1、数据库

存储数据的仓库，叫**DataBase** ，简称为 db.

MySQL：开源免费的中小型数据库，后来Sun公司收购了MySQL，而Oracle又收购了Sun公司。

目前Oracle推出了收费版本的MySQL，也提供了免费的社区版本。

数据库是以表为单位存储数据。

表中包括两个内容：

列：字段

行：记录

2、SQL

全称 Structured Query Language，**结构化查询语言**。操作关系型数据库的编程语言，定义了 一套操作关系型数 据库统一标准 。

2.1 SQL通用语法

在学习具体的SQL语句之前，先来了解一下SQL语言的通用语法。

1. SQL语句可以单行或多行书写，以分号结尾。
2. SQL语句可以使用空格/缩进来增强语句的可读性。
3. MySQL数据库的SQL语句不区分大小写，关键字建议使用大写。
4. 注释：
    1. 单行注释：-- 注释内容 或 # 注释内容
    2. 多行注释：/* 注释内容 */

2.2 SQL分类

SQL语句，根据其功能，主要分为五类：DDL、DML、DQL、DCL、TCL。

分类	全称	说明
DDL	Data Definition Language	数据定义语言，用来定义数据库对象(数据库，表，字段) **create(创建表) alter(修改) drop(删除) truncate(清空)**
DML	Data Manipulation Language	数据操作语言，用来对数据库表中的数据进行增删改 **insert（添加） delete（删除） update（更新）**
DQL	Data Query Language	数据查询语言，用来查询数据库中表的记录 **select**
DCL	Data Control Language	数据控制语言，用来创建数据库用户、控制数据库的访问权限 **grant （授予） revoke （取消）**
TCL	Transaction Control Language	事务控制语句 **commit （提交） rollback（回滚） savepoint （保存点）**

2.3 DQL 数据查询语言

```
# 查询所有
select * from 表名;

# 查询部分字段的数据
select 字段1，字段2...
from 表名;

# 查询员工表中的所有员工信息
select * from employees;

# 查询员工的员工编号，员工姓名，员工入职日期，职位
select employee_id,first_name,
            hire_date,job_id
from employees;
```

2.4 算术运算符

```
+  -  *   /(DIV)  %(MOD)
```

```sql
# +  -  *  /(DIV)  %(MOD)
# 查询员工的员工编号，姓名，工资，工资+100，工资-100，工资*5，工资除2，工资余3
select employee_id,first_name,salary,
       salary+100,salary-100,salary*5,salary/2,salary%3,
       salary div 2,salary mod 3
from employees;
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

employee_id	first_name	salary	salary+100	salary-100	salary*5	salary/2	salary%3	salary div 2	salary mod 3
100	Steven	24000.00	24100.00	23900.00	120000.00	12000.000000	0.00	12000	0.00
101	Neena	17000.00	17100.00	16900.00	85000.00	8500.000000	2.00	8500	2.00
102	Lex	17000.00	17100.00	16900.00	85000.00	8500.000000	2.00	8500	2.00
103	Alexander	9000.00	9100.00	8900.00	45000.00	4500.000000	0.00	4500	0.00
104	Bruce	6000.00	6100.00	5900.00	30000.00	3000.000000	0.00	3000	0.00
105	David	4800.00	4900.00	4700.00	24000.00	2400.000000	0.00	2400	0.00
106	Valli	4800.00	4900.00	4700.00	24000.00	2400.000000	0.00	2400	0.00

2、 重命名

1）as 关键字
2）用空格

```sql
/*
  重命名:
  1) AS
  2) 空格
*/
select employee_id as '员工编号',first_name as 员工姓名,salary as 工资,
       salary+100 '工资+100',salary-100 '工资-100',salary*5,salary/2,salary%3,
       salary div 2,salary mod 3
from employees;
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

员工编号	员工姓名	工资	工资+100	工资-100	salary*5	salary/2	salary%3	salary div 2	salary mod 3
100	Steven	24000.00	24100.00	23900.00	120000.00	12000.000000	0.00	12000	0.00
101	Neena	17000.00	17100.00	16900.00	85000.00	8500.000000	2.00	8500	2.00
102	Lex	17000.00	17100.00	16900.00	85000.00	8500.000000	2.00	8500	2.00
103	Alexander	9000.00	9100.00	8900.00	45000.00	4500.000000	0.00	4500	0.00
104	Bruce	6000.00	6100.00	5900.00	30000.00	3000.000000	0.00	3000	0.00

```sql
# 查询员工的编号，姓名，年薪
# 空值在参与运算时，结果也是空值
select employee_id as 员工编号,first_name 姓名,
       (salary+salary*commission_pct)*12 年薪
from employees;
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

员工编号	姓名	年薪
142	Curtis	(Null)
143	Randall	(Null)
144	Peter	(Null)
145	John	235200.00
146	Karen	210600.00
147	Alberto	187200.00

3、 去重 distinct

```sql
# distinct 去重
# 查询部门编号
select distinct department_id
from employees;
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

department_id
(Null)
10
20
30
40

```sql
# 查询有哪些经理
select distinct manager_id,department_id
from employees;
```

摘要 | 结果 1 | 剖析 | 状态

anager_id	department_id
(Null)	90
100	90
102	60
103	60
101	100
108	100
100	30

4、过滤

在查询结果的基础上筛选出满足条件的记录。

```sql
select 字段    -- 决定查询结果都显示哪些列的数据
from 表
where 条件     -- 决定查询结果都显示哪些行的数据
```

```sql
-- 过滤 WHERE子句
-- 查询工资大于10000的员工员工编号，姓名，工资，职位
select employee_id,first_name,salary,job_id
from employees
where salary > 10000;

-- 查询职位不是AD_PRES的员工编号 姓名 职位
select employee_id,first_name,job_id
from employees
where job_id != 'AD_PRES';

-- 查询在1987-06-17入职的员工信息
select employee_id,first_name,hire_date
from employees
where hire_date = '1987-06-17';
```

4.1 比较运算符

- **> >= < <= = !=(<>)**

```sql
# 查询部门编号不能90 的员工编号 姓名 部门编号
select employee_id,first_name,department_id
from employees
where department_id <> 90;
```

- **between 值1 and 值2： 在两个值之间的范围(包括边界值)**
- **in()： 等于其中的一个值**
- **like : 模糊查询**
  - **通配符：% 0个或多个字符**
  - **_ 1个字符**
- **is null : 是空值**
- **is not null： 不是空值**

