数据库篇

1.数据发展过程：

- 文件操作

```
"""
小飞----123---18
中飞-456-19
大飞|789|20
"""
```

- 软件开发目录规范

```
"""
流星蝴蝶剑

- config
- core
- lib
- log
- bin
- db
readme.md
"""
```

- 数据库程序

  关系型数据库：

  Mysql、Oracle、SQL server、DB2、Access

  - 数据之间有关系或约束
  - 数据通常以表格的形式储存

name（str）	pwd	age（int）
xiaofei	123	18
xiaoming	258	19
xiaojun	368	20

  非关系型数据库：

  MongoDB、Redis、Memcache

  - 存储数据通常都是key、value的形式

2.Mysql引入

> mysql就是一款基于网络通信的应用程序，底层一定是用的socket

mysql服务端支持自己的客户端操作数据、也支持其他编程语言，**怎么解决语言不通的问题?**

- 让服务端精通所有编程语言
- 统一语言（sql语句）

3.MySQL安装

参考mysql安装文档：https://active.clewm.net/FrcyFA

4.SQL语句初体验

```
-- 启动和连接数据库
net start mysql -- 启动数据库服务(需要管理员权限)
net stop mysql -- 关闭数据库服务(需要管理员权限)
mysql -u root -p -- 连接到数据库

-- 每一条sql语句都是以分号结尾的

-- 库 --> 文件夹
-- 表 --> 文件
-- 记录 --> 文件里面的一行行数据
-- 杠杠是单行注释
/*
多行注释
*/

show databases; --查看所有数据库
\s -- 查看数据库字符编码以及其他信息的
\c -- 结束当前语句，或者用ctrl+c
exit; -- 退出连接
quit; -- 退出连接
help <命令> -- 查看命令的帮助信息
```

- 操作库

```
-- 增（create）
create database db1;
create database db1 charset=utf8;  -- 推荐
-- 删（drop）
drop database db2; -- 删除库
-- 改（alter）
alter database db2 charset=utf8; -- 改数据库的字符编码(当前mysql5.1版本之后只能改数据库的字符编码，不能改数据库名称了)
-- 查（show、describe(desc)）
show databases; -- 查所有数据库
show create database db1; -- 查一个，查看创建这个数据库的sql语句
```

- 操作表

```
select database(); -- 查看当前所在数据库
use db1; -- 切换数据库
-- 增（create）
```

```sql
create table movies(id int,name char); -- 创建表（默认字符编码，就是库的字符编码）
create table movies(id int,name char) charset=utf8; -- 创建表
-- 删（drop）
drop table movies; -- 删除当前库中的表
-- 改（alter）
alter table movies rename to MOVIES; -- 修改表名
alter table movies modify name char(4); -- 修改表字段类型
alter table movies change name NAME char(5); -- 修改表字段和类型
-- 查（show、describe(desc)）
show tables; -- 查看当前库下所有的表
show create table movies; -- 查看创建这个数据表的sql语句
describe movies; -- 查看一张表的结构，简写：desc movies;

-- 所有对表的操作都可以使用绝对路径的方式，这样即便不操作数据库，也可操作对应数据库的表
create table db.movies(id int,name char(4)) charset=utf8; -- 在指定的库中创建表
drop table db.movies; -- 删除指定库中的表
alter table db.movies change name NAME char(16); -- 修改指定库中表字段和类型
show create table db.movies; -- 查看指定库中的数据库表的sql语句
describe db.movies; -- 查看指定库的一张表的结构，简写：desc db.movies;
```

- 操作记录

```sql
-- 增 (insert into <表名> values())
insert into movies values(1,'流浪地球'); -- 插入一条记录
insert into movies values(1,'流浪地球'),(2,'三体'); -- 插入多条记录
-- 删 (delete from <表名> where)
delete from movies where name='三体';
-- 改 (update xx set)
update movies set name='满江红' where id=1;
-- 查 (select <字段> from <表名>)
select * from db1.movies; -- 查一个表的所有数据
select id,name from db1.movies; -- 查询这张表所有数据的id,name字段
select user,host from mysql.user; -- 查看user表里面所有用户的user字段和host字段
```

5.SQl语句分类

类型	描述	关键字
DDL	数据库定义语言，用来定义和管理数据库或者数据表	create,alter,drop
DML	数据操纵语言，用来操作数据	insert,update,delete
DQL	数据库查询语言，用来查询数据库	select
DCL	数据库控制语言，权限控制	grant,revoke,commit,rollback

6.库的操作

> 参考之前的笔记

```
-- 增（create）
create database [if not exists] <库名> [charset=utf8];
-- 删（drop）
drop database [if not exists] <库名>;
-- 改（alter）
alter database <库名> [charset=utf8];
-- 查（show、describe(desc)）
show databases;
show create database <库名>;
```

