Working with Databases and Java

Pedro Contreras

Department of Computer Science
Royal Holloway, University of London

January 30, 2008

Outline

- Introduction to relational databases
- Introduction to Structured Query Language (SQL)
- Entity Relationship modelling
- Working with databases and Java
- References
- Questions

What is a database?

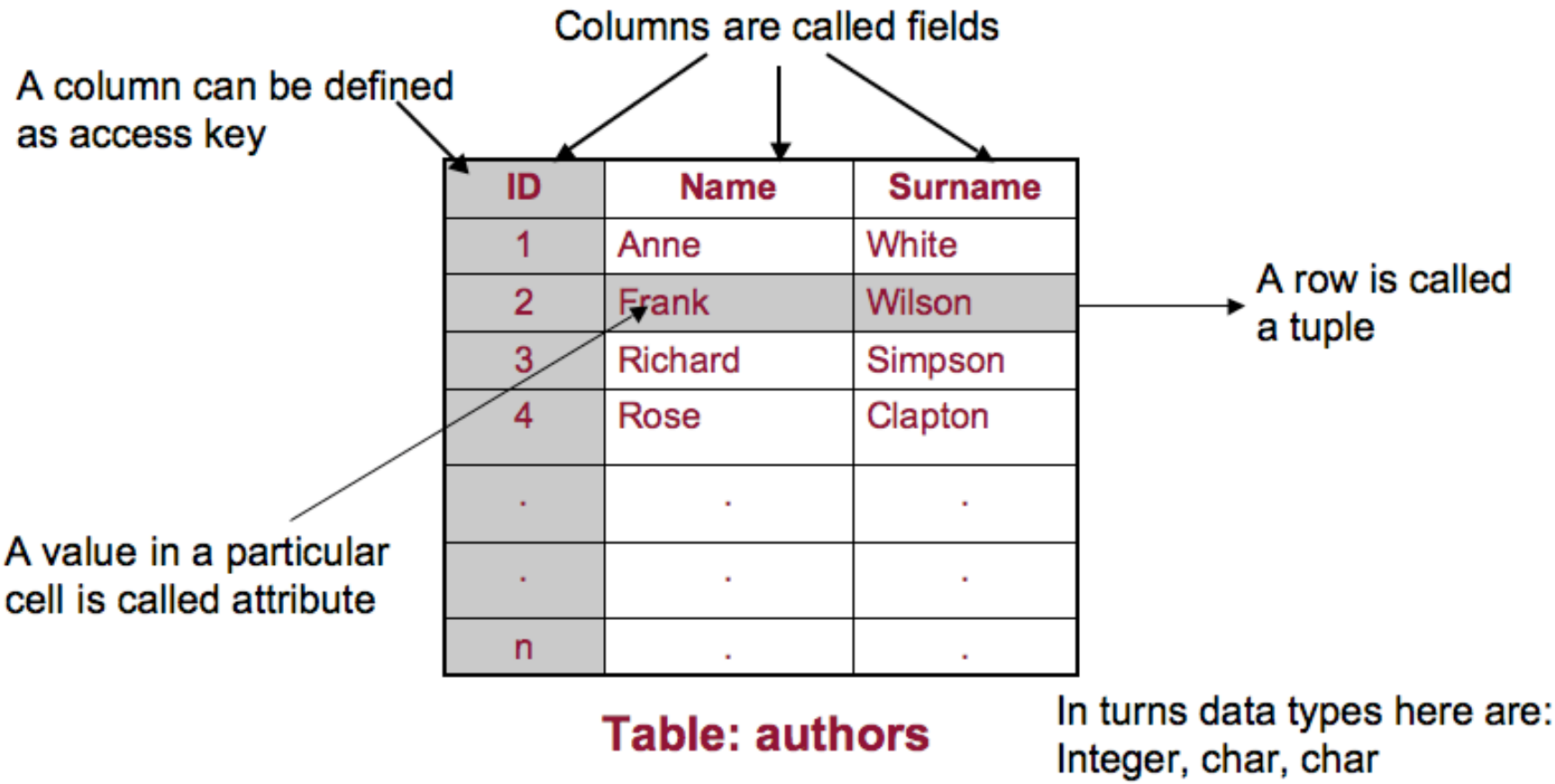
A database is an organised collection of data.

