Working with Databases and Java

Department of Computer Science
Royal Holloway, University of London

Pedro Contreras
pedro @ cs.rhul.ac.uk

Introduction

- 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.

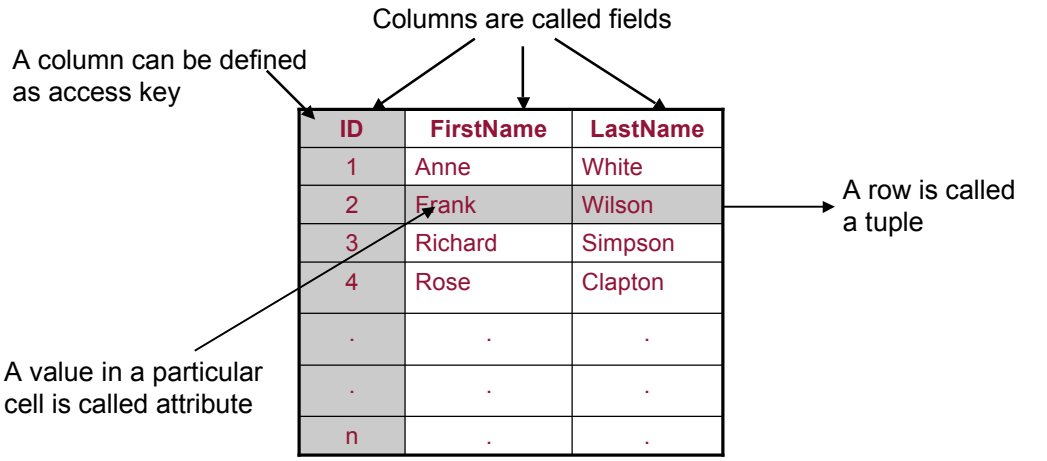
3

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.

4

Database table example



ID	FirstName	LastName
1	Anne	White
2	Frank	Wilson
3	Richard	Simpson
4	Rose	Clapton
.	.	.
.	.	.
n	.	.

Table: authors

In turns data types here are: Integer, char, char

5

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

6

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
- Data retrieval
  - **Select**
    - **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

7

SQL Example

Following the table presented before:

- INSERT INTO **authors** (ID, FirstName, LastName) VALUES**(5, 'John, 'Sheen'**)
- UPDATE **authors** SET **FirstName = 'Martin'** WHERE **ID = 5**
- DELETE FROM **authors** WHERE **ID = 5**
- SELECT **FirstName, LastName** FROM **authors**

8

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 *Conceptual Model* of the database, 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

