

## Dynamic SQL

=====

In the previous code we are inserting static data in the table which means that every time the program will run it will insert same record which is unrealistic.

To solve this problem, JDBC provides us a concept called dynamic SQL where in the query we pass the data given by the user.

JDBC provides us 2 ways for writing dynamic SQL queries:

1. Using string concatenation.
2. Using **PreparedStatement** interface.

## Using String Concatenation

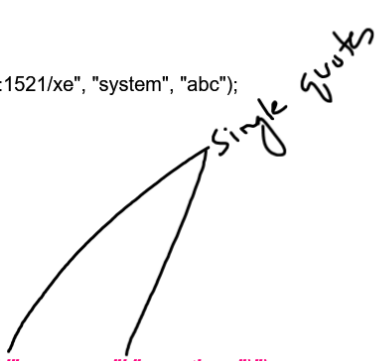
```
import java.sql.*;
import java.util.Scanner;

public class JdbcEx3 {
    public static void main(String[] args) {
        Connection conn=null;
        try{
            conn=DriverManager.getConnection("jdbc:oracle:thin:@//localhost:1521/xe", "system", "abc");
            System.out.println("Connected successfully to the DB");
            Statement st=conn.createStatement();

            Scanner kb=new Scanner(System.in);
            System.out.println("Enter movie id:");
            int id=kb.nextInt();
            System.out.println("Enter movie name");
            kb.nextLine();// for clearing the buffer
            String mname=kb.nextLine();
            System.out.println("Enter movie rating:");
            int mrating=kb.nextInt();

            int ans=st.executeUpdate("insert into allmovies values("+id+" "+" "+mname+" "+" "+mrating+" ")");
            System.out.println("Rows affected:"+ans);

        } // Rest of the code same as before
    }
}
```



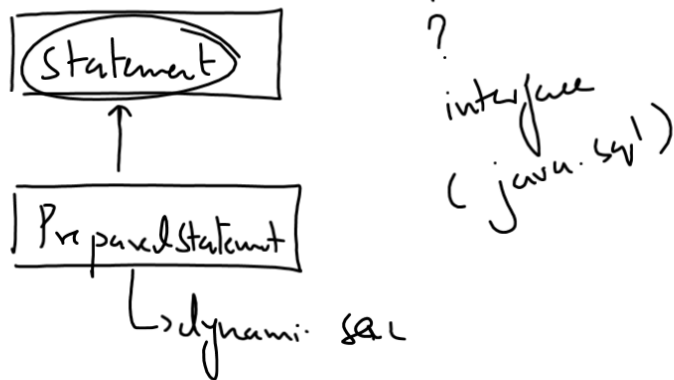
### Drawbacks of String concatenation

=====

1. Although, we are able to produce dynamic SQL but it is very difficult to concatenate variable names with query string since we have to use + operator multiple times.
2. If the database changes we might need to change the way of writing this query. For example: if our database is oracle and we are inserting Date from Java in oracle then we ourselves will have to use the format **DD-MON-YY**. Now if the database changes to MySQL the Date pattern will have to be changed. Thus the code will no flexible.
3. String concatenation is not safe from a very popular hacking attack called as **SQL Injection**.

So as a professional developer we should never use string concatenation approach rather we have a better approach called as **PreparedStatement**.

## Using PreparedStatement



① Create it's obj

```
PreparedStatement ps = conn.prepareStatement("Ins  
... - values (?, ?, ?)");
```

② Replace ? with act data

```
ps.setInt(1, id);  
ps.setString(2, mname);
```

Arrows from the circled '1' and '2' in the code point to the first and second '?' placeholders in the SQL statement above.

③ Execute

int an = ps.executeUpdate();