参考： <http://blog.csdn.net/jiangwei0910410003/article/details/26143977>

以**[Oracle](http://lib.csdn.net/base/oracle" \o "Oracle知识库" \t "_blank)**为例吧

Statement为一条Sql语句生成执行计划，  
如果要执行两条sql语句  
**select colume from table where colume=1;  
select colume from table where colume=2;**  
会生成两个执行计划  
一千个查询就生成一千个执行计划！

PreparedStatement用于使用绑定变量**重用**执行计划

**select colume from table where colume=:x;**  
通过set不同数据只需要生成一次执行计划，可以重用

是否使用绑定变量对系统影响非常大，生成执行计划极为消耗资源  
两种实现 速度差距可能成百上千倍

1.PreparedStatement是预编译的,对于批量处理可以大大提高效率.也叫JDBC存储过程

2.使用 Statement 对象。在对**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**只执行一次性存取的时侯，用 Statement 对象进行处理。PreparedStatement对象的开销比Statement大，对于一次性操作并不会带来额外的好处。

3.statement每次执行sql语句，相关数据库都要执行sql语句的编译，preparedstatement是预编译的，preparedstatement支持批处理

4.**Code Fragment 1:**

**[java]** [view plain](http://blog.csdn.net/jiangwei0910410003/article/details/26143977) [copy](http://blog.csdn.net/jiangwei0910410003/article/details/26143977)

1. String updateString = "UPDATE COFFEES SET SALES = 75 " + "WHERECOF\_NAME LIKE ′Colombian′";
2. stmt.executeUpdate(updateString);

**Code Fragment 2:**

**[java]** [view plain](http://blog.csdn.net/jiangwei0910410003/article/details/26143977) [copy](http://blog.csdn.net/jiangwei0910410003/article/details/26143977)

1. PreparedStatement updateSales = con.prepareStatement("UPDATE COFFEES SETSALES = ? WHERE COF\_NAME LIKE ? ");
2. updateSales.setInt(1, 75);
3. updateSales.setString(2, "Colombian");
4. updateSales.executeUpdate();

片断2和片断1的区别在于，后者使用了PreparedStatement对象，而前者是普通的Statement对象。PreparedStatement对象不仅包含了SQL语句，而且大多数情况下这个语句已经被预编译过，因而当其执行时，只需DBMS运行SQL语句，而不必先编译（数据库执行性能较高但是预编译消耗了资源）。当你需要执行Statement对象多次的时候，PreparedStatement对象将会大大降低运行时间，当然也加快了访问数据库的速度。  
这种转换也给你带来很大的便利，不必重复SQL语句的句法，而只需更改其中变量的值，便可重新执行SQL语句。选择PreparedStatement对象与否，在于相同句法的SQL语句是否执行了多次，而且两次之间的差别仅仅是变量的不同。如果仅仅执行了一次的话，它应该和普通的对象毫无差异，体现不出它预编译的优越性（但是此时preparedstatement会消耗资源相比statement来说）。

5.执行许多SQL语句的JDBC程序产生大量的Statement和PreparedStatement对象。通常认为PreparedStatement对象比Statement对象更有效,特别是如果带有不同参数的同一SQL语句被多次执行的时候。PreparedStatement对象允许数据库预编译SQL语句，这样在随后的运行中可以节省时间并增加代码的可读性。

然而，在**[oracle](http://lib.csdn.net/base/oracle" \o "Oracle知识库" \t "_blank)**环境中，开发人员实际上有更大的灵活性。当使用Statement或PreparedStatement对象时，Oracle数据库会缓存SQL语句以便以后使用。在一些情况下,由于驱动器自身需要额外的处理和在**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)**应用程序和Oracle服务器间增加的网络活动，执行PreparedStatement对象实际上会花更长的时间。

然而，除了缓冲的问题之外，至少还有一个更好的原因使我们在企业应用程序中更喜欢使用PreparedStatement对象,那就是安全性。传递给PreparedStatement对象的参数可以被强制进行类型转换，使开发人员可以确保在插入或查询数据时与底层的数据库格式匹配。

当处理公共Web站点上的用户传来的数据的时候，安全性的问题就变得极为重要。传递给PreparedStatement的字符串参数会自动被驱动器忽略。最简单的情况下，这就意味着当你的程序试着将字符串“D'Angelo”插入到VARCHAR2中时，该语句将不会识别第一个“，”，从而导致悲惨的失败。几乎很少有必要创建你自己的字符串忽略代码。

在Web环境中，有恶意的用户会利用那些设计不完善的、不能正确处理字符串的应用程序。特别是在公共Web站点上,在没有首先通过PreparedStatement对象处理的情况下，所有的用户输入都不应该传递给SQL语句。此外，在用户有机会修改SQL语句的地方，如HTML的隐藏区域或一个查询字符串上，SQL语句都不应该被显示出来。  
在执行SQL命令时，我们有二种选择：可以使用PreparedStatement对象，也可以使用Statement对象。无论多少次地使用同一个SQL命令，PreparedStatement都只对它解析和编译一次。当使用Statement对象时，每次执行一个SQL命令时，都会对它进行解析和编译。  
  
第一：

prepareStatement会先初始化SQL，先把这个SQL提交到数据库中进行预处理，多次使用可提高效率。  
createStatement不会初始化，没有预处理，没次都是从0开始执行SQL

第二：

prepareStatement可以替换变量  
在SQL语句中可以包含?，可以用ps=conn.prepareStatement("select\* from Cust where ID=?");  
int sid=1001;  
ps.setInt(1, sid);  
rs = ps.executeQuery();  
可以把?替换成变量。  
而Statement只能用（字符串拼接）  
int sid=1001;  
Statement stmt = conn.createStatement();  
ResultSet rs = stmt.executeQuery("select \* from Cust where ID="+sid);  
来实现。  
  
第三：

prepareStatement会先初始化SQL，先把这个SQL提交到数据库中进行预处理，多次使用可提高效率。  
createStatement不会初始化，没有预处理，没次都是从0开始执行SQL