# 1、数据库概述及数据库准备

## 1.1、SQL概述

SQL，一般发音为sequel，SQL的全称Structured Query Language)，SQL用来和数据库打交道，完成和数据库的通信，SQL是一套标准。但是每一个数据库都有自己的特性别的数据库没有,当使用这个数据库特性相关的功能,这时SQL语句可能就不是标准了.(90%以上的SQL都是通用的)

## 1.2、什么是数据库

数据库（DataBase，简称DB），通常是一个或一组文件，保存了一些符合特定规格的数据,数据库对应的英语单词是DataBase,简称:DB,数据库软件称为数据库管理系统（DBMS），全称为DataBase Management System，如：Oracle、MS SQL Server、MySql、Sybase、informix、DB2、interbase、PostgreSql 。

思考：DB DBMS SQL三者之间的关系？

程序员写SQL语句，DBMS负责执行这个SQL语句，然后对数据库中数据进行管理。

## 1.3、MySql概述

MySQL最初是由“MySQL AB”公司开发的一套关系型数据库管理系统（RDBMS-Relational Database Mangerment System）。

MySQL不仅是最流行的开源数据库，而且是业界成长最快的数据库，每天有超过7万次的下载量，其应用范围从大型企业到专有的嵌入应用系统。

MySQL AB是由两个瑞典人和一个芬兰人：David Axmark、Allan Larsson和Michael “Monty” Widenius在瑞典创办的。

在2008年初，Sun Microsystems收购了MySQL AB公司。在2009年，Oracle收购了Sun公司，使MySQL并入Oracle的数据库产品线。

## 1.4、MySql的安装

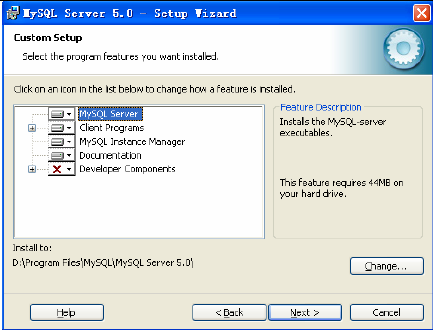
打开下载的mysql安装文件mysql-essential-5.0.22-win32.msi，双击运行，出现如下界面



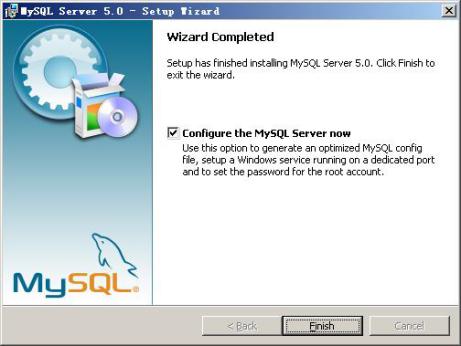
按“Next”继续



选择安装类型，有“Typical（默认）”、“Complete（完全）”、“Custom（用户自定义）”三个选项，我们选择“Custom”，有更多的选项，也方便熟悉安装过程



上一步选择了 Custom 安装，这里将设定 MySQL 的组件包和安装路径，设定好之后，单击 Next 继续安装。

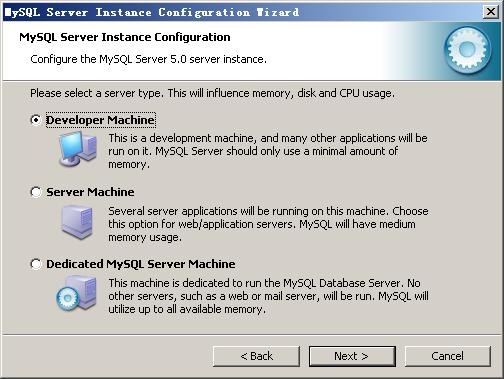


现在软件安装完成了，出现上面的界面，将 “Configure the Mysql Server now”前面的勾打上，点“Finish”结束软件的安装并启动mysql配置向导。

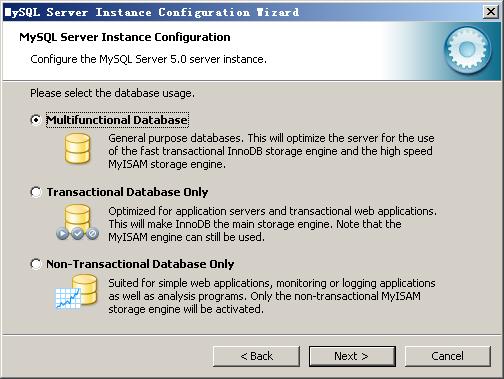
mysql配置向导启动界面，按“Next”继续。



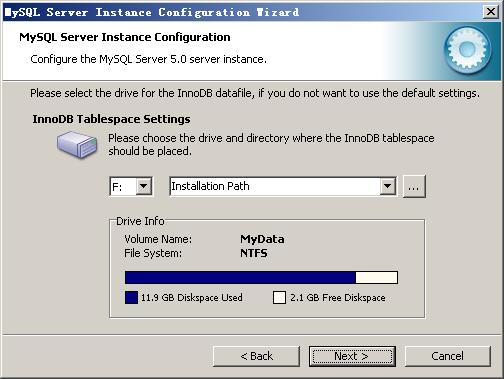
选择配置方式，“Detailed Configuration（手动精确配置）”、“Standard Configuration（标准配置）”，我们选择“Detailed Configuration”，方便熟悉配置过程。



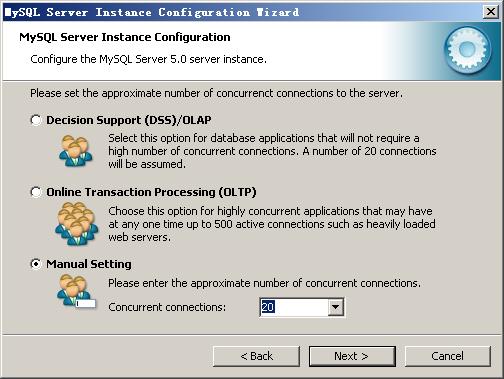
选择服务器类型，“Developer Machine（开发测试类，mysql占用很少资源）”、“Server Machine（服务器类型，mysql占用较多资源）”、“Dedicated MySQL Server Machine（专门的数据库服务器，mysql占用所有可用资源）”，大家根据自己的类型选择了，一般选“Server Machine”，不会太少，也不会占满。



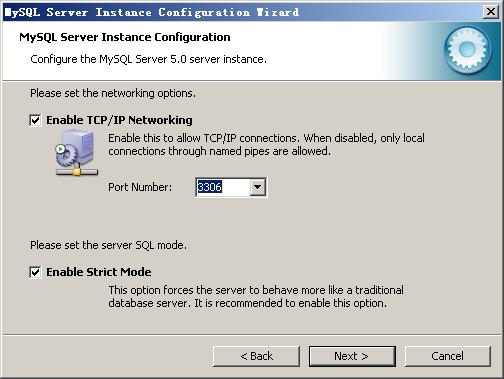
选择mysql数据库的大致用途，“Multifunctional Database（通用多功能型，能很好的支持InnoDB与MyISAM存储引擎）”、“Transactional Database Only（服务器类型，专注于事务处理，一般）”、“Non-Transactional Database Only（非事务处理型，较简单，主要做一些监控、记数用，对MyISAM数据类型的支持仅限于non-transactional），随自己的用途而选择了，我这里选择“Multifunctional Database”， 按“Next”继续。



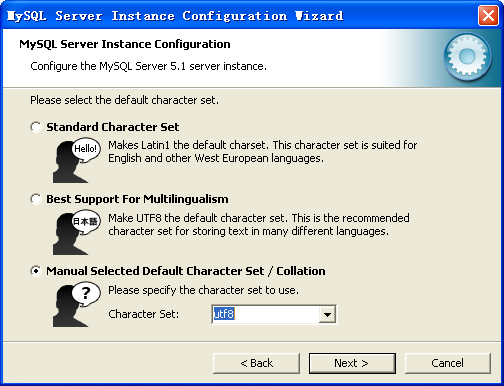
对InnoDB Tablespace进行配置，就是为InnoDB 数据库文件选择一个存储空间，如果修改了，要记住位置，重装的时候要选择一样的地方，否则可能会造成数据库损坏，当然，对数据库做个备份就没问题了，这里不详述。我这里没有修改，使用用默认位置，直接按“Next”继续。



选择您的网站的一般mysql访问量，同时连接的数目，“Decision Support(DSS)/OLAP（20个左右）”、“Online Transaction Processing(OLTP)（500个左右）”、“Manual Setting（手动设置，自己输一个数）”，我这里选“Decision Support(DSS)/OLAP)”，按“Next”继续



是否启用TCP/IP连接，设定端口，如果不启用，就只能在自己的机器上访问mysql数据库了，我这里启用，把前面的勾打上，Port Number：3306，在这个页面上，您还可以选择“启用标准模式”（Enable Strict Mode），按“Next”继续。



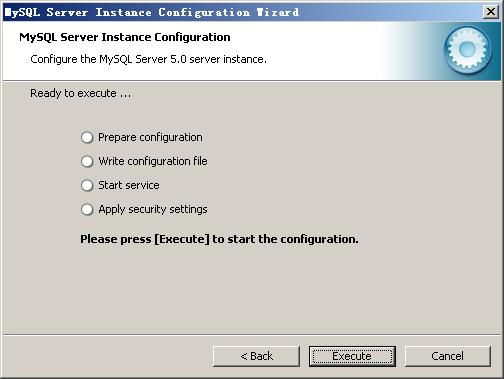
这个比较重要，就是对mysql默认数据库语言编码进行设置，第一个是西文编码，我们要设置的是utf8编码，按 “Next”继续。



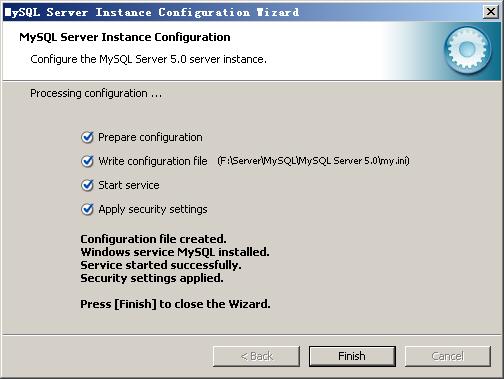
选择是否将mysql安装为windows服务，还可以指定Service Name（服务标识名称），是否将mysql的bin目录加入到Windows PATH（加入后，就可以直接使用bin下的文件，而不用指出目录名，比如连接，“mysql.exe -uusername -ppassword;”就可以了，不用指出mysql.exe的完整地址，很方便），我这里全部打上了勾，Service Name不变。按“Next”继续。



