一、sql语法

1、安装数据库

2、关系数据库的基本概念和mysql基本命令

数据库管理系统，Database Management System ，简称dbms

dbms是所有数据的知识库，他负责管理数据的存储、安全、一致性、并发、恢复和访问灯操作。有一个数据字典（系统表）用于存储他拥有的每个事物的相关信息，列入：名字、结构、位置、类型等，这种关于数据的数据也被称为元数据（metadata）

按时间顺序出现了一下几种数据库

网状型

层次型

关系型

面向对象

对于关系数据库而言，存储数据单元就是表。数据表是存储数据的逻辑单元，可以把数据表认为是由行列组成的表格，其中一行也成为一条记录，一列称为一个字段。为数据库建表，需指定表有多少列，每列的数据类型，以及约束条件是啥子。

mysql命令（命令以；结尾）

查看当前实例下包括哪些数据库，show databases；

删除数据库：drop database test2；

操作数据库，需要先需要先指定数据库：use 数据库名;

查询当前数据下的表：show tables;

查看表结构：desc 表名;

3、sql语法基础

sql，Strucrtured QueryLanguage，结构化的查询语言。

sql是操作和检索关系数据库的表准语言。标准的sql可以操作任何关系型的数据库

（1）在数据库中检索信息（2）对数据库中数据进行更新（3）改变数据库结构（4）更改数据库安全设置（5）对用户授权

标准sql语句类型

（1）查询语句主要由select关键字完成

（2）DML（Data Manipulation language，数据库操作语言）主要由insert、update、delete三个关键字

（3）DDL（Date Definition language数据库定义语言）主要由create，alter，drop，truncate四个关键字组成。

设计表工具 power designer

（4）DCL（Date Control language，数据库控制语言）主要由grant、revoke两个关键字

（5）事物控制语句，主要由commit、rollback和savepoint三个关键字组成

SQL语句大小写不敏感

定义表名、列名、索引的规则如下

（1）标识符以字母开头（2）标识符可以包含字母，数字、特殊字符（# ,\_ ,$）（3）不要使用数据库的保留字和关键字（4）在同一个数据库的同一个模式下，表名不应该重复

DDL语句

创建（create）修改（alter）删除（drop）数据库对象

常见的数据库对象

|  |  |  |
| --- | --- | --- |
| 对象名称 | 关键字 | 描述 |
| 表 | Table | 表说存储数据的逻辑单元，由行和列组成。行又叫记录，列又称为字段 |
| 数据字典 |  | 系统表，存放数据库相关信息的表，系统表的数据由数据库维护 |
| 约束 | Constraint | 执行数据校验的规则，保证数据的完整性 |
| 视图 | View | 一个或多个表的逻辑显示。视图不存储数据 |
| 索引 | Index | 用于提高查询性能，相当于目录 |
| 函数 | Function | 用于完成特定计算，返回一个值 |
| 存储过程 | Procedure | 用于完成一次完整的业务处理，没有返回值，通过传出参数将多个值传给调用环境 |
| 触发器 | Tigger | 相当于一个事件监听器 |

创建表的语法

create table [模式名] 表名{

columName1 dataType [default expr],

~~

}

create table test{

id int，

price decimal，

name varchar（255） default ‘null’，

desc text,

img blob,

date datatime

}

MYSQL数据类型（列）

|  |  |
| --- | --- |
| 类型 | 说明 |
| tinyint/smallint/mediumint/int/bigint | 1字节/2字节/3字节/4字节/8字节 |
| flioat/double |  |
| decimal | 精确小数类型 |
| date | 日期 |
| time | 时间 |
| datetime | 日期时间 |
| timestamp | 时间戳 |
| year |  |
| char | 定长字符串 |
| varchar | 变长 |
| binary | 定长二进制字符类型，以二进制保存字符串 |
| tinyblob/blob/mediumblob/longblob | 1字节/2/3/4的二进制大对象，可存音乐，图片的二进制数据。可存储：255b、64k、16mb、4g |
| tinytext/text/mediumtext/longtext | 文本 |
| enum（value，value） | 枚举类型，该列的取值为（）里的值 |
| set（，，，，，） | 集合类型，取值范围集合里的多个 |
|  |  |

