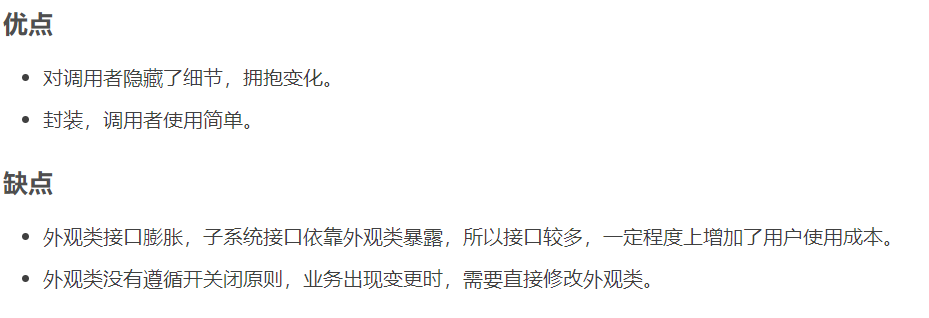
滴水穿石,每一滴都意义非凡   
冰冻三尺,每一度都有价值

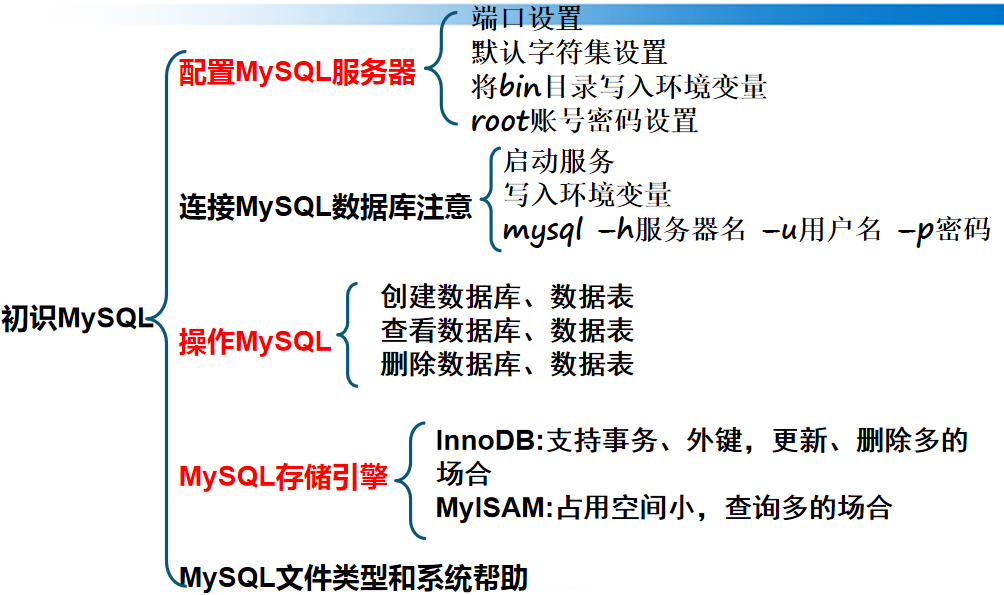
外观模式;

它是一个系统对外提供访问的一个统一的入口（外观类），该入口会整体糅合完成你所需要的功能或者业务。

这个入口隐藏了子系统完成所需要功能的复杂性，可以使得子系统对外来说更加容易调用。

提供一个统一的接口供使用者访问，隐蔽内部实现





# 在java中链接数据库

1. 导入数据库的连接包
2. 加载驱动(反射)

Class.forName(“com.mysql.jdbc.Driver”);

1. 获取数据库链接

String url = “jdbc:mysql://localhost:3306/数据库名”；

String username /password= 账号/密码

Connection con = DriverManager.getConnection(url,username,passwo..)

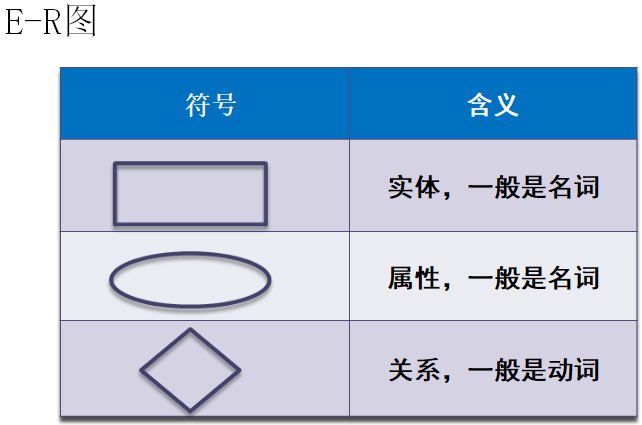
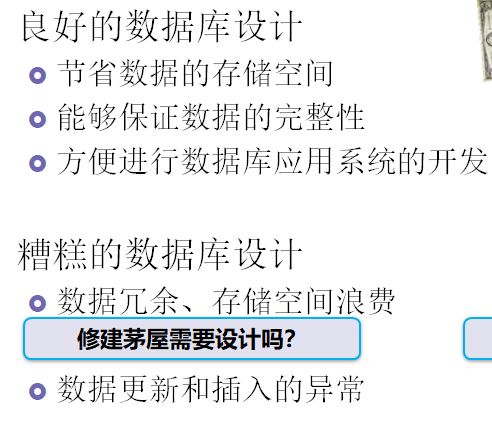
1. 获取预处理状态

PreparedStatement ps = con.prepare…

1. 得到结果集

ResultSet rs = ps.executeQuery();

1. 从数据库中读取数据(循环读取)
2. 关闭(倒着关闭)



