MySql - 基础

1.基本Sql语句

1.操作文件夹(库)

查

```
#进入mysql时输入下面指令,可以查看到所有的数据库
show databases
#use + 对应的数据库名字,进入对应的数据库
#另外还可以使用下面指令来查看创建的库
show create database db1
```

增

```
#输入如下指令,新增文件夹(库),db1为库的名字,后面可以跟编码格式,此时默认为utf-8
create database db1 ;
```

改

```
#在mysql下输入如下指令
alter database db1;
```

删

```
#输入如下指令删除库
drop database db1;
```

2.操作文件(表)

切换文件

```
#切换文件夹
use db1;
#查看当前所在的文件夹
select database();
```

增

```
#输入如下指令新增文件, 其中  t1为创建的文件名(表头字段)
# id好比一个变量,sql中是一个强类型语言,定义字段时候回强制规定规则
create table t1(id int,name char);
```

查

```
#查看当前的这张t1表
show create table t1;

#查看所有的表
show tables;

#查看表的详细信息
describe t1;
desc t1;
```

改

```
#modify  修改的意思
alter table t1 modify name ;

#改变name为答谢的NAME
alter table t1 change name NAME;
```

删

```
#删除表
drop table t1;
```

3.操作文件内容(记录)

增

```
#插入一条数据
insert t1(id name) values(1,"alex");
```

查

```
#查找时指令
select id from db1.t1;
select id,name from db1.t1;
select * from db1.t1:
```

改

```
#修改的指令
update db1.t1 set  name ="hehe";
update db1.t1 set name ="alex" where id =2
```

删

```
#删除的指令
delete from t1;
delete from t1 where id  =2;
```

2.完整性约束

1. not null 与 default

- 如果单独设置not null 不能插入空值
- 如果即设置了not null,又指定default,可以插入空值,会走default

(1)默认值可以为空

```
mysql> create table t1(id int);# id字段默认可以为空
Query OK, 0 rows affected (0.05 sec)

mysql> desc t1;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| id    | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
row in set (0.03 sec)
mysql> insert into t1 values(); #给t1表插一个空的值
Query OK, 1 row affected (0.00 sec)
#查询结果如下
mysql> select * from t1;
+------+
| id   |
+------+
| NULL |
+------+
row in set (0.00 sec)
默认值可以为空
```

(2)设置not null后 ,插入值不能为空

```
#给 num设置不能为空
mysql> create table t2(num int not null);
Query OK, 0 rows affected (0.47 sec)

mysql> desc t2;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| num   | int(11) | NO   |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
1 row in set (0.01 sec)

#此时报错,显示不能插空值
mysql> insert into t2 values();
ERROR 1364 (HY000): Field 'num' doesn't have a default value
```

(3) 如果即设置了not null,又指定default,可以插入空值,值为default,当插入值不是空值,结果是自己的值

```
#创建 t3表此时设置not null,并且default为 4
```

```
mysql> create table t3(num int not null default 4);
Query OK, 0 rows affected (0.46 sec)

mysql> desc t3;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| num   | int(11) | NO   |     | 4       |       |
+-------+---------+------+-----+---------+-------+
1 row in set (0.01 sec)
#当插入数据为空时,此时是默认值生效
mysql> insert into t3 values();
Query OK, 1 row affected (0.11 sec)
#但是也可以插入值但是此时却是给的值而不是默认值
mysql> insert into t3 values(2);
Query OK, 1 row affected (0.11 sec)

mysql> select * from db2.t3;
+-----+
| num |
+-----+
|   4 |
|   2 |
+-----+
2 rows in set (0.00 sec)
```

2. unique key

(1)单列唯一

```
# 创建t4表,给name设置单列唯一
mysql> create table t4(id int not null,name char(20) unique);
Query OK, 0 rows affected (0.47 sec)

mysql> desc t4;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| id    | int(11)  | NO   |     | NULL    |       |
| name  | char(20) | YES  | UNI | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> insert into t4 values(1,'alex');
Query OK, 1 row affected (0.35 sec)

mysql> insert into t4 values(1,'alex');
#有于给name设置了单列为一,因此插入相同的值会报错
ERROR 1062 (23000): Duplicate entry 'alex' for key 'name'
#当将值改为与第一次不同时候,发现此时插入成功
mysql> insert into t4 values(1,'alex2');
Query OK, 1 row affected (0.34 sec)
mysql> select * from db2.t4;
```

```
+----+-------+
| id | name  |
+----+-------+
|  1 | alex  |
|  1 | alex2 |
+----+-------+
2 rows in set (0.00 sec)
```

两种书写方式:

```
create table t4(
    id int not null,
    name char(20) unique
);

create table t4(
    id int not null,
    name char(20),
    unique(name)
);
insert into t4(id,name) values(1,'alex');
insert into t4(id,name) values(1,'wusir');
```

(2)多列唯一:只要有一列是相同的就不能插入.

```
#书写
create table t5(
    id int,
    name char(20),
    unique(id),
    unique(name)
);
```

```
#创建文件t5 ,此时将id 和name都设置 unique
mysql> create table t5(
    ->     id int,
    ->     name char(20),
    ->     unique(id),
    ->     unique(name)
    -> );
Query OK, 0 rows affected (0.58 sec)
#插入一组值
mysql> insert into t5 values(2,'alex');
ERROR 1062 (23000): Duplicate entry 'alex' for key 'name'
#插入有相同内容的值时候报错
mysql> insert into t5 values(1,'alex2');
ERROR 1062 (23000): Duplicate entry '1' for key 'id'
#当所有内容都不同时才能插入
mysql> insert into t5 values(2,'alex2');
Query OK, 1 row affected (0.10 sec)

mysql> desc t5;
```

```
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| id    | int(11)  | YES  | UNI | NULL    |       |
| name  | char(20) | YES  | UNI | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> select * from db2.t5;
+------+-------+
| id   | name  |
+------+-------+
|    1 | alex  |
|    2 | alex2 |
+------+-------+
2 rows in set (0.00 sec)
```

