**UNIT-I**

**Chapter -II**

**Syllabus:**

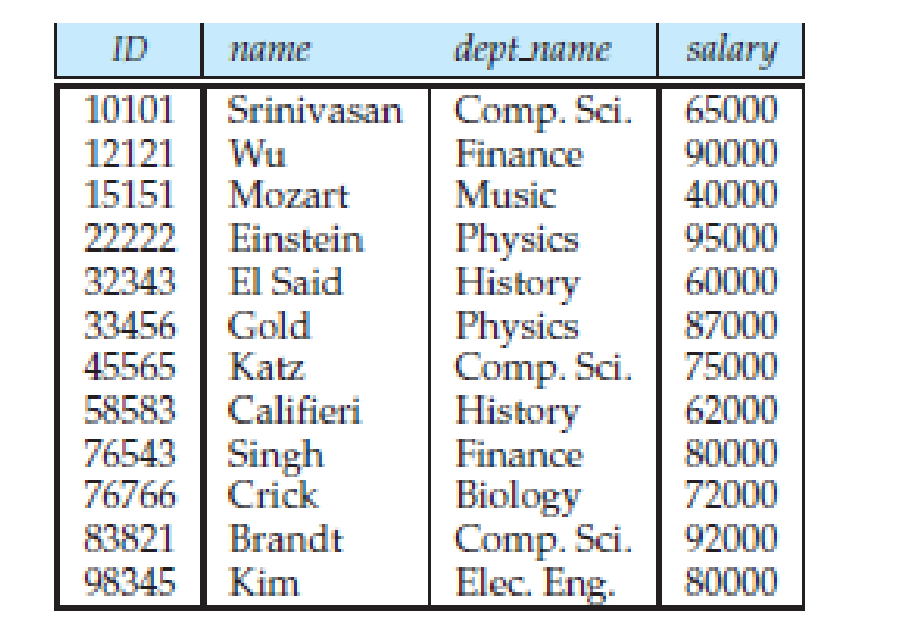
**Introduction to the Relational Model:Structure of Relational Databases, Keys, Database Schema,Relational Query Languages, Relational Operations.**

**Relational Model:**

* The relational model is an abstract model used to organize and manage thedata stored in a database. It stores data in two-dimensional inter related tables, also known as relations.
* The relational model is the primary data model for commercial dataprocessing applications.

**Structure of Relational Databases:**

* A relational database consists of a collection of **tables**, each of which is assigned a unique name.
* For example, consider the *instructor* table of Figure:1, which stores information about instructors. The table has four column headers: *ID*, *name*, *dept name*, and *salary*.
* Each row of this table records information about an instructor, consisting of the instructor’s *ID*, *name*, *dept name*, and *salary*.



**Figure 1: The *instructor* relation**

* The term**relation** is used to refer to a tablein the relational model.
* The term **tuple** is used to refer to a row.
* The term **attribute** refers to a column of a table.
* The term **relation instance** to refer to a specific instance of a relation, i.e., containing a specific set of rows.
* Examining Figure 1, we can see that the relation *instructor* has four attributes:

***ID*, *name*, *dept\_name*, and *salary*.**

* The instance of *instructor* shown in Figure 1 has 12 tuples, corresponding to 12 instructors.
* The order in which tuples appear in a relation is irrelevant since a relation is a *set* of tuples.
* For each attribute of a relation, there is a set of permitted values, called the **domain** of that attribute. Thus, the domain of the *salary* attribute of the *instructor* relation is the set of all possible salary values, while the domain of the *name* attribute is the set of all possible instructor names.
* A domain is **atomic** if elements of the domain are considered to be indivisible units.
* For example, suppose the table *instructor* had an attribute *phone number*, which can store a set of phone numbers corresponding to the instructor. Then the domain of *phone number* would not be atomic, since an element of the domain is a set of phone numbers, and it has subparts, namely the individual phone numbers in the set.
* The **null** value is a special value that signifies that the value is unknown or does not exist.

**Database Schema**

* The **database schema** is the logical design of the database.
* The **database instance** is a snapshot of the data in the database at a given instant in time.
* The concept of a relation corresponds to the programming-language notion of a variable, while the concept of a **relation schema** corresponds to the programming-language notion of type definition.
* In general, a relation schema consists of a list of attributes and their corresponding domains. The concept of a relation instance corresponds to the programming-language notion of a value of a variable. The value of a given variable may change with time; similarly, the contents of a relation instance may change with time as the relation is updated.
* Consider the *department* relation, the schema for that relation is

***department* (*dept\_name*, *building*, *budget*)**

* Note that the attribute *dept name* appears in both the *instructor* schema and the *department* schema. This duplication is not a coincidence. Rather, using common attributes in relation schemas is one way of relating tuples of distinct relations.
* For example, suppose we wish to find the information about all the instructors who work in the Watson building. We look first at the *department* relation to find the *dept\_name*of all the departments housed in Watson. Then, for each such department, we look in the *instructor* relation to find the information about the instructor associated with the corresponding *dept name*.
* Similarly we have some other relations in University database. Following are such relation schemas.
* *section* (*course\_id*, *sec\_id*, *semester*, *year*, *building*,

*room\_number*, *time\_slot\_id*)

* *teaches* (*ID*, *course\_id*, *sec\_id*, *semester*, *year*)
* *student* (*ID*, *name*, *dept\_name*, *tot\_cred*)
* *advisor* (*s\_id*, *i\_id*)
* *takes* (*ID*, *course\_id*, *sec\_id*, *semester*, *year*, *grade*)
* *classroom* (*building*, *room\_number*, *capacity*)
* *time slot* (*time\_slot\_id*, *day*, *start\_time*, *end\_time*)

