第二周：认识sql并学习数据库的基础操作

关系型数据库和非关系型数据库

- 关系型数据库
  - 常见类型
    - Oracle、DB2、PostgreSQL、Microsoft SQL Server、Microsoft Access、MySQL
  - 关系型数据库是依据关系模型来创建的数据库。
  - 所谓关系模型就是“一对一、一对多、多对多”等关系模型，关系模型就是指二维表格模型,因而一个关系型数据库就是由二维表及其之间的联系组成的一个数据组织。
  - 关系型数据可以很好地存储一些关系模型的数据，比如一个老师对应多个学生的数据（“多对多”），一本书对应多个作者（“一对多”），一本书对应一个出版日期（“一对一”）
  - 关系模型是我们生活中能经常遇见的模型，存储这类数据一般用关系型数据库
  - 关系模型包括数据结构（数据存储的问题，二维表）、操作指令集合（SQL语句）、完整性约束(表内数据约束、表与表之间的约束)。
- 非关系型数据库
  - 常见类型
    - mongoDB、Hbase、redis、MemcacheDB
  - 非关系型数据库主要是基于“非关系模型”的数据库（由于关系型太大，所以一般用“非关系型”来表示其他类型的数据库）
  - 列模型：存储的数据是一列列的。关系型数据库以一行作为一个记录，列模型数据库以一列为一个记录。（这种模型，数据即索引，IO很快，主要是一些分布式数据库）
  - 键值对模型：存储的数据是一个个“键值对”，比如name:liming,那么name这个键里面存的值就是liming
  - 文档类模型：以一个个文档来存储数据，有点类似“键值对”

基础语句

```
// 登录
mysql -u root -p;
// 查询数据库
show databases;
// 查询当前库的表
show tables;
// 查询当前表的列
show COLUMNS from people;
// 查询具体的值
select name from people;
select * from people;
// 去重
```

```
    select distinct name from people;
    // 限制多少条
    select name from people limit 1;
    // 偏移多少条
    select name from people limit 1 offset 1;
    // 排序
    select name from people order by id desc;
    // 条件查询
    select name from people where id = 4;
    // 模糊查询
    select name from people WHERE name like 'T%';
    select name from people WHERE id BETWEEN 2 and 3;
    select name from people WHERE id = 1 or id = 2;
    select name from people WHERE id = 1 and name like 'T%';
    select name from people WHERE id in (2, 4);
    select name from people WHERE name like 'T%' and not id in (1, 3);
    // 正则匹配
    select name from people WHERE name regexp 'T';
    select name from people WHERE name regexp 'hello[0-9]';
    // 函数
    select CONCAT(name,'(', id, ')') as name from people;
    select id + id as id from people;
    select name from people where year(birth) = '2019' and month(birth) = '5';
    select people_id, count(*) as bookcount from book group by people_id
having bookcount >= 2;
    // 子查询
    select name
    from people
    where id in (
      select people_id
      from book
      where name like 'book%'
    )
    //
    select p.name, b.name from people as p, book as b where p.id =
b.people_id;
    select count(*) from (select p.name as pname, b.name from people as p,
book as b where p.id = b.people_id) as t;
    // 左右连接
    select * from people as p LEFT OUTER JOIN book as b on p.id = b.people_id;
    select * from people as p right OUTER JOIN book as b on p.id =
b.people_id;
    // 并集
    select name from book where people_id = 1 union select name from book
where people_id = 2
    // 插入数据
    insert into people values(null, 'hh', 13)
    insert into people(name, age) values('haha', 18)
    insert into people(name,age) values('haha', 18),('h1h1', 19)
```

```
    // 更新数据
    update people set name = 'heihei', age = 2
    update people set name = 'hei', age = 20 where id = 1
    // 删除数据
    delete from people where id = 1
    delete from people
    // 创建表
    create table people (
      id int(10) not null auto_increment,
      name varchar(255) null,
      age int(10) null,
      primary key (id)
    )
    // 修改表
    alter table people add sex varchar(20) null;
    alter table people drop sex;
    alter table book drop foreign key book_people_id;
    alter table book add constraint book_people_id foreign key (people_id)
references people (id);
    // 删除表
    drop table book1;
    // 重命名
    rename table book to book1;
    // 创建视图
    create or replace view test1 as select `p`.`name` AS `pname`,`b`.`name` AS
`bname` from (`people` `p` join `book` `b`) where (`p`.`id` =
`b`.`people_id`);
    // 删除视图
    drop view test1;
    // 创建存储过程
    create procedure test (
      in tid int(10),
      out aage int(10)
    )
    begin
      select avg(age) into aage
      from people
      where id > tid;
    end
    call test(3, @avg);
    select @avg;
    // 创建触发器
    create trigger test1 after insert on people for each row select 'hhaha'
into @ee;
    insert into people (name, age) values ('aaa', 2);
    select @ee;
    drop trigger test1;
    // 事务操作
    select * from people;
```

```
start transaction;
delete from people where id = 1;
rollback;
--commit;

start transaction;
delete from people where id = 2;
savepoint flag1;
delete from people where id = 32;
savepoint flag2;

rollback to flag1;

select * from people;

select user from user;
// 用户操作
create user test identified by '_123456Root';
rename user test to test1;
// 授权
grant select on centos.* to test1;
show grants for test1;
drop user test1;
```

mysql 常用函数

- 字符串

```
contact(str1, str2)
trim(str)
rtrim(str)
ltrim(str)
upper(str)
left(str)
length(str)
lower(str)
locate(str)
substring(str)
// 匹配 发音  类似的字符串
soundex(str)
```

- 日期

```
adddate()
addtime()
curdate()
curtime()
```

```
    date()
    datediff()
    date_add()
    date_format()
    day()
    dayofweek()
    hour()
    minute()
    month()
    now()
    second()
    time()
    year()
```

- 数值处理

```
    abs()
    cos()
    exp()
    mod()
    pi()
    rand()
    sin()
    sqrt()
    tan()
```

- 聚集函数

```
    avg()
    count()
    max()
    min()
    sum()
```

mysql数据引擎:

```
* InnoDB
    * 事务引擎，不支持全文搜索
* MEMORY
    * 功能跟MyISAM相同，数据存在内存，速度快，适合临时表
* MyISAM
    * 性能极高，支持全文搜索，不支持事务
```

注意事项

- 引擎能混用，外键不能跨引擎
- mysql 游标只能应用于 存储过程，而且变量的声明必须在游标声明的前面。游标简单点来说代表

一个结果集

- mysql 日志有很多种，主要查看 /var/log/mysqld-err.log 和 mysqlbinlog /var/lib/binlog.000007

参考资料

- mysql必知必会
- 关系数据库和非关系数据库 *