The computer program used to administrate and manage this data is called a database management system (DBMS), e.g. MySQL, Oracle DB, DB2, MS Access, Ingres, Sybase, etc.

How the data is stored?

- Relational databases stores data in tables, where columns are called fields, and rows are called tuples.
- Columns are defined in conjunction with their data type, e.g. char, varchar, integer, float, double, etc
- Also a column can be a key to access information in a table.

Database table example



Structured Query Language (SQL)

This is the most popular computer language used to
Create, Modify, Delete and Retrieve
data from a relational database management system

SQL transactions

Most frequent SQL data transaction are:

- **Insert**, used to insert rows into a table
- **Delete**, used to delete rows in a table
- **Update**, used to modify values in a existing table
- **Select**, used to retrieve data
  From, used to indicate from which tables the data will be taken
  Where, used to indicate rows to be retrieval
  Group by, used to combine rows
  Order by, used to indicate columns to sort result

SQL Example

```
1  $ INSERT INTO authors (ID, FirstName, LastName)
2     VALUES(5, ''John'', ''Sheen'')
3
4  $ UPDATE authors SET FirstName = ''Martin'' WHERE ID = 5
5
6  $ DELETE FROM authors WHERE ID = 5
7
8  $ SELECT FirstName, LastName FROM authors
```

