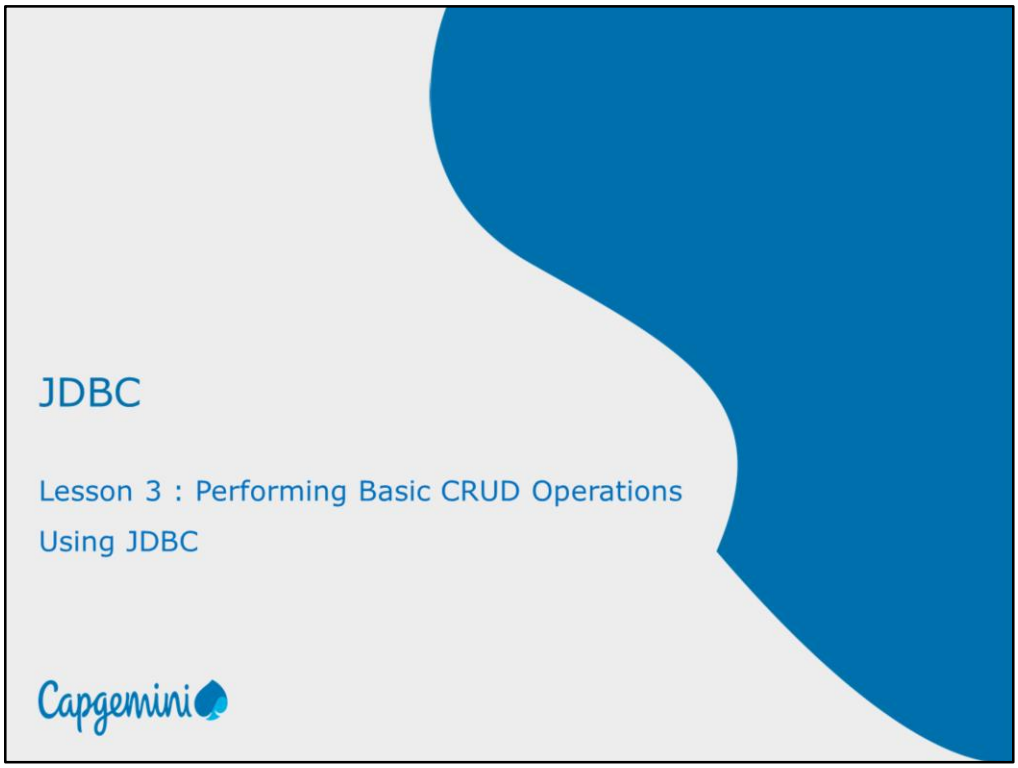
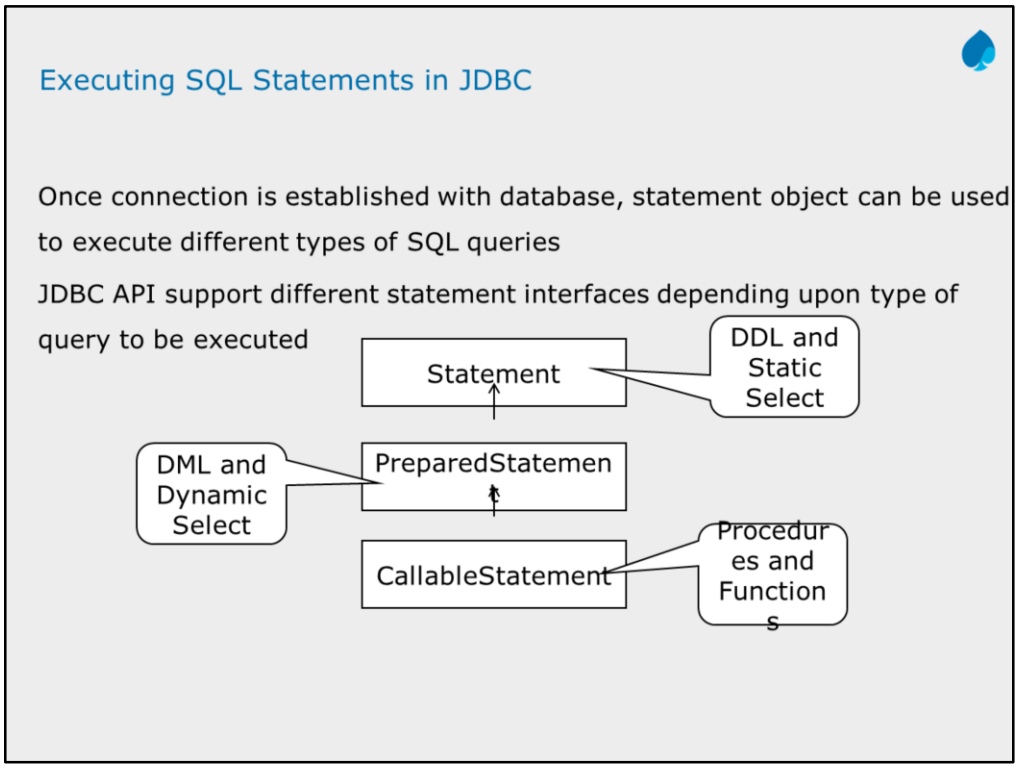

