# CIS 3145 Class Notes: Text Chapter 20

## Java Database Connectivity (JDBC)

**Objectives**

* Describe Java database drivers and how to add them to a project
* Describe how to connect to a database in Java
* Create statement objects and prepared statement objects
* Use a result set to read and update data from a database

Statement Object

Connection Object

Java App

Driver

Database

Result Set

**Overview**

Multiple objects are needed to connect a program to an object. The **driver** class manages the communication from for a particular database to a particular programming language. In Java the class that holds a **driver** for a specific database is located in the project library.

If a database is a store, then the **connection** object is the instructions for how to get to the physical location of the store and any identification needed to get into the store. Costco requires a shopper have a membership while Safeway does not. The **connection** object holds the path to the database, a user name and password if needed.

The **statement** object is like the grocery list you take to the store. It holds the SQL statements used to read data from the database. Unlike a grocery list the SQL **statement** object can also create, delete, or update data in the database. Executing a query method on the **statement** object is the process of going to the store and getting the items on the list.

The **result set** object is the same as the grocery bag that holds the items purchased at the store. The **result set** holds the data from the database and must be processed to get the data into the program. When there is more than one row of data, loops are used to take the data from the result set and then put into arrays or collections.

**Java Database Drivers**

A java program will use the JDBC driver manager to connect to one of 4 types of drives. The driver program will then connect to the DBMS.

**Type 1**: a JDBC- ODBC (Open Database Connectivity) driver will connect to an ODBC Driver. The ODBC driver must be on the client machine. ODBC drives are commonly available and unique to each operating system. Registering the ODBC driver on the client machine is time consuming. Each client machine must be worked on.

**Type 2**: a Native Protocol Partly Java Driver will connect directly to the DBMS. The client machine must have this driver registered. Like type 1 this can be time consuming.

**Type 3**: a Net Protocol all Java Driver converts the JDBC calls into a ‘Net Protocol’. The net protocol is translated by middleware on the database server into a native DBMS protocol. The middle ware is flexible enough to work with different Database Management Systems and no client code is required.

**Type 4**: a Native protocol all Java driver will connect directly with a DBMS. No client code is needed but the driver must change if the DBMS changes.

**Add a database driver to a project**

To connect to the MySQL database in a Java project a JAR file must be added to the library for the project. The MySQL jar file is “mysql-connector-java-5.1.23-bin.jar”.

Add the database **‘jar’** file to the project library (for example derby.jar).

**Connection Object**

A connection object is created with the static getConnection method of the Driver Manager class. There are three getConnection methods and each one requires the location of the database as a URL string. If a password is required than the user name and password are the second and third arguments for calling the getConnection method. Because the getConnection method can throw an [SQLException](https://docs.oracle.com/javase/8/docs/api/java/sql/SQLException.html) or [SQLTimeoutException](https://docs.oracle.com/javase/8/docs/api/java/sql/SQLTimeoutException.html) the method must be inside a try…catch block.

**Statement Object**

A **statement** object is used to retrieve a **result Set**. A **connection** object is used to create a **statement** object with the createStatement method.

***Statement vs. Prepared Statements***

The **prepared statement** is used when we want to use a variable value for a field. The SQL statement is written with a question mark which acts as a place holder for the value that will be supplied dynamically while the program is running. The prepared statement is also faster because it can be cached on the database, and it is more secure.

“SELECT Code, Description, ListPrice FROM Product WHERE Code = ?”

**Result sets**

The **result** **set** object is a local version of a table from the database. It has the rows and columns. The result set must then be processed to put the data into local variables that are used to process and display the data in the application.

As we move through the rows of a result set we talk about a **cursor** that points to the current row. The **next**() and **previous**() methods move the cursor of the result set one row at a time forward or backwards respectively. The **first**() and **last**() methods move the cursor to the first or last row.

Some result sets are **read only** and others can be **updated**. Result sets can also be **Forward Only**, meaning that we move from the beginning to the end of the program, or the can be **Scrollable**, meaning we can move back and forth through the result set. This is similar to *sequential access* and *random access* file processing. The **Forward Only** is the same as **sequential** access, while **Scrollable** is equivalent to **random access**.

**Database errors**:murach.db.DBException: java.sql.SQLException:

Watch for SQL exception errors. This can be caused if an SQL script has not been run, or there is an error in the SQL statement. Use the MySQL Workbench to trouble shoot the SQL statement.

Because the SQL statement objects have the potential to create an error, all methods that use these objects must either **catch** or **throw** an error. If an error is thrown by the method with the SQL statement object then the code that **calls** this method, typically code in the main method, must catch the error and report the problem to the user. Eventually errors must be caught at some level of the code. The sooner the error is caught the better.