如果JDBC连接处于自动提交模式，默认情况下，则每个SQL语句在完成后都会提交到数据库。
对于简单的应用程序可能没有问题，但是有三个原因需要考虑是否关闭自动提交并管理自己的事务 -

- 提高性能
- 保持业务流程的完整性
- 使用分布式事务

事务能够控制何时更改提交并应用于数据库。 它将单个SQL语句或一组SQL语句视为一个逻辑单元，如果任何语句失败，整个事务将失败。
要启用手动事务支持，而不是使用JDBC驱动程序默认使用的自动提交模式，请调用Connection对象的setAutoCommit()方法。 如果将布尔的false传递给setAutoCommit()，则关闭自动提交。 也可以传递一个布尔值true来重新打开它。
例如，如果有一个名为conn的Connection对象，请将以下代码关闭自动提交 -

```
conn.setAutoCommit(false);
```

提交和回滚
完成更改后，若要提交更改，那么可在连接对象上调用commit()方法，如下所示：

```
conn.commit( );
```

Java
否则，要使用连接名为conn的数据库回滚更新，请使用以下代码 -

```
conn.rollback( );
```

Java
以下示例说明了如何使用提交和回滚对象 -

```
try{
   //Assume a valid connection object conn
   conn.setAutoCommit(false);
   Statement stmt = conn.createStatement();

   String SQL = "INSERT INTO Employees  " +
                "VALUES (106, 20, 'Rita', 'Tez')";
   stmt.executeUpdate(SQL);
   //Submit a malformed SQL statement that breaks
   String SQL = "INSERTED IN Employees  " +
                "VALUES (107, 22, 'Sita', 'Singh')";
   stmt.executeUpdate(SQL);
   // If there is no error.
   conn.commit();
}catch(SQLException se){
   // If there is any error.
   conn.rollback();
}
```

在这种情况下，上述INSERT语句不会成功执行，因为所有操作都被回滚了。

使用保存点

新的JDBC 3.0新添加了Savepoint接口提供了额外的事务控制能力。大多数现代DBMS支持其环境中的保存点，如Oracle的PL/SQL。

设置保存点(Savepoint)时，可以在事务中定义逻辑回滚点。 如果通过保存点(Savepoint)发生错误时，则可以使用回滚方法来撤消所有更改或仅保存保存点之后所做的更改。

Connection对象有两种新的方法可用来管理保存点 -

- **setSavepoint(String savepointName): - 定义新的保存点，它还返回一个Savepoint对象。**
- **releaseSavepoint(Savepoint savepointName): - 删除保存点。要注意，它需要一个Savepoint对象作为参数。 该对象通常是由setSavepoint()方法生成的保存点。**

有一个*rollback (String savepointName)*方法，它将使用事务回滚到指定的保存点。

以下示例说明了使用Savepoint对象 -

```
try{
   //Assume a valid connection object conn
   conn.setAutoCommit(false);
   Statement stmt = conn.createStatement();

   //set a Savepoint
   Savepoint savepoint1 = conn.setSavepoint("Savepoint1");
   String SQL = "INSERT INTO Employees " +
                "VALUES (106, 24, 'Curry', 'Stephen')";
   stmt.executeUpdate(SQL);
   //Submit a malformed SQL statement that breaks
   String SQL = "INSERTED IN Employees " +
                "VALUES (107, 32, 'Kobe', 'Bryant')";
   stmt.executeUpdate(SQL);
   // If there is no error, commit the changes.
   conn.commit();

}catch(SQLException se){
   // If there is any error.
   conn.rollback(savepoint1);
}
```

在这种情况下，上述INSERT语句都不会成功，因为所有操作都被回滚了。