设置完毕，按“Next”继续。



确认设置无误，如果有误，按“Back”返回检查。按“Execute”使设置生效。

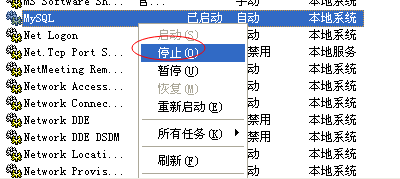


设置完毕，按“Finish”结束mysql的安装与配置

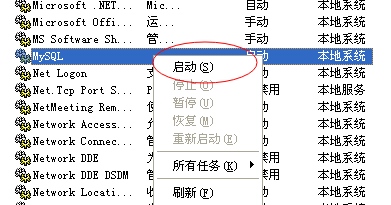
可以通过服务管理器管理 MYSQL 的服务。

通过命令调用服务管理器:services.msc

停止 MYSQL 的服务。

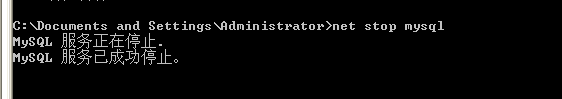


启动 MYSQL 的服务。

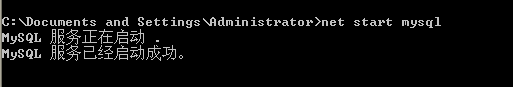


也可以在 DOS 中直接通过命令行的形式进行控制。

停止 MYSQL 的服务。



启动 MYSQL 的服务。



## 1.5、表

表(table)是一种**结构化的文件**，可以用来存储特定类型的数据，如：学生信息，课程信息，都可以放到表中。另外表都有特定的名称，而且不能重复。表中具有几个概念：列、行、主键。 列叫做字段(Column),行叫做表中的记录,每一个字段都有:字段名称/字段数据类型/字段约束/字段长度

学生信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 学号（主键） | 姓名 | 性别 | 年龄 |
| 00001 | 张三 | 男 | 20 |
| 00002 | 李四 | 女 | 20 |

## 1.6、SQL的分类

数据查询语言(DQL-Data Query Language)

代表关键字:select

数据操纵语言(DML-Data Manipulation Language)

代表关键字:insert,delete,update

数据定义语言(DDL-Data Definition Language)

代表关键字:create ,drop,alter,

事务控制语言(TCL-Transactional Control Language)

代表关键字:commit ,rollback;

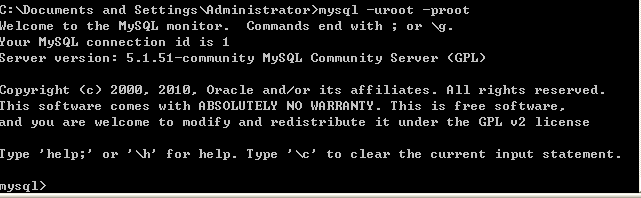
数据控制语言(DCL-Data Control Language)

代表关键字:grant,revoke.

## 1.7、导入演示数据

使用MySQL命令行客户端来装载数据库。

1. 连接MySql



创建“bjpowernode”数据库

mysql> create database bjpowernode;

选择数据库

mysql> use bjpowernode

导入数据

mysql>source D:\ bjpowernode.sql

删除数据库(这里不要做!)

mysql> drop database bjpowernode;

# 2、常用命令

## 2.1、查看msyql版本

* MySQL程序选项具有以下两种通用形式：
  + 长选项，由单词之前加两个减号组成
  + 短选项，由单个字母之前加一个减号组成

C:\Users\Administrator>mysql --version

mysql Ver 14.14 Distrib 5.5.36, for Win32 (x86)

C:\Users\Administrator>mysql -V

mysql Ver 14.14 Distrib 5.5.36, for Win32 (x86)

## 2.2、创建数据库

1. create database 数据库名称;

create database bjpowernode;

1. use 数据库名称

use bjpowernode;

## 2.3、查询当前使用的数据库

select database();

查询数据库版本也可以使用

select version();

## 2.4、终止一条语句

如果想要终止一条正在编写的语句，可键入\c。

## 2.5、退出mysql

可使用\q、QUIT或EXIT：

如：

mysql> \q (ctrl+c)

## 2.6、表的复制

create table dept2 as select \* from dept;

# 3、查看“演示数据”的表结构

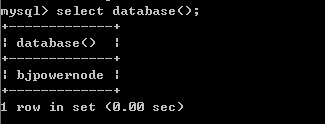
## 3.1、查看和指定现有的数据库

Show databases；

## 3.2、指定当前缺省数据库

Use bjpowernode；

## 3.3、查看当前使用的库



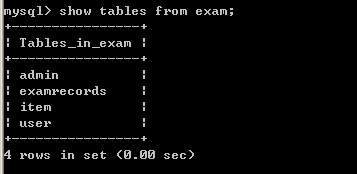
## 3.4、查看当前库中的表

Show tables；

## 3.5、查看其他库中的表

show tables from <database name>;

如查看exam库中的表



## 3.6、查看表的结构

desc <table name>;

## 3.7、查看表的创建语句

show create table <table name>;

# 存储引擎

（了解内容）

\* 存储引擎只有在MySQL当中才有这个概念/术语，其它数据库中没有。

\* 什么是存储引擎呢？在MySQL当中数据库存储表有很多不同的方式，每一种方式都被称为存储引擎。有很多存储引擎，就表示有很多不同的存储方式。

在MySQL当中表有很多不同的存储方式。选择不同的存储引擎，底层在组织这张表的时候，存储方式就是不同的。oracle中也有这种机制，但是没有这个概念。名字不叫“存储引擎”。

\* 查看当前MySQL支持的存储引擎有哪些？

SHOW ENGINES\G当前MySQL支持9个存储引擎，当前MySQL的版本号是：5.5.36版本不同，支持的存储引擎不同。

\* 怎么指定存储引擎？

在建表语句当中，最后的位置可以指定存储引擎以及字符集。

CREATE TABLE `dept` (

`DEPTNO` int(2) NOT NULL,

`DNAME` varchar(14) DEFAULT NULL,

`LOC` varchar(13) DEFAULT NULL,

PRIMARY KEY (`DEPTNO`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

create table t\_test(id int);

CREATE TABLE `t\_test` (

`id` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8

通过以上测试，可以得知mysql默认采用的存储引擎是：InnoDB方式，默认字符集是utf8

create table t\_test1(id int) ENGINE=InnoDB DEFAULT CHARSET=utf8;

\* 常见的存储引擎？

MyISAM：

- 比较常用的一种存储方式。

- 采用MyISAM方式，一个表被分为三个文件：

xxx.frm：存储表结构

xxx.MYD：存储数据

xxx.MYI：存储索引

- 特点：可被转换为压缩、只读表来节省空间

- MyISAM这种存储引擎不支持事务。

InnoDB：

InnoDB存储引擎是MySQL的缺省引擎

特点：

– 每个InnoDB表在数据库目录中以.frm格式文件表示

– InnoDB表空间tablespace被用于存储表的内容

– 提供一组用来记录事务性活动的日志文件

– 用COMMIT(提交)、SAVEPOINT及ROLLBACK(回滚)支持事务 处理

– 提供全ACID兼容

– 在MySQL服务器崩溃后提供自动恢复

– 多版本（MVCC）和行级锁定

– 支持外键及引用的完整性，包括级联删除和更新

最大的特点：支持事务。但是处理速度不是最快的。

MEMORY：

这种存储引擎，数据没有存储到硬盘文件上，直接存储在mysql的内存当中。

特点：执行速度极快。

但是不支持事务，另外数据容易丢失，因为这个数据没有持久化。

之前低版本的mysql当中，这种存储引擎叫做HEAP引擎。

# 事务

【重点】

\* 事务对应的英语单词：transaction

\* 什么是事务，事务有什么作用？

事务是一个完整的不可再分的最小的工作单元。

一个完整的事务就是对应一个完整的业务流程。

\* 在SQL方面，支持事务机制的语句只有DML！

只有insert delete update才会支持事务机制，其它语句都不支持。

insert delete update在执行过程中都是操作的内存，还没有真正的修改文件，最终提交事务的时候，文件才会被修改。

\* 事务的应用案例：

我们从建行的账户A向账户B中进行转账，

A账户减去钱，B账户加上钱，显然是需要执行两条UPDATE语句的。

这两条UPDATE语句必须保证同时成功，或者同时失败，要想使用这种机制

就必须借助数据库的事务机制。

事务可以达到同时成功或者同时失败。

\* 事务的本质是什么？

事务的本质就是多个DML语句共同联合起来同时成功或者同时失败！

那么这个DML语句到底是多少个呢？不一定，具体需要看这个业务流程需要几条DML语句共同参与。

\* 事务涉及到的术语：

开启事务：start transaction

提交事务: commit

回滚事务: rollback

\* 无论是提交事务还是回滚事务，事务都会结束。

提交事务：都成功了。

回滚事务：其中的某一步失败了。

提交事务和回滚事务最终的数据都是安全的，有效的，合法的。

\* 事务的4大特性：

原子性：事务是最小的单元，不可再分。

一致性：同时成功或者同时失败。

隔离性：事务A和事务B之间具有隔离。

持久性：事务的保证，事务最终的结束依靠数据的持久化。（从内存持久化到硬盘文件）

ACID四个特征。

\* 重点研究隔离性：

事务A和事务B之间有隔离，隔离有级别，理论上有4个级别：

- 读未提交(Read Uncommited)：

1.这个最低级别是理论值，数据库一般最低起步2档。

2.对方事务没有提交的数据，在我当前事务当中能读取到。

3.存在的问题是：脏读现象（dirty read）

- 读提交(read commited)：

1.oracle默认的级别，也是最常用的隔离级别。

2.对方事务提交之后的数据，在我当前事务当中能读取到，没有提交的 读不到。

3.解决了什么问题：解决了脏读现象。

4.存在的问题是：不可重复读

- 可重复读(repeatable read)：

1.mysql默认

2.对方事务提交之后的数据在我当前事务中也读取不到。

3.解决了什么问题：解决了不可重复读。

4.存在的问题是：幻影读。存在数据假象。

- 序列化(serializable)：

1.事务不可并发，事务排队执行。

2.最高级别，解决了：数据假象。效率最低。最安全。

\* MySQL事务是自动提交。（默认情况下是事务自动提交机制。）

什么是自动提交？

只要执行“一条”DML语句，则自动提交一次。

怎么取消mysql的自动提交机制？

start transaction; // 开启事务。

提交事务：commit;

回滚事务：rollback;