(3)联合唯一:多列相同时不能插入

**适用场景 学生选课

```
#书写
create table t6(
    id int,
    name char(20),
    unique(id,name)
);
```

```
#创建t6表 ,设置联合唯一
mysql> create table t6(
    -> id int,
    ->     name char(20),
    ->     unique(id,name)
    -> );
Query OK, 0 rows affected (0.47 sec)

mysql> desc t6;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| id    | int(11)  | YES  | MUL | NULL    |       |
| name  | char(20) | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> insert into t6 values(1,"alex");
Query OK, 1 row affected (0.35 sec)
#当插入相同的两列记录会报错
mysql> insert into t6 values(1,"alex");
ERROR 1062 (23000): Duplicate entry '1-alex' for key 'id'
mysql> insert into t6 values(2,"alex");
Query OK, 1 row affected (0.11 sec)
```

```
#但是当两列记录中的一项内容不同就可以插入
mysql> insert into t6 values(1,"wusir");
Query OK, 1 row affected (0.10 sec)

mysql> select *from t6;
+------+-------+
| id   | name  |
+------+-------+
|    1 | alex  |
|    1 | wusir |
|    2 | alex  |
+------+-------+
3 rows in set (0.01 sec)
```

3. 主键primary key (not null + unique 的结果)

(1)单列主键 不能为空 ,并且是唯一

```
#书写
# primary key 索引(针对于大量数据) 查询速度要快
create table t7(
    id int primary key,
    name varchar(10) unique
);
#相同结果的写法
create table t8(
    id int not null unique,
    name varchar(10) unique
);
```

```
#创建t7表 此时为id设置主键,并且 name唯一
mysql> create table t7(
    ->     id int primary key,
    ->     name varchar(10) unique
    -> );
Query OK, 0 rows affected (0.51 sec)

mysql> desc t7;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | NULL    |       |
| name  | varchar(10) | YES  | UNI | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
#当插入空值时候回报错
mysql> insert into t7 values();
ERROR 1364 (HY000): Field 'id' doesn't have a default value

mysql> insert into t7(id,name) values(1,"wusir");
Query OK, 1 row affected (0.37 sec)
#id唯一且不能为空
```

```
mysql> insert into t7(id,name) values(1,"wusir");
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
mysql> insert into t7(id,name) values(1,"wusir2");
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
mysql> insert into t7(id,name) values(2,"wusir2");
Query OK, 1 row affected (0.35 sec)

mysql> select *from db2.t7;
+----+--------+
| id | name   |
+----+--------+
|  1 | wusir  |
|  2 | wusir2 |
+----+--------+
2 rows in set (0.00 sec)
```

(2)联合主键

```
#书写方式
create table t9(
    id int,
    name varchar(10),
    primary key(id,name)
);
```

```
#创建了t9表,此时可以设置空值
mysql> create table t9(
    ->     id int,
    ->     name varchar(10),
    ->     primary key(id,name)
    -> );
Query OK, 0 rows affected (0.30 sec)

mysql> desc t9;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | 0       |       |
| name  | varchar(10) | NO   | PRI |         |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> insert into t9 values();
Query OK, 1 row affected (0.11 sec)

mysql> insert into t9(id name) values(1,"alex");
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds
to your MySQL server version for the right syntax to use near 'name) values(1,"alex")' at
line 1
mysql> insert into t9(id, name) values(1,"alex");
Query OK, 1 row affected (0.10 sec)
```

```
mysql>  insert into t9(id, name) values(1,"alex2");
Query OK, 1 row affected (0.11 sec)

mysql> select *from db2.t9;
+----+-------+
| id | name  |
+----+-------+
|  0 |       |
|  1 | alex  |
|  1 | alex2 |
+----+-------+
3 rows in set (0.00 sec)
```

**学生选课可以为空

4.auto_increment (自增长)

```
#书写方式
create table student(
    id int primary key auto_increment,
    name varchar(20) not null,
    sex enum('male','female') default 'male',
    ip varchar(20) unique
);

insert into student(name,sex,ip) values ('alex','female','127.0.0.5'),
('wusir','male','173.45.32.1');
```

```
mysql> create table student(
    ->     id int primary key auto_increment,
    ->     name varchar(20) not null,
    ->     sex enum('male','female') default 'male',
    ->     ip varchar(20) unique
    -> );
Query OK, 0 rows affected (0.23 sec)
mysql> insert into student(name,sex,ip) values ('alex','female','127.0.0.5'),
('wusir','male','173.45.32.1');
Query OK, 2 rows affected (0.37 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> desc student;
+-------+----------------------+------+-----+---------+----------------+
| Field | Type                 | Null | Key | Default | Extra          |
+-------+----------------------+------+-----+---------+----------------+
| id    | int(11)              | NO   | PRI | NULL    | auto_increment |
| name  | varchar(20)          | NO   |     | NULL    |                |
| sex   | enum('male','female') | YES  |     | male    |                |
| ip    | varchar(20)          | YES  | UNI | NULL    |                |
+-------+----------------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)

mysql> select *from student;
```

```
+----+-------+--------+-------------+
| id | name  | sex    | ip          |
+----+-------+--------+-------------+
|  1 | alex  | female | 127.0.0.5   |
|  2 | wusir | male   | 173.45.32.1 |
+----+-------+--------+-------------+
2 rows in set (0.00 sec)
```

