

UNIT-1

Ques 1. Define dbms and file management system ?

Ans- Database management system (DBMS) is a collection of interrelated data and a set of programs to access those data. Some of the very well known DBMS are Microsoft Access, Microsoft [SQL Server](#), Oracle, [SAP](#), dBASE, FoxPro, IBM dB2, SQLite etc.

A file management system is an abstraction to store, retrieve, management and update a set of files. A File Management System keep track on the files and also manage them. Let's take an example of hierarchical Management System using its directories manage the different files in a tree structure.

Ques 2. What are the disadvantages of file management system over DBMS ?

Ans- The disadvantages of file management systems over DBMS are:

- a) Data redundancy and inconsistency
- b) Difficulty in accessing data
- c) Data isolation
- d) Integrity problems
- e) Atomicity problems
- f) Concurrent access anomalies

Ques 3. Which have more quick response DBMS or file management system ? how ?

Ans- Fast response to information requests: Because data are integrated into a single database, complex requests can be handled much more rapidly if the data were located in separate, non-integrated files. In many businesses, faster response means better customer service.

Ques 4. How DBMS provide program data insulation and data abstraction?

Ans- DBMS provide following levels of data abstraction

- a) Physical level
- b) logical level
- c) view level

Ques 5. Is there any difference between DBMS and file management system in terms of ACID properties?

Ans- DBMS ensures data integrity by managing transactions through ACID test = atomicity, consistency, isolation, durability. While such integrity is absent in file management system.

Ques 6. What are the advantages of DBMS over file management system ?

Ans- The advantages of DBMS over file management system are:

- a) Control redundancy
- b) Restrict unauthorized access
- c) Provide multiple user interfaces
- d) Enforce integrity constraints.
- e) Provide backup and recover

Ques 7. What are basic differences between DBMS and file management system ?

Ans- These are basic differences between DBMS and file management

- 1) Flexibility
- 2) Fast response to information requests
- 3) Multiple access
- 4) Lower user training costs
- 5) Less storage

Ques 8. Is there any goal differences in file management system and DBMS ?

Ans- Following are the goal differences between File Management System and DBMS-

1) Data Management- An FMS should provide data management services to the application. Generality with respect to storage devices. The FMS data abstractions and access methods should remain unchanged irrespective of the devices involved in data storage.

2) Validity- An FMS should guarantee that at any given moment the stored data reflect the operations performed on them.

3) Protection- Illegal or potentially dangerous operations on the data should be controlled by the FMS.

4) Concurrency- In multiprogramming systems, concurrent access to the data should be allowed with minimal differences.

5) Performance- Compromise data access speed and data transfer rate with functionality.

Ques 9. Which one take utilization of resources DBMS or file management system and how ?

Ans- Database is multi user and provides utilization of resources whereas in file management there is no utilization of resources.

Ques 10. How data redundancy is controlled in DBMS while not in file management system ?

Ans- It reduces data redundancy (duplication of data) and maximizes database integrity (data without errors). So DBMS is less redundant than file management system.

You can also download these interview questions on DBMS Vs File Management System in pdf from here [Interview Questions and Answers on Difference between DBMS and File Management System.pdf](#)

If you like to share your knowledge over this topic. write it in the comments and let's make a discussion over this.

Types of users in DBMS

Users are of 4 types:

1. Application programmers or Ordinary users
2. End users
3. Database Administrator (DBA)
4. System Analyst

1. Application programmers or Ordinary users: These users write application programs to interact with the database. Application programs can be written in some programming language such a COBOL, PL/I, C++, JAVA or some higher level fourth generation language. Such programs access the database by issuing the appropriate request, typically a SQL statement to DBMS.

