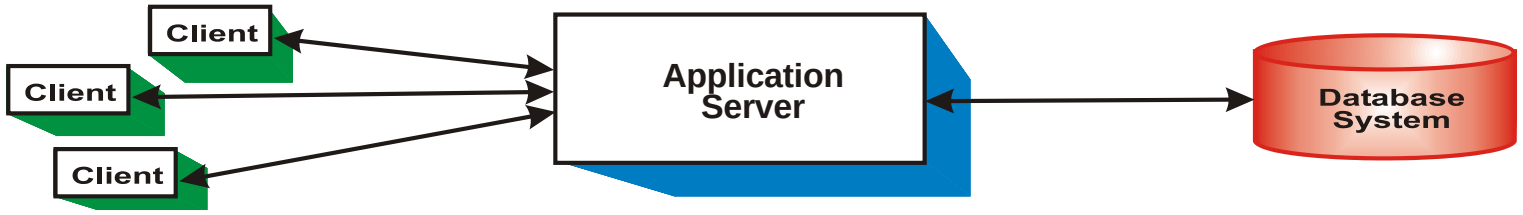
Network Computing Lecture 9

Professor Louise E. Moser

Spring 2012

Database Systems

- A **Database System** is typically used in the back-end tier of a three-tier architecture
  - **Client - Front-End Tier**
    - Browser-based client or Web server
  - **Application Server - Middle Tier**
    - Performs the business logic processing
  - **Database System - Back-End Tier**
    - Stores the data in database records on stable storage, i.e., disk



Database System

- A **Database System**, also referred to as a **Database Management System (DBMS)**, consists of
  - **Database Server**
  - **Database**, hierarchically structured into
    - **Tables**
      - **Records** (horizontal rows)
      - **Fields** (vertical columns)
    - Examples: Oracle, IBM DB2, Microsoft Access, MySQL (open source)

Database Tables

Customer

username	password	name	lastname	contact	address
fkart	pass	Firat	Kart	fkart@ece.ucsb.edu	Goleta, CA
moser	pass	Louise E.	Moser	moser@ece.ucsb.edu	

Movie

ID	Title	Director	Genre	Year
6677	The Fellowship of the Ring	Peter Jackson	Action,Adventure,Fantasy,Thriller	2001
5566	The Two Towers	Peter Jackson	Action,Adventure,Fantasy,Thriller	2002
5443	Iron Man	Jon Favreau	Action,Adventure,Drama,Sci-Fi,Thriller	2008

Database Records

- A **Database Record** contains
  - **Key Field** – Used to retrieve the record and also to cross-reference records in different tables
  - **Data Fields -** Whatever data you want to record
- Example: Customer Record
  - username is the Key Field
  - password, name, lastname, contact, address are the Data Fields

Customer					
username	password	name	lastname	contact	address
fkart	pass	Firat	Kart	fkart@ece.ucsb.edu	Goleta, CA
moser	pass	Louise E.	Moser	moser@ece.ucsb.edu	

  - Data can also be used to search the database
- Data in a record in one table can be the key for a record in another table

Database Structuring

- Try to avoid repeating the same data in different records in the same table
- Don't do this:

Customer Rating						
name	lastname	contact	title	director	year	rating
Firat	Kart	fkart@ece.ucsb.edu	Iron Man	Jon Favreau	2008	3
Firat	Kart	fkart@ece.ucsb.edu	The Two Towers	Peter Jackson	2002	5

Database Structuring

- Try to avoid repeating the same data in different records in different tables
- Don't do this:

Customer Rating

name	lastname	contact	title	director	year	rating
Firat	Kart	fkart@ece.ucsb.edu	Iron Man	Jon Favreau	2008	3
Firat	Kart	fkart@ece.ucsb.edu	The Two Towers	Peter Jackson	2002	5

Customer Purchase

name	lastname	contact	title	director	year	amoun	price
Firat	Kart	fkart@ece.ucsb.edu	Iron Man	Jon Favreau	2008	1	24.99
Firat	Kart	fkart@ece.ucsb.edu	The Two Towers	Peter Jackson	2002	2	15.99

