MySQL，主讲：汤小洋

一、MySQL简介

1. 介绍

1.1 什么是数据库?

数据库：Database，按照数据结构来组织、存储和管理数据的仓库，简单来说就是存储数据的仓库

数据库管理系统：用来管理数据库的软件系统，常见：MySQL、Oracle、SQL Server、DB2、Sybase、Access等

1.2 什么是MySQL?

MySQL：是一个开源的关系型数据库管理系统，由瑞典MySQL AB公司开发，后来被Oracle收购，所以目前属于Oracle公司

特点：体积小、速度快、成本低、开源，中小型网站都使用MySQL数据库

版本：企业版Enterprise、社区版Community

DBA：Database Administrator 数据库管理员

2. 安装MySQL

2.1 版本

分平台：Windows、Linux、Mac

分版本：5.x 6.x 7.x 8.x

2.2 安装

安装位置：/usr/local/mysql

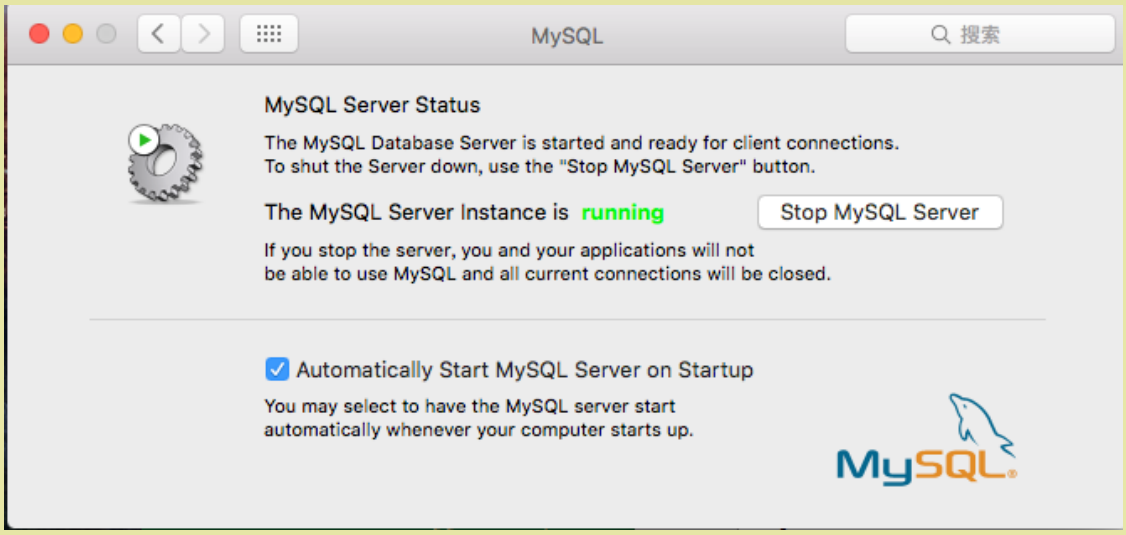
- bin 可执行文件
- data 数据库文件
- my.cnf 核心配置文件

2.3 服务

安装MySQL后，会在操作系统中添加一个MySQL服务

需要先启动服务才能使用MySQL：

- 系统偏好设置——>MySQL
- 勾选Automatically Start MySQL Server on Startup，设置开机自动启动MySQL服务



二、基本操作

1. 连接MySQL

语法：

```
mysql -u 用户名 -p密码 -h 数据库服务器地址 -D 数据库名
```

安装MySQL以后，默认有一个管理员 root

2. 查看数据库和表

```
show databases;  -- 查看当前所有数据库
use 数据库名; -- 切换数据库
show tables; -- 查看当前数据库中所有表
select database();  -- 显示当前操作的数据库
select user();  -- 显示当前登陆的用户
```

mysql库是系统库，包含MySQL的相关系统信息，不要修改

3. 导入初始数据

3.1 导入数据

以.sql结尾的文件是数据库脚本文件

先连接登陆MySQL，然后执行如下命令：

```
source /Users/appleuser/Desktop/init.sql
```

3.2 表结构

```
desc 表名;    -- 查看表的结构
select * from 表名; -- 查看表中的所有数据
```

emp雇员表

列名	类型	含义
empno	int 整数	雇员编号
ename	varchar 字符串	雇员姓名
job	varchar 字符串	工作、职位
mgr	int 整数	上司/领导的编号
hiredate	date 日期	入职时间
sal	double 小数	薪水、工资
comm	int 整数	奖金
deptno	int 整数	部门编号