修改表使用alter table

增加列：

alter table 表名 add（

column\_name1 datatype[default ‘x’]

````

）

alter table test add comment varchar(500);

修改列

alter table 表名 modify column\_name datatype [default expr] [first|after col\_name]

alter table test modify comment varchar(1000);

删除列：

alter table 表名 drop column\_name;

alter table test drop comment;

重命名表和重命名列（mysql特殊语法）:

重命名表：

alter table 表名 rename to 新表名;

alter table test rename to test1;

重命名列：

alter table 表名 change old\_column new\_column\_name type [default expr] [first|after col\_name]

alter table test1 change date date datetime;

truncate表

一次性删除表里的所有数据，不能指定要删除的行，效率比delete高。

语法格式：

truncate 表名；

删除表

drop table 表名；

删除表后：表结构被删除、表对象不再存在。表里的数据也被删除。该表的索引、约束也被删除。

数据库约束

通过约束可以更好的保证数据表里的数据完整性

（1）not null 列不能为空（2）unique唯一约束，指定列和组合列不能重复

（3）primary key 主键 指定列可以唯一标识记录（4）foreign key 指定该行记录从属于主表中的一条记录，主要用完成参照完整性（5）check 检查，指定一个boolean表达式，用于指定对应列必须满足改表达式

根据约束对列的限制，分为两类

（1）单列约束，制作用在一列（2）多列约束，每个约束可以约束多个数据列

针对5种约束的建立和删除：

（1）、NOT NULL约束

创建表时语法格式：

create table 表名(

       colum\_name datatype not null,

   …

)

也可以在使用alter table 修改表时增加或删除约束：

增加非空约束：

alter table 表名

modify colnum\_name datatype not null;

删除非空约束：

alter table 表名

modify colnum\_name datatype null;

unique唯一约束

唯一约束用于保证指定列或指定组合不能出现重复值，为某列创建唯一约束时，mysql会为该列响应创建唯一索引。如果不给唯一约束命名，唯一约束默认与列名相同

唯一约束可以使用列级约束语法建立，也可以使用表级约束语法建立，如果需为多列建立唯一约束，只能用表级约束语法

列级：在列后加unique

create table test\_unique(

id int not null,

name varchar(200) unique

)

表级：create table test2\_unique(

id int not null,

name varchar(200) unique,

pass varchar(200),

constraint test2\_unique(name,pass)

);

在修改表的结构时使用add关键字增加唯一约束

alter table test2\_unique add unique(name,pass);

在修改表时使用modify关键字，为单列采用列级约束增加唯一约束：

alter table test2\_unique modify name varchar(200) unique;

删除唯一约束：

alert table test2\_unique drop index test2\_c;

PRIMARY KEY约束

主键约束等于唯一约束+非空约束。主键列可以唯一标识表中的一条记录。

一张表只能有一个主键，主键可以是由多列组成。多列组成的主键称为联合主键。

当创建主键约束时，mysql在主键约束所在列或列组合上默认建立对应的唯一索引。

建表时创建主键约束（列级语法）：

create table test\_primary(

    id int primary key,

    name varchar(200)

);

建表时创建主键约束，使用表级语法：

create table test2\_primary(

    id int not null,

    name varchar(200),

    pass var char(200),

    constraint test2\_primary key(id)

);

创建表时创建主键约束，以多列建立组合主键，使用表级语法：

create table test3\_primary(

    name varchar(200),

    pass var char(200),

    constraint test3\_primary key(name,pass)

);

建表后增加主键约束，使用列级语法：  
alter table test3\_primary

modify name varchar(200) primary key;

建表后增加主键约束，使用表级语法：

alter table test3\_primary add primary key(name,pass);

删除主键：

alter table test3\_primary drop primary key;

设置主键是否自增长：