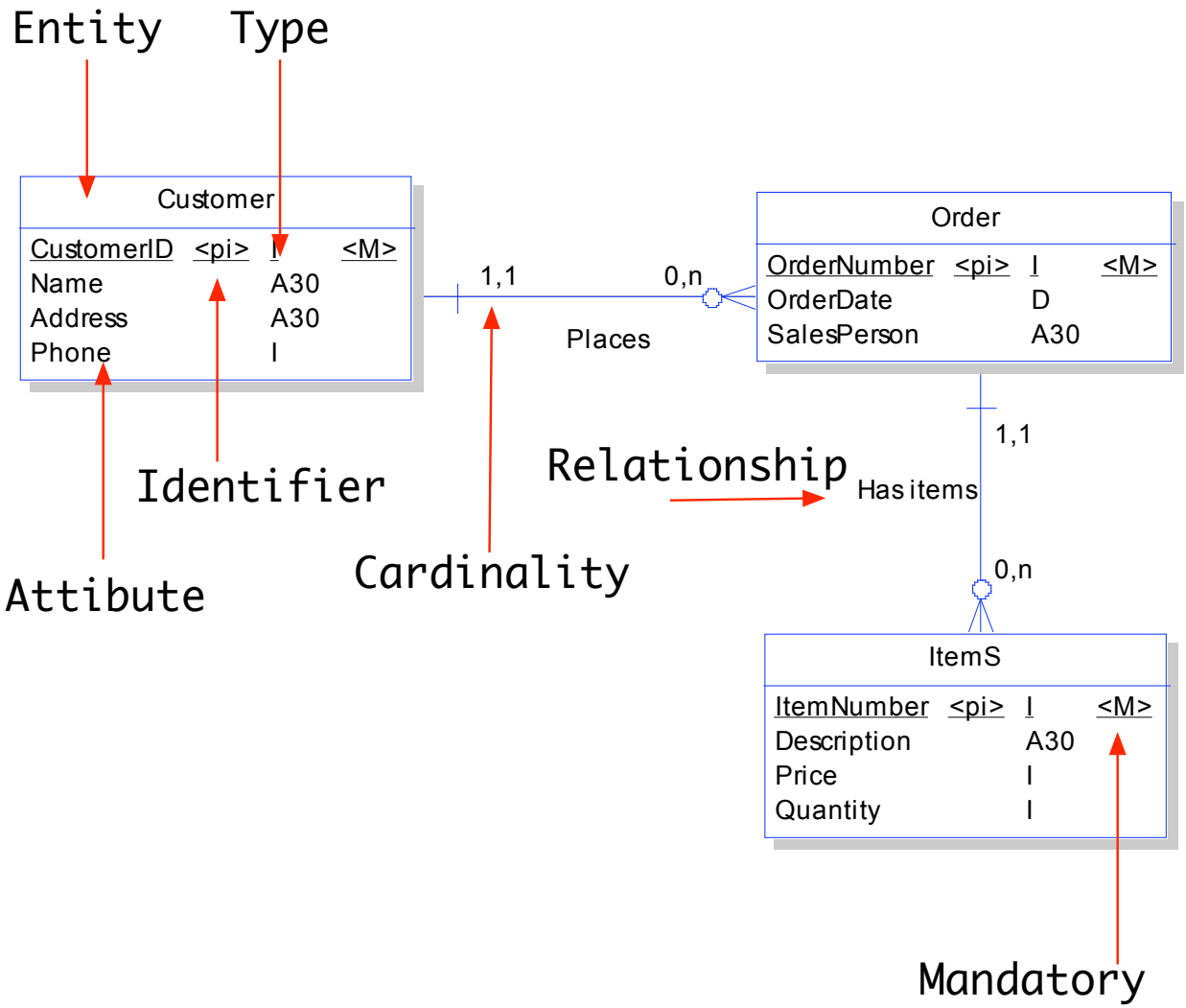
Entity–Relationship modelling

- This is a set of rules used to interpret and specify the logic behind a problem when designing databases.
- E-R model is a database Conceptual Model, which is practice has many variations, but in general uses representation of mainly 4 constructs
- An E-R model can not be implement directly on a database, but from this a Physical model can be derived