测试隔离级别。

- 查看当前会话的隔离级别：SELECT @@session.tx\_isolation

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

| @@session.tx\_isolation |

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

| REPEATABLE-READ |

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

- 查看全局的隔离级别：SELECT @@global.tx\_isolation

- 设置全局的隔离级别：

set global transaction isolation level read committed; //读未提交

- 怎么测试？

开启两个窗口模拟 事务A 和 事务B

# 索引

（理论是重点，面试经常问）：中等重点。

什么是索引？

\* 索引是所有数据库当中都存在的机制。

\* 索引相当于一本书的目录。

\* 一本书的目录有什么作用？提高查询效率而存在的，所以索引的作用 也是提高查询/检索的效率。

\* 索引是给字段添加的。

\* 索引是一个对象，存储在硬盘文件当中的对象。xxx.MYI

怎么创建索引？怎么删除索引？

\* 我们程序员不需要关心“索引”在数据库底层创建的原理是什么。（一 本书的目录是怎样构建出来，程序员不需要管。）

\* 程序员应该主要关心：我使用什么sql语句可以创建索引。

\* mysql在进行DQL语句(select)查询的时候有两种扫描机制：

第一种：全表扫描机制。

第二种：根据索引进行检索。

我们有一张表：emp，表中有empno和ename字段，假设sql语句是这样写的：

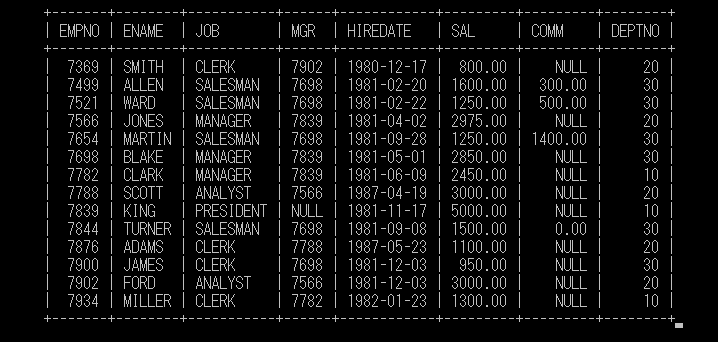
select \* from emp where ename = 'FORD';

执行的时候，先看ename上有没有添加索引，

没有索引的时候，MYSQL会对这个ename字段中所有的数据全 部扫描。

当这个ename字段上添加索引之后，MYSQL优先根据索引进行区域性扫描，缩小扫描的范围，提高检索速度。

\* 索引的实现原理：



每一条数据都有一个真实存储的物理地址

创建索引对象：

create index emp\_ename\_index on emp(ename);

emp\_ename\_index对象（索引对象）：索引会自动排序



select \* from emp where ename = 'JAMES';

DBMS发现ename字段上有索引，这个时候会找索引对象，找到J区，

进行区域性扫描，扫到JAMES之后，获取JAMES的物理地址： fdsafds45af45dsa4f5dsa78f9d7s8然后这条sql语句就变成了：select \* from emp where 物理地址 = 'fdsafds45af45dsa4f5dsa78f9d7s8';根据物理地址定位数据，根本不需要扫描表，直接通过硬盘的地址检索的。速度很快，和表的大小无关。

\* 创建索引的语法：

create index emp\_ename\_index on emp(ename);

\* 删除索引的语法：

drop index emp\_ename\_index on emp;

什么情况下我们会考虑给字段添加索引呢？

\* 条件一：数据量大。

\* 条件二：该字段是否经常出现在where子句中。（经常出现在where 子句当中的字段需要添加索引。）

\* 条件三：该字段很少参与DML操作。（字段发生改变之后，索引也是 需要去维护的，重新排序....）

多个字段联合起来可以添加一个索引吗？

可以的。

create index emp\_ename\_deptno\_index on emp(ename,deptno);

ename,deptno两个字段联合起来添加一个索引。

注意：主键自动添加索引，一般情况下能通过主键检索的尽可能通过主键查询。

create table t\_user(

id int primary key,

name varchar(255)

);

id字段上会自动添加索引对象。

根据主键查询检索速度是最快的。

**17.1、索引原理**

索引被用来快速找出在一个列上用一特定值的行。没有索引，MySQL不得不首先以第一条记录开始，然后读完整个表直到它找出相关的行。表越大，花费时间越多。对于一个有序字段，可以运用二分查找（Binary Search），这就是为什么性能能得到本质上的提高。MYISAM和INNODB都是用B+Tree作为索引结构

（主键，unique 都会默认的添加索引）

**17.2、索引的应用**

##### 16.1、创建索引

如果未使用索引，我们查询 工资大于 1500的会执行全表扫描



**什么时候需要给字段添加索引：**

**-表中该字段中的数据量庞大**

**-经常被检索，经常出现在where子句中的字段**

**-经常被DML操作的字段不建议添加索引**

**索引等同于一本书的目录**

**主键会自动添加索引，所以尽量根据主键查询效率较高。**

如经常根据sal进行查询，并且遇到了性能瓶颈，首先查看程序是否存算法问题，再考虑对sal建立索引，建立索引如下：

1、create unique index 索引名 on 表名(列名);

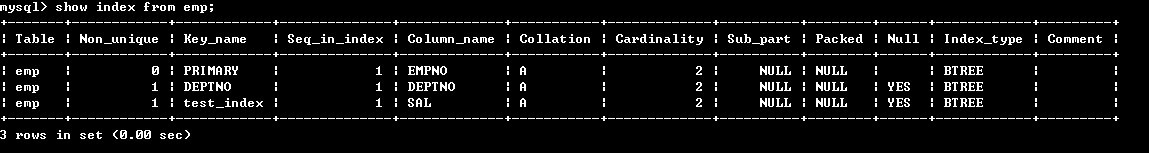
create unique index u\_ename on emp(ename);  
2、alter table 表名 add unique index 索引名 (列名);

|  |
| --- |
| create index test\_index on emp (sal); |



##### 16.2、查看索引

|  |
| --- |
| show index from emp; |

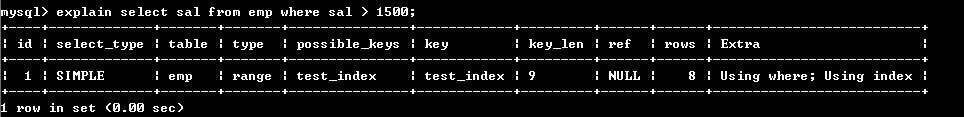


##### 16.3、使用索引

注意一定不可以用select \* … 可以看到type!=all了，说明使用了索引

|  |
| --- |
| explain select sal from emp where sal > 1500; |

条件中的sal使用了索引



如下图：假如我们要查找sal大于1500的所有行，那么可以扫描索引，索引时排序的，结果得出7行，我们知道不会再有匹配的记录，可以退出了。  
如果查找一个值，它在索引表中某个中间点以前不会出现，那么也有找到其第一个匹配索引项的定位算法，而不用进行表的顺序扫描（如二分查找法）。  
这样，可以快速定位到第一个匹配的值，以节省大量搜索时间。数据库利用了各种各样的快速定位索引值的技术，通常这些技术都属于DBA的工作。

##### 16.4、删除索引

|  |
| --- |
| DROP INDEX index\_name ON talbe\_name  ALTER TABLE table\_name DROP INDEX index\_name  ALTER TABLE table\_name DROP PRIMARY KEY  其中，前两条语句是等价的，删除掉table\_name中的索引index\_name。  第3条语句只在删除PRIMARY KEY索引时使用，因为一个表只可能有一个PRIMARY KEY索引，  mysql> ALTER TABLE EMP DROP INDEX test\_index;  删除后就不再使用索引了，查询会执行全表扫描。 |

# 视图：view

（中等重点程度。）

\* 视图在Oracle当中也是存在的，所有数据库管理系统当中都有视图这种机制。

\* 什么是视图？有什么作用？

视图：可以看做一张虚拟表。

视图：实际上可以理解为站在不同的角度去看待这些数据。

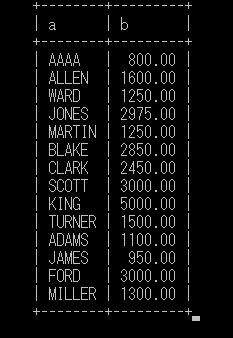
对视图虚拟表进行操作，实际上真正操作的是视图指向的原表数据。

作用1：

隐藏真实表的实现细节。

create view myview3 as select e.ename as a,e.sal b from emp1 e;

mysql> select \* from myview3;



作用2：

隐藏复杂的实现，对外提供简单的视图。

有一条非常复杂的sql语句，每一次写这个sql都比较费事。怎么办？

可以把这条dql语句当做一个视图创建出来，以后在java程序中面向视图进行查询，代码简化了很多。

复杂的DQL：select e.ename,d.dname from emp1 e join dept1 d on e.deptno = d.deptno;

create view myview2 as select e.ename,d.dname from emp1 e join dept1 d on e.deptno = d.deptno;

mysql> select \* from myview2;

\* 怎么创建视图？

create view myview1 as select ename,sal from emp1;

解释：

将select ename,sal from emp1; 这个查询语句以视图对象的方式创建出来。

注意：

视图对象只能对应一个DQL语句。

只能把DQL(select)语句当做一个视图对象创建出来。

\* 怎么删除视图？

drop view if exists myview1;

\* 对视图进行增删改查操作？

- 有了视图对象之后，可以通过视图对象对数据进行CRUD操作

CRUD：增删改查（公司当中程序员聊天的时候不会说增删改查的，都说CRUD操作。）

Create：增

Retrive：检索

Update：更新

Delete：删除

- mysql> select \* from myview1;



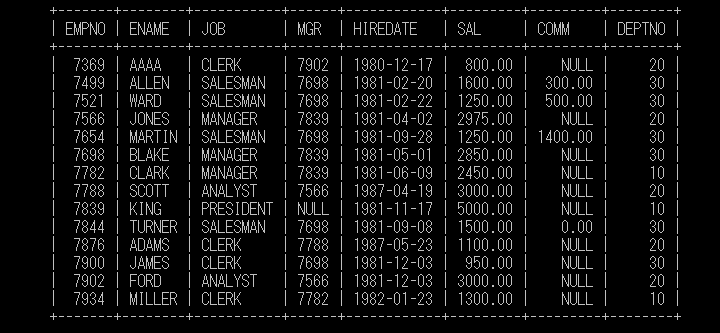
- 面向视图更新数据：mysql> update myview1 set ename = 'AAAA' where sal = 800;

mysql> select \* from myview1;



视图进行CRUD之后，视图指向的原表数据也会发生改变：

mysql> select \* from emp1;



实际上：面向视图进行CRUD的时候，视图只是一个逻辑名称，真正修改的还是视图所指向的原表物理数据。

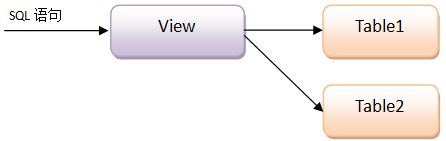
**18.1、什么是视图**

* 视图是一种根据查询（也就是SELECT表达式）定义的数据库对象，用于获取想要看到和使用的局部数据。
* 视图有时也被成为“虚拟表”。
* 视图可以被用来从常规表（称为“基表”）或其他视图中查询数据。
* 相对于从基表中直接获取数据，视图有以下好处：
  + 访问数据变得简单
  + 可被用来对不同用户显示不同的表的内容

用来协助适配表的结构以适应前端现有的应用程序

视图作用：

* 提高检索效率
* 隐藏表的实现细节【面向视图检索】



**18.2、创建视图**

如下示例：查询员工的姓名，部门，工资入职信息等信息。

|  |
| --- |
| select ename,dname,sal,hiredate,e.deptno from emp e,dept d where e.deptno  = e.deptno and e.deptno = 10; |

