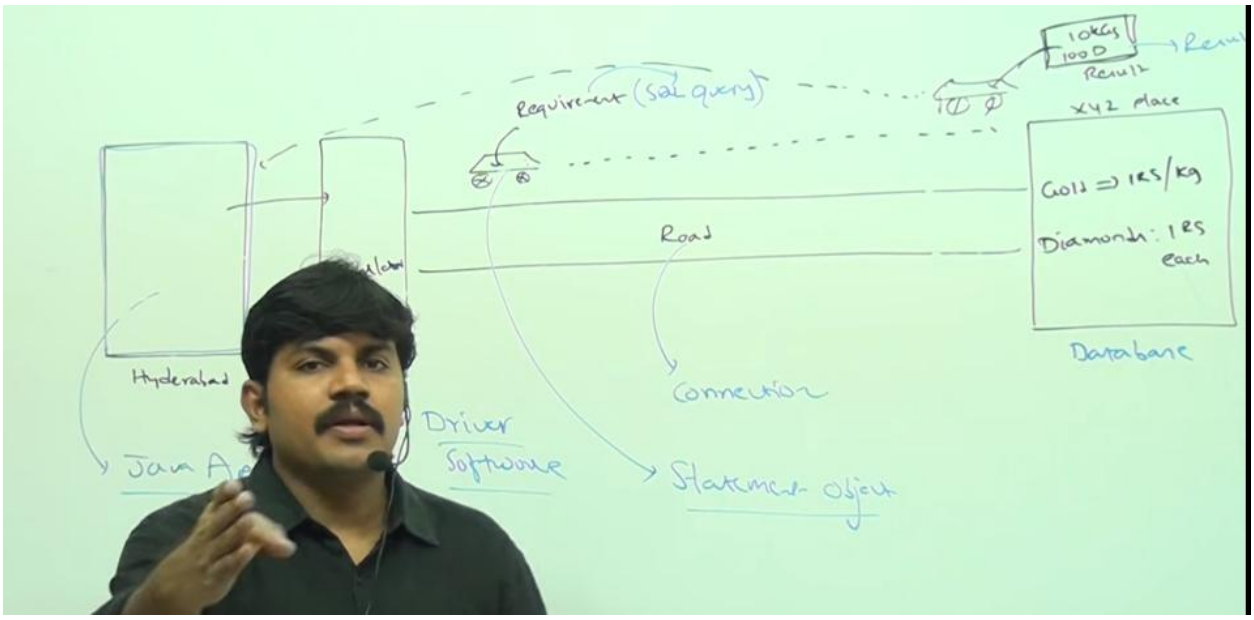


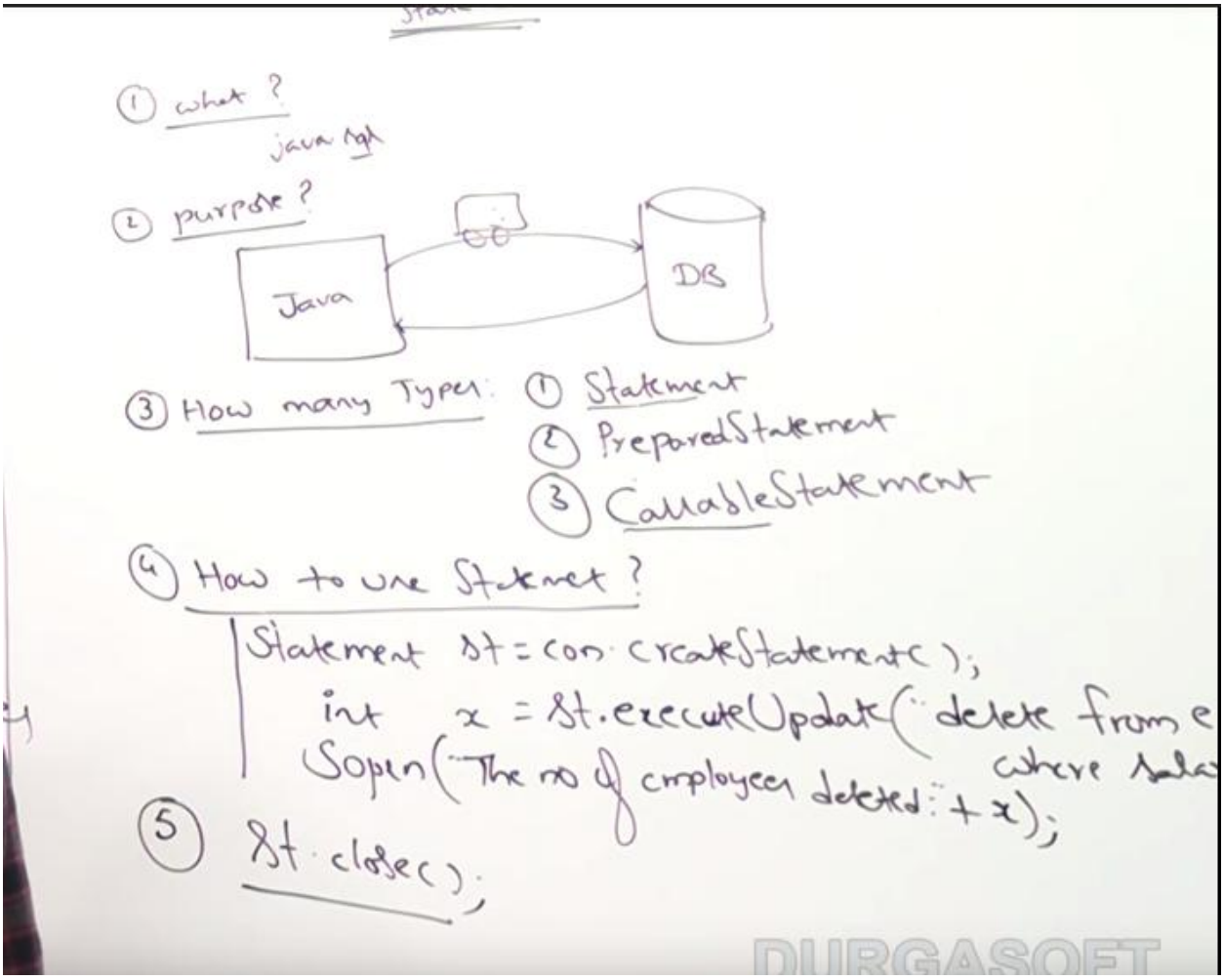

```
① // Class.forName("oracle.jdbc.OracleDriver");        [ojdbc6.jar  Oracle 11g]

② Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "scott");

③ Statement st = con.createStatement();

④ ResultSet rs = st.executeQuery("select * from gold_diamonds");
   while(rs.next())
   {
      System.out.println(rs.getString("item") + "..." + rs.getInt("quantity"));
   }

⑤ con.close();
```




```
import java.sql.*;
public class JDBCDemo
{
   public static void main(String[] args) throws Exception
   {
      //Class.forName("oracle.jdbc.OracleDriver");
      Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","scott","ti
      Statement st = con.createStatement();
      ResultSet rs =st.executeQuery("select * from gold_diamonds");
      while(rs.next())
      {
         System.out.println(rs.getString("item")+"...."+rs.getInt("quantity"));
      }
      con.close();
   }
}
```

JDBC Driver:

1. JDBC-ODBC bridge driver
2. Native-API Driver (Partially Java driver)
3. Network Protocol driver (fully Java driver)
4. Thin Driver (fully Java driver)

5 steps to connect to the database in java.

1. Load and Register Driver software(optional from JDBC 4)
2. Creating connection
3. Creating statement
4. Executing queries
5. Getting the result set
6. Closing connection

The forName() method of Class class is used to register the driver class.

This method is used to dynamically load the driver class.

Public static void forName(String className) throws classNotFoundException

Class.forName("oracle.jdbc.driver.OracleDriver");

DriverManager class:

- Acts as an interface between the user and the drives.

Connection Interface:

- A connection is the session between the java application and the database.
- The connection interface is a factory of Statement, PreparedStatement and DatabaseMetaData.

Statement Interface:

ResultSet interface:

- The object of Resultset maintains a curser pointing to a row of a table, cursor points to before the first row.