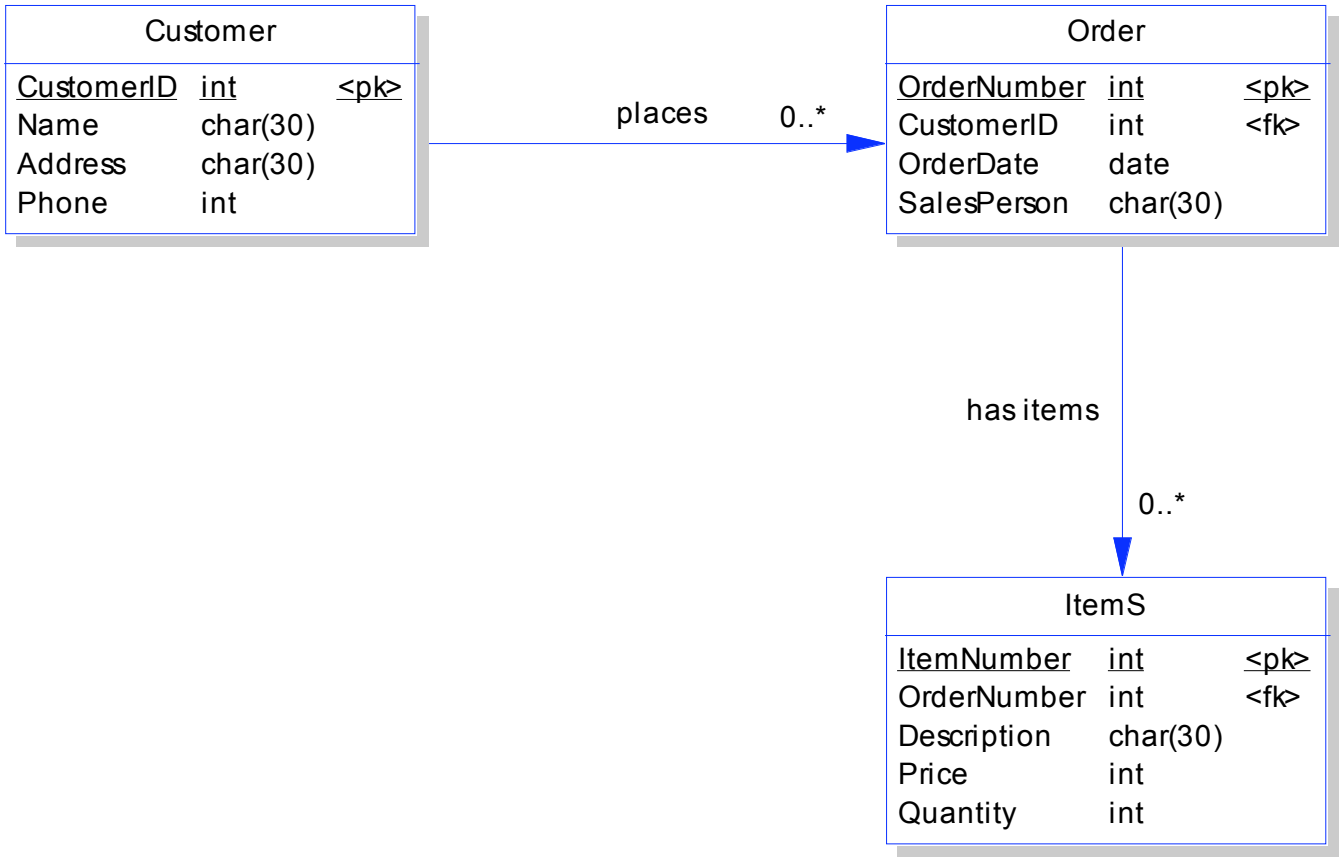
E–R modelling constructs

- Entity: a relevant object to be modelled, e.g. customers, products, employee
- Attributes: a characteristic of an entity, e.g. attributes from a customers could be: name, surname, and address
- Identifiers: a special attribute used to identify a specific instance of an entity, e.g.
  Identifier of a **customer** could be an
  textbfcustomer ID
  Identifier of a **employee** could be the **employee code**
- Relationship: association between two entities, e.g.
  A **customer** places a **customer order**
  A **student** enrols in a **course**
  A **course** is taught by a **faculty member**