为什么使用视图？因为需求决定以上语句需要在多个地方使用，如果频繁的拷贝以上代码，会给维护带来成本，视图可以解决这个问题

|  |
| --- |
| create view v\_dept\_emp as select ename,dname,sal,hiredate,e.deptno from emp e,dept d where e.deptno  = e.deptno and e.deptno = 10; |
| create view v\_dept\_avg\_sal\_grade as select a.deptno, a.avg\_sal, b.grade  from (select deptno, avg(sal) avg\_sal from emp group by deptno) a, salgrade b  where a.avg\_sal between b.losal and b.hisal; /\*注意mysql不支持子查询创建视图\*/ |

**18.3、修改视图**

|  |
| --- |
| alter view v\_dept\_emp as select ename,dname,sal,hiredate,e.deptno from e  mp e,dept d where e.deptno = 20; |

**18.4、删除视图**

|  |
| --- |
| drop view if exists v\_dept\_emp; |

# 17、DBA命令：

\* java程序员重点掌握数据的导入和导出：这涉及到数据的备份。程序员平时会操作的。

导出命令（在dos命令窗口）：

导出当前数据库中所有的表：

mysqldump bjpowernode>D:\bjpowernode.sql -uroot -proot

导出指定表：

mysqldump bjpowernode emp>D:\bjpowernode.sql -uroot -proot

导入命令（在登录mysql之后执行）：

create database bjpowernode;

use bjpowernode;

source D:\bjpowernode.sql

\* 其它的命令：

例如：

创建用户、授权、撤销权限等，这些是DBA常用的，程序员可以不掌握。

新建用户

|  |
| --- |
| CREATE USER username IDENTIFIED BY 'password';  说明:username——你将创建的用户名, password——该用户的登陆密码,密码可以为空,如果为空则该用户可以不需要密码登陆服务器.  例如：  create user p361 identified by '123';  --可以登录但是只可以看见一个库 information\_schema |

授权

|  |
| --- |
| 命令详解  mysql> grant all privileges on dbname.tbname to 'username'@'login ip' identified by 'password' with grant option;   1. dbname=\*表示所有[数据库](http://www.2cto.com/database/" \t "_blank) 2. tbname=\*表示所有表 3. login ip=%表示任何ip 4. password为空，表示不需要密码即可登录 5. with grant option; 表示该用户还可以授权给其他用户  * 细粒度授权   首先以root用户进入mysql，然后键入命令：grant select,insert,update,delete on \*.\* to p361 @localhost Identified by "123";  如果希望该用户能够在任何机器上登陆mysql，则将localhost改为 "%" 。   * 粗粒度授权   我们测试用户一般使用该命令授权，  GRANT ALL PRIVILEGES ON \*.\* TO 'p361'@'%' Identified by "123";  注意:用以上命令授权的用户不能给其它用户授权,如果想让该用户可以授权,用以下命令:  GRANT ALL PRIVILEGES ON \*.\* TO 'p361'@'%' Identified by "123" WITH GRANT OPTION;   privileges包括：   1. alter：修改数据库的表 2. create：创建新的数据库或表 3. delete：删除表数据 4. drop：删除数据库/表 5. index：创建/删除索引 6. insert：添加表数据 7. select：查询表数据 8. update：更新表数据 9. all：允许任何操作 10. usage：只允许登录 |

回收权限

|  |
| --- |
| 命令详解  revoke privileges on dbname[.tbname] from username;  revoke all privileges on \*.\* from p361;  use mysql  select \* from user  进入 mysql库中  修改密码;  update user set password = password('qwe') where user = 'p646';  刷新权限;  flush privileges |

导出整个数据库

在windows的dos命令窗口中执行：mysqldump bjpowernode>D:\bjpowernode.sql -uroot -p123

导出指定库下的指定表

在windows的dos命令窗口中执行：mysqldump bjpowernode emp> D:\ bjpowernode.sql -uroot –p123

# 数据库设计的三范式

（重点内容）

\* 什么是数据库设计三范式？

数据库设计范式就是设计数据库表的时候需要遵守的规范。

常见规范有三个。

\* 数据库设计三范式，适用于所有的数据库管理系统。包括Oracle、MySQL、DB2、Sybase....

\* 数据库三范式这一块面试官经常问一个问题：

简单描述一下数据库设计三范式都有哪些？

\* 第一范式：任何一张表都应该有主键字段，并且每一个字段都是原子性不可再分。

简述：有主键，字段不可再分。

\* 第二范式（建立在第一范式的基础上。）：所有非主键字段完全依赖主键，不能产生部分依赖

产生部分依赖存在的问题是：数据冗余，空间浪费。（这也是为什么复合主键很少用的原因。）

简述：不能产生部分依赖

\* 第三范式（建立在第二范式基础上。）：非主键字段不能传递依赖于主键字段

简述：不能产生传递依赖。

\* 数据库设计范式的目的：减少数据的冗余，节省空间。

\* 表的设计：（最好的办法是背诵口诀）

一对一:

有两种实现方式：主键共享策略；外键唯一策略。

主键共享：

丈夫表husband

id(pk) name

------------------

1 a

2 b

3 c

妻子表wife

id(pk+fk) name

-------------------

1 x

2 y

3 z

外键唯一策略：【首先把“一对一”看成“一对多”，在多的表的外键上添加unique约束。】

丈夫表husband

id(pk) name

------------------

1 a

2 b

3 c

妻子表wife

id(pk) name hid(fk+unique)

------------------------------

1 x 1

2 y 3

3 z 2

一对多: 两张表，多的表加外键。

一个部门对应多个员工

部门表dept（一）

deptno(pk) dname

------------------------

1 销售部

2 研发一部

员工表emp（多）

empno(pk) ename deptno(fk)

-----------------------------------------------

1111 zs 1

2222 ls 1

3333 ww 2

多对多: 三张表，关系表两个外键。【多对多实际上是两个一对多组成的。】

学生表student

sno(pk) sname

--------------------

100 a

200 b

300 c

课程表course

cno(pk) cname

-------------------

11 x

22 y

33 z

学生选课表student\_course（关系表）

id(pk) sno(fk) cno(fk)

--------------------------------------

1 100 11

2 100 22

3 100 33

4 200 22

5 200 33

6 300 11

7 300 22

\* 在实际开发中，还是以满足客户的需求为准，不一定100%的遵守范式。

可能会拿冗余换速度。【作为面试的技巧。说出来。加分的。】

insert into dept1(deptno) values(11);

select \* from dept1;

# JDBC

## 19.1、JDBC介绍

1、学习JDBC，在eclipse当中直接创建“Java Project”即可。

2、JDBC是什么？

\* JDBC是Java语言连接不同数据库的一套API（应用程序编程接口）。

\* Java DataBase Connectivity（简称JDBC），Java语言连接数据库。

\* JDBC是SUN公司制定的一套类库，程序员通过这套类库的字节码(class)可以访问不同的数据库。

\* JDBC不是专门为MySQL准备的，也就是说以后java程序员只需要编写一套JDBC程序，这套程序

就可以操作MySQL数据库，代码不需要做任何改动，也可以操作Oracle数据库，这说明JDBC这套

类库是一套接口interface。

最重要：JDBC程序只写一份，可以连接到不同的数据上。

JDBC这套API的：

源码在哪？C:\Program Files\Java\jdk1.8.0\_101\src.zip 【java.sql.\*】

字节码在哪？C:\Program Files\Java\jdk1.8.0\_101\jre\lib\rt.jar【java.sql.\*】

帮助文档在哪？JDK的帮助文档当中就有JDBC的帮助信息【java.sql.\*】

JDBC的这套类库接口在java.sql包下。

3、API是什么？

\* 有同学说：API是帮助文档！（这句话是错误的。）

\* API就是一套类库，一套字节码，一套class文件！

\* JavaSE的API在哪里？

C:\Program Files\Java\jdk1.8.0\_101\jre\lib\rt.jar

\* API包括三个元素：

API源码

API字节码

API帮助文档

\* 对于java的API来说，任何一个API都有对应的：源码、字节码、帮助文档。

4、写一个模拟程序，来模拟一下JDBC，体验一下JDBC的本质是什么！！！！

JDBC本质就是一套接口。

5、JDBC接口的实现类谁写的？

各大数据库生产厂家的java程序员负责编写JDBC接口的实现类，

这些实现类就是一堆.class文件，他们发布这些class文件的时候，

把这些class文件打成jar包进行发布，程序员连接数据库的时候，

需要去官网下载jar包，这些jar包有一个特殊的名字叫做：驱动。

MySQL的驱动jar

Oracle的驱动jar

DB2的驱动jar

SqlServer的驱动jar。

.....

6、JDBC开发前的准备：

将驱动jar包配置到环境变量classpath当中。

7、JDBC开发六步：

第一步：注册驱动

【作用：告诉Java程序我们即将要连接的数据库是什么版本的】

第二步：获取数据库连接对象

【程序执行到此处表示JVM和MySQL数据库建立连接，通道打开。】

第三步：获取数据库操作对象

【数据库操作对象主要任务是执行SQL语句。】

第四步：执行SQL语句

【执行SQL语句】

第五步：处理查询结果集

【当SQL语句是一个DQL语句的话，是有查询结果的，需要处理查询结果。】

第六步：释放资源

【关闭所有资源。】

8、注册驱动的本质代码：

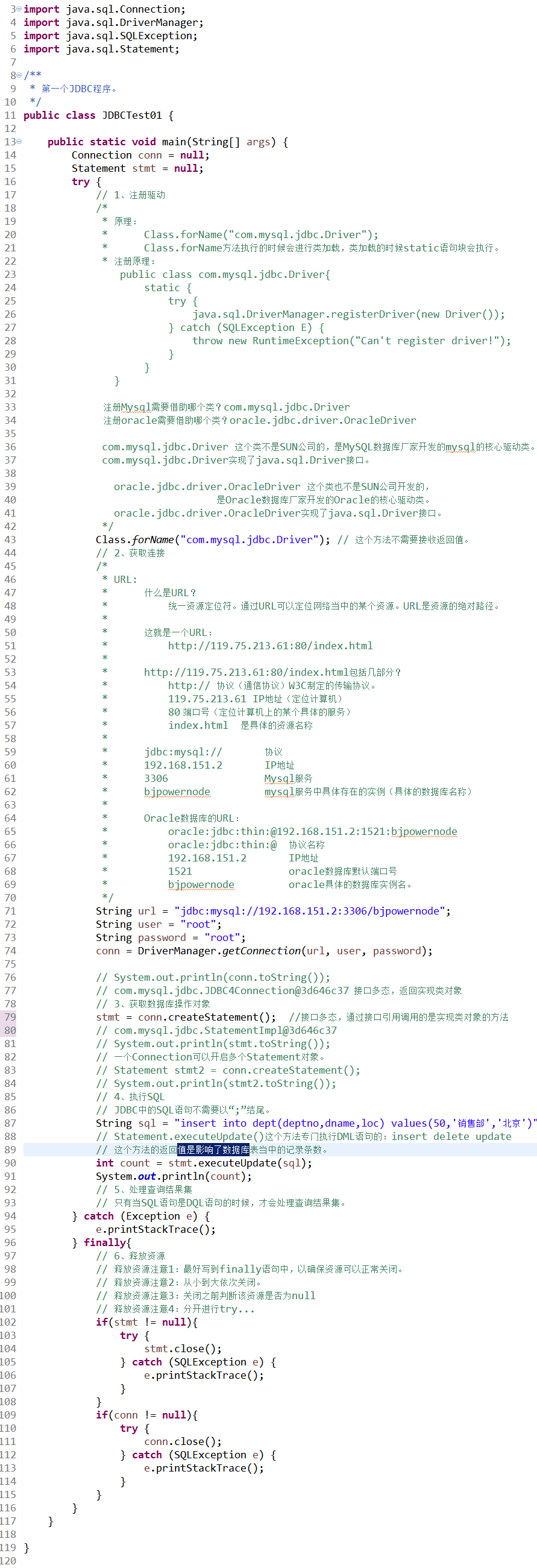
mysql驱动注册：

java.sql.DriverManager.registerDriver(new com.mysql.jdbc.Driver());