2. End Users: End users are the users, who use the applications developed. End users need not know about the working, database design, the access mechanism etc. They just use the system to get their task done. End users are of two types:

a) Direct users b) Indirect users

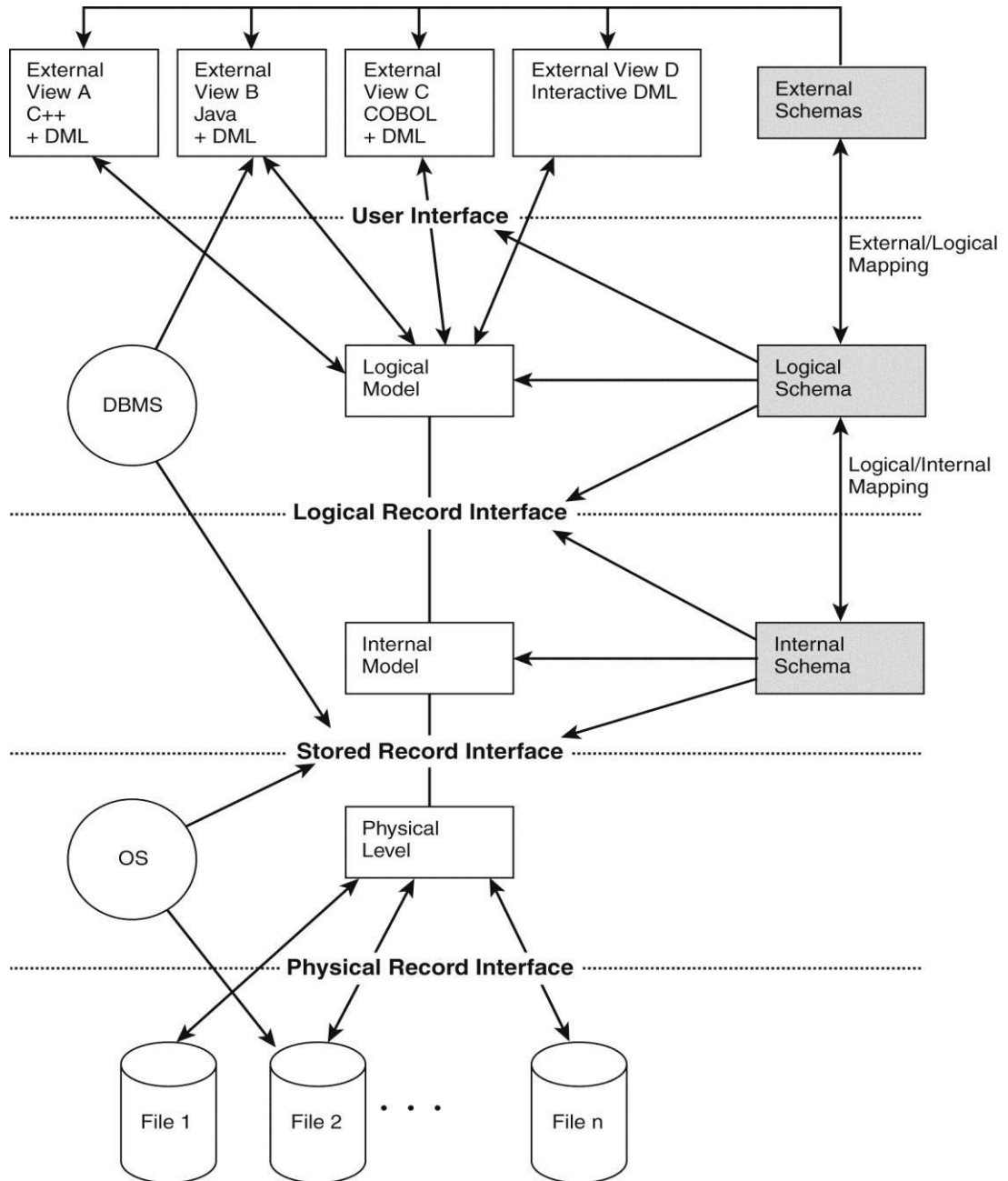
a) Direct users: Direct users are the users who se the computer, database system directly, by following instructions provided in the user interface. They interact using the application programs already developed, for getting the desired result. E.g. People at railway reservation counters, who directly interact with database.

b) Indirect users: Indirect users are those users, who desire benefit form the work of DBMS indirectly. They use the outputs generated by the programs, for decision making or any other purpose. They are just concerned with the output and are not bothered about the programming part.