dept部门表

列名	类型	含义
deptno	int 整数	部门编号
dname	varchar 字符串	部门名称
loc	varcher 字符串	部门位置

salgrade工资等级表

列名	类型	含义
grade	int 整数	等级编号
losal	int 整数	工资下限
hisal	int 整数	工资上限

bonus奖金表

三、SQL

SQL：Structured Query Language 结构化查询语言，用来对数据库进行查询、更新和管理的一种特殊的语言

包含三个部分：

- DML

  Data Manipulation Language 数据操作语言

  用于检索或更新数据库：insert delete update select 增删改查

- DDL

  Data Definition Language 数据定义语言

用于定义数据的结构：create alter drop

- DCL

  Data Control Language 数据控制语言

  用于定义数据库用户的权限：grant revoke

四、 查询操作

1. 简介

1.1 语法

```
select 列名1 别名1,列名2 别名2... from 表名;
```

示例：

```
select ename from emp;
select ename,job,hiredate from emp;
select * from emp;
select ename 姓名,job 职位,hiredate 入职时间 from emp;
select empno,ename,sal "your salary" from emp;  -- 别名包含空格时，需要使用双引号引起来
```

1.2 用法

- 字符串连接concat()

  ```
  select concat('编号为',empno,'的雇员，姓名为',ename,'，职位为',job)
  from emp;
  ```

- 四则运算 + - * %

  例：查询雇员姓名及年薪

  ```
  select ename 雇员姓名,sal*12 年薪 from emp;
  select ename 雇员姓名,(sal+comm)*12 年薪 from emp;  -- 有问题的
  select ename 雇员姓名,(sal+ifnull(comm,0))*12 年薪 from emp;  -- 使用ifnull()
  ```

  在MySQL中，NULL与任何值进行运算，结果都为NULL

- 去除重复列 distinct

  例：查询所有的职位

  ```
  select distinct job from emp;
  select ename,job from emp; -- 在去除重复列时只有当所有列都相同时才能去除
  ```

2. 限定查询

语法：

```
select 列名1 别名1,列名2 别名2...
from 表名
where 条件;
```

2.1 比较运算符

`> < >= <= = !=或<>`

例：查询工资大于1500的雇员信息

```
select * from emp where sal>1500;
```

例：查询雇员编号不是7369的雇员信息

```
select * from emp where empno!=7369;
```

例：查询姓名是smith的雇员编号、姓名、工资、入职时间

```
select empno,ename,sal,hiredate from emp where ename='smith';
```

注：字符串要使用引号引起来，同时MySQL中不区分大小写

2.2 null 或 not null

例：查询没有奖金的雇员信息

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

注：判断是否为null时使用的是is，不能使用比较运算符=

2.3 and

例：查询基本工资大于1000，并且可以获取奖金的雇员姓名、工资、奖金

```
select ename,sal,comm from emp where sal>1000 and comm is not null;
```

2.4 or

例：查询从事销售工作，或工资大于等于2000的雇员信息

```
select * from emp where job='salesman' or sal>=2000;
```

2.5 not

例：查询从事非销售工作，并且工资不小于1500的雇员的编号、姓名、职位、工资、入职时间

```
select empno,ename,job,sal,hiredate from emp where job!='salesman' and sal
>= 1500;
select empno,ename,job,sal,hiredate from emp where not (job='salesman' and
sal<1500);  -- 有问题的
select empno,ename,job,sal,hiredate from emp where (not job='salesman')
and (not sal<1500);
```

2.6 between and

例：查询基本工资大于1500，但小于3000的雇员信息

```
select * from emp where sal>1500 and sal<3000;
select * from emp where sal between 1500 and 3000;
```

注：between and 包含临界值

例：查询1981年入职的雇员编号、姓名、入职时间、所在部门编号

```
select empno,ename,hiredate,deptno from emp where hiredate between '1981-
1-1' and '1981-12-31';
```

注：日期必须使用引号引起来

2.7 in 或 not in

例：查询编号为7369、7499、7788的雇员信息

```
select * from emp where empno=7369 or empno=7499 or empno=7788;
select * from emp where empno in (7369,7499,7788);
```

例：查询姓名为smith、allen、king的雇员编号、姓名、入职时间

```
select empno,ename,hiredate from emp where ename in
('smith','allen','king');
select empno,ename,hiredate from emp where ename not in
('smith','allen','king');
```

2.8 like

用来进行模糊查询，需要结合通配符一起使用

常用通配符：

- % 可以匹配任意长度字符
- _ 只能匹配单个字符

例：查询雇员姓名以S开头的雇员信息

```
select * from emp where ename like 'S%';
```

例：查询雇员姓名中包含M的雇员信息

```
select * from emp where ename like '%M%';
```

例：查询从事销售工作，并且姓名长度为4个字符的雇员信息

```
select * from emp where job='salesman' and ename like '____';
```