4.2 逻辑运算符

- and、&& 并且
- or 、 || 或者
- not 取反

```sql
-- 其他比较运算符
-- between and 两个值之间的
-- 查询工资是8000~10000之间的员工信息
select *
from employees
where salary between 8000 and 10000;

-- in() 等于其中的一个值
-- 查询在80 90 100部门工作的员工信息
select employee_id,first_name,department_id
from employees
```

```
where department_id in(80,90,100);

-- like 模糊查询
-- 通配符 ： % 代表的是0个或多个字符
#                       _ 代表的是1个字符

# 查询员工姓名包含t的员工信息
select first_name,salary
from employees
where first_name like '%t%';


# 查询第三个字符是t的员工信息
select first_name,salary
from employees
where first_name like '__t%';

# 查询员工姓名中包含 a 且包含 e 的员工信息
select first_name,salary
from employees
where first_name like '%a%' and first_name like '%e%';

# is null 是空值
# is not null 不是一个空值

# 查询 奖金不为空 的员工信息
select employee_id,first_name,salary,commission_pct
from employees
where commission_pct is not null;

# <=> 安全等于，专用于比较空值的
select employee_id,first_name,salary,commission_pct
from employees
where commission_pct <=> null
where commission_pct is null;
```

5、排序

- **升序 asc ，默认**
- **降序 desc**

```
select 字段
from 表
where 条件
order by 字段 asc/desc;
```

```sql
# 查询员工编号，姓名，工资，按照工资降序排序
select employee_id,first_name,salary
from employees
order by salary desc;

# 查询员工的编号，姓名，职位，工资，
# 职位IT_PROG，按照工资升序排序
select employee_id,first_name,job_id,salary
from employees
where job_id = 'IT_PROG'
order by salary;
```

5.1 按照别名排序

```sql
# 按照别名排序
# 查询员工编号，员工姓名，职位，年薪,按照年薪升序排序
select employee_id,first_name,job_id,salary * 12 as sal
from employees
order by sal
```

```sql
# 注意：别名可以排序，但是不能用别名作为过滤的条件

# 查询员工编号，员工姓名，职位，年薪，年薪大于5万的
select employee_id,first_name,job_id,salary * 12 as sal
from employees
where sal > 50000;
```

消息 | 摘要 | 状态

查询

```
select employee_id,first_name,job_id,salary * 12 as sal
from employees
where sal > 50000
```

消息

```
1054 - Unknown column 'sal' in 'where clause'
```

5.2 多列排序

```sql
# 查询员工信息，按照部门编号升序排序，按照工资降序排序
select employee_id,first_name,department_id,salary
from employees
order by department_id asc,salary desc;
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

employee_id	first_name	department_id	salary
178	Kimberely	(Null)	7000.00
200	Jennifer	10	4400.00
201	Michael	20	13000.00
202	Pat	20	6000.00
114	Den	30	11000.00
115	Alexander	30	3100.00
116	Shelli	30	2900.00

6、分页 limit

```
一页显示10行记录，第三页的数据是从 ？ 到 ？
1 ，  1 - 10
2 ，  11 - 20
3 ，  21 - 30

语法结构： limit [位置偏移量]，行数


# 分页
# 每页10行记录，查询第一页的数据
select *
from employees
# limit 0,10;
limit 10;

# 查询第3页的数据
select *
from employees
limit 21,10;
```

7、函数

函数 是指一段可以直接被另一段程序调用的程序或代码。 也就意味着，这一段程序或代码在MySQL中已经给我们提 供了，我们要做的就是在合适的业务场景调用对应的函数完成对应的业务需求即可。

MySQL中的函数主要分为以下四类： **字符串函数、数值函数、日期函数、流程函数。**

函数的分类：

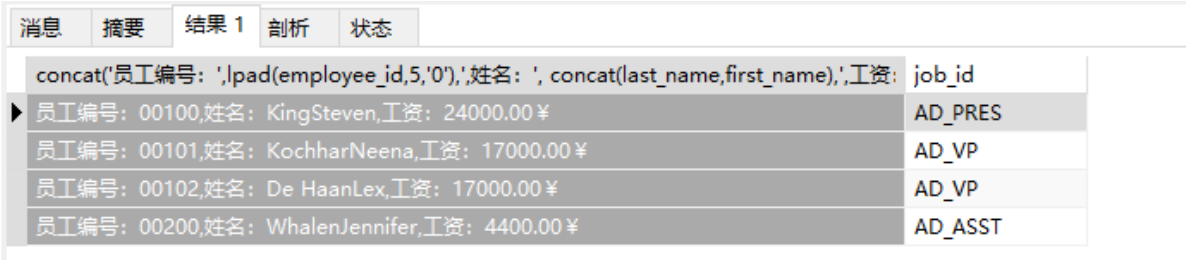
- 单行函数
  - 每一行都有一个结果
- 多行函数（聚合函数）
  - 对多行记录操作，返回一个结果

7.1 字符串函数

MySQL中内置了很多字符串函数，常用的几个如下：

函数	功能
CONCAT(S1,S2,...Sn)	字符串拼接，将S1, S2, ... Sn拼接成一个字符串
LOWER (str) / LCASE(str)	将字符串str全部转为小写
UPPER (str) / UCASE(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
LTRIM (str)	去掉字符串 str 开始处空格
RTRIM (str)	去掉字符串 str 结尾处空格
SUBSTRING (str,start,len) SUBSTR()	返回从字符串str从start位置起的len个长度的字符串
REPLACE()	将一个字符串中的指定子串替换为另一个字符串
LENGTH / CHAR_LENGTH(S) / CHARACTER_LENGTH(s)	返回字符串 s 得字符数
FIND_IN_SET(S1,S2)	返回在字符串s2中与s1匹配的字符串的位置
FORMAT(x,n)	可以将数字x进行格式化“#,###.##”,将x保留到小数点后n位，最后一位四舍五入
INSERT(s1,x,len,s2)	字符串 s2 替换 s1 的x位置开始长度为 len 的字符串
LOCATE(S1 , S) / POSITION(S1 IN s)	从字符串 s 中获取 s1 得开始位置
REPEAT (s，n)	将字符串 s 重复 n 次
REVERSE(s)	将字符串s的顺序反过来
strcmp(s1,s2)	比较字符串 s1 和 s2，如果 s1 与 s2 相等返回0 ，如果 s1>s2 返回 1，如果 s1<s2 返回 -1