JDBC
Lesson 3 : Performing Basic CRUD Operations
Using JDBC
Capgemini

Lesson Objectives

After completing this lesson, participants will be able to

- Execute Static SQL statements
- Understand Result Set
- Understand Prepared Statements
  - Retrieve(Select)
  - Insert(Create)
  - Update
  - Remove(Delete)



Using Statements:

java.sql.Statement interface in JDBC enables to send the SQL queries to database and retrieve data. Statement interface is suitable for executing DDL queries and Select queries which has no input.

Statement interface is further extended as PreparedStatement, which is recommended for DML and select queries that involves input parameters.

PreparedStatement is in turn extended as CallableStatement, which is suitable for calling database stored procedures and functions.

Statement Interface

Used for executing a static SQL statement

Methods :

- ResultSet executeQuery(String SQL)
- int executeUpdate(String SQL)
- boolean execute (String SQL)

Executing Static SQL using Statement Interface

```
Statement st=conn.createStatement();
ResultSet rs=st.executeQuery("SELECT * FROM emp");
while(rs.next()){
  System.out.println("Emo No = "+rs.getInt("eno"));
  System.out.println("Emo Name = "+rs.getString("ename"));
}
```

Iterating Through ResultSets

Java.sql.ResultSet interface represents the result set of a database query

When the ResultSet is first returned,the starting cursor position is before the first row of data.

Methods of ResultSet

Methods of ResultSet are divided into 3 Categories

1.Navigational Methods

beforeFirst() previous () afterLast() next()

first() absolute() last() getRow()