例：查询1981年入职的雇员编号、姓名、入职时间、所在部门编号

```
select empno,ename,hiredate,deptno from emp where hiredate like '1981%';
```

3. 排序

3.1 语法

语法：

```
select 列名1 别名1,列名2 别名2...
from 表名
where 条件
order by 排序列1 asc|desc,排序列2 asc|desc...;
```

asc表示升序，desc表示降序，省略时默认按升序

3.2 示例

例：查询所有雇员信息，按工资由低到高排序

```
select * from emp order by sal;
```

例：查询部门10的雇信息，按工资由高到低排序，如果工资相同，则按入职时间由早到晚排序

```
select * from emp where deptno=10 order by sal desc,hiredate;
```

例：查询雇员编号、姓名、年薪，按年薪由高到低排序

```
select empno,ename,(sal+ifnull(comm,0))*12 income from emp order by income
desc;
```

五、多表查询

1. 简介

同时从多张表中查询数据，一般来说多张表之间都会存在某种关系

2. 基本用法

2.1 语法

```
select 列名1 别名1,列名2 别名2...
from 表名1 别名1,表名2 别名2...
where 条件
order by 排序列1 asc|desc,排序列2 asc|desc...;
```

例：将emp表和dept表进行多表查询（笛卡尔积）

```
select * from emp,dept;
```

通过将两张表的关联字段进行比较，去掉笛卡尔积，多表查询时一般都会存在某种关系

```
select * from emp,dept where emp.deptno=dept.deptno;
```

2.2 示例

例：查询雇员编号、雇员姓名、工资、所在部门名称及位置（等值连接）

```
select empno,ename,sal,dname,loc
from emp e,dept d
where e.deptno=d.deptno;
```

例：查询雇员姓名、工资、入职时间、所在部门编号、部门名称

```
select e.ename,e.sal,e.hiredate,d.deptno,d.dname
from emp e,dept d
where e.deptno=d.deptno;  -- 如果多张表中出现同名的列，当查询时需要指定前缀
```

例：查询雇员姓名、雇员工资、领导姓名、领导工资（自身连接）

```
select e.ename 雇员姓名,e.sal 雇员工资,m.ename 领导姓名,m.sal 领导工资
from emp e,emp m
where e.mgr=m.empno;
```

例：查询雇员姓名、雇员工资、部门名称、领导姓名、领导工资

```
select e.ename 雇员姓名,e.sal 雇员工资,d.dname 部门名称,m.ename 领导姓名,m.sal
领导工资
from emp e,dept d,emp m
where e.deptno=d.deptno and e.mgr=m.empno;
```

例：查询雇员姓名、雇员工资、部门名称、工资所在等级（非等值连接）

```
select e.ename 雇员姓名,e.sal 雇员工资,d.dname 部门名称,s.grade 工资等级
from emp e,dept d,salgrade s
where e.deptno=d.deptno and e.sal between s.losal and s.hisal;
```

例：查询雇员姓名、雇员工资、部门名称、雇员工资等级、领导姓名、领导工资、领导工资等级

```
select e.ename 雇员姓名,e.sal 雇员工资,d.dname 部门名称,s.grade 雇员工资等
级,m.ename 领导姓名,m.sal 领导工资,sm.grade 领导工资等级
from emp e,dept d,salgrade s,emp m,salgrade sm
where e.deptno=d.deptno and e.sal between s.losal and s.hisal and
e.mgr=m.empno and m.sal between sm.losal and sm.hisal;
```

3. SQL99标准

3.1 简介

SQL99标准，也称为SQL1999标准，是1999年制定的

分类：内连接、外连接

3.2 内连接

使用inner join…on

语法：

```
select 列名1 别名1,列名2 别名2...
from 表名1 别名1 inner join 表名2 别名2 on 多表间的关联关系
where 条件
order by 排序列1 asc|desc,排序列2 asc|desc...;
```

例：查询雇员编号、雇员姓名、工资、部门名称

```
select e.empno,e.ename,e.sal,d.dname
from emp e inner join dept d on e.deptno=d.deptno;
```

例：查询工资大于1500的雇员姓名、工资、部门名称、领导姓名

```
select e.ename,e.sal,d.dname,m.ename
from emp e inner join dept d on e.deptno=d.deptno inner join emp m on
e.mgr=m.empno
where e.sal>1500;
```

```
select e.ename,e.sal,d.dname,m.ename
from emp e,dept d,emp m
where e.deptno=d.deptno and e.mgr=m.empno and e.sal>1500;
```

