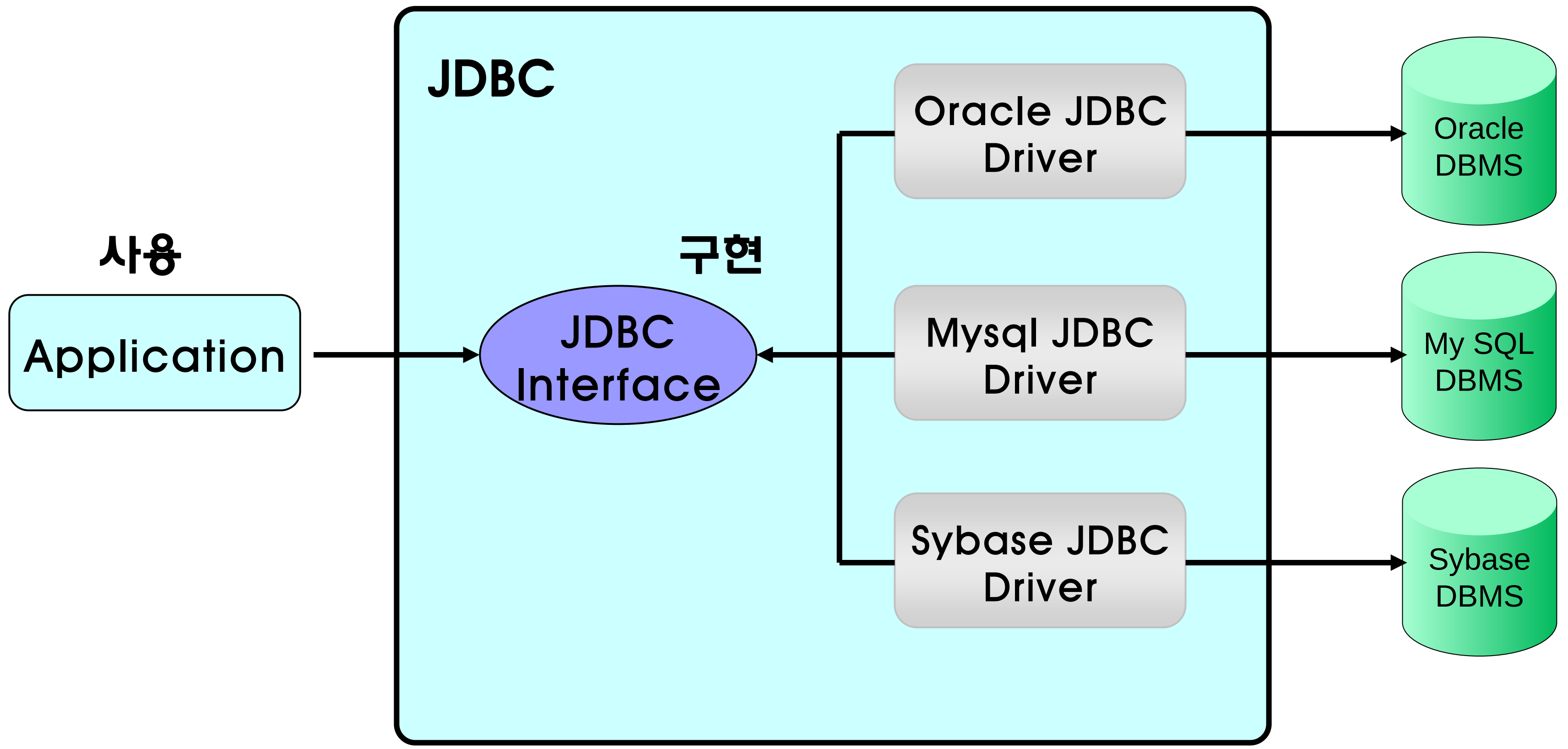
Java Programming with JDBC

학습 목표

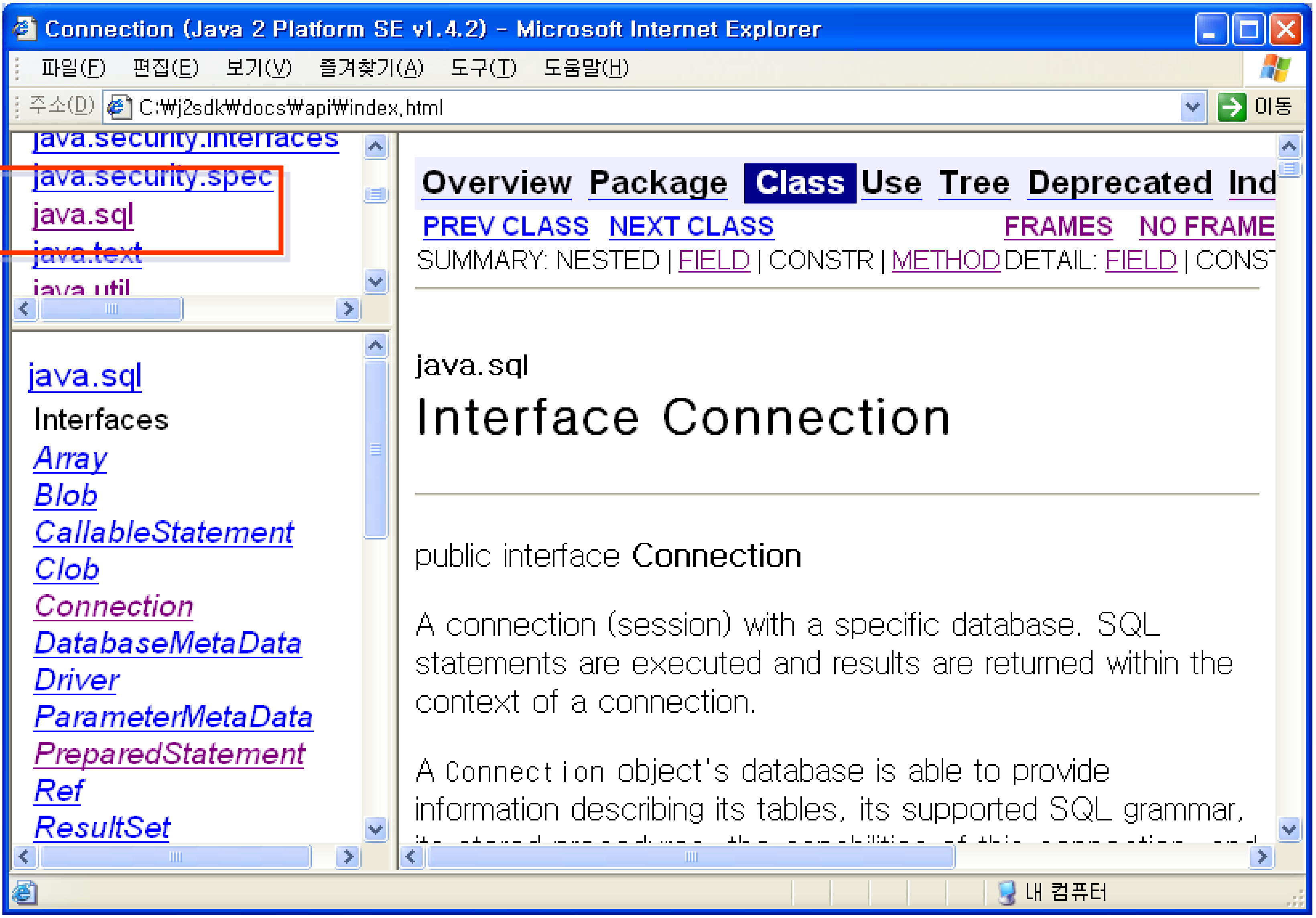
1. JDBC 개요
2. JDBC Coding 절차
3. SELECT / UPDATE
4. Statement/ PreparedStatement

