数据库语法总结

数据库操作：

查询所有数据库

```
show databases;
```

查询当前数据库

```
select database();
```

创建数据库

```
create database [if not exists] 数据库名 [default charset 字符集] [collate 排列规则];
```

删除数据库

```
drop database[if exists] 数据库名;
```

使用数据库

```
use 数据库名;
```

注：有[ ]的可以选择性填写，填写时不需要加[]

数据表操作：

查询当前数据库内所有表

```
show tables;
```

查询表结构

```
desc 表名;
```

查询指定表的建表语句

```
show create table 表名;
```

创建数据表

```
create table 表名(
```

```
    字段 1 字段 1 类型[comment 字段 1 注释],
    字段 2 字段 2 类型[comment 字段 2 注释],
    字段 3 字段 3 类型[comment 字段 3 注释],
    ......
    字段 n 字段 n 类型[comment 字段 n 注释]
)[comment 表注释];
```

注：1、最后一个字段的创建时，没有逗号

2、使用 desc 查看表时，看不到注释，只能看到基本形式，需要使用 `show create table 表名;` 来查看注释

数值类型：

类型	大小或用途
整数	
tinyint	1
smallint	2
mediumint	3
int 或 integer	4
bigint	8

注：使用时，有负号的直接默认，无负号的需后缀 unsigned

有符号(SIGNED)范围	无符号(UNSIGNED)范围
(-128，127)	(0，255)
(-32768，32767)	(0，65535)
(-8388608，8388607)	(0，16777215)
(-2147483648，2147483647)	(0，4294967295)
(-2^63，2^63-1)	(0，2^64-1)

浮点数	
float	4
double	8

decimal	依赖于精度 M（数值总长度）和标度 D（小数点后的位数）

字符串：

char	固定长度字符串
varchar	可变长度字符串
tinyblob	不超过 255 个字符串的二进制数据
tinytext	短文本字符串
blob	二进制形式的长文本数据
text	长文本数据
mediumblob	二进制形式的中等长度文本数据
mediumtext	中等长度文本数据
longblob	二进制形式的极大文本数据
longtext	极大文本数据

注：1、这里大致分为 blob 和 text 两种：blob 用于二进制，text 用于文本
2、char 和 varchar 在使用时都要后缀括号
如：char（10）：必须是十个字符串
varchar（10）：长度在 10 个字符串以内

日期类型：

date	日期值
time	时间值或持续时间
year	年份值
datetime	混合日期和时间值
timestamp	混合日期和时间值，时间戳

注：

范围	格式
1000-01-01 至 9999-12-31	YYYY-MM-DD
-838:59:59 至 838:59:59	HH:MM:SS
1901 至 2155	YYYY
1000-01-01 00:00:00 至 9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS
1970-01-01 00:00:01 至 2038-01-19 03:14:07	YYYY-MM-DD HH:MM:SS

表修改：

添加字段

alter table 表名 add 字段名 类型(长度) [comment 注释] [约束];

修改数据类型

alter table 表名 modify 字段名 新数据类型(长度);

修改字段名和字段类型

alter table 表名 change 旧字段 新字段 类型(长度) [comment 注释] [约束];

删除字段

alter table 表名 drop 字段

修改表名

alter table 表名 rename to 新表名;

删除表

drop table[if exists] 表名;

删除指定表，并重新创建该表

truncate table 表名;

注：删除指定表，并重新创建该表的用法中，原表所保留的数据并不会保留

表中数据修改：

添加：

给指定字段添加数据

```
insert into 表名 (字段名 1,字段名 2,....) values (值 1,值 2,...);
```

给全部字段添加数据

```
insert into 表名 values(值 1,值 2,...);
```

批量添加数据

```
insert into (字段名 1,字段名 2,....) values (值 1,值 2,...),（值 1,值 2,...),（值 1,值 2,...);
insert into 表名 values(值 1,值 2,...),(值 1,值 2,...),(值 1,值 2,...);
```

注：1、插入数据时，字段与值的顺序是一一对应
2、字符串和日期类型放在引号中
3、注意数据大小不能超出数据类型范围

修改数据

```
update 表名 set 字段名 1=值 1,字段名 2=值 2,……[where 条件];
```

删除数据

```
delete from 表名 [where 条件];
```

注：1、删除语句的条件可以有，也可以没有，如果没有条件，则会删除整张表的所有数据
2、删除语句不能删除某一个字段的值（可以使用 update）

查询基础：

1、基础语法

```
select        字段列表
```

from 表名列表
where 条件列表
group by 分组字段列表
having 分组后条件列表
order by 排序字段列表——asc升序、desc降序（不写默认是升序）
limit 分页参数

