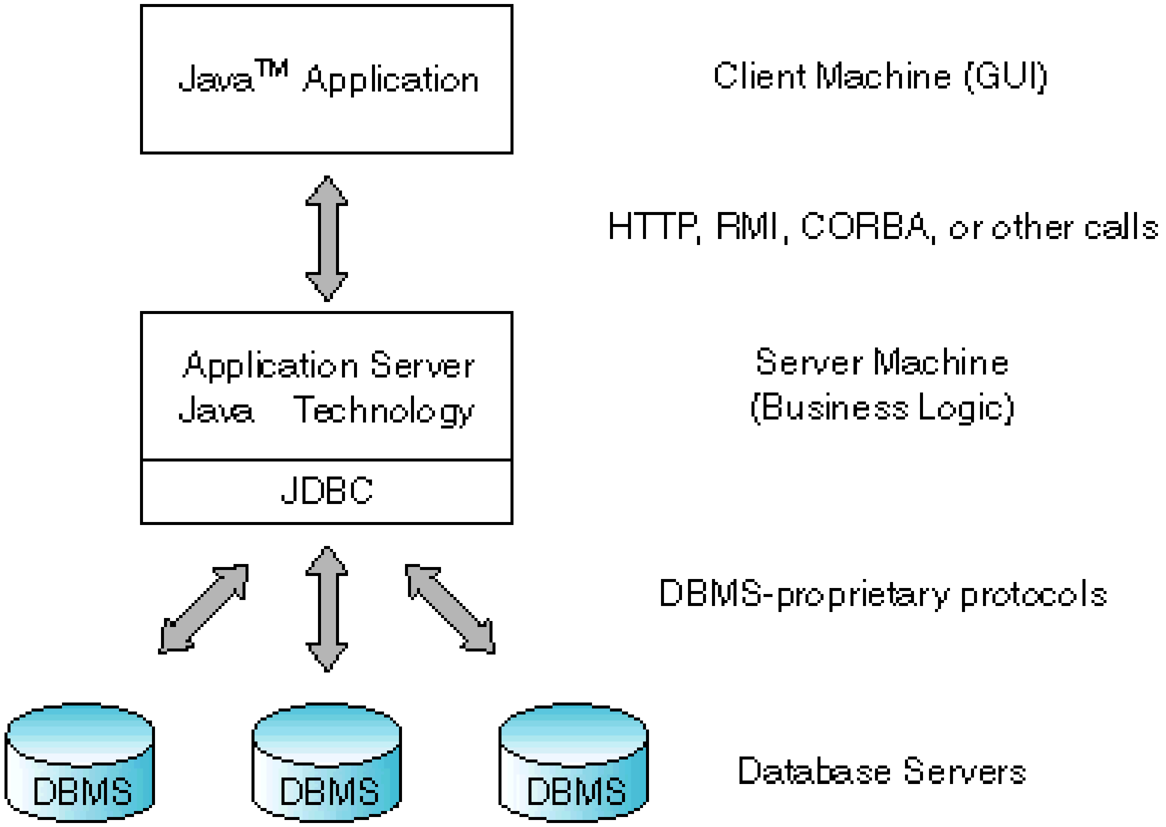
Working with Databases using JDBC


Java™ Application
Client Machine (GUI)
HTTP, RMI, CORBA, or other calls
Application Server Java Technology
JDBC
Server Machine (Business Logic)
DBMS-proprietary protocols
DBMS
DBMS
DBMS
Database Servers