JDBC (Java Database Connectivity)

❑ 자바 언어에서 Database에 접근할 수 있게 해주는 Programming API



JDBC API



Connection (Java 2 Platform SE v1.4.2) - Microsoft Internet Explorer

파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)

주소(D) C:₩j2sdk₩docs₩api₩index.html 이동

java.security.interfaces
java.security.spec
java.sql
java.text
java.util

java.sql

Interfaces
Array
Blob
CallableStatement
Clob
Connection
DatabaseMetaData
Driver
ParameterMetaData
PreparedStatement
Ref
ResultSet

Overview Package **Class** Use Tree Deprecated Ind

PREV CLASS NEXT CLASS FRAMES NO FRAME

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONS

java.sql

Interface Connection

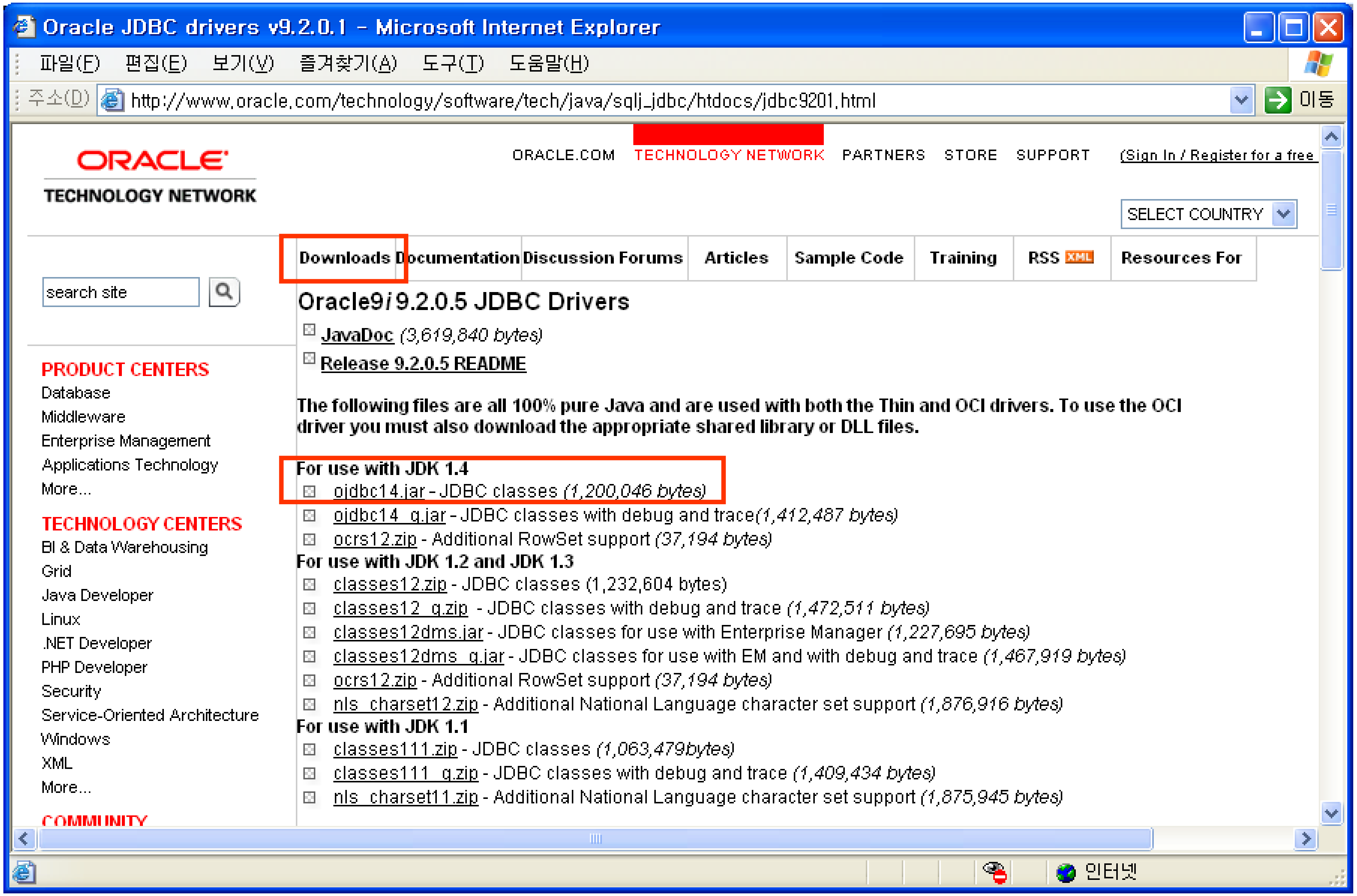
public interface **Connection**

A connection (session) with a specific database. SQL statements are executed and results are returned within the context of a connection.

A `Connection` object's database is able to provide information describing its tables, its supported SQL grammar,

내 컴퓨터

JDBC Driver Download – Oracle (www.oracle.com)


Oracle JDBC drivers v9.2.0.1 - Microsoft Internet Explorer
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
주소(D) http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc9201.html
이동
ORACLE TECHNOLOGY NETWORK
ORACLE.COM TECHNOLOGY NETWORK PARTNERS STORE SUPPORT (Sign In / Register for a free
SELECT COUNTRY
Downloads Documentation Discussion Forums Articles Sample Code Training RSS XML Resources For
search site
PRODUCT CENTERS
Database
Middleware
Enterprise Management
Applications Technology
More...
TECHNOLOGY CENTERS
BI & Data Warehousing
Grid
Java Developer
Linux
.NET Developer
PHP Developer
Security
Service-Oriented Architecture
Windows
XML
More...
COMMUNITY
인터넷