auto\_increment

（4）FOREIGN KEY约束

外键约束主要用于保证一张或者两张表之间的参照完整性。外键是构建于一个表的两个字段或者两个表的两个表的两个字段自检的参照关系。子表外键列的值必须在主表被参照列的范围之内，或者为空。

当主表的记录被从表记录参照时，主表记录不允许删除，必须先把从表里参照该记录的所有记录全部删除后，才可以删除主表的该记录。另外一种方式，删除主表记录时级联删除从表中所有参照该记录的从表记录。

从表外键参照的只能是主表的主键或唯一的列。

建立外键时，mysql也会为该列建立索引。

外键约束应用的场景，一对多、一对一的关联关系。通常是在“多”端添加外键列。例如：老师-学生、班级-学生、公司-职员。

create table teacher\_table(

    teacher\_id int auto\_increment,

    teacher\_name varchar(200),

    primary key(teacher\_id)

);

列级语法：

create table student\_table(

    student\_id int auto\_increment primary key,

    student\_name varchar(200),

    teacher\_id int references teacher\_table(teacher\_id)

);

表级语法：

create table student\_table(

    student\_id int auto\_increment primary key,

    student\_name varchar(200),

    teacher\_id int,

    foreign key (teacher\_id) references teacher\_table(teacher\_id)

);

表级语法，指定外键约束的名称：

create table student\_table(

    student\_id int auto\_increment primary key,

    student\_name varchar(200),

    teacher\_id int,

    constraint student\_teacher\_fk foreign key (teacher\_id)

     references teacher\_table(teacher\_id)

);

删除外键约束：

alter table student\_table drop foreign key student\_teacher\_fk;

增加外键约束：

alter table student\_table  add foreign key (teacher\_id)

references teacher\_table(teacher\_id);

如果想定义当删除主表记录时，从表记录随之删除，需要在外键约束后添加on delete cascade或on delete set null。第一种情况，当删除参照该主表记录时从表的相关记录全部删除。第二种情况，当删除参照主表记录时，从表相关记录的外键值设置为null。

create table student\_table(

    student\_id int auto\_increment primary key,

    student\_name varchar(200),

    teacher\_id int,

    constraint student\_teacher\_fk foreign key (teacher\_id)

     references teacher\_table(teacher\_id)

     on delete cascade/on delete set null

);

1.6索引

索引是存放在模式中的一个数据库对象。创建索引的唯一作用就是加速对表的查询，索引通过使用快速路径访问方法快速定位数据，从而减少了磁盘的I/O。

创建索引的方式：

（1）、自动：当在表上定义主键、唯一、外键约束时，系统会为该数据列自动创建对应的索引。

（2）、手动：可以通过create index …语句创建索引。

删除索引的方式：

（1）、自动：删除表时，该表上的索引自动删除。

（2）、手动：通过drop index … 删除索引。

创建索引语法格式：

create index index\_name on table (column [,column]…);

create index index\_student\_name on student\_table (student\_name);

create index index\_name\_id on student\_table (student\_name,teacher\_id);

删除索引：

drop index 索引名 on 表名；

drop index index\_student\_name on student\_table;

索引的好处是能提高查询效率，索引的坏处：

（1）、当数据表中记录被添加、删除、修改时，数据库系统需要维护索引，有一定的系统开销。

（2）、需要占用一定的存储空间。

1.7视图

视图不是数据表，不能存储数据。视图只是一个或多个数据表中数据的逻辑显示。

使用视图的优点是：

（1）、可以限制对数据的访问。

（2）、可以使复杂的查询变得简单。

（3）、提供数据的独立性。

（4）、提供对相同数据的不同显示。

建立视图的语法格式：

create or replace view 视图名 as 查询语句

create or replace view student\_view as

select student\_id,student\_name from student\_table;

