insert into st9 (id,name) values (2, '李白');

-- 添加一条记录,不使用默认地址
insert into st9 values (3, '李四光', '深圳');
```

- 疑问:如果一个字段设置了非空与唯一约束，该字段与主键的区别?

1) 主键数在一个表中，只能有一个。不能出现多个主键。主键可以单列，也可以是多列。
2) 自增长只能用在主键上

6. 外键约束（录屏 17）

① 单表的缺点

创建一个员工表包含如下列(id, name, age, dep_name, dep_location),

id 主键并自动增长,添加 5 条数据

```
CREATE TABLE emp (
   id INT PRIMARY KEY AUTO_INCREMENT,
   NAME VARCHAR(30),
   age INT,
   dep_name VARCHAR(30),
   dep_location VARCHAR(30)
 );
 -- 添加数据
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('张三', 20, '研发部', '广州
');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('李四', 21, '研发部', '广州
');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('王五', 20, '研发部', '广州
');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('老王', 20, '销售部', '深圳
');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('大王', 22, '销售部', '深圳
');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('小王', 18, '销售部', '深圳
');
```

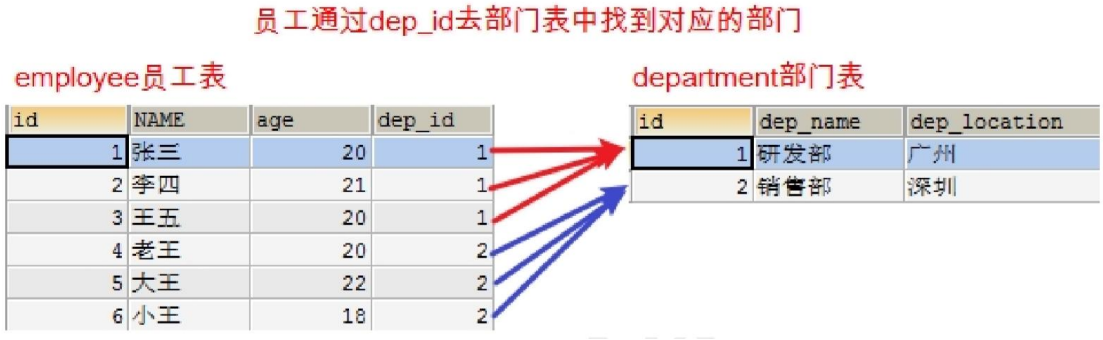
id	name	age	dep_name	dep_location
1	张三	20	研发部	广州
2	李四	21	研发部	广州
3	王五	20	研发部	广州
4	老王	20	销售部	深圳
5	大王	22	销售部	深圳
6	小王	18	销售部	深圳

- 以上数据表的缺点:

1) 数据冗余

2) 后期还会出现增删改的问题

- 解决方案：分成 2 张表




```
-- 创建部门表(id,dep_name,dep_location)
-- 一方，主表
 create table department(
   id int primary key auto_increment,
   dep_name varchar(20),
   dep_location varchar(20)
 );

-- 创建员工表(id,name,age,dep_id)
-- 多方，从表
create table employee(
   id int primary key auto_increment,
   name varchar(20),
   age int,
   dep_id int    -- 外键对应主表的主键
)

-- 添加 2 个部门
insert into department values(null, '研发部','广州'),(null, '销售部', '深圳');
select * from department;

-- 添加员工,dep_id 表示员工所在的部门
INSERT INTO employee (NAME, age, dep_id) VALUES ('张三', 20, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('李四', 21, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('王五', 20, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('老王', 20, 2);
```

```
INSERT INTO employee (NAME, age, dep_id) VALUES ('大王', 22, 2);
INSERT INTO employee (NAME, age, dep_id) VALUES ('小王', 18, 2);


 select * from employee;
```

- 问题：当我们在 employee 的 dep_id 里面输入不存在的部门,数据依然可以添加.但是并没有对应的部门， 实际应用中不能出现这种情况。employee 的 dep_id 中的数据只能是 department 表中存在的 id

id	NAME	age	dep_id
1	张三	20	1
2	李四	21	1
3	王五	20	1
4	老王	20	2
5	大王	22	2
6	小王	18	2
7	老张	18	6

id	dep_name	dep_location
1	研发部	广州
2	销售部	深圳

- 目标：需要约束 dep_id 只能是 department 表中已经存在 id
- 解决方式：使用外键约束

在从表中与主表主键对应的那一列，如：员工表中的 dep_id

主表： 一方，用来约束别人的表

从表： 多方，被别人约束的表

id	NAME	age	dep_id
1	张三	20	1
2	李四	21	1
3	王五	20	1
4	老王	20	2
5	大王	22	2
6	小王	18	2

id	dep_name	dep_location
1	研发部	广州
2	销售部	深圳

② 外键约束

- 创建约束的语法

新建表时增加外键：

[CONSTRAINT] [外键约束名称] FOREIGN KEY

已有表增加外键：

ALTER TABLE 从表 ADD [CONSTRAINT] [外键约束名称] FOREIGN KEY (外键字段名) REFERENCES 主表(主 键字段名);

举例：

```
-- 1) 删除副表/从表
employee drop table employee;
-- 2) 创建从表 employee 并添加外键约束 emp_depid_fk
-- 多方，从表
create table employee(
   id int primary key auto_increment,
   name varchar(20),
   age int,
   dep_id int,   -- 外键对应主表的主键
   -- 创建外键约束
   constraint emp_depid_fk foreign key (dep_id) references department(id)
 )

-- 3) 正常添加数据
INSERT INTO employee (NAME, age, dep_id) VALUES ('张三', 20, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('李四', 21, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('王五', 20, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('老王', 20, 2);
INSERT INTO employee (NAME, age, dep_id) VALUES ('大王', 22, 2);
INSERT INTO employee (NAME, age, dep_id) VALUES ('小王', 18, 2);
 select * from employee;

-- 4) 部门错误的数据添加失败
-- 插入不存在的部门
-- Cannot add or update a child row: a foreign key constraint fails
INSERT INTO employee (NAME, age, dep_id) VALUES ('老张', 18, 6);
```

③ 删除外键 （录屏 18）

ALTER TABLE 从表 drop foreign key 外键名称;

```
 -- 删除 employee 表的 emp_depid_fk 外键
alter table employee drop foreign key emp_depid_fk;
```

```
-- 在 employee 表情存在的情况下添加外键
alter table employee add constraint emp_depid_fk foreign key (dep_id) references department(id);
```

④ 外键的级联

出现新的问题：

```
select * from employee;
select * from department;
-- 要把部门表中的 id 值 2，改成 5，能不能直接更新呢?
-- Cannot delete or update a parent row: a foreign key constraint fails
update department set id=5 where id=2;

-- 要删除部门 id 等于 1 的部门，能不能直接删除呢?
-- Cannot delete or update a parent row: a foreign key constraint fails
delete from department where id=1;
```

- 什么是级联操作：

在修改和删除主表的主键时，同时更新或删除副表的外键值，称为级联操作