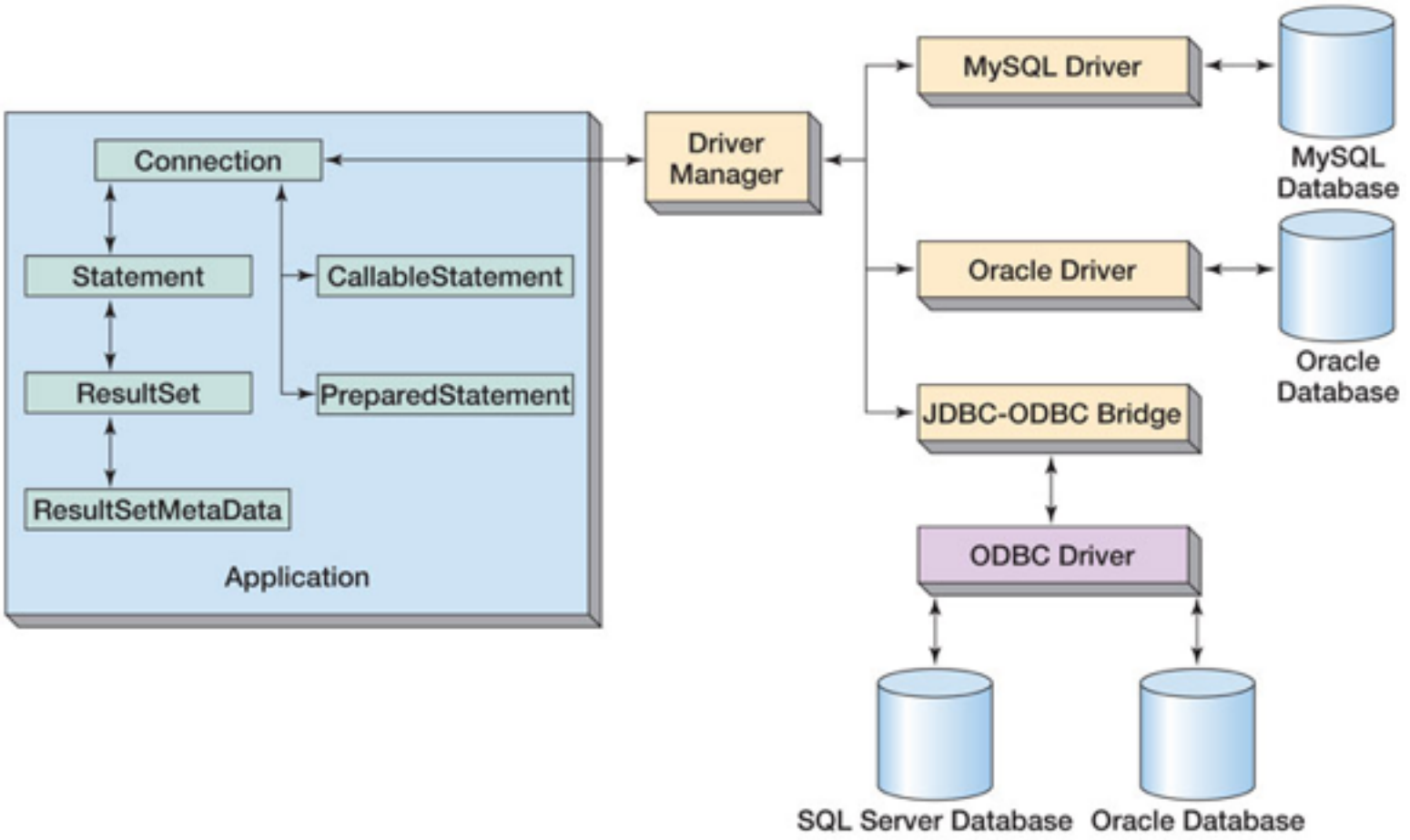
JDBC

- **JDBC** is an alternative to ODBC and ADO that provides database access to programs written in Java.
- JDBC drivers are available for most DBMS products:
  - http://java.sun.com/products/jdbc

JDBC Driver Types

Driver Type	Characteristics
1	JDBC-ODBC bridge. Provides a Java API that interfaces to an ODBC driver. Enables processing of ODBC data sources from Java.
2	A Java API that connects to the native-library of a DBMS product. The Java program and the DBMS must reside on the same machine, or the DBMS must handle the intermachine communication, if not.
3	A Java API that connects to a DBMS-independent network protocol. Can be used for servlets and applets.
4	A Java API that connects to a DBMS-dependent network protocol. Can be used for servlets and applets.