3. Database Administrator (DBA): Database Administrator (DBA) is the person which makes the strategic and policy decisions regarding the data of the enterprise, and who provide the necessary technical support for implementing these decisions. Therefore, DBA is responsible for overall control of the system at a technical level. In database environment, the primary resource is the database itself and the secondary resource is the DBMS and related software administering these resources is the responsibility of the Database Administrator (DBA).

4. System Analyst: System Analyst determines the requirement of end users, especially naïve and parametric end users and develops specifications for transactions that meet these requirements. System Analyst plays a major role in database design, its properties; the structure prepares the system requirement statement, which involves the feasibility aspect, economic aspect, technical aspect etc. of the system.

Data and Related Structures



Data are actually stored as bits, or numbers and strings, but it is difficult to work with data at this level.

It is necessary to view data at different levels of abstraction.

Schema:

- Description of data at some level. Each level has its own schema.

We will be concerned with three forms of schemas:

- physical,
- conceptual, and
- external.

Physical Data Level

The **physical schema** describes details of how data is stored: files, indices, etc. on the random access disk system. It also typically describes the record layout of files and type of files (hash, b-tree, flat).

Early applications worked at this level - explicitly dealt with details. E.g., minimizing physical distances between related data and organizing the data structures within the file (blocked records, linked lists of blocks, etc.)

Problem:

- Routines are hardcoded to deal with physical representation.
- Changes to data structures are difficult to make.
- Application code becomes complex since it must deal with details.
- Rapid implementation of new features very difficult.

Conceptual Data Level

Also referred to as the Logical level

Hides details of the physical level.

- In the relational model, the conceptual schema presents data as a set of tables.

The DBMS maps data access between the conceptual to physical schemas automatically.

- Physical schema can be changed without changing application:
- DBMS must change mapping from conceptual to physical.
- Referred to as **physical data independence**.

External Data Level

In the relational model, the **external schema** also presents data as a set of relations. An external schema specifies a **view** of the data in terms of the conceptual level. It is tailored to the needs of a particular category of users. Portions of stored data should not be seen by some users and begins to implement a level of security and simplifies the view for these users

Examples:

- Students should not see faculty salaries.
- Faculty should not see billing or payment data.

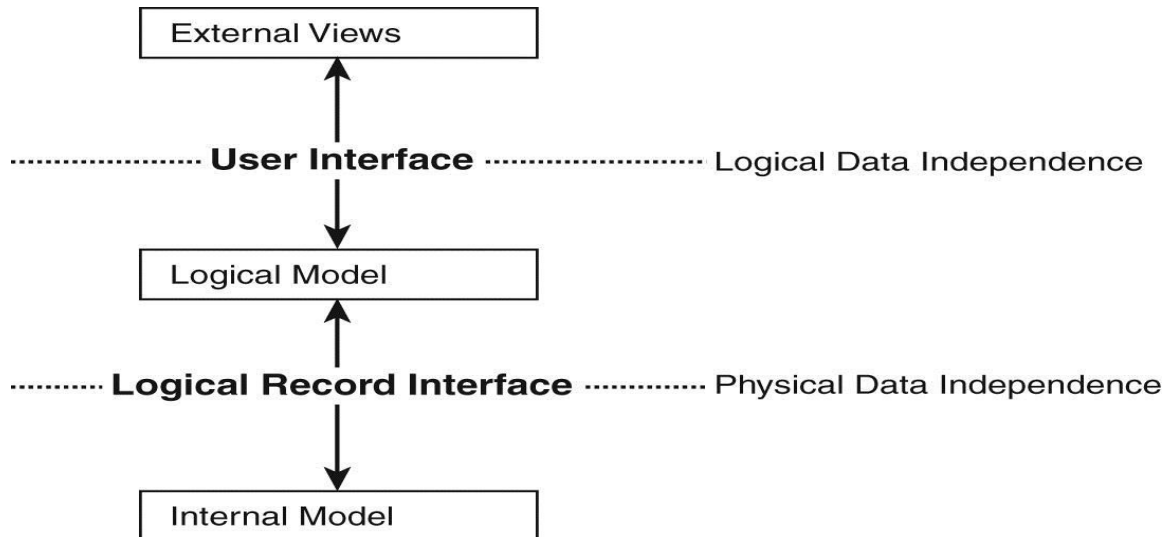
Information that can be derived from stored data might be viewed as if it were stored.