```
/*
    练习 ：
        员工编号：不够五位前面补0
        完整姓名
        工资在后面加￥

        显示结果：员工编号：00100 ，  姓名：StevenKing，工资：24000.00￥
        条件：职位的前两个字符是AD
*/

-- select
lpad(employee_id,5,'0'),concat(last_name,first_name),concat(salary,'￥')

select concat('员工编号：',lpad(employee_id,5,'0'),',姓名：',
            concat(last_name,first_name),',工资：',   concat(salary,'￥')),
            job_id
from employees
where substr(job_id,1,2) = 'AD';
```



消息 | 摘要 | 结果 1 | 剖析 | 状态

concat('员工编号：',lpad(employee_id,5,'0'),',姓名：', concat(last_name,first_name),',工资：	job_id
员工编号：00100,姓名：KingSteven,工资：24000.00￥	AD_PRES
员工编号：00101,姓名：KochharNeena,工资：17000.00￥	AD_VP
员工编号：00102,姓名：De HaanLex,工资：17000.00￥	AD_VP
员工编号：00200,姓名：WhalenJennifer,工资：4400.00￥	AD_ASST

7.2 数值函数

ceil()	**向上取整**
floor	向下取整
mod()	求余数
rand()	返回0-1之间的随机小数
round()	四舍五入
truncate(x,y)	保留指定小数位数，不进行四舍五入（截断）

```
# 数值函数
# 向上取整
select ceil(3.14) -- 4

# 向下取整
select floor(3.14) -- 3

# 求余数
select mod(5,3) -- 2

# 随机数
select rand(); -- 0~1之间的随机小数

# 四舍五入
# 第二个参数代表的是保留小数点后几位
select round(3.1415926,2) -- 3.14
select round(3.1415926,0) -- 3
select round(45.56,0) -- 46
select round(45.56,-1) -- 50

# 截断
select truncate(3.1455926,2) -- 3.14
select truncate(3.1455926,0) -- 3
select truncate(45.56,-1) -- 40
```

7.3 日期函数

curdate()	**当前日期，只包含：年、月、日**
curtime()	当前时间，只包含：时、分、秒
now()	当前日期和时间
YEAR()	年
MONTH()	月
DAY()	日
datediff()	两个日期相差的天数
date_add()	在指定日期上加上若干天
date_sub	减去指定的时间间隔
to_days()	将日期转成天数
from_days()	将天数转成日期
date()	在指定的日期+时间上获取日期
time()	在指定的日期+时间上获取时间
hour()	时
minute()	分
second()	秒
DATE_FORMAT()	日期格式化 %y年%m月%d日 %r
last_day()	一个月中的最后一天
DAYNAME()	星期几
DAYOFMONTH（）	本月的第几天
DAYOFWEEK（）	本周的星期几 1是星期日 2是星期一
DAYOFYEAR（）	本年的第几天

```
# 日期函数
select curdate() 日期,  -- 2025-01-17
             curtime() 时间,  -- 09:22:27
             now() 日期和时间; -- 2025-01-17 09:22:27

# 查询员工截止到今天入职的天数
select first_name,hire_date,
                datediff(now(),hire_date) 入职天数
from employees;

# 10天后的日期
select date_add(CURDATE(),INTERVAL 10 DAY) -- 2025-01-27

# 5年后的日期
```

```sql
select date_add(CURDATE(),INTERVAL 5 YEAR) -- 2030-01-17

# 1年前的日期
select date_sub(CURDATE(),INTERVAL 1 YEAR) -- 2024-01-17

# 获取年 、月 、日
select now(),
             year(now()),
             month('1987-5-10'),
             day('1987-5-10');

# 将日期转成天数
select to_days(now()) -- 739633

# 将天数转成日期
select from_days(739633) -- 2025-01-17

# 在指定的日期+时间上获取日期
select date(now()) -- 2025-01-17
# 在指定的日期+时间上获取时间
select time(now()) -- 09:51:01
-- 小时
select hour(now()) -- 9
-- 分钟
select MINUTE(now()) -- 52
-- 秒
select SECOND(now()) -- 59


-- 将日期转换成指定的格式
select now(),DATE_FORMAT(now(),'%y年%m月%d日 %r') 日期

-- 查询每个月最后一天入职的员工
select first_name,hire_date
from employees
where hire_date = LAST_DAY(hire_date)

-- 星期几
select DAYNAME(now()) -- Friday
-- 本月的第几天
select DAYOFMONTH('2025-1-17')  -- 17
-- 本周的星期几 1是星期日 2是星期一 ...
select DAYOFWEEK('2025-1-17') -- 6
-- 本年的第几天
select DAYOFYEAR(now()) -- 17
```

```sql
-- 练习：查询每个员工的入职天数，并根据入职天数降序排序。
select first_name,
       datediff(curdate(),hire_date) 入职天数
from employees
order by 入职天数 desc

-- 入职了多少周？
select first_name,
       datediff(curdate(),hire_date)/7 周,
       ROUND(datediff(curdate(),hire_date)/7,0)
from employees
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

first_name	周	ROUND(datediff(curdate
Alexander	1548.1429	1548
Shelli	1412.2857	1412
Sigal	1434.1429	1434
Guy	1365.7143	1366

```sql
-- 30年前入职的员工信息
select first_name,hire_date
from employees
where hire_date <= DATE_SUB(curdate(),interval 30 year)
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

first_name	hire_date
Steven	1987-06-17
Neena	1989-09-21
Lex	1993-01-13
Alexander	1990-01-03
Bruce	1991-05-21
Nancy	1994-08-17
Daniel	1994-08-16
Den	1994-12-07
Jennifer	1987-09-17
Susan	1994-06-07
Hermann	1994-06-07
Shelley	1994-06-07
William	1994-06-07

7.4 流程函数

1、if(value , t , f)
如果value为true,返回t，否则返回f

2、ifnull(value1,value2)
如果value1不为空，返回value1,否则返回value2

3、表达式

```
        case 字段 when 比较值1 then 要执行的内容
                        when 比较值2 then 要执行的内容
                          ...
                          else 执行的内容
        end
```

相当于switch case

```sql
-- 流程语句
-- IF(expr1,expr2,expr3)
select if(10/2,'偶数','奇数');

-- ifnull(a,b) a不为空，返回a ，为空返回b
select ifnull(manager_id,first_name)
from employees
```

```sql
-- 查询员工的年薪
-- 空值在参与运算时得到是一个空值
select first_name,salary,job_id,
       (salary+salary*ifnull(commission_pct,0))*12 年薪
from employees;

```

消息 | 摘要 | 结果 1 | 剖析 | 状态

first_name	salary	job_id	年薪
Steven	24000.00	AD_PRES	288000.00
Neena	17000.00	AD_VP	204000.00
Lex	17000.00	AD_VP	204000.00
Alexander	9000.00	IT_PROG	108000.00
Bruce	6000.00	IT_PROG	72000.00