2、查询多个字段

select 字段 1,字段 2,字段 3...from 表名；
select*from 表名;

注：* 代表查询所有字段

3、设置别名

select 字段 1 [as 别名 1],字段 2[as 别名 2] ... from 表名;

4、去除重复记录

select distinct 字段列表 from 表名;

注：去除重复记录的操作仅用于查看，并不是真正意义上在表中删除

条件查询：

1、语法

select 字段列表 from 表名 where 条件列表;

2、条件

比较运算符	功能
>	大于
>=	大于等于
<	小于
<=	小于等于
=	等于
<> 或 !=	不等于
BETWEEN ... AND ...	在某个范围之内(含最小、最大值)
IN(...)	在in之后的列表中的值，多选一
LIKE 占位符	模糊匹配(_匹配单个字符, %匹配任意个字符)
IS NULL	是NULL

逻辑运算符	功能
AND 或 &&	并且 (多个条件同时成立)
OR 或 ‖	或者 (多个条件任意一个成立)
NOT 或 ！	非，不是

注：like语句中，_表示有一个字符，___表示有三个字符；而%表示有任意个字符
例图：

```
select *from emp where name like '__';
select *from emp where idcard like '%X';
```

筛选null字段的语法：

```
select * from emp where idcard is null;
select * from emp where idcard is not null;
```

多选项的语法：

```
select * from emp where idcard is null;
select * from emp where idcard is not null;
```

聚合函数：

1、介绍

将**一列数据**作为一个整体，进行纵向计算
即，只作用于某一列

2、常见聚合函数

count	统计数量
max	最大值
min	最小值
avg	平均值
sum	求和

3、语法

select 聚合函数(字段列表) from 表名;

注：聚合函数的语法中，null值不参与任何计算
例图：

```sql
select COUNT(*) from emp;
select COUNT(age) from emp;
-- 只要是非空字段都能统计数据

select max(age) from emp;
select min(age) from emp;
select sum(age) from emp where workaddress='江苏';
```

分组查询：

1、语法

select 字段列表 from 表名 [where 条件] group by 分组字段名 [having 分组后过滤条件]

2、where于having的区别

）where分组之前就先过滤，不满足条件的不参与分组；而having是分组之后对结果进行筛选

）where可以对聚合函数进行判断，而having可以

注：执行顺序一般为where>聚合函数>having

分组之后，查询的字段一般为聚合函数和分组字段，查询其他字段无意义

例图：

```sql
-- 根据性别分组，分别统计男女员工数量、平均年龄
select gender,count(*) from emp group by gender;
select gender,avg(age) from emp group by gender;
-- 查询年龄小于45的员工，并根据工作地址分组，获取员工数量大于3的工作地址
select workaddress,COUNT(*) from emp where age<45 group by workaddress having count(workaddress)>3;
```

排序查询：

1、语法

select 字段列表 from 表名 order by 字段1 排序方式1，字段2 排序方式2;

2、排序方式

）asc：升序（默认值）

）desc：降序

注：如果是多字段排序，当第一个字段值相同时，才会根据第二字段进行排序

例图：

```
-- 根据年龄对公司员工进行排序
select name,age from emp order by age asc;
select name,age from emp order by age desc;
-- 根据入职时间排序
select * from emp order by entrydate;
select * from emp order by entrydate desc;
-- 先根据年龄升序排序，年龄相同的，再根据入职时间降序排序
select name,age,entrydate from emp order by age asc,entrydate desc;
```

分页查询：

1、语法

select 字段列表 from 表名 limit 起始索引,查询记录数;

注：

)若起始索引从0开始，起始索引=（查询页码-1）*每页显示记录数

）分页查询是数据库的方言（数据库之间不同的地方），不同的数据库有不同的实现，MySQL中的方言是limit

）如果查询的是第一页数据，起始索引可以省略，直接简写为limit 10

例图：

```
-- 查询第一页员工数据，每页展示10条记录
select *from emp limit 0,10;
select *from emp limit 10;

-- 查询第二页员工数据，每页展示10条记录
-- 起始索引=（页码-1）*该页展示记录数
select *from emp limit 10,10;
```

执行顺序：

1、from　　表名列表

2、where　　条件列表

3、group by 分组字段列表
4、select 字段列表
5、having 分组后条件列表
6、order by 排序字段列表——asc升序、desc降序（不写默认是升序）
7、limit 分页参数