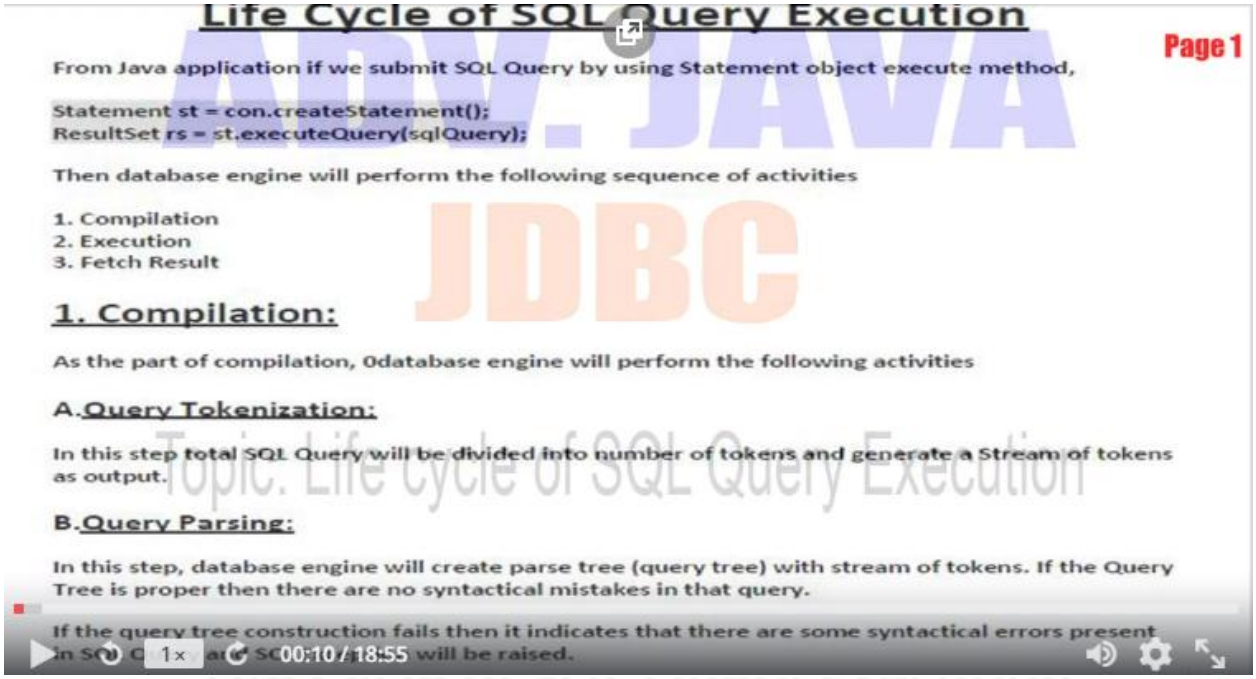
PreparedStatement interface:

It is a subinterface of Statement. It is used to execute parameterized query.
The performance of the application will be faster if you use preparedStatement, because query is compiled only once.

ResultSetMetaData Interface:

Gets metadata about table, like total number of column, column name, column type etc.

DatabaseMetadata Interface:



C.Query Optimization:

The main purpose of query optimization is to improve performance. In this step optimized query tree will be constructed.

2. Execution of sql Query:

Once compilation success then database engine will take that query tree as input and execute that query by using interpreter.

3. Fetch the Result:

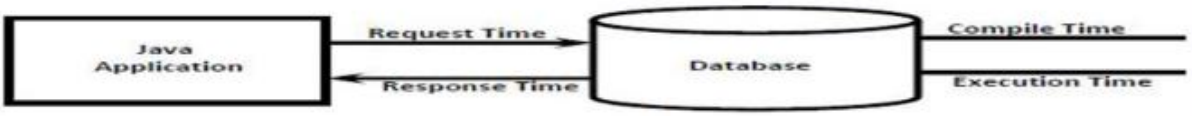
Database engine will provide result of SQL Query either in the form of ResultSet (for select query) OR in the form of rowCount (for non-select query) to the Java application.

Need of PreparedStatement:

In the case of normal Statement, whenever we are executing SQL Query, every time compilation and execution will be happened at database side.

Page 1

```
Statement st = con.createStatement();
ResultSet rs = st.executeQuery ("select * from employees");
```



Total Time per Query = Req.T+C.T+E.T+Resp.T
= 1 ms + 1 ms + 1 ms + 1 ms = 4ms

per 1000 Queries = 4 * 1000ms = 4000ms

Sometimes in our application,we required to execute same query multiple times with same or different input values.

Eg1:

In IRCTC application,it is common requirement to list out all possible trains between 2 places

```
select * from trains where source='XXX' and destination='YYY';
```

Query is same but source and destination places may be different. This query is required to execute lakhs of times per day.

Eg2:

In BookMyShow application, it is very common requirement to display theatre names where a particular movie running in a particular city

```
select * from theatres where city='XXX' and movie='YYY';
```

1× 00:11 / 21:21

In this case this query is required to execute lakhs of times per day. May be with different movie names and different locations.

For the above requirements if we use Statement object, then the query is required to compile and execute every time, which creates performance problems.

Page 2

To overcome this problem, we should go for PreparedStatement.

The main advantage of PreparedStatement is the query will be compiled only once even though we are executing multiple times, so that overall performance of the application will be improved.

We can create PreparedStatement by using prepareStatement() method of Connection interface.