```sql
-- 查询员工编号，姓名，工资，经理编号
-- 经理编号为100 工资涨10%
#           为101 工资减100
#           为102 工资加100
select employee_id,salary,manager_id,
       case manager_id when 100 then salary+salary*0.1
                       when 101 then salary-100
                       when 102 then salary+100
                       else salary
       end 工资
from employees;

```

消息 | 摘要 | 结果 1 | 剖析 | 状态

employee_id	salary	manager_id	工资
100	24000.00	(Null)	24000.00
101	17000.00	100	18700.00
102	17000.00	100	18700.00
103	9000.00	102	9100.00
104	6000.00	103	6000.00
105	4800.00	103	4800.00
106	4800.00	103	4800.00

8、多表查询

通过sql语句，连接两张或两张以上表进行查询。

8.1 笛卡尔集

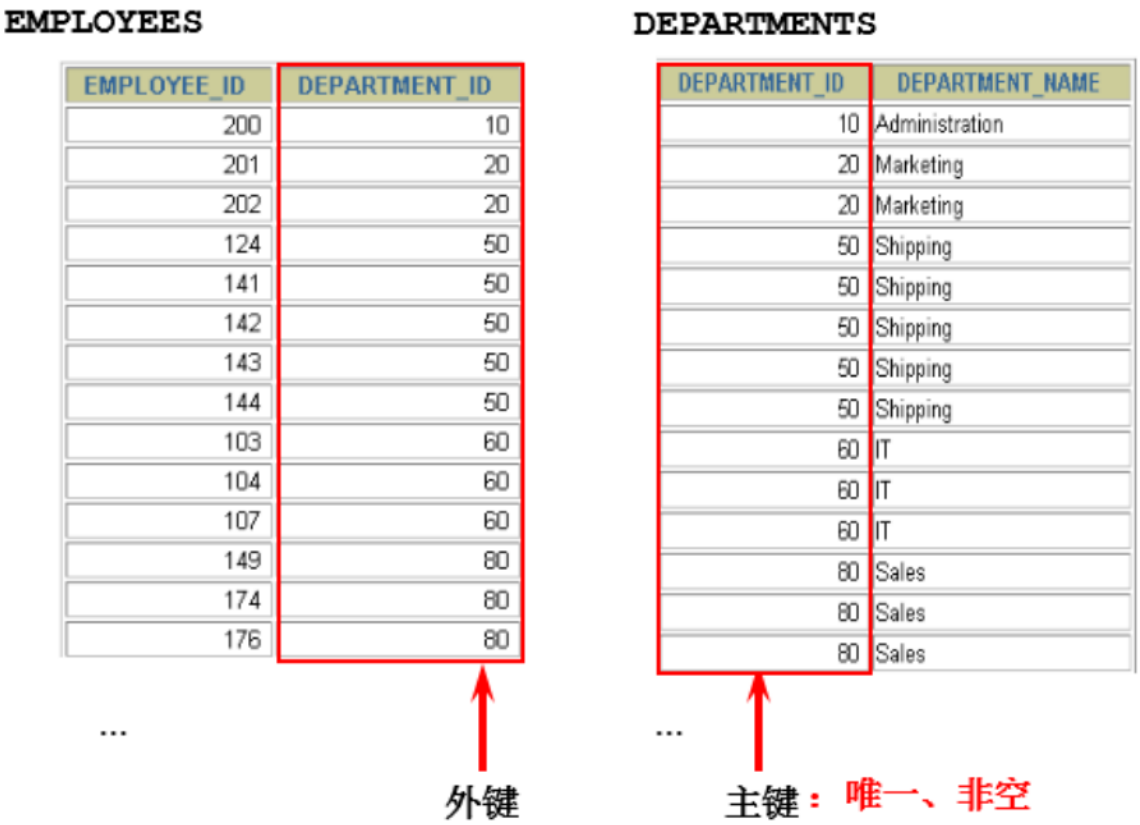
多张表中的数据进行**交叉连接**，查询结果是多表中的记录数的乘积，没有连接条件。

例如： A 表中有8行记录，B表中有5行记录，查询结果就是8*5=40行记录。

```
-- 查询员工表和部门表的信息
-- 员工编号，员工姓名，工资，部门名称
select employee_id,salary,department_name
from employees,departments;

-- 交叉连接也可以写成 cross join ,其中corss 也可以省略
select employee_id,salary,department_name
from employees corss join departments;
```

8.2 等值连接

将一张表中的主键字段与另一张表中的外键字段做为连接条件。




```
# 2、等值连接
#     一张表中的主键字段与另一张表中的外键字段做为连接的条件
#     主键：一张表中的唯一且不为空的字段
#     外键：来自于另一张表中的主键字段

# 员工编号，员工姓名，工资，部门名称
# 连接条件：  员工表中的部门编号 等于 部门表中的部门编号
select employee_id,first_name,salary,department_name
from employees e,departments d
where e.department_id = d.department_id;


# 员工编号，员工姓名，工资，部门名称,所在城市
select employee_id,first_name,salary,department_name,city
from employees e,departments d,locations l
where e.department_id = d.department_id AND
```

```
      d.location_id = l.location_id;


# sql1999的语法结构：
-- 等值连接
-- inner join  ,其中 inner 也可以省略不写
select employee_id,first_name,d.department_id,department_name
from employees e inner join departments d
on e.department_id = d.department_id

-- 连接条件，也可以使用 using()
-- 前提： using中的字段必须是两表中的共同字段，字段名称、类型必须相同
--       使用 using 不需要加别名，明确所属
select employee_id,first_name,department_id,department_name
from employees inner join departments
using(department_id)
```

8.3 非等值连接

EMPLOYEES

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Zlotkey	10500
Abel	11000
Taylor	8600

JOB_GRADES

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

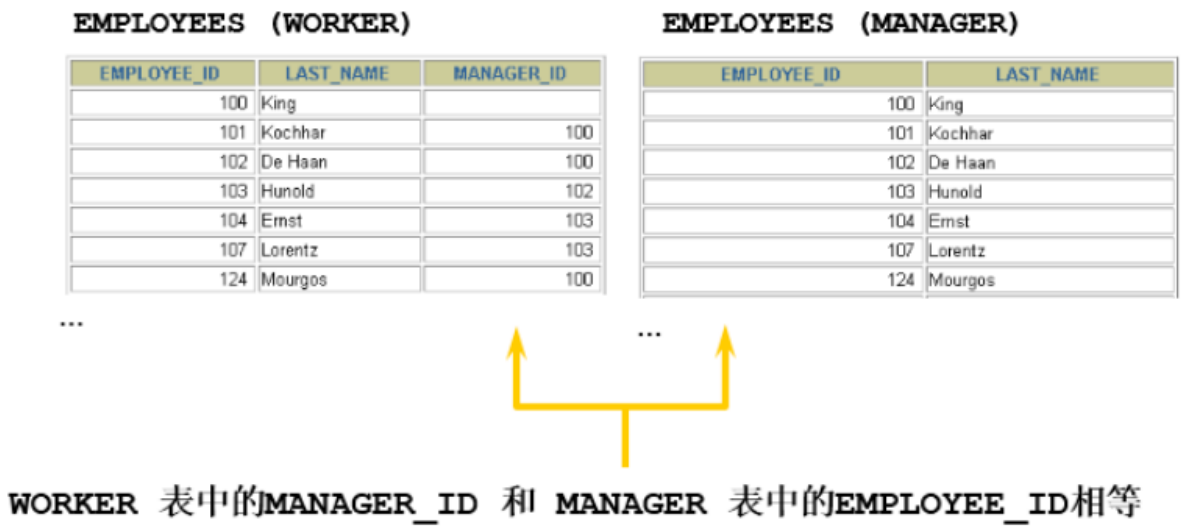
EMPLOYEES表中的列工资应在JOB_GRADES表中的最高工资与最低工资之间