- GPA not stored, calculated when needed.

Applications are written in terms of an external schema. The external view is computed when accessed. It is not stored. Different external schemas can be provided to different categories of users. Translation from external level to conceptual level is done automatically by DBMS at run time. The conceptual schema can be changed without changing application:

- Mapping from external to conceptual must be changed.
- Referred to as **conceptual data independence**.

Data Independence



Logical data independence

- Immunity of external models to changes in the logical model
- Occurs at user interface level

Physical data independence

- Immunity of logical model to changes in internal model
- Occurs at logical interface level

Data Model

Schema: description of data at some level (e.g., tables, attributes, constraints, domains)

Model: tools and languages for describing:

- Conceptual/logical and external schema described by the data definition language (DDL)
- Integrity constraints, domains described by DDL
- Operations on data described by the data manipulation language (DML)
- Directives that influence the physical schema (affects performance, not semantics) are described by the storage definition language (SDL)

Entity-Relationship Model

A semantic model, captures meanings



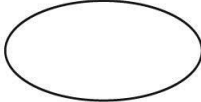

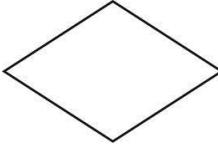
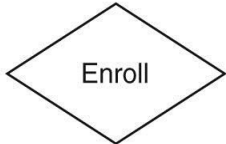

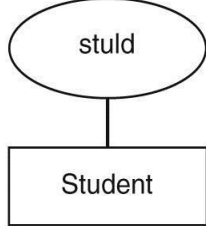
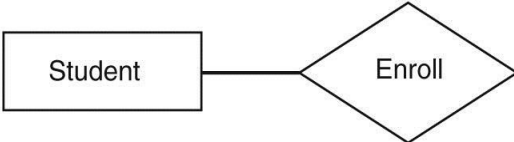
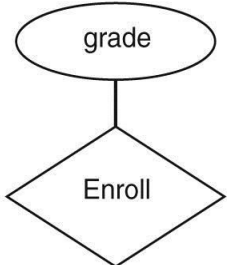
E-R modeling is a *conceptual level* model