Database Structuring

Customer

username	password	name	lastname	contact	address
fkart	pass	Firat	Kart	fkart@ece.ucsb.edu	Goleta, CA
moser	pass	Louise E.	Moser	moser@ece.ucsb.edu	

Movie

ID	Title	Director	Genre	Year
6677	The Fellowship of the Ring	Peter Jackson	Action,Adventure,Fantasy,Thriller	2001
5566	The Two Towers	Peter Jackson	Action,Adventure,Fantasy,Thriller	2002
5443	Iron Man	Jon Favreau	Action,Adventure,Drama,Sci-Fi,Thriller	2008

Customer Rating

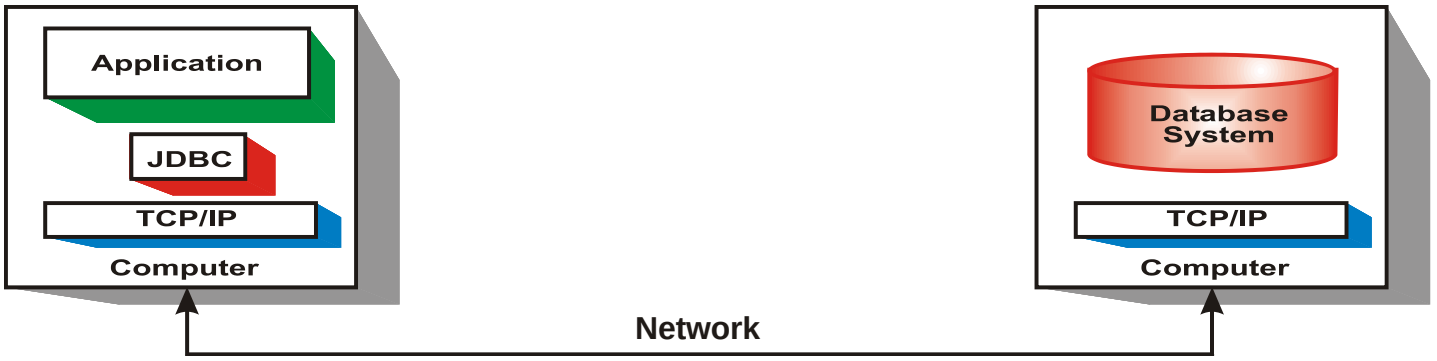
username	pID	rating
fkart	6677	3
fkart	5443	5

Customer Purchase

name	pID	amount	price
fkart	6677	1	24.99
fkart	5443	2	15.99

Java Database Connectivity

- **Java Database Connectivity** (**JDBC**) is a set of APIs that enable Java applications to connect to a database system
- Using JDBC, Java applications can store and retrieve information in a database system using **Structured Query Language** (**SQL**) statements
- JDBC encapsulates the functionality of database systems and abstracts the data for the application programmer



Structured Query Language

- **Structured Query Language (SQL)** is a standard relational database language
- A **Relational Database** consists of a set of tables
  - A **Table** is a set of records
  - Within a Table, all of the records have
    - Same number of fields
    - Same field names
- A Relational Database may also define
  - Constraints and Operations on the Tables, but we will not be concerned much about that

Structured Query Language

- **Structured Query Language (SQL)** enables you to
  - Create Tables
  - Insert Values into the Tables and Update those Values
  - Query the Tables and Retrieve the Results of the Queries
- SQL is completely declarative, i.e., it has no program control statements

Creating a Database Table in SQL

- The ***CREATE TABLE*** statement in SQL allows you to create a new table in a database
- You must specify the table name and a key, e.g.,

CREATE TABLE *tablename (*

key INTEGER ***PRIMARY KEY****,*

fName VARCHAR(30),

lName VARCHAR(30)

)

Inserting Values into a Database Table in SQL

- The ***INSERT INTO*** statement in SQL allows you to insert values into the data fields of the records in a database table
- You must specify the table name and the values to be inserted