```
# 查询员工编号，姓名，工资，工资等级
select employee_id,first_name,salary,grade_level
from employees,job_grades
where salary between lowest_sal and highest_sal;
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

employee_id	first_name	salary	grade_level
100	Steven	24000.00	E
101	Neena	17000.00	E
102	Lex	17000.00	E
103	Alexander	9000.00	C
104	Bruce	6000.00	C
105	David	4800.00	B

8.4 自连接




```
# 自连接，把一张表拆分成两张表，自己和自己连接
# 查询员工编号，员工姓名，经理编号，经理姓名
# 连接条件：员工的经理编号 = 经理的员工编号
select e.employee_id 员工编号,e.first_name 员工姓名,
            m.employee_id 经理编号,m.first_name 经理姓名
from employees e,employees m
where e.manager_id = m.employee_id;
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

员工编号	员工姓名	经理编号	经理姓名
101	Neena	100	Steven
102	Lex	100	Steven
103	Alexander	102	Lex
104	Bruce	103	Alexander
105	David	103	Alexander
106	Valli	103	Alexander
107	Diana	103	Alexander
108	Nancy	101	Neena

```
181 # 查询员工表中员工工资大于8000的员工姓名和他的经理姓名
182 select e.first_name 员工姓名,e.salary 员工工资,m.first_name 经理姓名
183 from employees e,employees m
184 where e.manager_id = m.employee_id AND
185       e.salary > 8000;
186
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

员工姓名	员工工资	经理姓名
Den	11000.00	Steven
Adam	8200.00	Steven
John	14000.00	Steven
Karen	13500.00	Steven
Alberto	12000.00	Steven

8.5 外连接

除了查询满足条件的记录以外，外连接还可以查询某一方不满足条件的记录。

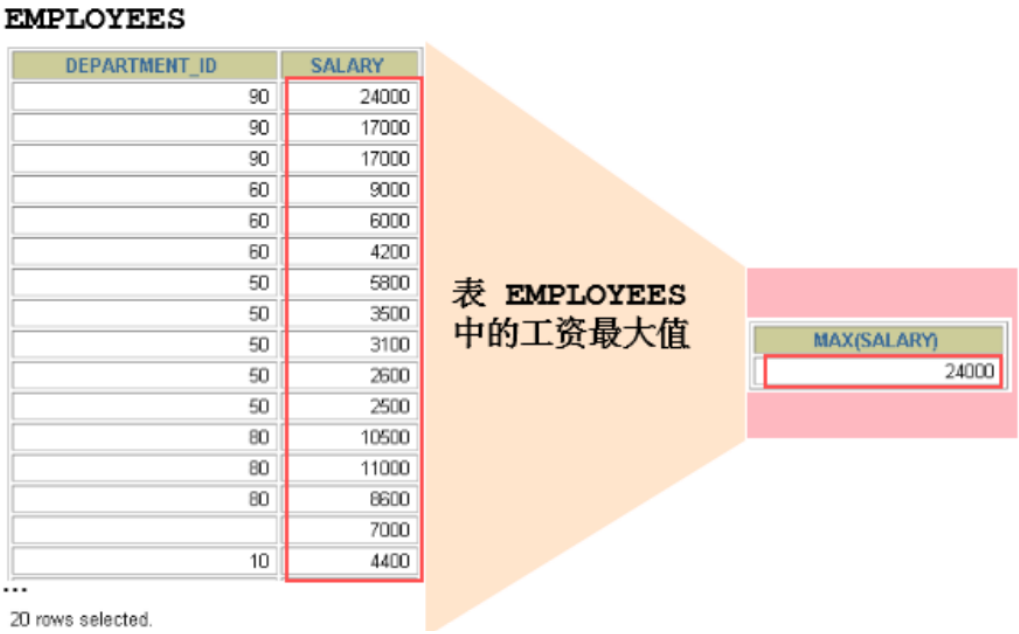
- 左外连接：
  - **left outer join / left join**
  - 查询的结果是等值连接（内连接）+ 左边中不满足连接条件的记录
- 右外连接
  - **right outer join / right join**
  - 查询的结果是等值连接（内连接）+ 右边中不满足连接条件的记录
- 满外链接
  - **union**
  - 将左外连接与右外连接拼接起来
- **outer**关键字可以省略不写。

```
# 外连接，可以查询出不满足连接条件的记录
#         左外连接 ：左表中不满足条件的记录+等值连接的结果


# 查询员工表中的所有员工信息和所在的部门名称
# 查询没有部门的员工
select employee_id,first_name,e.department_id,department_name
from employees e left outer join departments d
on e.department_id = d.department_id;


# 右外连接 ：右表中不满足条件的记录+等值连接的结果
# 没有员工的部门
select employee_id,first_name,d.department_id,department_name
from employees e right outer join departments d
on e.department_id = d.department_id;

# outer可以省略不写
select employee_id,first_name,d.department_id,department_name
from employees e right join departments d
on e.department_id = d.department_id;

# union
select employee_id,first_name,d.department_id,department_name
from employees e left join departments d
on e.department_id = d.department_id
union
select employee_id,first_name,d.department_id,department_name
from employees e right join departments d
on e.department_id = d.department_id;
```

9、聚合函数和分组查询

9.1 聚合函数

- 什么是聚合函数
  - 聚合函数作用于一组数据，并对一组数据返回一个值。

- 

- 聚合函数类型
  - AVG() 平均值
  - SUM() 和
  - MAX() 最大值
  - MIN() 最小值
  - COUNT() 统计个数
- **聚合函数不能嵌套调用。**比如不能出现类似“AVG(SUM(字段名称))”形式的调用。

```
# 查询员工表中最高工资 、 最低工资、平均工资、工资总和、公司人数
select Max(salary) 最高工资,
             min(salary) 最低工资,
             avg(salary) 平均工资,
             sum(salary) 工资总和,
             count(employee_id),count(*) 人数
from employees;
```