用户管理：

1、查询用户

```
use mysql;
select *from user;
```

2、创建用户

```
create user '用户名'@'主机名' identified by '密码';
```

3、修改用户密码

```
alter user '用户名'@'主机名' identified with mysql_native_password by '新密码';
```

4、删除用户

```
drop user '用户名'@'主机名';
```

注：1、若想让创建的用户可以在任意主机访问数据库，则将主机名写为%
2、这类语句对于开发人员来说使用较少，一般为数据库管理员使用

权限控制：

常用权限：

all，all privileges	所有权限
select	查询数据
insert	插入数据
update	修改数据
delete	删除数据
alter	修改表
drop	删除数据库/表/视图
create	创建数据库/表

更多的可以查询官方文档

语法

1、查询权限
show grants for ’用户名’@’主机名’;

2、授予权限
grant 权限列表 on 数据库名,表名 to ’用户名’@’主机名’;

3、撤销权限
revoke 权限列表 on数据库名,表名 from ’用户名’@’主机名’;

注：1、多个权限用逗号分隔
　　2、授权时，数据库和表名可以使用*进行通配，代表所有
例图：

```
-- 查询权限
show grants for 'heima'@'%';

-- 授予权限
grant all on root.* to 'heima'@'%';

-- 撤销权限
revoke all on root.* from 'heima'@'%';
```

函数：

字符串函数

concat(S1,S2...Sn)　　字符串拼接，将 S1,S2,...Sn 拼接成一个个字符串

lower(str)　　将字符串 str 全部转为小写
upper(str)　　将字符串 str 全部转为大写

lpad(str,n,pad)　　左填充，用字符串 pad 对 str 的左边进行填充，达到 n 个字符串长度

rpad(str,n,pad)	右填充，用字符串 pad 对 str 的右边进行填充，达到 n 个字符串长度
trim(str)	去掉字符串头部和尾部的空格
substring(str,start,len)	返回从字符串 str 从 start 位置起的 len 个长度的字符串

使用方法：select 函数(参数)

例：

```
select concat('早上好','中国','现在我有冰淇凌');
select lower('HHHHH');
select upper('hhhhh');
select lpad('01',5,'-');
select rpad('01',5,'-');
select trim('   123 321    ')
select substring('早上好中国现在我有冰淇凌',1,3);
```

数值函数

ceil(x)	向上取整
floor(x)	向下取整
mod(x,y)	返回 x/y 的模
rand()	返回 0~1 内的随机数
round(x,y)	求参数 x 的四舍五入的值，保留 y 位小数

使用方法：select 函数(参数)

例：

```
select rand();
select round(3.1415,2);

select round(rand(),6)*10e5;
select lpad(round(rand(),6)*10e5,6,0);
select lpad(round(rand()*1000000,0),6,0);
```

日期函数

curdate()	返回当前日期

curtime()	返回当前时间
now()	返回当前日期和时间
year(date)	获取指定 date 的年份
month(date)	获取指定 date 的月份
day(date)	获取指定 date 的日期
date_add(date,interval expr type)	返回一个指定日期/时间值加上一个时间间隔expr 后的时间值
datediff(date1,date2)	返回起始时间date1和结束时间date2之间的天数

注：1、date_add(date,interval expr type)中，expr type为时间间隔的时间值
2、datediff(date1,date2)的值为date1减去date2

流程控制函数

if(value,t,f)	如果value为true，则返回t，否则返回f
ifnull(value1,value2)	如果value不为空，返回value1，否则返回value2
case when [val1] then [res1]...else [default] end	如果val1为true，返回res1，否则返回default默认值
case[expr] when [val1] then [res1]...else [default] end	如果expr的值等于val1，返回res1，...否则返回default默认值

例图：

```
select
    id,
    name,
    (case when math >= 85 then '优秀' when math >=60 then '及格' else '不及格' end ) '数学',
    (case when english >= 85 then '优秀' when english >=60 then '及格' else '不及格' end ) '英语',
    (case when chinese >= 85 then '优秀' when chinese >=60 then '及格' else '不及格' end ) '语文'
from score;
```

