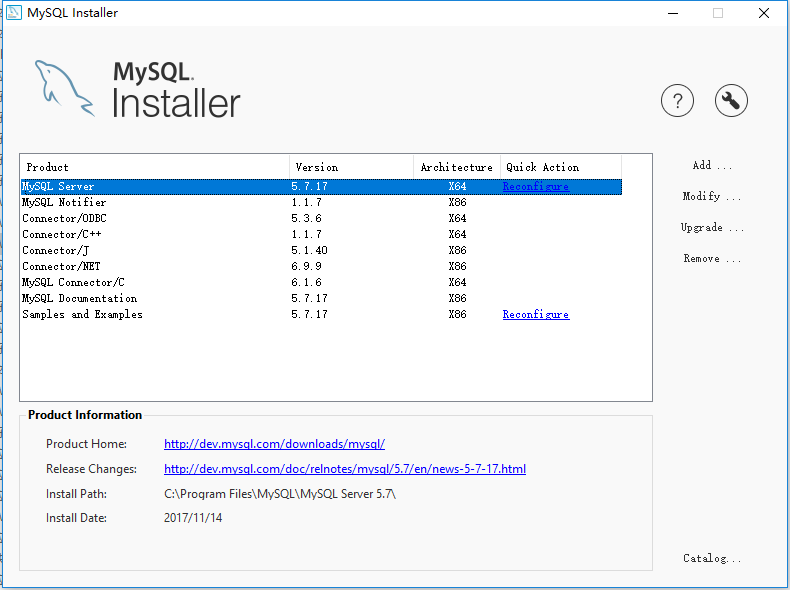
2018年2月13日星期二 22:00

MySQL安装目录



**章节：**

1. 数据库概述及安装
2. MySQL的安装及登录
3. 常用命令
4. 演示数据结构
5. 简单的查询
6. 条件查询
7. 数据排序 asc/desc
8. 处理函数
9. select语句总结
10. 连接查询/跨表查询
11. 子查询
12. union合并(相加)集合
13. limit使用
14. 表
15. 存储引擎
16. 事务Transaction
17. 索引（了解）
18. 视图
19. DBA命令
20. 数据库设计三大范式
21. 34题经典SQL试题

**相关概念介绍：**

数据库管理系统 DBMS

数据库/仓库 DB

结构化查询语言 SQL

**系统概述：**

1. 什么是数据库：一个或一组文件，保存了一些符合特定规格的数据，DataBase，数据库软件称为数据库管理系统，DBMS，DataBase Management System，例如：Oralce，MySQL，SQL Server，Sybase，informix、DB2等
2. MySQL发展历史
3. SQL概述：Structured Query Language，结构化查询语言，是一套标准，使用SQL完成和数据库的通信，每个不同的数据库管理系统有不同的特性，有一些特定的SQL特性，大概占10%

**数据库管理系统安装：**

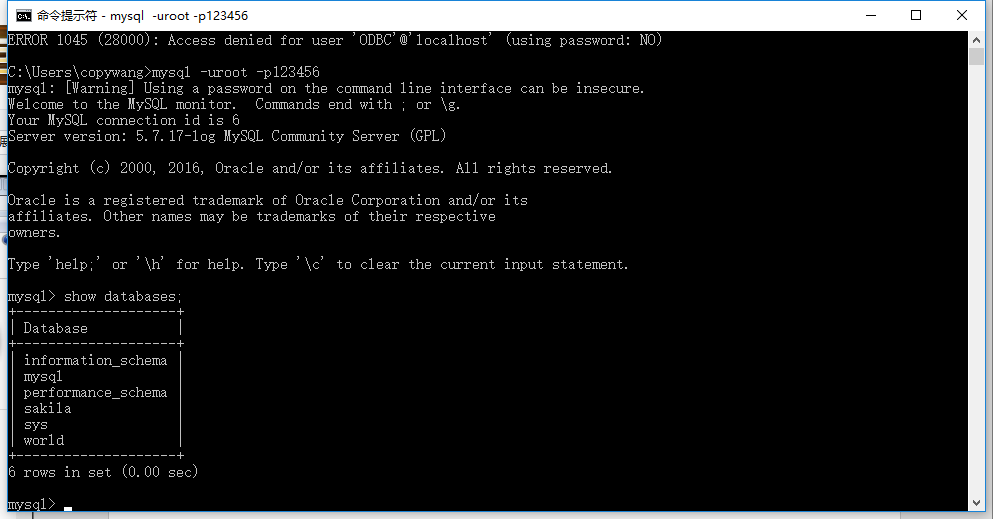
默认端口号3306，会被攻击，改成3366

添加系统环境变量：

**C:\Program Files\MySQL\MySQL Server 5.7\bin**

登录：

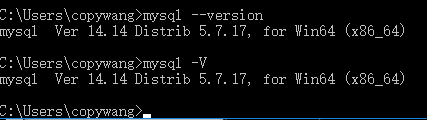
mysql -uroot -p123456



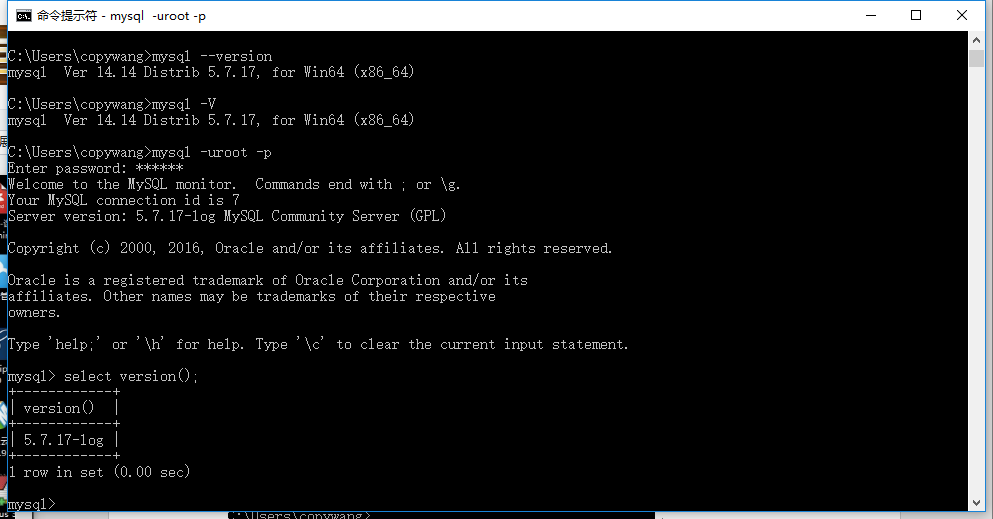
**查询数据库版本：**

mysql --version

mysql -V



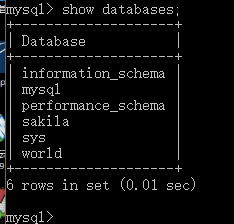
select version();



**常用基本命令：**

查看有哪些数据库：

show databases;



使用某个数据库：

use world;



查看有哪些表：

行表示记录

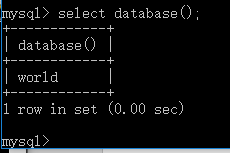
列表示字段：字段名称，类型，长度，字段约束

show tables;



查看自己在哪个数据库下：

select database();



直接查看其他数据库中有哪些表：

show tables from mysql;

退出数据库：

exit;

**SQL语句分类：**

DQL：数据查询语句，select

DML：数据操作语句，insert/delete/update

DDL：数据库定义语句，create/drop/alter

TCL：事务控制语言，commit/rollback

**创建数据库：**

create database bjpowernode;

**导入数据库脚本**

数据库脚本：以.sql后缀结尾的文件

导入数据库脚本：

1. 选定数据库
2. source命令

mysql> use bjpowernode;

Database changed

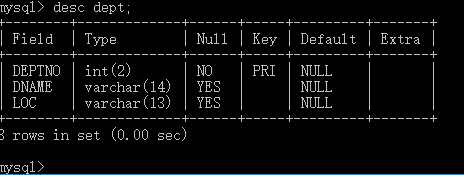
mysql> source C:\Users\copywang\Desktop\bjpowernode.sql

后面不需要分号

**数据库表的介绍：**

·show talbes;

desc dept;

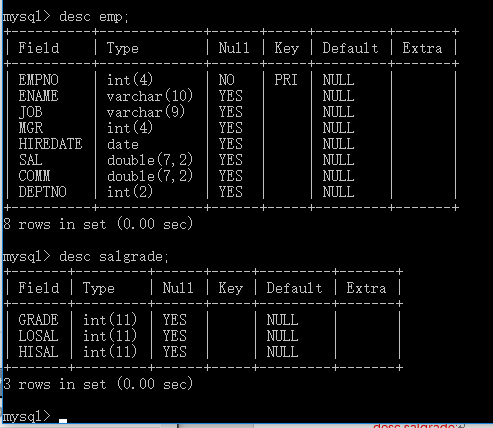


字段，字段类型，字段长度，字段约束

以DEPTNO为例，字段为DEPTNO，字段类型为INT，字段长度为2，约束是字段不能为空且为主键

desc emp;

desc salgrade;



double(7,2) 表示七位数，包含2位小数，从0到99999.99

**1、动力节点\_MySQL\_001\_MySQL数据库管理系统前言**

**2、动力节点\_MySQL\_002\_MySQL数据库管理系统相关概念介绍**

**3、动力节点\_MySQL\_003\_MySQL数据库管理系统概述**

**4、动力节点\_MySQL\_004\_MySQL数据库管理系统安装过程介绍**

**5、动力节点\_MySQL\_005\_MySQL数据库管理系统安装实际操作**

**6、动力节点\_MySQL\_006\_数据库常用命令\_查看数据库版本**

**7、动力节点\_MySQL\_007\_数据库常用命令\_常用基本命令**

**8、动力节点\_MySQL\_008\_SQL语句分类**

**9、动力节点\_MySQL\_009\_导入数据库脚本**

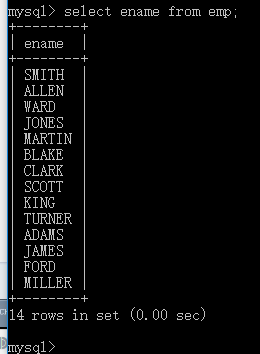
**10、动力节点\_MySQL\_010\_数据库表的介绍**

**11、动力节点\_MySQL\_011\_简单查询\_单个字段和多个字段查询**

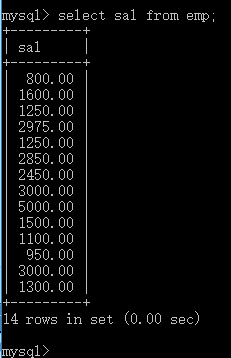
**简单SQL查询：**

单个字段和多个字段

select ename from emp;



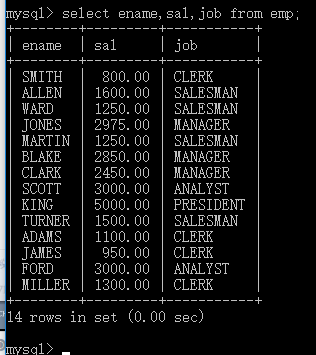
select sal from emp;



select ename,sal from emp;



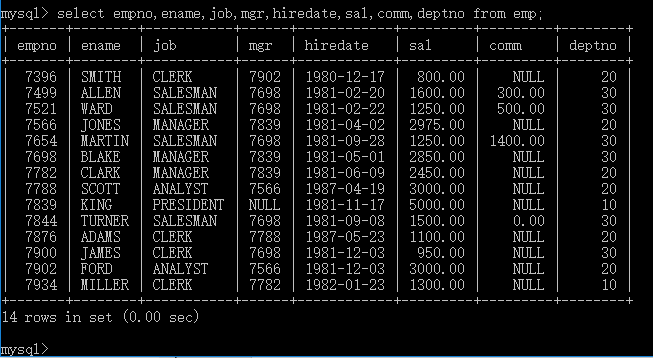
select ename,sal,job from emp;



查询语句不会修改数据库中的数据，只适用于显示。

**12、动力节点\_MySQL\_012\_简单查询\_查询全部字段**

select empno,ename,job,mgr,hiredate,sal,comm,deptno from emp;

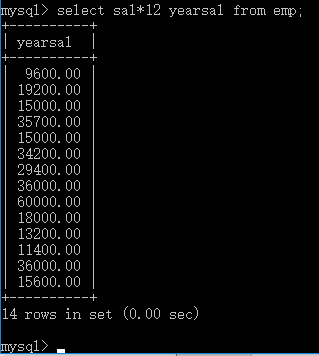


或者使用（不建议）：效率比前一个方法要低，因为要把\*转成所有具体字段，而且语义不明确

select \* from emp;

**13、动力节点\_MySQL\_013\_简单查询\_查询员工年薪及给字段起别命**

select sal\*12 yearsal from emp;

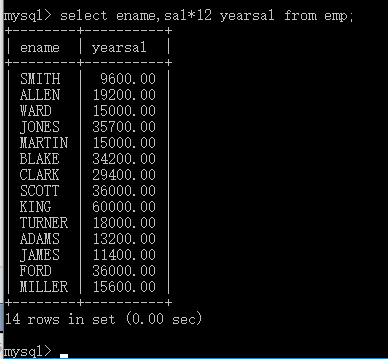


select ename,sal\*12 as yearsal from emp;

或者：

select ename,sal\*12 yearsal from emp;

不建议省略as



字段名可以进行数学计算，起别名

**14、动力节点\_MySQL\_014\_条件查询介绍**

关键字：where

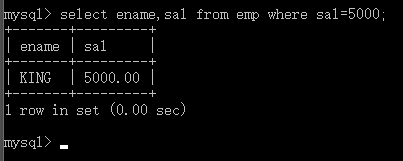
select 字段 from 表名 where 条件;

**15、动力节点\_MySQL\_015\_条件查询\_条件查询支持的运算符介绍**

|  |  |
| --- | --- |
| 运算符 | 说明 |
| = | 等于 |
| <>或!= | 不等于 |
| < | 小于 |
| <= | 小于等于 |
| > | 大于 |
| >= | 大于等于 |
| between ... and ... | 两个值之间，相当于>= and <= |
| is null | 为null（is not null 不为空） |
| and | 并且 |
| or | 或者 |
| in | 包含，相当于多个or（not in 不在这个范围中） |
| not | Not可以取非，用于is和in中 |
| like | Like为模糊查询，支持%或者下划线匹配  %匹配任意个字符  下划线只匹配一个字符 |

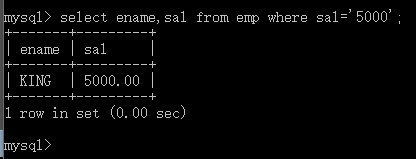
**16、动力节点\_MySQL\_016\_条件查询\_等号操作运算符**

select ename,sal from emp where sal=5000;

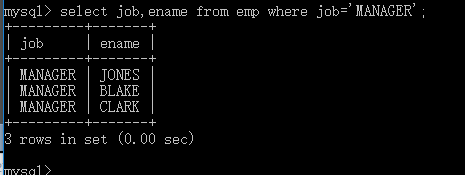


MySQL支持自动类型转换，比如（但是其他数据库不支持，所以不要这么写）：

select ename,sal from emp where sal=’5000’;



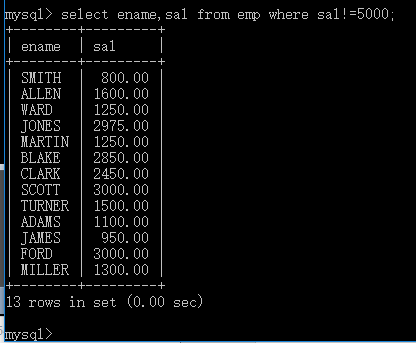
select job,ename from emp where job=’MANAGER’;



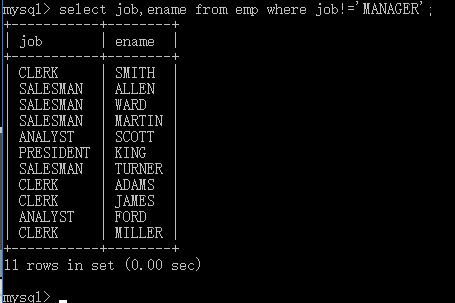
这里使用单引号是SQL的标准写法，MySQL也支持使用双引号，开发中只使用单引号