  INSERT INTO *tablename* ***VALUES*** *(x, y)*

Selecting Parts of a Database Table in SQL

- The ***SELECT*** statement in SQL allows you to retrieve a copy of specific parts of a database table
- You must specify the table name and a filter (constraint), e.g.,

  SELECT *column name list* ***FROM*** *tablename* ***WHERE*** *filter*

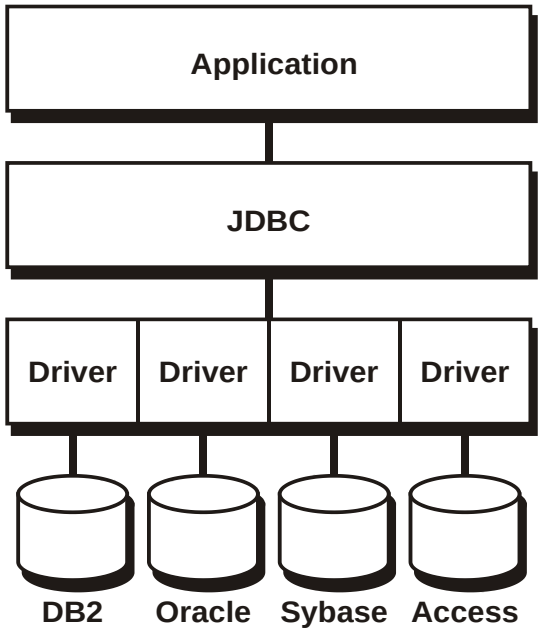
- Filters are specified as conditionals
- *SELECT* returns a table as the result

Open Database Connectivity

- **Open Database Connectivity** (**ODBC**) is a precursor to JDBC that was developed by Microsoft
- ODBC is a C-based interface to a SQL DBMS
- ODBC abstracts the proprietary APIs of different databases for the application programmer
- A standard ODBC driver is included with JDBC

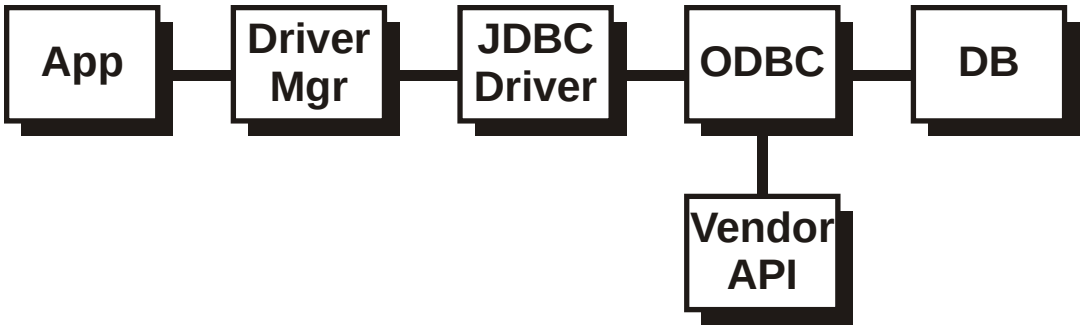
Database Drivers

- A DBMS is usually installed on a separate computer in the backend tier that is network accessible from a server computer in the middle tier
- Here we think of a DBMS as a device, rather than as software
- We access the DBMS through a standard database driver, rather than through native APIs
- A database driver gives JDBC a means of communicating with the DBMS
- There are four types of database drivers



Type 1: JDBC / ODBC Bridge Driver

- Translates JDBC calls into equivalent ODBC calls and lets ODBC do the work
- Requires an ODBC driver to be installed on each computer and uses some proprietary vendor APIs
- Has relatively low performance