E-R图：映射基数：实体关系图

一对一、一对多、多对一、多对多

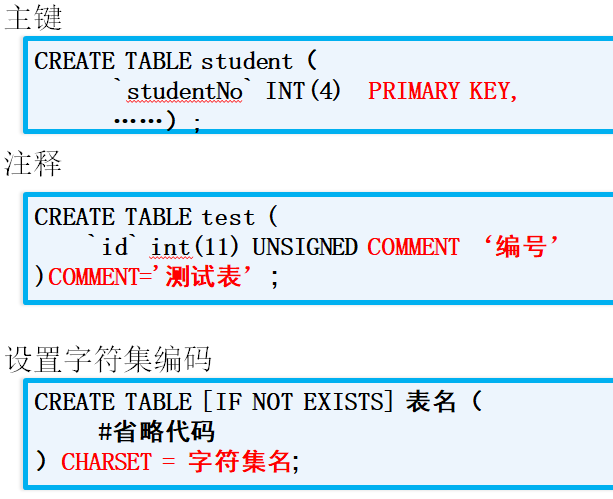
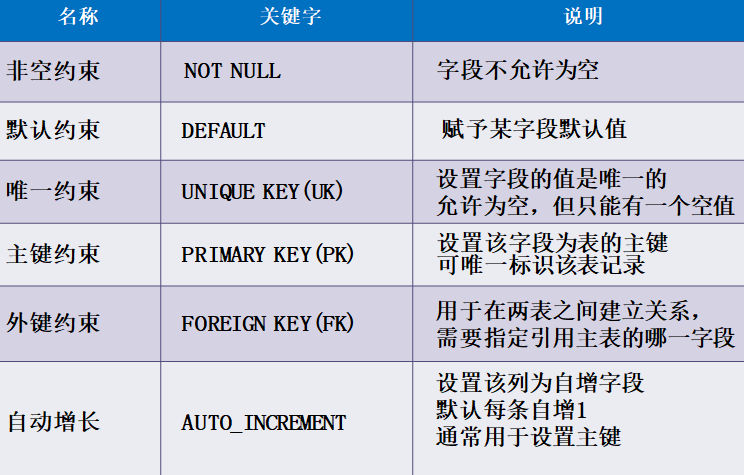
# 三大范式

第一范式：确保每列都是不可再分的最小数据单元(原子性)

第二范式：在一的基础上，每个表只能描述一件事

第三范式：满足前两的基础上，并且除了**主键列以外的列**都直接依赖于主键列

**字段的约束及属性**

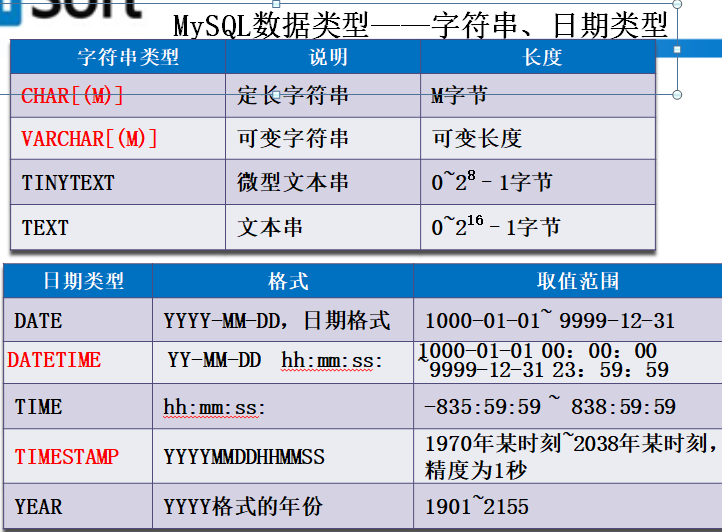


**MySQL存储引擎**

**使用myisam：不需要事务，空间小，以查询为主**

**使用innoDB：多删除、更新操作，安全性高，事务处理及并发控制**

**SQL语言包括：**



# Orcale语句类型

DML:(数据操作语言)用来插入、修改和删除表中的数据。Eg：insert、update、delete

DDL:(数据定义语言)在数据库中创建或删除数据库对象的操作。Eg;drop

DQL:(数据查询语言)用来对数据库中的数据进行查询。指select语句

DCL:(数据控制语言)用来控制数据库组件的存取许可、存取权限

**Mysql数据类型—数值类型**

Char性能好（某个字段长度固定时用），varchar可节省空间（常用）

# SQL操作语句

**auto\_increment:自动增长；comment:注释；primary key:主键**(每张表的标识)；int默认11位

**外键：foreing key(外键) references** 主键**表(主键id)**

## 1.创建一个UTF8的数据库

Create database 数据库名 character set 字符集(utf8);

## 2.使用数据库

**use 数据库名；**

## --3.删除数据库

**drop database 数据库名;**

## 9.删除表

**drop table 表名**

## **删除**(**delete,tuncate**)

(删除某个信息后再次插入值后，自增长跳过此序号，继续向下增长)

Delete from 表名 where name= ‘张三’;

(删除表中所有数据，序号并重新开始)

truncate table 表名;

## 删除字段(**drop**)

Alter table 表名 drop 字段名;

## 3.查看数据库的字符集

show create database 数据库名;

## 4.查看数据库列表(所有)

show databases;(最后带着S)

## 5.查看当前数据库名

Select database();

## 6.修改数据库的字符集(alter:修改)

alter database 数据库名 character set utf8;

**7.创建学生信息表(创建时必须切换到具体的数据库)**

**Create table 表名(**

**列名 数类型(name varchar(20),)**

**主键(primary key)，自增长(auto\_increment)**

**最后一列不用加逗号**

**);**

## 8.查看表结构

**desc 表名;**

## 10.查看建表语句以及字符集

Show create table 表名;