注：ifnull(value1,value2)中，value1为空（即”）时，不返回值，填写为null时，才返回value2

约束：

概述

非空约束　　限制该字段的数据不能为null
not null

唯一约束　　保证该字段的所有数据都唯一且不重复
unique

主键约束　　主键是一行数据的唯一标识,要求非空且唯一
primary key

默认约束　　保存数据时,若未指定该字段的值,则为默认值
default

检查约束　　保证字段值满足某一个条件
check

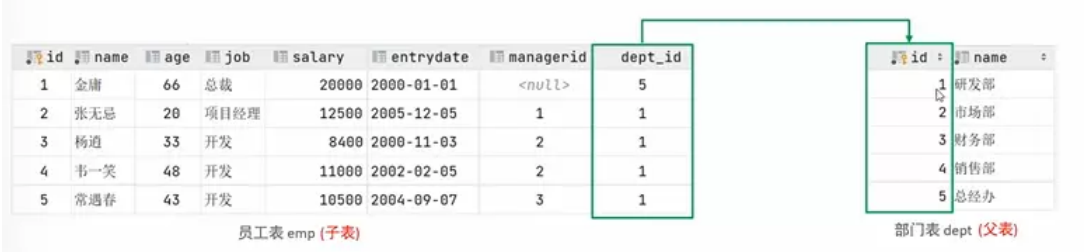
外键约束　　用来让两张表的数据建立连接,保证数据一致和完整
foreign key

注：约束是作用于表中字段上的，可以在创建表/修改表的时候添加约束
　　另外也可以在可视化图形操作中直接进行增删改查操作
例图：

字段名	字段含义	字段类型	约束条件	约束关键字
id	ID唯一标识	int	主键，并且自动增长	PRIMARY KEY, AUTO_INCREMENT
name	姓名	varchar(10)	不为空，并且唯一	NOT NULL，UNIQUE
age	年龄	int	大于0，并且小于等于120	CHECK
status	状态	char(1)	如果没有指定该值，默认为1	DEFAULT
gender	性别	char(1)	无	

外键约束

让两张表的数据库之间建立连接，例如：



id	name	age	job	salary	entrydate	managerid	dept_id
1	金庸	66	总裁	20000	2000-01-01	<null>	5
2	张无忌	20	项目经理	12500	2005-12-05	1	1
3	杨逍	33	开发	8400	2000-11-03	2	1
4	韦一笑	48	开发	11000	2002-02-05	2	1
5	常遇春	43	开发	10500	2004-09-07	3	1

员工表 emp (子表)

id	name
1	研发部
2	市场部
3	财务部
4	销售部
5	总经办

部门表 dept (父表)

注：上述这两张表在数据库层面并未建立连接，所以无法保证数据的一致性和完整性

添加外键

create table 表名（

字段名　数据类型

...

[constraint] [外键名称] foreigen key (外键字段) references 主表 (主表列名)

);

alter table 表名 add constraint 外键名称 foreign key(外键字段名) references 主表 (主表列名);

删除外键

alter table 表名 drop foreign key 外键名称;

例图：

```
-- 建立两张表的外键关联
alter table emps add constraint fk_emps_dept_id foreign key (dept_id) references dept(id);

-- 删除外键
alter table emps drop foreign key fk_emps_dept_id;
```

删除/更新行为

no action与restrict(默认，一般不用特别加上)

当父表中删除/更新对应记录时，首先检查记录是否有对于外键，如果有则不允许删除/更新（这两个用法一致）

cascade

当父表中删除/更新对应记录时，首先检查记录是否有对于外键，如果有，则也删除/更新外键在子表中的记录

set null

当父表中删除对应记录时，首先检查记录是否有对于外键，如果有则设置子表中该外键值为null（这就要求该外键允许取null值）

set default
父表有变更时，子表将外键列设置成一个默认的值（innodb不支持，一般不用）

语法
alter 表名 add constraint 外键名称 foreign (外键字段) references 主表名(主表字段) on update cascade on delete casade;

例图：

```
alter table emps add constraint fk_emps_dept_id foreign key (dept_id) references dept(id) on update cascade on DELETE cascade;

alter table emps add constraint fk_emps_dept_id foreign key (dept_id) references dept(id) on update set null on DELETE set null ;
```