删除视图：

drop view 视图名;

drop view student\_view;

DML语法

dml主要操作数据库表数据，使用dml可以完成（1）插入数据（2）修改数据（3）删除数据

由 insert into，update和delete三个命令组成

1、insert into语句

用于向指定数据表中插入数据，语法格式：

insert into table\_name[(column[,column...])] values(value[,value...]);

insert into teacher\_table values(null,'王老师');

insert into taeacher\_table (teacher\_name) values('李老师');

insert into student\_table values(null,'王五',1);

2、update语句

update用于修改数据表中的记录，每次可以修改多条记录，通过使用where子句限定修改哪些记录。Where子句是一个条件表达式，是一个限定语句，只有条件成立时，才修改数据库中对应的记录。语法格式：

update table\_name set column1=value1 [,column2=value2...]

[where 条件]

update student\_table set student\_name='王五' where teacher\_id = 1;

update student\_table set student\_name='张三' where student\_id = 1;

3、delete from语句

delete from语句用于删除指定数据表的记录。可以一次删除多行，通过where子句限定要删除的记录。

语法格式：

delete from table\_name [where 条件]

delete from student\_table where student\_id = 1;

delete from student\_table;

select 语句的功能就是查询数据，select语句不仅可以执行单表查询，而且可以执行多表连接查询。

select column1,column2... from 数据源 [where 条件]

数据源可以是表、视图、临时表等，查询所有列select \*

语法格式：

select column1,column2... from 数据源 [where 条件]

select \* from student\_table;

select student\_name from student\_table;

select student\_name from student\_table where teacher\_id=1;

where子句中可以使用算数表达式。

select student\_name from student\_table where teacher\_id\*2=2;

select中使用算数运算符。

select student\_id +5,student\_name from student\_table;

concat函数进行字符串连接运算：

select concat(student\_name,'同学') from student\_table;

select concat(student\_name,teacher\_id) from student\_table;

给列命别名：as关键字 或 列后直接跟别名

select student\_name as name,teacher\_id as id from student\_table;

select student\_name  name,teacher\_id  id from student\_table;

distinct关键字重查询结构中清楚重复行：

select distinct student\_name  name,teacher\_id  id from student\_table;

where子句中可以使用比较运算符：>,>=,<,<=,=,<>基本的比较运算符。Sql中的比较运算符不仅可以比较数据之间的大小，也可以比较字符串、日期之间的大小。

特殊的比较运算符：

|  |  |
| --- | --- |
| 运算符 | 含义 |
| 列名 between  exp1 and exp2 | 列>=exp1 并且 <= exp2 |
| 列 in（exp1,exp2…） | 列等于括号中任意一个表达式 |
| like | 字符串匹配，%，\_ |
| is null | 要求指定值等于null |

select \* from student\_table where teacher\_id <> 1;

select \* from student\_table WHERE student\_id BETWEEN 12 and 15;

select \* from student\_table WHERE student\_id IN (12,13,14,15);

select \* from student\_table WHERE student\_name LIKE '王五';

select \* from student\_table WHERE student\_name LIKE '王五\_';

select \* from student\_table WHERE student\_name LIKE '王五%';

如果where子句后有个多个条件需要组合，and、or逻辑运算符来组合两个条件，not对逻辑表达式求否。

select \* from student\_table WHERE student\_name = '王五' AND teacher\_id = 3;