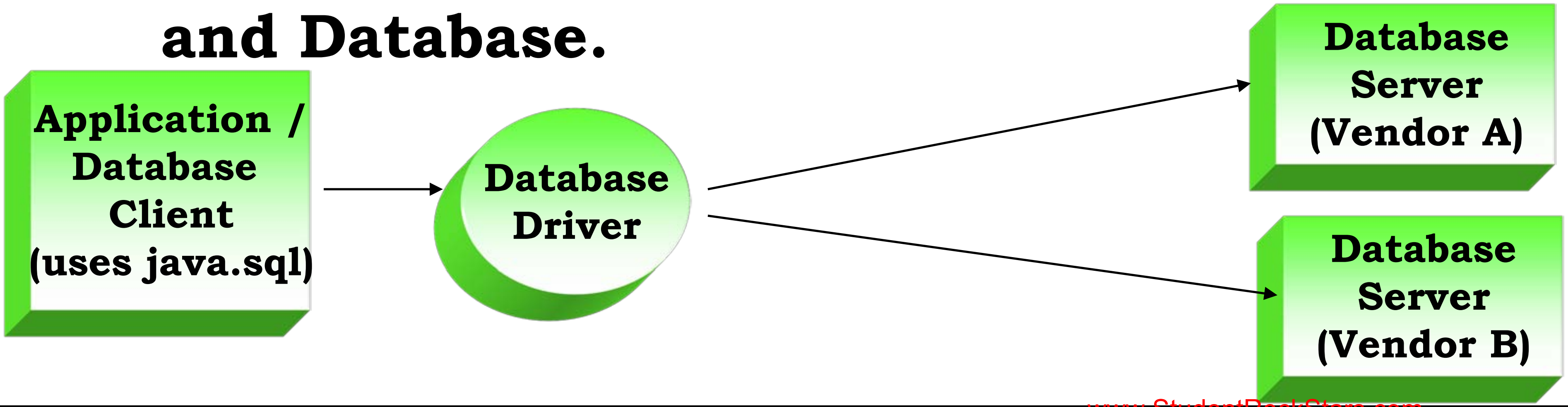
Introduction

- JavaSoft released **java.sql** as an optional package to be used with JDK1.0.2
- This package is referred to as JDBC.
- JDBC is now an integral part of JDK since verion JDK1.1 onwards.
- Though JDBC is often referred to as *Java DataBase Connectivity*, JavaSoft says JDBC is not an acronym.

JDBC BASICS

JDBC usage can be segregated into following :

1. **Application** – that uses java.sql API to retrieve/query a database.
2. **Database** – a repository system for organizing data in a structured way.
3. **Database Driver** – A separate entity which provides interface between the Application and Database.


Application / Database Client (uses java.sql)
Database Driver
Database Server (Vendor A)
Database Server (Vendor B)

Bridge Driver

- ODBC is a Microsoft provided, popular Database driver, which can communicate to any database.
- JavaSoft along with Intersolv developed their first driver named JDBC-ODBC bridge driver, which uses ODBC to communicate with a database.
- Eventually, objective was to make available drivers which can communicate directly, which is happening currently.

  Example: oci driver, IDS driver…
- The bridge driver was only a temporary solution.

Bridge driver

- Why not use ODBC driver directly?

  - ODBC was developed using C.
  - Accessing ODBC directly from a Java program could be problematic due to usage of pointers and programming constructs.
  - Rewriting ODBC in Java could be time consuming.

Driver Types

1. JDBC-ODBC Bridge plus ODBC Driver – Partly Java
2. Native-API Partly Java Driver – Partly Java. It talks to database using native driver of the database
   Ex: IBM Database protocol for DB2.
   SQLNet protocol for Oracle.
3. JDBC-Net Pure Java Driver – uses standard protocol (ex: HTTP). It communicates to a database access server which translates to database specific protocol. Ex: IDS JDBC Driver
4. Native-Protocol Pure Java Driver – It uses database specific protocol. Ex: MM.MySQL

Steps to using Bridge driver

1. Create a data source name using ODBC
2. Load the database driver
3. Establish a Connection to the database
4. Create a Statement object
5. Execute SQL Query statement(s)
6. Retrieve the ResultSet Object
7. Retrieve record/field data from ResultSet object for processing
8. Close ResultSet Object
9. Close Statement Object
10. Close Connection Object

Sample Code - 1

```
import java.sql.*;

public class first_jdbc_prg
{
        public static void main( String args[] )
        {
                try{www.StudentRockStars.com
                        Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
                }catch( Exception e ){        System.out.println( e );        }

                try{
                        Connection con = DriverManager.getConnection(
"jdbc:odbc:test" );
                        Statement stmt = con.createStatement();

                        String query_string = "select * from courses";

                        ResultSet rs = stmt.executeQuery( query_string );
```

Sample Code -1 (contd)

```
                String course_id, dept_id, course_no, course_lvl, course_name;

                while( rs.next() )
                {

                        course_id = rs.getString( "Course_ID" );
                        dept_id = rs.getString( "Department_ID" );
                        course_no = rs.getString( "CourseNumber" );
                        course_lvl = rs.getString( "CourseLevel" );
                        course_name = rs.getString( "CourseName" );
/*
                        course_id = rs.getString( 1 );
                        dept_id = rs.getString( 2 );
                        course_no = rs.getString( 3 );
                        course_lvl = rs.getString( 4 );
                        course_name = rs.getString( 5 );
*/
```

Sample Code -1 (contd)

```
                                        System.out.println( course_id + "..." + dept_id + "..."
+ course_no + "..." + course_lvl + "..." + course_name );
                              }

                              rs.close();
                              stmt.close();
                              con.close();

                    }catch( Exception e ){      System.out.println( e );      }
          }
}
```