**17、动力节点\_MySQL\_017\_条件查询\_不等号操作运算符**

select ename,sal from emp where sal!=5000;



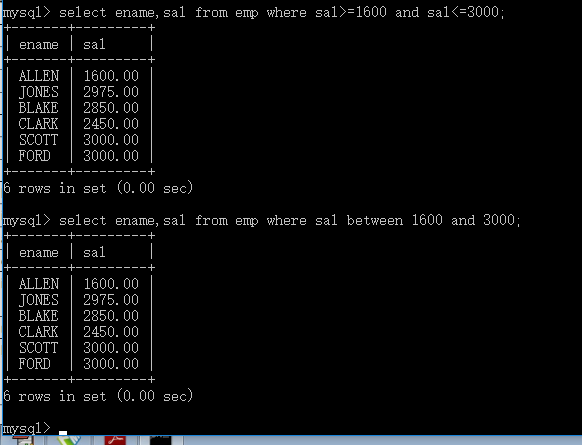
select job,ename from emp where job!=’MANAGER’;



**18、动力节点\_MySQL\_018\_条件查询\_区间操作运算符**

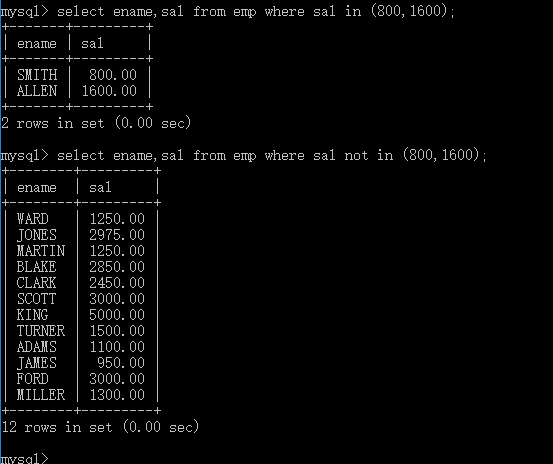
select ename,sal from emp where sal>=1600 and sal<=3000;

select ename,sal from emp where sal between 1600 and 3000;



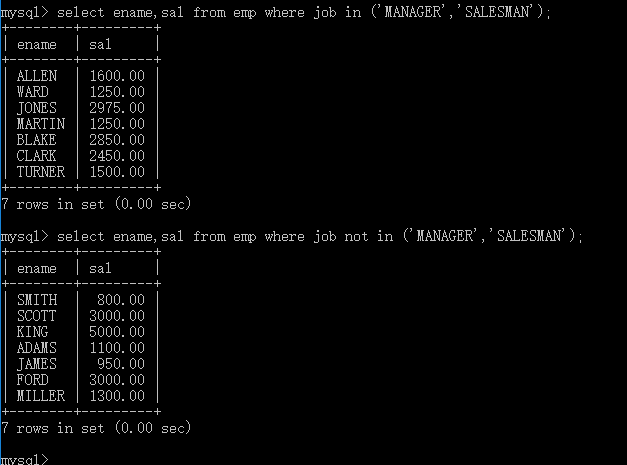
select ename,sal from emp where sal in (800,1600);

select ename,sal from emp where sal not in (800,1600);



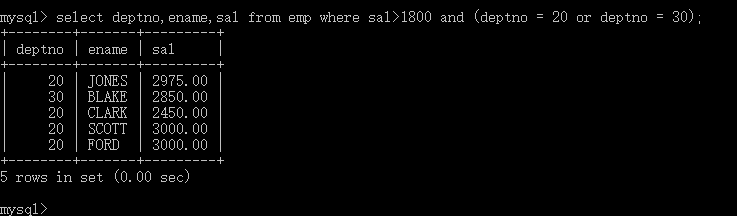
select ename,sal from emp where job in (‘MANAGER’,SALESMAN’);

select ename,sal from emp where job not in (‘MANAGER’,SALESMAN’);



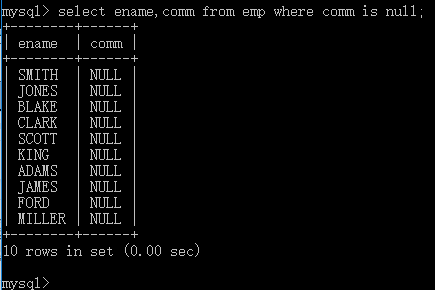
**19、动力节点\_MySQL\_019\_条件查询\_and优先级高于or**

select deptno,ename,sal from emp where sal>1800 and (deptno = 20 or deptno = 30);



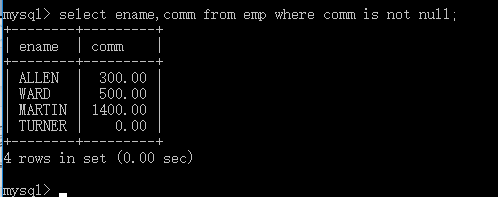
**20、动力节点\_MySQL\_020\_条件查询\_is运算符**

select ename,comm from emp where comm is null;



**null不是数字，是空值，不能进行数学计算，因此不能用=**号

select ename,comm from emp where comm is not null;



select ename,comm from emp where comm=0;

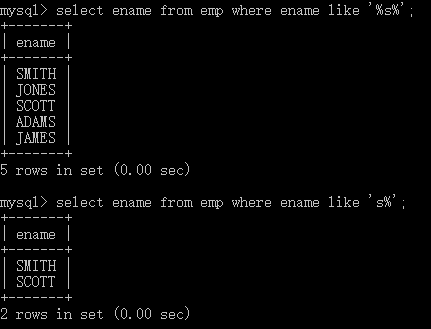
**21、动力节点\_MySQL\_021\_条件查询\_like模糊查询**

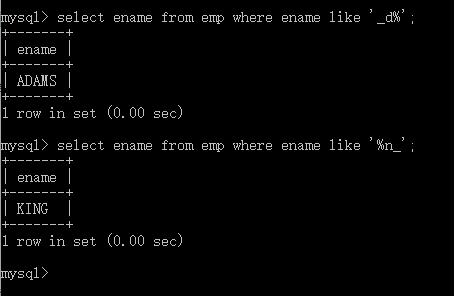
select ename from emp where ename like ‘%s%’;

select ename from emp where ename like ‘s%’;

select ename from emp where ename like ‘\_d%’;

select ename from emp where ename like ‘%n\_’;





**22、动力节点\_MySQL\_022\_数据排序概述**

单一字段排序：order by

默认是是asc

降序是desc

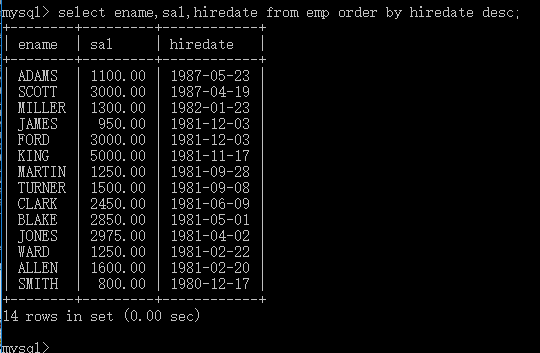
order by 放在where之后

**23、动力节点\_MySQL\_023\_数据排序\_通过字段排序**

select ename,sal from emp order by sal;

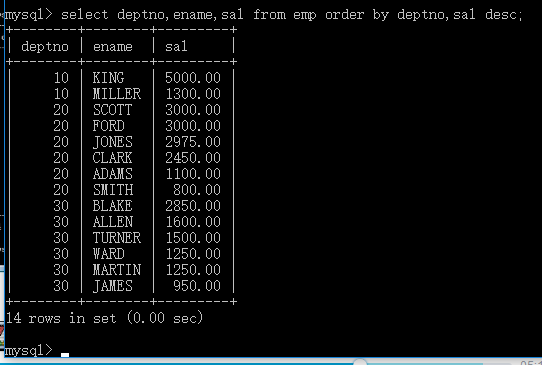
select ename,sal from emp order by sal desc;

select ename,sal,hiredate from emp order by hiredate desc;



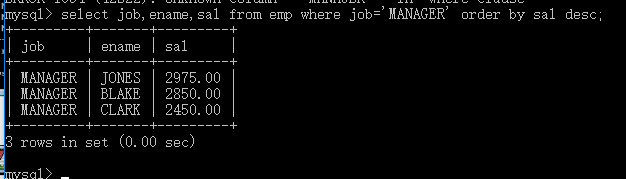
多个字段排序

select deptno,ename,sal from emp order by deptno,sal desc;



多个字段排序的时候，首先按照第一个字段排序，如果第一个字段相同，那么按照第二个字段再进行排序，以此类推

select job,ename,sal from emp where job=’MANAGER’ order by sal desc;

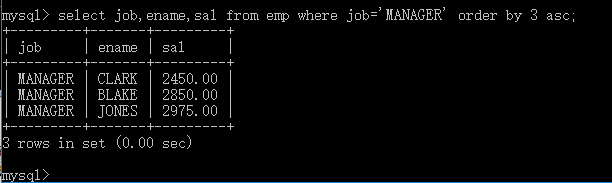


有条件查询的时候，order by放在where之后

**24、动力节点\_MySQL\_024\_数据排序\_通过字段下标排序**

字段下标从1开始，这里的下标指的是你查询显示的列的下边，比如job,ename,sal，那么job就是1，sal是3

select job,ename,sal from emp where job=’MANAGER’ order by 3 asc;



**25、动力节点\_MySQL\_025\_回顾上午内容**

复习回顾从01到24的内容

**26、动力节点\_MySQL\_026\_数据处理函数\_单行处理函数概述**

单行处理函数

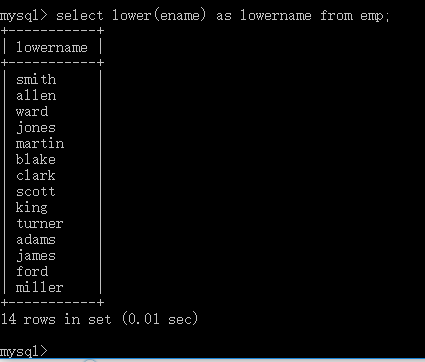
|  |  |
| --- | --- |
| lower | 转换小写 |
| upper | 转换大写 |
| substr | 截取子串（substr(被截取的字符串，起始下标，截取的长度） |
| length | 取长度 |
| trim | 去空格 |
| str\_to\_date | 将字符串转换成日期 |
| Date\_format | 格式化日期 |
| Format | 设置千分位 |
| Round | 四舍五入 |
| Rand() | 生成随机数 |
| Ifnull | 可以把null转换成一个具体值 |

注意：数据处理函数是该数据本身特有的，有些函数可能在其他数据库不起作用

单行的意思：处理一行输出一行

**27、动力节点\_MySQL\_027\_数据处理函数\_单行处理函数\_转换大小写函数**

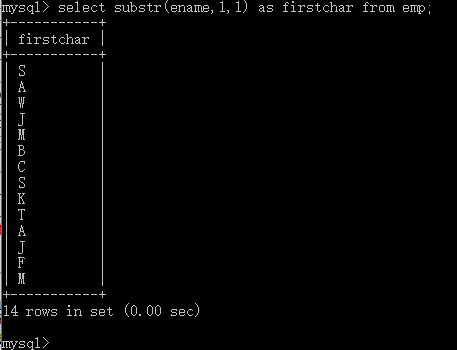
select lower(ename) as lowername from emp;



select upper(ename) as uppername from emp;

**28、动力节点\_MySQL\_028\_数据处理函数\_单行处理函数\_substr截取字符串函数**

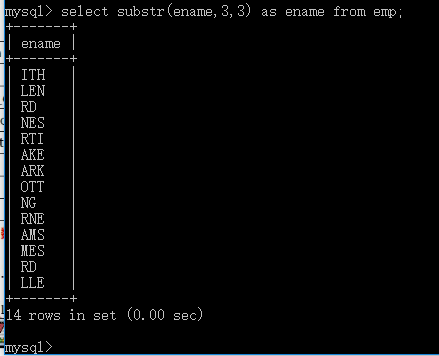
select substr(ename,1,1) as firstchar from emp;



起始下标是从1开始的！

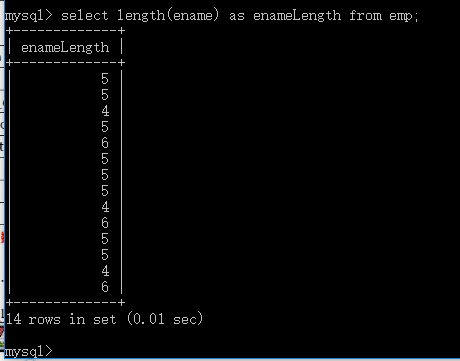
取3,4,5个字符：

select substr(ename,3,3) as ename from emp;



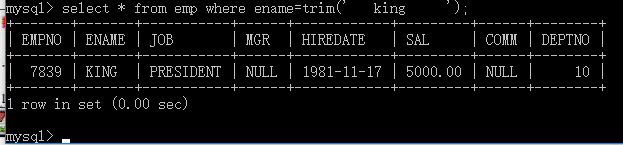
**29、动力节点\_MySQL\_029\_数据处理函数\_单行处理函数\_length和trim函数**

select length(ename) as enameLength from emp;



去掉前后空格：一般在用户输入的时候，查询条件里面会用到（MySQL会自动去掉后面的空格）

select \* from emp where ename=trim(‘ king ‘);



**30、动力节点\_MySQL\_030\_数据处理函数\_单行处理函数\_round四舍五入函数**

select round(123.56);

select round(123.56,0);

上面两个输出都是124

select round(123.56,1);

输出是123.6

select round(123.56,-1);

输出是120

**31、动力节点\_MySQL\_031\_数据处理函数\_单行处理函数\_rand随机数函数**

select rand();

输出0到1闭区间的随机数

select round(rand()\*122,0);

输出0到122闭区间的随机数

**32、动力节点\_MySQL\_032\_数据处理函数\_单行处理函数\_case\_when\_then\_else\_end函数**

匹配工作岗位，为MANAGER时，薪水上调10%，为SALESMAN的时候，薪水上调50%

select job,ename,

(case job

when ‘MANAGER’ then sal\*1.1

when ‘SALESMAN’ then sal\*1.5

else sal

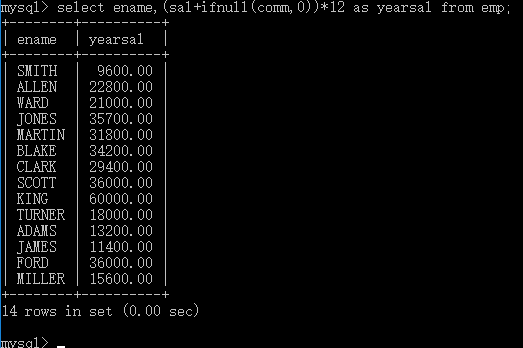
end) as newsal from emp;



**33、动力节点\_MySQL\_033\_数据处理函数\_单行处理函数\_ifnull空值处理函数**

select ename,(sal+ifnull(comm,0))\*12 as yearsal from emp;

统计计算了补贴的所有员工年收入，如果没有补贴，则comm为0



**34、动力节点\_MySQL\_034\_数据处理函数\_单行处理函数\_str\_to\_date函数**

用法：str\_to\_date(‘日期字符串’,’日期格式’)

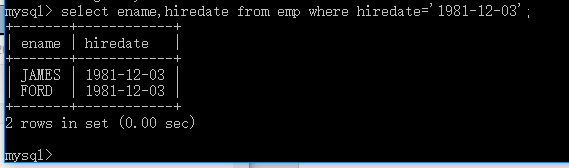
日期格式包括：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 格式符 | 功能 |  | 格式符 | 功能 |
| 1 | %Y | 4位年份 |  | %y | 2位年份 |
| 2 | %m | 月，01...12 |  | %c | 月,1...12 |
| 3 | %d | 日 |  |  |  |
| 4 | %H | 24小时制 |  | %h | 12小时制 |
| 5 | %i | 分钟，格式00...59 |  |  |  |
| 6 | %S或者%s | 秒，格式00...59 |  |  |  |

查询1981-12-03入职的员工

1.自动类型转换

select ename,hiredate from emp wher hiredate=’1981-12-03’;