7.表操作

存储引擎

> 表的类型，就是存储引擎

```
show engines; -- 查看所有的存储引擎

'InnoDB':支持事务、行级锁定和外键
'MRG_MYISAM':相同的MyISAM表的集合
'MEMORY':基于哈希，存储在内存中，适用于临时表
'BLACKHOLE':/dev/null存储引擎（您写入其中的任何内容都会消失）
'MyISAM':MyISAM存储引擎
'CSV': CSV存储引擎
'ARCHIVE':归档存储引擎
'PERFORMANCE_SCHEMA': 性能模式
'FEDERATED':联合MySQL存储引擎

Create table t1(id int,name char)engine=innodb;
Create table t2(id int,name char)engine=myisam;
Create table t3(id int,name char)engine=memory;
Create table t4(id int,name char)engine=blackhole;

insert into t1 values(1,'a');
insert into t2 values(1,'a');
insert into t3 values(1,'a');
insert into t4 values(1,'a');
```

创建表的语法

```
create table <表名>(
    <字段名1><字段类型>[(宽度)] [约束条件],
    <字段名2><字段类型>[(宽度)] [约束条件],
    <字段名3><字段类型>[(宽度)] [约束条件],
    <字段名4><字段类型>[(宽度)] [约束条件1,约束条件2]
)engine=innodb charset=utf8;
-- 宽度指的是字符个数，或者说字符串长度，关闭严格模式之后，即便宽度超了，也可以插进去（截取字符），5.6版本之后默认开启严格模式

-- 约束条件,注意顺序
[unsigned][zerofill][not null]
-- unsigned 无符号
-- zerofill 0填充
-- not null 非空
```

修改表的语法

```
-- 修改存储引擎
alter table <表名> engine=<存储引擎名称>;

-- 修改表名
alter table <表名> rename to <新表名>;

-- 增加字段
alter table <表名> add <字段名> <字段类型>[(宽度)] [约束条件] [first|after <字段名>];

-- 删除字段
alter table <表名> drop <字段名>;

-- 修改字段
alter table <表名> modify <字段名> <新字段类型>[(宽度)] [约束条件]; -- 修改表字段类型
alter table <表名> change <旧字段名> <新字段名> <新字段类型>[(宽度)] [约束条件]; -- 修改表字段类型和名称
```

删除和复制表语法

```
-- 删除表
drop table <表名>;

-- 复制创建表
create table <新表名> select * from <旧表名> [条件];

-- 复制创建表结构
create table <新表名> like <旧表名>;
```

8.数据类型

- 数值
    - int(整形)

-- 整形创建的时候设置的宽度是其实是显示宽度，所以在创建时候无需对其设置宽度（这里指的是数据大小），只有以下几种大小，用unsigned约束条件设置为无符号范围，不设置约束条件时默认为有符号范围

类型	大小	范围（有符号）	范围（无符号）	描述
tinyint	1Bytes	(-128，127)	(0,255)	很小的整数
smallint	2Bytes	(-32768，32767)	(0,65535)	较小的整数
mediumint	3Bytes	(-8388608，8388607)	(0,16777215)	一般的整数
int	4Bytes	(-2147483648，2147483647)	(0,4294967295)	标准的整数
bigint	8Bytes	(-9223372036854775808，9223372036854775807)	(0,18446744073709551615)	极大的整数

- float

  ==乘二取整==

  3.625

  3 -> 11

  0.625 -> 0.625x2 = ==1==.25 0.25x2 = ==0==.5 0.5x2 = ==1== -> 0.101

  3.625 -> 11.101

  > 浮点数的表现形式：==符号.尾数==.基数.==指数==
  >
  > ±m*2^e^
  >
  > 单精度浮点数（32位）：符号（1位）+指数（8位）+尾数（23位）
  >
  > 双精度浮点数（64位）：符号（1位）+指数（11位）+尾数（52位）

  符号：1 -> 负，0 -> 正或0

  指数：

  10000001 -> 129 -> 2

10000000 -> 128 -> 1

01111111 -> 127-> 0

01111110 -> 126 -> -1

01111101 -> 125 -> -2

尾数：把小数点前的二进制数，固定为1的规则

3.625 x 2^-1^ -> 3.625 x 0.5 = 1.8125

11.101 -> 1.10100000000000000000000 尾数：==10100000000000000000000==

1.8125 x 2^1^ = 3.625 指数：==^1^== -> 127+1=128 -> ==10000000==

3.625 -> 11.101的存储方式：0 10000000 10100000000000000000000

1.1101 -> 1 * 2^0^ + 1 * 2^-1^ + 1 * 2^-2^ + 0 * 2^-3^ + 1 * 2^-4^ = 1 + 0.5 + 0.25 + 0 + 0.0625 = 1.8125 * 2^1^ = 3.625

3.1

3 -> 11

0.1 -> 0.1x2 = 0.2 0.2x2 = 0.4 0.4x2 = 0.8 0.8x2 = 1.6 0.6x2 = 1.2

3.1 -> 11.0 0011 0011 0011 0011.....