select \* from student\_table WHERE student\_name = '王五' OR teacher\_id = 3;

select \* from student\_table WHERE student\_name is NULL;

select \* from student\_table WHERE student\_name is  not NULL;

sql中比较运算符、逻辑运算符的优先级

|  |  |
| --- | --- |
| 运算符 | 优先级（数字小的优先） |
| 所有的比较运算符 | 1 |
| not | 2 |
| and | 3 |
| or | 4 |

如果sql需要改变优先级的默认值，可以使用括号，括号的优先级比所有的运算符高。

执行查询后的结果默认按插入顺序排列，如果需要查询结果按某列值的大小排序，order by关键字。

select \* from student\_table ORDER BY teacher\_id DESC,student\_id;

10 数据函数

单行函数和多行函数，多行数据成为分组函数、聚簇函数。函数属于数据库可以直接调用。

单行函数分类：

日期时间函数、数值函数、字符函数、转换函数、其他函数（位函数、流程控制函数、加密函数、信息函数）。

列字符长度函数：char\_length

获取当前日期的函数：curdate()

获取当前时间的函数：curtime()

加密函数：MD5(‘www’)

处理null的函数：

Ifnull(expr1,expr2)：如果expr1为null，则返回expr2，否则返回expr1.

nullif(expr1,expr2):若果expr1和expr2相等，返回null，否则返回expr1。

If(expr1,expr2,expr3)：类似三目运算符。

Isnull(expr):如果expr为null返回true，否则返回false。

流程控制语句，又叫case函数：

SELECT student\_name

CASE teacher\_id

    WHEN 1 THEN '王老师'

    WHEN 2 THEN '李老师'

    WHEN 3 then '张老师'

    ELSE '其他老师'

END

FROM student\_table;

SELECT student\_name,

CASE

    WHEN teacher\_id <= 1 THEN '王老师'

    WHEN teacher\_id > 1 THEN '李老师 or 张老师'

    ELSE '其他老师'

END

from student\_table;

分组和组函数

组函数是多行函数，又叫聚簇函数、聚集函数。组函数将一组记录作为整体计算，每组记录返回一个结果。常用的5个组函数：

avg（[distinct|all]expr）:计算多行expr的平均值。Expr可以是变量、常量或数据列。数据类型必须是数值类型。使用distinct表示不计算重复值。

count(\*|[distinct|all]expr):计算expr的总记录数。expr可以是变量、常量或数据列。不限制数据类型；使用\*表示计算表中的记录数。使用distinct表示不计算重复值。

max(expr):计算多行expr的最大值。

min(expr):计算多行expr的最小值。

sum([distinct|all]expr)：计算多行expr的总和。

SELECT COUNT(\*) from student\_table；

SELECT MAX(student\_id) from student\_table；

SELECT sum(student\_id) from student\_table；

group by ：对记录进行显示分组。

SELECT count(\*) from student\_table GROUP BY teacher\_id,student\_name;

如果需要对分组进行过滤，使用having子句，having子句后面是一个条件表达式，只有满足了条件表达式的分组才会选出来。

SELECT count(\*) from student\_table

GROUP BY teacher\_id,student\_name

HAVING count(\*) >=2;

多表连接查询

多表连接查询有两种规范，较早的SQL92规范支持如下几种多表连接查询。

等值连接

非等值连接

外连接

广义笛卡尔积。

SQL99规范提供了可读性更好的多表连接语法。并提供了更多类型的连接查询。SQL99支持如下几种多表连接查询。

交叉连接

交叉连接和sql92的广义笛卡尔积类似，没有连接条件。

SELECT s.\*,t.\* from student\_table s

CROSS JOIN teacher\_table t;

自然连接

自然连接无序指定连接条件，但是自然连接是有连接条件的，自然连接会以连个表中的同名列作为连接条件；如果两个表中没有同名列，则自然连接与交叉连接效果完全一样。

SELECT s.\*,t.\* from student\_table s

NATURAL JOIN teacher\_table t;

使用using子句的连接

Using子句可以指定一列或多列，用于显示指定两个表中作为连接条件。（版本不支持）

SELECT s.\*,t.\* from student\_table s

NATURAL JOIN teacher\_table t USING(teacher\_id);

使用on子句的连接

On子句的连接是最常用的连接方式，SQL99语法的连接条件放在on子句中指定。

SELECT s.\*,t.\* from student\_table s JOIN teacher\_table t

on s.teacher\_id = t.teacher\_id;

SELECT s.\*,t.\* from student\_table s JOIN teacher\_table t