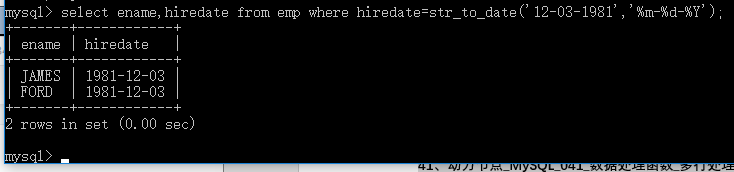


2.标准格式

MySQL默认的日期格式为%y-%m-%d

下面这个例子把varchar转换成DATE：

select ename,hiredate from emp where hiredate = str\_to\_date(’12-03-1981’,’%m-%d-%Y’);



另外一种使用方式在insert中

**35、动力节点\_MySQL\_035\_数据处理函数\_单行处理函数\_str\_to\_date函数\_总结**

1、日期是数据库本身的特使，也是数据库本身机制中的一个重要内容；

2、每一个数据库处理日期采用的机制都不一样，所以在实际开发中将日期字段定义为DATE类型的情况很少

3、如果使用日期类型，java程序则不能通用，实际开发中会使用“日期字符串”来表示日期

**例子：**

1. 创建t\_student表，插入含有日期的数据

create table t\_student(

id int(10),

name varchar(32),

birth date);

1. 插入数据 1980-01-18

insert into t\_student(id,name,birth) values (1,’jack’,’1980-01-18’);

执行成功，因为格式和MySQL数据库默认的日期格式相同，会自动类型转换

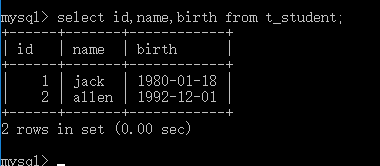
1. 插入数据 01-18-1980

insert into t\_student(id,name,birth) values (1,’jack’,’01-18-1980’);

插入失败

正确写法：

insert into t\_student(id,name,birth) values (1,’jack’,str\_to\_date(’01-18-1980’,’%m-%d-%Y’));



**36、动力节点\_MySQL\_036\_数据处理函数\_单行处理函数\_dateformat函数**

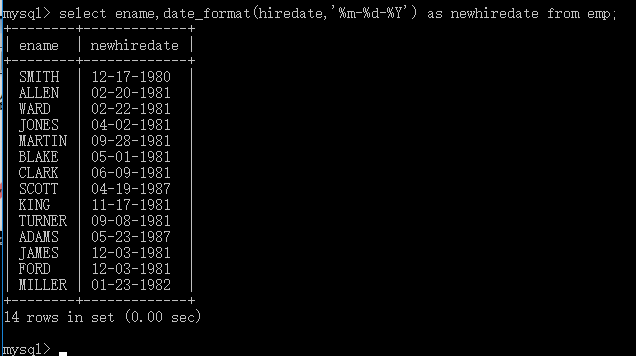
日期类型转换成特定格式的日期字符串

data\_format(日期类型数据，’日期格式’);

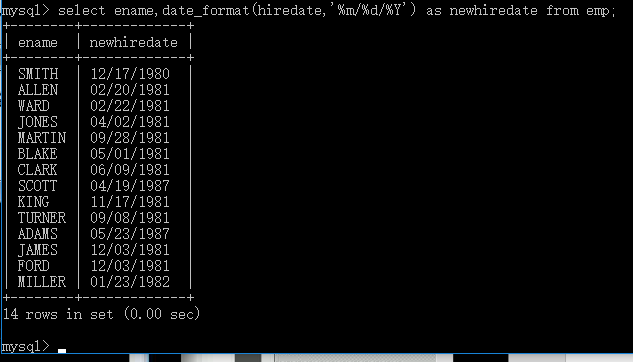
例子：

1. 查询员工的入职日期

select ename,data\_format(hiredate,’%m-%d-%Y’) as newhiredate from emp;



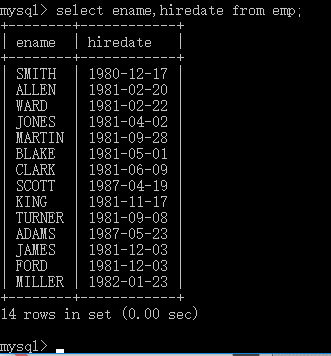
select ename,data\_format(hiredate,’%m/%d/%Y’) as newhiredate from emp;



MySQL日期默认格式示例：

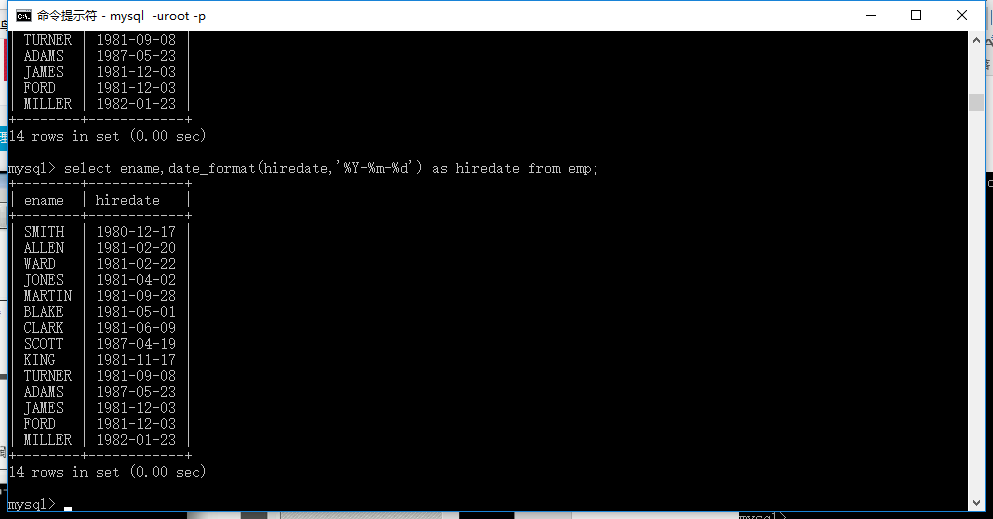
1. hiredate自动转换成varchar类型，采用年月日格式

select ename,hiredate from emp;



1. 使用date\_format格式

select ename,date\_format(hiredate,’%Y-%m-%d’) as hiredate from emp;



date\_format函数主要用于数据库查询操作时，实际工作中，客户需要日期以特定格式展示的时候，需要使用该函数。

**37、动力节点\_MySQL\_037\_数据处理函数\_多行处理函数概述**

分组函数/聚合函数/多行处理函数

常用以下几种：

|  |  |
| --- | --- |
| sum | 求和 |
| avg | 平均 |
| max | 最大值 |
| min | 最小值 |
| count | 记录数 |

单行函数式一行输入对应一行输出

多行处理函数是多汗输入对应一行输出

**注意：**

1. 分组函数会自动忽略空值，不需要手动增加where条件排除空值；
2. 分组函数不能直接使用在where关键字后

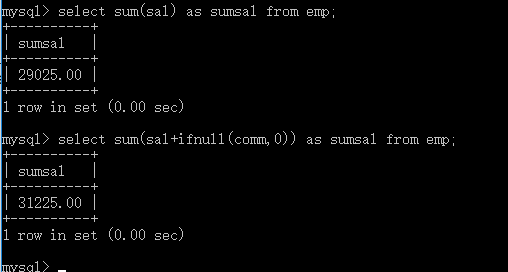
**38、动力节点\_MySQL\_038\_数据处理函数\_多行处理函数\_sum求和函数**

**例子：**

**薪水总和：**

select sum(sal) as sumsal from emp;

select sum(sal+ifnull(comm,0)) as sumsal from emp;



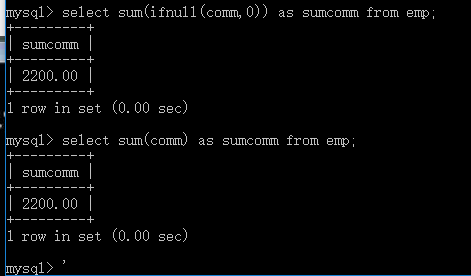
**每个月的补助总和：**

select sum(ifnull(comm,0)) as sumcomm from emp;

**或者**

select sum(comm) as sumcomm from emp;

空值无需做处理，但是还是建议加上ifnull

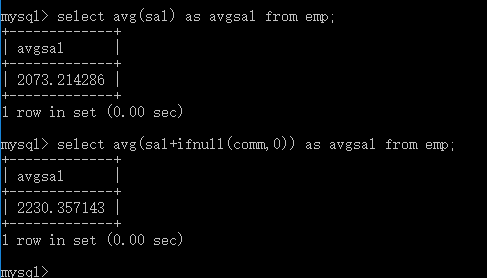


**39、动力节点\_MySQL\_039\_数据处理函数\_多行处理函数\_avg求平均值函数**

**求工资平均值：**

select avg(sal) as avgsal from emp;

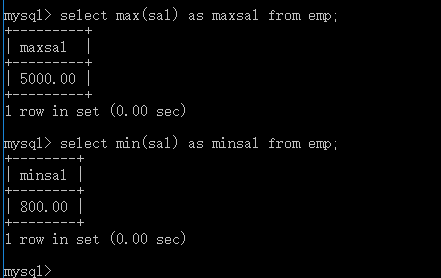
select avg(sal+ifnull(comm,0)) as avgsal from avg;



**40、动力节点\_MySQL\_040\_数据处理函数\_多行处理函数\_max和min求最大最小值函数**

select max(sal) as maxsal from emp;

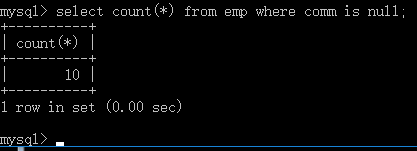
select min(sal) as minsal from emp;



**41、动力节点\_MySQL\_041\_数据处理函数\_多行处理函数\_count取得记录数函数**

**select count(\*) from emp where comm is null;**

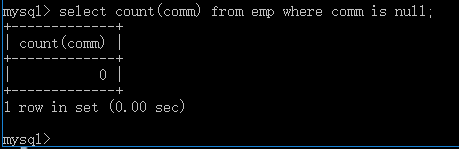
**count(\*)表示多行记录**



**如果写成：**

**select count(comm) from emp where comm is null;**

**结果为0，因为count()会自动忽略空值**

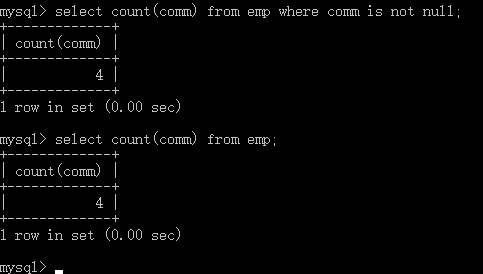


**count(\*)统计的是结果集的总条数，count(字段名)统计的是该字段值不为null的总条数**

**select count(comm) from emp where comm is not null;**

**等价于**

**select count(comm) from emp;**



**42、动力节点\_MySQL\_042\_数据处理函数\_distinct去除重复记录概述**

distinct：将查询结果中的某一个字段的重复记录去除

用法：distinct 字段名或distinct 字段名1，字段名2 ...

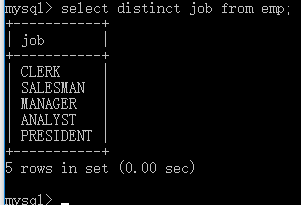
distinct A 区处于字段名A相同的记录

distinct A,B 去除与字段名A和字段名B相同的记录

**注意：DISTINCT只能出现在所有字段的最前面，后面接多个字段为多字段联合去重**

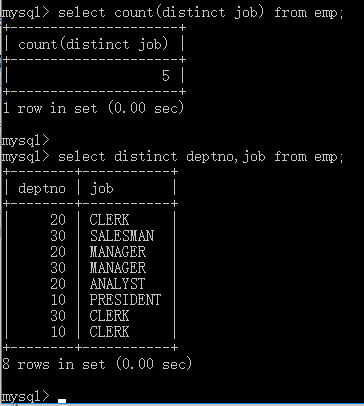
**43、动力节点\_MySQL\_043\_数据处理函数\_distinct去除重复记录\_通过例子讲解**

select distinct job from emp;



select count(distinct job) from emp;

select distinct deptno,job from emp;



**44、动力节点\_MySQL\_044\_数据处理函数\_分组查询\_01**

**group by**

**语句格式：**

select

字段

from

表名

group by

字段;

**例子：**

select

job,max(sal) as maxsal

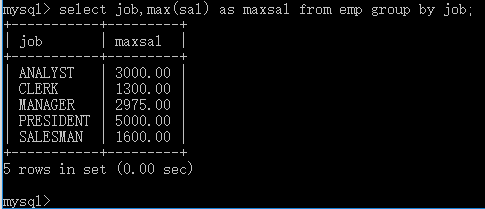
from

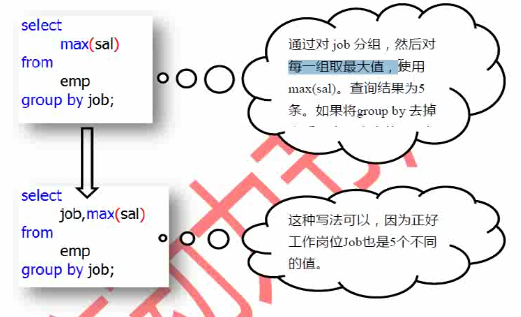
emp

group by

job;

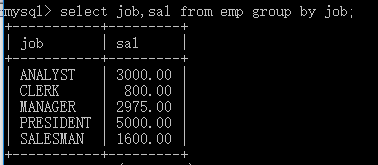
查找每个职业中，最高的薪水





**select job,sal from emp group by job;**

**这种语句，只取拿到的第一个记录**



**45、动力节点\_MySQL\_045\_数据处理函数\_分组查询\_02**

**例子：**

select

job,ename,max(sal) as maxsal

from

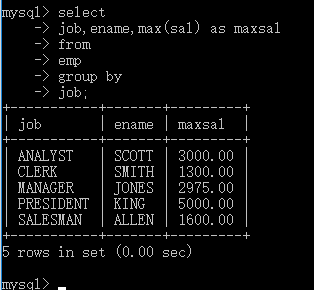
emp

group by

job;

查找每个职业中，最高的薪水，同时会取出每个职业的第一个名字（没有意义，SCOTT不一定是3000收入的人）

在ORACLE中会直接报错



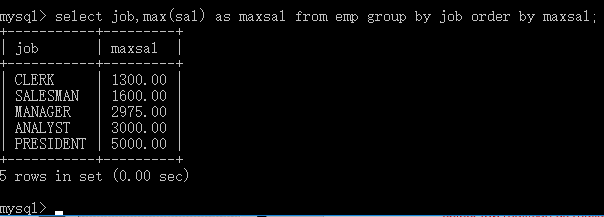
**46、动力节点\_MySQL\_046\_数据处理函数\_分组查询\_03**

**例子：**

**计算每个工作岗位的最高薪水，并且按照由低到高排序**

**先按照工作分组，然后排序**

select job,max(sal) as maxsal from emp group by job order by maxsal;



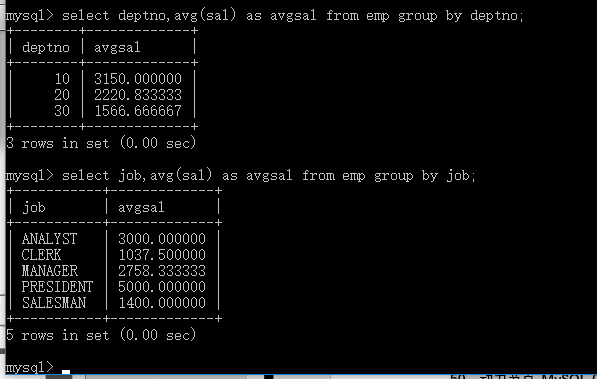
例子：

计算每个部门的平均薪水

按照部门编号分组，对每一组薪水求平均值

select deptno,avg(sal) as avgsal from emp group by deptno;