## 11.查看所有的表(多个s)

show tables;

## 修改表名

alter table **原有**表名 rename **修改**的名;

## 添加字段

alter table 表名 add 字段名、长度(password varchar(32)) not null;

## 修改字段名称

Alter table 表名 change 原字段名 修改字段名、属性 not null;

## 修改现有列的类型，长度约束

Alter table 表名 modify 字段名、属性 not null；

## 创建有外键的表

Create table 表名(

Sid：外键，主要是关联其他表的主键

......

sid int(4),

Foreign key(sid) references 要关联表名的表名(主键)

);

## 插入语句

* 1. 所有字段全部手工插入

insert into 表名values(字段：101,’小威’,’男’,20,’紫薇里’,123456789)

* 1. 录入部分字段(id是自动增长的，不插入);

Insert into 表名(字段名) values(‘名’,’男’,20.....);

* 1. 批量插入

Insert into 表名(字段名) values

(‘名’,’男’,20.....),

.........

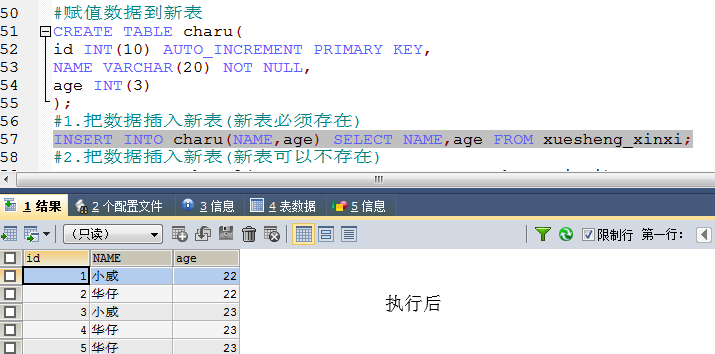
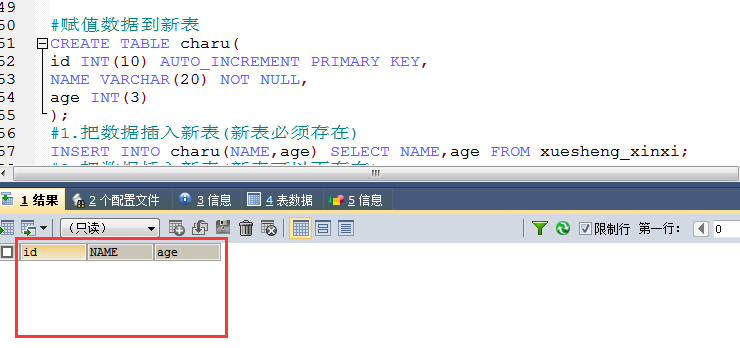
(‘名’,’男’,20.....);

## 赋值数据到新表

1. 把数据插入新表**(新表必须存在)**

insert into cha\_ru(name,age) select name,age from xuesheng\_xinxi;

(把xuesheng\_xinxi中的名字，年龄插入到cha\_ru表中)



1. 把数据插入到新表**(新表可以不存在)**

Create table charu(select name,age from xuesheng\_xinxi);

(同上，把xuesheng\_xinxi插入到charu表中)

## **修改**(根据id更新性别)

1. Update 表名 set sex=’男’ where id=2;

(id为2的性别修改成 ‘男’)

1. 不等于(<>)

Update 表名 set sex=’男’ where id<>3;

**(<>：不等于；除了id为3以外的所有性别都修改成 ‘男’)**

1. 同时成立(and)

Update 表名 set sex=’男’ where id=3 and name=’名字’;

(只有两个条件成立才能修改性别)

1. 只要有一个成立(or)

Update 表名 set sex=’男’ where id=3 or id=105;

(两个条件中有一个成立就可以)

1. 非(not)

Update 表名 set sex=’男’ where not id=3;

(和2的效果差不多)

## 查询表中所有信息

Select \* from 表名;

## 查询表中姓名和年龄

Select 姓名,年龄 from 表名;

## 查询在....之间(根据id查)(between:之间的)

Select \* from 表名 where id>101 and id<106;(**不包括**这两个数)

Select \* from 表名where id between 101 and 106;(**包含**着两个数)

## 查询表中所有的姓名和年龄

Select name,age from 表名;

## 通过as关键字起别名(as后面加，输出的中文)

Select name as 姓名 from 表名;

## 27.平均年龄(age)

Select avg (字段:age) as 别名 from 表名;

## 28.总人数(count)

Select count(字段:id) as 别名 from 表名;

Select count(\*) as 别名 from 表名;

## 29.最大年龄(max)

Select max(字段:age) as 别名 from 表名;

## 30.最小(min)

Select min(字段:age) as 别名 from 表名;

## 31.总年龄(sum)

Select sum(字段:age) as 别名 from 表名;

## 32.当前日期和时间

Select now();

* 1. 当前日期(curdate)

Select curdate();

* 1. 当前时间(curtime)

Select curtime();

## 34.查找年龄比2号大的(age 必须和子查询字段一样)

Select name , age from 表名 where age>(select age from 表名 where id = 2);

## 35.Exists后面是判断真假值的，若为真前面输出

## 36.聚合函数

1. sum:求和
2. min、max：返回指定列中的最大、小值
3. avg：求平均值
4. count：求指定列的记录总数
5. Median：首先根据e1表达式指定的列，对值进行排序；