Oracle9*i* 9.2.0.5 JDBC Drivers

- JavaDoc *(3,619,840 bytes)*
- **Release 9.2.0.5 README**

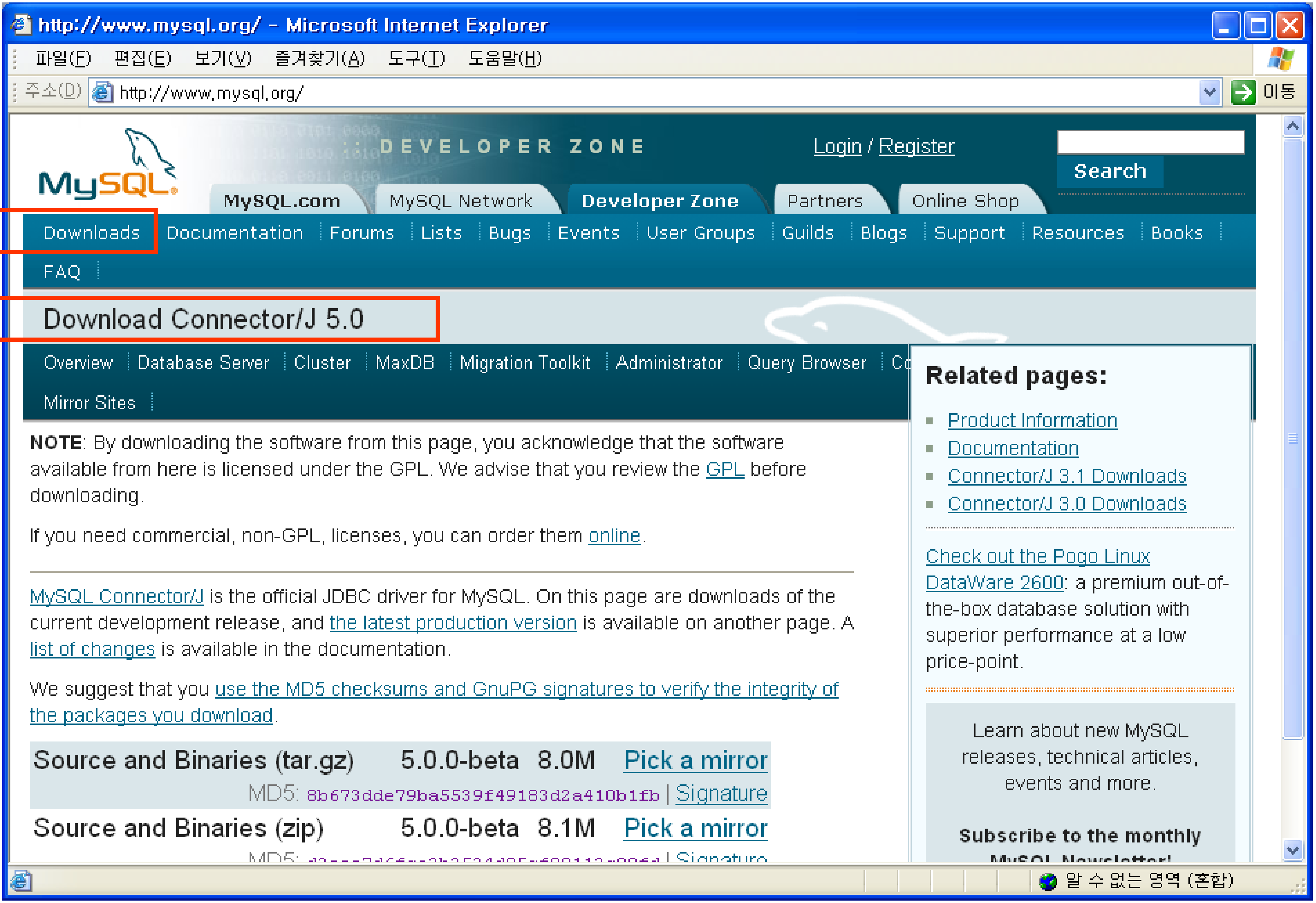
The following files are all 100% pure Java and are used with both the Thin and OCI drivers. To use the OCI driver you must also download the appropriate shared library or DLL files.

For use with JDK 1.4
- ojdbc14.jar - JDBC classes *(1,200,046 bytes)*
- ojdbc14_g.jar - JDBC classes with debug and trace*(1,412,487 bytes)*
- ocrs12.zip - Additional RowSet support *(37,194 bytes)*

For use with JDK 1.2 and JDK 1.3
- classes12.zip - JDBC classes (1,232,604 bytes)
- classes12_g.zip - JDBC classes with debug and trace *(1,472,511 bytes)*
- classes12dms.jar - JDBC classes for use with Enterprise Manager *(1,227,695 bytes)*
- classes12dms_g.jar - JDBC classes for use with EM and with debug and trace *(1,467,919 bytes)*
- ocrs12.zip - Additional RowSet support *(37,194 bytes)*
- nls_charset12.zip - Additional National Language character set support *(1,876,916 bytes)*

For use with JDK 1.1
- classes111.zip - JDBC classes *(1,063,479bytes)*
- classes111_g.zip - JDBC classes with debug and trace *(1,409,434 bytes)*
- nls_charset11.zip - Additional National Language character set support *(1,875,945 bytes)*

JDBC Driver Download – MySql (www.mysql.org)



Download Connector/J 5.0

NOTE: By downloading the software from this page, you acknowledge that the software available from here is licensed under the GPL. We advise that you review the GPL before downloading.

If you need commercial, non-GPL, licenses, you can order them online.

MySQL Connector/J is the official JDBC driver for MySQL. On this page are downloads of the current development release, and the latest production version is available on another page. A list of changes is available in the documentation.

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

Source and Binaries (tar.gz)	5.0.0-beta	8.0M	Pick a mirror
MD5: 8b673dde79ba5539f49183d2a410b1fb | Signature			
Source and Binaries (zip)	5.0.0-beta	8.1M	Pick a mirror

Related pages:

- Product Information
- Documentation
- Connector/J 3.1 Downloads
- Connector/J 3.0 Downloads