精度问题：如果小数最后一位不是5的话，不精确，延长循环让精度更高

无法保存极小的小数：最小非零值

类型	大小（m表示所有位数，d表示小数位数）	范围（有符号）	范围（无符号）	描述
float	4 Bytes float(m, d) m最多255，d最多30	(-3.402 823 466 E+38，-1.175 494 351 E-38)，0，(1.175 494 351 E-38，3.402 823 466 351 E+38)	0，(1.175 494 351 E-38，3.402 823 466 E+38)	单精度浮点数值
double	8 Bytes double(m, d) m最多255，d最多30	(-1.797 693 134 862 315 7 E+308，-2.225 073 858 507 201 4 E-308)，0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	双精度浮点数值
decimal	decimal(m,d) m最大65，d最大30	依赖于m和d的值	依赖于m和d的值	小数值

- 字符
  - char（定长字符串）

> char(10) 最多能存10个字符，如果超过10个字符，会直接报错，如果不足10个字符，空格补齐(会补充在数据尾部，但查询的时候会自动把尾部的空格去掉，所以当尾部数据有空格时，不要用char)
>
> 缺点：浪费空间
>
> 优点：存取速度快
>
> 范围：0-255字符

  - varchar（变长字符串）

> varchar(10) 最多能存10个字符，如果超过10个字符，会直接报错，如果不足10个字符，直接存
>
> 优点：节省空间
>
> 缺点：存取速度慢　头hello头fei头heiheihei
>
> 范围：0-65535字节，一行的最大字节数。（注：数据库一行数据的最大字节不可超过65535）
>
> 最大字符数 =（65535-行其它字段总字节数-null标志字节数-长度标志字节数）/ 字符集单字符最大字节数。（注：utf8字符集一个字符占三个字节，gbk字符集一个字符占三个字节）

- 时间日期
  - year(年)
  - date(日期：年-月-日)
  - time(时间：时-分-秒)
  - datetime(日期时间：年-月-日 时-分-秒)

> 8字节，1000～9999

  - timestamp

> 4字节，1970～2038

```
create table user(
    id int,
    name varchar(16),
    born year,
    brith date,
    active time,
    reg_time datetime);
insert into user values(1,'fei',now(),now(),now(),now());
insert into user values(2,'jack','2025','2025-10-10','12:00:00','2025-10-10 12:00:00');
```

- 枚举

> enum：单选

```
create table t14(
    id int,
    name varchar(16),
    gender enum('male','female','other')
    );
insert into t14 values(1,'fei','male');
-- 注：只能在给定的范围内插入一条
```

- 集合

> set：多选

```
create table t15(
    id int,
    name varchar(16),
    hobby set('tea','cola','beer')
    );
insert into t15 values(1,'fei','tea');
insert into t15 values(1,'fei','tea,cola');
-- 注：可在给定的范围内插入多条，用逗号隔开
```

9.约束条件

-- 约束条件,注意顺序([unsigned] [zerfill]放前面)
-- unsigned zerofill not null default unique

- unsigned 无符号
- zerfill 0填充
- not null 非空
- default 设置默认值
- unique 唯一

```
create table t16(id int unsigned zerofill not null);-- unsigned、zerofill属于类型约束，not null属于字段约束，所以约束条件需要先写类型约束后写字段约束
| Field | Type                        | Null | Key | Default | Extra |
+-------+-----------------------------+------+-----+---------+-------+
| id    | int(10) unsigned zerofill | NO   |     | NULL    |       |

-- 单列唯一:一个字段单独唯一
create table user(id int unique, name vachar(16) unique);

create table user(id int, name vachar(16),
                  unique(id),
                  unique(name)); -- 可把unique写在字段类型后，或者写为unique(字段名称)
-- 联合唯一：多个字段一起唯一
create table app(id int,
                 host varchar(15),
                 port int,
                 unique(host,port),
                 unique(id));
```

```
insert into app values
(1,'192.168.3.1',3306),
(2,'192.168.3.2',9000),
(3,'192.168.3.2',3306);

insert into app values
(4,'192.168.3.1',3306);
```