涉及到删除时:

```
#当删除第二条记录时候,会发现此时的结果任然保留id为2 时的序号
mysql> delete from student where id =2;
Query OK, 1 row affected (0.49 sec)

mysql> select *from student;
+----+-------+--------+------------+
| id | name  | sex    | ip         |
+----+-------+--------+------------+
|  1 | alex  | female | 127.0.0.5  |
|  3 | alex1 | male   | 127.1.0.12 |
+----+-------+--------+------------+
2 rows in set (0.00 sec)
#在插入记录时候,会在后续排列
mysql> insert into student(name,sex,ip) values ('alex3','male','127.1.5.72');
Query OK, 1 row affected (0.11 sec)

mysql>  select *from student;
+----+-------+--------+------------+
| id | name  | sex    | ip         |
+----+-------+--------+------------+
|  1 | alex  | female | 127.0.0.5  |
|  3 | alex1 | male   | 127.1.0.12 |
|  4 | alex3 | male   | 127.1.5.72 |
+----+-------+--------+------------+
3 rows in set (0.00 sec)
#但是可以指定id进行插入
mysql> insert into student(id,name,sex,ip) values (2,'wusir','male','127.3.5.9');
Query OK, 1 row affected (0.11 sec)

mysql> mysql> Ctrl-C -- exit!
 select *from student;
+----+-------+--------+------------+
| id | name  | sex    | ip         |
+----+-------+--------+------------+
|  1 | alex  | female | 127.0.0.5  |
|  2 | wusir | male   | 127.3.5.9  |
|  3 | alex1 | male   | 127.1.0.12 |
|  4 | alex3 | male   | 127.1.5.72 |
+----+-------+--------+------------+
4 rows in set (0.00 sec)
#清空记录
mysql> delete from student;
Query OK, 4 rows affected (0.36 sec)
```

```
mysql>  delete from student;
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+---------------+
| Tables_in_db2 |
+---------------+
| student       |
| t1            |
| t2            |
| t3            |
| t4            |
| t5            |
| t6            |
| t7            |
| t9            |
+---------------+
9 rows in set (0.11 sec)
#但是会保留自增长的序号,后续插入
mysql> insert into student(name,sex,ip) values ('alex3','male','127.1.5.72');
Query OK, 1 row affected (0.36 sec)

mysql>  select *from student;
+----+-------+------+------------+
| id | name  | sex  | ip         |
+----+-------+------+------------+
|  5 | alex3 | male | 127.1.5.72 |
+----+-------+------+------------+
1 row in set (0.00 sec)
#但是使用truncate时候,不会保留增长的序号
mysql> truncate table student;
Query OK, 0 rows affected (0.21 sec)

mysql> show tables;
+---------------+
| Tables_in_db2 |
+---------------+
| student       |
| t1            |
| t2            |
| t3            |
| t4            |
| t5            |
| t6            |
| t7            |
| t9            |
+---------------+
9 rows in set (0.00 sec)

mysql> select *from student;
Empty set (0.00 sec)
```

```
mysql> insert into student(name,sex,ip) values ('alex3','male','127.1.5.72');
Query OK, 1 row affected (0.35 sec)
#再次插入时候从一开始
mysql> select *from student;
+----+-------+------+------------+
| id | name  | sex  | ip         |
+----+-------+------+------------+
|  1 | alex3 | male | 127.1.5.72 |
+----+-------+------+------------+
1 row in set (0.00 sec)
```

清空表区分delete和truncate的区别：

```
delete from t1      #  如果有自增id，新增的数据，仍然是以删除前的最后一样作为起始。
truncate table t1   # 数据量大，删除速度比上一条快，且直接从零开始。
```

5. foreign key(外键)

```
 # 先创建主表
mysql> create table dep(
    -> id int primary key auto_increment,
    ->     name char(10) unique,
    ->     dep_desc varchar(50) not null
    -> );
Query OK, 0 rows affected (0.47 sec)

#创建从表
mysql> create table emp(
    ->     eid int primary key auto_increment,
    ->     name char(10) not null,
    ->     age int not null,
    ->     dep_id int,
    ->     school_id int,
    ->     constraint fk_dep foreign key(dep_id) references dep(id) );
Query OK, 0 rows affected (0.55 sec)
#主表插入数据
mysql> insert into dep(name,dep_desc) values('校长部','校长管理有限部门'),('公关部','公关管理有限部门'),('IT部门','IT技 术有限部门'),('财务部','管钱很多部门');
Query OK, 4 rows affected (0.11 sec)
Records: 4  Duplicates: 0  Warnings: 0
#从表插入数据
mysql> insert into emp(name,age,dep_id)
    -> values
    -> ('alex',18,1),
    -> ('wusir',30,2),
    -> ('吴老板',20,3),
    -> ('马老板',18,4),
    -> ('邱老板',20,2),
    -> ('女神',16,3);
Query OK, 6 rows affected (0.06 sec)
Records: 6  Duplicates: 0  Warnings: 0
#将从表中对应主表的id为2 的项删除
```

```
mysql> delete from emp where dep_id =2;
Query OK, 2 rows affected (0.37 sec)

mysql> select *from emp;
+-----+-----------+-----+--------+-----------+
| eid | name      | age | dep_id | school_id |
+-----+-----------+-----+--------+-----------+
|   1 | alex      |  18 |      1 |      NULL |
|   3 | 吴老板    |  20 |      3 |      NULL |
|   4 | 马老板    |  18 |      4 |      NULL |
|   6 | 女神      |  16 |      3 |      NULL |
+-----+-----------+-----+--------+-----------+
4 rows in set (0.00 sec)
#此时可以将主表中id为2的部门删除
mysql> delete from dep where id=2;
Query OK, 1 row affected (0.35 sec)

mysql> select *from dep;
+----+-----------+--------------------------+
| id | name      | dep_desc                 |
+----+-----------+--------------------------+
|  1 | 校长部    | 校长管理有限部门         |
|  3 | IT部门    | IT技术有限部门           |
|  4 | 财务部    | 管钱很多部门             |
+----+-----------+--------------------------+
3 rows in set (0.00 sec)
#但是若果直接删除,会报错,因为此时该id对应的存在内容
mysql> delete from dep where id=3;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
(`db2`.`emp`, CONSTRAINT `fk_dep` FOREIGN KEY (`dep_id`) REFERENCES `dep` (`id`))
```

所以要同步删除和同步更新

```
#当在创建从表时添加    on delete cascade   on update cascade 同步更新同步删除
mysql> select *from dep;
+----+-----------+--------------------------+
| id | name      | dep_desc                 |
+----+-----------+--------------------------+
|  1 | 校长部    | 校长管理有限部门         |
|  2 | 公关部    | 公关管理有限部门         |
|  3 | IT部门    | IT技术有限部门           |
|  4 | 财务部    | 管钱很多部门             |
+----+-----------+--------------------------+
4 rows in set (0.00 sec)
#直接删除部门可以,而对应的部门内的人也删除
mysql> delete from dep where id=2;
Query OK, 1 row affected (0.35 sec)

mysql> select *from dep;
+----+-----------+--------------------------+
| id | name      | dep_desc                 |
+----+-----------+--------------------------+
|  1 | 校长部    | 校长管理有限部门         |
```

```
|  3 | IT部门     | IT技术有限部门           |
|  4 | 财务部     | 管钱很多部门             |
+----+-----------+-------------------------+
3 rows in set (0.00 sec)

mysql> select *from emp;
+-----+-----------+-----+--------+-----------+
| eid | name      | age | dep_id | school_id |
+-----+-----------+-----+--------+-----------+
|   1 | alex      |  18 |      1 |      NULL |
|   3 | 吴老板    |  20 |      3 |      NULL |
|   4 | 马老板    |  18 |      4 |      NULL |
|   6 | 女神      |  16 |      3 |      NULL |
+-----+-----------+-----+--------+-----------+
4 rows in set (0.00 sec)
```