**Keys:**

* Keys play an important role in the relational database.It is used to uniquely identify any tuple or record or row of data from the table. It is also used to establish and identify relationships between tables.
* **Super key:**A superkey of a relation is a set of one or more attributes whose values are guaranteed to identify tuples in the relation uniquely.
* **Example: {*ID*} and {ID,name} are both superkeys of *instructor.***
* **Candidate Key:** minimal superkey, that is, a set of attributes that forms a superkey, but none of whose subsets is a super key.
* **Example: {*ID*} is a candidate key for *Instructor***
* **Primary Key:** One of the candidate keys of a relation is chosen as its primary key. It is chosen by the database designer as the principal means of identifying tuples within a relation.
* **Example: {*ID*} is a Primary key for *Instructor***
* **Foreign Key:**A relation, say r1, may include among its attributes the primary key of anotherrelation, say r2. This attribute is called a **foreign key** from r1, referencing r2.

The relation r1 is also called the **referencing relation** of the foreign key dependency,

and r2 is called the **referenced relation**of the foreign key.

* **Example:** Theattribute dept\_name in instructor relation is a foreign key frominstructor, referencing department,since dept\_name is the primary key of department.
* **Referential integrity constraint**:A referential integrity constraint requires that the values appearing in specified attributes of any tuple in the referencing relation also appear in specified attributes of at least one tuple in the referenced relation.
* There are two referential integrity constraint:
* Insert Constraint: Value cannot be inserted in referencingrelation if the value is not lying-in referenced relation.
* Delete Constraint:Value cannot be deleted from referenced relation if the value is laying in referencing relation.

**Schema Diagrams**

* A database schema, along with primary key and foreign key dependencies, can be depicted by **schema diagrams**.
* Below Figure shows the schema diagram for our university organization. Each relation appears as a box, with the relation’s name at the top in blue, and the attributes listed inside the box.
* Primary key **attributes are shown underlined**. Foreign key dependencies appear as arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation.

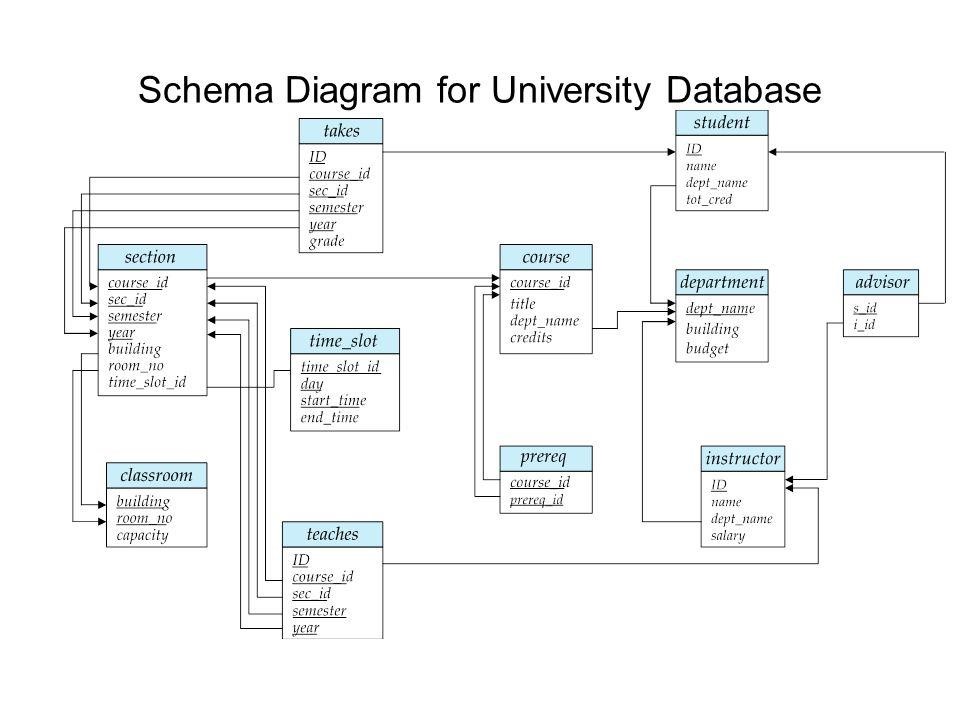


Figure 2: Schema diagram for the university database.

**Relational Query Languages:**

* A query language is a language in which a user requests information from thedatabase. These languages are usually on a level higher than that of a standardprogramming language.
* Query languages can be categorized as either proceduralor nonprocedural.
* In a **procedural language**, the user instructs the system toperform a sequence of operations on the database to compute the desired result.
* **Example:**Relational Algebra
* In a **nonprocedural language**, the user describes the desired information withoutgiving a specific procedure for obtaining that information.
* **Example:**Tuple Relational Calculus and Domain Relational Calculus

**Relational Operations:**

* All procedural relational query languages provide a set of operations that can be

applied to either a single relation or a pair of relations.

* These operations havethe desired property that their result is always a single relation. Thisproperty allows one to combine several of these operations in a modular way.
* Since the result of a relational query is itself a relation, relationaloperations can be applied to the results of queries as well as to the given set of relations.
* Relational operations on a single relation are: Selection and Projection.
* Relational operations on multiple relations are: join, Cartesian product, Union, Intersection, and Set difference.