- primary key 主键（约束特性：不为空，且唯一）

```
-- 单列主键
create table t19(
    id int primary key,
    name varchar(16));

insert into t19 values(1,'fei'),(2,'jack');
insert into t19 values(1,'rose'); -- 插入相同的id报错
insert into t19(name) values('rose'); -- 插入空id报错

-- 创建表时不指定主键的话，会默认把唯一且不为空的字段指定为组件
create table t20(
    name varchar(16),
    age int unique not null);

-- 复合主键
create table app1(id int,
                  host varchar(15),
                  port int,
                  primary key(host,port),
                  unique(id));

insert into app1 values
(1,'192.168.3.1',3306),
(2,'192.168.3.2',9000),
(3,'192.168.3.2',3306);

insert into app values
(4,'192.168.3.1',3306);
```

- auto_increment：自动递增

```
create table db4;
use db4;
create table t1(id int primary key auto_increment,
                name varchar(16)
               );

insert into t1(name) values('fei'),('jack'),('rose');-- 无需传id，自动递增（会在最后插入的id基础上递增）

-- 清空表(删除表数据)
delete from t1; -- 清空所有数据，不清空自增值
delete from t1 where id=1; -- 删除id为1的数据
truncate t1; -- 清空所有数据，并且清空自增值
```

- foreign key：外键约束

```
create table dep(
    id int primary key,
    name varchar(16),
    `desc` varchar(64)
);

create table emp(
    id int primary key,
    name varchar(16),
    gender enum('male','female'),
    mobile varchar(11),
    dep_id int,
    foreign key(dep_id) references dep(id)
    on delete cascade
    on update cascade
); -- 把emp表的dep_id字段关联到dep表的id字段
-- on delete cascade on update cascade的作用：删除与更新同步，使得存在外键的表不需要去改或者删除外键表的数据，可之间修改本数据表。当本数据表中引用的外键数据没有了时自动删除外键表中的该条数据，当本数据表中的外键数据更新时自动更新外键表

insert into dep values(1,'研发部','造火箭的'),(2,'销售部','卖火箭的'),(3,'人事部','裁员的');
insert into emp values(1,'fei','male','131',1),
    (2,'大仙','male','137',2),
    (3,'tom','male','138',2),
    (4,'jack','male','135',3),
    (5,'rose','female','139',3);

delete from emp where dep_id=1;
delete from dep where id=1;

update dep set name = '市场部' where id=2;
update dep set id = 999 where id=2;

-- 大型项目不建议用外键，小型项目可用
```

10.关系

- 多对一或一对多

```
create table dep(
    id int primary key,
    name varchar(16),
    `desc` varchar(64)
);

create table emp(
    id int primary key,
    name varchar(16),
    gender enum('male','female'),
    mobile varchar(11),
    dep_id int,
    foreign key(dep_id) references dep(id)
    on delete cascade
    on update cascade
);

insert into dep values(1,'研发部','造火箭的'),(2,'销售部','卖火箭的'),(3,'人事部','裁员的');
insert into emp values(1,'fei','male','131',1),
    (2,'大仙','male','137',2),
    (3,'tom','male','138',2),
    (4,'jack','male','135',3),
    (5,'rose','female','139',3);
```

- 多对多

```
create table song(
    id int primary key auto_increment,
    name varchar(16) not null
);
create table singer(
    id int primary key auto_increment,
    name varchar(16) not null
);
create table song2singer(
    id int primary key auto_increment,
    singer_id int not null,
    song_id int not null,
    foreign key(singer_id) references singer(id) on delete cascade on update cascade,
    foreign key(song_id) references song(id) on delete cascade on update cascade
); -- 这张表是用来建立上面两张表的多对多关系的，即将两张表的外键都建立在另外一张表内

insert into song(name) values
('以父之名'),
('夜的第七章'),
('止战之殇'),
('夜曲'),
('北京欢迎你');
```

```
insert into singer(name) values
('周杰伦'),
('刘欢'),
('韩红'),
('成龙');
insert into song2singer(singer_id, song_id) values
(1,1),
(1,2),
(1,3),
(1,4),
(2,5),
(3,5),
(4,5);
```

- 一对一

```
create table customer(
    id int primary key,
    name varchar(16),
    gender enum('male', 'female'),
    mobile varchar(11),
);

create table owner(
    id int primary key,
    room_number varchar(16),
    area int,
    is_loan enum(true,false),
    customer_id int unique,
    foreign key(customer_id) references customer(id) on delete cascade on update
cascade
); -- 保证customer_id的唯一性即可保证数据表的一对一性质


insert into customer values
(1, 'fei', 'male', '131'),
(2, '大仙', 'female', '150'),
(3, 'tom', 'female', '139'),
(4, 'jack', 'male', '135'),
(5, 'rose', 'male', '137');

insert into owner values
(1, '688', 300, false, 2),
(1, '233', 180, true, 4);
```