3.单表查询

1.单表查询的语法

```
一、单表查询的语法
   SELECT 字段1,字段2... FROM 表名
                  WHERE 条件
                  GROUP BY field
                  HAVING 筛选
                  ORDER BY field
                  LIMIT 限制条数
二、关键字的执行优先级（重点）

重点中的重点：关键字的执行优先级
from
where
group by
having
select
distinct
order by
limit

1.找到表:from
2.拿着where指定的约束条件，去文件/表中取出一条条记录
3.将取出的一条条记录进行分组group by，如果没有group by，则整体作为一组
4.将分组的结果进行having过滤
5.执行select
6.去重
7.将结果按条件排序：order by
8.限制结果的显示条数
```

2.单表查询关键字

```
mysql> create table employee(
    ->     id int primary key auto_increment,
```

```
    ->     name  varchar(20) not null,
    ->     sex enum('male','female') not null default 'male',
    ->     age int(3) unsigned not null default 28,
    ->     hire_date date not null,
    ->     post varchar(50),
    ->     post_comment varchar(100),
    ->     salary  double(15,2),
    ->     office int,
    ->     depart_id int
    -> );
Query OK, 0 rows affected (0.46 sec)

mysql> desc employee;
+--------------+----------------------+------+-----+---------+----------------+
| Field        | Type                 | Null | Key | Default | Extra          |
+--------------+----------------------+------+-----+---------+----------------+
| id           | int(11)              | NO   | PRI | NULL    | auto_increment |
| name         | varchar(20)          | NO   |     | NULL    |                |
| sex          | enum('male','female') | NO   |     | male    |                |
| age          | int(3) unsigned      | NO   |     | 28      |                |
| hire_date    | date                 | NO   |     | NULL    |                |
| post         | varchar(50)          | YES  |     | NULL    |                |
| post_comment | varchar(100)         | YES  |     | NULL    |                |
| salary       | double(15,2)         | YES  |     | NULL    |                |
| office       | int(11)              | YES  |     | NULL    |                |
| depart_id    | int(11)              | YES  |     | NULL    |                |
+--------------+----------------------+------+-----+---------+----------------+
rows in set (0.01 sec)

mysql> insert into employee(name ,sex,age,hire_date,post,salary,office,depart_id) values
    -> ('egon','male',18,'20170301','老男孩驻沙河办事处外交大使',7300.33,401,1),
    -> ('alex','male',78,'20150302','teacher',1000000.31,401,1),
    -> ('wupeiqi','male',81,'20130305','teacher',8300,401,1),
    -> ('yuanhao','male',73,'20140701','teacher',3500,401,1),
    -> ('liwenzhou','male',28,'20121101','teacher',2100,401,1),
    -> ('jingliyang','female',18,'20110211','teacher',9000,401,1),
    -> ('jinxin','male',18,'19000301','teacher',30000,401,1),
    -> ('xiaomage','male',48,'20101111','teacher',10000,401,1),
    ->
    -> ('歪歪','female',48,'20150311','sale',3000.13,402,2),
    -> ('丫丫','female',38,'20101101','sale',2000.35,402,2),
    -> ('丁丁','female',18,'20110312','sale',1000.37,402,2),
    -> ('星星','female',18,'20160513','sale',3000.29,402,2),
    -> ('格格','female',28,'20170127','sale',4000.33,402,2),
    ->
    -> ('张野','male',28,'20160311','operation',10000.13,403,3),
    -> ('程咬金','male',18,'19970312','operation',20000,403,3),
    -> ('程咬银','female',18,'20130311','operation',19000,403,3),
    -> ('程咬铜','male',18,'20150411','operation',18000,403,3),
    -> ('程咬铁','female',18,'20140512','operation',17000,403,3)
    -> ;
Query OK, 18 rows affected (0.42 se
```

数据创建

2.1 where 约束

```
where子句中可以使用
1.比较运算符：>、<、>=、<=、<>、!=
2.between 80 and 100 ： 值在80到100之间
3.in(80,90,100)值是80或90或100
4.like 'xiaomagepattern': pattern可以是%或者_。%小时任意多字符，_表示一个字符(通配符)
5.逻辑运算符：在多个条件直接可以使用逻辑运算符 and or not
```

注:'' 是空字符串,不是null

```
mysql> select name,post_comment from employee where post_comment='';
Empty set (0.00 sec)

mysql>
mysql> update employee set post_comment='' where id=2;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select name,post_comment from employee where post_comment='';
+------+--------------+
| name | post_comment |
+------+--------------+
| alex |              |
+------+--------------+
1 row in set (0.00 sec)
```

2.2 group by 分组查询

基本概述

```
#1、首先明确一点：分组发生在where之后，即分组是基于where之后得到的记录而进行的
#2、分组指的是：将所有记录按照某个相同字段进行归类，比如针对员工信息表的职位分组，或者按照性别进行分组等
#3、为何要分组呢?
    取每个部门的最高工资
    取每个部门的员工数
    取男人数和女人数
小窍门：‘每’这个字后面的字段，就是我们分组的依据
#4、大前提：
    可以按照任意字段分组，但是分组完毕后，比如group by post，只能查看post字段，如果想查看组内信息，需要借助于聚合函数
```

关于ONLY_FULL_GROUP_BY

```
mysql> select * from employee group by post;
+----+--------+--------+-----+------------+----------------------------------------+------
--------+------------+--------+-----------+
| id | name   | sex    | age | hire_date  | post                                    |
post_comment | salary     | office | depart_id |
+----+--------+--------+-----+------------+----------------------------------------+------
--------+------------+--------+-----------+
| 14 | 张野   | male   |  28 | 2016-03-11 | operation                               | NULL
       |   10000.13 |    403 |         3 |
|  9 | 歪歪   | female |  48 | 2015-03-11 | sale                                    | NULL
       |    3000.13 |    402 |         2 |
|  2 | alex   | male   |  78 | 2015-03-02 | teacher                                 |
       | 1000000.31 |    401 |         1 |
|  1 | egon   | male   |  18 | 2017-03-01 | 老男孩驻沙河办事处外交大使              | NULL
    |    7300.33 |    401 |         1 |
+----+--------+--------+-----+------------+----------------------------------------+------
--------+------------+--------+-----------+
4 rows in set (0.00 sec)
```

```
#由于没有设置ONLY_FULL_GROUP_BY,于是也可以有结果，默认都是组内的第一条记录，但其实这是没有意义的
如果想分组，则必须要设置全局的sql的模式为ONLY_FULL_GROUP_BY
mysql> set global sql_mode='ONLY_FULL_GROUP_BY';
Query OK, 0 rows affected (0.00 sec)

#查看MySQL 5.7默认的sql_mode如下:
mysql> select @@global.sql_mode;
+--------------------+
| @@global.sql_mode  |
+--------------------+
| ONLY_FULL_GROUP_BY |
+--------------------+
row in set (0.00 sec)

mysql> exit;#设置成功后，一定要退出，然后重新登录方可生效

 # 继续验证通过 groupby分组之后,只能查看当前字段,如果要查看组内信息,需要借助聚合函数.
mysql> select * from employee group by post;
ERROR 1055 (42000): 'db15.employee.id' isn't in GROUP BY

mysql>  select post from employee group by post;
+----------------------------------------+
| post                                   |
+----------------------------------------+
| operation                              |
| sale                                   |
| teacher                                |
| 老男孩驻沙河办事处外交大使             |
+----------------------------------------+
rows in set (0.00 sec)
```

