DBMS-Data Base Management System

- It is a software that helps us to store, manipulate and retrieve data from a database
- DBMS contains information about a particular enterprise
- DBMS provides an environment that it both convenient and efficient to use

Purpose of DBMS

- Data redundancy and inconsistency- data is repeated in different files and Can even be different in different files.
- Difficulty in accessing data- It is not easy to retrieve information using a conventional file processing system. Convenient and efficient information retrieval is almost impossible using conventional file processing system.
- Integrity problems- The data values may need to satisfy some integrity constraints. For example the salary field value must be grater than 5000. We have to handle this through program code in file processing systems. But in database we can declare the integrity constraints along with definition itself.

Purpose of DBMS Systems

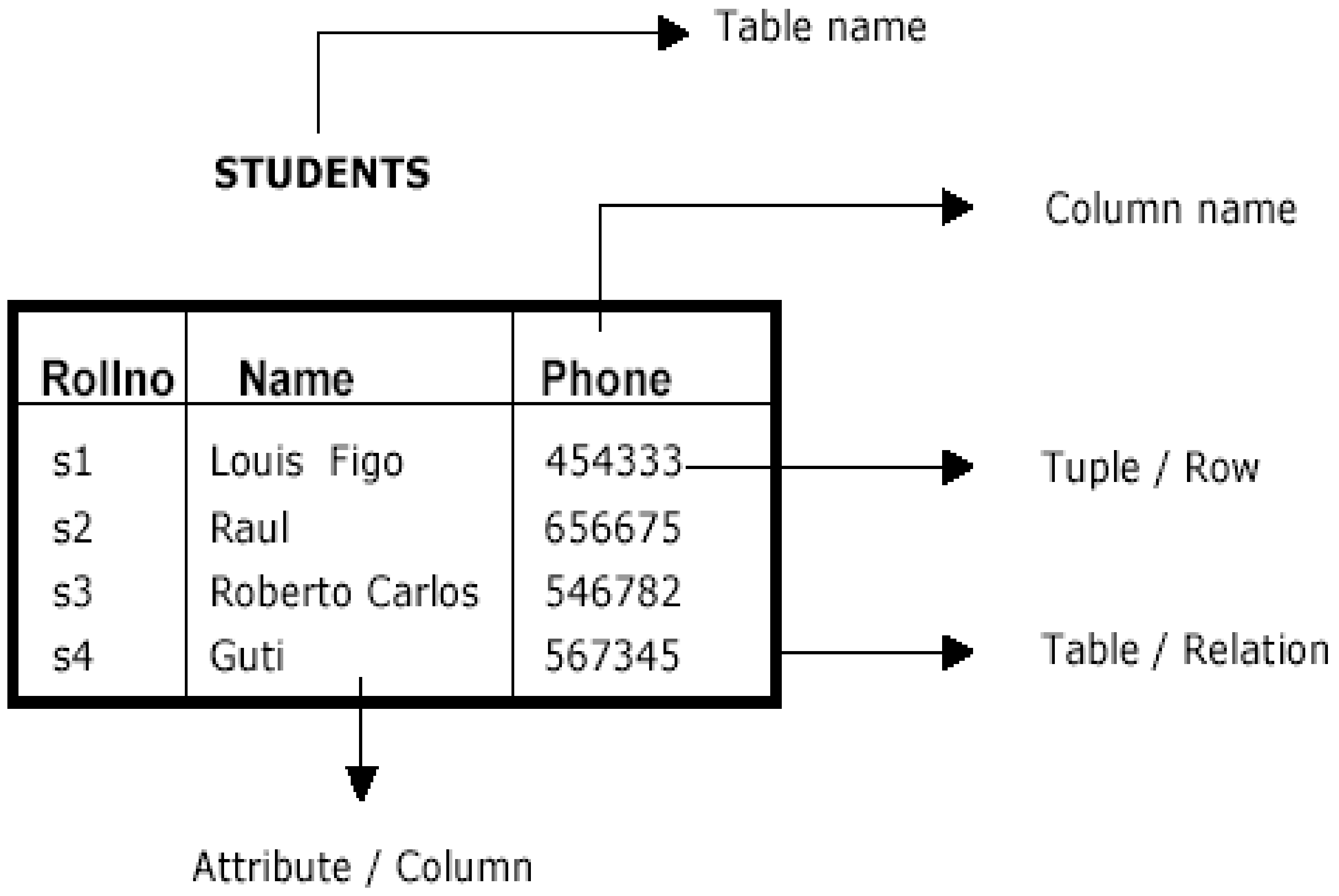
- Data isolation – Data are scattered in various files, and the files may be in different format, writing new application program to retrieve data is difficult.