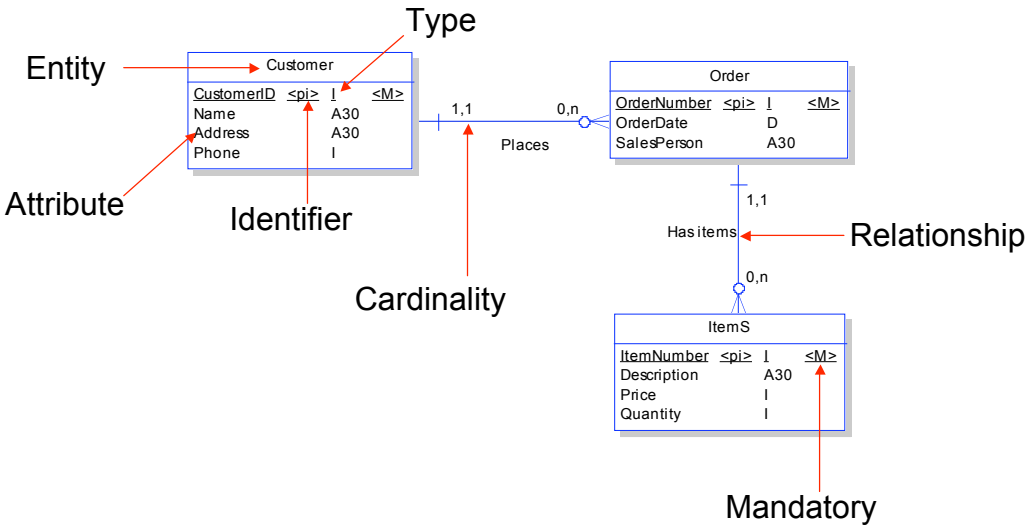
9

E-R Modelling Constructs

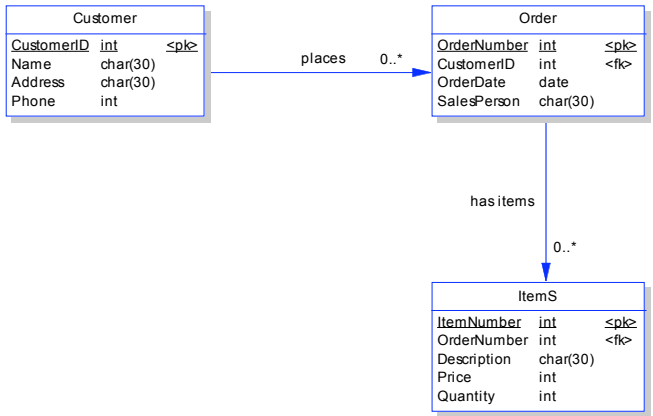
- **Entity**: a relevant object to be modelled, e.g. customers, products, employee
- **Attributes:** a characteristic of an entity, e.g. attributes from a ***customers*** could be: name, surname, and address
- **Identifiers:** a special attribute used to identify a specific instance of an entity, e.g.
  Identifier of a ***customer*** could be an ***customer 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***

10

E-R Conceptual Diagram



E-R Physical Diagram

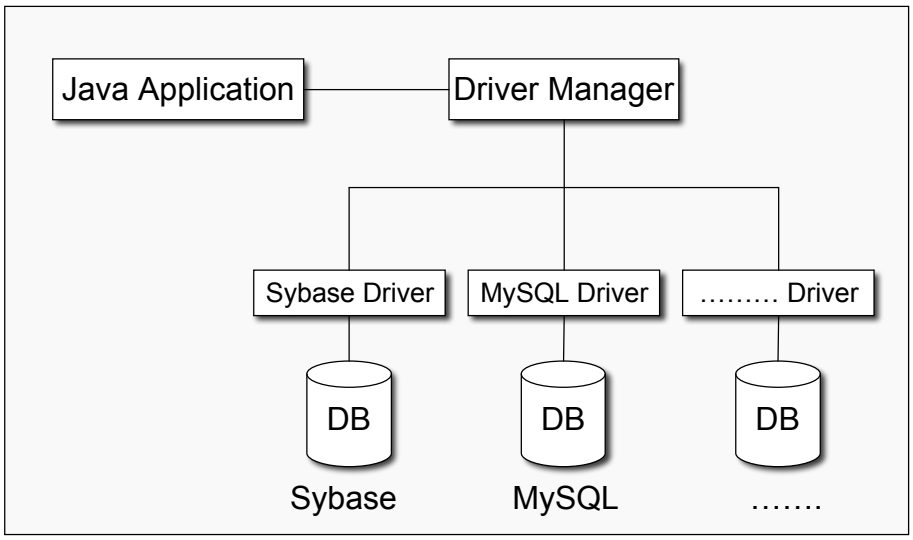


Java (JDBC) and Databases

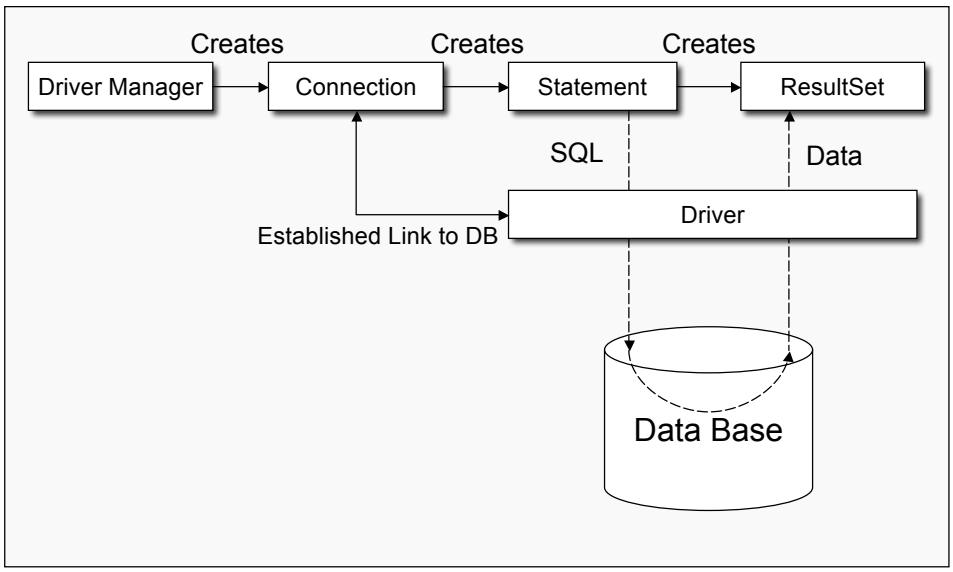
Java database technology relies on JDBC (Java Database Connectivity) libraries.