2.3 聚合函数

```
max() 最大值
min() 最小值
sum() 求和
count() 求总个数
avg() 求平均值
#强调：聚合函数聚合的是组的内容，若是没有分组，则默认一组  注意:是组内信息
```

```
mysql> select post,count(id) from employee group by post;
+----------------------------------------+-----------+
| post                                   | count(id) |
+----------------------------------------+-----------+
| operation                              |         5 |
| sale                                   |         5 |
| teacher                                |         7 |
| 老男孩驻沙河办事处外交大使             |         1 |
+----------------------------------------+-----------+
4 rows in set (0.37 sec)

mysql> select post,max(salary) from employee group by post;
+----------------------------------------+-------------+
| post                                   | max(salary) |
+----------------------------------------+-------------+
| operation                              |    20000.00 |
| sale                                   |     4000.33 |
| teacher                                |  1000000.31 |
| 老男孩驻沙河办事处外交大使             |     7300.33 |
+----------------------------------------+-------------+
4 rows in set (0.35 sec)

mysql> select post,min(salary) from employee group by post;
+----------------------------------------+-------------+
| post                                   | min(salary) |
+----------------------------------------+-------------+
| operation                              |    10000.13 |
| sale                                   |     1000.37 |
| teacher                                |     2100.00 |
| 老男孩驻沙河办事处外交大使             |     7300.33 |
+----------------------------------------+-------------+
4 rows in set (0.00 sec)

mysql> select post,avg(salary) from employee group by post;
+----------------------------------------+---------------+
| post                                   | avg(salary)   |
+----------------------------------------+---------------+
| operation                              |  16800.026000 |
| sale                                   |   2600.294000 |
| teacher                                | 151842.901429 |
| 老男孩驻沙河办事处外交大使             |   7300.330000 |
+----------------------------------------+---------------+
4 rows in set (0.34 sec)

mysql> select post,sum(salary) from employee group by post;
```

```
+--------------------------------------+-------------+
| post                                 | sum(salary) |
+--------------------------------------+-------------+
| operation                            |    84000.13 |
| sale                                 |    13001.47 |
| teacher                              |  1062900.31 |
| 老男孩驻沙河办事处外交大使              |     7300.33 |
+--------------------------------------+-------------+
4 rows in set (0.00 sec)
```

概念:虚拟表 (不是物理存在的,只是被构造出来的)

```
mysql> select id,name,1 from employee;
+----+------------+---+
| id | name       | 1 |
+----+------------+---+
|  1 | egon       | 1 |
|  2 | alex       | 1 |
|  3 | wupeiqi    | 1 |
|  4 | yuanhao    | 1 |
|  5 | liwenzhou  | 1 |
|  6 | jingliyang | 1 |
|  7 | jinxin     | 1 |
|  8 | xiaomage   | 1 |
|  9 | 歪歪       | 1 |
| 10 | 丫丫       | 1 |
| 11 | 丁丁       | 1 |
| 12 | 星星       | 1 |
| 13 | 格格       | 1 |
| 14 | 张野       | 1 |
| 15 | 程咬金     | 1 |
| 16 | 程咬银     | 1 |
| 17 | 程咬铜     | 1 |
| 18 | 程咬铁     | 1 |
+----+------------+---+
18 rows in set (0.00 sec)

mysql> select post,count(1) from employee group by post;
+--------------------------------------+----------+
| post                                 | count(1) |
+--------------------------------------+----------+
| operation                            |        5 |
| sale                                 |        5 |
| teacher                              |        7 |
| 老男孩驻沙河办事处外交大使              |        1 |
+--------------------------------------+----------+
4 rows in set (0.00 sec)
```

给虚拟表起别名

```
mysql> select A.a from (select post,count(1) as a from employee group by post) as A;
+---+
| a |
+---+
| 5 |
| 5 |
| 7 |
| 1 |
+---+
4 rows in set (0.10 sec)
```

练习2:

```
1. 查询岗位名以及岗位包含的所有员工名字
2. 查询男员工与男员工的平均薪资，女员工与女员工的平均薪资
```

```
mysql> select post,group_concat(name) from employee group by post;
+----------------------------------------+-----------------------------------------------
-----------+
| post                                   | group_concat(name)
          |
+----------------------------------------+-----------------------------------------------
-----------+
| operation                              | 程咬铁,程咬铜,程咬银,程咬金,张野
      |
| sale                                   | 格格,星星,丁丁,丫丫,歪歪
      |
| teacher                                |
xiaomage,jinxin,jingliyang,liwenzhou,yuanhao,wupeiqi,alex |
| 老男孩驻沙河办事处外交大使             | egon
      |
+----------------------------------------+-----------------------------------------------
-----------+
4 rows in set (0.38 sec)

mysql> select sex,avg(salary) from  employee group by sex;
+--------+---------------+
| sex    | avg(salary)   |
+--------+---------------+
| male   | 110920.077000 |
| female |   7250.183750 |
+--------+---------------+
2 rows in set (0.00 sec)
```

2.4 HAVING 过滤

where 和 having 的区别

- 执行优先级从高到低：where > group by > having
- Where 发生在分组group by之前，因而Where中可以有任意字段，但是绝对不能使用聚合函数

```
mysql> select * from employee where avg(age)>20;
ERROR 1111 (HY000): Invalid use of group function       --->组功能的使用无效
```

