A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.
Keeping organizational information in a file-processing system has a number of major disadvantages

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation
- Integrity problems
- Atomicity problems
- Concurrent-access anomalies
- Security problems

Instances and Schemas: Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an instance of the database. The overall design of the database is called the database schema. The concept of database schemas and instances can be understood by analogy to a program written in a programming language. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an instance of a database schema.

- A relational database consists of a collection of **tables**, each of which is assigned a unique name.
- In general, a row in a table represents a **relationship** among a set of values. Since a table is a collection of such relationships, there is a close correspondence between the concept of **table** and the mathematical concept of **relation**, from which the relational data model takes its name. In mathematical terminology, a tuple is simply a sequence (or list) of values. A relationship between n values is represented mathematically by an n-tuple of values, that is, a tuple with n values, which corresponds to a row in a table.
- Thus, in the relational model the term **relation** is used to refer to a table, while the term **tuple** is used to refer to a row. Similarly, the term **attribute** refers to a column of a table.
- The order in which tuples appear in a relation is **irrelevant**, since a relation is a set of tuples. Thus, whether the tuples of a relation are listed in sorted order, or are unsorted, does not matter.
- For each attribute of a relation, there is a set of permitted values, called the **domain** of that attribute.
- **Relation instance** refer to a specific instance of a relation, that is, containing a specific set of rows.
- In general, a **relation schema** consists of a list of attributes and their corresponding domains.
- The concept of a **relation schema** corresponds to the programming language notion of type definition. 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. In contrast, the **schema of a relation** does not generally change.
- In terms of database, the **database schema** is the logical design of the database, and the **database instance**, is a snapshot of the data in the database at a given instant in time.

Keys: We must have a way to specify how tuples within a given relation are distinguished. This is expressed in terms of their attributes. That is, the values of the attribute values of a tuple must be such that they can uniquely identify the tuple. In other words, no two tuples in a relation are allowed to have exactly the same value for all attributes.
Superkey: A **superkey** is a set of **one or more attributes** that, taken collectively, allow us to identify uniquely a tuple in the relation.

- A superkey may contain **extraneous** attributes. Here, extraneous attributes means if we ignore these attributes, the reaming attributes will still work as a superkey.

Candidate key: We are often interested in superkeys for which no proper subset is a superkey. Such **minimal superkeys** are called **candidate keys**.

Primary key: We shall use the term primary key to denote a candidate key that is chosen by the database designer as the principal means of identifying tuples within a relation. The designation of a key represents a constraint in the real-world enterprise being modeled. Thus, primary keys are also referred to as **primary key constraints**.

A key (whether primary, candidate, or super) is a **property** of the **entire relation**, rather than of the **individual tuples**. Any two individual tuples in the relation are prohibited from having the same value on the key attributes at the same time.

Foreign key: A **foreign-key constraint** from attribute(s) A of relation r_1 to the primary-key B of relation r_2 states that on any database instance, the value of A for each tuple in r_1 must also be the value of B for some tuple in r_2. Attribute set A is called a foreign key from r_1, referencing r_2. The relation r_1 is also called the **referencing relation** of the foreign-key constraint, and r_2 is called the **referenced relation**.

- For example, the attribute `dept_name` in **instructor** is a **foreign key** from instructor, referencing **department**. Note that `dept_name` is the primary key of **department**.
- Here, the name of the attribute is same in both the relations – **instructor** and **department**. However, the name can be different also.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

(a) `instructor` relation.

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

(b) `department` relation.

Figure 1: `instructor` and `department` relations.

Note that in a **foreign-key constraint**, the referenced attribute(s) must be the **primary key** of the **referenced relation**. The more general case, a **referential-integrity constraint**, relaxes the requirement that the referenced attributes form the primary key of the referenced relation.

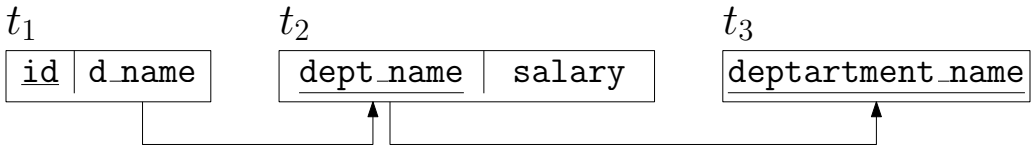

Figure 2: Three relations t_1, t_2 and t_3 where the same attribute in t_2 is a **primary key** in t_2 and is also a **foreign key** from t_2, referencing t_3.