若排序后，总记录为奇数，则返回排序队列中，位于中间的值；

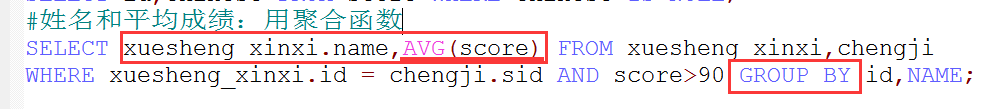
若排序后，总记录为偶数，则对位于排序队列中，中间两个值进行求平均，返回这个平均值；

## Order by :(排序)

## Group by:(分组)在聚合函数和其他字段一起使用时，后面要用group by (group by:去重)

**having必须跟随group by**

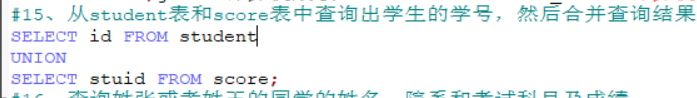
(1) 语句用于结合合计函数，根据一个或多个列对结果集进行分组



* 1. (当聚合函数和其他字段一起使用时用，group by 后面跟聚合函数前面的数：group by 比 order by 靠前)

## 37.Union

* 1. 用于合并两个或多个 SELECT 语句的结果集
  2. UNION 内部的 SELECT 语句必须拥有相同数量的列。列也必须拥有相似的数据类型。同时，每条 SELECT 语句中的列的顺序必须相同



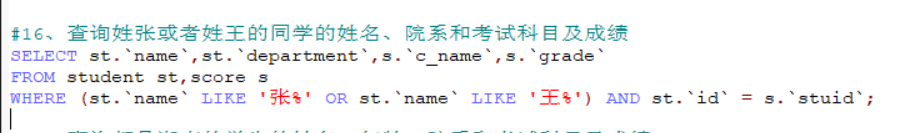
* 1. 如果允许重复的值，请使用 UNION ALL
  2. UNION 结果集中的列名总是等于 UNION 中第一个 SELECT 语句中的列名
  3. UNION ALL 命令和 UNION 命令几乎是等效的，不过 **UNION ALL 命令会列出所有的值**

## 38.笛卡尔积

* 1. 内连接:(inner join)
  2. 左外连接:(left join)
  3. 右外连接:(right join)

## 39.and,or

* 1. and:多个条件同时成立
  2. or:多个条件中有一个成立就可以
  3. 两者可结合起来使用(使用圆括号来组成复杂的表达式)



## 40.any, all

* 1. **Any：**(任何一个)

SELECT \* FROM score

WHERE english > ANY(SELECT math FROM score WHERE math>80);

* 1. **All：(所有)**

SELECT \* FROM score

WHERE english > ALL(SELECT math FROM score WHERE math>80);

## 41.求最大值(MAX)

* 1. Max：SELECT MAX(score) FROM chengji;
  2. SELECT \* FROM chengji c

WHERE NOT EXISTS(SELECT 1 FROM chengji c2 WHERE c.score < c2.score);

自连接查最大



## 42.模糊查询

* 1. %：SELECT \* FROM chengji WHERE sid LIKE '1%';

以1开头的，后面无所谓

* 1. \_(下划线):一个下划线代表一个字符

## 43.查询为空(is null)

SELECT \* FROM score WHERE chinese IS NULL;

## 44.排序(order by)

* 1. 正序：asc

SELECT \* FROM score ORDER BY id;

* 1. 倒序：desc(按照id倒序)

SELECT \* FROM score ORDER BY id DESC;

## 45.采用**exists(存在)检测是否**有人考试成绩达到86分以上，如果有，使用select语句按从大到小排，显示前五名的id和成绩

SELECT sid,score FROM chengji

WHERE EXISTS(SELECT score FROM chengji WHERE score>86)

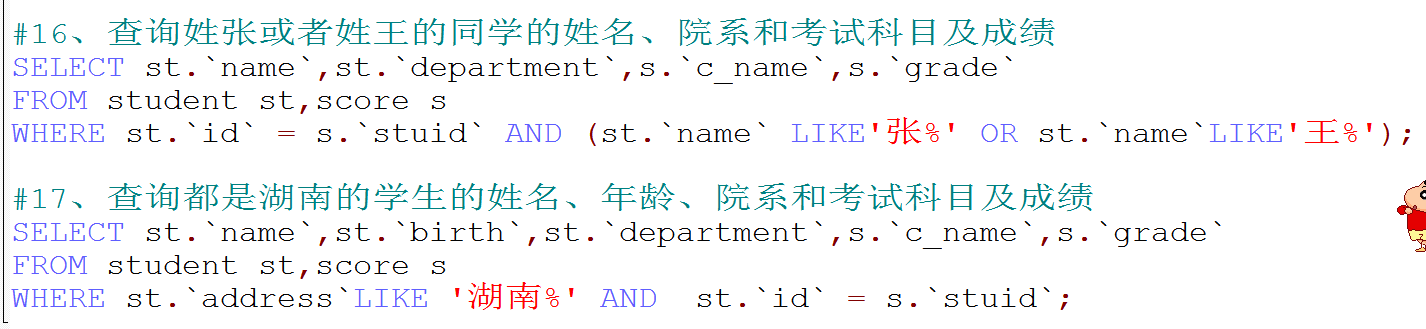
ORDER BY score LIMIT 4;

## 模糊查询(like)

查询时，字段中的内容并不一定与查询内容完全皮匹配

查询姓张的学生信息

**SELECT id,NAME FROM student WHERE NAME LIKE '张%';**



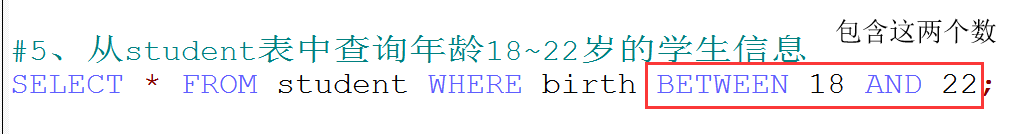
## 模糊查询(is null)