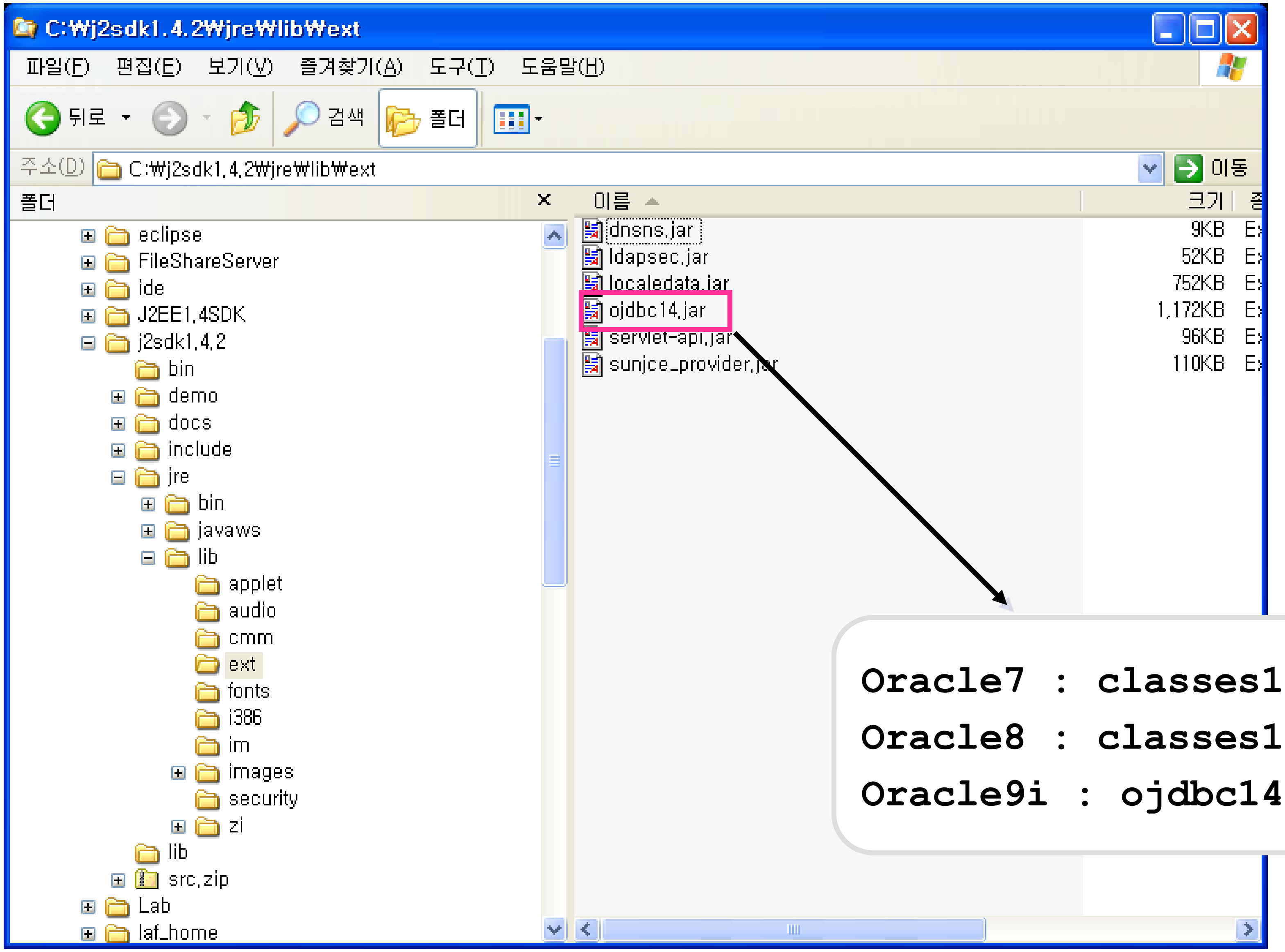
Check out the Pogo Linux DataWare 2600: a premium out-of-the-box database solution with superior performance at a low price-point.

Learn about new MySQL releases, technical articles, events and more.

Subscribe to the monthly

JDBC Driver

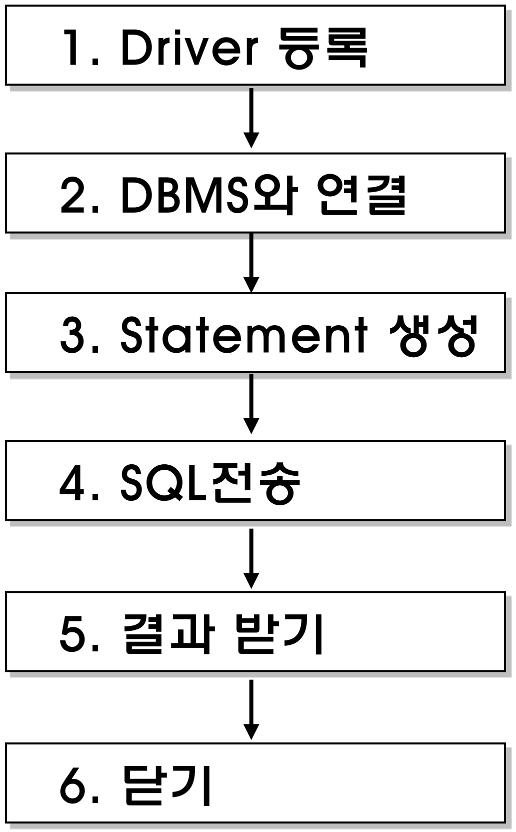
❑ *JAVA_HOME* \jre\lib\ext 에 driver를 추가해야 함 : ojdbc14.jar



이름	크기
dnsns.jar	9KB
ldapsec.jar	52KB
localedata.jar	752KB
ojdbc14.jar	1,172KB
servlet-api.jar	96KB
sunjce_provider.jar	110KB

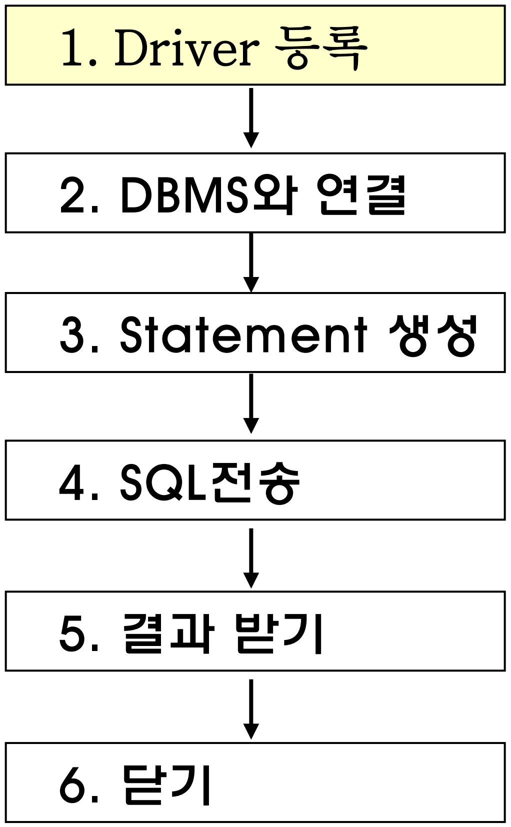
```
Oracle7  : classes111.zip
Oracle8  : classes12.zip
Oracle9i : ojdbc14.jar
```