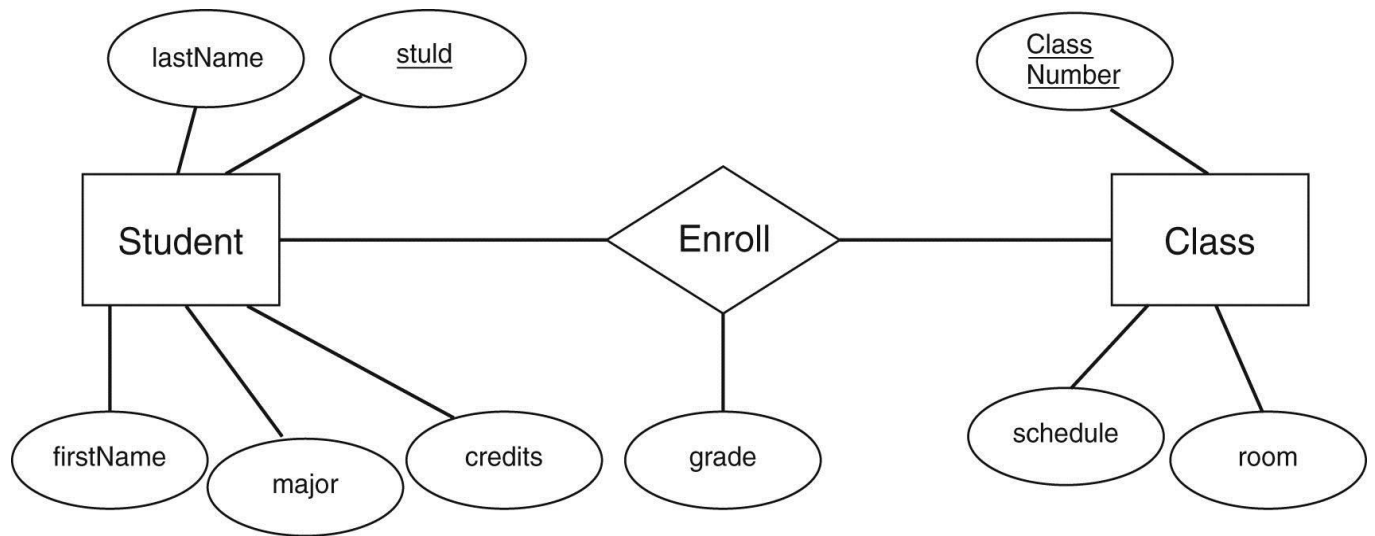
Proposed by P.P. Chen in 1970s

- **Entities** are real-world objects about which we collect data
- **Attributes** describe the entities
- **Relationships** are associations among entities
- **Entity set** – set of entities of the same type
- **Relationship set** – set of relationships of same type

Relationships sets may have descriptive attributes

Represented by E-R diagrams

SYMBOL	NAME	MEANING	EXAMPLE
	Rectangle	Entity Set	
	Oval	Attribute	
	Diamond	Relationship	
	Line	Links: Attribute to Entity	
		Entity Set to Relationship	
		Attribute to Relationship	



Relational Model

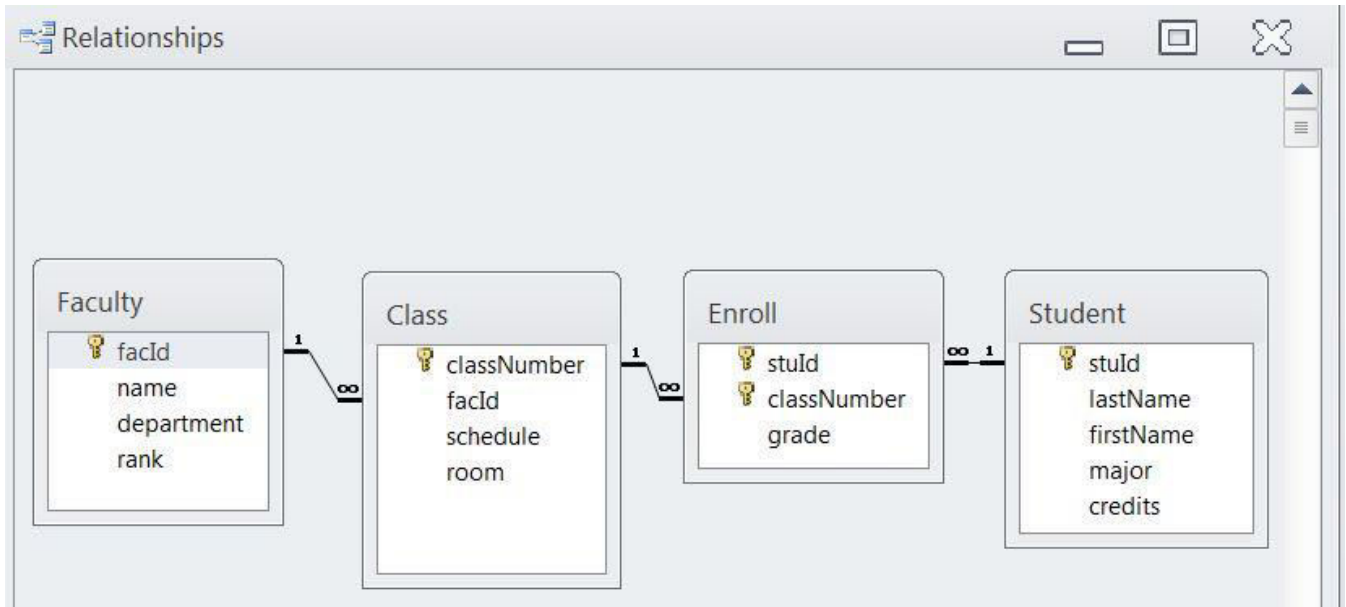
Record- and table-based model

Relational database modeling is a *logical-level* model

Proposed by E.F. Codd

- Based on mathematical relations
- Uses relations, represented as tables
- Columns of tables represent attributes
- Tables represent relationships as well as entities