3.3 外连接

分类：

- 左外连接 left outer join…on，也称为左连接left join…on

  以左边的表作为主表，无论如何都会显示主表中的所有数据
- 右外连接 right outer join…on，也称为右连接right join…on

  以右边的表作为主表，无论如何都会显示主表中的所有数据

语法：

```
select 列名1 别名1,列名2 别名2...
from 表名1 别名1 left join 表名2 别名2 on 多表间的关联关系
where 条件
order by 排序列1 asc|desc,排序列2 asc|desc...;
```

例：查询雇员姓名、工资、领导姓名、领导工资（有的雇员没有领导）

```
select e.ename,e.sal,m.ename,m.sal
from emp e,emp m
where e.mgr=m.empno;  -- 有问题的
```

```
select e.ename,e.sal,m.ename,m.sal
from emp e inner join emp m on e.mgr=m.empno;  -- 有问题的
```

```
select e.ename,e.sal,m.ename,m.sal
from emp e left join emp m on e.mgr=m.empno;
```

```
select e.ename,e.sal,m.ename,m.sal
from emp m right join emp e on e.mgr=m.empno;
```

例：查询部门编号、部门名称、部门位置、部门中雇员姓名、工资

```
select d.deptno,d.dname,d.loc,e.ename,e.sal
from dept d left join emp e on d.deptno=e.deptno
order by d.deptno;
```

六、聚合函数和分组统计

1. 聚合函数

聚合函数，称为统计函数

常用函数函数：

- count() 总数量
- max() 最大值
- min() 最小值
- sum() 和
- avg() 平均值

例：查询部门30的总人数

```
select count(empno) 总人数 from emp where deptno=30;
select max(sal) from emp;
select ename 雇员姓名,avg(sal) 平均工资 from emp where deptno=10;  -- 不合理
```

注：聚合函数在统计时会忽略NULL值

例：查询部门30的最高工资、最低工资、平均工资

```
select max(sal),min(sal),round(avg(sal),2) from emp where deptno=30;
```

2. 分组统计

2.1 语法

```
select 列名1 别名1,列名2 别名2...
from 表名1 别名1 left join 表名2 别名2 on 多表间的关联关系
where 分组前的条件
group by 分组列
having 分组后的条件
order by 排序列1 asc|desc,排序列2 asc|desc...;
```

2.2 示例

例：查询每个部门的平均工资

```
select deptno 部门编号,avg(sal) 平均工资
from emp
group by deptno;
```

```
select d.dname 部门名称,avg(sal) 平均工资
from emp e,dept d
where e.deptno=d.deptno
group by d.dname;
```

注：

- 在MySQL中分组统计时可以查询出分组列以外的其他列，而在Oracle中不行
- 建议将要查询出的列作为分组列

例：查询部门的名称及每个部门的员工数量

```
select d.dname 部门名称,count(e.empno) 员工数量
from dept d left join emp e on d.deptno=e.deptno
group by d.dname;
```

例：查询平均工资大于2000的部门的编号和平均工资

```
select deptno,avg(sal)
from emp
group by deptno
having avg(sal)>2000;
```

例：查询出非销售人员的职位名称，以及从事同一工作的雇员的月工资总和，并且要满足工资总和大于5000，查询的结果按月工资总和的升序排列

```
select job,sum(sal) sum
from emp
where job!='salesman'
group by job
having sum(sal)>5000
order by sum;
```

例：查询部门平均工资的最大值

```
select max(avg(sal))
from emp
group by deptno;  -- MySQL中不支持
```

注：在MySQL中聚合函数不能嵌套使用，而Oracle中可以

```
select max(temp.avg)
from (select avg(sal) avg from emp group by deptno) temp;
```

七、子查询

1. 简介

一个查询中嵌套着另一个查询，称为子查询

- 子查询必须放在小括号中
- 子查询可以出现在任意位置，如select、from、where、having等

2. 基本用法

2.1 语法

```
select (子查询)
from (子查询) 别名
where (子查询)
group by
having (子查询)
```

2.2 示例

例：查询工资比7566高雇员信息

```
-- 使用多表连接
select e2.*,e1.ename,e1.sal
from emp e1,emp e2
where e1.empno=7566 and e2.sal>e1.sal;
```

```
-- 使用子查询
select sal from emp where empno=7566;
select * from emp where sal> (select sal from emp where empno=7566);
```

例：查询工资比部门30员工的工资高的雇员信息

```
select sal from emp where deptno=30;
select * from emp where sal>(select sal from emp where deptno=30); -- 错误用户
```