JDBC – JDBC Coding 절차



JDBC – JDBC Coding 절차

1. DriverManager에 해당 DBMS Driver 등록



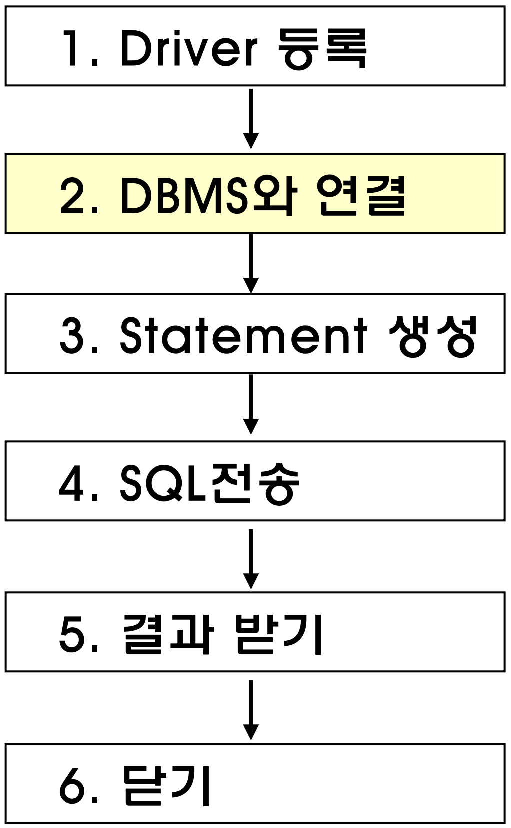
```
Class.forName( "oracle.jdbc.driver.OracleDriver“ );
```

cf)

```
Class.forName( “com.microsoft.jdbc.sqlserver.SQLServerDriver” );

Class.forName( "org.gjt.mm.mysql.Driver“ );
```

JDBC – JDBC Coding 절차

2. 해당 Driver로부터 Connection instance를 획득



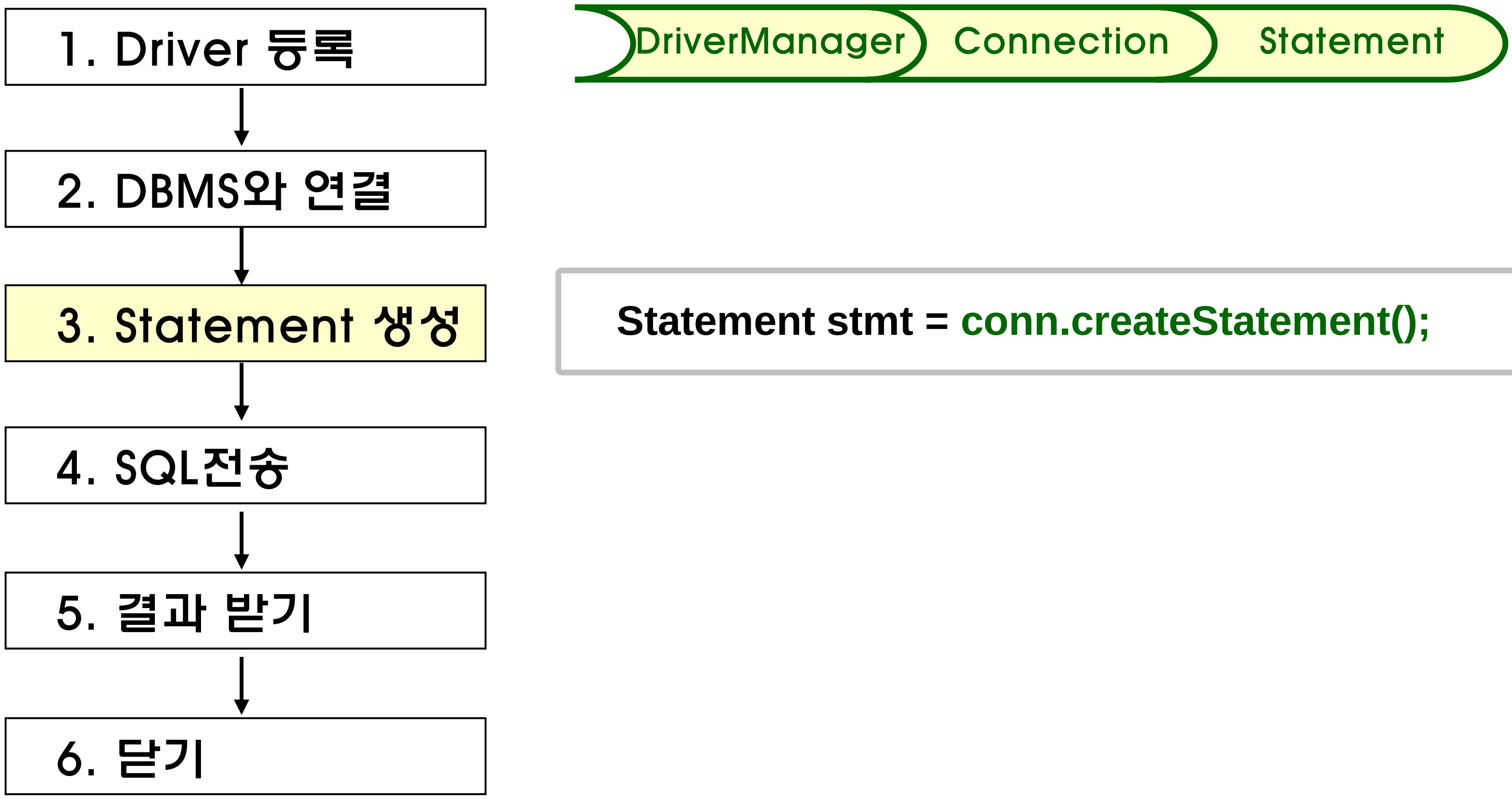
DriverManager
Connection

```
public static Connection getConnection( String url,
                                        String user,
                                        String password )
throws SQLException
```

```
Connection conn =
    DriverManager.getConnection(
        “jdbc:oracle:thin:@192.168.0.200:1521:VCC”,
        “SEXXXXX”,
        “SEXXXXX” );
```

JDBC – JDBC Coding 절차

3. Connection instance로부터 Statement instance획득



```
Statement stmt = conn.createStatement();
```

JDBC – JDBC Coding 절차

4. Statement method를 이용하여 SQL 실행
5. 실행후 결과를 ResultSet(SELECT) 혹은 int형 변수(DML)로 받아 처리

$$\begin{tikzcd}
\boxed{\text{1. Driver 등록}} \arrow[d] \\
\boxed{\text{2. DBMS와 연결}} \arrow[d] \\
\boxed{\text{3. Statement 생성}} \arrow[d] \\
\boxed{\text{4. SQL전송}} \arrow[d] \\
\boxed{\text{5. 결과 받기}} \arrow[d] \\
\boxed{\text{6. 닫기}}
\end{tikzcd}$$