11、记录的查询语法

```
select distinct <字段....> from <库名>.<表名>
    where <条件>
    group by <分组条件>
    having <过滤条件>
    order by <排序字段> {ASC | DESC} -- 默认升序ASC
    limit n;
```

例子

```
create database db5;
use db5;
create table emp(
    id int primary key auto_increment,
    name varchar(16) not null,
    gender enum('male', 'female') not null,
    age int not null,
    salary float(10, 2),
    dep varchar(32),
    notes varchar(64)
);

insert into emp(name, gender, age, salary, dep) values
('关羽', 'male', 20, 8000, '技术部'),
('张飞', 'male', 25, 12000, '技术部'),
('赵云', 'male', 19, 6800, '技术部'),
('马超', 'male', 26, 11000, '技术部'),
('黄忠', 'female', 48, 15000, '技术部'),
('夏侯惇', 'male', 36, 34000, '技术部'),
('典韦', 'male', 19, 6500, '技术部'),
('吕布', 'female', 20, 9000, '技术部'),
('周瑜', 'female',32, 36000, '技术部'),
('文丑', 'male', 27, 24000, '技术部'),

('刘备', 'male', 32, 4000, '市场部'),
('诸葛亮', 'male', 27, 2700, '市场部'),
('庞统', 'male', 37, 4200, '市场部'),
('徐庶', 'male', 36, 4000, '市场部'),
('荀彧', 'male', 25, 2400, '市场部'),
('荀攸', 'male', 25, 2400, '市场部'),
('鲁肃', 'male', 43, 4300, '市场部'),
('司马懿', 'female', 44, 5000, '市场部'),
('杨修', 'male', 19, 800, '市场部'),
('丁仪', 'male', 49, 3500, '市场部'),

('宋江', 'male', 30, 4000, '人事部'),
('吴用', 'male', 38, 3000, '人事部'),
('扈三娘', 'female', 42, 2500, '人事部'),
('顾大嫂', 'female', 38, 3300, '人事部'),
('孙二娘', 'female', 32, 2400, '人事部'),
('丁得孙', 'male', 32, 2800, '人事部'),


('柴进', 'male', 30, 4200, '财务部'),
('卢俊义', 'male', 44, 4000, '财务部');

-- 四则运算
select name,salary*12 years from emp;
select name,salary*12 as yearly_salary years from emp;
select name,salary*12 yearly_salary years from emp;

-- 设置显示格式
-- 姓名：关羽 年龄：20 薪资：8000
```

```
-- 姓名：张飞 年龄：25 薪资：12000
-- concat
select concat('姓名：',name, '  年龄：',age) as '姓名，年龄', concat('薪资：',salary)
as '薪资' from emp;
select concat(name,'-',age,'-',salary) as '姓名-年龄-薪资' from emp;
-- concat_ws
select concat_ws('-',name,age,salary) as '姓名-年龄-薪资' from emp;
```

- where

```
select name,age from emp where age > 35;
select name,age,dep from emp where dep = '技术部' and age > 35;
-- 年龄大于等于30，小于等于40
select name,age from emp where age >= 30 and age <= 40;
select name,age from emp where age between 30 and 40; -- 闭区间，包含30和40，用
between and

-- 年龄小于30，或者大于40
select name,age from emp where age<30 or age>40;
select name,age from emp where not age between 30 and 40; -- 对30到40之间取反就是小
于30，或者大于40

-- 工资为3000，4000，5000的员工
select name,salary from emp where salary=3000 or salary=4000 or salary=5000;
select name,salary from emp where salary in (3000,4000,5000); -- in表示是否在此范围
内

-- is null
select * from emp where notes=''; -- 空字符串不是null
select * from emp where notes is null; -- 查询是否为空字符串
select * from emp where notes is not null;

-- like模糊匹配
-- _表示正则里面的.
-- %表示正则里面的.*
select * from emp where name like '荀_'; -- 有几个下划线表示有几个字符
select * from emp where name like '丁__';
select * from emp where name like '丁%'; -- 百分号表示多个字符

-- 正则表达式
select * from emp where name regexp '^丁.*'; -- regexp后面跟正则表达式

-- 查询丁或者荀开头的名字
select * from emp where name regexp '^(丁|荀).*';
```