oracle驱动注册：

java.sql.DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

## 19.2、JDBC 6步



## 019.3、JDBC结果集



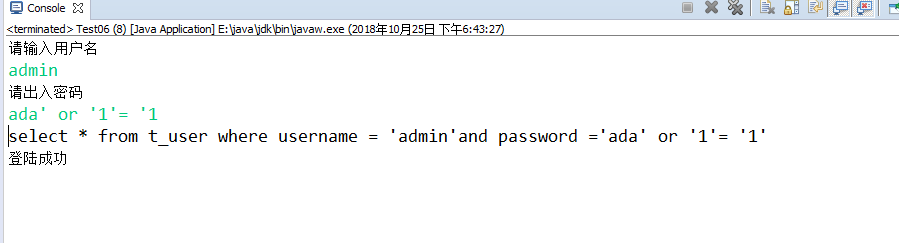
## 19.4、根路径读文件





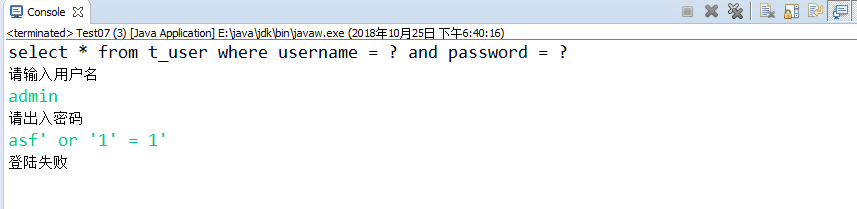
## 19.5、用户登录注入





## 19.6、SQL注入处理





## 19.7、Statement也有用

采用PreparedStatement的SQL :

select ename,sal from emp order by sal 'asc';

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

采用Statement的SQL :

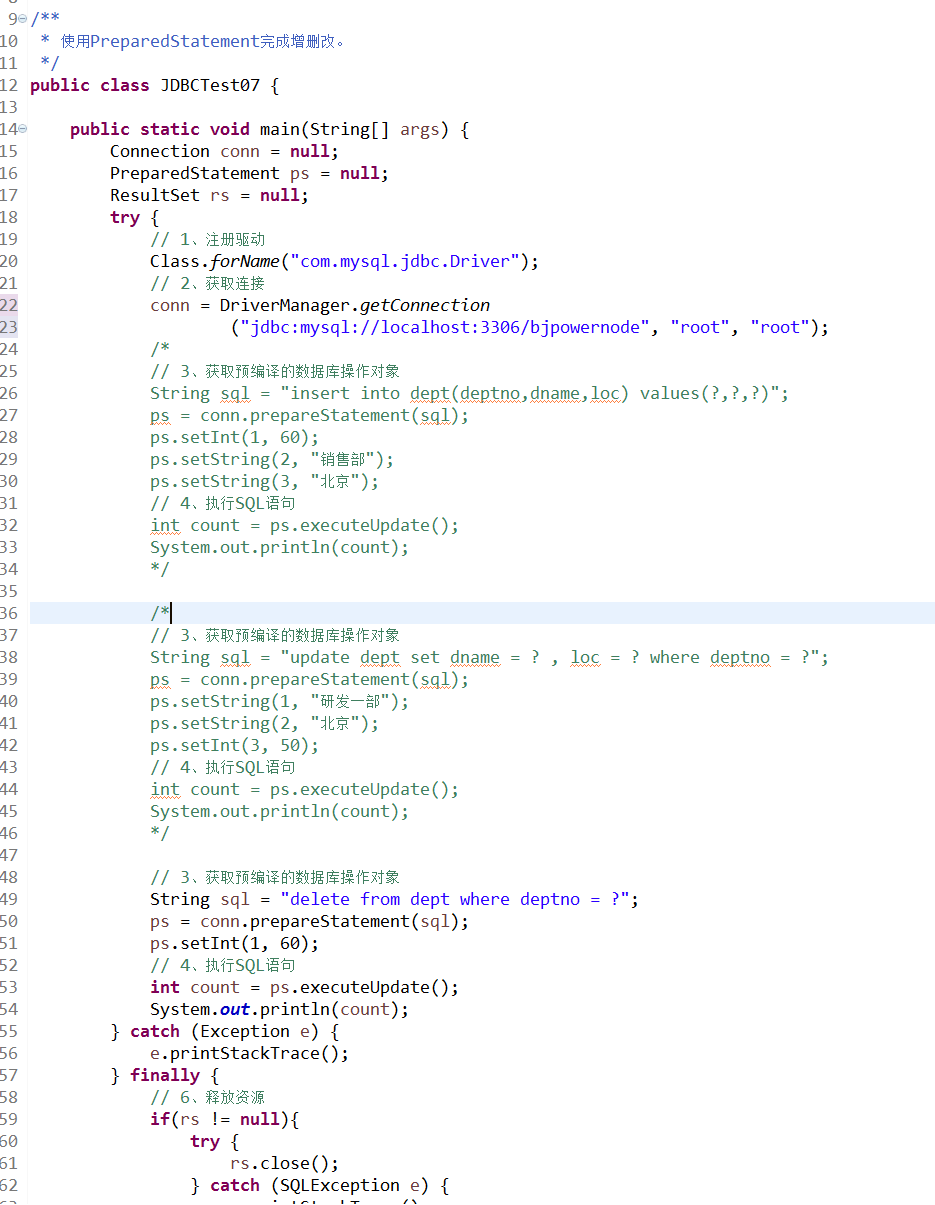
select ename,sal from emp order by sal asc;

Statement可以完成字符串的拼接。



## 19.8、PrepareStatement应用

### 1、Prepare完成增删改



### Prepare模糊查询



1. 区别
2. Statement用于字符串拼接
3. P是预编译之后给占位符传值。
4. S先编译后 p效率高，类型检查，类型安全

## 19.9、JDBC事务



## 19.10、行级锁





事务B加行级锁，所有对整张表的增删改都不能执行，

直到事务A提交结束事务。

# 注解Annotation

1、Annotation

\* 被称为注解类型、注释类型

\* 属于JavaSE当中的内容

\* 这里所描述的“注释类型”，其中的注释不是指在java源代码中编写 的注释信息。

不是这个：

// 单行注释

/\*

多行注释

\*/

/\*\*

\* javadoc注释

\*/

\* 这里所描述的“注释类型/注解类型/Annotation”是一种java语言中的 数据类型，当然它属于引用数据类型， “注释类型” 编译之后也是 生成xxx.class文件。

2、Annotation，这种注释类型在java编程中有什么用？

作用：标注的作用，标记的作用，标记给“其它机制”参考的。

其它的机制可以参考这个标记，有了这个标记应该怎么做，没有这个标 记又应该怎么做！

3、JDK有没有内置的自带的Annotation？

java.lang包下自带了3个注释类型：

java.lang.Override

java.lang.Deprecated

java.lang.SuppressWarnings

Override源代码：

public @interface Override {

}

Deprecated源代码：

public @interface Deprecated {

}

SuppressWarnings源代码：

public @interface SuppressWarnings {

String[] value();

}

4、关于java.lang.Override注解。

java.lang.Override，这个注解类型专门用来标注方法，被标注的方法

必须是重写父类的方法，该方法上有这个标记之后，编译器开始检测该

方法，当该方法不是重写父类的方法，编译器会报错。

@Override注解：

只能用来标注方法，不能标注其它元素。

Override注解只在编译阶段起作用，和运行期无关。

在方法上使用@Override注解：

该注解只被保留在java源代码当中，字节码当中没有这个代码。

5、注释类型都可以标注什么？

可以标注类

可以标注方法

可以标注局部变量

可以标注成员变量

可以标注....

可以出现在：类上、局部变量上、方法上、成员变量、接口上等....

6、注释类型怎么用？语法格式是什么？

@注释类型名

元素

例如：

@Override

public String toString(){

return "";

}

7、关于java.lang.Deprecated注解：

该注解标注的元素已过时。

这是一种提醒机制。

带有这个注解的元素表示已过时了，有更好的替代方案。

并且这个注解会生成到帮助文档当中。

这个已过时注解是给开发人员参考的。

8、关于java.lang.SuppressWarnings注解：

SuppressWarnings源代码：

public @interface SuppressWarnings { // 注解

// 注解的属性（当然该属性同时又可以当做方法用。）

// 该属性的名字叫做：value

// 该属性的数据类型是：String[]

String[] value();

}

有一条规则：

当一个“注解类型”中有属性的时候，在使用该注解的时候，属性 必须赋值。

不赋值，编译器报错。

怎么给注解的属性赋值？

语法格式：

@注释类型名(属性名=属性值,属性名=属性值,属性名=属性值....)