DriverManager | Connection | Statement | ResultSet

Select

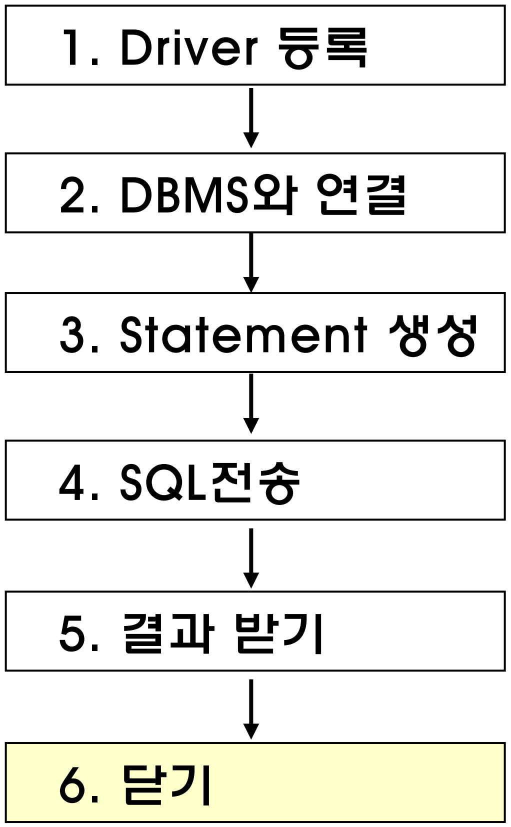
```
String query = "SELECT ID, LAST_NAME FROM EMP";
ResultSet rset = stmt.executeQuery( query );
while ( rset.next() ) {
    System.out.println( rset.getString( "ID" ) + "\t" +
                                rset.getString( 2 ) );
}
```

DML

```
String query = "UPDATE EMP  " +
                        "  SET LAST_NAME  =  'KIM'  "+
                        "  WHERE  ID =  '100000'  ";
int result = stmt.executeUpdate( query );
```

JDBC – JDBC Coding 절차

6. 사용한 자원 반납



Select

```
rset.close();
stmt.close();
conn.close();
```

DML

```
stmt.close();
conn.close();
```

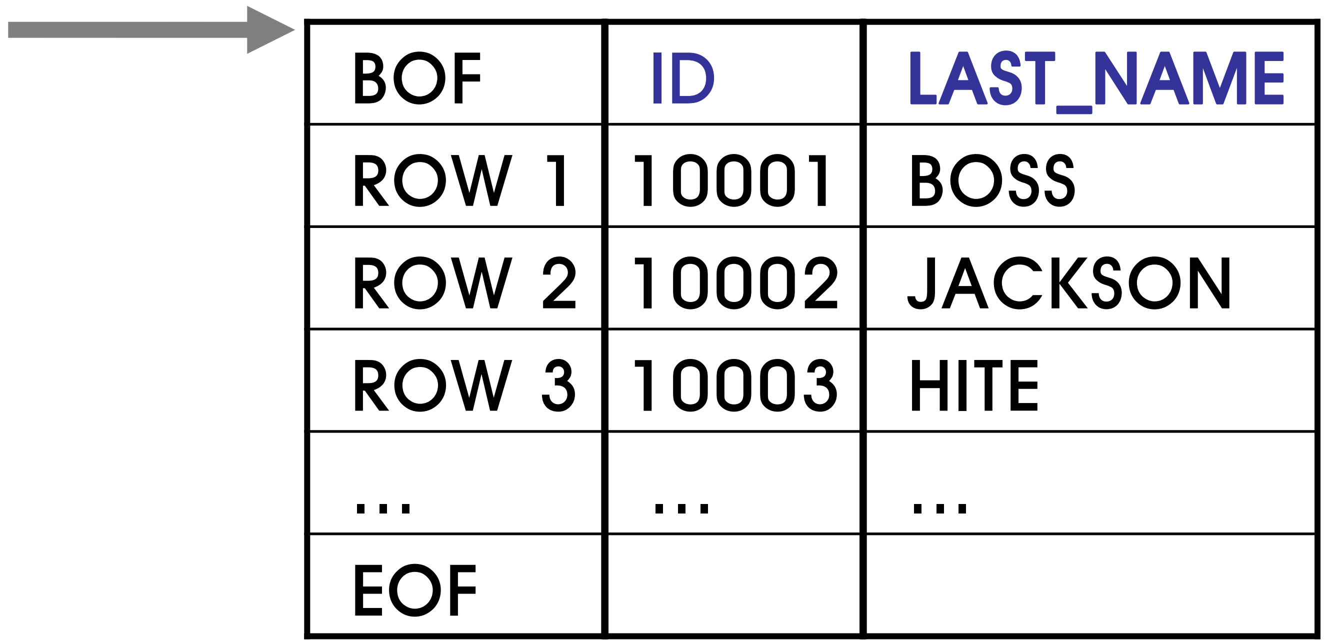
JDBC – SELECT

```
package jdbc;

import java.sql.*;

public class EmpList {

    public static void main( String[] args )
                            throws SQLException, ClassNotFoundException  {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rset = null;
        String url = "jdbc:oracle:thin:@192.168.0.200:1521:VCC";
        String query = null;

        // 1. DBMS Driver 로딩
        Class.forName( "oracle.jdbc.driver.OracleDriver“ );

        // 2. Connection 객체 획득
        conn = DriverManager.getConnection( url , “SEXXXXX" , “SEXXXXX“ );

        // 3. Statement 객체 생성
        stmt = conn.createStatement();

        // 4. SQL 실행
        query =  "SELECT ID   "           +
                 "          ,LAST_NAME "   +
                 " FROM EMP " ;
```