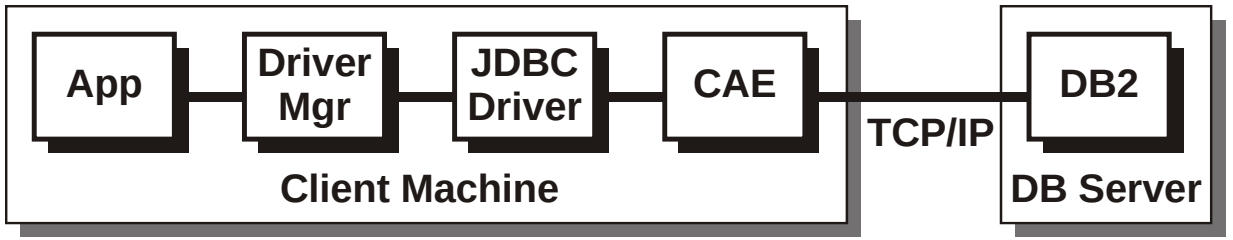


Type 1: JDBC / ODBC Bridge Driver (cont)

- Example: Microsoft Access
  - ODBC driver makes a *.mdb* file look like a DBMS
    - The *.mdb* file is not a DBMS
  - The ODBC driver must be able to find the *.mdb* file on the mapped drive, which can be anywhere in your network
  - If the DBMS is on a computer that runs Unix or Linux (rather than Windows), this won't work
- For an Applet, the DBMS must be on the Applet's home computer, because an Applet can make a network connection only to its home computer
  - This restriction can adversely affect performance

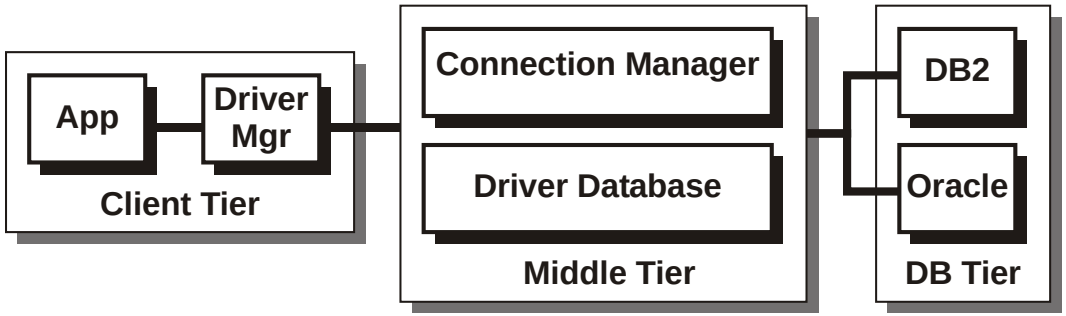
Type 2: Hybrid Driver

- Provides a partial Java / partial native API to the DBMS
- Example: IBM's DB2 Universal Database (UDB) provides
  - Native driver in the form of the Client Enabler (CAE)
    - Is installed on each computer that is a client of the DBMS
    - Allows access to any DB2 database to which the client computer has network connectivity
- JDBC driver must be placed in your JVM's *CLASSPATH*
- Supports connection pooling at the database for better performance



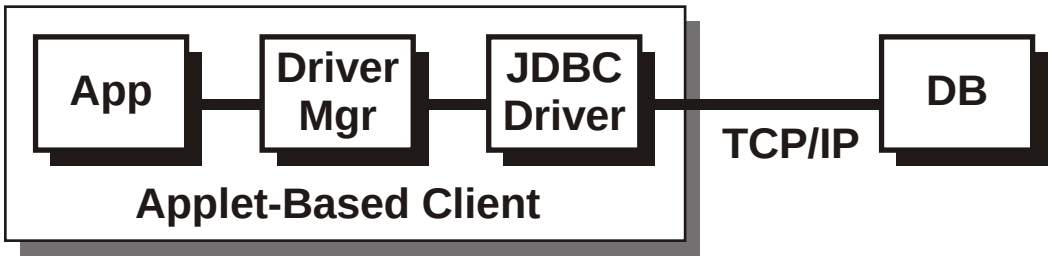
Type 3: Network Protocol Driver

- Converts JDBC calls into database-independent network-protocol calls and transmits them to the middle-tier server
- Middle-tier server usually runs on a separate computer from the DBMS
  - Converts the database-independent network-protocol calls into native-protocol calls for the target database
  - Is a JDBC driver for the Client Driver Manager
  - Uses a Type 1 or Type 2 driver for its connection to the database
  - Is a Connection Manager for the database
    - When started, opens a number of database connections
    - Routes database interactions to open database connections