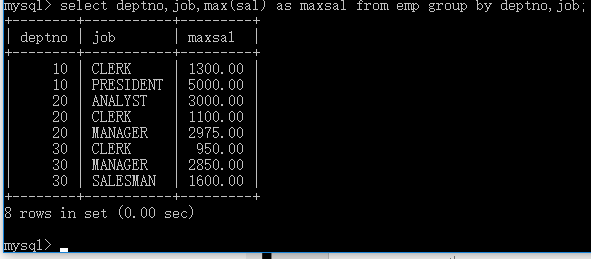
select job,avg(sal) as avgsal from emp group by job;



例子：

计算出不同部门不同岗位的最高薪水

select deptno,job,max(sal) as maxsal from emp group by deptno,job;



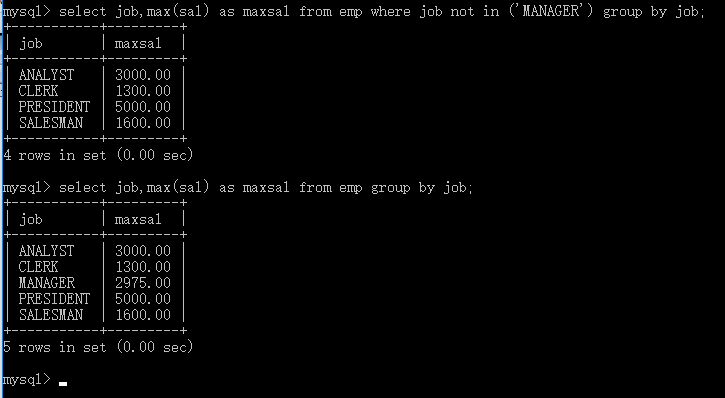
用法和distinct类似，多个字段联合

例子：

找出每个工作岗位的最高薪水，除了MANAGER之外

select job,max(sal) as maxsal from emp where job not in (‘MANAGER’) group by job;

select job,max(sal) as maxsal from emp where job != ‘MANAGER’ group by job;

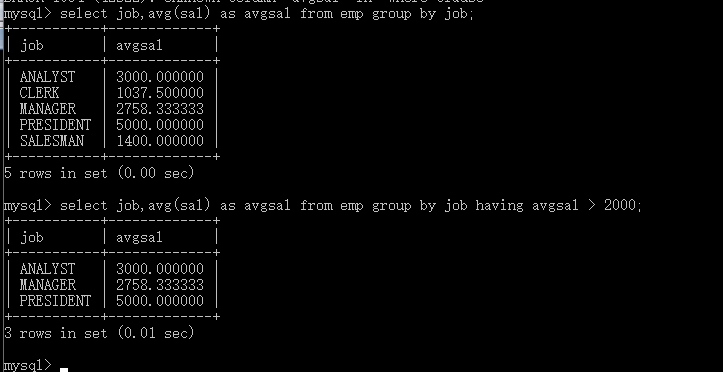


**47、动力节点\_MySQL\_047\_数据处理函数\_having数据过滤用法**

**例子：找出每个工作岗位的平均薪水，要求显示平均薪水大于2000的**

select job,avg(sal) as avgsal from emp group by job;

select job,avg(sal) as avgsal from emp group by job having avgsal > 2000;



**having 必须和 group by搭配使用，分组后再筛选**

**48、动力节点\_MySQL\_048\_数据处理函数\_having与where数据过滤用法区别**

where 在 group by之前

having 在 group by之后

**49、动力节点\_MySQL\_049\_select语句总结**

**一个完整的SQL语句如下：**

select

xxx

from

xxx

where

xxx

group by

xxx

having

xxx

order by

xxx

1. from 将硬盘上的表文件加载到内存
2. where 将符合条件的数据行摘取出来，生成一张临时表
3. group by 根据列中的数据种类，将当前临时表划分成若干个新的临时表
4. having 可以过滤掉group by生成的不符合条件的临时表
5. select 对当前表进行整列读取
6. order by 对select生成的临时表，进行重新排序，生成新的临时表
7. limit 对最终生成的临时表的数据行进行截取

**50、动力节点\_MySQL\_050\_课程回顾**

//复习回顾

**51、动力节点\_MySQL\_051\_跨表查询\_迪卡尔积现象**

**实际开发中，数据是存储在多张表中，表与表之间存在联系，检索数据的时候需要多表联合检索，称为跨表查询**

查询员工名称和部门名称

ename -> emp

dname -> dept

select ename,dname from emp,dept;

这个语句没有加条件限制，因此结果是两个表的乘积

比如表A

1

2

3

表B

4

5

6

结果是

1 4

1 5

1 6

2 4

2 5

2 6

3 4

3 5

3 6

称为 笛卡尔积

**52、动力节点\_MySQL\_052\_跨表查询\_分类介绍**

跨表查询：

年代划分：

SQL92

SQL99

连接方式划分：

内连接

等值连接

非等值连接

自连接

外连接

左外连接

右外连接

全连接

**53、动力节点\_MySQL\_053\_跨表查询\_根据年代分类\_SQL92语法**

例子：

显示每个员工信息，并显示所属部门名称

select ename,dname from emp as e,dept as d where e.deptno = d.deptno;



这个语句的匹配次数依然是笛卡尔积

**54、动力节点\_MySQL\_054\_跨表查询\_根据年代分类\_SQL99语法**

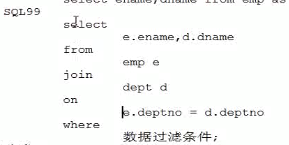
例子：

显示每个员工信息，并显示所属部门名称

select e.ename,d.dname from emp as e join dept as d on e.deptno = d.deptno;



格式



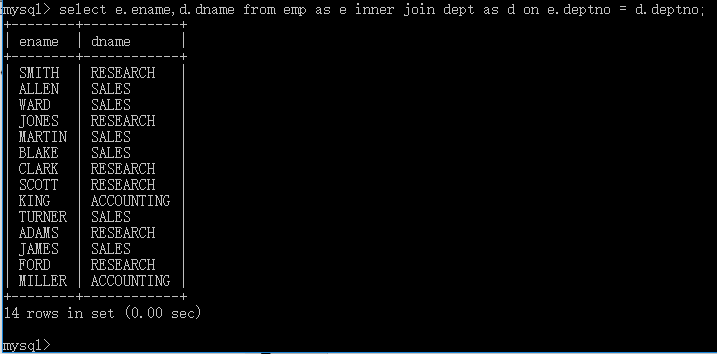
**开发使用SQL99，不使用SQL92**

**55、动力节点\_MySQL\_055\_跨表查询\_根据连接方式分类\_内连接\_等值连接**

例子：

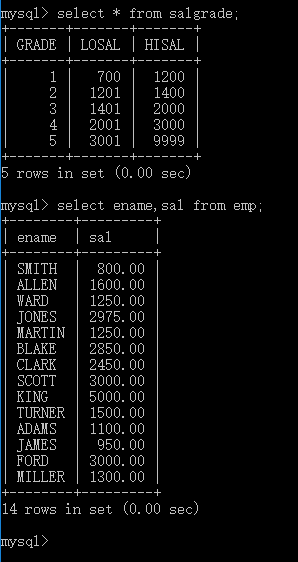
显示每个员工信息，并显示所属部门名称

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

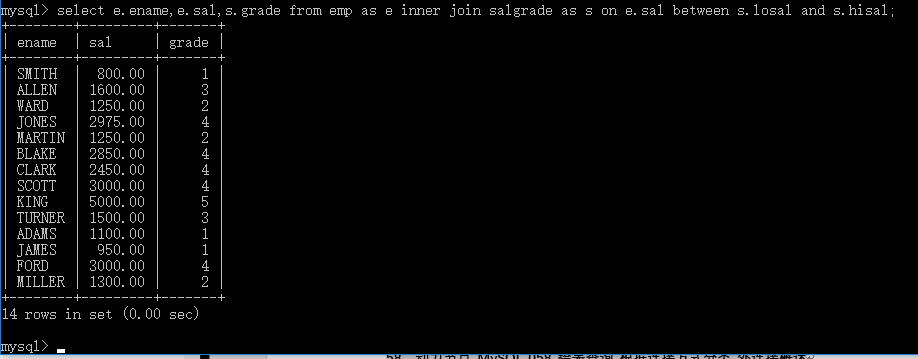


**56、动力节点\_MySQL\_056\_跨表查询\_根据连接方式分类\_内连接\_非等值连接**

查询员工薪水对应的薪水等级



select e.ename,e.sal,s.grade from emp as e inner join salgrade as s on e.sal between s.losal and s.hisal;

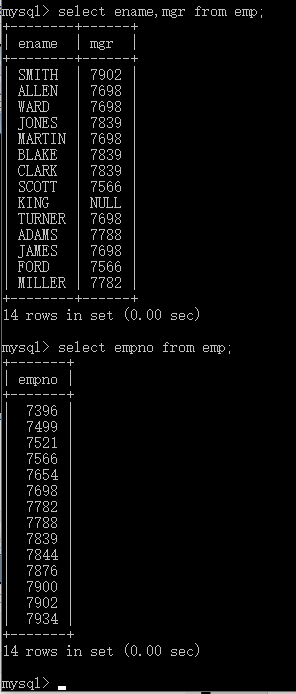


**总共执行了14\*5=70次**

**57、动力节点\_MySQL\_057\_跨表查询\_根据连接方式分类\_内连接\_自连接**

**例子：**

**查询员工所对应的领导名称，显示员工名称和领导名称**



select a.ename as empename,b.ename as mgrename from emp a join emp b on a.mgr = b.empno;



这个语句相当于把emp表做了两次别名处理，一张表看成两张表

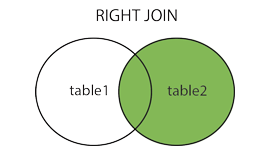
**58、动力节点\_MySQL\_058\_跨表查询\_根据连接方式分类\_外连接概述**

A表和B表能够完全匹配的记录查询出来之外，将其中一张表的记录无条件的完全查询出来，对方表没有匹配的记录时，会自动模拟出null值与之匹配

**注意：外连接的查询结果条数>=内连接的查询结果条数**

**59、动力节点\_MySQL\_059\_跨表查询\_根据连接方式分类\_外连接\_右外连接**

包含右边表的全部行（不管左边表是否存在与它们匹配的行），以及左边表中全部匹配的行

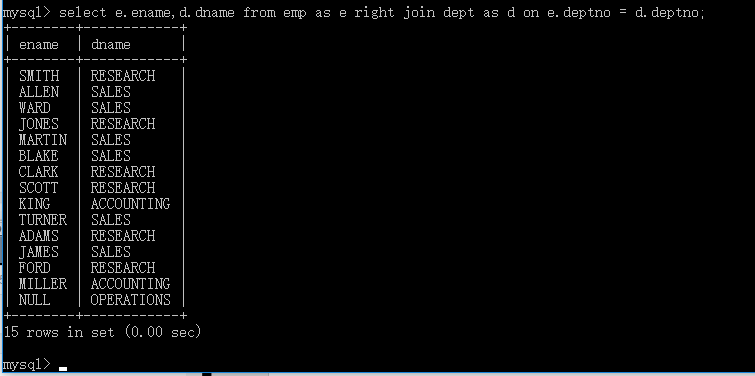


**例子：**

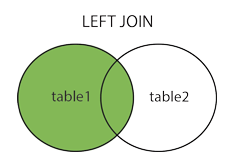
select e.ename,d.dname from emp e right join dept d on e.deptno = d.deptno;

等价于：省略了outer

select e.ename,d.dname from emp as e right outer join dept as d on e.deptno = d.deptno;

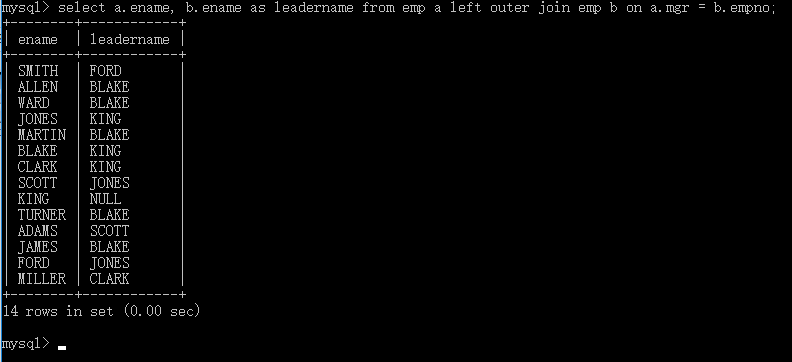


**60、动力节点\_MySQL\_060\_跨表查询\_根据连接方式分类\_外连接\_左外连接**

包含左边表的全部行（不管右边表是否存在与它们匹配的行），以及右边表中全部匹配的行

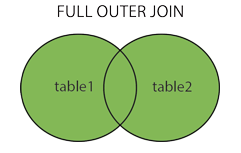
例子：

select a.ename, b.ename as leadername from emp a left outer join emp b on a.mgr = b.empno;



这里KING是没有领导的 显示为NULL

**了解：全连接**



**61、动力节点\_MySQL\_061\_跨表查询\_根据连接方式分类\_通过哪此关键字区分内外连接**

通过 right 或者 left 区分

**62、动力节点\_MySQL\_062\_跨表查询\_多表查询语法介绍**

例子：查询员工部门名称，领导名称和薪水等级

ename -> emp

dname -> dept

grade -> salgrade

select

d.dname,

e.ename,

b.ename as leadername,

s.grade

from

emp e

join

dept d

on

e.deptno = d.deptno

join

emp b

on

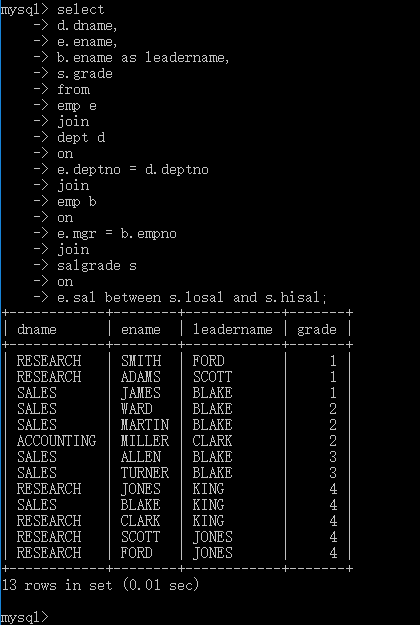
e.mgr = b.empno

join

salgrade s

on

e.sal between s.losal and s.hisal;



**语句分步完成**

**63、动力节点\_MySQL\_063\_子查询\_where关键字后的嵌套查询**

**子查询：select语句嵌套select语句**

**注意：select子句可以出现在select/from/where关键字后面，例如**

**select ... (select) ... [很少时候用，了解即可]**

**from ... (select) ...**

**where ... (select) ...**

**例子：**

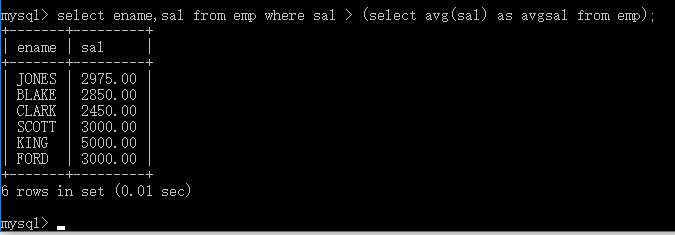
**找出薪水比公司平均薪水高的员工，要求显示员工名和薪水**

错误语句：分组函数不能直接使用在where关键字后面

select ename,sal from emp where sal > avg(sal);

**正确语句：**

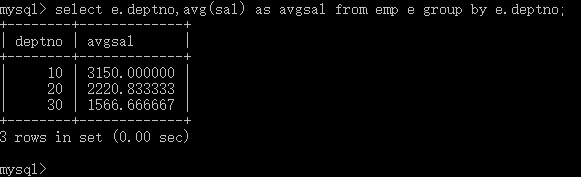
select ename,sal from emp where sal > (select avg(sal) as avgsal from emp);



**64、动力节点\_MySQL\_064\_子查询\_from关键字后的嵌套查询**

**例子：找出每个部门的平均薪水，并且要求显示平均薪水的薪水等级**

select e.deptno,avg(sal) as avgsal from emp e group by e.deptno;