on s.teacher\_id > t.teacher\_id;

全外连接或左、右外连接。

左外连接left [outer] join:

SELECT s.\*,t.\* from student\_table s left JOIN teacher\_table t

on s.teacher\_id = t.teacher\_id;

右外连接right [outer]join:

SELECT s.\*,t.\* from student\_table s right JOIN teacher\_table t

on s.teacher\_id = t.teacher\_id;

全外连接：full [outer] join（版本不支持）

SELECT s.\*,t.\* from student\_table s full JOIN teacher\_table t

on s.teacher\_id = t.teacher\_id;

子查询

子查询就是指在查询语句中另外嵌套另一个查询。支持多层嵌套。子查询可以出现在两个位置：

from 语句后当成数据表，该子查询的实质是一个临时视图。

where条件后，作为过滤条件的值。

使用子查询需要注意的地方：

子查询使用括号括起来。

把子查询当成数据时，可以为该子查询起别名，特别是作为前缀来限定列时，必须给子查询起别名。

把子查询作为过滤条件时，将子查询放在比较运算符的右侧，增强查询的可读性。

把子查询作为过滤条件时，单行子查询使用单行运算符，多行子查询使用多行运算符。

子查询作为数据表：

SELECT \* from (SELECT \* from student\_table) t

where t.teacher\_id > 2;

子查询作为过滤条件（单行、单列值）：

SELECT \* from student\_table t

where t.teacher\_id >

(SELECT teacher\_id from teacher\_table WHERE teacher\_name='王老师');

如果子查询返回多个值，需要使用in 、any和all等关键字。

SELECT \* from student\_table t

where t.teacher\_id IN

(SELECT teacher\_id from teacher\_table);

Any和all可以与>,>=,<,<=,<>,=运算符结合使用，与any结合使用分别表示大于、大于等于、小于、小于等于、不等于、等于其中任意一个值。与all结合使用分别表示大于、大于等于、小于、小于等于、不等于、等于全部值。=any与in的作用一样。

<any只要小于值列表的最大值中的最大值即可。>any只要大于值列表中的最小值即可。<all要求小于值列表中的最小值；>all要求大于值列表中的最大值。

SELECT \* from student\_table t

where t.student\_id >

all(SELECT teacher\_id from teacher\_table);

多行、多列子查询，此时where子句中应该有对应的数据列，并使用括号括起来：

SELECT \* from student\_table t

where (student\_id,student\_name) IN

(SELECT teacher\_id,teacher\_name from teacher\_table);

集合运算

Select语句查询的结果是一个包含多条数据的结果集，类似于数学中的集合，可以进行交（intersect）、并（union）、差（minus）运算。

进行集合运算，必须满足两个条件：

1、两个结果集包含的数据量必须相等。

2、数据类型必须一样。

并（union）运算：

SELECT student\_id,student\_name from student\_table s

UNION

SELECT teacher\_id,teacher\_name from teacher\_table t;

minus运算：mysql版本不支持

可以通过子查询实现：

SELECT \* from student\_table t

where t.student\_name not IN

(SELECT teacher\_name from teacher\_table);

Intersect运算：mysql版本不支持

可以通过子查询实现：

SELECT \* from student\_table t

where t.student\_name IN

(SELECT teacher\_name from teacher\_table);

JDBC基础

JDBC API是java访问数据库的接口规范，各家厂商根据接口规范，提供具体的实现类。这些实现类被封装为jar包，以jdbc驱动程序供开发者使用。

Jdbc可以完成3个基本工作：（1）建立与数据库的连接。（2）执行sql语句。（3）获得sql语句的执行结果。

应用JDBC

jdbc编程步骤

加载驱动

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

通过DriverManager获得数据库连接。

DriverManager.getConnection(String URL,String user,String pass)

Mysql数据库的URL写法：

jdbc:mysql://hostname:port/databasename;

Oracle的URL写法：

jdbc:oracle:thin:@hostname:port:databasename

通过数据库连接创建statement对象。Connection创建Statement的方法有如下三种：

createStatement():创建基本的statement对象。