  Concurrent access by multiple users- If multiple users are updating the same data simultaneously it will result in inconsistent data state. In file processing system it is very difficult to handle this using program code. This results in concurrent access anomalies.

  Security problems-Enforcing Security Constraints in file processing system is very difficult.

Relational Database Management System (RDBMS)

- ✓ A DBMS that is based on relational model is called as RDBMS. Relation model is most successful mode of all three models. Designed by E.F. Codd, relational model is based on the theory of sets and relations of mathematics.
- ✓ Relational model represents data in the form a table. A table is a two dimensional array containing rows and columns. Each row contains data related to an entity such as a student. Each column contains the data related to a single attribute of the entity such as student name.
- ✓ One of the reasons behind the success of relational model is its simplicity. It is easy to understand the data and easy to manipulate.



Rollno	Name	Phone
s1	Louis Figo	454333
s2	Raul	656675
s3	Roberto Carlos	546782
s4	Guti	567345

Figure 1: A table in relational model.

DBMS vs. RDBMS

- • Relationship among tables is maintained in a RDBMS whereas this not the case of DBMS.
- • DBMS is used for simpler business applications whereas RDBMS is used for more complex applications.
- • Although the foreign key concept is supported by both DBMS and RDBMS but its only RDBMS that enforces the rules.
- • RDBMS solution is required by large sets of data whereas small sets of data can be managed by DBMS.

JDBC

Java Database Connectivity in short called as JDBC is a java API (Java Programming interface) which enables the java programs to execute SQL statements. (The Java API is the set of classes included with the Java Development Environment. These classes are written using the Java language and run on the JVM. The Java API includes everything from collection classes to GUI classes)

It is an application programming interface that defines how a java programmer can access the   database in tabular format from Java code using a set of standard interfaces and classes written in the Java programming language. JDBC provides methods for querying and updating the data in Relational Database Management system such as SQL, Oracle etc.

.

JDBC

The Java application programming interface provides a mechanism for dynamically loading the correct Java packages and drivers and registering them with the JDBC Driver Manager that is used as a connection factory for creating JDBC connections which supports creating and executing statements such as SQL INSERT, UPDATE and DELETE. Driver Manager is the backbone of the jdbc architecture

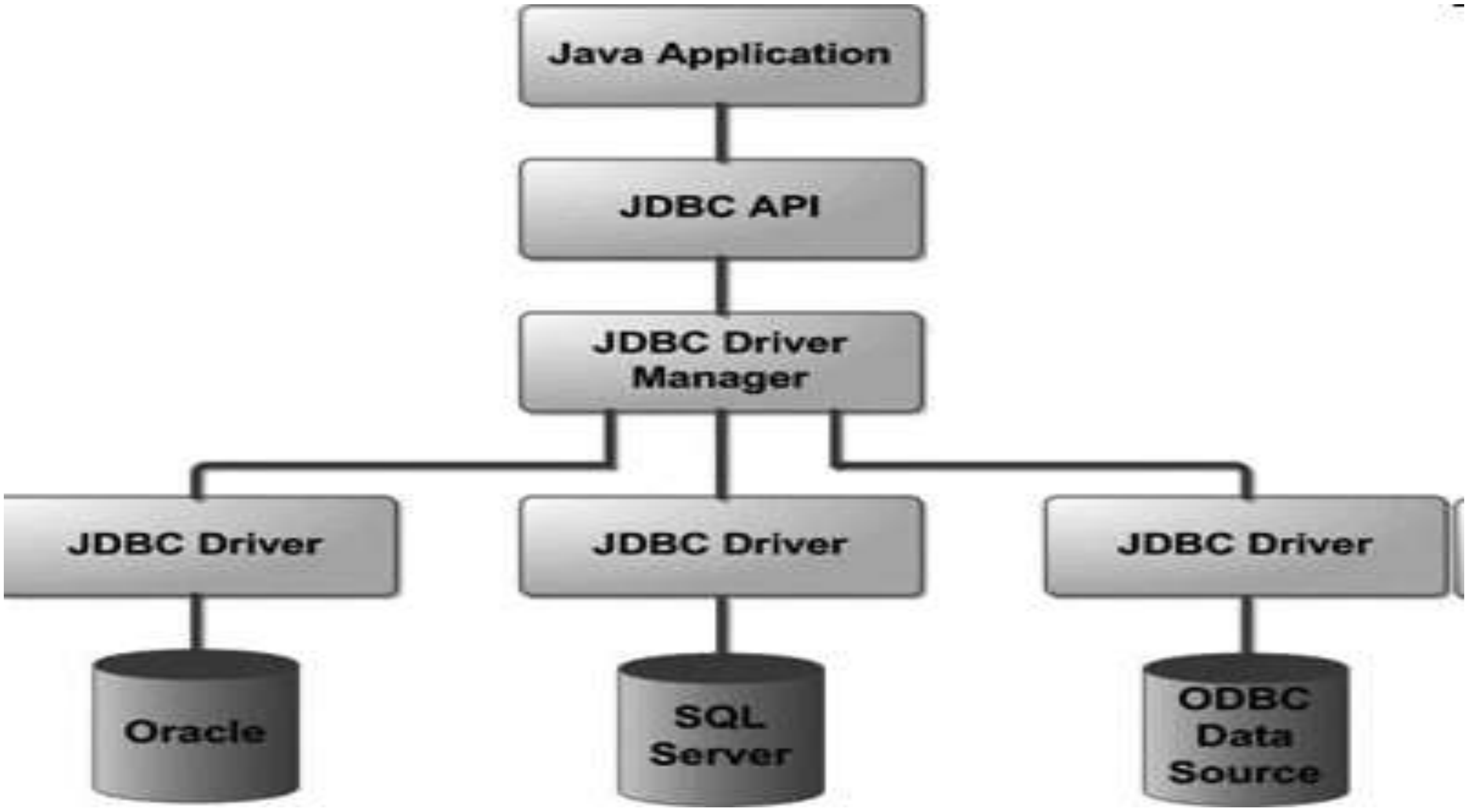
In short JDBC helps the programmers to write java applications that manage these three programming activities:

1. It helps us to connect to a data source, like a database.
2. It helps us in sending queries and updating statements to the database and
3. Retrieving and processing the results received from the database in terms of answering to your query.

What is JDBC?

- JDBC provides Java applications with access to most database systems via SQL
- The architecture and API closely resemble Microsoft's ODBC
- JDBC 1.0 was originally introduced into Java 1.1
  - JDBC 2.0 was added to Java 1.2
- JDBC is based on SQL-92
- JDBC classes are contained within the java.sql package
  - There are few classes
  - There are several interfaces

JDBC Architecture:



JDBC Architecture

The JDBC Architecture consists of two layers:

1.The JDBC API, which provides the **application-to-JDBC Manager** connection.

2. The JDBC Driver API, which supports the **JDBC Manager-to-Driver** Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is the backbone of the JDBC architecture. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Common JDBC Components:

The JDBC API provides the following interfaces and classes:

- **DriverManager:** This interface manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication subprotocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
- **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects
- **Connection :** Interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.

.

Common JDBC Components:

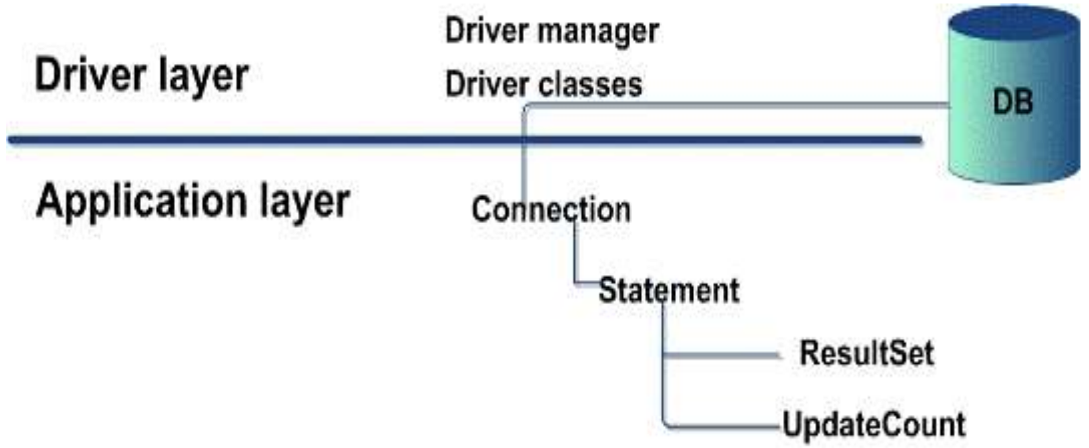
- **Statement :** Statement acts like a vehicle through which SQL commands can be sent. Through the connection object we create statement kind of objects.