2.Get Methods

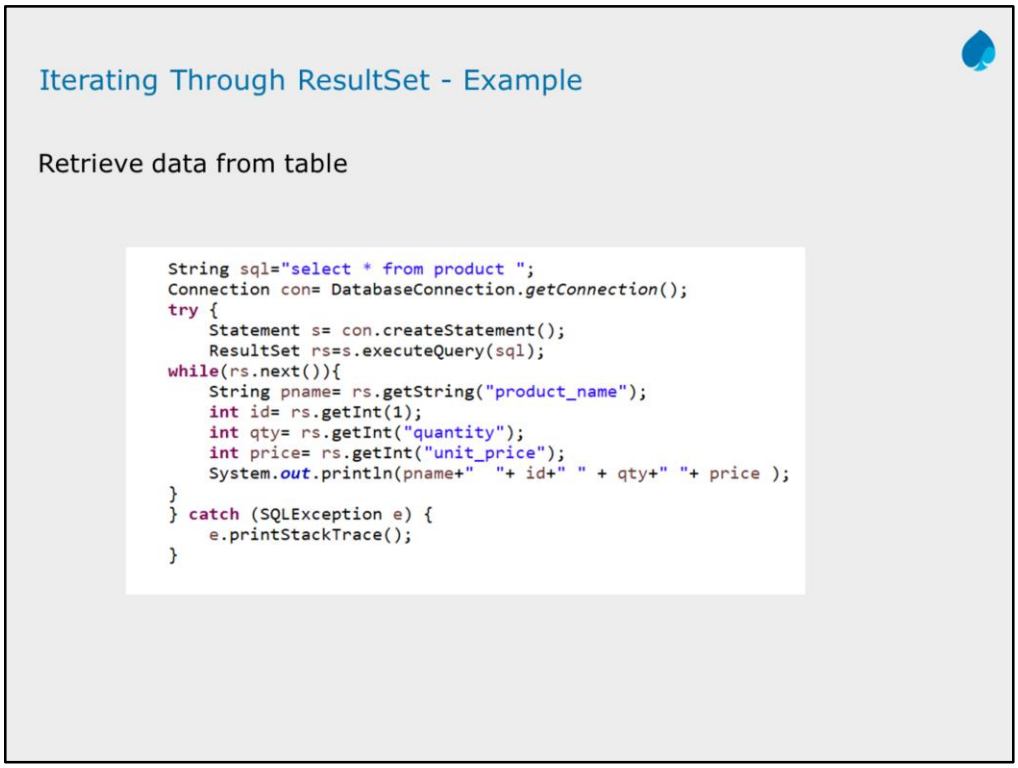
get() method for each data type

Each get method has 2 version :

1.one that takes column name

2.one that takes column index

3.Update Methods : will discuss later



Retrieve data from table:

Once your statement object is ready, you need to execute that statement to get the records from the database.

The executeQuery() method of the Statement returns the ResultSet object which represent the front end table of database records.

Further, you can traverse/iterate through ResultSet to retrieve the records one by one.

Use getXXX() methods to retrieve each data field value, for example rs.getInt("eno") will retrieve the employee number of the current record from rs.

Types of ResultSet:

Resultset contains results of the SQL query. There are three basic types of resultset.

Forward-only

As the name suggests, this type can only move forward and are non-scrollable.

Scroll-insensitive

This type is scrollable which means the cursor can move in any direction. It is insensitive which means any change to the database will not show change in the resultset while it opens.

Scroll-sensitive

This type allows cursor to move in any direction and also propagates the changes done to the database.

Demo

Execute the :

- Select.java

Understanding Scrollable ResultSets

The Result set is by default read only in forward direction.

Many times we need to navigate on resultset

ResultSet Types :

1.Type_Forward_only : result set can be navigated only in forward direction

2.Type_Scroll_Insensative : The cursor can scroll forward and backward and result set is not sensitive to the changes done by others.

3.Type_Scroll_Sensative : cursor can scroll forward and backward and result set is sensitive to the changes done by others.

ResultSet Concurrency Types

1.Concur_Read_Only : It create the resultset read-only

2.Concur_Updatable : It create the updatable resultset

ResultSet - Example

Demo

Execute the Demo on Scrollable ResultSet

Understanding Updatable ResultSets

Updatable ResultSet allows modification to data in a table through ResultSet.

Update method is available for doing updates in table for each data type.

1.One that takes in a column name

2.One that takes in a column index

To update String column value of current row

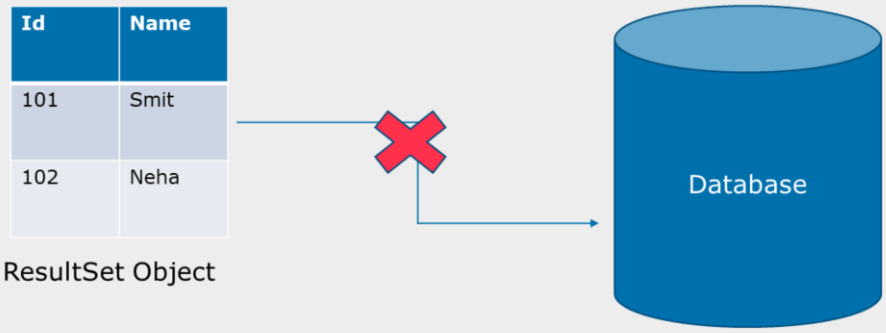
public void updateString(int columnindex,String value)

public void updateString(String columnname,String value)

All the update methods throw SQLException

Understanding Updatable ResultSet

rs.updateString("EmpName","Smith") : will update the value in resultset and not In the underlying Database.