把上面作为临时表t

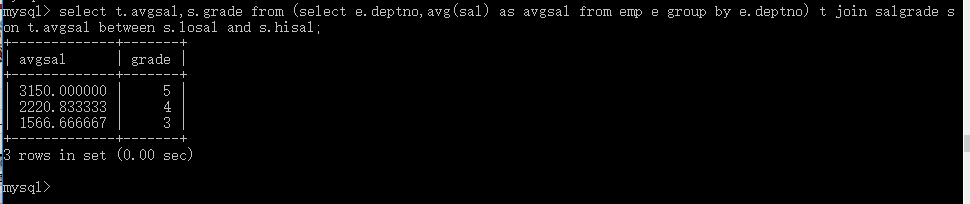
select e.deptno,avg(sal) as avgsal from emp e group by e.deptno;

select t.avgsal,s.grade from t join salgarde s on t.avgsal between s.losal and s.hisal;

最终语句：

select t.avgsal,s.grade from (select e.deptno,avg(sal) as avgsal from emp e group by e.deptno

) t join salgarde s on t.avgsal between s.losal and s.hisal;



**65、动力节点\_MySQL\_065\_UNION合并查询结果集**

**例子：查询出工作岗位是manager和salesman的员工**

select ename,job from emp where job = ‘MANAGER’ or job = ‘SALESMAN’;

select ename,job from emp where job in (‘MANAGER’,’SALESMAN’);

**//如果是not in的话，括号里的值是and的关系**

使用UNION

select ename,job from emp where job = ‘MANAGER’

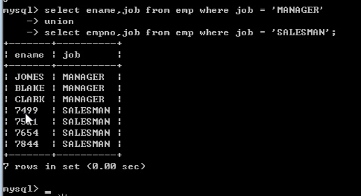
union

select ename,job from emp where job = ‘SALESMAN’;

**注意：**

**两个查询子句的字段要一样，ename/job**

**如果第二个ename改成empno**



**输出结果是一串数字，没有意义**

**66、动力节点\_MySQL\_066\_limit用法概述**

**limit只在MySQL中有效**，获取一个表前几行或者中间某几行数据

用法：

limit起始下标m，长度n

m从0开始，表示第一行记录

n表示从第m+1条开始，取N行记录

下标从0开始，但是长度是从1开始

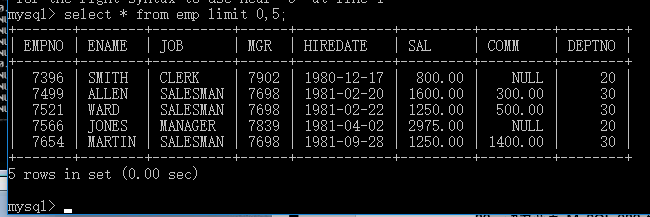
**67、动力节点\_MySQL\_067\_limit实例讲解**

**例子：取得前5个员工信息**

select \* from emp limit 0,5;

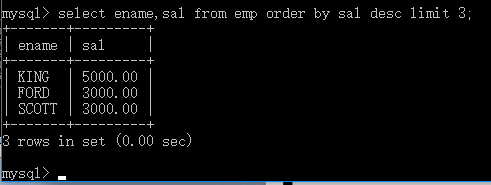
或者

select \* from emp limit 5;



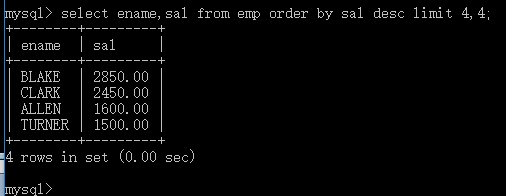
**例子：取薪水前3名的员工**

select ename,sal from emp order by sal desc limit 3;



**例子：取薪水排名第5到第8**

select ename,sal from emp order by sal desc limit 4,4;



**68、动力节点\_MySQL\_068\_limit实现分页**

例子：emp表，每页显示3条记录

页码 pageNo

记录数 pageSize

分析：

第1页 1,2,3 limit 0,3

第2页 4,5,6 limit 3,3

第3页 7,8,9 limit 6,3

limit规律： limit (pageNo-1)\*pageSize,pageSize

**69、动力节点\_MySQL\_069\_表\_表的定义**

表：数据库基本组成单元，行和列组成，行是记录，列是字段，字段包括：字段名称，类型，长度，约束。

**70、动力节点\_MySQL\_070\_表\_创建表的语法**

**属于数据库定义语言：**

create table 表名(

字段1 字段类型(长度) 字段约束,

字段2 字段类型(长度) 字段约束,

字段3 字段类型(长度) 字段约束,

...

字段N 字段类型(长度) 字段约束,

primary key(字段)

)engine=InnoDB default charset=utf8;

commit;

**CREATE TABLE IF NOT EXISTS `runoob\_tbl`(**

**`runoob\_id` INT UNSIGNED AUTO\_INCREMENT,**

**`runoob\_title` VARCHAR(100) NOT NULL,**

**`runoob\_author` VARCHAR(40) NOT NULL,**

**`submission\_date` DATE,**

**PRIMARY KEY ( `runoob\_id` )**

**)ENGINE=InnoDB DEFAULT CHARSET=utf8;**

**实例解析：**

**如果你不想字段为 NULL 可以设置字段的属性为 NOT NULL， 在操作数据库时如果输入该字段的数据为NULL ，就会报错。**

**AUTO\_INCREMENT定义列为自增的属性，一般用于主键，数值会自动加1。**

**PRIMARY KEY关键字用于定义列为主键。 您可以使用多列来定义主键，列间以逗号分隔。**

**ENGINE 设置存储引擎，CHARSET 设置编码。**

**71、动力节点\_MySQL\_071\_表\_MySQL常用数据类型介绍**

|  |  |  |
| --- | --- | --- |
| 数据类型 | 占用字节数 | 描述 |
| char | char(n) | 定长字符串，存储空间大小固定，使用char(2)表示类型或状态 |
| varchar | varchar(n) | 变长字符串，存储空间等于实际数据空间  只包含英文字符的字符串 |
| int | 4个字节 | 表示整型  比如自增ID和表示数量 |
| bigint | 8个字节 | 长整型  比如自增ID，数据量比较大的情况下使用 |
| float | float(有效数字位数，小数位) | 数值型 |
| double | double(有效数字位数，小数位) | 数值型 |
| date | 8个字节 | 表示日期和时间 |
| BLOB |  | 二进制大对象，Binary Large Object，视频图像等，开发中基本不用，只需要存文件地址 |
| CLOB |  | 字符大对象，Character Large Object，开发中很少用，最大存4G+的字符串 |

注意：

VARCHAR和CHAR对比

1. 都是字符串
2. VARCHAR可以根据实际数据长度分配空间，需要进行长度判断，因此执行效率比较低
3. CHAR不需要动态分配空间，执行效率高，可能会导致空间浪费
4. 字段中数据不具备伸缩性的时候，采用CHAR存储，比如（状态值）性别MALE和FEMALE，其他就是用VARCHAR，比如姓名等

**72、动力节点\_MySQL\_072\_表\_实例讲解创建表及删除表**

**例子：创建学生表**

create table t\_student(

no int(4) not null,

name varchar(32) not null,

gender char(1) not null,

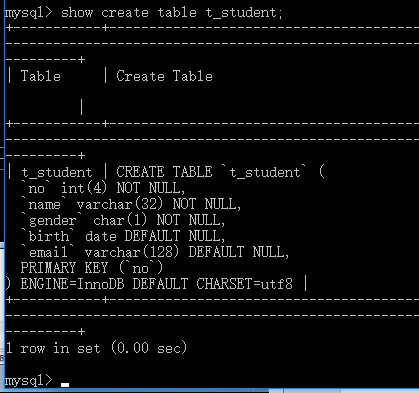
birth date,

email varchar(128),

primary key(no));

**查看建表语句：**

show create table t\_student;



**删除表：**

drop table t\_student;

**如果存在就删除表：**

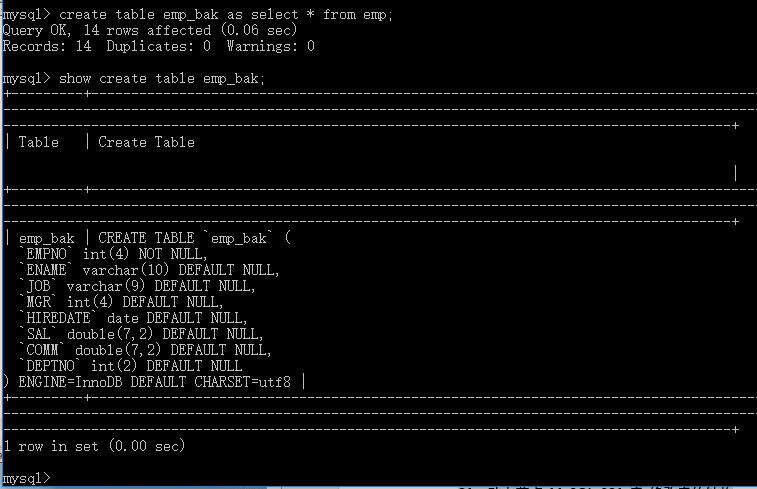
drop table if exists t\_student;

**73、动力节点\_MySQL\_073\_表\_复制表语法**

create table emp\_bak as select \* from emp;

**查看创建表语句：**

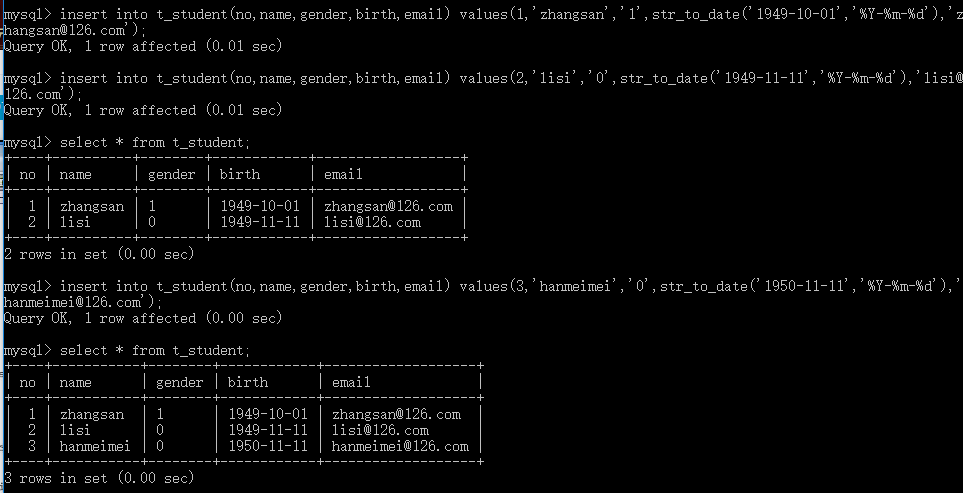
show create table emp\_bak;

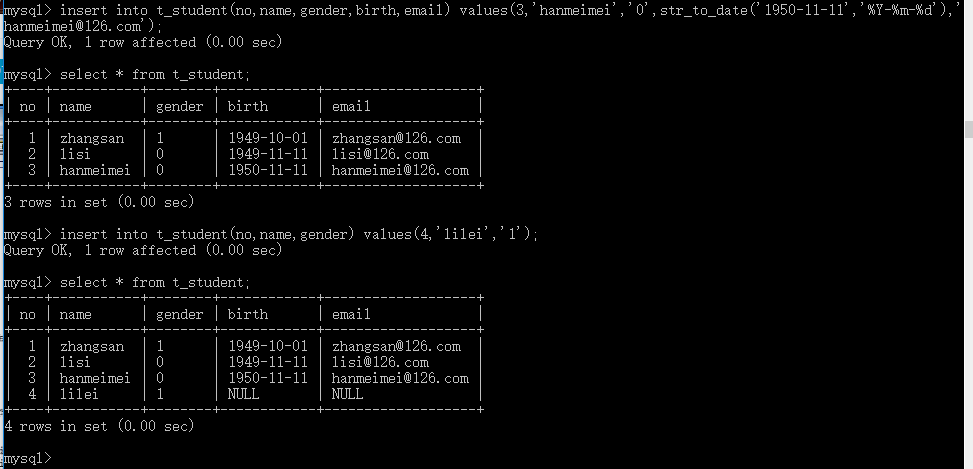


**74、动力节点\_MySQL\_074\_表\_数据操作语言DML语句\_insert语法**

**DML: insert/update/delete**

**insert into 表名(字段1，字段2，...，字段N) values （字段1的值，字段2 的值，...，字段N的值）;**





**75、动力节点\_MySQL\_075\_上午课程回顾**

**//复习回顾**

**76、动力节点\_MySQL\_076\_表\_数据操作语言DML语句\_update语法**

update 表名 set 字段名称 = 字段值，字段名称 = 字段值 where 限制条件；

**必须加限制条件！！否则就是全表修改**

update t\_student set birth = str\_to\_date(‘1951-10-10’,’%Y-%m-%d’), email = ‘lilei@126.com’ where name = ‘lilei’;



**77、动力节点\_MySQL\_077\_表\_数据操作语言DML语句\_delete语法**

**语法：**

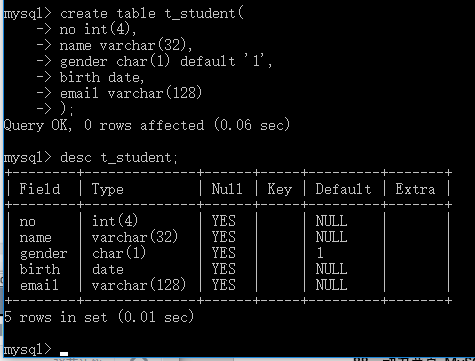
delete from t\_student; //删除所有

delete from t\_student where no = 4; //加限制条件

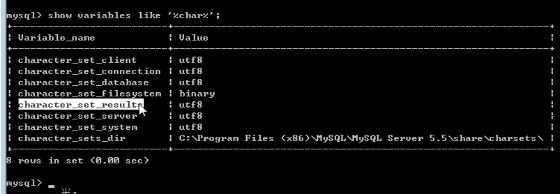
**注意：这属于物理删除，无法恢复**

**78、动力节点\_MySQL\_078\_表\_设置表中字段默认值**

关键字：default



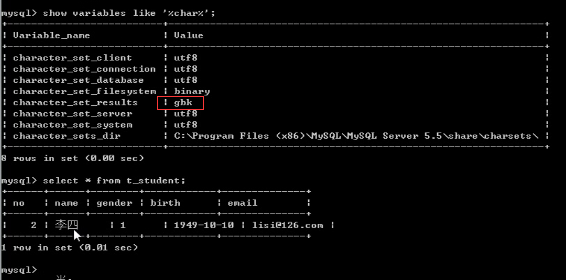
**79、动力节点\_MySQL\_079\_表\_MySQL-Front工具使用介绍**



**字符集改成gbk**



**再查一下**



**80、动力节点\_MySQL\_080\_表\_数据操作语言DML语句\_快速向表中插入数据**

**insert**

insert into emp\_bak select \* from emp where job = 'MANAGER';

select count(\*) from emp\_bak;

注意：快速插入保证字段类型和数量一致

**81、动力节点\_MySQL\_081\_表\_修改表的结构**

**ALTER关键字**

1. **新增：ALTER TABLE 表名 ADD 字段名 字段类型（长度）；**
2. **修改：ALTER TABLE 表名 MODIFY 字段名 字段类型（长度）；**
3. **删除：ALTER TABLE 表名 DROP 字段名；**

drop table if exists t\_student;

create table t\_student(

no int(10),

name varchar(32)

);

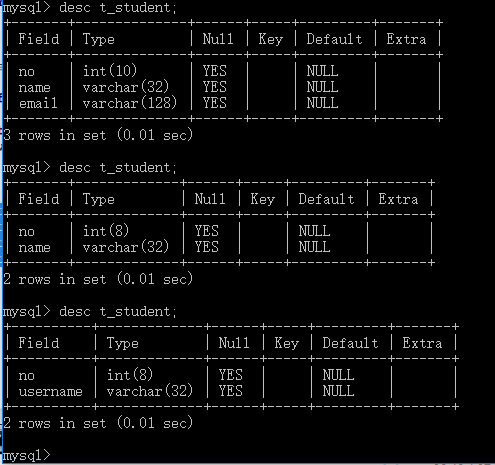
alter table t\_student add email varchar(128);

alter table t\_student modify no int(8);

alter table t\_student drop email;