这样使用：

@SuppressWarnings(value={"all"})

记住：

当数组中只有一个元素的时候，大括号可以省略。

当属性名是value的时候，属性名可以省略。

@SuppressWarnings("all")

9、自定义注解（程序员自己写注解类型）

自定义注解语法格式：

public @interface 注释类型名{

属性1;

属性2;

// 每一个属性怎么定义？

属性的类型 属性名();

// 例如：

String value();

}

注解当中的属性类型只能是：

byte,short,int,long,float,double,boolean,char

Byte,Short,Integer,Long,Float,Double,Boolean,Character

String、枚举类型以及以上类型的所有数组形式。

在自定义注解的时候，需要使用其它的注解标注这个注解，专门

负责标注注解的注解被称为元注解(meta-annotation)，元注解有：

java.lang.annotation包下：

@Target

@Retention

@Documented

@Inherited

以上4个元注解是专门用来标注其它注解类型的。

10、关于元注解：@Target

这个注解主要的作用是：

指定被标注的注解只能出现的位置。

@Target(value = {ElementType.TYPE , ElementType.METHOD} )

public @interface MyAnnotation{

}

11、关于元注解：@Retention

这个注解主要的作用是：

指定被标注的注解的保存位置。

@Retention(value = RetentionPolicy.RUNTIME)

public @interface MyAnnotation { // 这个注解可以被反射机制读取到。

}

保持性策略：

RetentionPolicy.SOURCE 表示注解被保存到java源文件

RetentionPolicy.CLASS 表示注解被保存到class文件，但不会被反 射机制读取到。

RetentionPolicy.RUNTIME 表示注解被保存到class文件，并且可以被 反射机制读取到。

12、关于元注解：@Documented

@Documented 这个元注解标注的注解可以生成到帮助文档当中。

13、关于元注解：@Inherited

@Inherited 这个元注解标注的注解支持继承。

@Inherited

public @interface MyAnnotation{

}

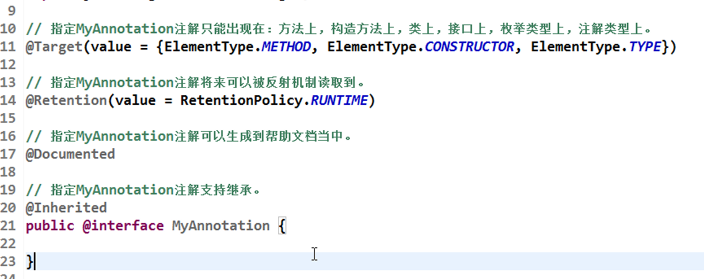
@MyAnnotation

public class A{

}

public class B extends A{

}



14、后期所学习的框架有：

Spring

SpringMVC

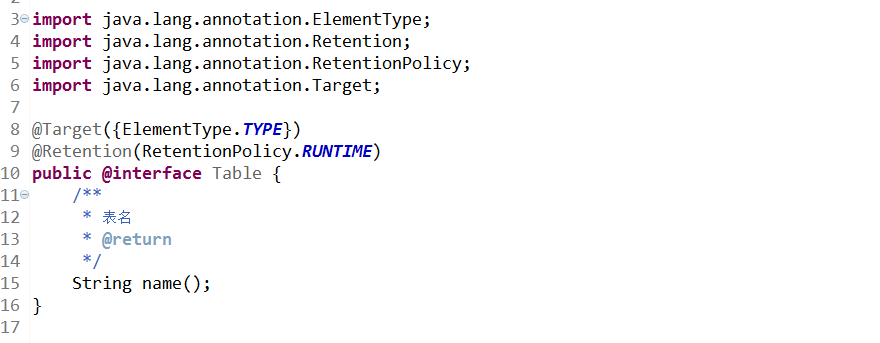
MyBatis

现在比较流行的JavaEE三大框架：SSM.

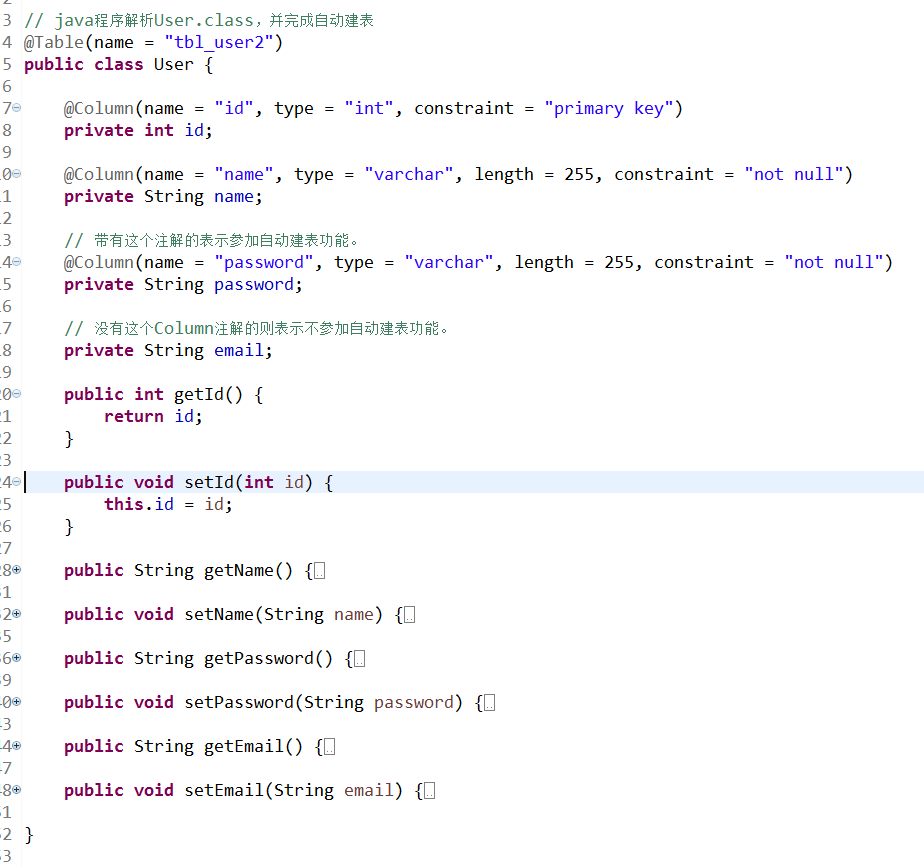
这些框架都自带了很多注解类型，每一个注解都有自己的作用。

现在必须理解注解在java开发中有什么作用？标记的作用。

自定义注解完成自动创建表









# 附32道练习题

1、取得每个部门最高薪水的人员名称

第一步：取得每个部门最高薪水（按部门编号分组找出薪水最大值）

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

第二步：将以上的表看做临时表t，t表和emp e表连接

select

t.deptno,t.maxsal,e.ename

from

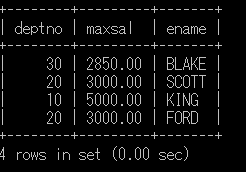
(select deptno,max(sal) as maxsal from emp group by deptno) t

join

emp e

on

t.deptno = e.deptno and t.maxsal = e.sal;



2、哪些人的薪水在部门的平均薪水之上

第一步：部门的平均薪水

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

第二步：和emp表进行表关联

select

e.ename,e.deptno,e.sal

from

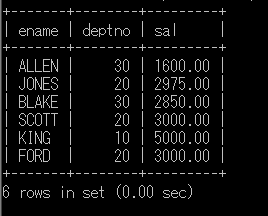
emp e

join

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

on

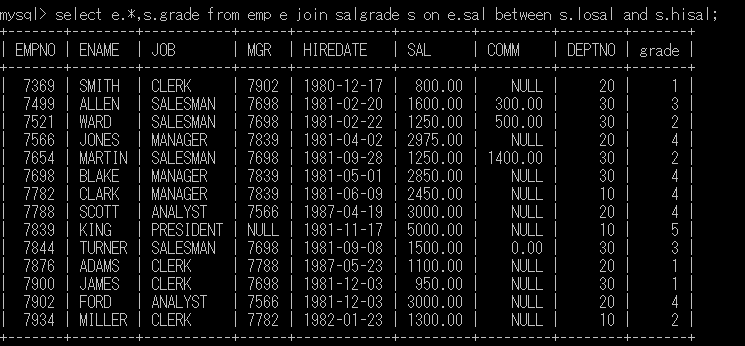
t.deptno = e.deptno and e.sal > t.avgsal;



1. 取得部门中（所有人的）平均的薪水等级（薪水等级的平均值）

第一步：找出每个人的薪水等级

select e.\*,s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal;



第二步：在以上查询结果的基础之上“继续”往下按照deptno分组，计算grade 的平均值。

select

e.deptno, avg(s.grade) as avggrade

from

emp e

join

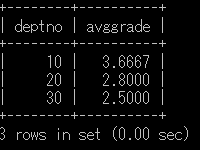
salgrade s

on

e.sal between s.losal and s.hisal

group by

e.deptno;



3+、取得部门中（所有人的）平均薪水的等级

第一步：找出每个部门的平均薪水（按照部门编号分组，求sal的平均值）

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

第二步：将以上查询结果当做临时表t，t表和salgrade s表连接

select

t.\*,s.grade

from

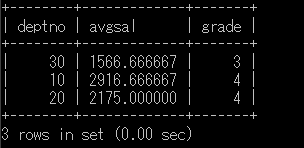
salgrade s

join

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

on

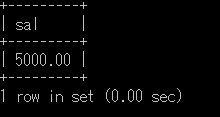
t.avgsal between s.losal and s.hisal;



1. 不准用组函数（Max），取得最高薪水(给出两种解决方案)

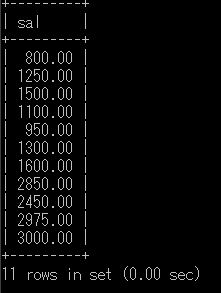
第一种方案：limit（降序排序取第一个）

select sal from emp order by sal desc limit 1;



第二种方案：表的自连接

select distinct a.sal from emp a join emp b on a.sal < b.sal;



select

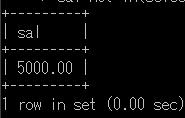
sal

from

emp

where

sal not in(select distinct a.sal from emp a join emp b on a.sal < b.sal);



1. 取得平均薪水最高的部门的部门编号（至少给出两种解决方案）

第一种方案：limit

第一步：每个部门的平均薪水

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

第二步：按照avgsal降序取第1个

select

deptno,avg(sal) as avgsal

from

emp

group by

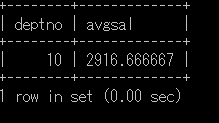
deptno

order by

avgsal desc

limit

1;



第二种方案：max函数。

select

deptno,avg(sal) as avgsal

from

emp

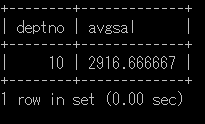
group by

deptno

having

avg(sal) =

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



1. 取得平均薪水最高的部门的部门名称

先进行emp e和dept d的表连接，连接之后把连接结果当做1张表，然后按照部门名称分组。

select

d.dname, avg(sal) as avgsal

from

emp e

join

dept d

on

e.deptno = d.deptno

group by

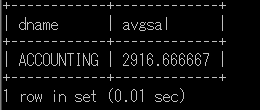
d.dname

order by

avgsal desc

limit

1;

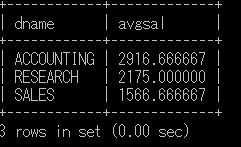


1. 求平均薪水的等级最低的部门的部门名称

第一步：按照部门名称分组，计算每个部门的平均薪水

select d.dname,avg(e.sal) avgsal from emp e join dept d

on e.deptno = d.deptno group by d.dname;



将以上查询结果看做临时表t，t表和salgrade s表连接

select

t.\*,s.grade

From

(select d.dname,avg(e.sal) avgsal from emp e join dept d

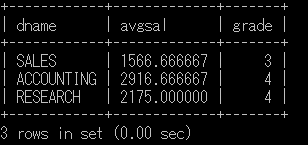
on e.deptno = d.deptno group by d.dname) t

join

salgrade s

on

t.avgsal between s.losal and s.hisal;



第二步：找出每个部门平均薪水的等级的最小值。

select grade from salgrade where (select avg(sal) as avgsal from emp group by deptno order by avgsal asc limit 1) between losal and hisal;