  Statement stmt = conn.createStatement();

  This method returns object which implements statement interface
- **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
- **SQLException:** This class handles any errors that occur in a database application.

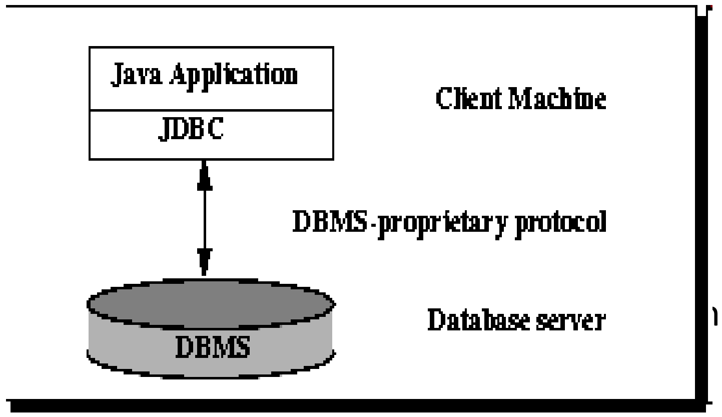
How the JDBC application works?

.A JDBC application can be logically divided into two layers:
1. **Driver layer**
2. **Application layer**
Driver layer consists of DriverManager class and the available JDBC drivers.
The application begins with requesting the DriverManager for the connection.
An appropriate driver is choosen and is used for establishing the connection. This connection is given to the application which falls under the application layer.
The application uses this connection to create Statement kind of objects, through which SQL commands are sent to backend and obtain the results.



JDBC Models

The JDBC API supports both two-tier and three-tier processing models for database access.

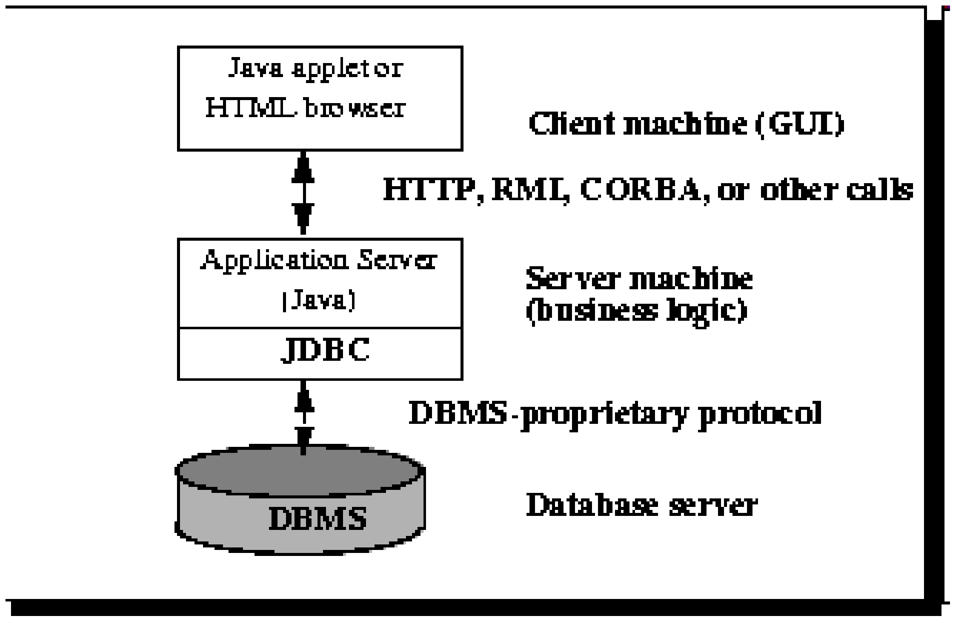


itecture for Data Access.

In the two-tier model, a Java application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed. A user's commands are delivered to the database or other data source, and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server. The network can be an intranet, which, for example, connects employees within a corporation, or it can be the Internet.

Three-tier model

- In the three-tier model, commands are sent to a "middle tier" of services, which then sends the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that it simplifies the deployment of applications. Finally, in many cases, the three-tier architecture can provide performance advantages.



Java applet or HTML browser
Client machine (GUI)
HTTP, RMI, CORBA, or other calls
Application Server (Java)
JDBC
Server machine (business logic)
DBMS-proprietary protocol
DBMS
Database server

Figure 2: Three-tier Architecture for Data Access.