ResultSet Object

To update the value in database ,we need to call updateRow() method on resultset.

Understanding Updatable ResultSet - Methods

1. updateRow() : updates the row in Database
2. deleteRow () : deletes the current row from Database
3. refrestRow() : refreshes the data in resultset to reflect any recent changes in the database.
4. cancelRowUpdate() : cancels any updated made to the current row .
5. insertRow() : insert new row in database

Updatable ResultSet – Example

```
    PreparedStatement ps=
            con.prepareStatement(sql,
                                 ResultSet.TYPE_SCROLL_SENSITIVE,
                                 ResultSet.CONCUR_UPDATABLE);

    ps.setInt(1, 1000);
    ps.setInt(2, 3000);
    ResultSet rs=ps.executeQuery();
while(rs.next()){
    String pname= rs.getString("product_name");
    int id= rs.getInt("product_id");
    int qty= rs.getInt("quantity");
    System.out.println(pname+"  "+ id+" " + qty);
    rs.updateInt("quantity", qty+100);
    rs.updateRow();
}
```

Demo

Demo on Updatable ResultSets

Understanding PreparedStatement

Benefits PreparedStatement :

Since the SQL statements are precompiled one, it improves performance of Application.

Easy to Set SQL Parameter value

Prevents SQL Dependency Injection Attacks

Understanding PreparedStatement Methods

PreparedStatement ps =connection.prepareStatement("Select * from emp where salary > ? And department_code= ? ");

To Bind the values in this PreparedStatement

setXXX() methods are available.

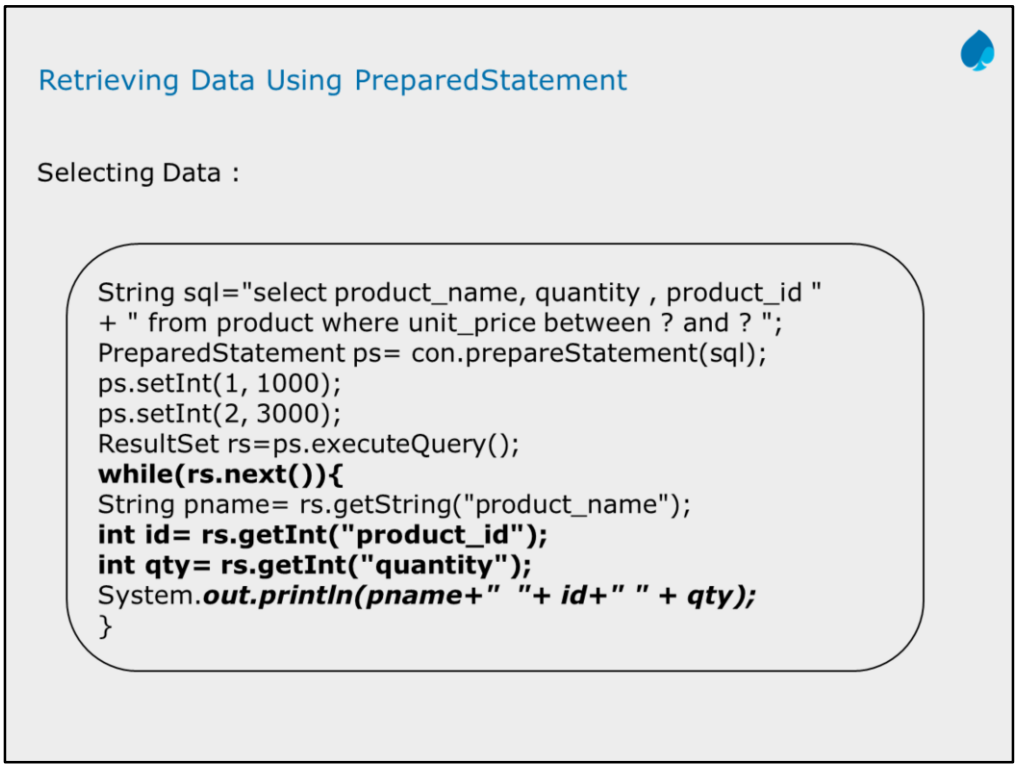
XXX can be replaced by any java datatypes .