JDBC – SELECT

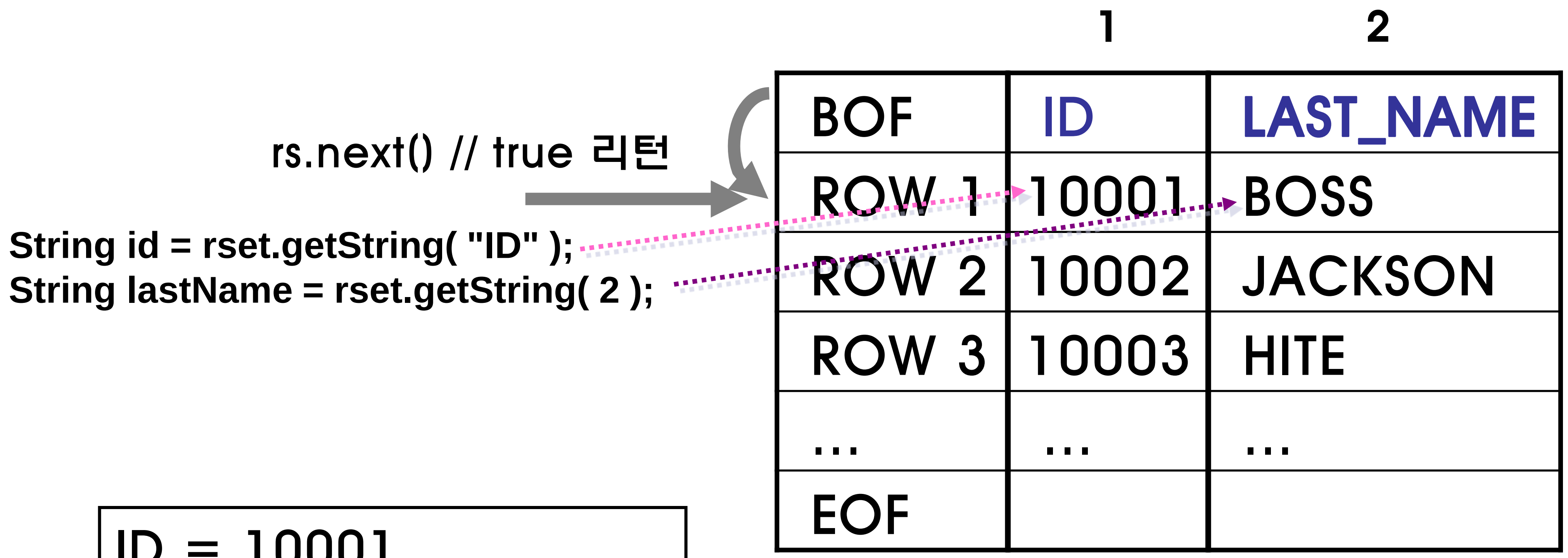
```
        rset = stmt.executeQuery( query.toString() );

        System.out.println( "ID\t\t\tLAST_NAME\n“ );
        System.out.println( "======================================== \n“ );

        // 5. ResultSet을 이용한 결과 처리
        while( rset.next() ){
            System.out.println( rset.getString( "ID“ ) + "\t\t\t" +
                    rset.getString( 2 ) );
        }

        // 6. 사용할 Resource 반납
        rset.close();
        stmt.close();
        conn.close();
    }
}
```

JDBC – SELECT



BOF	ID	LAST_NAME
ROW 1	10001	BOSS
ROW 2	10002	JACKSON
ROW 3	10003	HITE
...	...	...
EOF		

JDBC – SELECT



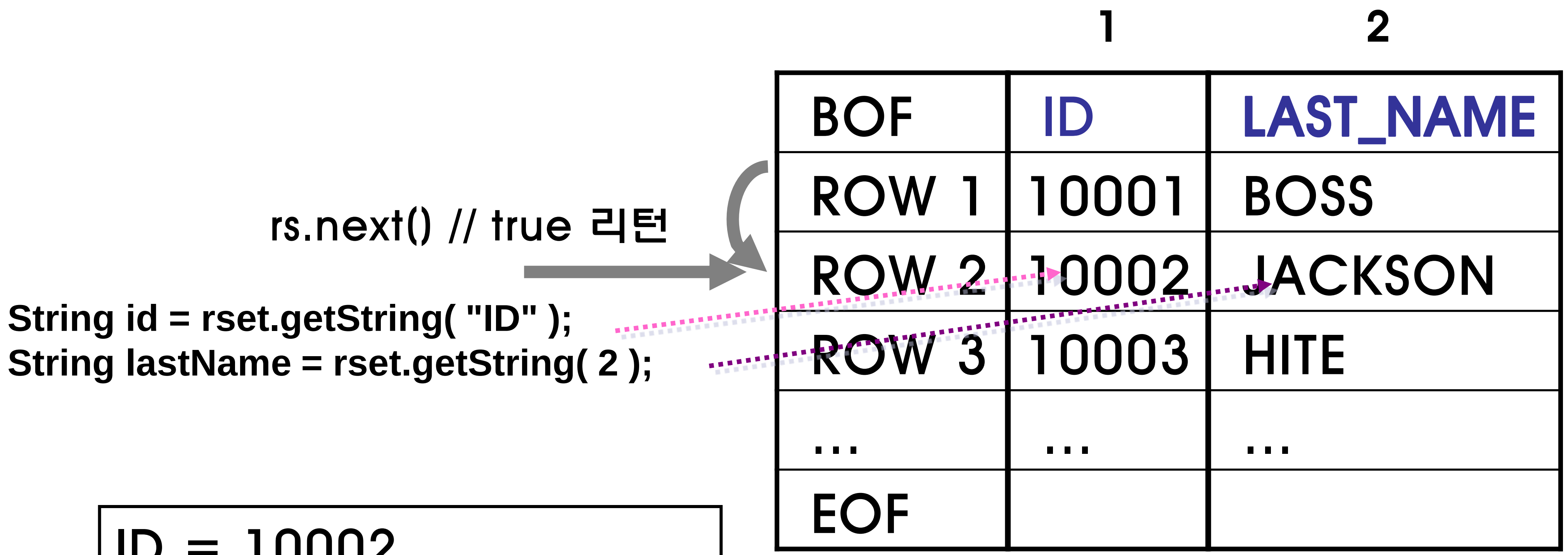
```
String id = rset.getString( "ID" );
String lastName = rset.getString( 2 );
```