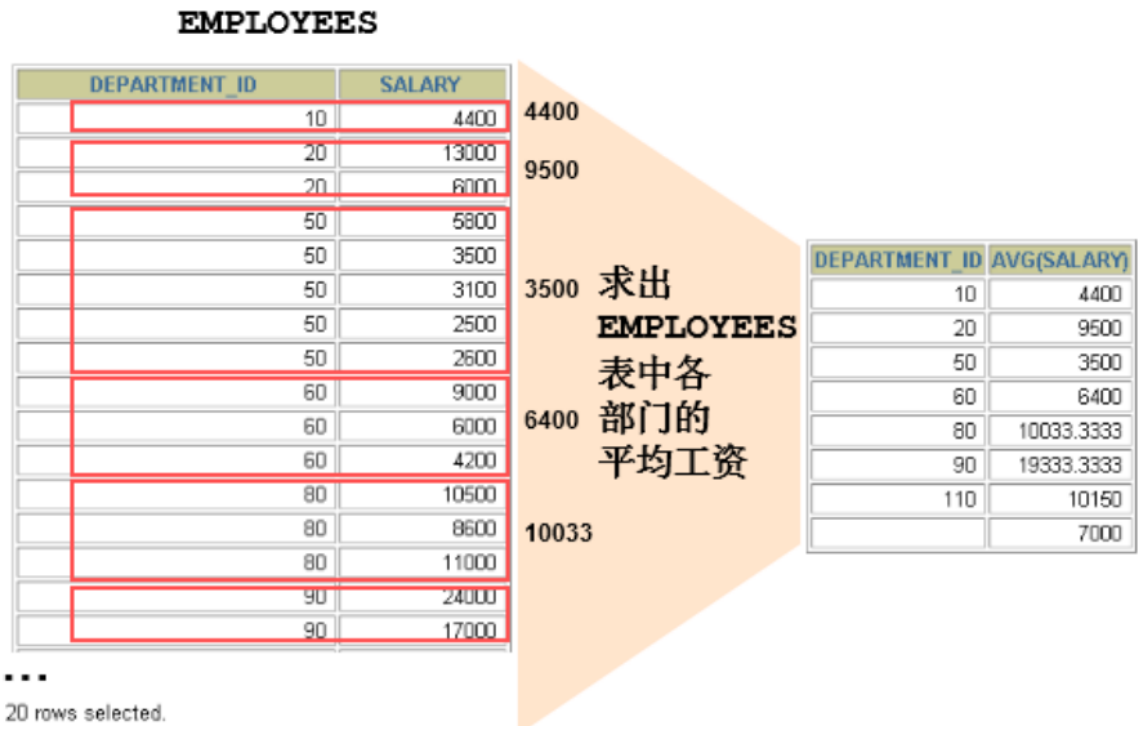
消息 | 摘要 | 结果 1 | 剖析 | 状态

最高工资	最低工资	平均工资	工资总和	count(employee_id)	人数
24000.00	2100.00	6461.682243	691400.00	107	107

- **聚合函数忽略空值的**

```
# 查询有多少人有奖金?
# 聚合函数忽略空值的
select count(commission_pct) from employees; -- 35

# 可以用 ifnull 解决这个问题
select count(ifnull(commission_pct,0)) from employees; -- 107
```

9.2 分组查询



语法结构：

```
select 字段
from 表
where 条件
group by 字段
having 分组后的条件
order by 排序
limit 行数
```

```
# 查询各个部门的平均工资
# 按照平均工资降序排序
select department_id,avg(salary) 平均工资
from employees
group by department_id
order by avg(salary) desc;

# 查询工资大于2500，并且各个部门的平均工资大于5000的员工信息
select department_id,avg(salary)
from employees
where salary > 2500
group by department_id
having avg(salary) > 5000;

# 注意：除了分组函数以外的字段都要放在group by 中
```

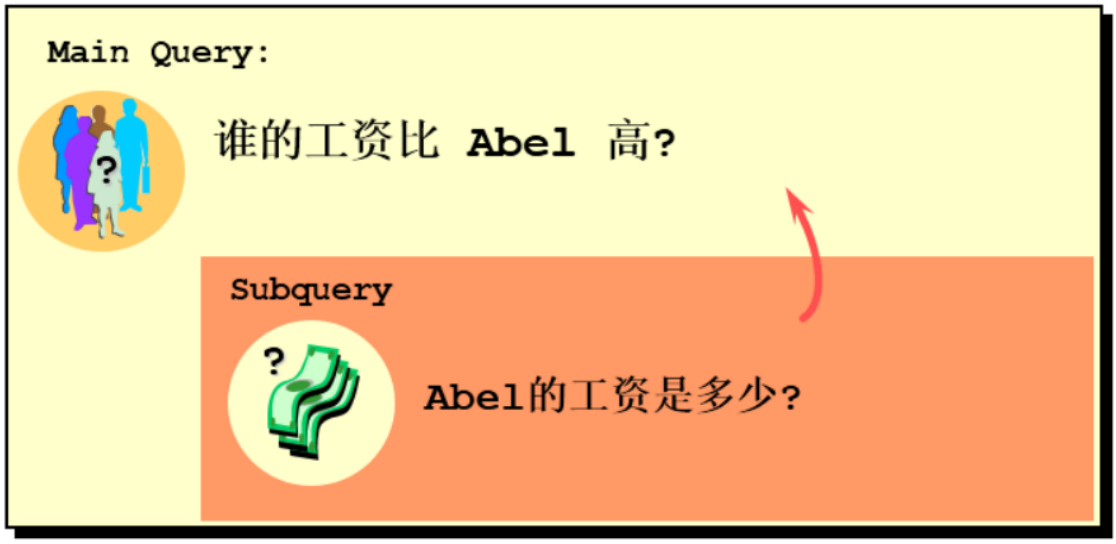
```sql
# 查询部门名称，以及该部门下的平均工资，工资总和，最高工资，最低工资，部门人数
select department_name,avg(salary),
       sum(salary),max(salary),
       min(salary),count(*)
from employees join departments
using(department_id)
group by department_name
```