多表查询：

多表关系

1、一对多（多对一）
案例：一个部门对应多个员工，一个员工对应一个部门
实现：在多的地方建立外键，指向一方的主键

2、多对多
案例：一个学生选修多门课程，一门课程供多个学生选择
实现：建立第三张中间表，中间表至少包含两个外键，分别关联两方主键
例图：

```
-- 多对多
create table student(
    id int auto_increment primary key comment '主键ID',
    name varchar(10) comment '姓名',
    no varchar(10) comment '学号'
)comment '学生表';
insert into student values (null,'哈利波特','2000100101'),(null,'韦斯莱','2000100102'),(null,'赫敏','2000100103'),(null,'马尔福','2000100104');

create table course(
    id int auto_increment primary key comment '主键ID',
    name varchar(10) comment '课程名称'
)comment '课程表';

insert into course values (null,'java'),(null,'php'),(null,'mysql'),(null,'hadoop');
```

```
-- 建立中间表，来维护学生表和课程表之间的关系
create table student_course(
    id int auto_increment comment '主键' primary key ,
    studentid int not null comment '学生ID',
    courseid int not null comment '课程ID',
    constraint fk_courseid foreign key (courseid) references course(id),
    constraint fk_studentid foreign key (studentid) references student(id)
)comment '学生课程中间表';

insert into student_course values (null,1,1),(null,1,2),(null,1,3),(null,2,2),(null,2,3),(null,3,4);
```

3、一对一

案例：用户与个人信息的对应关系

实现：将一张表的基础字段放在一张表中，其他详细字段放在另一张表中，在任意一方加入外键，关联另一方的主键，并设置外键为唯一的(unique)

例图：

```
-- ---------------------------------- 一对一 ---------------------------
create table tb_user(
    id int auto_increment primary key comment '主键ID',
    name varchar(10) comment '姓名',
    age int comment '年龄',
    gender char(1) comment '1: 男 , 2: 女',
    phone char(11) comment '手机号'
) comment '用户基本信息表';

create table tb_user_edu(
    id int auto_increment primary key comment '主键ID',
    degree varchar(20) comment '学历',
    major varchar(50) comment '专业',
    primaryschool varchar(50) comment '小学',
    middleschool varchar(50) comment '中学',
    university varchar(50) comment '大学',
    userid int unique comment '用户ID',
    constraint fk_userid foreign key (userid) references tb_user(id)
) comment '用户教育信息表';


insert into tb_user(id, name, age, gender, phone) values
        (null,'黄渤',45,'1','18800001111'),
        (null,'冰冰',35,'2','18800002222'),
        (null,'码云',55,'1','18800008888'),
        (null,'李彦宏',50,'1','18800009999');

insert into tb_user_edu(id, degree, major, primaryschool, middleschool, university, userid) values
        (null,'本科','舞蹈','静安区第一小学','静安区第一中学','北京舞蹈学院',1),
        (null,'硕士','表演','朝阳区第一小学','朝阳区第一中学','北京电影学院',2),
        (null,'本科','英语','杭州市第一小学','杭州市第一中学','杭州师范大学',3),
        (null,'本科','应用数学','阳泉第一小学','阳泉区第一中学','清华大学',4);
```

注：其中的unique用于保证一对一关系

多表查询概述

多表查询时会出现一个用户对应多个部门的情况，造成因笛卡尔积形成的冗余次数，因此需要加入一个条件来避免

例图：

```
-- 多表查询概述
select *from emps,dept where emps.dept_id=dept_id;
```

分类：

连接查询：

内连接：

相当于查询A、B交集部分的数据

隐式内连接

`select 字段列表 from 表1,表2 where 条件...;`

显式内连接

`select 字段列表 from 表1 [inner] join 表2 on 连接条件...;`

例图：

```
-- 内连接

-- 1、查询每一个员工的姓名，及关联的部门（隐式内连接）
select emps.name,dept.name from emps,dept where emps.dept_id=dept.id;

-- 2、查询每一个员工的姓名，及关联的部门（显式内连接）
select e.name,d.name from emps e inner join dept d on e.dept_id = d.id;
```

外连接：

左外连接：查询左表(表1)所有数据，以及两张表交集部分数据

`select 字段列表 from 表1 left [outer] join 表2 on 条件...;`

右外连接：查询右表(表2)所有数据，以及两张表交集部分数据

`select 字段列表 from 表1 right [outer] join 表2 on 条件...;`

例图：

```
-- 外连接

-- 1、查询emps表的所有信息，和对应的部门信息（左外连接）
select emps.*,dept.name from emps left outer join dept on emps.dept_id=dept.id;

-- 1、查询dept表的所有信息，和对应的部门信息（右外连接）
select d.*,e.* from emps e right join dept d on e.dept_id =d.id ;
```

注：条件都是emps.dept_id=dept.id;