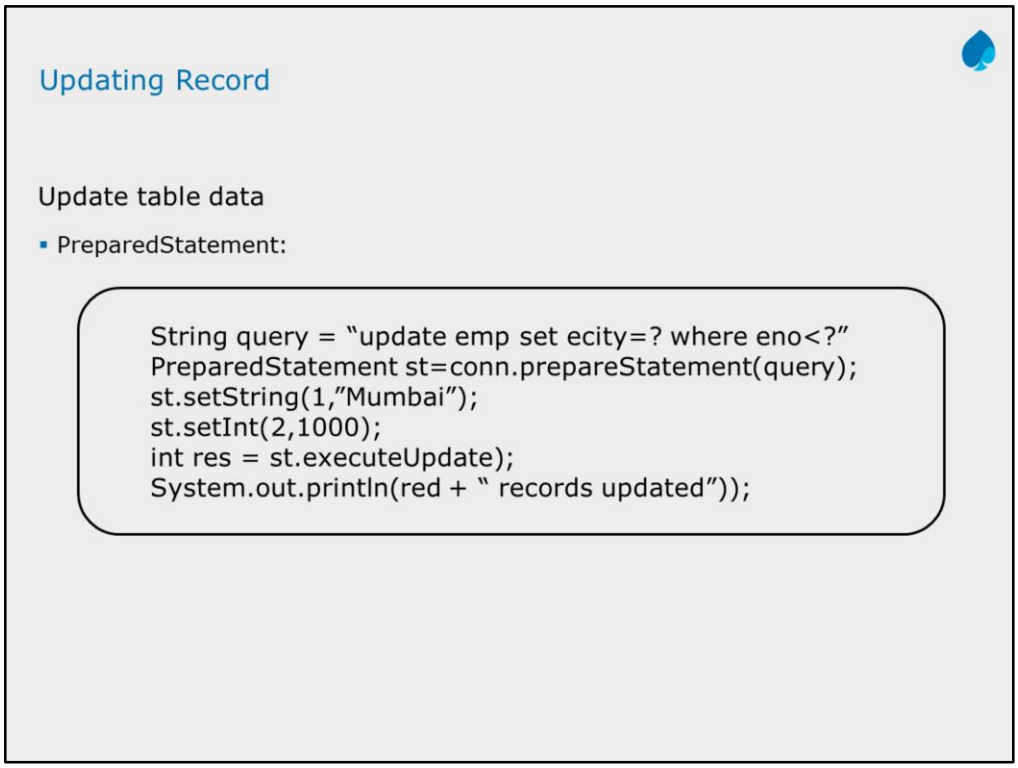
e.g. setInt(P1,P2) (P1 -> position ,P2 -> value)

setInt(1,1000);

setString("emp_name","Neha");

setDouble("salary",23000.22);