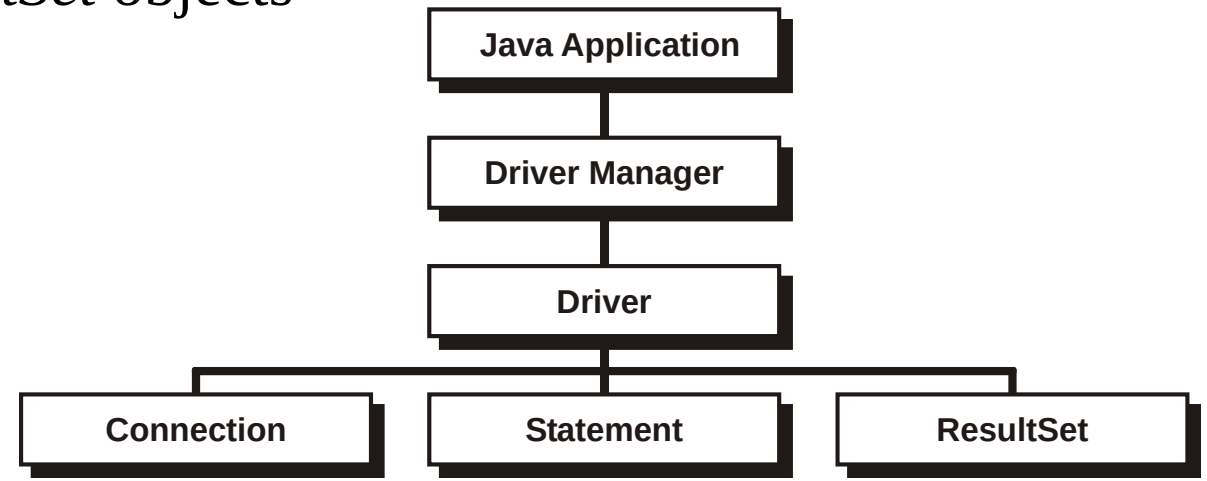
Type 4: All Java Driver

- Requires no special software on the client computers
- Good for Applet-based clients, because everything that the client requires is contained in the downloaded Applet



The JDBC Driver Manager and Driver

- Once the Driver is installed, you use the Driver Manager to load the Driver into your Java application object
- The Driver Manager
  - Groups together drivers so that multiple databases can be accessed from the same Java application object
  - Uses implementations of the *Connection*, *Statement*, and *ResultSet* objects



The JDBC Driver Manager and Driver (cont)

- The Driver creates and implements *Connection*, *Statement*, and *ResultSet* objects for the specific database
  - ***Connection*** - Establishes the link between the Java application and the DBMS
  - ***Statement*** - Encapsulates a query written in SQL and enables a JDBC object to compose a sequence of steps to look up information in the database
  - ***ResultSet*** - Contains a sequence of specific rows (with specific columns) obtained from a *Statement* call
- Using a *Connection*, the Driver forwards a *Statement* to the DBMS and the DBMS returns a *ResultSet*
- Using the *ResultSet*'s iterator *next()*, the JDBC object can step through each row in *ResultSet* and retrieve individual column fields (by field name or index) using the *getXXX()* methods of *ResultSet*

Using the Network Module

```
public class NetworkModule {
  public void initNetwork();
  ...
  public void shutdownNetwork();
}
```

- Create the connection to the database using the *initNetwork()* method
- …
- Close the connection to the database using the *shutdownNetwork()* method

initNetwork() Method

- Load the database driver and create the connection to the database

```
public void initNetwork() {
  // Load the database driver
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  // Create the URL representation of the database
  String url = "jdbc:odbc:Schedule";
  // Create the connection to the database
  dbConnection = DriverManager.getConnection(
                                    url, "username", "password");
}
```

shutdownNetwork() Method

- Close the connection to the DBMS to ensure that the DBMS has enough connections for other applications to connect to it

```
public void shutdownNetwork(){
}
```