消息 | 摘要 | 结果 1 | 剖析 | 状态

department_name	avg(salary)	sum(salary)	max(salary)	min(salary)	count(*)
IT	5760.000000	28800.00	9000.00	4200.00	5
Finance	8600.000000	51600.00	12000.00	6900.00	6
Purchasing	4150.000000	24900.00	11000.00	2500.00	6
Shipping	3475.555556	156400.00	8200.00	2100.00	45
Sales	8955.882353	304500.00	14000.00	6100.00	34
Administration	4400.000000	4400.00	4400.00	4400.00	1

10、子查询

子查询指一个查询语句嵌套在另一个查询语句内部的查询，这个特性从MySQL 4.1开始引入。




```sql
select *
from 表
where 字段 > (select 字段 from 表名 where 条件);

在（）中写的查询就叫子查询。
```

- 子查询（内查询）在主查询之前一次执行完成。
- 子查询的结果被主查询（外查询）使用 。
- 注意事项
  - 子查询要包含在括号内
  - 将子查询放在比较条件的右侧
  - 单行操作符对应单行子查询，多行操作符对应多行子查询

```sql
# 把员工表中的数据添加到emp2这个表中
# 创建表的同时将另一张表中的数据粘贴过来
create table emp2
as select * from emp;
```

```
select * from emp2

# 通过子查询创建表
create table emp3
as select * from (select empno,ename from emp) e

# 将员工的工资改成与lisa一样的

update emp2 set salary = (select salary from emp where ename='lisa')

-- 利用子查询，将emp中的数据全部添加到emp2中
INSERT into emp2 select * from emp;

-- 部分字段的数据
INSERT into emp2(empno,ename) select empno,ename from emp;
```

10.1 单行比较运算符

单行子查询，查询结果返回的是一个值。

操作符	含义
=	equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
<>	not equal to

```
# 查询谁的工资比 Nancy 高
# 1、查询Nancy的工资是多少？
select salary from employees where first_name = 'Nancy';

# 2、拿其他员工的工资与其比较
select first_name,salary
from employees
where salary > (select salary from employees where first_name = 'Nancy');

# 返回job_id与141号员工相同，salary比143号员工多的员工姓名，job_id和工资
select first_name,job_id,salary
from employees
where job_id = (select job_id from employees where employee_id = 141) AND
            salary > (select salary from employees where employee_id = 143)

# 查询工资最少的员工姓名，job_id,salary
select min(salary) from employees

select first_name,job_id,salary
from employees
```

```
where salary = (select min(salary) from employees)


# 查询每个部门最低工资大于部门50的最低工资的部门id和工资
select min(salary) from employees where department_id = 50

select department_id,salary
from employees
group by department_id
having min(salary) > (select min(salary) from employees where department_id =
50)
```

10.2 多行比较操作符

操作符	含义
IN	等于列表中的**任意一个**
ANY	需要和单行比较操作符一起使用，和子查询返回的**某一个**值比较
ALL	需要和单行比较操作符一起使用，和子查询返回的**所有**值比较
SOME	实际上是ANY的别名，作用相同，一般常使用ANY

```
# 返回其它job_id中比job_id为‘IT_PROG’任一工资低的员工的员工号、姓名、job_id 以及salary
select salary from employees where job_id = 'IT_PROG';

select employee_id,first_name,job_id,salary
from employees
where salary <any (select salary from employees where job_id = 'IT_PROG')

# 查询工资在5000-10000之间的员工所在部门的所有员工信息
select *
from employees
where department_id in(select department_id from employees where salary between
5000 and 10000)

# 工资大于50这个部门的所有员工工资的员工信息
select *
from employees
where salary > (select max(salary) from employees where department_id = 50)

select *
from employees
where salary >all (select salary from employees where department_id = 50)

# 查询平均工资最低的部门id
select department_id
from employees
group by department_id
having avg(salary) = (SELECT MIN(avg_sal)
                      FROM (
                            SELECT AVG(salary) avg_sal
                            FROM employees
```

```
                                GROUP BY department_id
                                ) dept_avg_sal)
```

11、DDL 数据定义语言

创建表 **create**、修改表 **alter**、删除表 **drop**

数据类型	描述
INT	从-2^31到2^31-1的整型数据。存储大小为 4个字节
CHAR(size)	定长字符数据。若未指定，默认为1个字符，最大长度255
VARCHAR(size)	可变长字符数据，根据字符串实际长度保存，**必须指定长度**
FLOAT(M,D)	单精度，占用4个字节，M=整数位+小数位，D=小数位。 D<=M<=255,0<=D<=30，默认M+D<=6
DOUBLE(M,D)	双精度，占用8个字节，D<=M<=255,0<=D<=30，默认M+D<=15
DECIMAL(M,D)	高精度小数，占用M+2个字节，D<=M<=65，0<=D<=30，最大取值范围与DOUBLE相同。
DATE	日期型数据，格式'YYYY-MM-DD'
BLOB	二进制形式的长文本数据，最大可达4G
TEXT	长文本数据，最大可达4G

11.1 创建表 修改表 删除表

```
# 创建表 create table
create table dept(
    # 字段名 类型 长度 有无默认值 约束
    deptno int,
    dname varchar(99),
    loc varchar(99)
)


# 修改表 alter table
#   1、添加新的字段 add COLUMN
#   2、删除字段 drop 字段名
#   3、修改字段 modify 字段 类型 长度

# 向部门表dept中添加经理编号
alter table dept add column man_id int

# 修改经理编号为字符类型
alter table dept modify man_id varchar(99)
```

```
# 删除经理编号这个字段
alter table dept drop man_id

# 删除表
drop table dept


# 清空表
truncate table dept;
```

12、DML 语句

数据定义语言，可以向表中**添加数据（insert），删除数据(delete)，修改数据(update)**

12.1 添加

```
# 添加数据
# 1) 添加一行数据（部分不为空的字段）
insert into 表(字段1，字段2，字段3) values(值1，值2，值3)

注意： values后的值的顺序、类型、个数要与 表名后面的字段保持一致。

# 2）添加多行数据（所有字段）
insert into 表 values(值1，值2，值3),
                     (值1，值2，值3),
                     (值1，值2，值3)

注意：values后的值的顺序、类型、个数必须要与表中的字段的个数、顺序、类型保持一致。
```

```
/* 创建emp表
        empno 员工编号 int 自动递增的字段(auto_increment)值从1开始，
        ename 员工姓名 valchar(99),
        sex 员工姓名，varchar(10),
        age 年龄 int,
        salary 工资 double,
        hireDate 入职日期 date
*/
create table emp(
    empno int auto_increment primary key, -- 自动递增的字段必须是一个主键字段 primary
key
    ename varchar(99),
    sex varchar(10),
    age int,
    salary double default 0,
    hireDate date
)


# 插入一行数据
```

```
insert into emp(ename,sex,age,salary,hireDate)
values('tom','男',18,5800.66,now());

# 插入多行数据
# 自动提增的字段、有默认值的字段，可以使用default 代替
insert into emp values(default,'lisa','女',19,8000,'2025-1-1'),
                                                (default,'jack','男',19,8000,'2025-1-1'),
                                                (default,'susan','女',19,8000,'2025-1-1')


select * from emp;
```