**不常用的修改字段名称方法：**

alter table t\_student change name username varchar(32);



**82、动力节点\_MySQL\_082\_约束的概述**

目的：保证表中数据的完整和有效

英语：constraint

定义：对表中数据的限制条件

**83、动力节点\_MySQL\_083\_约束\_非空约束**

not null

表明字段必须由具体数据，不能为NULL

**84、动力节点\_MySQL\_084\_约束\_唯一性约束\_列级写法**

unique

表明字段必须不能重复，保持唯一

**列级约束：写在字段后面，保证邮箱唯一性**

create table t\_student(

no int(10),

name varchar(32) not null,

email varchar(128) unique

);

**85、动力节点\_MySQL\_085\_约束\_唯一性约束\_表级写法**

**表级约束：**

create table t\_student(

no int(10),

name varchar(32) not null,

email varchar(128),

unique(email)

);

效果和84一样。

表级支持多个字段联合约束

create table t\_student(

no int(10),

name varchar(32) not null,

email varchar(128),

**unique(name,email)**

);

只有当name和email都一样的时候，才是重复值，只要有一个值不一样，就认为不是重复记录

约束起别名：

create table t\_student(

no int(10),

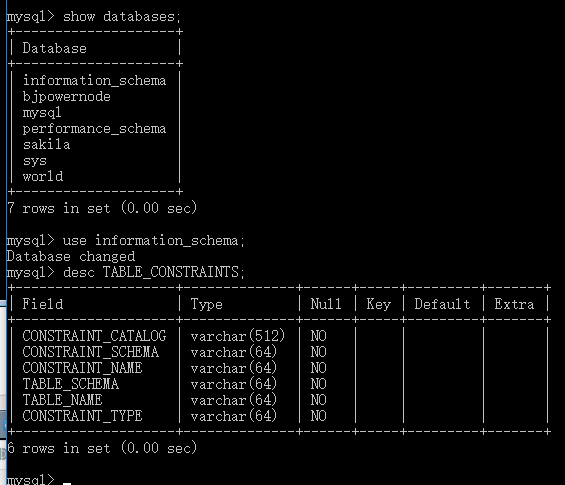
name varchar(32) not null,

email varchar(128),

**constraint t\_user\_name\_email\_unique unique(name,email)**

);

查询约束：



select CONSTRAINT\_NAME from TABLE\_CONSTRAINTS where table\_name ='t\_user';

**86、动力节点\_MySQL\_086\_约束\_非空约束与唯一性约束联合使用**

**例子：手机号，不为空，不能重复**

not null unique

**87、动力节点\_MySQL\_087\_约束\_主键约束概述及相关概念**

primary key 简称PK

例子：身份证

字段为：ID/NAME/GENDER/AGE

ID就是身份证号，作为主键约束，不能重复

**三个术语：**

**主键约束、主键字段、主键值**

**表中某个字段添加主键约束之后，该字段成为主键字段，主键字段中出现的每一个数据都成为主键值。**

**88、动力节点\_MySQL\_088\_约束\_主键约束的作用**

1、添加了主键 primary key 的字段“不能重复也不能为空”，而且会自动添加“索引-index”，提高检索效率。

2、一张表必须有主键，否则这张表就是无效的（数据库设计第一范式），主键值是当行数据的唯一标识，就是表中两行数据完全相同，但是由于主键不同，也可以认为是两行完全不同的数据。

**89、动力节点\_MySQL\_089\_约束\_主键约束\_根据个数分类\_单一主键和复合主键**

单一主键：给一个字段添加主键约束；

列级： id int(10) primary key

表级： 在创建数据库表的时候，最后一个加primary key(id)

上面两种写法效果一样

复合主键：

constraint t\_user\_id\_name\_pk primary key(id,name)

只有当id和name都一致的时候，才认为是重复的数据

**无论是单一主键还是复合主键，一张表中有且只能有一个主键约束**

**90、动力节点\_MySQL\_090\_约束\_主键约束\_根据性质分类\_自然主键和业务主键**

按照业务性质分类：

自然主键：主键值是一个自然数，与业务没有任何关系

业务主键：主键值和表中业务紧密相关，如果业务发生变化，则主键也会受到影响，所以业务主键使用较少，大多数情况下使用自然主键，比如身份证号位数扩展

**91、动力节点\_MySQL\_091\_increment函数\_自动生成主键值**

auto\_increment

自增数字auto\_increment，用来自动生成主键值，是MySQL独有的函数，默认从1开始，步进1递增

**例子：**

id int(4) primary key auto\_increment

**使用insert的时候就不需要填写id的值了**

**92、动力节点\_MySQL\_092\_课程回顾**

**//复习回顾**

**93、动力节点\_MySQL\_093\_约束\_外键约束概述及相关概念**

**foreign key 简称FK**

**外键约束，外键字段，外键值**

**给某个字段添加外键约束之后，这个字段就是外键字段，字段中的数据就是外键值。**

**单一外键：给一个字段添加外键约束**

**复合外键：给多个字段联合添加外键**

**同一张表中可以有多个外键存在**

**94、动力节点\_MySQL\_094\_约束\_外键约束\_根据个数分类\_单一外键和复合外键**

**95、动力节点\_MySQL\_095\_约束\_外键约束\_实例创建外键约束**

**需求：设计数据库表用来存储学生和班级信息（给出两种方案）**

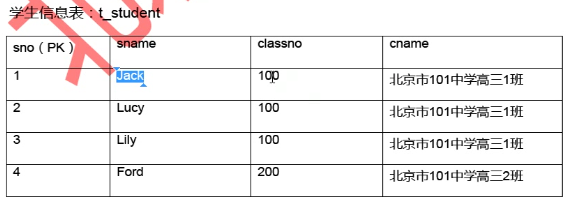
**需求分析：**

**学生表t\_student包含：sno,sname,classno,cname**

**学生和班级的关系，一个班级有多个学生，一个学生只能属于一个班级，属于一对多关系**

**第一种解决方案：**

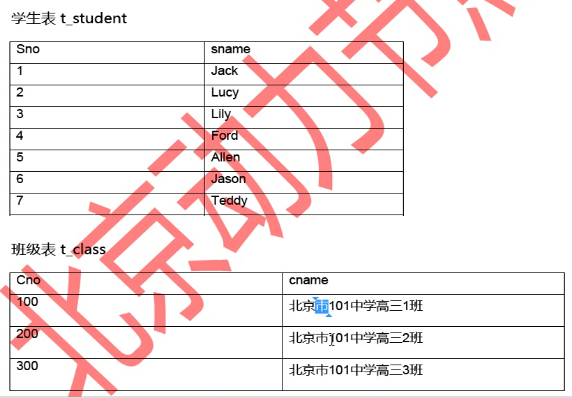
**把学生和班级信息都存储到一张表中**



**问题：高三一班重复出现，数据冗余**

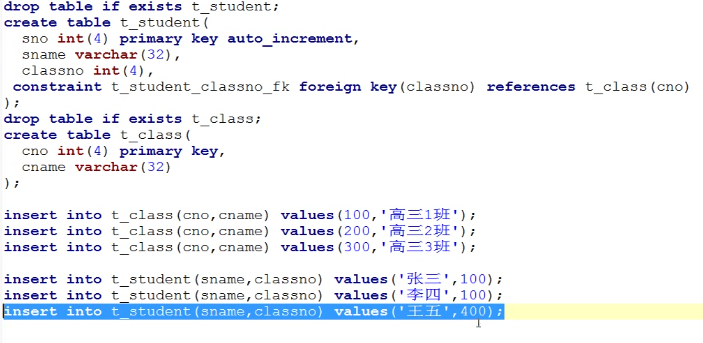
**第二种解决方案：**

**创建1张学生表，1张班级表**



**为了保证t\_student表中的cno字段的数据必须来之t\_class中的cno，需要给t\_student的cno字段添加外键约束，cno成为外键字段，100、200、300就是外键值，cno此处为单一外键**

**完整语句：**



**注意：**

1. **外键字段可以为NULL，空外键值为孤儿数据**
2. **被引用的字段必须unique约束（就是父表的主键）**
3. **外键引用之后，就可以区分父表和子表，t\_class为父表，t\_student为子表**

**96、动力节点\_MySQL\_096\_约束\_外键约束\_应该注意事项**

**先创建父表，再创建子表，先在父表插入数据，再在子表中插入数据**

**97、动力节点\_MySQL\_097\_约束\_外键约束\_查询出学生所对应的班级名称**

select ts.sname,tc.cname from t\_student as ts join t\_class as tc on ts.cno = tc.cno;

**注意：MySQL中没有提供修改外键约束的语法**

**98、动力节点\_MySQL\_098\_约束\_外键约束\_级联更新与级联删除概述**

**用法：添加级联更新与级联删除的时候，需要在外键约束后添加关键字**

**注意：级联更新与极限删除操作谨慎使用，因为级联操作会导致数据改变或者删除**

**99、动力节点\_MySQL\_099\_约束\_外键约束\_级联更新与级联删除\_级联删除的操作**

级联删除：on delete cascade

**先删除外键约束：**

ALTER TABLE 表名 DROP FOREIGN KEY 外键字段;

alter table t\_student drop foreign key t\_student\_classno\_fk;

**添加外键约束：**

alter table t\_student add constraint t\_student\_classno\_fk foreign key(classno) references t\_class(cno) on delete cascade;

**100、动力节点\_MySQL\_100\_约束\_外键约束\_级联更新与级联删除\_级联更新的操作**

级联删除：on update cascade

**先删除外键约束：**

ALTER TABLE 表名 DROP FOREIGN KEY 外键字段;

alter table t\_student drop foreign key t\_student\_classno\_fk;

**添加外键约束：**

alter table t\_student add constraint t\_student\_classno\_fk foreign key(classno) references t\_class(cno) on update cascade;

**101、动力节点\_MySQL\_101\_存储引擎的概述**

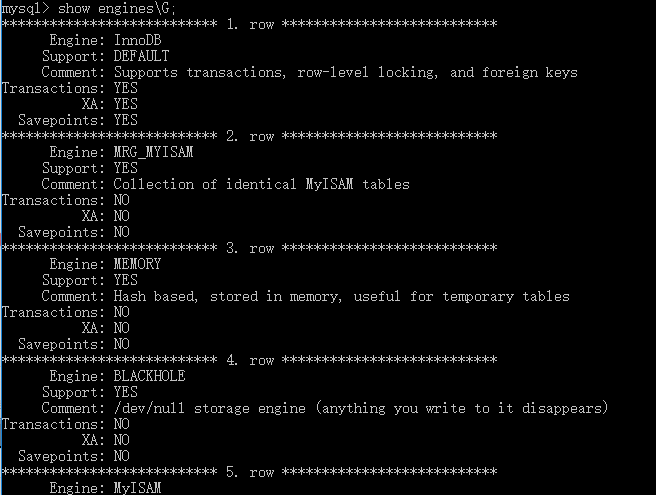
存储引擎是MySQL特有，其他数据库没有

本质：

1. 通过采用不同的技术将数据存储在文件或内存中
2. 每一种技术都有不同的存储机制，提供不同的功能，具备不同的能力
3. 选用不同的技术，可以获得额外的速度和功能，改善应用

查看存储引擎命令

show engines\G;



一共9种，默认是InnoDB

**102、动力节点\_MySQL\_102\_存储引擎\_存储引擎的基本操作**

1、创建表时，可以使用ENGINES=InnoDB指定引擎类型

create table table\_name(

no int(2)

)engine = InnoDB;

2、如果创建表时没有指定存储引擎，会使用默认的存储引擎

3、默认的存储引擎可以在安装目录的my.ini中配置default-storage-engin指定

4、修改表的存储引擎：alter table table\_name engine = new\_engine\_name;

5、查看表使用的存储引擎

show create table emp\G;

或者

show table status like ‘emp’\G;

**103、动力节点\_MySQL\_103\_存储引擎\_常用的存储引擎\_MyISAM**

1、是MySQL数据库最常用的

2、管理的表具备以下特性

a) 格式文件 存储表的结构 mytable.frm

b) 数据文件 存储表的数据 mytable.MYD

c) 索引文件 存储表的索引 mytable.MYI

3、可压缩、只读表，节省空间

**104、动力节点\_MySQL\_104\_存储引擎\_常用的存储引擎\_InnoDB**

1、是MySQL默认的存储引擎

2、管理的表具备以下特性

a) 每个InnoDB表在数据库目录中以.frm格式文件表示

b) InnoDB表空间tablespace被用于存储表的内容

c) 提供一组用来记录事务性活动的日志文件

d) 用commit/savepoint/roolback支持事务处理

e) 提供全部ACID兼容

f）在MySQL服务器崩溃后提供自动回复

g）多版本（MVCC）和行级锁定

h）支持外键及引用的完整性，包括级联更新和删除

**105、动力节点\_MySQL\_105\_存储引擎\_常用的存储引擎\_MEMORY**

1、 数据存储在内存中，且行的长度固定，因此非常快

2、 管理的表具备以下特性：

a) 在数据库目录中，每个表以.frm格式文件表示

b) 表数据及索引被存储在内存中

c) 表级锁机制

d) 字段属性不能包含TEXT或者BLOB字段

3、 旧名HEAP引擎

**106、动力节点\_MySQL\_106\_存储引擎\_如何选择合适的存储引擎**

1、MyISAM表适合于大量数据读而少量数据更新的混合操作。MyISAM表的另一种适用情形是使用压缩的只读表

2、如果查询中包含较多的数据更新操作，应该使用InnoDB，其行级锁机制和多版本的支持为数据读取和更新的混合提供了良好的并发机制（由事务控制）

3、使用MEMORY存储引擎存储非永久需要的数据，或者是能够从基于磁盘的表中重新生成测数据。

**107、动力节点\_MySQL\_107\_索引的概述**

index

相当于一本字典目录，提高程序的检索/查询效率，表中的每一个字段都可以添加索引

主键自动添加索引，能通过主键查询的尽量通过主键查询，效率较高

索引也是存储在磁盘文件中

1、 索引和表相同，是一个对象，表示存储在硬盘文件中，索引是表的一部分，因此也存放在硬盘文件中

**108、动力节点\_MySQL\_108\_索引\_索引的检索方式及什么情况下创建索引**

MySQL数据库有2中检索方式：

1. 全表扫描（效率较低）

例子：查询ename=’KING’

select \* from emp where ename=’KING’;

如果ename没有添加索引，那么通过ename过滤数据的时候，ename字段会全表扫描

1. 通过索引检索（提高查询效率）

创建索引的情况：

1. 该字段的数据量庞大
2. 该字段很少使用DML操作（索引需要维护，DML操作太多的时候，影响检索效率）
3. 该字段经常出现在筛选条件where中

实际开发中根据项目需求或客户需求综合调整

**109、动力节点\_MySQL\_109\_索引\_索引的应用**

1、创建索引

语法：

create index 索引名 on 表名（列名）;

create unique index 索引名 on 表名（列名）;

注：添加unique表示在该表中的该列添加一个唯一性约束

例如：create index dept\_dname\_index on dept(dname);

2、查看索引

show index from 表名;

3、删除索引

drop index 索引名on 表名;

**110、动力节点\_MySQL\_110\_视图概述及应用**

view

视图在数据库管理系统中也是一个对象，以文件形式存在

视图也对应了一个查询结果，只是从不同的角度查看数据

语法：

create view 视图名称 as 查询语句;

例如：create view myview as select \* from emp;

show tables;

可以查看到新建的myview视图

视图底层也是表

查看视图的语句

show create view my view;

删除：drop view if exists myview;

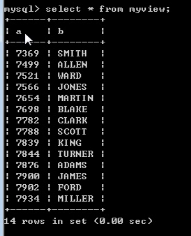
**111、动力节点\_MySQL\_111\_视图的作用\_隐藏表的实现细节**

隐藏表的实现细节

例子

create view myview as select empno as a, ename as b from emp;

select \* from myview;



**112、动力节点\_MySQL\_112\_视图的作用\_提高检索效率**

提高检索效率

例子

create view myview2 as

select e.ename,d.dname from emp e join dept d on e.deptno = d.deptno;