- group by

```
-- group by必须放在where后面
select dep from emp group by dep; -- 分组之后只能查分组的字段

-- 聚合函数
count
max
min
```

```
sum
avg

-- 统计每个部门的员工数量
select dep,count(id) as dep_count from emp group by dep;

-- 每组的年龄最大是多少
select dep,max(age) from emp group by dep;

-- 每组最低工资
select dep,min(salary) from emp group by dep;

-- 每组工资总和
select dep,sum(salary) from emp group by dep;

-- 每组平均工资
select dep,avg(salary) from emp group by dep;

-- 统计所有员工的平均工资
select avg(salary) from emp;

-- 查询每个部门所有员工名字
select dep, group_concat(name) from emp group by dep; -- group_concat用于对分组内容进行拼接

-- 统计每个部门年龄大于45的员工数量
select dep,count(id) from emp where age>45 group by dep;

-- 统计男员工和女员工的数量
select gender,count(id) from emp group by gender;

select gender g,count(id) as emp_count from emp group by g having emp_count>10; -- 别名用法举例
```

- having

```
-- 查询所有部门内，员工数量小于5的部门，以及该部门内的员工名字和员工数量
select dep,count(id),group_concat(name) from emp group by dep having count(id)<5;

-- 查询各部门年龄大于35的员工超过3个人的部门名，以及大于35的人数
select dep,count(id) from emp where age>35 group by dep having count(id)>3;
```

- order by

```
-- 对所有员工，按工资进行升序排列
select * from emp order by salary asc; -- asc可不写，因为默认就是asc升序

-- 对所有员工，按工资进行降序排列
select * from emp order by salary desc;

-- 对所有员工，按工资进行降序排列,工资相同的按id字段升序排列
select * from emp order by salary desc, id asc;
```

- limit

```
-- 只要展示五条数据
select * from emp limit 5;

-- 找出公司工资最高的5个人
select * from emp order by salary desc limit 5;

-- 分页
select * from emp limit 0,10; -- 从第0条开始取十条
select * from emp limit 10,10; -- 从第10条开始取十条
select * from emp limit 20,10; -- 从第20条开始取十条
```

12、函数

- 字符串处理函数

```
select concat('a', 'b', 'c');       -- 字符串拼接
select lower('HELLO');              -- 转小写
select upper('hello');              -- 转大写
select lpad('hello', 10, '-');      -- 左填充，用字符串"-"在"hello"的左边填充至10个字符，原字符串长度超过10，则会截取至10个字符
select rpad('hello', 10, '-');      -- 右填充，用字符串"-"在"hello"的右边填充至10个字符，原字符串长度超过10，则会截取至10个字符
select trim('   hello  ');          -- 去掉字符串两端的空格
select substring('hello',2,3);      -- 字符串切片，从第2个字符开始，切3个字符
```

- 数值处理函数

```
select ceil(1.3);           -- 向上取整
select floor(1.6);          -- 向下取整
select mod(10,3);           -- 获取10/2的模
select rand();              -- 获取0～1的随机小数
select round(3.1415926,4);  -- 对3.1415926四舍五入，保留4位小数
```

- 日期处理函数

```
select '当前日期', curdate();                          -- 获取当前日期
select '当前时间', curtime();                          -- 获取当前时间
select '当前日期时间', now();                         -- 获取当前日期和时间
select '年', year('2023-10-08 18:36:59');          -- 获取date_time的年份
select '月', month('2023-10-08 18:36:59');         -- 获取date_time的月份
select '日', day('2023-10-08 18:36:59');           -- 获取date_time的日期
select '时', hour('2023-10-08 18:36:59');          -- 获取date_time的小时
select '分', minute('2023-10-08 18:36:59');        -- 获取date_time的分钟
select '秒', second('2023-10-08 18:36:59');        -- 获取date_time的秒

-- 时间差计算，interval单位:year,month,day,hour,minute,second, microsecond
select date_add(now(), interval 1 year);                    -- 获取从now()开始,1年后的时间
select date_add('2023-10-08 18:36:59', interval 1 month);   -- 获取指定时间1月后的时间
select datediff('2025-10-08 18:36:59', now());              -- 获取两个时间之间的天数，第一个值减去第二个值
```

```
select timediff('2023-10-08 18:36:59', '2023-10-08 12:00:00'); -- 获取两个时间之间的时间差值
```

- 流程控制函数

```
-- 单分支
select name, if(gender='male', '小哥哥', '小姐姐')as '性别' from emp;   -- 如果性别等于male返回小哥哥，否则返回小姐姐
select ifnull(gender, '未知') from emp;                -- 如果第一个值不为空，则返回第一个值，否则返回第二个值
select ifnull(notes, '没有任何内容哦') from emp;

-- 多分支
-- case..when..then..end
-- case <字段> when <条件1> then <条件1满足返回> when <条件2> then <条件2满足返回>...end
-- 工资大于10000：核心员工
-- 工资5000-10000：普通员工
-- 工资5000以下：新员工
select
    name,
    case when salary>10000 then '核心员工' when salary>5000 then '普通员工' else '新员工' end as '员工级别'
from emp;

-- 技术部：高级技术顾问
-- 人事部：HR
-- 其它：销售经理
select
    name,
    case dep when '技术部' then '高级技术顾问' when '人事部' then 'HR' else '销售经理' end as '职位'
from emp;
```