List of Drivers

- **Bridge driver**
  - sun.jdbc.odbc.JdbcOdbcDriver
    - jdbc:odbc:<dsn>
- **Cloudscape**
  - **COM.cloudscape.core.JDBCDriver**
    - jdbc:cloudscape:[database name and location]
- **PostGRESQL www.StudentRockStars.com**
  - org.postgresql.Driver
    - jdbc:postgresql://[host]:[port]/[database name]
- MySQL
  - com.mysql.jdbc.Driver
    - jdbc:mysql://[host]:3306/[databasename]
- Oracle
  - oracle.jdbc.driver.OracleDriver
    - jdbc:oracle:thin:@[host]:1521:[sid]

Using java.sql API

Class DriverManager

•**Allows setting up of connections to databases and access to information about drivers.**

- **public Connection getConnection(String url)** → Protocol:subprotocol:name
- **public Connection getConnection(String url, String name, String password)**
- **public Connection getConnection(String url, Properties args)**
- **public Driver getDriver(String url)**
- **public Enumeration getDrivers()**
- **public void registerDriver(Driver)**
- **public void deregisterDriver(Driver)**
- **public void setLoginTimeout(int)**
- **public int getLoginTimeout()**
- **public void setLogStream(PrintStream)**
- **public void PrintStream getLogStream()**
- **public void println(String)**

Interface Driver

- This interface is implemented by JDBC Drivers.

  - public Connection connect(String, Properties)
  - public boolean acceptsURL(String)
  - public int getMajorVersion()
  - public int getMinorVersion()
  - public boolean jdbcCompliant()
  - public DriverPropertyInfo getPropertyInfo(String, Properties) []

  - www.StudentRockStars.com

Class DriverPropertyInfo

•Provides information about a JDBC driver
•It defines following five instance variables to describe driver properties:

```
public String name
public String description
public String value
public String choices[ ]
public boolean required
```

Sample Code - 2

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

class DriverApp
{
        public static void main (String args[])
        {
                try {
                        Class.forName ("ids.sql.IDSDriver");
                        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                }catch( Exception e ){     System.out.println( e );          }

                try{
                        Enumeration drivers = DriverManager.getDrivers();
                        System.out.println( "Following Drivers were found to be
loaded\n" );

                        Driver driver;
```

Sample Code -2 (contd)

```
        while( drivers.hasMoreElements() )
        {
                driver = (Driver) drivers.nextElement();

                System.out.println( "Driver : " + driver.getClass().getName() );
                System.out.println( "Major Version : " + driver.getMajorVersion() );
                System.out.println( "Minor Version : " + driver.getMinorVersion() );
                System.out.println( "Is JDBC complaint : " + driver.jdbcCompliant() );

                System.out.println( "\n" );
        }
        }catch(Exception se ){       System.out.println( se );            }
        }
}
```

Interface Connection

•An object that implements the interface Connection encapsulates a connection to database.

public Statement	createStatement()
public PreparedStatement	prepareStatement(String)
public CallableStatement	prepareCall(String)
public String	nativeSQL(String)
public void	close()
public boolean	isClosed()
public DatabaseMetaData	getMetaData()
public SQLWarning	getWarnings()
public void	clearWarnings()
public void	setAutoCommit(boolean)
public boolean	getAutoCommit()
public void	commit()
public void	rollback()
public Savepoint	setSavepoint()
public void	rollback(Savepoint)
public void	releaseSavepoint(Savepoint)

Interface DatabaseMetaData

•Gives information about structure and capabilities of a database.

```
public boolean allProceduresAreCallable()
public boolean allTablesAreSelectable()
public java.lang.String getURL()
public java.lang.String getUserName()
public boolean isReadOnly()
public java.lang.String getDatabaseProductName()
public java.lang.String getDatabaseProductVersion()
public java.lang.String getDriverName()
public java.lang.String getDriverVersion()
public int getDriverMajorVersion();
public int getDriverMinorVersion();
public boolean usesLocalFiles()
public boolean supportsMixedCaseIdentifiers()
```

Interface DatabaseMetaData…

public boolean storesUpperCaseIdentifiers()
public boolean storesLowerCaseIdentifiers()
public boolean storesMixedCaseIdentifiers()
public String getSQLKeywords()
public boolean supportsGroupBy()
public boolean supportsMultipleResultSets()
public boolean supportsMultipleTransactions()
public boolean supportsNonNullableColumns()

Sample Code -3