**113、动力节点\_MySQL\_113\_DBA简单的介绍**

**1、新建用户**

**create user username identified by ‘password’;**

**例子：**

**create user p361 identified by ‘123’;**

**登录后只能看见information\_schema一个库**

1. **授权**

**grant all privileges on dbname.tbname to ‘username’@’login ip’ identified by ‘password’ with grant option**

**dbname = 数据库，\*表示所有数据库**

**tbname = 表，\*表示所有表**

**login ip = 登录IP，%表示任意IP**

**password = 登录密码，空表示不需要密码**

**with grant option：表示这个用户可以授权其他用户**

**细粒度授权**

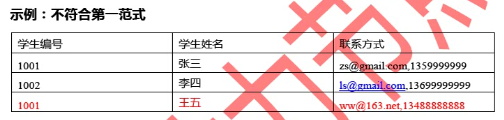
1. **root登录mysql**
2. **grant select,insert,update,delete on \*.\* to p361 @localhost identified by ‘123’;**
3. **localhost改为%表示可以再任何机器上登录mysql**
4. **回收授权**
5. **导入导出**

**114、动力节点\_MySQL\_114\_数据库设计三范式\_第一范式**

**主键、字段不能再分**

要求有主键，数据库中不能出现重复记录，每一个字段是原子性不能再分

实例：不符合第一范式：

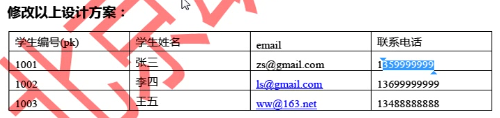


上面存在的问题：

数据存在重复记录，数据不唯一，没有主键

联系方式可以再分为邮箱和手机号，不是原子性

修改方案：



结论：

1. 每一行必须唯一，也就是每个表必须有主键，这是设计数据库的最基本要求
2. 主键主要通常采用数值型或者定长字符串表示
3. 关于列不可再分，应该根据具体的情况来决定，如联系方式，为了开发商的遍历有坑呢就采用一个字段，不分为邮箱和手机号

**115、动力节点\_MySQL\_115\_数据库设计三范式\_第二范式**

**非主键字段完全依赖主键**

第二范式在第一范式的基础上，要求数据库中所有非主键字段完全依赖主键（严格意义上说，尽量不要使用联合主键）

示例1，数据仍然可能重复



示例2，确定主键，学生编号，教师编码，出现冗余



分析

1. 上面例子虽然确定了主键，但是表会出现大量的冗余，主要涉及到的冗余字段是 学生姓名 和教师姓名
2. 出现冗余的原因是，学生姓名部分依赖了主键的一个字段 学生编号，没有依赖 教师编号

教师姓名部分依赖了逐渐的一个字段 教师编号，这就是第二范式部分依赖

解决方案：

老师和学生是多对多的关系

学生信息表：学生编号PK，学生姓名

教师信息表：教师编号PK，教师姓名

多对多的关系，需要有1张关系表：

教师-学生关系表：

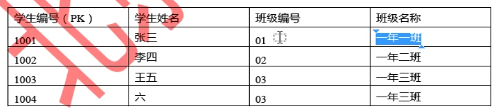
学生编号PK fk（学生表的学生编号），教师编号PK fk（教师表的教师编号）

结论：典型的多对多关系

**116、动力节点\_MySQL\_116\_数据库设计三范式\_第三范式**

简历在第二范式基础上，要求非主键字段**不能产生传递依赖**与主键字段

例子1，学生信息表



班级名称存在冗余，因为班级名称没有直接依赖主键（学生编号）

班级名称依赖于班级编号，班级编号依赖于学生编号（学生调班之后班级编号就变了），这就是传递依赖

一对多的设计：在多的一方添加外键

解决方案：

学生信息表：学生编号PK，学生姓名，班级编号FK

班级信息表：班级编号PK，班级名称

**117、动力节点\_MySQL\_117\_数据库设计\_数据库表关系\_一对一关系如何设计**

三范式总结

1. 一对一

方案1：分两张表存储，共享主键

t\_husband t\_wife两张表

t\_husband

hno(PK),hname

t\_wife

wno(PK，同时也是外键FK，引用t\_husband的主键),wname

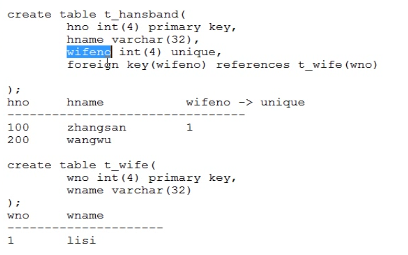
方案2：两张表存储，**外键唯一**

t\_husband

hno(PK),hname,wifeno(FK-unique)

t\_wife

wno(FK),wname



**118、动力节点\_MySQL\_118\_数据库设计\_实际开发中如何做**

1、尽量遵循三大范式

2、根据实际需求进行取舍，有时候冗余换速度，最终目的，满足需求

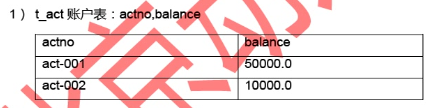
**119、动力节点\_MySQL\_119\_事务\_什么是事务**

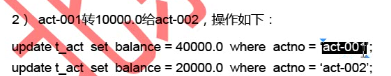
transaction

1. 一个最小的不可再分的工作单元
2. 通常一个事务对应一个完整的业务（如：银行转账业务）
3. 一个完整的业务需要批量的DML语句共同完成
4. 事务只和DML语句有关系
5. 以上所描述的额批量DML语句，数量和业务逻辑相关

**例子：银行转账**

**示例，账户转账**





分析：

1. 上面两条DML语句必须都执行成功，要么就不执行
2. 第一条DML执行成功之后，不能修改数据库，而只是记录操作，这个记录是在内存中完成的
3. 第二条DML也成功之后，底层数据库文件的数据此时完成同步
4. 如果第二条执行失败，会清空所有的历史操作记录，数据库不变

结论：上面的功能要借助事务完成

**120、动力节点\_MySQL\_120\_事务\_事务的四个特性**

ACID

原子性（atomicity），事务是最小单位，不可再分

一致性（consistency），所有DML语句操作，必须保证全部成功或者全部失败

隔离性（isolation），一个事务不会影响其他事务

持久性（durability），事务完成之后，对数据库做的修改会持久的保存在数据库中，不会被回滚

**121、动力节点\_MySQL\_121\_事务\_事务的相关概念**

开启事务：start transaction

结束事务：end transaction

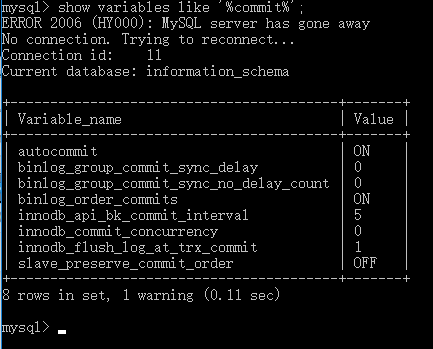
提交事务：commit transaction

回滚事务：rollback transaction

重点：

MySQL默认是自动提交的

show variables like ‘%commit%’;



操作：

手动开始事务

start transaction;

DML

commit;

设置不要自动提交：

set autocommit = OFF;

**122、动力节点\_MySQL\_122\_事务\_事务隔离级别概述**

四个隔离级别

read uncommitted 读未提交，级别最低

事务A和事务B，事务A未提交的数据，事务B可以读取

这里读取到的数据可以叫做脏数据，或者，脏读 Dirty Read

这种级别一般只是理论上存在，数据库默认的隔离级别都高于这个

read committed 读已提交

事务A每一次提交，事务B读取到的数据库都会变化

会导致：B一直都在读取变化的数据库

Oracle数据库管理系统的默认隔离级别为可重复读

repeatable read 可重复读（MySQL的默认级别）

事务A每次提交都不影响事务B读取的数据库内容，也就是事务B读取到的数据库一直都不变化

会导致：B感知不到数据库变化

serializable 串行化

事务排队，一个完成后才能继续下一个

这种级别很少使用，吞吐量小，用户体验不好

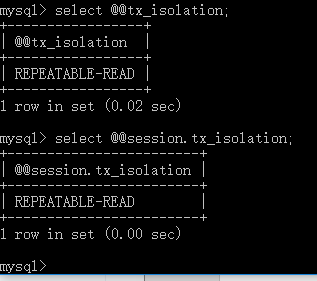
**123、动力节点\_MySQL\_123\_事务\_演示事务隔离级别**

**包括会话级别，全局级别**

查看当前会话级隔离级别

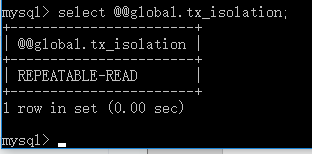
select @@tx\_isolation;

select @@session.tx\_isolation;



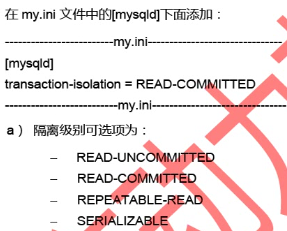
全局隔离级别

select @@global.tx\_isolation;



**设置服务器缺省隔离级别**

第一种方法：修改my.ini配置文件



第二种方法，通过命令方式设置事务隔离级别

set transaction isolation level isolation-level;

isolation-level可选4种隔离级别

设置作用范围：

session

global

set global transaction isolation level READ COMMITED;

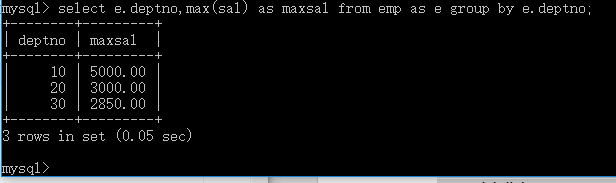
set session transaction isolation level READ COMMITED;

**124、动力节点\_MySQL\_124\_34道经典SQL试题\_第1题**

取得每个部门最高薪水的人员名称

第一步：取得每个部门的最高薪水

select e.deptno,max(sal) as maxsal from emp as e group by e.deptno;



第二步：把上面这个表作为临时表t，和emp表做表连接

select e.deptno,e.ename,t.maxsal,e.sal

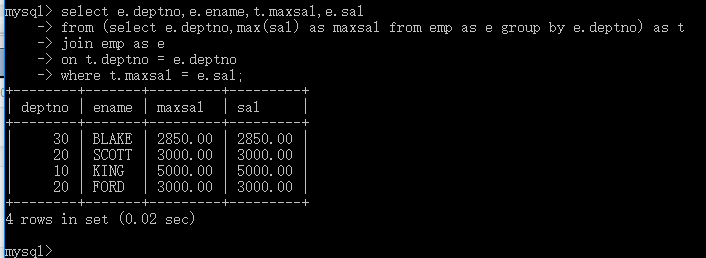
from (select e.deptno,max(sal) as maxsal from emp as e group by e.deptno) as t

join emp as e

on t.deptno = e.deptno

where t.maxsal = e.sal

order by e.deptno;



**125、动力节点\_MySQL\_125\_34道经典SQL试题\_第2题**

那些人的薪水在部门平均薪水之上

mysql> select e.deptno,avg(e.sal) as avgsal from emp e group by e.deptno;

+--------+-------------+

| deptno | avgsal |

+--------+-------------+

| 10 | 3150.000000 |

| 20 | 2220.833333 |

| 30 | 1566.666667 |

+--------+-------------+

3 rows in set (0.01 sec)

mysql> select t.deptno,t.avgsal,e.ename,e.sal from (select e.deptno,avg(e.sal) as avgsal from emp e group by e.deptno) t join emp e on t.deptno = e.deptno where e.sal > t.avgsal order by e.deptno;

+--------+-------------+-------+---------+

| deptno | avgsal | ename | sal |

+--------+-------------+-------+---------+

| 10 | 3150.000000 | KING | 5000.00 |

| 20 | 2220.833333 | JONES | 2975.00 |

| 20 | 2220.833333 | CLARK | 2450.00 |

| 20 | 2220.833333 | SCOTT | 3000.00 |

| 20 | 2220.833333 | FORD | 3000.00 |

| 30 | 1566.666667 | ALLEN | 1600.00 |

| 30 | 1566.666667 | BLAKE | 2850.00 |

+--------+-------------+-------+---------+

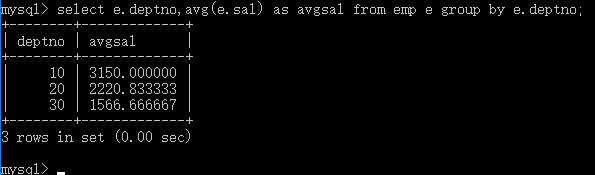
7 rows in set (0.00 sec)

**126、动力节点\_MySQL\_126\_34道经典SQL试题\_第3题\_第一种情况**

取得部门中（所有人的）平均薪水的等级

取得部门的平均薪水：

select e.deptno,avg(e.sal) as avgsal from emp e group by e.deptno;



根据平均薪水划分等级：

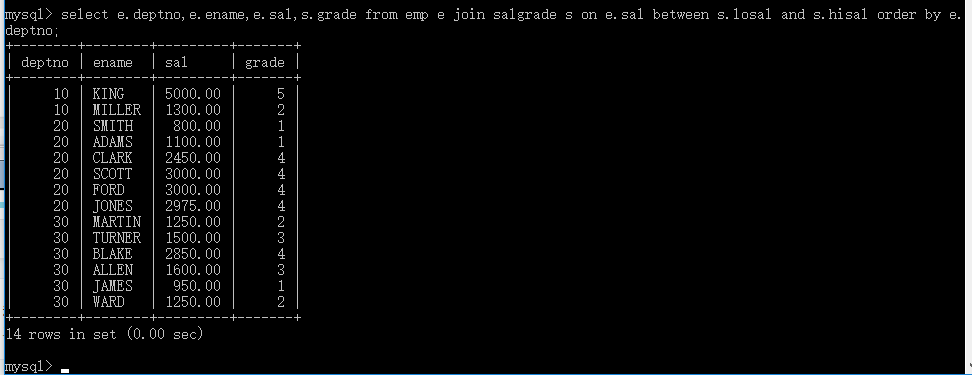
select t.deptno,t.avgsal,s.grade from (select e.deptno,avg(e.sal) as avgsal from emp e group by e.deptno) t join salgrade s on t.avgsal between s.losal and s.hisal order by t.deptno;

**127、动力节点\_MySQL\_127\_34道经典SQL试题\_第3题\_第二种情况**

取得部门中（所有人的）平均的薪水等级

每个人的薪水等级

select e.deptno,e.ename,e.sal,s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal order by e.deptno;



再求平均：

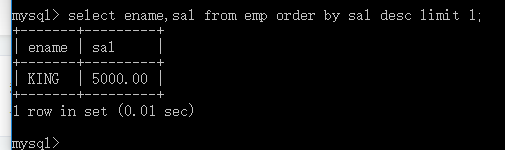
select t.deptno,avg(t.grade) as avgGrade from (select e.deptno,e.ename,e.sal,s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal order by e.deptno) group by t.deptno;

**128、动力节点\_MySQL\_128\_34道经典SQL试题\_第4题**

不用组函数MAX取得最高薪水

方案1：

select ename,sal from emp order by sal desc limit 1;



方案2：

A表：select sal from emp;

B表：select sal from emp;

从A表中的sal取出小于B表sal的sal，只有5000没被取出来

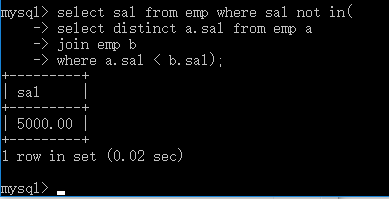
记得去重，默认会显示所有对比记录

select sal from emp where sal not in(

select distinct a.sal from emp a

join emp b

where a.sal < b.sal);



**129、动力节点\_MySQL\_129\_34道经典SQL试题\_第5题**

取得平均薪水最高的部门的部门编号（**可能存在相同平均薪水的部门**）

先获取平均薪水：

select e.deptno,avg(e.sal) as avgsal from emp e group by e.deptno order by e.deptno;

上面作为临时表，获取最大的平均薪水：

select max(t.avgsal) as maxAvgSal from (select e.deptno,avg(e.sal) as avgsal from emp e group by e.deptno order by e.deptno) t;