第三步：第一步和第二步结合。

select

t.\*,s.grade

from

(select d.dname,avg(e.sal) avgsal from emp e join dept d

on e.deptno = d.deptno group by d.dname) t

join

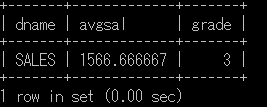
salgrade s

on

t.avgsal between s.losal and s.hisal

where

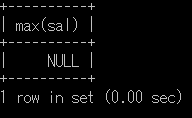
s.grade = (select grade from salgrade where (select avg(sal) as avgsal from emp group by deptno order by avgsal asc limit 1) between losal and hisal);



1. 取得比普通员工(员工代码没有在mgr字段上出现的)的最高薪水还要高的领 导人姓名

第一步：取得普通员工的最高薪水

select max(sal) from emp where empno not in(select distinct mgr from emp);



NULL:在mysql当中表示空，不是一个值，什么也没有。

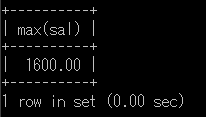
NULL:在数据库方面，大家可以把NULL当做“所有”，还可以当做“无穷大”。

重点1：not in不会忽略NULL。需要程序员手动去除NULL。

重点2：in会自动忽略NULL。不需要程序员手动去除NULL。

修改：

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



第二步：比1600高的员工信息。

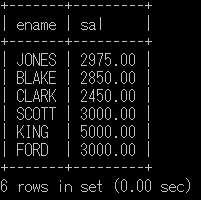
select

ename,sal from emp

where

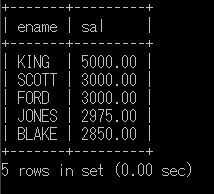
sal >

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



1. 取得薪水最高的前五名员工

select ename,sal from emp order by sal desc limit 5;



有bug: 第五名和第六名的工资如果相等

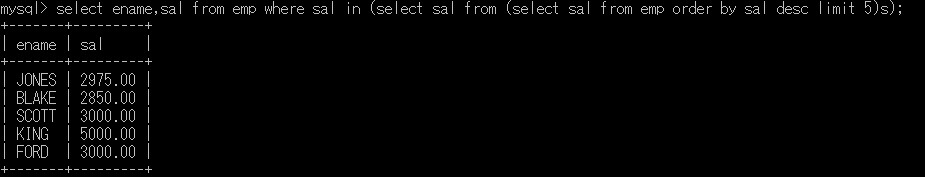
select ename,sal from emp where sal in (select sal from emp order by sal desc limit 5);



select ename,sal from emp where sal in (select sal from (select sal from emp order by sal desc limit 5)); //from后边如果是一个表，必须给表加别名

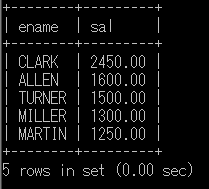


select ename,sal from emp where sal in (select sal from (select sal from emp order by sal desc limit 5)s);



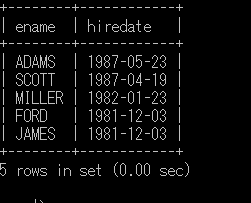
1. 取得薪水最高的第六到第十名员工

select ename,sal from emp order by sal desc limit 5,5;



1. 取得最后入职的5名员工

select ename,hiredate from emp order by hiredate desc limit 5;



12、取得每个薪水等级有多少员工

select

s.grade,count(\*)

from

emp e

join

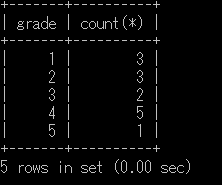
salgrade s

on

e.sal between s.losal and s.hisal

group by

s.grade;



14、列出所有员工及领导的姓名

select

a.ename as '员工', b.ename as '领导'

from

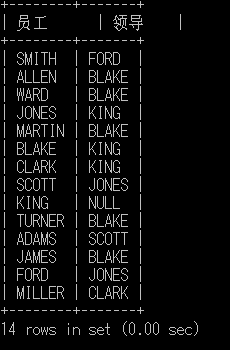
emp a

left join

emp b

on

a.mgr = b.empno;



15、列出受雇日期早于其直接上级的所有员工的编号,姓名,部门名称

select

a.empno,a.ename,a.hiredate,b.ename as leader,b.hiredate,d.dname

from

emp a

join

emp b

on

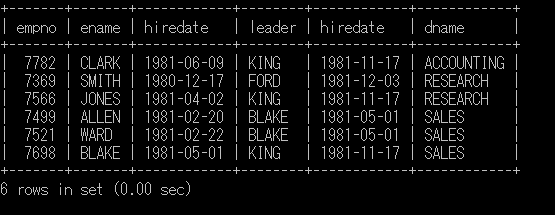
a.mgr = b.empno and a.hiredate < b.hiredate

join

dept d

on

a.deptno = d.deptno;



16、列出部门名称和这些部门的员工信息,同时列出那些没有员工的部门

select

d.\*,e.\*

from

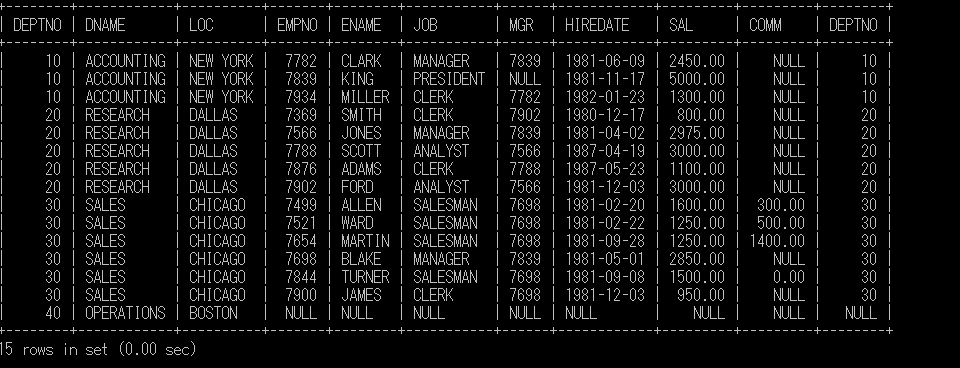
dept d

left join

emp e

on

d.deptno = e.deptno;



17、列出至少有5个员工的所有部门

select

deptno,count(\*)

from

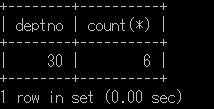
emp

group by

deptno

having

count(\*) > 5;



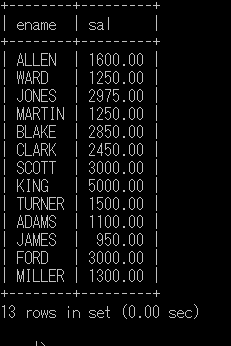
18、列出薪金比"SMITH"多的所有员工信息

Select

ename,sal from emp

Where

sal > (select sal from emp where ename = 'SMITH');



19、列出所有"CLERK"(办事员)的姓名及其部门名称,部门的人数

第一步：所有岗位是CLERK 的员工姓名及其部门编号,部门名称

select

e.ename,e.deptno,d.dname

from

emp e

join

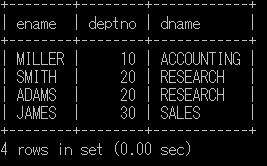
dept d

on

d.deptno = e.deptno

where

e.job = 'CLERK';



第二步：每个部门的员工人数

select

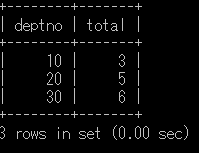
deptno,count(\*) total

from

emp

group by

deptno t



第三步：上边两个表关联

select

e.ename,e.deptno,d.dname

from

emp e

join

dept d

on

d.deptno = e.deptno

join

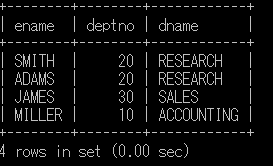
(select deptno,count(\*) total from emp group by deptno) t

on

t.deptno = e.deptno

where

e.job = 'CLERK';



20、列出最低薪金大于1500的各种工作及从事此工作的全部雇员人数

select

job,count(\*)

from

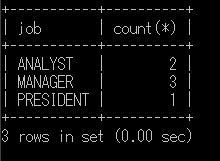
emp

group by

job

having

min(sal) > 1500;



21、列出在部门"SALES"<销售部>工作的员工的姓名,假定不知道销售部的部门编号

select

ename

from

emp

where

deptno = (select deptno from dept where dname = 'SALES');



22、列出薪金高于公司平均薪金的所有员工,所在部门,上级领导,雇员的工资等级

select

a.ename,d.dname,b.ename mgname,s.grade

from

emp a

left join

emp b

on

a.mgr = b.empno

join

dept d

on

d.deptno = a.deptno

join

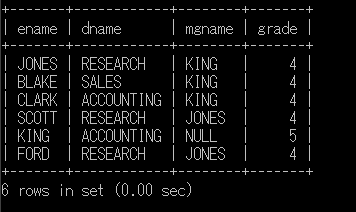
salgrade s

on

a.sal between s.losal and hisal

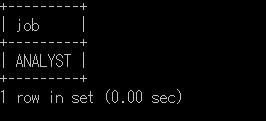
where

a.sal > (select avg(sal) from emp);



23、列出与"SCOTT"从事相同工作的所有员工及部门名称

select job from emp where ename = 'SCOTT';



select

e.ename,d.dname

from

emp e

join

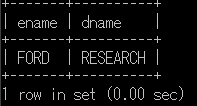
dept d

on

e.deptno = d.deptno

where

e.job = (select job from emp where ename = 'SCOTT') and e.ename <> 'SCOTT';



24、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

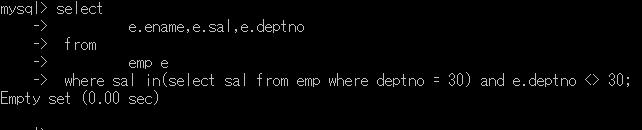
select

e.ename,e.sal,e.deptno

from

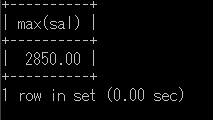
emp e

where sal in(select sal from emp where deptno = 30) and e.deptno <> 30;



25、列出薪金高于在部门30工作的所有员工的薪金的员工姓名和薪金.部门名称

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



select

e.ename,e.sal,d.dname

from

emp e

join

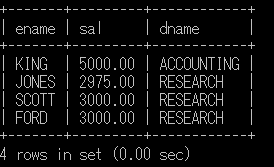
dept d

on

e.deptno = d.deptno

where

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



26、列出在每个部门工作的员工数量,平均工资和平均服务期限（外连接）

select

e.\*,d.\*

from

emp e

right join

dept d

on

e.deptno = d.deptno;

select

d.deptno,count(e.ename) as total,round(ifnull(avg(e.sal),0)) as avgsal, ifnull(round(avg(timestampdiff(year,e.hiredate,now()))),0) as avgyear

from

emp e

right join

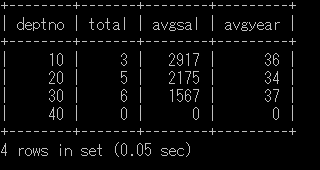
dept d

on

e.deptno = d.deptno

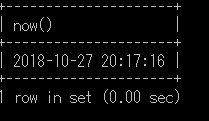
group by

d.deptno;