JDBC Components



Database Programming Steps

1. Establish a connection
2. Begin transaction
3. Create a statement object
4. Associate SQL with the statement object
5. Provide values for statement parameters
6. Execute the statement object
7. Process the results
8. End transaction
9. Release resources

Using JDBC

1a. Load the driver:

- The driver class libraries need to be in the CLASSPATH for the Java compiler and for the Java virtual machine.
- The most reliable way to load the driver into the program is:

```
Class.forName(string).newInstance();
```

1b. Establish a connection to the database:

- A connection URL string includes the literal jdbc:, followed by the name of the driver and a URL to the database

```
String url = "jdbc:oracle:thin:@localhost:1521:csodb";
```

jdbc "subprotocol""subname" host port database

- Create a Connection object:

```
Connection con = DriverManager.getConnection(url,
                                    dbUser, dbPassword);
```

Using JDBC (Continued)

2. Create a statement object

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

3. Associate SQL with the statement object

insert into students (name,address) values ('Ganesh','1 ABC Street ABC Bangalore');

```
String queryString = "create table students "
                   + "(name varchar(30), id int, phone char(9))";
```

4. Process the statement:

Example statements:

```
ResultSet          rs     = stmt.executeQuery(querystring);
int                result = stmt.executeUpdate(updatestring);
ResultSetMetaData  rsMeta = rs.getMetaData();
```

- Compiled queries can be processed via a **PreparedStatement** object
- Stored procedures can be processed via a **CallableStatement** object

5. Release connection

con.close()

Using a PreparedStatement

```
PreparedStatement prepStmt = con.prepareStatement(
            "INSERT INTO Artist (ArtistID, Name, "
            + "Nationality, BirthDate, DeceasedDate)"
            + "VALUES (ArtistSeq.nextVal, ?, ?, ?, ? )" );

// Now supply values for the parameters
// Parameters are referenced in order starting with 1.
prepStmt.setString( 1, "Galvan" );
prepStmt.setString( 2, "French" );
prepStmt.setInt   ( 3, 1910 );
prepStmt.setNull  ( 4, Types.INTEGER );

prepStmt.executeUpdate();
System.out.println( "Prepared statement executed" );

// Now do it again
prepStmt.setString( 1, "Monet" );
prepStmt.setString( 2, "French" );
prepStmt.setInt   ( 3, 1840 );
prepStmt.setInt   ( 4, 1879 );

prepStmt.executeUpdate();
System.out.println( "Prepared statement executed again" );
```

Create Table

```
CREATE TABLE students
(
id INTEGER NOT NULL GENERATED
ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
name VARCHAR(24) NOT NULL,
address VARCHAR(1024),
CONSTRAINT primary_key PRIMARY KEY (id)
) ;

CREATE TABLE users
(
id INTEGER NOT NULL GENERATED
ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
name varchar(100),
email_id varchar(100) ,
password varchar(100) ,
join_date timestamp ,
age int  ,
```

Transactions And Isolation

1. Transactions are started with con.setAutoCommit(false)
2. Four levels of isolation is possible

Isolation Level	Transactions	Dirty Reads	Non-Repeatable Reads	Phantom Reads
TRANSACTION_NONE	Not supported	*Not applicable*	*Not applicable*	*Not applicable*
TRANSACTION_READ_COMMITTED	Supported	Prevented	Allowed	Allowed
TRANSACTION_READ_UNCOMMITTED	Supported	Allowed	Allowed	Allowed
TRANSACTION_REPEATABLE_READ	Supported	Prevented	Prevented	Allowed
TRANSACTION_SERIALIZABLE	Supported	Prevented	Prevented	Prevented

Transactions Test

- Get two connections to the database:
- setAutoCommit(false) on both connections
- con.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);
- “select * from users” on con1
- insert a user using con2
- update an existing user in con2
- con1.commit(); con2.commit();