	1	2
BOF	ID	LAST_NAME
ROW 1	10001	BOSS
ROW 2	10002	JACKSON
ROW 3	10003	HITE
...	...	...
EOF		

ID = 10001

LAST_NAME = BOSS

JDBC – SELECT



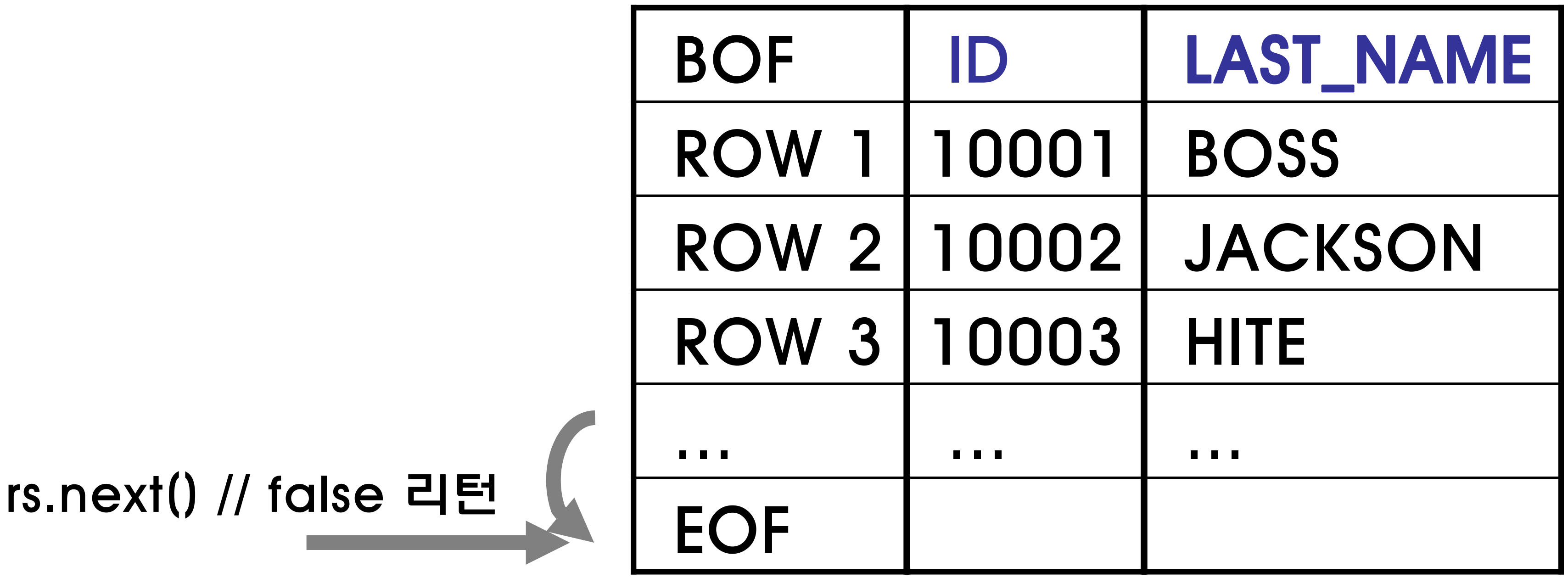
rs.next() // true 리턴

```
String id = rset.getString( "ID" );
String lastName = rset.getString( 2 );
```

	1	2
BOF	ID	LAST_NAME
ROW 1	10001	BOSS
ROW 2	10002	JACKSON
ROW 3	10003	HITE
...	...	...
EOF		

```
ID = 10002

LAST_NAME = JACKSON
```

JDBC – SELECT



BOF	ID	LAST_NAME
ROW 1	10001	BOSS
ROW 2	10002	JACKSON
ROW 3	10003	HITE
...	...	...
EOF		

JDBC – SELECT

❑ StringBuffer Class 사용

```
StringBuffer query = new StringBuffer();
try{
        // 1. DBMS Driver 로딩
        Class.forName( "oracle.jdbc.driver.OracleDriver“ );

        ....
        // 4. SQL 실행
        query.append( “SELECT ID                    " )
                  .append( "            , LAST_NAME " )
                  .append( "FROM EMP                " );
       rs = stmt.executeQuery( query.toString() );
        ....
   } catch( ClassNotFoundException ce){
        ce.printStackTrace();
   } catch( SQLException se){
        se.printStackTrace();
   } finally {
        // 6. 사용할 Resource 반납
        try {
            rs.close();
            stmt.close();
            conn.close();
        } catch ( SQLException e ) {
            e.printStackTrace();
        }
   }
```

JDBC – SELECT

❑ Exception Handling 로직 추가

```
StringBuffer query = new StringBuffer();
try{
     // 1. DBMS Driver 로딩
     Class.forName( "oracle.jdbc.driver.OracleDriver“ );


     ....
     // 4. SQL 실행
     query.append( “SELECT ID              " )
           .append( "         , LAST_NAME " )
           .append( "FROM EMP             " );
    rs = stmt.executeQuery( query.toString() );
     ....
  } catch( ClassNotFoundException ce){
     ce.printStackTrace();
  } catch( SQLException se){
     se.printStackTrace();
  } finally {
     // 6. 사용할 Resource 반납
     try {
        rs.close();
        stmt.close();
        conn.close();
     } catch ( SQLException e ) {
        e.printStackTrace();
     }
  }
```

JDBC – UPDATE Example

```
public class UpdateTest {

    public static void main( String[] args ) throws  SQLException, ClassNotFoundException {
        Connection conn = null;
        Statement stmt = null;
        String url = "jdbc:oracle:thin:@192.168.0.200:1521:VCC";
        StringBuffer query = new StringBuffer();
        int updateCount = 0;

        Class.forName( "oracle.jdbc.driver.OracleDriver“ );
        conn = DriverManager.getConnection( url , “SEXXXXX" , “SEXXXXX" );
        conn.setAutoCommit( false );

        query.append( "UPDATE EMP                    “ )
                .append( "SET LAST_NAME = 'HITE'     “ )
                .append( "WHERE ID = '10004'         “ );
        stmt = conn.createStatement();

        updateCount = stmt.executeUpdate( query.toString() );
        System.out.println( "업데이트된 행의 갯수 : " + updateCount );

        if( updateCount == 1 ){
            conn.commit();
        }else{
            conn.rollback();
        }

        stmt.close();
        conn.close();
```

JDBC – PreparedStatement

```
public class PreparedUpdateTest {

    public static void main( String[] args ) throws SQLException, ClassNotFoundException {
        Connection conn = null;
        PreparedStatement pstmt = null;
        String url = "jdbc:oracle:thin:@192.168.0.200:1521:VCC";
        StringBuffer query = new StringBuffer();
        int updateCount = 0;

        Class.forName( "oracle.jdbc.driver.OracleDriver“ );
        conn = DriverManager.getConnection( url , “SEXXXXX" , “SEXXXXX" );
        conn.setAutoCommit( false );
        query.append( "UPDATE EMP              “ )
                .append( "SET LAST_NAME = ?   “ )
                .append( "WHERE ID = ?            “ );

        pstmt = conn.preparedStatement( query.toString() );
        pstmt.setString( 1, "HITE2" );
        pstmt.setString( 2, "10005" );

        updateCount = pstmt.executeUpdate();
        if( updateCount == 1 ){
            conn.commit();
        }else{
            conn.rollback();
        }

        pstmt.close();
        conn.close();
```

JDBC – Statement vs PreparedStatement

	Statement	PreparedStatement
장점	원하는 Query를 직접 넣어주기 때문에 직관적이고 사용하기 쉽다.	같은 Query를 반복 수행해야 하는 경우 성능이 좋다. (loop 이용이 용이)
단점	실행시마다 SQL문을 해석해서 오버헤드가 크다.	코드가 길어질 수 있다.

Sample	Statement stmt = conn.createStatement();
	
	stmt.executeUpdate("Insert into emp values ('21421', 'Kim')");
	stmt.executeUpdate("Insert into emp values ('32211', Hong)");
	
	…
		PreparedStatement pstmt = conn.preparedStatement(“ Insert into emp values (?, ?) “);
		
		pstmt.setString(1, "21421");
		pstmt.setInt(2, "Kim");
		pstmt.executeUpdate();
		
		pstmt.setString(1, "32211");
		pstmt.setInt(2, "Hong");
		pstmt.executeUpdate();
		
		…