E–R conceptual diagram example



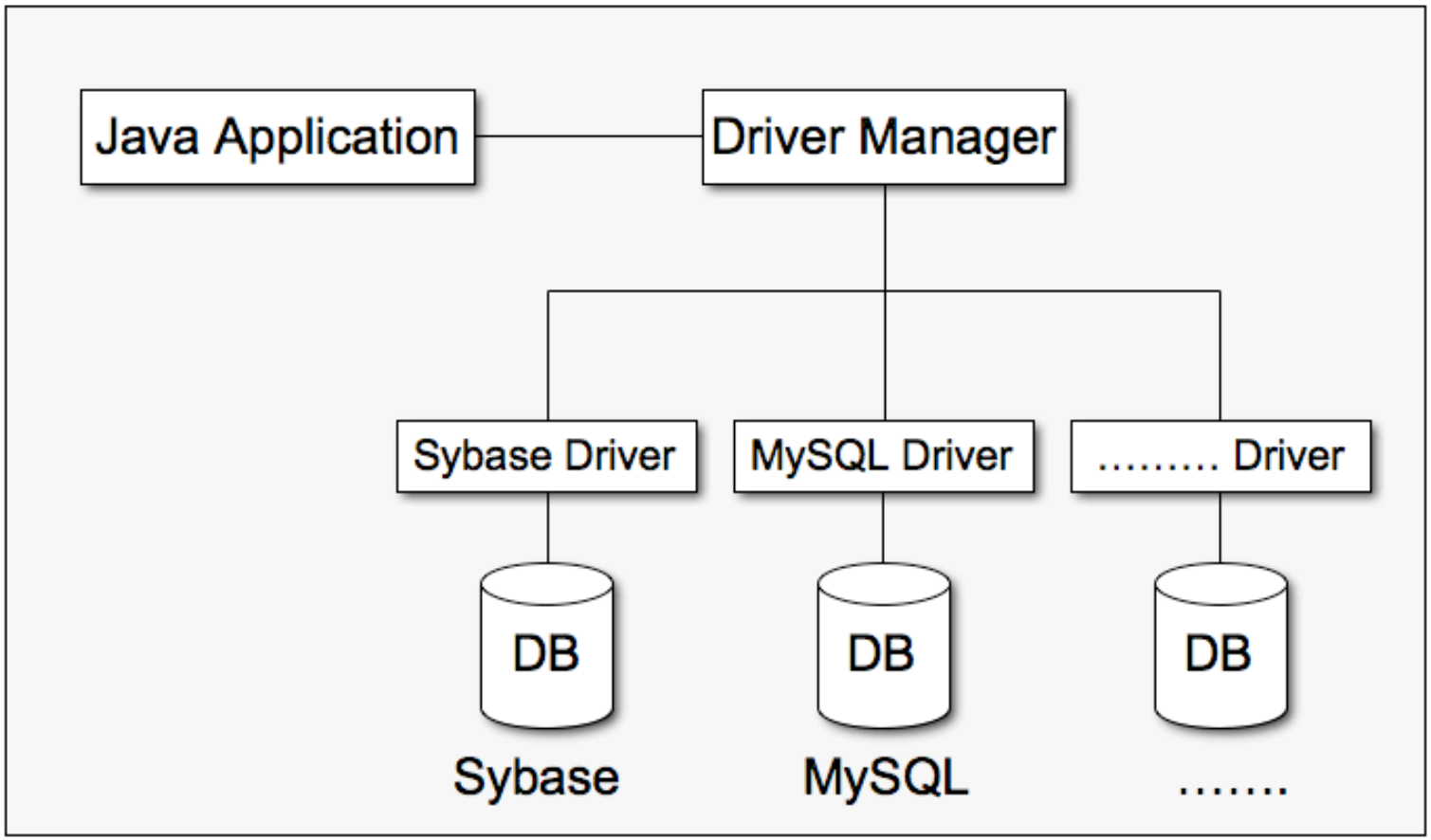
E-R physical diagram example



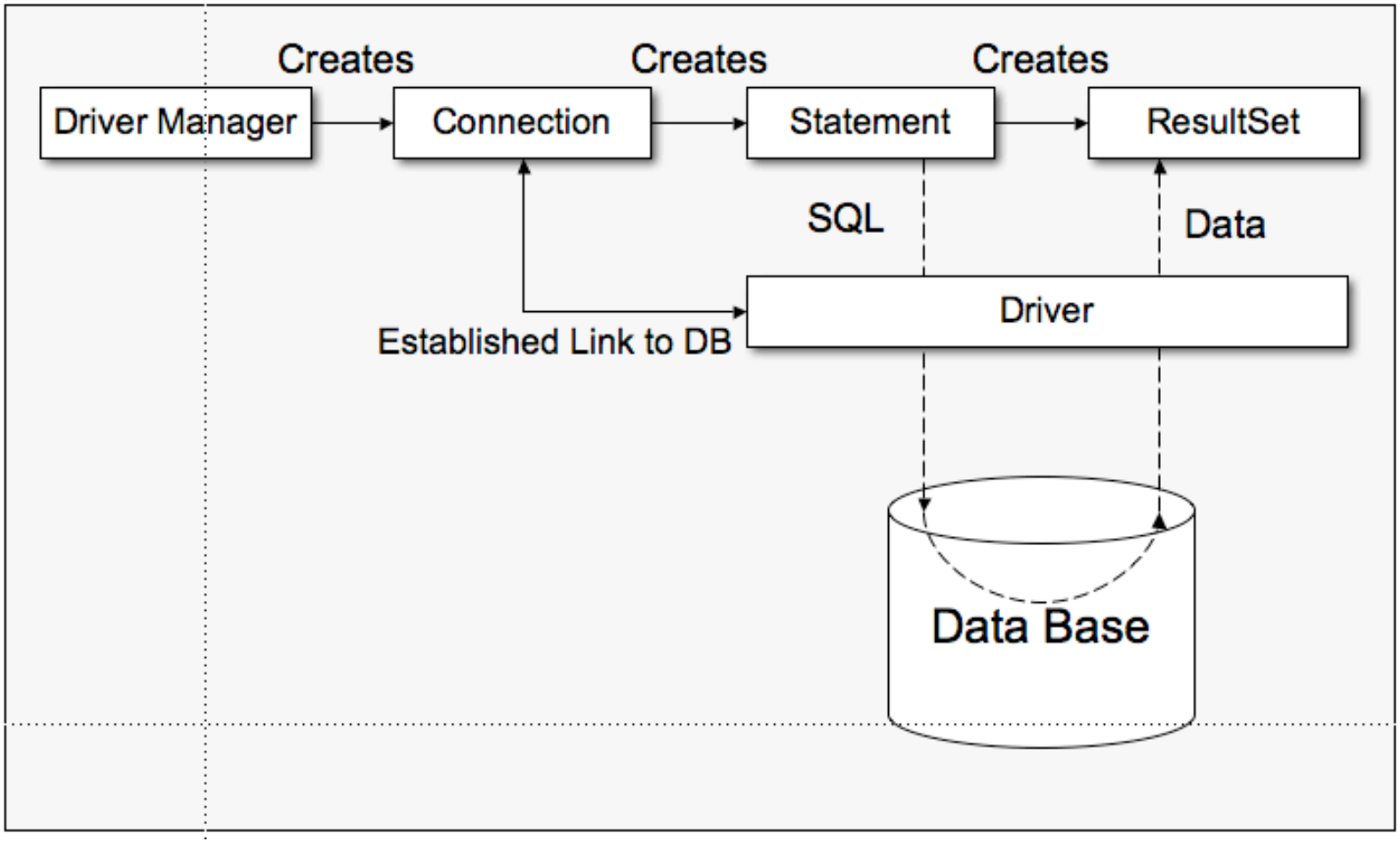
Java (JDBC-ODBC) and databases

- Java database technology relies on JDBC API(Java Database Connectivity) libraries
- JDBC allows to connect to any database that supports ODBC (Open Database Connectivity)
- JDBC architecture is based on a collection of Java interfaces and classes that enables us to connect to data sources, to create, and to execute SQL statements

Java and databases representation



Java and databases in detail



DB transactions with Java

- Create Database
- Create table
- Insert data
- Update data
- Delete data
- Select data

Creating a database with Java

```
1  import java.sql.*;
2
3  public class CreateDatabase {
4  public Create (String url, String user, String password) throws Exception {
5
6  String dbname = ``Customer'';
7
8  try {
9     // Register the JDBC driver for MySQL.
10    Class.forName(``com.mysql.jdbc.Driver'');
11    // Get a connection to MySQL database
12    Connection con = DriverManager.getConnection(url, user, password);
13    Statement stmt = con.createStatement();              // Create statement
14
15    stmt.executeUpdate(``DROP DATABASE '' + dbname);     // Drop db if exist
16    stmt.executeUpdate(``CREATE DATABASE '' + dbname);   // Create a new db
17
18    stmt.close();      // Close statement
19    con.close();       // Close connection
20    }
21  catch (Exception e) {
22     System.out.print(e);
23  }
24 }
25 }
```

Creating tables with Java

```
1   import java.sql.*;
2
3   public class Create {
4   public void Create (String url, String user, String password) throws Exception {
5
6   try {
7     // Register the JDBC driver for MySQL.
8      Class.forName(``com.mysql.jdbc.Driver'');
9      // Get a connection to MySQL database
10     Connection con = DriverManager.getConnection(url, user, password);
11     Statement stmt = con.createStatement();              // Create statement
12
13     // Create authors table
14     stmt.executeUpdate(``CREATE TABLE authors (ID int NOT NULL,'' +
15                        ``FirstName char(30) NOT NULL,'' +
16                        ``LatName char(30) NOT NULL)'');
17     stmt.close();      // Close statement
18     con.close();       // Close connection
19     }
20    catch (Exception e) {
21     System.out.print(e);
22    }
23  }
24  }
```