把某一字段中内容为空的记录查询出来

**SELECT id,NAME FROM student WHERE NAME is NULL;**

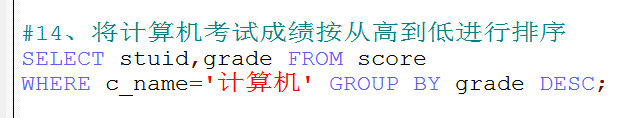
## 模糊查询(between..and)

把某一字段中内容在特定范围内的记录查询出来



## 数据查询-排序(asc:升序 desc：降序)

若多列排序先排前面的然后在排后面的

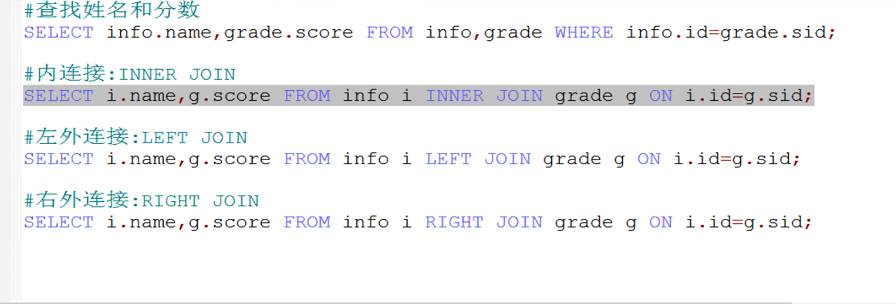


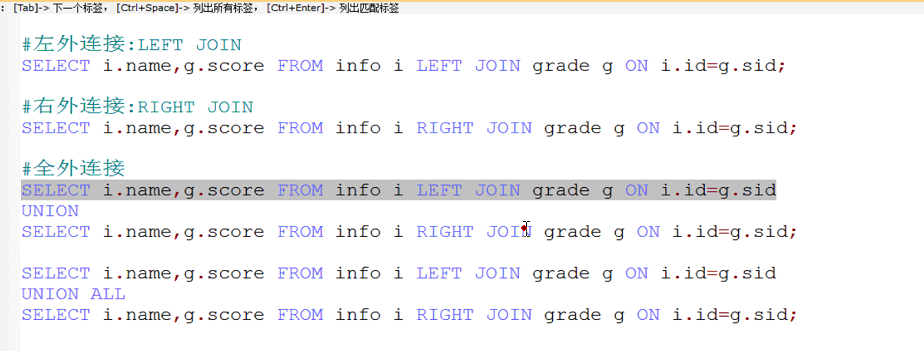
## 分组查询(group by)在36条下

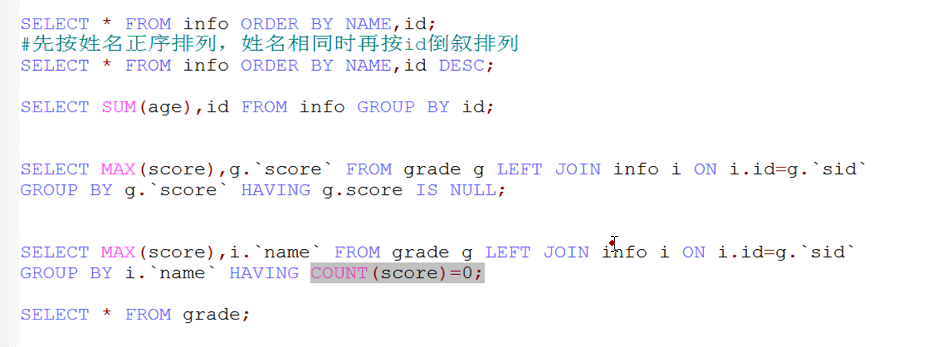
## 对比：where-->group by-->having-->

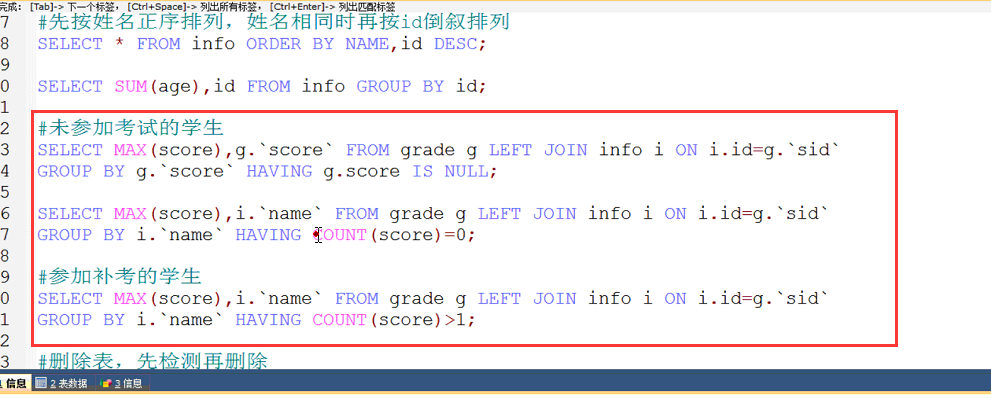
# 去重

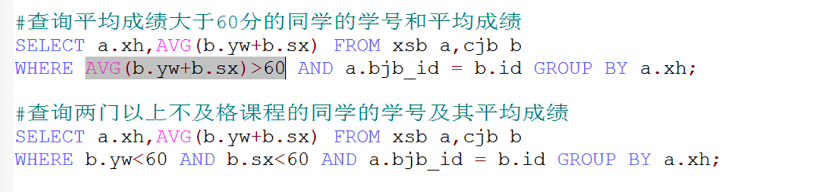
1. **distinct**
2. **group by**
3. **rowid**
4. 第四章