12.2 修改

```
update set 表名 字段 = 值，字段2 = 值 [where]
```

```
# 修改
# update 表名 set 字段=值，字段=值 【where】

# 将empno 为1 的员工姓名改成张三，年龄改成28
update emp set ename='张三',age = 28 where empno = 1;

# 将性别为男的员工，工资涨5%
update emp set salary = salary+salary*0.5 where sex='男'
```

12.3 删除数据

```
# delete from 表名 【where】


# 删除 今天入职的员工
delete from emp where hireDate = '2025-1-20';

# 清空表
delete from emp;
```

> delete 和 truncate 的区别：
>
> delete删除数据，添加新的记录，自动递增的字段顺序编号
>
> truncate删除数据，添加新的记录，自动递增的字段编号从1开始

13、约束

概念：约束是作用于表中字段上的规则，用于限制存储在表中的数据。 目的：保证数据库中数据的正确、有效性和 完整性。

约束	描述	关键字
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是唯一、不重复的	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，如果未指定该字段的值，则采用默认值	DEFAULT
检查约束(8.0.16版本之后)	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之间建立连接，保证数据的一致性和完整性	FOREIGN KEY

> 注意：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。

- **建表的同时添加约束**

```
/* 约束 : 用来规范表中的字段

    主键约束： primary key 非空且唯一的字段，一张表中只能有一个主键字段
                                                搭配着 auto_increment（自动递增）
一起使用
  唯一约束： unique 唯一，允许有空值
    非空约束： not null
    默认值：   default
    检查约束： check
    外键约束： foreign key 用来连接另一张表中的主键字段
*/
create table user_tb(
    id int primary key auto_increment,
    name varchar(10) not null unique,
    age int check(age > 0 && age <= 120),
    status char(1) default 1,
    gender char(1)
)

insert into user_tb values(default,'tom',18,default,'男'),

(default,'lisa',18,default,'女')


select * from user_tb
```

- **建表后添加约束**

```
alter table 表
add constraint 约束名
foreign key(外键字段) references 主键表(主键字段)
```

```
# 创建dept
create table dept(
    deptno int primary key auto_increment,
    dname varchar(99)
)

insert into dept values(default,'销售部一');
insert into dept values(default,'销售部二');

insert into dept values(default,'市场部一');
insert into dept values(default,'市场部二');

select * from dept;

# 给emp表添加一个字段，外键连接dept表中的deptno
alter table emp add column deptno int

# 给deptno 添加一个外键，连接 dept 表中的 deptno字段
# constraint 约束，deptno_fk约束的名字
/*
    alter table 表
    add constraint 约束名
    foreign key(外键字段) references 主键表(主键字段)
*/
alter table emp
add constraint deptno_fk
foreign key(deptno) references dept(deptno)


# 向emp表中的添加数据
insert into emp values(default,'李四','男',22,9000,now(),1)

select * from emp

# 注意：
#   1.子目录（外键表）中的数据必须是父目录（主键表）中存在的
#   2.主键与外键关联后，主表中的数据不能随意删除和修改

delete from dept where deptno = 1;

# 级联删除和级联更新
alter table emp
add constraint deptno_fk foreign key(deptno) references dept(deptno)
on update cascade on delete cascade;


# 删除外键约束
```

```
alter table emp drop FOREIGN KEY deptno_fk;
```

14、事务 TCL

把多个操作看成是一个整体，要么都执行，要么就都不执行。

```
# 事务：把多个操作看成是一个整体，要么都执行，要么就都不执行。
# 创建account表

create table account(
    id int primary key auto_increment,
    name varchar(99),
    money double check(money>0)  -- 检查约束，余额必须大于0
)

# 插入两行记录
insert into account values(default,'张三',1000),(default,'李四',2000)

select * from account;

# 1.张三的金额减1000
update account set money = money - 1000 where name = '张三';

# 2.李四的金额加1000
update account set money = money + 1000 where name = '李四';


# mysql 中的DML操作是自动提交的，需要改成是手动提交
SET @@autocommit = 0;

# 开启事务
start TRANSACTION;

# 1.张三的金额减1000
update account set money = money - 1000 where name = '张三';

# 2.李四的金额加1000
update account set money = money + 1000 where name = '李四';

# 提交
COMMIT;

# 回滚
ROLLBACK;
```

14.1 事务四大特性

- 原子性（Atomicity）：事务是不可分割的最小操作单元，要么全部成功，要么全部失败。
- 一致性（Consistency）：事务完成时，必须使所有的数据都保持一致状态。
- 隔离性（Isolation）：数据库系统提供的隔离机制，保证事务在不受外部并发操作影响的独立环境下运行。
- 持久性（Durability）：事务一旦提交或回滚，它对数据库中的数据的改变就是永久的。

- 上述就是事务的四大特性，简称 ACID 。

15、试图 View

从表中抽取出部分数据存放到视图中。

```
# 创建视图
create view 视图的名字
as select 字段1，字段2，字段3 from 表 where 条件;
```

```
-- 视图 VIEW
-- 将表中的部分数据抽取出来，利用试图方便查询

-- 创建视图
create view empView
as select employee_id,first_name,salary,
                    job_id,hire_date,department_id
from employees where department_id = 50;

-- 查询最低工资的员工信息
select *
from empView
where salary = (select min(salary) from empView);

-- 创建视图，给字段起别名
-- 方式一：
create view empView2
as select employee_id empno,first_name ename,salary sal
from employees where department_id = 50;

-- 方式二：
create view empView3(empno,ename,sal)
as select employee_id,first_name,salary
from employees where department_id = 50;

-- 创建或修改视图
create or replace view empView3(empno,ename,sal,name)
as select employee_id,first_name,salary,concat(first_name,last_name)
from employees where department_id = 50;

select * from empView3

-- 创建复杂的视图
-- 部门名称，最低工资，最高工资，平均工资，总工资，人数
create view empView4
(dname,minSal,maxSal,avgSal,sumSal,num)
as select
department_name,min(salary),max(salary),avg(salary),sum(salary),count(*)
from employees inner join departments
using(department_id)
group by department_name;
```

```sql
select * from empView4;

-- 向视图中插入数据
create view empView5 as select * from emp

select * from empView5

insert into empView5 values(default,'张三','男',22,18000,now(),2)

select * from emp


-- 删除视图
drop view empView4
```