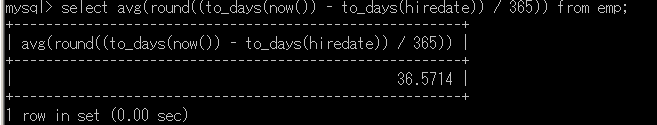


MySQL中获取系统当前时间：（了解now()函数，to\_days()函数可以将日期转换成天数。）

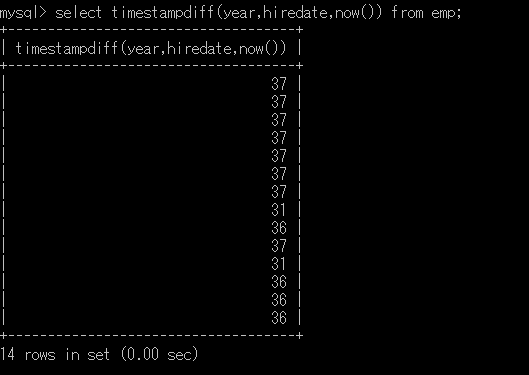
mysql> select now();



select avg(round((to\_days(now()) - to\_days(hiredate)) / 365)) from emp;



select timestampdiff(year,hiredate,now()) from emp;



27、列出所有员工的姓名、部门名称和工资

select

e.ename,d.dname,e.sal

from

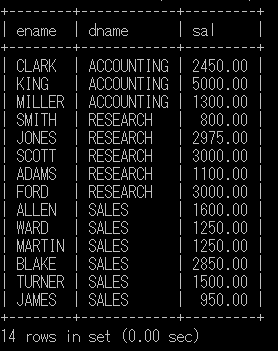
emp e

join

dept d

on

e.deptno = d.deptno;



28、列出所有部门的详细信息和人数

select

d.deptno,d.dname,d.loc,count(e.ename)

from

dept d

left join

emp e

on

d.deptno = e.deptno

group by

d.deptno,d.dname,d.loc;



错误结果：

select

d.deptno,d.dname,d.loc,count(\*)

from

dept d

left join

emp e

on

d.deptno = e.deptno

group by

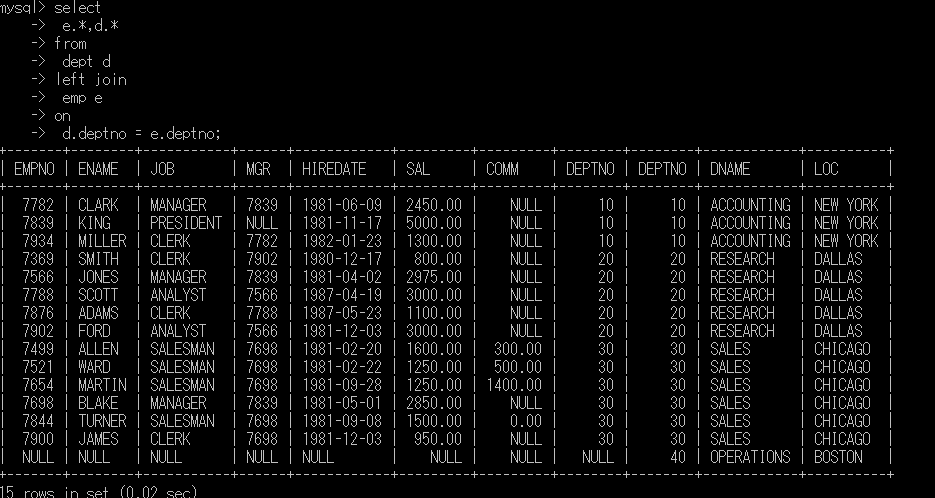
d.deptno,d.dname,d.loc;



原因：count(\*) 结果中所有记录的条数

Count(e.ename/........) 会忽略空。

Count(d.dname/loc/deptno) 结果等同cont(\*)

如下图：

1. 列出各种工作的最低工资及从事此工作的雇员姓名

select

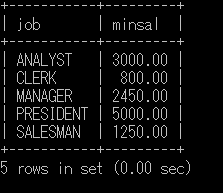
job,min(sal) as minsal

from

emp

group by

job;



select

e.ename,e.job,e.sal

from

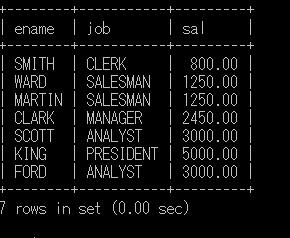
emp e

join

(select job,min(sal) as minsal from emp group by job) t

on

e.job = t.job and e.sal = t.minsal;



1. 列出各个部门的MANAGER(领导)的最低薪金

select

deptno,job,min(sal)

from

emp

Where

job = 'MANAGER'

group by

deptno,job;



31、列出所有员工的年工资,按年薪从低到高排序 ifnull函数。

select

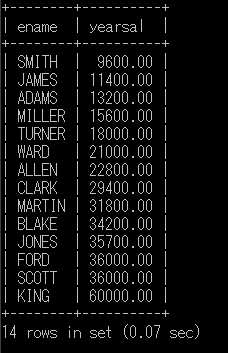
e.ename,(sal + ifnull(e.comm,0)) \* 12 as yearsal

from

emp e

order by

yearsal asc;



31、求出员工领导的薪水超过3000的员工名称与领导名称

select

a.ename,a.sal as '员工薪水',b.ename as leader,b.sal as '领导薪水'

from

emp a

join

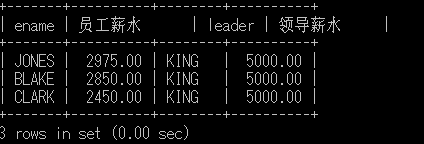
emp b

on

a.mgr = b.empno

where

b.sal > 3000;



33、求出部门名称中,带'S'字符的部门员工的工资合计、部门人数

select

d.dname,ifnull(sum(sal),0),count(e.ename)

from

dept d

left join

emp e

on

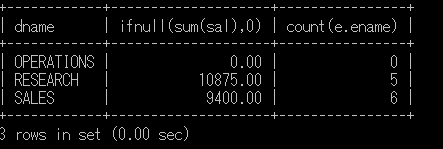
d.deptno = e.deptno

where

d.dname like '%S%'

group by

d.dname;



34、给任职日期超过30年的员工加薪10%

update emp set sal = sal \* 1.1 where timestampdiff(YEAR,hiredate,now()) > 30;；

13、面试题

有3个表S(学生表)，C（课程表），SC（学生选课表）

S（SNO，SNAME）代表（学号，姓名）

C（CNO，CNAME，CTEACHER）代表（课号，课名，教师）

SC（SNO，CNO，SCGRADE）代表（学号，课号，成绩）

S表

Sno sname

---------------------

1 学生1

2 学生2

3 学生3

4 学生4

C表

cno cname cteacher

----------------------------------

1 语文 张

2 政治 王

3 英语 李

4 数学 赵

5 物理 黎明

SC表

sno cno scgrade

--------------------------------

1 1 40

1 2 30

1 3 20

1 4 80

1 5 60

2 1 60

2 2 60

2 3 60

2 4 60

2 5 40

3 1 60

3 3 80

问题：

1、找出没选过“黎明”老师的所有学生姓名。

第一步；从c表中找出黎明老师的课程编号

select cno from c where cteacher = '黎明';

第二步：从sc表中找出选黎明老师的学生的学号

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

第三步：从s表中选择sno不在上表中的学生姓名

select

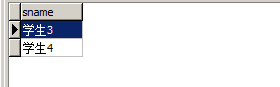
sno,sname

from

s

where

sno not in (select sno from sc where cno = ( select cno from c where cteacher = '黎明'));



2、列出2门以上（含2门）不及格(<60分的)学生姓名及平均成绩。

第一步：找出不及格门数超过两门的学生姓名

select

sc.sno ,s.sname

from

sc sc

join

s

on

sc.sno = s.sno

where

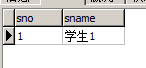
sc.scgrade < 60

group by

sc.sno

having

count(\*) > 2;



第二步:求出每个学生的平均成绩

select

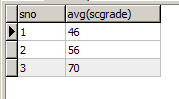
sno,avg(scgrade)

from

sc

group by

sno;



第三步：将上边两张表进行表关联

select

t1.\*,t2.avggrade

from

(....) t2

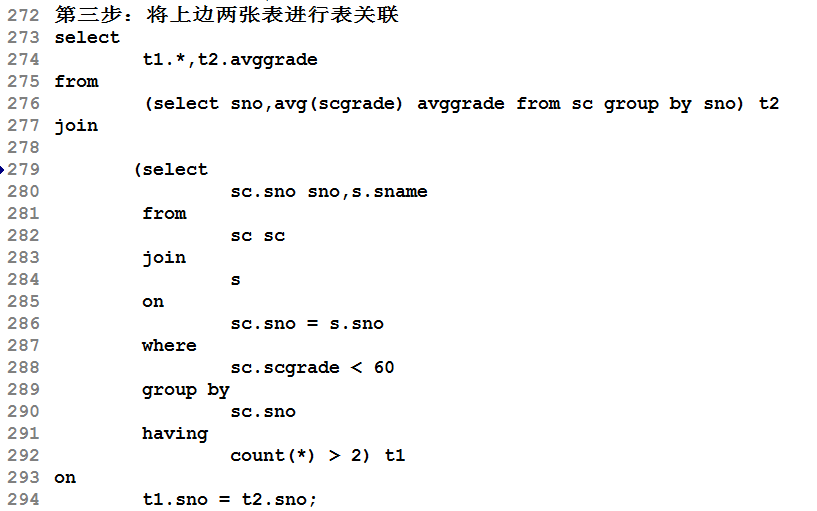
join

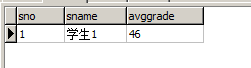
(....) t1

on

t1.sno = t2.sno;

完整代码：





3、既学过1号课程又学过2号课所有学生的姓名

第一步：找出学了1号课的学生

select sno from sc where cno = 1;

第二步：在上边的学生中找出选了2号课的学生

select sno from sc where

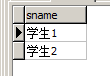
cno = 2

And

sno in (select sno from sc where cno = 1); t

第三步；在s表中找出学号在上表的学生姓名

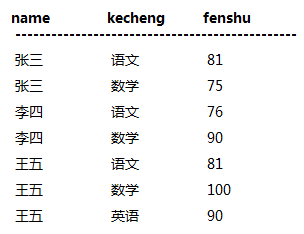
select sname from s where sno in (t);



第三步还可以写成：将t表和学生表进行关联。

**35**

**35、用一条SQL语句查询出每门课都大于80分的学生姓名；**

****

select

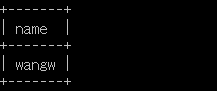
distinct name

from

t\_student

where

name not in( select name from t\_student where(fenshu < 80));



select

name

from

t\_student

group by

name

having

min(fenshu) > 80;