Inserting data with Java

```
1   import java.sql.*;
2
3   public class Insert {
4   public void Insert(String url, String user, String password) throws Exception {
5
6   try {
7
8      // Register the JDBC driver for MySQL.
9      Class.forName(``com.mysql.jdbc.Driver'');
10     // Get a connection to MySQL database
11     Connection con = DriverManager.getConnection(url, user, password);
12     Statement stmt = con.createStatement();              // Create statement
13
14     // Insert data into a table
15     stmt.executeUpdate(
16      ``INSERT INTO authors (ID, FirstName, LastName) VALUES(5, `John`, `Sheen`)'');
17     stmt.close();             // Close statement
18     con.close();              // Close connection
19   }
20   catch (Exception e) {
21     System.out.print(e);
22   }
23  }
24  }
```

Updating data with Java

```
1   import java.sql.*;
2
3   public class Update {
4   public void Update(String url, String user, String password) throws Exception {
5
6   try {
7
8      // Register the JDBC driver for MySQL.
9      Class.forName(``com.mysql.jdbc.Driver'');
10     // Get a connection to MySQL database
11     Connection con = DriverManager.getConnection(url, user, password);
12     Statement stmt = con.createStatement();              // Create statement
13
14     // Update data in a table
15     stmt.executeUpdate(``UPDATE authors SET FirstName = `Martin`  WHERE ID = 5'');
16     stmt.close();             // Close statement
17     con.close();              // Close connection
18     }
19
20    catch (Exception e) {
21      System.out.print(e);
22    }
23  }
24  }
```

Delete data with Java

```
1   import java.sql.*;
2
3   public class Delete {
4   public void Delete(String url, String user, String password) throws Exception {
5
6   try {
7
8      // Register the JDBC driver for MySQL.
9      Class.forName(``com.mysql.jdbc.Driver'');
10     // Get a connection to MySQL database
11     Connection con = DriverManager.getConnection(url, user, password);
12     Statement stmt = con.createStatement();              // Create statement
13
14     // Delete register in a table
15     stmt.executeUpdate(``DELETE FROM authors WHERE ID = 5'');
16     stmt.close();              // Close statement
17     con.close();               // Close connection
18     }
19
20    catch (Exception e) {
21      System.out.print(e);
22    }
23  }
24  }
```

Retrieving data with Java

```
1   import java.sql.*;
2
3   public class Select {
4   public void Select(String url, String user, String password) throws Exception {
5
6   try {
7
8      // Register the JDBC driver for MySQL.
9      Class.forName(``com.mysql.jdbc.Driver'');
10     // Get a connection to MySQL database
11     Connection con = DriverManager.getConnection(url, user, password);
12     Statement stmt = con.createStatement();              // Create statement
13
14     // Select FirstName and LastName from the table authors
15     ResultSet rs = stmt.executeQuery(
16     ``SELECT FirstName, LastName FROM authors'');  // SELECT * FROM authors
17     ...
18     // Print result set
19     con.close();              // Close connection
20     }
21
22    catch (Exception e) {
23       System.out.print(e);
24    }
25  }
26  }
```

Additional methods

The JDBC package also offers a series of additional methods that are very useful when working with databases, e.g.

- Retrieving warnings
- Retrieving exceptions
- Create objects to store results
- etc

References

- Sun Microsystems, `http://java.sun.com/javase/technologies/database`
- Ivor Horton, Beginning Java 2 JDK 1.3.0 Edition, Wrox Press, Chapter 19. `http://java.sun.com/developer/Books/javaprogramming/begjava2/ch19.pdf`
- George Reese, Database Programming with JDBC and Java, O'Reilly. pp 48-57