- Having发生在分组group by之后，因而Having中可以使用分组的字段，无法直接取到其他字段,可以使用聚合函数

```
mysql>  select * from employee having salary>1000000;
ERROR 1463 (42000): Non-grouping field 'salary' is used in HAVING clause
# 必须分组之后才能使用having
```

这针对的是5.6版本 ,对于5.7 版本 (整体作为一组 可以执行sql)

分组 和 having练习

- 查询各岗位内包含的员工个数小于2的岗位名、岗位内包含员工名字、个数

```
mysql> select post,group_concat(name),count(1) from employee group by post having
count(1)<2;
+----------------------------------------+--------------------+----------+
| post                                   | group_concat(name) | count(1) |
+----------------------------------------+--------------------+----------+
| 老男孩驻沙河办事处外交大使             | egon               |        1 |
+----------------------------------------+--------------------+----------+
1 row in set (0.00 sec)
```

- 查询各岗位平均薪资大于10000且小于20000的岗位名、平均工资

```
mysql> select post,avg(salary) from employee group by post having avg(salary)>10000
and avg(salary)<20000;
+-----------+--------------+
| post      | avg(salary)  |
+-----------+--------------+
| operation | 16800.026000 |
+-----------+--------------+
1 row in set (0.00 sec)
```

- 按职位分组后,组内平均年龄大于25岁的组的名称,人数,组内平均年龄

```
mysql> select post,count(1),avg(age) from employee group by post having avg(age)>25;
+---------+----------+----------+
| post    | count(1) | avg(age) |
+---------+----------+----------+
| sale    |        5 |  30.0000 |
| teacher |        7 |  49.1429 |
+---------+----------+----------+
2 rows in set (0.00 sec)
```

2.5 order by

按照单列排序

```
mysql> select * from employee order by age;
mysql> select * from employee order by age asc;
mysql> select * from employee order by age desc;
```

按照多列排序(例: 先按照age升序排序,如果年纪相同,则按照id降序)

```
mysql> select * from employee order by age asc,id desc;
```

小练习

查询所有员工信息，先按照age升序排序，如果age相同则按照hire_date降序排序

```
mysql> select * from employee order by age asc,hire_date desc;
```

查询各岗位平均薪资大于10000的岗位名、平均工资,结果按平均薪资升序排列

```
mysql> select post,avg(salary) from employee group by post having avg(salary)>10000 order
by avg(salary) asc;
+-----------+---------------+
| post      | avg(salary)   |
+-----------+---------------+
| operation |  16800.026000 |
| teacher   | 151842.901429 |
+-----------+---------------+
2 rows in set (0.00 sec)
```

查询各岗位平均薪资大于10000的岗位名、平均工资,结果按平均薪资降序排列

```
mysql> select post,avg(salary) from employee group by post having avg(salary)>10000 order
by avg(salary) desc;
+-----------+---------------+
| post      | avg(salary)   |
+-----------+---------------+
| teacher   | 151842.901429 |
| operation |  16800.026000 |
+-----------+---------------+
2 rows in set (0.00 sec)
```

2.6 limit 限制

默认初始位置为0

```
mysql> select * from employee order by salary desc limit 3;
+----+-----------+------+-----+------------+-----------+--------------+------------+-------
--+-----------+
| id | name      | sex  | age | hire_date  | post      | post_comment | salary     | office
| depart_id |
+----+-----------+------+-----+------------+-----------+--------------+------------+-------
--+-----------+
|  2 | alex      | male |  78 | 2015-03-02 | teacher   |              | 1000000.31 |    401
|         1 |
|  7 | jinxin    | male |  18 | 1900-03-01 | teacher   | NULL         |   30000.00 |    401
|         1 |
| 15 | 程咬金    | male |  18 | 1997-03-12 | operation | NULL         |   20000.00 |    403
|         3 |
+----+-----------+------+-----+------------+-----------+--------------+------------+-------
--+-----------+
3 rows in set (0.00 sec)
```

从第0开始,即先查出第一条,然后包含这一条在内往后查5条

```
mysql> select * from employee order by salary desc limit 0,5;
+----+-----------+--------+-----+------------+-----------+--------------+------------+-----
---+-----------+
| id | name      | sex    | age | hire_date  | post      | post_comment | salary     |
office | depart_id |
+----+-----------+--------+-----+------------+-----------+--------------+------------+-----
---+-----------+
|  2 | alex      | male   |  78 | 2015-03-02 | teacher   |              | 1000000.31 |
401 |         1 |
|  7 | jinxin    | male   |  18 | 1900-03-01 | teacher   | NULL         |   30000.00 |
401 |         1 |
| 15 | 程咬金    | male   |  18 | 1997-03-12 | operation | NULL         |   20000.00 |
403 |         3 |
| 16 | 程咬银    | female |  18 | 2013-03-11 | operation | NULL         |   19000.00 |
403 |         3 |
| 17 | 程咬铜    | male   |  18 | 2015-04-11 | operation | NULL         |   18000.00 |
403 |         3 |
+----+-----------+--------+-----+------------+-----------+--------------+------------+-----
---+-----------+
5 rows in set (0.00 sec)
```

从第5开始,即先查出第一条,然后包含这一条在内往后查5条

```
mysql> select * from employee order by salary desc limit 5,5;
+----+-----------+--------+-----+------------+-----------+--------------+----------+------
--+-----------+
| id | name      | sex    | age | hire_date  | post      | post_comment | salary   |
office | depart_id |
+----+-----------+--------+-----+------------+-----------+--------------+----------+------
--+-----------+
| 18 | 程咬铁    | female |  18 | 2014-05-12 | operation | NULL         | 17000.00 |    403
|         3 |
| 14 | 张野      | male   |  28 | 2016-03-11 | operation | NULL         | 10000.13 |
 403 |         3 |
|  8 | xiaomage  | male   |  48 | 2010-11-11 | teacher   | NULL         | 10000.00 |
 401 |         1 |
|  6 | jingliyang | female |  18 | 2011-02-11 | teacher   | NULL         |  9000.00 |
 401 |         1 |
|  3 | wupeiqi   | male   |  81 | 2013-03-05 | teacher   | NULL         |  8300.00 |
 401 |         1 |
+----+-----------+--------+-----+------------+-----------+--------------+----------+------
--+-----------+
5 rows in set (0.00 sec
```