自连接：

当前表与自身的连接查询，自链接必须使用表别名

语法：

`select 字段列表 from 表A 别名A join 表A 别名B on 条件...;`

自连接查询，可以是内连接查询，也可以是外连接查询。

注：自连接因为用的是同一张表，所以一定要起别名

例图：

```sql
-- 自连接

-- 1、查询员工及其所属领导的名字
select a.name,b.name from emps a,emps b where a.managerid=b.id;

-- 2、查询所有员工emps及其领导的名字emps，如果员工没有领导，也要查询出来
select a.name '员工',b.name '领导' from emps a left join emps b on a.managerid=b.id;
```

联合查询：

对于union查询，就是把多次查询的结果合并起来，形成一个新的查询结果集

语法：

select 字段列表 from 表A...
union [all]
select 字段列表 from 表B...;

注：

1、对于联合查询的多张表的列数必须保持一致，字段类型也需要保持一致
2、union all会把所有数据直接合并在一起，而union会对合并之后的数据去重

例图：

```sql
-- 1、将薪资低于5000元的员工和年龄大于50岁的员工全部查询出来
select * from emps where salary<5000
union all
select *from emps where age>50;
-- 去除重复部分
select * from emps where salary<5000
union
select *from emps where age>50;
```

子查询：

概念：SQL语句中嵌套查询select语句，成为嵌套语句，又称子查询

语法：

select * from t1 where column1={select column1 from t2}

注：子查询外部的语句可以是insert/update/delete/select的任何一个

根据子查询结果不同，分为：
）标量子查询（子查询结果为单个值）
）列子查询（子查询结果为一列）
）行子查询（子查询结果为一行）
）表子查询（子查询结果为多行多列）

根据子查询位置，分为：where之后、from之后、select之后。

标量子查询：

子查询返回的结果是单个值（数字、字符串、日期等），最简单的形式，这种查询称为标量子查询

常用操作符号：=、<>、>、>=、<、<=

例图：

```
-- 标量子查询


-- 1、查询销售部所有员工信息
select * from emps where dept_id=(select id from dept where name='销售部');

-- 2、查询在张无忌入职之后的员工信息

select * from emps where entrydate>(select entrydate from emps where name='张无忌');
```

列子查询：

子查询返回的结果是一列（可以是多行），这种子查询成为列子查询

常用操作符：in、not in、any、some、all

操作符	描述
IN	在指定的集合范围之内，多选一
NOT IN	不在指定的集合范围之内
ANY	子查询返回列表中，有任意一个满足即可
SOME	与ANY等同，使用SOME的地方都可以使用ANY
ALL	子查询返回列表的所有值都必须满足

例图：

```sql
-- 列子查询


-- 1、查询销售部和市场部的所有员工信息
select * from emps where dept_id in (select id from dept where name='销售部'or name='市场部');

-- 2、查询比财务部所有员工工资都高的员工信息
-- a
select id from dept where name='财务部';
-- b
select salary from emps where dept_id=(select id from dept where name='财务部');
-- c
select *from emps where salary>all(select salary from emps where dept_id=(select id from dept where name='财务部'));

-- 3、查询比研发部其中任意一个人工资都高的员工信息
select *from dept where name ='研发部';
select salary from emps where dept_id=(select *from dept where name ='研发部');
select *from emps where salary>any(select salary from emps where dept_id=(select id from dept where name ='研发部'));
```

行子查询：

子查询返回的结果是一行（也可以是多列），这种子查询成为行字查询
常用操作符：=、<>、in、not in

例图：

```sql
-- 行子查询


-- 查询与张无忌的薪资及其直属领导相同的员工信息
select salary,managerid from emps where name='张无忌';
select *from emps where (salary,managerid)=(select salary,managerid from emps where name='张无忌');
```

表子查询：

子查询返回的结果是多行多列，这种子查询称为表子查询
常用操作符:in

例图：

```sql
-- 表子查询


-- 1、查询与韦一笑、常遇春的职位和薪资相同的员工信息

-- a:查询职位与薪资
select job,salary from emps where name='韦一笑' or name='常遇春';
-- b:查询与其相同的员工信息
select *from emps where (job,salary) in (select job,salary from emps where name='韦一笑' or name='常遇春');


-- 2、查询入职日期是2004-04-30之后的员工信息及其部门信息

-- a:查询入职日期在其之后的员工信息
select *from emps where entrydate>'2004-4-30';
-- b:查询这部分员工，及其对应的部门信息
select e.*,d.* from (select *from emps where entrydate>'2004-4-30')e left join dept d on e.dept_id=d.id;
```