```
import java.sql.*;

public class DatabaseMetaData_prg
{
		public static void main( String args[] )
		{
				try{
						Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
				}catch( Exception e ){		System.out.println( e );	}
				try{
						Connection con = DriverManager.getConnection(
"jdbc:odbc:test" );

						DatabaseMetaData meta_data = con.getMetaData();

						System.out.println( "Database : " +
meta_data.getDatabaseProductName() );
						System.out.println( "Version : " +
meta_data.getDatabaseProductVersion() );
```

Sample Code -3 (Contd)

```
                              System.out.println( "User name : " +
meta_data.getUserName() );
                              System.out.println( "URL : " + meta_data.getURL() );
                              System.out.println( "Driver name : " +
meta_data.getDriverName() );
                              System.out.println( "Driver version : " +
meta_data.getDriverVersion() );
                              System.out.println( "Driver Major version : " +
meta_data.getDriverMajorVersion() );
                              System.out.println( "Driver Minor version : " +
meta_data.getDriverMinorVersion() );
                              System.out.println( "Uses local files : " +
meta_data.usesLocalFiles() );
                              System.out.println( "Keyword list : " +
meta_data.getSQLKeywords() );
                              con.close();
                    }catch( Exception e ){      System.out.println( e );      }
          }
}
```

Interface ResultSet

- It encapsulates the records retrieved from a query statement

public java.lang.String	getString(int)
public boolean	getBoolean(int)
public byte	getByte(int)
public short	getShort(int)
public int	getInt(int)
public long	getLong(int)
public float	getFloat(int)
public double	getDouble(int)
public java.sql.Date	getDate(int)
public java.sql.Time	getTime(int)

Interface ResultSet…

public String	getString(java.lang.String)
public boolean	getBoolean(java.lang.String)
public byte	getByte(java.lang.String)
public short	getShort(java.lang.String)
public int	getInt(java.lang.String)
public long	getLong(java.lang.String)
public float	getFloat(java.lang.String)
public double	getDouble(java.lang.String)
public java.sql.Date	getDate(java.lang.String)
public java.sql.Time	getTime(java.lang.String)
public boolean	next()
public void	close()
public java.sql.SQLWarning	getWarnings()
public void	clearWarnings()
public java.sql.ResultSetMetaData	getMetaData()

Interface ResultSetMetaData

- It provides information about contents of a ResultSet.

public java.lang.String	getTableName(int)
public boolean	isReadOnly(int)
public boolean	isWritable(int)
public int	getColumnCount()
public int	getColumnDisplaySize(int)
public java.lang.String	getColumnName(int)
public int	getColumnType(int)
public boolean	isCurrency(int)
public int	isNullable(int)
public boolean	isSigned(int)

Sample Code -4

```
import java.sql.*; www.StudentRockStars.com

public class ResultSetMetaData_prg
{
		public static void main( String args[] )
		{
				try{
						Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
				}catch( Exception e ){		System.out.println( e );	}

				try{
						Connection con = DriverManager.getConnection(
"jdbc:odbc:test" );
						Statement stmt = con.createStatement();

						ResultSet rs = stmt.executeQuery( "select * from courses" );

						ResultSetMetaData rsmd = rs.getMetaData();
```

www.StudentRockStars.com

Sample Code -4 (Cond)

```
System.out.println( "Table name : " + rsmd.getTableName( 1 ) );
System.out.println( "Read Only : " + rsmd.isReadOnly( 1 ) );
System.out.println( "Writable : " + rsmd.isWritable( 1 ) );
System.out.println( "No. of Columns : " + rsmd.getColumnCount() + "\n\n" );
for( int i =1; i <= rsmd.getColumnCount();  i++ )
{
        if( i < rsmd.getColumnCount() )
                System.out.print( rsmd.getColumnName( i ) + "  |  " );
        else
                System.out.println( rsmd.getColumnName( i )  + "\n\n" );
}
```

Sample Code -4 (Cond)

```
                    while( rs.next() )
                    {
                          for( int i =1; i <= rsmd.getColumnCount();  i++ )
                          {
                                if( i < rsmd.getColumnCount() )
                                      System.out.print( rs.getString( i )
+ "  |  " );
                                else
                                      System.out.println( rs.getString(
i ) );
                          }
                    }

                    con.close();