过滤：

select e.deptno,avg(e.sal) as avgsal

from emp e

group by e.deptno

having avgsal = (select max(t.avgsal) as maxAvgSal from (select e.deptno,avg(e.sal) as avgsal from emp e group by e.deptno order by e.deptno) t)

order by e.deptno;

**130、动力节点\_MySQL\_130\_34道经典SQL试题\_第6题**

取得平均薪水最高的部门的部门名称

上面一题改一下

select e.deptno,avg(e.sal) as avgsal,d.dname

from emp e

join dept d

on e.deptno = d.deptno

group by e.deptno

having avgsal = (select max(t.avgsal) as maxAvgSal from (select e.deptno,avg(e.sal) as avgsal from emp e group by e.deptno order by e.deptno) t)

order by e.deptno;

**131、动力节点\_MySQL\_131\_34道经典SQL试题\_第7题**

求平均薪水的等级最低的部门的部门名称

1. 部门的平均薪水

select e.deptno,avg(e.sal) as avgsal,d.dname

from emp e

join dept d

on e.deptno = d.deptno

group by e.deptno,d.dname;

1. 平均薪水的等级，上面的作为临时表

select t1.deptno,t1.avgsal,s.grade

from t1

join salgrade s

on t1.avgsal between s.losal and s.hisal;

select t1.deptno,t1.avgsal,s.grade

from (select e.deptno,avg(e.sal) as avgsal,d.dname

from emp e

join dept d

on e.deptno = d.deptno

group by e.deptno,d.dname

) t1

join salgrade s

on t1.avgsal between s.losal and s.hisal;

1. 上面再作为临时表，拿出等级最低的等级是多少

select min(t2.grade) as minGrade from t2;

select min(t2.grade) as minGrade

from (select t1.deptno,t1.avgsal,s.grade

from (select e.deptno,avg(e.sal) as avgsal,d.dname

from emp e

join dept d

on e.deptno = d.deptno

group by e.deptno,d.dname

) t1

join salgrade s

on t1.avgsal between s.losal and s.hisal

) t2;

1. 再取出最终结果：

第二步的结果having做筛选

select t1.deptno,t1.avgsal,s.grade

from t1

join salgrade s

on t1.avgsal between s.losal and s.hisal

having s.grade = (select min(t2.grade) as minGrade from t2);

**132、动力节点\_MySQL\_132\_34道经典SQL试题\_第8题**

取得比普通员工（员工代码没有在MGR上出现的）的最高薪水还要高的经理人姓名

1. 找出普通员工
   1. 找出mgr

select distinct mgr from emp where mgr is not null;

select \* from emp where empno not in (select distinct mgr from emp where mgr is not null);

//注意一个问题：not in不会忽略NULL，如果直接写select \* from emp where empno not in (select distinct mgr from emp); 结果就为空，in会忽略NULL，如果去掉not，会显示记录

1.2 找出这里面的最高工资

select max(sal) as maxSal from emp where empno not in (select distinct mgr from emp where mgr is not null);

普通员工的最高薪水为1600

1. 找出经理人：

select e.ename from emp e where sal > (select max(sal) as maxSal from emp where empno not in (select distinct mgr from emp where mgr is not null)) and e.mgr is null;

**133、动力节点\_MySQL\_133\_34道经典SQL试题\_第9题**

取得薪水最高的前五名员工

select \* from emp order by sal desc limit 0,5;

**134、动力节点\_MySQL\_134\_34道经典SQL试题\_第10题**

取得薪水最高的第6到第10名员工

select \* from emp order by sal desc limit 5,5;

**135、动力节点\_MySQL\_135\_34道经典SQL试题\_第11题**

取得最后入职的5名员工

select \* from emp order by hiredate desc limit 0,5;

**136、动力节点\_MySQL\_136\_34道经典SQL试题\_第12题**

取得每个薪水等级有多少员工

先获得员工的薪水等级

select e.sal,s.grade

from emp e

join salgrade s

on e.sal between s.losal and s.hisal

order by s.grade;

上面作为临时表，按照等级分组

select t.grade,count(t.grade) from

(select e.sal,s.grade

from emp e

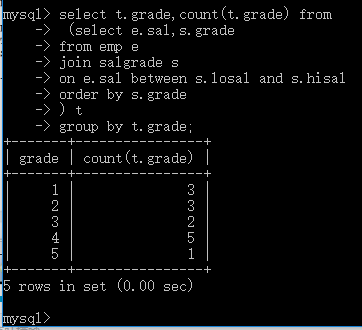
join salgrade s

on e.sal between s.losal and s.hisal

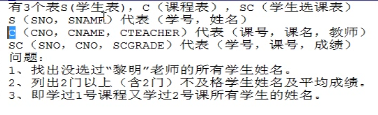
order by s.grade

) t

group by t.grade;



**137、动力节点\_MySQL\_137\_34道经典SQL试题\_第13题**



create table s(

sno int(4) primary key unique auto\_increment,

sname varchar(32)

);

create table c(

cno int(4) primary key unique auto\_increment,

cname varchar(32),

cteacher varchar(32)

);

create table sc(

sno int(4),

cno int(4),

scgrade double(3,1),

constraint sc\_sno\_cno\_pk primary key(sno,cno),

constraint sc\_sno\_fk foreign key(sno) references s(sno),

constraint sc\_cno\_fk foreign key(cno) references c(cno)

);

insert into s(sname) values('zhangsan');

insert into s(sname) values('lisi');

insert into s(sname) values('wangwu');

insert into s(sname) values('zhaoliu');

insert into c(cname,cteacher) values('java','wu');

insert into c(cname,cteacher) values('c++','wang');

insert into c(cname,cteacher) values('c#','zhang');

insert into c(cname,cteacher) values('mysql','guo');

insert into c(cname,cteacher) values('oracle','liming');

insert into sc(sno,cno,scgrade) values(1,1,30);

insert into sc(sno,cno,scgrade) values(1,2,50);

insert into sc(sno,cno,scgrade) values(1,3,80);

insert into sc(sno,cno,scgrade) values(1,4,90);

insert into sc(sno,cno,scgrade) values(1,5,70);

insert into sc(sno,cno,scgrade) values(2,1,30);

insert into sc(sno,cno,scgrade) values(2,2,50);

insert into sc(sno,cno,scgrade) values(2,3,80);

insert into sc(sno,cno,scgrade) values(2,4,60);

insert into sc(sno,cno,scgrade) values(2,5,60);

insert into sc(sno,cno,scgrade) values(3,1,30);

insert into sc(sno,cno,scgrade) values(3,2,50);

insert into sc(sno,cno,scgrade) values(3,3,80);

insert into sc(sno,cno,scgrade) values(3,4,70);

insert into sc(sno,cno,scgrade) values(3,5,60);

insert into sc(sno,cno,scgrade) values(4,1,30);

insert into sc(sno,cno,scgrade) values(4,2,50);

insert into sc(sno,cno,scgrade) values(4,3,80);

insert into sc(sno,cno,scgrade) values(4,4,90);

insert into sc(sno,cno,scgrade) values(4,5,70);

delete from sc where sno = 3 and cno = 5;

/\*liming老师的课程编号\*/

select cno from c where cteacher = 'liming';

/\*选了liming老师课程的学生编号\*/

select sno from sc where cno = (select cno from c where cteacher = 'liming');

/\*没有选liming老师课程的学生名字\*/

select sname from s where sno not in (select sno from sc where cno = (select cno from c where cteacher = 'liming'));

/\*\*

列出2门以上（含2门）不及格学生姓名及平均成绩

\*/

/\*按照学生编号分组，找出不及格学生的编号\*/

select sc.sno,count(\*) as courseNum from sc where scgrade < 60

group by sno

having courseNum >= 2;

/\*根据编号找学生姓名\*/

select sc.sno,s.sname,count(\*) as courseNum

from sc

join s

on sc.sno = s.sno

where scgrade < 60

group by sc.sno,s.sname

having courseNum >= 2;

/\*平均成绩\*/

select sc.sno,avg(sc.scgrade) as avggrade

from sc

group by sc.sno;

/\*两张表关联\*/

select t1.sno,t1.sname,t2.avggrade

from (select sc.sno,s.sname,count(\*) as courseNum

from sc

join s

on sc.sno = s.sno

where scgrade < 60

group by sc.sno,s.sname

having courseNum >= 2) t1

join (select sc.sno,avg(sc.scgrade) as avggrade

from sc

group by sc.sno

) t2

on t1.sno = t2.sno;

/\*\*

同时学1号课程和2号课程的所有学生姓名

\*/

select sno from sc where cno = 1;

select sno from sc where cno = 2;

select s.sno,s.sname

from sc

join s

on sc.sno = s.sno

where cno = 1 and sc.sno in (select sno from sc where cno = 2);

**138、动力节点\_MySQL\_138\_34道经典SQL试题\_第14题**

列出所有员工和领导的名字

把一张表看成两张表

select a.ename,b.ename as leaderName

from emp a

left join emp b

on a.mgr = b.empno;

**139、动力节点\_MySQL\_139\_34道经典SQL试题\_第15题**

列出雇佣日期早于其直接上级的所有员工编号，姓名，部门名称

select a.empno,a.ename

from emp a

left join emp b

on a.mgr = b.empno

join dept d

where a.deptno = d.deptno

where a.hiredate < b.hiredate;

**140、动力节点\_MySQL\_140\_34道经典SQL试题\_第16题**

列出部门名称和这些部门的员工信息，以及没有员工的部门

select d.dname,e.\*

from emp e

right join dept d

on e.deptno = d.deptno

**141、动力节点\_MySQL\_141\_34道经典SQL试题\_第17题**

列出至少有5个员工的所有部门

select e.deptno,count(e.ename) as totalEmp

from emp e

group by e.deptno;

select e.deptno,count(e.ename) as totalEmp

from emp e

group by e.deptno

having totalEmp >= 5;

**142、动力节点\_MySQL\_142\_34道经典SQL试题\_第18题**

列出薪水比smith多的员工信息

select sal from emp where ename = ‘SMITH’;

select \* from emp where sal > (select sal from emp where ename = ‘SMITH’);

**143、动力节点\_MySQL\_143\_34道经典SQL试题\_第19题**

列出所有CLERK的姓名，部门名称，部门人数

###########T1##########

select d.detpno,e.ename,d.dname

from emp e

join dept d

on e.deptno = d.deptno

where e.job = ‘CLERK’;

###########T2##########

select e.deptno,count(e.ename) as totalEmp

from emp e

group by e.deptno;

select t1.deptno,t1.dname,t1.ename,t2.totalEmp

from () t1

join () t2

on t1.deptno = t2.deptno;

**144、动力节点\_MySQL\_144\_34道经典SQL试题\_第20题**

列出最低薪水大于1500的各种工作及从事此工作的全部雇员人数

select e.job,min(e.sal) as minSal

from emp e

group by e.job;

select e.job,min(e.sal) as minSal

from emp e

group by e.job

having minSal > 1500;

select e.job,min(e.sal) as minSal,count(e.ename) as totalEmp

from emp e

group by e.job

having minSal > 1500;

**145、动力节点\_MySQL\_145\_34道经典SQL试题\_第21题**

列出在部门SALES工作的员工的姓名，假设不知道SALES的部门编号

select deptno from dept where dname = ‘SALES’;

select ename from emp where deptno = ();

**146、动力节点\_MySQL\_146\_34道经典SQL试题\_第22题**

列出薪水高于公司平均薪水的额所有员工，所在部门，上级领导，雇员的工资等级

select avg(sal) as avgsal from emp;

select

from emp e

join dept d

on e.deptno = d.deptno

where e.sal > 平均工资;

select d.dname,e.ename

from emp e

join dept d

on e.deptno = d.deptno

where e.sal > 平均工资;

select d.dname,e.ename,b.ename as leaderName

from emp e

join dept d

on e.deptno = d.deptno

join emp b

on e.mgr = b.empno

where e.sal > 平均工资;

select d.dname,e.ename,b.ename as leaderName,s.grade

from emp e

join dept d

on e.deptno = d.deptno

left join emp b

on e.mgr = b.empno

join salgrade s

on e.sal between s.losal and s.hisal

where e.sal > (select avg(sal) as avgsal from emp);

**147、动力节点\_MySQL\_147\_34道经典SQL试题\_第23题**

列出与SCOTT从事相同工作的所有员工和部门名称

select job

from emp

where ename = ‘SCOTT’;

select d.dname,e.\*

from emp e

join dept d

on e.deptno = d.deptno

where e.job = 工作;

select d.dname,e.\*

from emp e

join dept d

on e.deptno = d.deptno

where e.job = (select job

from emp

where ename = ‘SCOTT’);

**148、动力节点\_MySQL\_148\_34道经典SQL试题\_第24题**

列出薪金等于部门30中员工的薪金的其他员工的姓名和薪金

select distinct sal from emp where deptno = 30;

select ename,sal

from emp

where sal in(薪金) and deptno != 30;

select ename,sal

from emp

where sal in(select distinct sal from emp where deptno = 30) and deptno != 30;

**149、动力节点\_MySQL\_149\_34道经典SQL试题\_第25题**

列出薪金大于部门30中员工的薪金的其他员工的姓名、薪金、部门名称

select max(sal) as maxSal from emp where deptno = 30;

select e.ename,e.sal,d.dname

from emp e

join dept d

on e.deptno = d.deptno

where e.sal > (select max(sal) as maxSal from emp where deptno = 30);

**150、动力节点\_MySQL\_150\_34道经典SQL试题\_第26题**

列出在每个部门工作的员工数量、平均工资、入职总天数

to\_days(日期类型)

当前时间函数

now()

服务天数：

select ename,(to\_days(now()) - to\_days(hiredate)) as serviceDays from emp;

每个部门工作的员工

select e.deptno,count(e.ename) as totolEmp,avg(e.sal) as avgsal,avg(to\_days(now()) - to\_days(hiredate)) as avgserveryear

from emp e

group by e.deptno;

**151、动力节点\_MySQL\_151\_34道经典SQL试题\_第27题**

列出所有员工的姓名，部门名称，工资

select e.ename,d.dname,e.sal

from emp e

right join dept d

on e.deptno = d.deptno;

**152、动力节点\_MySQL\_152\_34道经典SQL试题\_第28题**

列出所有部门的详细信息和人数

select d.\*,count(e.ename) as totalEmp

from emp e

right join dept d

on d.deptno = e.deptno

group by e.deptno,d.dname,d.loc

order by d.deptno;

**153、动力节点\_MySQL\_153\_34道经典SQL试题\_第29题**

列出各种工作的额最低工资以及从事此工作的雇员姓名

select e.job,min(e.sal) as minsal

from emp e

group by e.job;

select e.ename

from emp e

join t

on e.job = t.job

where e.sal = t.minsal;

select e.ename

from emp e

join (select e.job,min(e.sal) as minsal

from emp e

group by e.job) t

on e.job = t.job

where e.sal = t.minsal;

**154、动力节点\_MySQL\_154\_34道经典SQL试题\_第30题**

列出各个部门的manager的最低薪水

select e.deptno,min(e.sal) as minsal

from emp e

where e.job = ‘MANAGER’

group by e.deptno;

**155、动力节点\_MySQL\_155\_34道经典SQL试题\_第31题**

求出所有员工的年工资，按照年薪从低到高排序

select e.ename,12\*(e.sal+ifnull(e.comm,0)) as yearSal

from emp e

order by yearSal asc;

**156、动力节点\_MySQL\_156\_34道经典SQL试题\_第32题**

求出员工领导的薪水超过3000的员工名称和领导名称

select e.ename,b.ename as leadername

from emp e

right join emp b

on e.mgr = b.empno

where b.sal > 3000;

**157、动力节点\_MySQL\_157\_34道经典SQL试题\_第33题**

求部门名称中带S字符的员工的工资合计，部门人数

select d.dname,sum(e.sal) as sumSal,count(e.ename) as totalEmp

from emp e

join dept d

on e.deptno = d.deptno

where d.dname like ‘%s%’

group by d.dname;

**158、动力节点\_MySQL\_157\_34道经典SQL试题\_第34题**

给任职日期超过30年的员工加薪10%

create table emp\_bak1 as select \* from emp;

update emp\_bak1 set sal = sal\*1.1 where (to\_days(now()) - to\_days(hiredate))/365 > 30;

**2018年2月17日星期六 23:30**