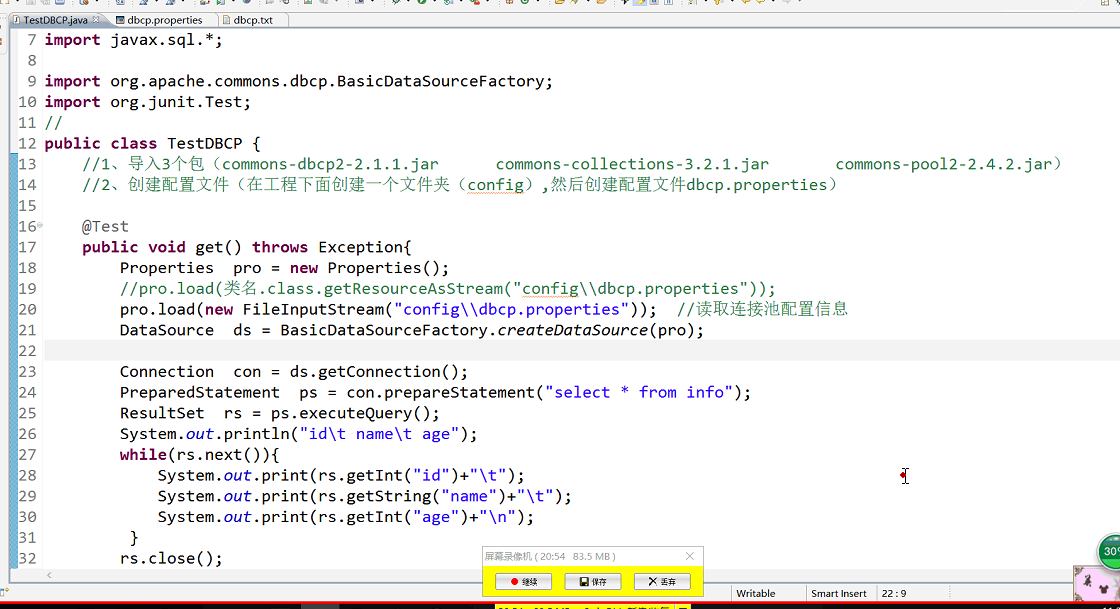












# 重启数据库：输入net start mysql

# 停止数据库：输入net stop mysql

# 事务

事务(**TRANSACTION**)是作为单个逻辑工作单元执行的一系列操作

## 什么是事务？事务有哪些特性？

事务：是数据库操作的**最小工作单元**，

是作为单个逻辑工作单元执行的一系列操作，这些操作一起向系统提交，要么都执行，要么都不执行；

事务是一组不可再分割的操作台

特性：原子性、一致性、隔离性、持久性

## 如何操作事务？

开始事务：begin或者start transaction

提交事务:commit

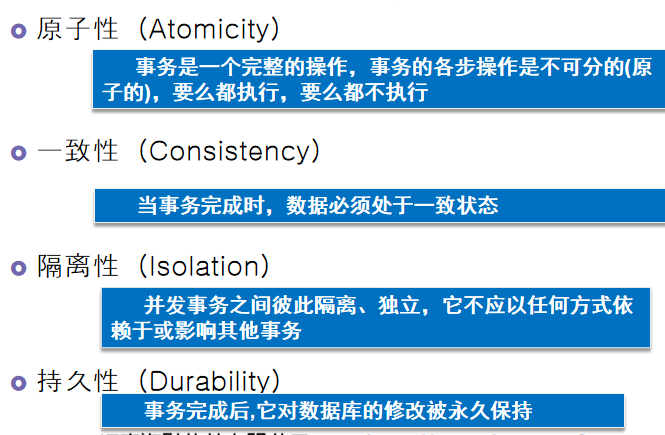
回滚事务:rollback

默认情况下，每条单独的SQL语句视为一个事物

关闭默认提交状态后，可手动开启、关闭事务(ACID)



特性(ACID)



用途

1. 筛选表中的行
2. 防止未经许可的用户访问敏感数据
3. 降低数据库的复杂程度
4. 将多个物理数据库抽象为一个逻辑数据库

设置自动开启和关闭

## 回滚点：savepoint

## 创建视图(view)

提高查询效率，防止用户访问敏感数据，为不同用户提供不同视图

Create view view\_name as

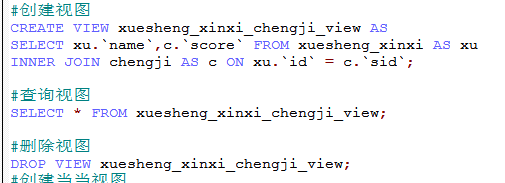
<select 语句>

## 查看视图

Select 字段..from view\_name;

## 删除视图

Drop view view\_name;



## 索引(index)

是一种优先组合数据的方式，为快速查找到指定数据

作用：(1)大大提高数据库的检索速度

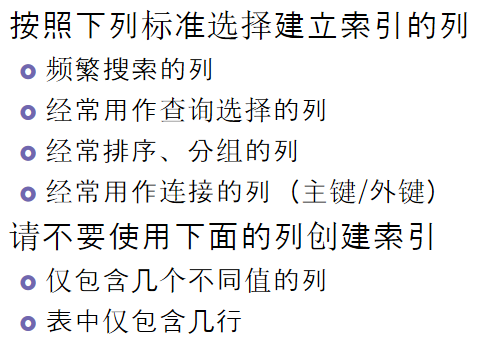
1. 改善数据库性能

按存储类型分类：

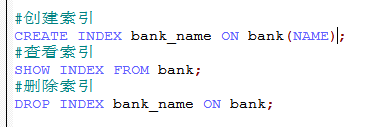
1. 树索引：InnoDB、MyISAM均支持

哈希索引

## **索引指导原则**



## 创建索引



Bank:表 bank\_name:表中字段

删除表时，该表索引同时会被删除

# 事务、并发控制

## **JDBC如何操作事务？**

Connection . setAutoCommit(false); // 开启事务(start transaction)