13、多表查询

```
create database db6;
use db6;
create table dep(
    id int primary key auto_increment,
    name varchar(16),
    notes varchar(64)
);
insert into dep(name) values
('总经办'),('技术部'),('市场部'),('人事部'),('财务部'),('后勤部');

create table emp(
    id int primary key auto_increment,
    name varchar(16) not null,
    gender enum('male', 'female') not null,
    age int not null,
    salary float(10, 2),
    post varchar(16),
    join_date date,
```

```
    leader_id int,
    dep_id int
);

insert into emp values
(1, '刘备', 'male', 32, 4000, '总经理', '2035-06-01', null, 1),

(2, '关羽', 'male', 20, 8000, '技术总监', '2035-06-05', 1, 2),
(3, '张飞', 'male', 25, 12000, '项目经理', '2035-06-10', 2, 2),
(4, '赵云', 'male', 19, 6800, '产品经理', '2035-06-10', 2, 2),
(5, '马超', 'male', 26, 11000, '后端开发', '2035-07-11', 2, 2),
(6, '黄忠', 'female', 48, 15000, '后端开发', '2035-07-22', 2, 2),
(7, '夏侯惇', 'male', 36, 34000, '后端开发', '2035-07-29', 2, 2),
(8, '典韦', 'male', 19, 6800, '后端开发', '2035-08-02', 2, 2),
(9, '吕布', 'female', 20, 9000, '前端开发', '2035-08-03', 2, 2),
(10, '周瑜', 'female',32, 36000, '前端开发', '2035-08-08', 2,  2),
(11, '文丑', 'male', 27, 24000, '测试', '2035-08-12', 2, 2),

(12, '诸葛亮', 'male', 27, 8000, '市场总监', '2035-06-05', 1, 3),
(13, '庞统', 'male', 37, 4200, '销售', '2035-06-06', 12, 3),
(14, '徐庶', 'male', 36, 4000, '销售', '2035-06-12', 12, 3),
(15, '荀彧', 'male', 25, 2400, '销售', '2035-06-10', 12, 3),
(16, '荀攸', 'male', 25, 2400, '销售', '2035-06-12', 12, 3),
(17, '鲁肃', 'male', 43, 4300, '销售', '2035-06-18', 12, 3),
(18, '司马懿', 'female', 44, 5000, '销售', '2035-06-20', 12, 3),
(19, '杨修', 'male', 19, 800, '销售', '2035-07-10', 12,  3),
(20, '丁仪', 'male', 49, 3500, '销售', '2035-07-11', 12, 3),

(21, '宋江', 'male', 30, 8000, '人事总监', '2035-06-05', 1, 4),
(22, '吴用', 'male', 38, 3000, '人事主管', '2035-06-06', 21, 4),
(23, '扈三娘', 'female', 42, 2500, '招聘专员', '2035-06-11', 21, 4),
(24, '顾大嫂', 'female', 38, 3300, '招聘专员', '2035-06-25', 21, 4),
(25, '孙二娘', 'female', 32, 2400, '绩效专员', '2035-07-22', 21, 4),
(26, '丁得孙', 'male', 32, 2800, '培训专员', '2035-08-10', 21, 4),


(27, '柴进', 'male', 30, 8000, '财务总监', '2035-06-05', 1, 5),
(28, '卢俊义', 'male', 44, 4000, '会计', '2035-08-19', 27, 5),
(29, '晁盖', 'male', 44, 3500, '出纳', '2035-08-20', 27, 5),

(30, '貂蝉', 'female', 36, 800, null, '2035-09-01', null, null);
```

笛卡尔积

> a b c
>
> 1 2 3
>
> a1 a2 a3 b1 b2 b3 c1 c2 c3

- 连接查询
  - 内连接（取的是两张表的交集）
    - 隐式内连接

```
select * from emp,dep where emp.dep_id=dep.id;
```

- 显式内连接

```
select <字段> from <表a> [inner]join <表b> on <条件>;

select * from emp join dep on emp.dep_id=dep.id;
```

- 外连接
  - 左外连接（取的是左表所有数据和交集部分）

    > 查询左表所有数据，包括交集部分

    ```
    select <字段> from <表a> left [outer] join <表b> on <条件>;