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



15

Java steps for working with DB

- Import the necessary classes
- Load the `JDBC` driver
- Identify the data source
- Allocate a `Connection` object
- Allocate a `Statement` object
- Execute a query using the `Statement` object
- Retrieve data form from the returned `ResultSet` object
- Close the `ResultSet`
- Close the `Statement` object
- Close the `Connection` object

16

DB transactions with Java

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

Creating tables with Java

```
import java.sql.*;
public class Create {
public Create (String url, String user, String password) throws Exception {

try {
   Statement stmt;                                    // Register the JDBC driver for MySQL.
   Class.forName("com.mysql.jdbc.Driver");           // Get a connection to MySQL database
   Connection con = DriverManager.getConnection(url, user, password);
   stmt = con.createStatement();                      // Create statement

 try {
   stmt.executeUpdate("DROP table if exists " + table);
   } catch (Exception e) {
   System.out.print(e);
   // Create authors
   stmt.executeUpdate(   "CREATE TABLE authors (ID int NOT NULL, FirstName char(30) NOT NULL," +
                         "LatName char(30) NOT NULL)");
   stmt.close();          // Close statement
   con.close();           // Close connection
   }
 catch (Exception e) {
 System.out.print(e);
 }
}
}
```

Inserting data with Java

```
import java.sql.*;

public class Insert {
public Insert(String url, String user, String password) throws Exception {

try {

  Statement stmt;                                  // Register the JDBC driver for MySQL.
  Class.forName("com.mysql.jdbc.Driver");          // Get a connection to the database
  Connection con = DriverManager.getConnection(url, user, password);          //Get a Statement object
  stmt = con.createStatement();                    // Create statement

  // Insert data into a table
  stmt.executeUpdate( "INSERT INTO authors (ID, FirstName, LastName) VALUES(5, 'John, 'Sheen')" );
  stmt.close();          // Close statement
  con.close();           // Close connection
 }
 catch (Exception e) {
 System.out.print(e);
 }
}
}
```


Updating data with Java

```
import java.sql.*;

public class Update {
public Update(String url, String user, String password) throws Exception {

try {

  Statement stmt;                                  // Register the JDBC driver for MySQL.
  Class.forName("com.mysql.jdbc.Driver");          // Get a connection to the database
  Connection con = DriverManager.getConnection(url, user, password);          //Get a Statement object
  stmt = con.createStatement();                    // Create statement

  // Update data into a table
  stmt.executeUpdate( "UPDATE authors SET FirstName = 'Martin'  WHERE ID = 5" );
  stmt.close();          // Close statement
  con.close();           // Close connection
  }

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

Deleting data with Java

```
import java.sql.*;

public class Delete {
public Delete(String url, String user, String password) throws Exception {

try {

   Statement stmt;                                    // Register the JDBC driver for MySQL.
   Class.forName("com.mysql.jdbc.Driver");            // Get a connection to the database
   Connection con = DriverManager.getConnection(url, user, password);          //Get a Statement
      object
   stmt = con.createStatement();                      // Create statement

   // Delete data into a table
   stmt.executeUpdate( "DELETE FROM authors WHERE ID = 5" );
   stmt.close();          // Close statement
   con.close();           // Close connection
   }

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


Retrieving data with Java

```
import java.sql.*;

public class Select {
public Select(String url, String user, String password) throws Exception {

try {

   Statement stmt;                                    // Register the JDBC driver for MySQL.
   Class.forName("com.mysql.jdbc.Driver");            // Get a connection to the database
   Connection con = DriverManager.getConnection(url, user, password);

   // Create statement object
   Statement stmt = con.createStatement();

   // Select FirstName and LastName from the table authors
   ResultSet rs = stmt.executeQuery( "SELECT FirstName, LastName FROM authors" );   // "SELECT * FROM
      authors"

   ....
   // Print result set
  ....
   con.close();          // Close connection
   }

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

Additional methods

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

References

- 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

Questions?