注：将子查询与比较运算符一起使用时，必须保证子查询返回的结果不能多于一个

例：查询雇员的编号、姓名、部门名称

```sql
-- 使用多表连接
select e.empno,e.ename,d.dname from emp e,dept d where e.deptno=d.deptno;
```

```sql
-- 子查询
select empno,ename,(select dname from dept where deptno=e.deptno) from emp
e;
```

总结：

- 一般来说，多表连接查询都可以使用子查询替换，但有的子查询不能使用多表连接查询来替换
- 子查询特点：灵活、方便，一般常作为增、删、改、查操作的条件，适合于操作一个表的数据
- 多表连接查询更适合于查看多表中数据

3. 子查询分类

可以分为三类：

- 单列子查询

  返回单行单列，使用频率最高

- 多行子查询

  返回多行单列

- 多列子查询

  返回单行多列或多行多列

3.1 单列子查询

例：查询工资比7654高，同时又与7900从事相同工作的雇员信息

```sql
select *
from emp
where sal>(
    select sal from emp where empno=7654
) and job=(
    select job from emp where empno=7900
);
```

例：查询工资最低的雇员的姓名、工作、工资

```sql
select ename,job,sal from emp where sal=(select min(sal) from emp);
```

例：查询工资高于公司平均工资的雇员信息

```sql
select * from emp where sal>(select avg(sal) from emp);
```

例：查询每个部门的编号和最低工资，要求最低工资大于等于部门30的最低工资

```
select deptno,min(sal)
from emp
group by deptno
having min(sal)>=(
    select min(sal) from emp where deptno=30
);
```

例：查询部门的名称、部门的员工数、部门的平均工资、部门的最低收入雇员的姓名

```
-- 拆分
select deptno,count(empno),avg(sal),min(sal)
from emp
group by deptno;
```

```
-- 方式1：使用子查询
select
    (select dname from dept where deptno=e.deptno) dname,
    count(empno),
    avg(sal),
    (select ename from emp where sal=min(e.sal)) ename
from emp e
group by deptno;
```

```
-- 方式2：使用多表连接查询
select d.dname,t.count,t.avg,e.ename
from (select deptno,count(empno) count,avg(sal) avg,min(sal) min from emp
group by deptno) t,dept d,emp e
where d.deptno=t.deptno and e.sal=t.min;
```

例：查询平均工资最低的工作及平均工资

```
-- 拆分
select min(t.avg)
from  (
  select avg(sal) avg from emp group by job
) t;


select job,avg(sal)
from emp
group by job
having avg(sal) = (
    select min(t.avg) from  (select avg(sal) avg from emp group by job) t
);
```

3.2 多行子查询

对于多行子查询，可以使用如下三种操作符：

- in

  例：查询所在部门编号大于等于20的雇员信息

  ```
  select * from emp where deptno>=20;
  select * from emp where deptno in (
      select deptno from dept where deptno>=20
  );
  ```

  例：查询工资与部门20中的任意员工相同的雇员信息se

  ```
  select * from emp where sal in (
      select sal from emp where deptno=20
  );
  ```

- any

  三种用法：

  ```
  =any：与任意一个相同，此时与in操作符功能一样
  >any：只要比里面最小的值大即可
  <any：只要比里面最大的值小即可
  ```

  ```
  select * from emp where sal <any (
      select sal from emp where deptno=20
  );
  ```

- all

  两种用法：

```
>all：比里面最大的值要大
<all：比里面最小的值要小
```

```
select * from emp where sal <all (
    select sal from emp where deptno=20
);
```

3.3 多列子查询

多列子查询一般出现在from子句中，作为查询结果集

例：在所在从事销售工作的雇员中找出工资大于1500的员工

```
select *
from (
    select * from emp where job='salesman'
) t
where t.sal>1500;
```

八、分页查询

1. limit关键字

用来限制查询返回的记录数

语法：

```
select 列名1 别名1,列名2 别名2...
from 表名1 别名1 left join 表名2 别名2 on 多表间的关联关系
where 分组前的条件
group by 分组列
having 分组后的条件
order by 排序列1 asc|desc,排序列2 asc|desc...
limit [参数1,]参数2
```

可以接收一个或两个数字：

- 参数1用来指定起始行的索引，索引从0开始，即第一行的索引为0
- 参数2用来指定返回的记录数量

例：查询工资的前3名

```
select * from emp order by sal desc limit 0,3;
select * from emp order by sal desc limit 3; -- 如果省略参数1，则默认为0，即从第1条开始返回
```