Connection . rollback(); // 回滚事务(rollback)

Connection . commit(); // 提交事务(commit)

## 在JDBC中使用事务

con.setAutoCommit(false):取消自动提交

con.commit();提交事务(整体提交)

con.rollback() 回滚所有操作(撤销以上的操作，将数据恢复为执行前状态)

## 并发控制

数据库并发就是不同的事务对同一部分数据执行操作

**并发易导致的问题**：

1. **脏读数据**：一个事务读取了另一个事务未提交的数据
2. **不可重复读**：在一个事务内读取表中的某一行数据，多次读取结果不同。一个事务读取到了另一个事务提交后的数据
3. **幻读**：在一个事务内读取到了别的事务插入的数据

脏读是读取前一事务未提交的脏数据，不可重复读是重新读取了前一事务已提交的数据

## 在JDBC中，通过connection设置事务隔离级别来对并发进行控制

事务隔离级别就是事务执行是受打扰的程度，不同隔离级别对应不同的干扰程度，**隔离级别越高，数据一致性越好，但并发性越差，效率越低**

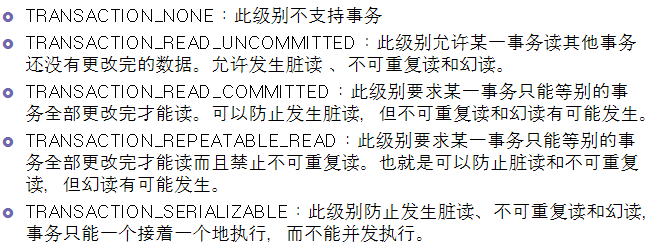
## 查看和设置隔离级别的方法

**getTransactionlation():查看隔离级别**

**setTransactionlation()：设置隔离级别**

## **Connection中的事务隔离级别**

**读未提交（READ UNCOMMITTED）**：未提交读隔离级别也叫读脏，就 是事务可以读取其它事务未提交的数据。  
**读已提交（READ COMMITTED）**：在其它数据库系统比 如 SQL Server 默认的隔离级别就是提交读，已提交读隔离 级别就是在事务未提交之前所做的修改其它事务是不可见的。  
**可重复读（REPEATABLE READ）**：保证同一个事务中的多次相同的查 询的结果是一致的，比如一个事务一开始查询了一条记录然后 过了几秒钟又执行了相同的查询，保证两次查询的结果是相同的， 可重复读也是 mysql 的默认隔离级别。  
**可串行化（SERIALIZABLE）**：可串行化就是保证读取的范围内没有新 的数据插入，比如事务第一次查询得到某个范围的数据，第二次 查询也同样得到了相同范围的数据，中间没有新的数据插入到 该范围中。



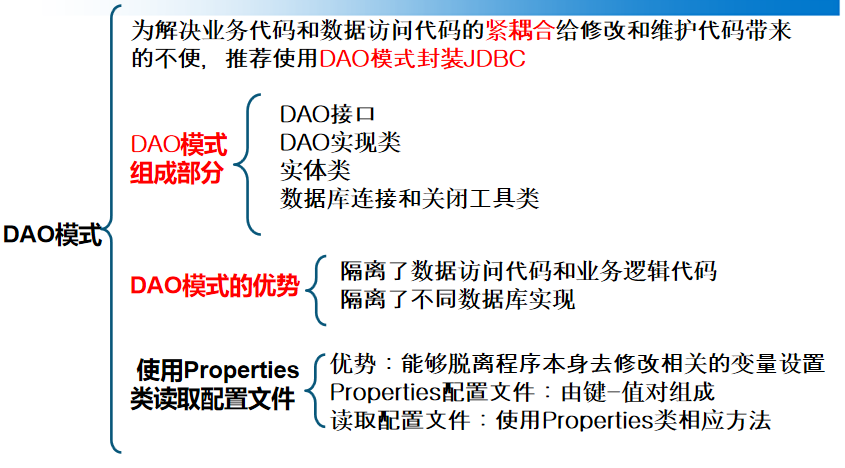
## JDBC异常和警告

通常用SQLException和SQLWarning表示异常和警告

### SQLException类

### 2.SQLWarning类

# Dao模式



## 持久化实现方法：

持久化：将程序中的数据在瞬时状态和持久状态间转换的机制

**实现：**数据库、普通文件、XML文件

## **为什么进行JDBC封装**

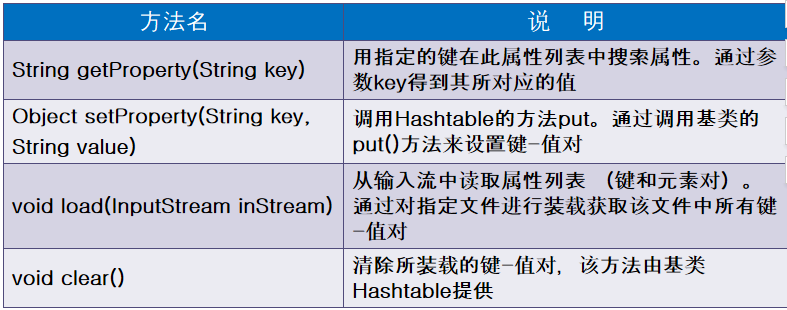
采用**面向接口编程**，可降低代码间的耦合性

1. 隔离业务逻辑代码和数据访问代码
2. 隔离不同数据库的实现
   1. 定义实体类：包含相关属性；实现属性的getter/setter
   2. 实现类的方法
   3. 将通用的操作(打开、关闭连接等)封到工具
   4. 工具类：增删改的通用方法
   5. 是先烈实现接口并继承数据库工具