    -- 查询所有员工数据和部门信息
    select emp.name,dep.name from emp left join dep on emp.dep_id =
    dep.id;
    ```

  - 右外连接（取的是右表所有数据和交集部分）

    > 查询右表所有数据，包括交集部分

    ```
    select <字段> from <表a> right [outer] join <表b> on <条件>;

    -- 查询所有部门数据以及对应员工数据
    select dep.name,emp.name from emp right join dep on emp.dep_id =
    dep.id;
    ```

- 自连接（是同一张表在连接）

> select <字段> from <表a> join <表b> **on <条件>;**
>
> **可以用内连接、外连接**

```
-- 查询员工和对应的领导名字
select a.name '员工',b.name '领导' from emp a join emp b on
a.leader_id=b.id;
-- 查询员工和对应的领导名字,包括没有领导的
select a.name '员工',b.name '领导' from emp a left join emp b on
a.leader_id=b.id;
```

- **联合查询**

> **把多次查询的结构合并在一起，但多条select的列数必须相同。关键字：union，union all**
>
> **select .... union [all] select ....;**
>
> **异同：使用union可去除重复数据，union all不行**

```sql
-- 查询薪资大于等于15000的员工，和部门表，以及年龄大于等于45的员工
select * from emp where salary>=15000
union
select * from emp where age>45;

-- 查询查询薪资大于等于15000的员工，和部门表
select name,age,gender from emp where salary>=15000
union
select * from dep;
```

- 子查询

> select之后，from之后，where之后

  - 标量子查询：子查询的结果是单个值

> 操作符：比较运算符

```sql
-- 查询技术部所有员工信息
    -- 1、查询技术部id
    select id from dep where name='技术部';
    -- 2、查询技术部所有员工信息
    select * from emp where dep_id=(select id from dep where name='技术部');

-- 查询薪资比黄忠高的员工信息
    -- 1、查询黄忠的薪资
    select salary from emp where name='黄忠';
    -- 2、查询薪资比黄忠高的员工信息
    select * from emp where salary>(select salary from emp where name='黄忠');
```

  - 列子查询：子查询的结果是一列
    - in
    - not in
    - any：子查询返回列表中，有任何一个满足即可
    - some：同any
    - all：子查询返回的列表的所有值，都必须满足

```sql
-- 列子查询
-- 查询人事部和财务部的所有员工
    -- 1、查询人事部和财务部id
    select id from dep where name in ('人事部','财务部');
    -- 2、根据id查询员工信息
    select * from emp where dep_id in (select id from dep where name in ('人事部','财务部'));

-- 查询比市场部所有人入职都晚的员工信息
    -- 1、查询市场部的id
    select id from dep where name = '市场部';
    -- 2、查询市场部所有人的入职日期
```

```
    select join_date from emp where dep_id=(select id from dep where
name = '市场部');
    -- 3、比市场部所有人入职都晚的员工信息
    select * from emp where join_date>all(select join_date from emp
where dep_id=(select id from dep where name = '市场部'));

    select max(join_date) from emp where dep_id=(select id from dep
where name = '市场部');
    select * from emp where join_date>(select max(join_date) from emp
where dep_id=(select id from dep where name = '市场部'));
```

- **行子查询：子查询的结果是一行**

> **操作符： = != in not in**

```
-- 行子查询
-- 查询和关羽薪资相同，并且领导也相同的员工信息
    -- 1、查询关羽的薪资和领导id
    select salary,leader_id from emp where name='关羽';
    -- 2、查询其他员工信息
    select * from emp where (salary,leader_id)=(select
salary,leader_id from emp where name='关羽');

-- 查询和关羽薪资不相同，并且领导也不相同的员工信息
select * from emp where (salary,leader_id)!=(select salary,leader_id
from emp where name='关羽');

-- in用法
select * from emp where (salary,leader_id)in ((8000,1),(6800,2));

-- not in用法
select * from emp where (salary,leader_id)not in ((8000,1),(6800,2));
```

- **表子查询：子查询的结果是多列**

> **操作符：in not in**

```
-- 表子查询
select salary,leader_id from emp where name = '关羽' or name = '赵云';
select * from emp where (salary,leader_id)in (
select salary,leader_id from emp where name = '关羽' or name = '赵云');

-- 查询工资为八千的员工信息，以及部门信息
    -- 1、查询工资为八千的员工信息
    select * from emp where salary=8000;
    -- 2、 查询这部分员工的部门信息
    select e.*,d.name from (select * from emp where salary=8000) e
left join dep d on e.dep_id=d.id;
    -- 其他解法
    select emp.*,dep.name from emp left join dep on emp.dep_id =
dep.id where emp.salary=8000;
```