prepareStatement(String sql):根据传入的sql语句创建预编译的Statement对象。

prepareCall(String sql):根据传入的sql语句创建CallableStatement对象。

使用Statement执行sql语句，有三种方法执行sql语句：

execute（）可以执行热播和类型的sql，结果处理麻烦

executeUpdate（）主要用来执行，dml和ddl语句，返回受sql语句影响的条数

executeQuery（）只能执行查询语句，返回结果集resultset对象

操作结果集。操作resultset对象

两类方法

A、移动resultset的记录指针：next（）previous（）first（）last（），beforefirst（）afterlast（）等

B、getxxx（）

执行sql语句方式

（1）执行ddl和dml语句

（2）使用execute执行任何sql语句

使用PreparedStaatement执行sql查询语句

在插入数据时可以使用？占位符代替sql中的值

jdbc提供了PreparedStaatement执行预编译的sql语句。PreparedStaatement是statement接口的子接口然后使用该对象多次高效执行该语句

在sql前，使用setxxx（int index，xxx value）方法设置参数值

PreparedStaatement对象的execute（）executeUpdate（）executeQuery（）

优势：

（1）、使用预编译方式，不用拼接sql。降低了编写sql的复杂性。

（2）、预防sql注入。

SELECT \* from student\_table WHERE student\_id = 20 and student\_password='' or TRUE or '';

处理Blob类型的数据

Blob（Binary Long Object）是二进制长对象的意思，Blob可以存储大文件，可以存储图片、声音、视频文件。将Blob数据插入数据库需要使用PreparedStatement，调用setBinaryStream(int index,InputStream x)，该方法可以为执行参数传入二进制输入流。

取出时从ResultSet中取出Blob数据，调用getBlob（int index），返回Blob对象，调用该对象的getBinaryStream（）方法获取该Blob数据的输入流。

事物处理

事物有4个特征：原子性：事物是应用中执行的最小单位，事物中的任务要么都执行成功，否则都不执行；一致性：事务的执行结果，必须使数据库从一个一致性状态变为另外一个一致性状态；隔离性：各个事务的执行互不干扰；持续性：持久性，指事务一旦提交，对数据所做的任何改变都记录到永久存储中。

这4个特性简称ACID性。

数据库的事物由以下语句组成：

（1）一组dml语句（2）一条ddl语句（3）一条dcl语句

当事务所包含的全部数据库操作执行成功后，应该提交事务。使这些修改永久的生效。提交事物2种方式：显示提交：commit；自动提交：当执行一条ddl或dcl自动提交

当事务中所包含的任意一个数据库操作执行失败后，应该回滚事务。使该事务中所做的修改全部失效。回滚方式：显示回滚rollback；自动回滚：系统错误和强制退出。mysql默认是关闭事务的。set autocommit=0|1设置事务是否打开。0关自动提交，开启事务

jdbc提供了事务支持JDBC连接的事务支持是Connection提供。默认打开自动提交，即关闭事务。

Con.setAutoCommit(false);

Con.commit();

Com.rollback();

6、使用连接池管理连接

数据库连接的建立及关闭是很耗费系统资源的操作。

数据库连接池的解决方案是：当应用程序启动时，系统主动建立足够的数据库连接，并将这些连接组成一个连接池。每次应用程序请求数据库时，不需要重新打开连接，从连接池中取已经存在的连接。使用完毕后，不关闭连接，而是将连接归还给连接池。

数据库连接池是Connection对象的工厂。数据库连接池的常用参数如下：

（1）、连接池的初试连接。

（2）、连接池的最大连接。

（3）、连接池的最小连接。

（4）、连接池每次增加的容量。

JDBC的数据库连接池使用javax.sql.DataSource。DataSource是一个接口，该接口通常由厂商提供具体实现。

6、DBCP数据源

DBCP是Apache软件下的开源连接池实现。如果需要使用该连接池，需要使用两个jar包：

commons-pool.jar:连接池实现的依赖库。

commons-dbcp.jar:连接池的实现。

Tomcat的连接池是采用该连接池实现的。

使用DBCP获得数据库连接的方式：

//创建数据源对象

BasicDataSource ds = new BasicDataSource();

//设置连接池所需要的驱动

ds.setDriverClassName(“com.mysql.jdbc.Driver”);

//设置连接数据库的URL

ds.setUrl(“jdbc:mysql://IP:port/数据库名”);

//设置用户名

ds.setUsername(“root”);

//设置密码

ds.setPassword(“root”);

//设置初试连接数

ds.setInitialSize(5);

//设置最大连接数

ds.setMaxActive(20);

//设置最小连接数

ds.setMinIdle(2);

//获得数据库连接

Connection con = ds.getConnection();

//释放数据库连接

con.close();

6.2C3P0数据源

C3P0性能更好一些，Hibernate就推荐使用该连接池。可以自动清理不再使用的Connection，Statement，ResultSet。C3p0.jar

，