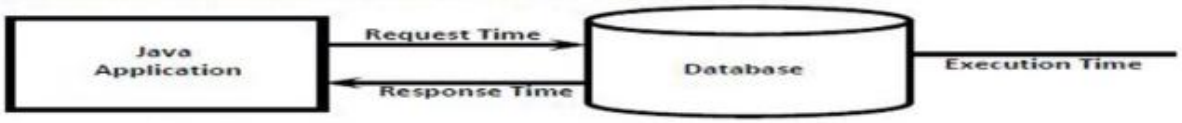
public PreparedStatement prepareStatement(String sqlQuery) throws SQLException

Eg: PreparedStatment pst=con.prepareStatement(sqlQuery);

At this line,sqlQuery will send to the database. Database engine will compile that query and stores in the database.
That pre compiled query will be returned to the java application in the form of PreparedStatement object.

Hence PreparedStatement represents "pre compiled sql query".

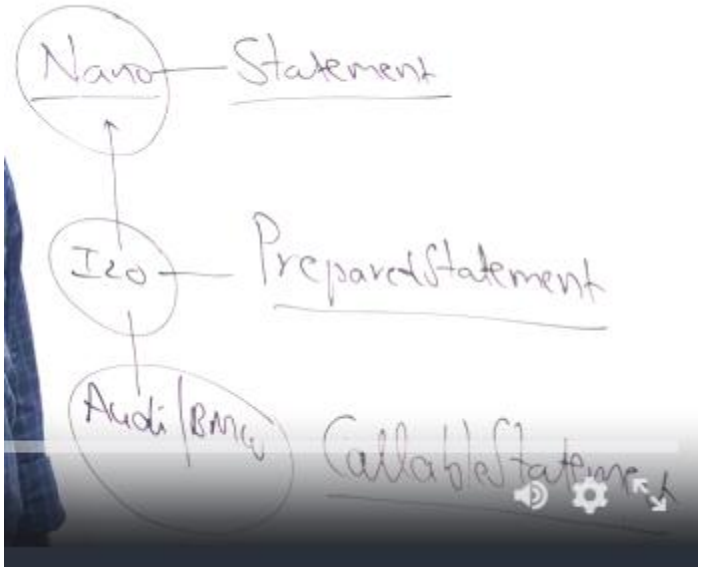
Whenever we call execute methods,database engine won't compile query once again and it will directly execute that query,so that overall performance will be improved.



Total Time per Query = Req.T+Resp.T+E.T
= 1 ms + 1 ms + 1 ms = 3ms

1000 Queries = 3000ms

1× 00:15 / 21:21



Differences Between Statement And PreparedStatement

Statement	PreparedStatement
1) At the time of creating Statement Object, we are not required to provide any Query. Statement st = con.createStatement(); Hence Statement Object is not associated with any Query and we can use for multiple Queries.	1) At the time of creating PreparedStatement, we have to provide SQL Query compulsory and will send to the Database and will be compiled. PS pst = con.prepareStatement(query); Hence PS is associated with only one Query.
2) Whenever we are using execute Method, every time Query will be compiled and executed.	2) Whenever we are using execute Method, Query won't be compiled just will be executed.
3) Statement Object can work only for Static Queries.	3) PS Object can work for both Static and Dynamic Queries.
4) Relatively Performance is Low.	4) Relatively Performance is High.
5) Best choice if we want to work with multiple Queries.	5) Best choice if we want to work with only one Query but required to execute multiple times.
6) There may be a chance of SQL Injection Attack.	6) There is no chance of SQL Injection Attack.
7) Inserting Date and Large Objects (CLOB and BLOB) is difficult.	7) Inserting Date and Large Objects (CLOB and BLOB) is easy.

Stored Procedures Introduction

In our programming if any code repeatedly required,then we can define that code inside a method and we can call that method multiple times based on our requirement.

Page 1

Hence method is the best reusuable component in our programming.

Similarly in the database programming,if any group of sql statements is repeatedly required then we can define those sql statements in a single group and we can call that group repeatedly based on our requirement.

This group of sql statements that perform a particular task is nothing but Stored Procedure.Hence stored procedure is the best reusable component at database level.

Hence Stored Procedure is a group of sql statements that performs a particular task.

These procedures stored in database permanently for future purpose and hence the name stored procedure.

Usually stored procedures are created by Database Admin (DBA).

Every database has its own language to create Stored Procedures.

Stored Procedures Introduction

Oracle has → PL/SQL
MySQL has → Stored Procedure Language
Microsoft SQL Server has → Transact SQL(TSQL

Page 2

Similar to methods stored procedure has its own parameters. Stored Procedure has 3 Types of parameters.

1. IN parameters(to provide input values)
2. OUT parameters(to collect output values)
3. INOUT parameters(to provide input and to collect output)

Eg 1 :

Z:=X+Y;

X,Y are IN parameters and Z is OUT parameter

Eg 2:

X:=X+X;