Successor to earlier record-based models—network and hierarchical



Object-oriented Model

Uses the E-R modeling as a basis but extended to include **encapsulation, inheritance**

Objects have both state and behavior

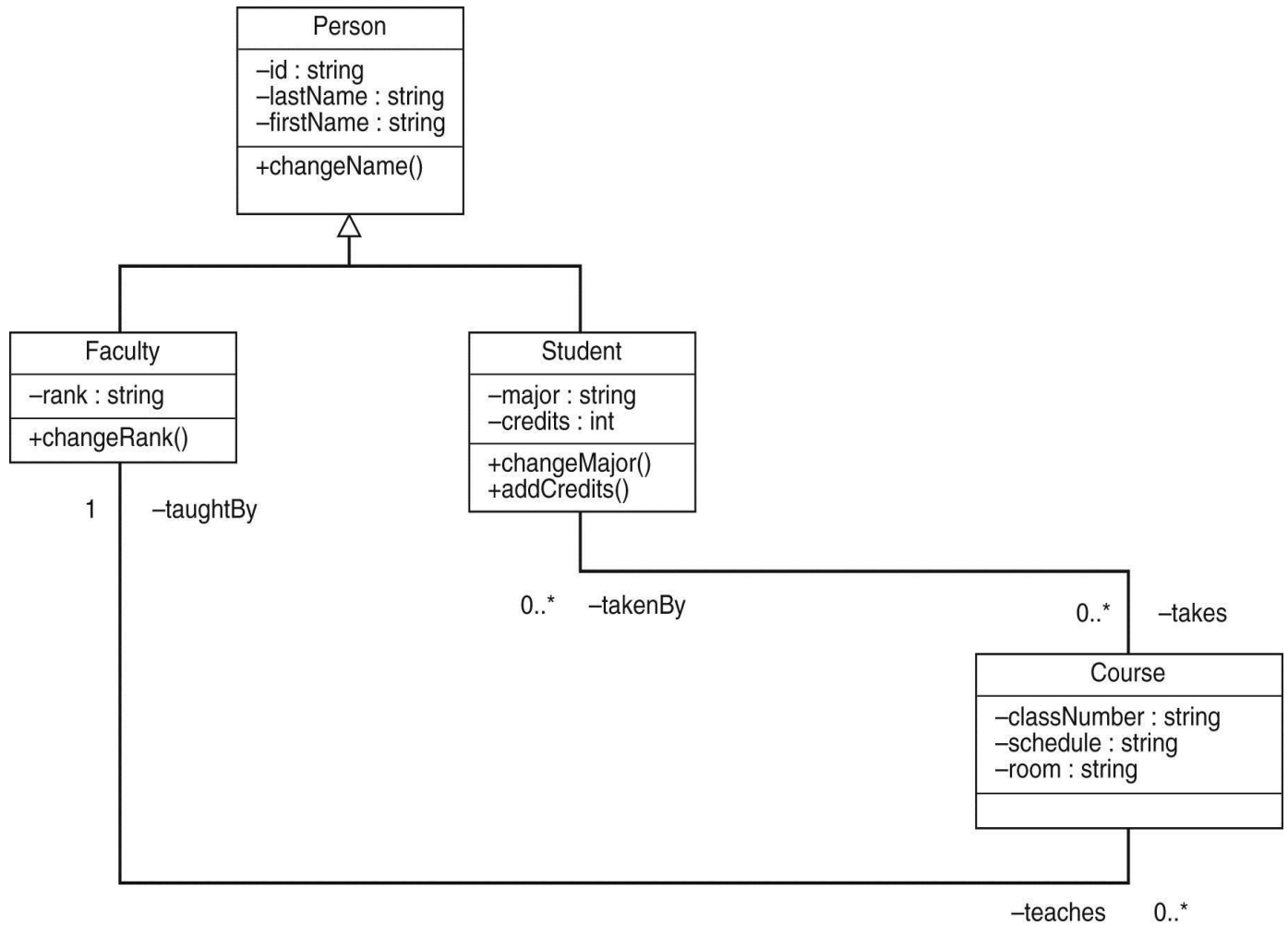
- **State** is defined by attributes
- **Behavior** is defined by methods (functions or procedures)

