1. 一组系列的数据库操作语句组合在一起称为事务，
BEGIN 开始一个事务(START **TRANSACTION**;)
COMMIT 提交事务
ROLLBACK 事务回滚
任何的 COMMIT 或者 ROLLBACK 命令都会触发事务的结束。
例如：
在终端[begin;]----开始事务；
SQL 语句...;
在终端[commit;]---提交结束事务。
另外，HeidiSQL 的查询执行和终端还是有区别的，终端可以处理更多的语句。
2. 在 MySQL 中，每一个查询，默认都是当做一个事务来处理的，所以 autocommit 默认是开启的。
如果要关闭自动提交事务功能，需要在执行各种语句之前执行一条语句
SET AUTOCOMMIT = 0;
注意，一旦关闭自动提交，后面所有的查询都不会立刻执行，**而是会等待 COMMIT 或者 ROLLBACK 的出现。**
注意，如果需要恢复自动提交，需要再执行
SET AUTOCOMMIT = 1;
3. 【**SET** AUTOCOMMIT = 1;】之前若有没有提交的语句会自动提交.
4. JDBC（Java Database Connectivity,java 数据库连接）, 涉及 java.sql,javax.sql 包
5. 连接数据库的步骤：
●注册驱动 (只做一次)
●建立连接(Connection)
●创建执行 SQL 的语句(Statement)
●执行语句
●处理执行结果(ResultSet)
●释放资源
6. 注册驱动：
public static final String DBDRIVER = "com.mysql.jdbc.Driver";
Class.forName(DBDRIVER);
说明，数据库驱动的反转域名不需要特别记，导入 mysql 的驱动包后，可以从包中拷贝得到！
7. 建立连接：
●Connection conn = DriverManager.getConnection(url, user, password);
●url 格式：JDBC:子协议:子名称//主机名:端口/数据库名？**属性名=属性值&…**
●User,password 可以用“属性名=属性值”方式告诉数据库；
●其他参数如：useUnicode=true&characterEncoding=UTF-8。
public static final String DBURL = "jdbc:mysql://192.168.1.10:3306/db_dbdemo2";
public static final String DBUSER = "root";
public static final String DBPWD = "root";
Connection conn= null; //数据库的连接

conn = DriverManager.getConnection(DBURL, DBUSER, DBPWD);

说明：如果从 java 中插入中文字到数据库为？时，可以在 url 后面加上字符编码来解决！

8. 创建执行 **SQL** 的语句**(Statement)**

●**Statement**

```
Statement st = conn.createStatement();
st.executeQuery(sql);
```

●**PreparedStatement**

```
String sql = “select * from table_name where col_name=?”;
PreparedStatement ps = conn.preparedStatement(sql);
    ps.setString(1, “col_value”); //设定第一个？的值
    ps.executeQuery();
```

9. 处理执行结果**(ResultSet)**

```
ResultSet rs = statement.executeQuery(sql);
While(rs.next()){
  rs.getString(“col_name”);
  rs.getInt(“col_name”);
  //…
}
```

10. 释放资源

●释放 ResultSet, Statement,Connection.

●数据库连接（Connection）是非常稀有的资源，用完后必须马上释放，如果 Connection 不能及时正确的关闭将导致系统宕机。**Connection 的使用原则是尽量晚创建，尽量早的释放。**

11. 基本的 **CRUD**（创建、读取、更新、删除）

●模板代码

```
Connection conn = null;
Statement st=null;
ResultSet rs = null;
try {
  //获得 Connection
  //创建 Statement
  //处理查询结果 ResultSet
} finally {
  //释放资源 ResultSet, Statement,Connection
}
```

12. **CRUD** 总结

●**增、删、改用 Statement.executeUpdate 来完成**，返回整数(匹配的记录数)，这类操作相对简单。

●**查询用 Statement.executeQuery 来完成，**返回的是 ResultSet 对象，ResultSet 中包含了查询的结果；查询相对与增、删、改要复杂一些，因为有查询结果要处理。

13. PreparedStatement 和 Statement

●在 SQL 中包含特殊字符或 SQL 的关键字(如：' or 1 or ')时 Statement 将出现不可预料的结果（出现异常或查询的结果不正确），可用 PreparedStatement 来解决。

●PreperedStatement（从 Statement 继承而来）相对 Statement 的优点：

A.没有 SQL 注入的问题。

B.Statement 会使数据库频繁编译 SQL，可能造成数据库缓冲区溢出。

C.数据库和驱动可以对 PreperedStatement 进行优化（只有在相关联的数据库连接没有关闭的情况下有效）。

14. 数据类型

●详细信息见 java.sql.Types

●几种特殊且比较常用的类型

1.DATA,TIME,TIMESTAMP→ date,time,datetime

存：ps.setDate(i,d); ps.setTime(i,t); ps.setTimestamp(i, ts);

取：rs.getDate(i); rs.getTime(i); rs.getTimestamp(i);

2.CLOB → text

存：ps.setCharacterStream(index, reader, length); ps.setString(i, s);

取：reader = rs. getCharacterStream(i);

reader = rs.getClob(i).getCharacterStream();

string = rs.getString(i);

3.BLOB → blob

存：ps.setBinaryStream(i, inputStream, length);

取：rs.getBinaryStream(i);　rs.getBlob(i).getBinaryStream();

15. 事务（ACID）

●原子性(atomicity)：组成事务处理的语句形成了一个逻辑单元，不能只执行其中的一部分。

●一致性(consistency)：在事务处理执行前后，数据库是一致的(数据库数据完整性约束)。

●隔离性(isolcation)：一个事务处理对另一个事务处理的影响。

●持续性(durability)：事务处理的效果能够被永久保存下来 。

●connection.setAutoCommit(false);//打开事务。

●connection.commit();//提交事务。

●connection.rollback();//回滚事务。