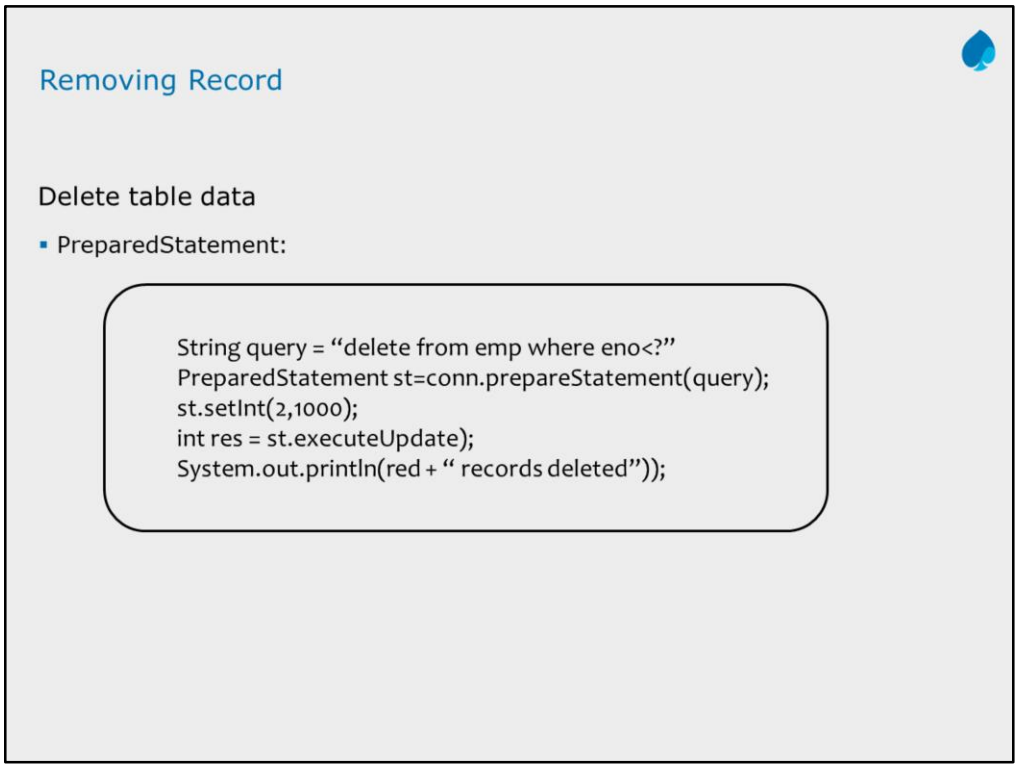

Retrieving Data Using PreparedStatement
Selecting Data :
String sql="select product_name, quantity , product_id "
+ " from product where unit_price between ? and ? ";
PreparedStatement ps= con.prepareStatement(sql);
ps.setInt(1, 1000);
ps.setInt(2, 3000);
ResultSet rs=ps.executeQuery();
while(rs.next()){
String pname= rs.getString("product_name");
int id= rs.getInt("product_id");
int qty= rs.getInt("quantity");
System.out.println(pname+" "+ id+" " + qty);
}



Update Table Data:

The executeUpdate() method is used to update records from the database table.

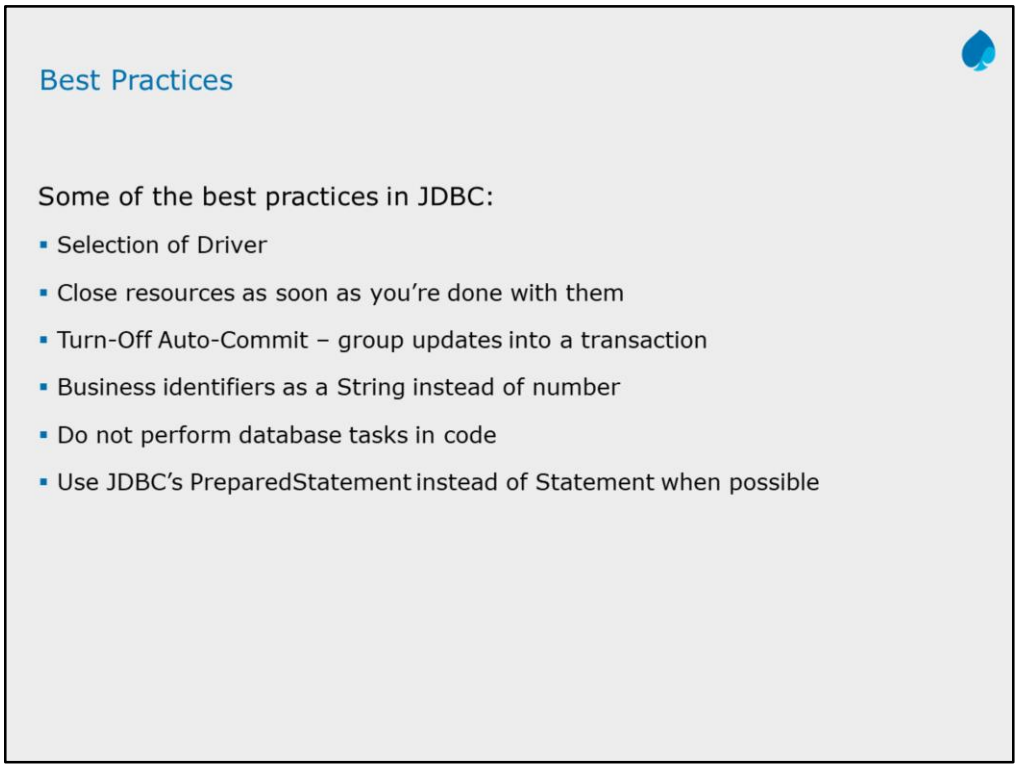
This method returns integer type value indicating the number of records affected in the database table.