例：查询工资大于1000的第4-8个用户

```
select * from emp where sal>1000 limit 3,5;
```

例：查询工资最低的用户

```
select * from emp order by sal limit 1;
```

2. 分页

例：每页显示4条（pageSize每页大小），显示第3页的内容（pageIndex页码）

```
select * from emp limit (pageIndex-1)*pageSize,pageSize  -- 计算
select * from emp limit (3-1)*4,4 -- 不能直接执行
```

注：在MySQL中limit后面的参数不能包含任何运算，实际开发中都是在编程语言中进行计算，然后将结果发送给数据库执行

九、常用函数

1. 字符串函数

- concat(s1,s2,s3….) 拼接字符串

  ```
  select concat('aa','bb','cc')
  select concat('aa','bb','cc') from dual;
  select concat('编号为',empno,'的员工，姓名为',ename) from emp;
  ```

  注：dual表是MySQL提供的一张虚拟表，主要是为了满足select...from...语法习惯，一般测试时使用，无实际意义
- lower(s) 将字符串变为小写 `select lower('Hello') from dual`
- upper(s) 将字符中变为大写 `select upper('Hello') from dual`
- length(s) 获取字符串的长度 `select length('hello') from dual`
- reverse(s) 将字符串反转 `select reverse('hello') from dual`
- trim(s) 去除字符串两边的空格 `select trim('  hello  ') from dual`，还有ltrim()和rtrim()，去除左边或右边的空格
- replace(s,s1,s2) 将字符串s中的s1替换为s2 `select replace('hello world','o','xx') from dual`
- repeat(s,n) 将字符串s重复n次后返回 `select repeat('hello',3) from dual`
- lpad(s,len,s1) 在字符串s的左边使用s1进行填充，直至长度为len `select lpad('hello',8,'x') from dual`
- rpad(s,len,s1) 在字符串s的右边使用s1进行填充，直至长度为len `select rpad('hello',8,'x') from dual`

- substr(s,i,len) 从第i个位置开始对字符串s进行截取，截取len个 `select substr('hello',2,3) from dual`

2. 数值函数

- ceil(n) 返回大于n的最小整数 `select ceil(10.1) from dual`
- floor(n) 返回小于n的最大整数 `select floor(10.1) from dual`
- round(n,y) 将n保留y位小数，四舍五入 `select round(3.1415,3) from dual`
- truncate(n,y) 将n保留y位小数，不四舍五入 `select truncate(3.1415,3) from dual`
- rand() 返回0到1的随机数 `select rand() from dual`

3. 日期和时间函数

- now() 返回当前日期时间 `select now() from dual`
- curdate() 返回当前日期 `select curdate() from dual`
- curtime() 返回当前时间 `select curtime from dual`
- year(date) 返回日期中的年 `select year('2018-2-14') from dual`
- month(date) 返回日期中的月 `select month('2018-2-14') from dual`
- day(date) 返回日期中的日 `select day('2018-2-14') from dual`
- timestampdiff(interval,datetime1,datetime2) 返回两个日期时间之间相隔的整数，单位由interval定义

  interval可取值：year、month、day、hour、minute、second

  `select timestampdiff(day,'1993-9-23','2018-11-22') from dual`
- date_format(date,pattern) 格式化日期 `select date_format(now(),'%Y年%m月%d日 %H:%i:%s') from dual`

  格式化参数：

  %Y 表示四位数字的年

  %m 表示两位数字的月

  %d 表示两位数字的日

  %H 表示两位数字的小时，24小时制

  %i 表示两位数字的分钟

  %s 表示两位数字的秒数

4. 流程控制函数

- if(f,v1,v2) 如果f为真，则返回v1，否则返回 v2 `select if(5>2,'yes','no') from dual`
- ifnull(v1,v2) 如果v1不为null，则返回v1，否则返回v2 `select ifnull(null,'0') from dual`
- case when f1 then v1 when f2 then v2….else v end 如果f1为真，则返回v1；如果f2为真，则返回v2...否则返回v

```
select case when 5>2 then 'yes' end from dual;
select case when 5<2 then 'yes' else 'no' end from dual;
select case when 5<2 then 'one' when 6>4 then 'two' else 'three'
end from dual;
```

5. 系统信息函数

- database() 返回当前操作的数据库 `select database()`
- user() 返回当前登陆的用户 `select user()`
- version() 返回MySQL服务器的版本 `select version()`

十、更新操作

1. insert

语法：

```
-- 语法1
insert into 表名 (列名1,列名2...) values (值1,值2...);
-- 语法2：一次性插入多条数据
insert into 表名 (列名1,列名2...) values (值1,值2...),(值1,值2...),(值1,值
2...)
```