4.多表查询

数据准备

```
create table department(
id int,
name varchar(20)
);

create table employee(
id int primary key auto_increment,
name varchar(20),
sex enum('male','female') not null default 'male',
age int,
dep_id int
);

#插入数据
insert into department values
(200,'技术'),
(201,'人力资源'),
(202,'销售'),
(203,'运营');

insert into employee(name,sex,age,dep_id) values
('egon','male',18,200),
('alex','female',48,201),
('wupeiqi','male',38,201),
('yuanhao','female',28,202),
('nvshen','male',18,200),
('xiaomage','female',18,204)
```

```
;

# 查看表结构和数据
mysql> desc department;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
rows in set (0.19 sec)

mysql> desc employee;
+--------+----------------------+------+-----+---------+----------------+
| Field  | Type                 | Null | Key | Default | Extra          |
+--------+----------------------+------+-----+---------+----------------+
| id     | int(11)              | NO   | PRI | NULL    | auto_increment |
| name   | varchar(20)          | YES  |     | NULL    |                |
| sex    | enum('male','female') | NO   |     | male    |                |
| age    | int(11)              | YES  |     | NULL    |                |
| dep_id | int(11)              | YES  |     | NULL    |                |
+--------+----------------------+------+-----+---------+----------------+
rows in set (0.01 sec)

mysql> select * from department;
+------+--------------+
| id   | name         |
+------+--------------+
|  200 | 技术         |
|  201 | 人力资源     |
|  202 | 销售         |
|  203 | 运营         |
+------+--------------+
rows in set (0.02 sec)

mysql> select * from employee;
+----+----------+--------+------+--------+
| id | name     | sex    | age  | dep_id |
+----+----------+--------+------+--------+
|  1 | egon     | male   |   18 |    200 |
|  2 | alex     | female |   48 |    201 |
|  3 | wupeiqi  | male   |   38 |    201 |
|  4 | yuanhao  | female |   28 |    202 |
|  5 | nvshen   | male   |   18 |    200 |
|  6 | xiaomage | female |   18 |    204 |
+----+----------+--------+------+--------+
rows in set (0.00 sec)
```

1. 多表连接查询

语法:

```
select 字段列表 from 表1 inner|left|right join 表2  on 表1.字段 = 表2.字段;
```

1.1 一个概念：笛卡尔积

```
mysql> select * from employee2,department;
+----+----------+--------+------+--------+------+--------------+
| id | name     | sex    | age  | dep_id | id   | name         |
+----+----------+--------+------+--------+------+--------------+
|  1 | egon     | male   |   18 |    200 |  200 | 技术         |
|  1 | egon     | male   |   18 |    200 |  201 | 人力资源     |
|  1 | egon     | male   |   18 |    200 |  202 | 销售         |
|  1 | egon     | male   |   18 |    200 |  203 | 运营         |
|  2 | alex     | female |   48 |    201 |  200 | 技术         |
|  2 | alex     | female |   48 |    201 |  201 | 人力资源     |
|  2 | alex     | female |   48 |    201 |  202 | 销售         |
|  2 | alex     | female |   48 |    201 |  203 | 运营         |
|  3 | wupeiqi  | male   |   38 |    201 |  200 | 技术         |
|  3 | wupeiqi  | male   |   38 |    201 |  201 | 人力资源     |
|  3 | wupeiqi  | male   |   38 |    201 |  202 | 销售         |
|  3 | wupeiqi  | male   |   38 |    201 |  203 | 运营         |
|  4 | yuanhao  | female |   28 |    202 |  200 | 技术         |
|  4 | yuanhao  | female |   28 |    202 |  201 | 人力资源     |
|  4 | yuanhao  | female |   28 |    202 |  202 | 销售         |
|  4 | yuanhao  | female |   28 |    202 |  203 | 运营         |
|  5 | nvshen   | male   |   18 |    200 |  200 | 技术         |
|  5 | nvshen   | male   |   18 |    200 |  201 | 人力资源     |
|  5 | nvshen   | male   |   18 |    200 |  202 | 销售         |
|  5 | nvshen   | male   |   18 |    200 |  203 | 运营         |
|  6 | xiaomage | female |   18 |    204 |  200 | 技术         |
|  6 | xiaomage | female |   18 |    204 |  201 | 人力资源     |
|  6 | xiaomage | female |   18 |    204 |  202 | 销售         |
|  6 | xiaomage | female |   18 |    204 |  203 | 运营         |
+----+----------+--------+------+--------+------+--------------+
24 rows in set (0.11 sec)
```

符合条件查询

```
mysql> select * from employee,department where employee.dep_id = department.id;
+----+---------+--------+------+--------+------+--------------+
| id | name    | sex    | age  | dep_id | id   | name         |
+----+---------+--------+------+--------+------+--------------+
|  1 | egon    | male   |   18 |    200 |  200 | 技术         |
|  2 | alex    | female |   48 |    201 |  201 | 人力资源     |
|  3 | wupeiqi | male   |   38 |    201 |  201 | 人力资源     |
|  4 | yuanhao | female |   28 |    202 |  202 | 销售         |
|  5 | nvshen  | male   |   18 |    200 |  200 | 技术         |
+----+---------+--------+------+--------+------+--------------+
5 rows in set (0.01 sec)
```

1.2 内连接

```
mysql> select * from employee inner join department on employee.dep_id = department.id;
+----+---------+--------+------+--------+------+--------------+
| id | name    | sex    | age  | dep_id | id   | name         |
+----+---------+--------+------+--------+------+--------------+
|  1 | egon    | male   |   18 |    200 |  200 | 技术         |
|  2 | alex    | female |   48 |    201 |  201 | 人力资源     |
|  3 | wupeiqi | male   |   38 |    201 |  201 | 人力资源     |
|  4 | yuanhao | female |   28 |    202 |  202 | 销售         |
|  5 | nvshen  | male   |   18 |    200 |  200 | 技术         |
+----+---------+--------+------+--------+------+--------------+
5 rows in set (0.00 sec)
```

通过上表可以看出,内连接是找到两张表共有的部分,相当于利用条件从笛卡尔积结果中筛选出了匹配的结果

(department没有204这个部门，因而employee表中关于204这条员工信息没有匹配出来) -> 与上面的符合条件查询相同

1.3 左连接或右连接 (优先显示左表或者右表的全部记录)