Update Table Data:

The executeUpdate() method is also used to delete records from the database table.

This method returns integer type value indicating the number of records affected in the database table.



JDBC Best Practices:

Following are some of the best practices used in JDBC:

Selection of Driver

Select a certified, high performance type 2 (Thin) JDBC driver and use the latest drivers release.

Close resources as soon as you are done with them (in finally)

For example: Statements, Connections, Resultsets, and so on.

Failure to do so will eventually cause the application to "hang", and fail to respond to user actions.

Turn-Off Auto-Commit

It is best practice to execute the group of the statement together which are the part of the same transaction. It helps to avoid the overhead of data inconsistency.

JDBC Best Practices:

4. Business identifiers as a String instead of number

Many problem domains use numbers as business identifiers. Credit card numbers, bank account numbers, and the like, are often simply that - numbers. *It should always be kept in mind, however, that such items are primarily identifiers, and their numeric character is usually completely secondary*. Their primary function is to identify items in the problem domain, not to represent quantities of any sort.

For example: It almost never makes sense to operate on numeric business identifiers as true numbers - adding two account numbers, or multiplying an account number by -1, are meaningless operations. That is, one can strongly argue that modeling an account number as a Integer is inappropriate, simply because it does not behave as an Integer.

Furthermore, new business rules can occasionally force numeric business identifiers to be abandoned in favor of alphanumeric ones. It seems prudent to treat the content of such a business identifier as a business rule, subject to change. As usual, a program should minimize the ripple effects of such changes.

In addition, Strings can contain leading zeros, while numeric fields will remove them.

5. Do not perform database tasks in code

Databases are a mature technology, and they should be used to do as much work as possible. Do not do the following in code, if it can be done in SQL instead:

ordering (ORDER BY)
filtering based on criteria (WHERE)
joining tables (WHERE, JOIN)
summarizing (GROUP BY, COUNT, AVG, STDDEV)

Any corresponding task implemented entirely in code would very likely:

be *much* less robust and efficient
take longer to implement
require more maintenance effort

6. Use JDBC's PreparedStatement instead of Statement when possible for the following reasons:

It is in general more secure. When a Statement is constructed dynamically from user input, it is vulnerable to SQL injection attacks. PreparedStatement is not vulnerable in this way.

There is usually no need to worry about escaping special characters if repeated compilation is avoided, its performance is usually better.

In general, it seems safest to use a Statement only when the SQL is of fixed, known form, with no parameters

Lab : JDBC

Lab : JDBC 4.0

Summary

In this lesson, you have learnt:

- Establishing the connection with database to perform database operations
- Different types of statement creation
- Different ways of executing statements
- And best practices for JDBC applications

Review Question

Question 1 : Which Statement is used when you want to pass parameter to the query?

- **Option 1 :** Statement
- **Option 2 :** PreparedStatement
- **Option 3 :** CallableStatement

Question 2 : _____ method is best suited to execute DDL Queries.

Question 3 : By default, a connection object is in auto-commit mode

- True/False