              }catch( Exception e ){      System.out.println( e );     }
       }
} www.StudentRockStars.com
```

Interface Statement

- **Provides a way to perform SQL queries**

public boolean	execute(java.lang.String)
public java.sql.ResultSet	executeQuery(java.lang.String)
public int	executeUpdate(java.lang.String)
public void	close()
public int	getMaxFieldSize()
public void	setMaxFieldSize(int)
public int	getMaxRows()
public void	setMaxRows(int)
public int	getQueryTimeout()
public void	setQueryTimeout(int)
public java.sql.SQLWarning	getWarnings()
public void	clearWarnings()

www.StudentRockStars.com

Interface PreparedStatement

- **Provides a way to use precompiled SQL statements**

public java.sql.ResultSet	executeQuery()
public boolean	execute()
public int	executeUpdate()
public void	setNull(int, int)
public void	setBoolean(int, boolean)
public void	setByte(int, byte)
public void	setShort(int, short)
public void	setInt(int, int)
public void	setLong(int, long)
public void	setFloat(int, float)
public void	setDouble(int, double)
public void	setString(int, java.lang.String)
public void	setDate(int, java.sql.Date)
public void	setTime(int, java.sql.Time)
public void	clearParameters()
public java.sql.ResultSetMetaData	getMetaData()
public java.sql.ParameterMetaData	getParameterMetaData()

Sample Code - 5

```
import java.sql.*;

public class preparestmt
{
		PreparedStatement prepare;

		public preparestmt()
		{
				try
				{
						Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
						Connection
conn=DriverManager.getConnection("jdbc:odbc:test");

						String sql="select CourseName from courses where
Department_ID=?";

						prepare=conn.prepareStatement( sql );
```

Sample Code – 5 (Contd)

```
prepare.clearParameters();
prepare.setString(1,"BIOL" );

ResultSet rs=prepare.executeQuery();

while(rs.next())
            System.out.println( rs.getString(1) );

System.out.println( "-------------------------------------------------" );

prepare.clearParameters();
prepare.setString(1,"MATH" );

rs=prepare.executeQuery();

while(rs.next())
            System.out.println( rs.getString(1) );
```

Sample Code – 5 (Contd)

```
                    }catch(SQLException e){   System.out.println( e );      }
                    catch(Exception e )          {          System.out.println( e );      }
          }

          public static void main(String p[])
          {
                    new preparestmt();
          }
}
```

Interface CallableStatement

•Provides a way to use stored procedures.

•Steps for using CallableStatement

•Invoke Connection.prepareCall() to create CallableStatement.
•Invoke CallableStatement.setXXX methods to pass values to the input (IN) parameters.
•Invoke the CallableStatement.registerOutParameter method to indicate which parameters are output-only (OUT) parameters
•Invoke CallableStatement.executeUpdate() to call the stored procedure.
•If the result sets are received, retrieve them.
•Invoke the CallableStatement.getXXX methods to retrieve values from the OUT parameters

Code snippet

•The following code illustrates calling a stored procedure that has two input parameters, one output parameter, and no returned ResultSets.

```
CallableStatement cstmt;

cstmt = ConnectionObj.prepareCall( “{call insert_test_tableName(?,?,?)}” );
cstmt.setInt(1, 100);
cstmt.setString(2, "ABCD");
cstmt.registerOutParameter(3, java.sql.Types.INTEGER);

cstmt.execute();

int id = cstmt.getInt(3); www.StudentRockStars.com
```

Transaction Processing

Use the following three methods of Connection interface to handle transaction Processing:

public void	setAutoCommit(boolean)
public boolean	getAutoCommit()
public void	commit()
public void	rollback()
public Savepoint	setSavepoint()
public void	rollback(Savepoint)
public void	releaseSavepoint(Savepoint)

Working with CloudScape

- 1. Download the following
  cloudview406.jar and
  jh.jar
from IBM's web site.
(http://www-1.ibm.com/support/docview.wss?rs=636&context
=SSCRVP&q=&uid=swg24001378&loc=en_US&cs=utf-8&lang=en+en)

- Set the classpath to cloudscape.jar, cloudutil.jar, cloudview406.jar and jh.jar.

Working with CloudScape

- Open Command window and the enter the following command :
-     java COM.cloudscape.tools.cview
- 5. Select file->new->database.
- 6. Give a database name to be stored.
- 7. use the respective options to create tables, to create fields and add table data.

Sample Code

```
import java.sql.*;

class test www.StudentRockStars.com
{
    public static void main( String a[])
    {
        try{
                Class.forName( "COM.cloudscape.core.JDBCDriver" );

                Connection con = DriverManager.getConnection(
    "jdbc:cloudscape:d:/database/phonebook" );

                Statement stmt =  con.createStatement();
                ResultSet rs = stmt.executeQuery( "select * from NAMES" );

                rs.next();
                System.out.println( rs.getString( 1 ) + "..." + rs.getString( 2 ) );
        }catch( Exception e){         System.out.println( e );      }
    }
}
```