左连接

```
mysql> select * from employee left join department on employee.dep_id = department.id;
+----+----------+--------+------+--------+------+--------------+
| id | name     | sex    | age  | dep_id | id   | name         |
+----+----------+--------+------+--------+------+--------------+
|  1 | egon     | male   |   18 |    200 |  200 | 技术         |
|  5 | nvshen   | male   |   18 |    200 |  200 | 技术         |
|  2 | alex     | female |   48 |    201 |  201 | 人力资源     |
|  3 | wupeiqi  | male   |   38 |    201 |  201 | 人力资源     |
|  4 | yuanhao  | female |   28 |    202 |  202 | 销售         |
|  6 | xiaomage | female |   18 |    204 | NULL | NULL         |
+----+----------+--------+------+--------+------+--------------+
6 rows in set (0.00 sec)
```

右连接

```
mysql> select * from employee right join department on employee.dep_id = department.id;
+------+---------+--------+------+--------+------+--------------+
| id   | name    | sex    | age  | dep_id | id   | name         |
+------+---------+--------+------+--------+------+--------------+
|    1 | egon    | male   |   18 |    200 |  200 | 技术         |
|    2 | alex    | female |   48 |    201 |  201 | 人力资源     |
|    3 | wupeiqi | male   |   38 |    201 |  201 | 人力资源     |
|    4 | yuanhao | female |   28 |    202 |  202 | 销售         |
|    5 | nvshen  | male   |   18 |    200 |  200 | 技术         |
| NULL | NULL    | NULL   | NULL |   NULL |  203 | 运营         |
+------+---------+--------+------+--------+------+--------------+
6 rows in set (0.00 sec)
```

1.4 全外连接 (显示两个表中的全部记录)

```
mysql> select * from employee left join department on employee.dep_id = department.id
    ->   union
    -> select * from employee right join department on employee.dep_id = department.id;
+------+----------+--------+------+--------+------+--------------+
| id   | name     | sex    | age  | dep_id | id   | name         |
+------+----------+--------+------+--------+------+--------------+
|    1 | egon     | male   |   18 |    200 |  200 | 技术         |
|    5 | nvshen   | male   |   18 |    200 |  200 | 技术         |
|    2 | alex     | female |   48 |    201 |  201 | 人力资源     |
|    3 | wupeiqi  | male   |   38 |    201 |  201 | 人力资源     |
|    4 | yuanhao  | female |   28 |    202 |  202 | 销售         |
|    6 | xiaomage | female |   18 |    204 | NULL | NULL         |
| NULL | NULL     | NULL   | NULL |   NULL |  203 | 运营         |
+------+----------+--------+------+--------+------+--------------+
7 rows in set (0.01 sec)
```

2. 符合条件连接查询

- 找出年龄大于25岁的员工以及员工所在的部门

```
mysql> select employee.name,department.name from employee inner join department on
employee.dep_id = department.id where age>25;
+---------+--------------+
| name    | name         |
+---------+--------------+
| alex    | 人力资源     |
| wupeiqi | 人力资源     |
| yuanhao | 销售         |
+---------+--------------+
3 rows in set (0.00 sec)
```

- 找出年龄大于25岁的员工以及员工所在的部门,

  并且以age字段的升序方式显示。

```
mysql> select employee.name,department.name from employee inner join department on
employee.dep_id = department.id where age>25 order by age asc;
+---------+--------------+
| name    | name         |
+---------+--------------+
| yuanhao | 销售         |
| wupeiqi | 人力资源     |
| alex    | 人力资源     |
+---------+--------------+
3 rows in set (0.00 sec)
```

3. 子查询

3.1 带in 关键字的子查询(练习题)

- 查询平均年龄在25岁以上的部门的名称

```
mysql> select dep_id from employee group by dep_id having avg(age)>25;
+--------+
| dep_id |
+--------+
|    201 |
|    202 |
+--------+
2 rows in set (0.38 sec)

mysql> select * from department where id in (select dep_id from employee group by dep_id
having avg(age)>25);
+------+--------------+
| id   | name         |
+------+--------------+
|  201 | 人力资源     |
|  202 | 销售         |
+------+--------------+
2 rows in set (0.00 sec)
```

- 查看技术部员工的姓名

```
mysql> select name from employee where dep_id in(select id from department where name
="技术");
+--------+
| name   |
+--------+
| egon   |
| nvshen |
+--------+
2 rows in set (0.00 sec)
```

- 查看不足一人的部门名

```
mysql> select name from department where id not  in(select dep_id from employee group by
dep_id );
+--------+
| name   |
+--------+
| 运营   |
+--------+
1 row in set (0.00 sec
```

3.2 带比较运算符的子查询

```
#比较运算符：=、!=、>、>=、<、<=、<>
#查询大于所有人平均年龄的员工名与年龄
mysql> select name,age from employee where age > (select avg(age) from employee);
+---------+------+
| name    | age  |
+---------+------+
| alex    |   48 |
```

```
| wupeiqi |   38 |
+---------+------+

#查询大于部门内平均年龄的员工名、年龄
思路:
       (1) 先对员工表(employee)中的人员分组(group by)，查询出dep_id以及平均年龄。
       (2)将查出的结果作为临时表,再对根据临时表的dep_id和employee的dep_id作为筛选条件将employee表和临时表进行内连接。
       (3)最后再将employee员工的年龄是大于平均年龄的员工名字和年龄筛选。



mysql> select t1.name,t1.age from employee as t1
             inner join
            (select dep_id,avg(age) as avg_age from employee group by dep_id) as t2
            on t1.dep_id = t2.dep_id
            where t1.age > t2.avg_age;
+------+------+
| name | age  |
+------+------+
| alex |   48 |
```

3.3 带EXISTS关键字的子查询

```
#EXISTS关字键字表示存在。在使用EXISTS关键字时，内层查询语句不返回查询的记录。而是返回一个真假值。True或False
#当返回True时，外层查询语句将进行查询；当返回值为False时，外层查询语句不进行查询
#department表中存在dept_id=203，Ture
mysql> select * from employee  where exists (select id from department where id=200);
+----+----------+--------+------+--------+
| id | name     | sex    | age  | dep_id |
+----+----------+--------+------+--------+
|  1 | egon     | male   |   18 |    200 |
|  2 | alex     | female |   48 |    201 |
|  3 | wupeiqi  | male   |   38 |    201 |
|  4 | yuanhao  | female |   28 |    202 |
|  5 | nvshen   | male   |   18 |    200 |
|  6 | xiaomage | female |   18 |    204 |
+----+----------+--------+------+--------+
#department表中存在dept_id=205，False
mysql> select * from employee  where exists (select id from department where id=204);
Empty set (0.00 sec)
```