## DAO模式的组成部分

1. DAO接口
2. DAO实现类
3. 实体类
4. DAO实现类的父类(BaseDao)

## 读取Properties类配置文件



# 批处理

## 实现方法：两种

1. Statement.addBatch(sql)
2. Preparestatement.addBatch()：批处理

## 两种批处理的比较

**Statement:**优：可以向数据库发送多条不同SQL语句

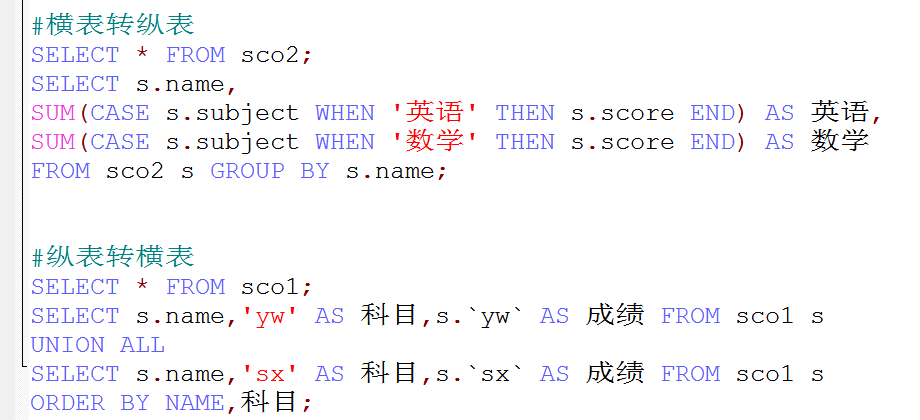
缺点：SQL语句没有预编译；

**preparedStatement:**优：发送的是预编译后的SQL语句，执行效率高

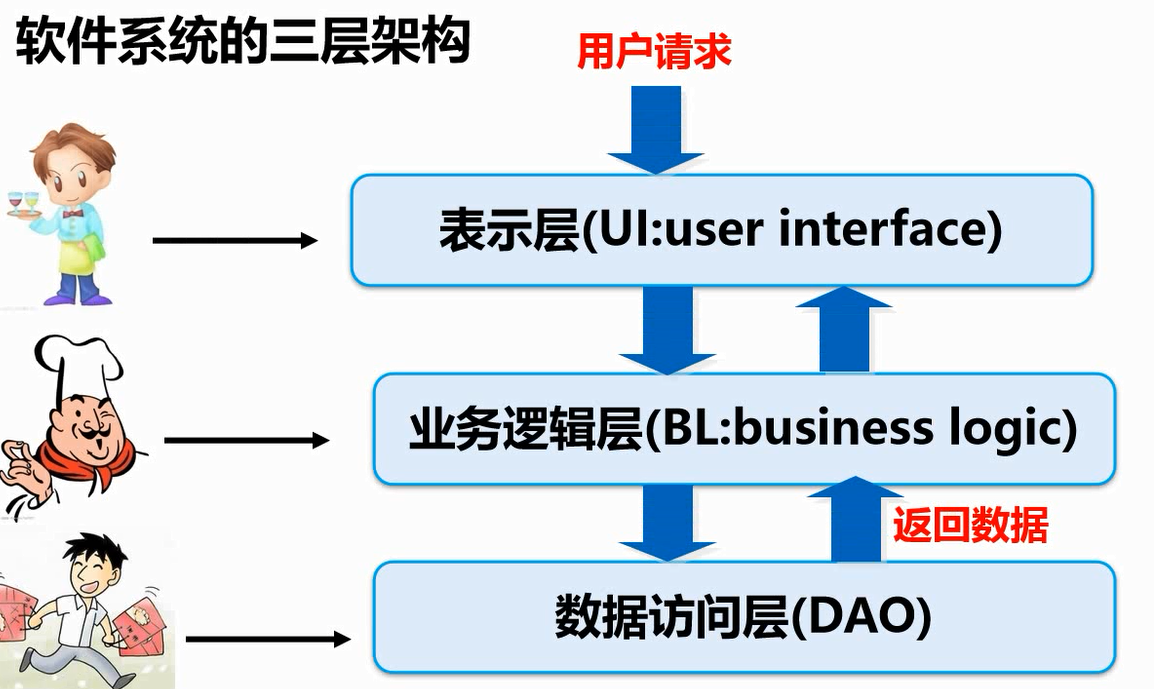
缺：只能应用在SQL语句相同，但参数不同的批处理中

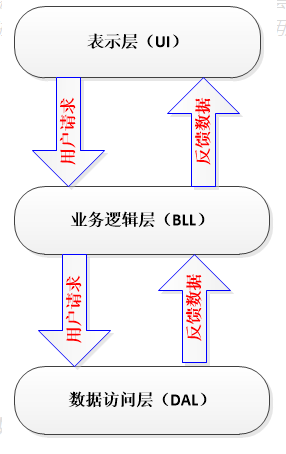
Ddl dml

# 横表纵表互转



# 三层架构





**三层架构：增删改，返回值都是int类型的，查的时候返回对象**

参数是对象

DAO实现类，先继承连接数据库的类然后实现DAO接口

**顺序**：服务层-->DAO层-->数据访问层