Designer defines classes with attributes, methods, and relationships

Class constructor method creates object instances

- Each object has a unique object ID
- Classes related by class hierarchies
- Database objects have persistence

Both *conceptual-level* and *logical-level* model



Object-relational model

Adds new complex datatypes to relational model

Adds objects with attributes and methods

Adds inheritance

SQL extended to handle objects in SQL:1999

Semi-structured Model

Collection of nodes, each with data, and with different schemas

Each node contains a description of its own contents

Can be used for integrating existing databases

XML tags added to documents to describe structure

XML tags identify elements, sub-elements, attributes in documents

XML DTD (Document Type Definition) or XML Schema used to define structure

UNIT-2

A **Relational Database management System**(RDBMS) is a database management system based on relational model introduced by E.F Codd. In relational model, data is represented in terms of tuples(rows).

RDBMS is used to manage Relational database. **Relational database** is a collection of organized set of tables from which data can be accessed easily. Relational Database is most commonly used database. It consists of number of tables and each table has its own primary key.

- The relational model represents the database as a collection of **relations**
- Each **relation** resembles a **table** of values
- When a relation is thought of as a **table** of values, each row in the table represents a collection of related data values
- A row is called a tuple
- A column header is called an attribute
- The table is called relation

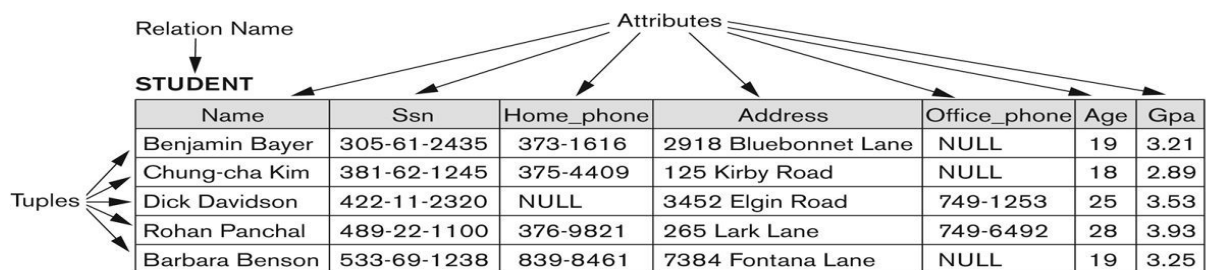


Figure 5.1
The attributes and tuples of a relation STUDENT.

- Each value in a tuple is an **atomic** value
- Hence, composite and multi-valued attributes are **not** allowed
- This model is sometimes called the **flat relational model** Much of the theory behind the relational model was developed with this assumption, which is called **first normal form** assumption

<u>Informal Terms</u>		<u>Formal Terms</u>
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation(instance)

concepts of domain, attribute, tuple, relation important of null values:

What is Domain?

- A **Domain** D is a set of **atomic** values.
- Atomic means that each value in the domain is indivisible as far as the relational model is concerned
- It means that if we separate an atomic value, the value itself become meaningless, for example:

- SSN
 - Local_phone_number
 - Names
 - Employee_ages
-

What is Table/Relation ?

In Relational database, a **table** is a collection of data elements organised in terms of rows and columns. A table is also considered as convenient representation of **relations**. But a table can have duplicate tuples while a true **relation** cannot have duplicate tuples. Table is the most simplest form of data storage. Below is an example of Employee table.

ID	Name	Age	Salary
1	Hemanth	34	13000
2	Bharath	28	15000
3	Nitin	20	18000
4	venky	42	19020

What is a Record/Row/tuple ?

A single entry in a table is called a **Record** or **Row**. A row is called a tuple

A **Record** in a table represents set of related data. For example, the above **Employee** table has 4 records. Following is an example of single record.