示例：

```
insert into dept (deptno,dname,loc) values (50,'市场部','南京');
insert into dept (deptno,dname) values (60,'开发部');
insert into dept values (70,'保洁部','上海'); -- 如果是依次插入表中所有的列，此时
可以省略列名
insert into dept values (11,'aaa','aaa'),(12,'bbb','bbb'),
(13,'ccc','ccc');
insert into emp (empno,ename,job,hiredate,sal,deptno) values (9527,'唐伯
虎','画家',now(),6666,50);
```

2. delete

语法：

```
delete from 表名 where 条件;
```

示例：

```
delete from dept where deptno=60;
delete from dept where dname='保洁部';
-- 删除市场部所有工资高于5000的员工
select deptno from dept where dname='市场部'

delete from emp where deptno=(select deptno from dept where dname='市场部')
and sal>5000;
```

注：`delete from emp` 会将表中所有数据都删除

3. update

语法：

```
update 表名 set 列名1=值1,列名2=值2... where 条件
```

示例：

```
update dept set dname='market' where dname='市场部';
update emp set job='manager',sal=8888,comm=666 where ename='smith';
```

十一、表和库的管理

1. 数据类型

整数：smallint、int、bigint

小数：float、double

日期时间：date、time、datetime、timestamp

字符串：varchar、char、text

其他：clob 存储文本大数据

blob 存储二进制大数据

2. 创建表

语法：

```
create table 表名
(
    列名 数据类型 特征,
    列名 数据类型 特征,
    ....
    列名 数据类型 特征
) charset=utf8;
```

示例：

```
create table user
(
    id int,
    username varchar(20),
    password varchar(50)
);
```

```
create table t_student
(
    id int primary key auto_increment,  -- 将id作为主键，自动增长，默认从1开始，每次递增1
    name varchar(10) not null, -- 不允许为空
    age int,
    sex varchar(8) not null default '女', -- 指定默认值
    address varchar(100),
    height double,
    birthday date
)charset=utf8;
```

```
insert into t_student (name,age,sex,birthday,height) values ('范婷婷',18,'女','1998-12-4',170.6);
insert into t_student (name,age,sex) values (null,10,'男');
insert into t_student values (null,'程瑞',19,'男','南京',176.6,now());
```

3. 修改表

- 添加列

  语法：

  ```
  alter table 表名 add 列名 数据类型
  ```

  示例：

  ```
  alter table t_student add weight double;
  ```

- 修改列类型

  语法：

  ```
  alter table 表名 modify 列名 数据类型
  ```

  示例：

```
alter table t_student modify name varchar(250);
```

- 修改列名

```
alter table 表名 change 原列名 新列名 数据类型
```

示例：

```
alter table t_student change sex gender varchar(8);
```

- 删除列

语法：

```
alter table 表名 drop 列名
```

示例：

```
alter table t_student drop weight;
```

- 修改表名

语法：

```
alter table 原表名 rename 新表名
或
rename table 原表名 to 新表名
```

示例：

```
alter table t_student rename student;
rename table student to t_student;
```

4. 删除表

语法：

```
drop table 表名;
drop table if exists 表名;
```

示例：

```
drop table user;
drop table if exists user;
```

5. 截断表

清空表中的数据，作用类似于没有条件的delete语句

语法：

```
truncate table 表名;
```

delete与truncate区别：

- delete会记录日志，所以速度慢，而truncate不记录日志，清空表并释放资源，速度快
- delete可以指定条件只删除表中的部分数据，而truncate只能用来清空表中的所有数据
- delete不会将自动增长列归零，而truncate会

6. 创建库

语法：

```
create database 数据库名 charset utf8;
create database if not exists 数据库名 charset utf8;
```

示例：

```
create database if not exists shop charset utf8;
```

7. 删除库

语法：

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

示例：

```
drop database if exists shop;
```

十二、约束

1. 简介

constraint约束是对表中的数据的一种限制，保证数据的完整性和有效性

2. 约束分类

有五种约束：

- 主键约束 primary key

  用来唯一的标识一条记录（数据），本身不能为空

- 唯一约束 unique

不允许出现重复值

- 检查约束 check

  判断数据是否符合指定条件

  注：MySQL会对check约束进行分析，但会忽略check约束，即不会强制执行此约束，可以通过SQL编程来解决

- 非空约束 not null

  不允许为null，但可以为空字符串''

- 外键约束 foreign key

  约束两表之间的关联关系

3. 添加约束

- 方式1：在创建表时添加约束

```
-- 约束没有名称
create table student  -- 学生表
(
    id int primary key,  -- 主键约束
    name varchar(10) not null,  -- 非空约束
    age int check(age between 1 and 120), -- 检查约束
    sex varchar(8) not null check(sex in ('male','female')), -- 多种约束
    IDCard varchar(18) unique, -- 唯一约束
    class_id int, -- 外键列
    foreign key (class_id) references class(c_id) -- 外键约束，引用主表中的主键
);
```

```
create table class  -- 班级表
(
    c_id int primary key,
    c_name varchar(20) not null,
    c_info varchar(200)
);
```

查看表的所有信息（约束）：

```
show create table 表名;
```

```
-- 为约束指定名称
create table student
(
    id int,
    name varchar(10) not null,  -- 非空约束
    age int,
    sex varchar(8) ,
    IDCard varchar(18),
    class_id int,
    constraint pk_id primary key (id),
    constraint ck_age check(age between 1 and 120),
    constraint ck_sex check(sex in ('male','female')),
    constraint uq_IDCard unique (IDCard),
    constraint fk_class_id foreign key (class_id) references
class(c_id)
);
```

- 方式2：在创建表之后再添加约束

```
create table student
(
    id int,
    name varchar(10) not null,  -- 非空约束只能在创建表时在列名后面指定
    age int,
    sex varchar(8) ,
    IDCard varchar(18),
    class_id int
);
```

为表添加约束，语法：

```
alter table 表名 add constraint 约束名 约束类型 约束内容
```

示例：

```
alter table student add constraint pk_id primary key (id);
alter table student add constraint ck_age check(age between 1 and
120);
alter table student add constraint ck_sex check(sex in
('male','female'));
alter table student add constraint uq_IDCard unique (IDCard);
alter table student add constraint fk_class_id foreign key
(class_id) references class(c_id);
```

4. 删除约束

语法：

- 删除主键约束 `alter table 表名 drop primary key`
- 删除外键约束 `alter table 表名 drop foreign key 约束名称`
- 删除唯一约束 `alter table 表名 drop index 约束名称`
- 删除非空约束 `alter table 表名 modify 列名 数据类型 null`

5. 注意事项

- 创建表时，必须先创建主表，再创建从表
- 删除表时，必须先删除从表，再删除主表
- 可以在创建表时指定级联删除，当主表数据被删除时，将自动删除从表中的相关数据

```
create table student  -- 学生表
(
    id int primary key,  -- 主键约束
    name varchar(10) not null,  -- 非空约束
    age int check(age between 1 and 120), -- 检查约束
    sex varchar(8) not null check(sex in ('male','female')), -- 多种约束
    IDCard varchar(18) unique, -- 唯一约束
    class_id int, -- 外键列
    foreign key (class_id) references class(c_id) on delete cascade  -- 级联删除
);
```

十三、用户和权限管理

1. 创建用户并授予权限

语法：

```
grant 权限列表 on 库名.表名 to 用户名@来源地址 identified by '密码';
```

示例：

```
grant select on test.emp to tom@localhost identified by '123';
grant select on shop.user to jack@localhost identified by '123';
grant select,update on shop.* to mike@'%' identified by '111';
grant delete on shop.user to mike@'%';
grant all on *.* to alice@'%' identified by '123';
```

注：test库是安装时默认创建的，默认情况下所有用户对该库都拥有最大的权限

只能管理员才具有创建用户的权限

2. 查看权限

语法：

```
show grants;  -- 查看自己的权限
show grants for 用户名@来源地址;  -- 查看其他人的权限
```

3. 撤销权限

语法：

```
revoke 权限列表 on 库名.表名 from 用户名@来源地址
```

示例：

```
revoke delete on shop.user from mike@'%';
```

4. 删除用户

语法：

```
use mysql;
delete from user where user='用户名';
flush privileges; -- 刷新权限
```

十四、事务处理

1. 简介

transaction

事务处理是用来保证数据操作的完整性

一个业务由若干个一次性的操作组成，这些操作要么都成功，要么都失败，如银行转账

事务特性ACID：

- 原子性（Atomicity）：不可再分
- 一致性（Consistency）：要保证数据前后的一致性
- 隔离性（Isolation）：两个事务的操作互不干扰
- 持久性（Durability）：一旦事务提交，不可回滚

2. 事务操作

MySQL默认是自动提交事务的，将每一条语句都当作一个独立的事务执行，可以通过autocommit关闭自动提交事务

查看autocommit模式：`show variables like 'autocommit'`

关闭自动提交：`set autocommit=off` 或 `set autocommit=0`

手动提交事务：`commit`

手动回滚事务：`rollback`