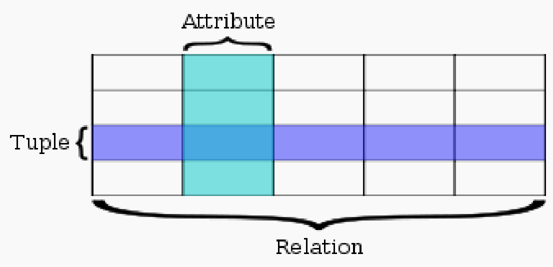
Terminology[edit]



Relational database theory uses mathematical terminology, which are roughly equivalent to the SQL database terminology concerning *normalization*. The table below summarizes some of the most important relational database terms and their SQL database equivalents. It was first introduced in 1970 following the work of E.F.Codd.

A row or tuple has a *relation* schema, but an entire database has a *relational* schema.

SQL	Relational database	Description
Row	***Tuple***	*Data set* with specific instances in the range of each member
Column name; Column data	***Attribute*** name; Attribute value	Labeled member in the set of elements common to all *data sets*; A name, word, number, phrase, etc.
Table	***Relation***; Base relvar	Formal data structure
Set of column names	Relation scheme; Set of attributes	A schema
View; Query result; Result set	Derived relvar	A data report from the RDBMS in response to a query

Relations or Tables[edit]

A *relation* is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints. The relational model specifies that the tuples of a relation have no specific order and that the tuples, in turn, impose no order on the attributes. Applications access data by specifying queries, which use operations such as *select* to identify tuples, *project* to identify attributes, and *join* to combine relations. Relations can be modified using the *insert*, *delete*, and *update* operators. New tuples can supply explicit values or be derived from a query. Similarly, queries identify tuples for updating or deleting. Tuples by definition are unique. If the tuple contains a candidate or primary key then obviously it is unique, however, a primary key need not be defined for a row or record to be a tuple. The definition of a tuple requires that it be unique. The definition does not require a Primary Key to be defined. The attributes of a tuple may be referred to as a super key.

Key

The values of the attribute values of an entity must be such that they can *uniquely identify* the entity. In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes.
A *key* allows us to identify a set of attributes that suffice to distinguish entities from each other. Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

KEYS

• An important constraint on the entities of an entity type is the **key** or **uniqueness constraint** on attributes.

• A key is an attribute (also known as column or field) or a combination of attribute that is used to identify records.

• Sometimes we might have to retrieve data from more than one table, in those cases we require to join tables with the help of keys.

• The purpose of the key is to bind data together across tables without repeating all of the data in every table

• Such an attribute is called a **key attribute**, and its values can be used to identify each entity uniquely.

• For example, the Name attribute is a key of the COMPANY entity type because no two companies are allowed to have the same name.

• For the PERSON entity type, a typical key attribute is SocialSecurityNumber.

• Sometimes, several attributes together form a key, meaning that the combination of the attribute values must be distinct for each entity.

• If a set of attributes possesses this property, we can define a composite attribute that becomes a key attribute of the entity type.
The various types of key with e.g. in SQL are mentioned below, (For examples let suppose we have an Employee Table with attributes 'ID' , 'Name' ,'Address' , 'Department_ID' ,'Salary')
(I) Super Key – An attribute or a combination of attribute that is used to identify the records uniquely is known as Super Key. A table can have many Super Keys.
E.g. of Super Key
1 ID
2 ID, Name
3 ID, Address
4 ID, Department_ID
5 ID, Salary
6 Name, Address
7 Name, Address, Department_ID ………… So on as any combination which can identify the records uniquely will be a Super Key.
(II) Candidate Key – It can be defined as minimal Super Key or

irreducible Super Key. In other words an attribute or a combination of attribute that identifies the record uniquely but none of its proper subsets can identify the records uniquely.

E.g. of Candidate Key

1 Code

2 Name, Address

For above table we have only two Candidate Keys (i.e. Irreducible Super Key) used to identify the records from the table uniquely. Code Key can identify the record uniquely and similarly combination of Name and Address can identify the record uniquely, but neither Name nor Address can be used to identify the records uniquely as it might be possible that we have two employees with similar name or two employees from the same house.

(III) Primary Key – A Candidate Key that is used by the database designer for unique identification of each row in a table is known as Primary Key. A Primary Key can consist of one or more attributes of a table. A primary key uniquely specifies a tuple within a table. In order for an attribute to be a good primary key it must not repeat. While natural attributes (attributes used to describe the data being entered) are sometimes good primary keys, surrogate keys are often used instead. A surrogate key is an artificial attribute assigned to an object which uniquely identifies it (for instance, in a table of information about students at a school they might all be assigned a student ID in order to differentiate them)

E.g. of Primary Key - Database designer can use one of the Candidate Key as a Primary Key. In this case we have "Code" and "Name, Address" as Candidate Key, we will consider "Code" Key as a Primary Key as the other key is the combination of more than one attribute.

(IV) Foreign Key – A foreign key is an attribute or combination of attribute in one base table that points to the candidate key (generally it is the primary key) of another table. The purpose of the foreign key is to ensure referential integrity of the data i.e. only values that are supposed to appear in the database are permitted.

E.g. of Foreign Key – Let consider we have another table i.e. Department Table with Attributes "Department_ID", "Department_Name", "Manager_ID", "Location_ID" with Department_ID as an Primary Key. Now the Department_ID attribute of Employee Table (dependent or child table) can be defined as the Foreign Key as it can reference to the Department_ID attribute of the Departments table (the referenced or parent table), a Foreign Key value must match an existing value in the parent table or be NULL.

(V) Composite Key – If we use multiple attributes to create a Primary Key then that Primary Key is called Composite Key (also called a Compound Key or Concatenated Key).

E.g. of Composite Key, if we have used "Name, Address" as a Primary Key then it will be our Composite Key.

(VI) Alternate Key – Alternate Key can be any of the Candidate Keys except for the Primary Key.

E.g. of Alternate Key is "Name, Address" as it is the only other Candidate Key which is not a Primary Key.

(VII) Secondary Key – The attributes that are not even the Super Key but can be still used for identification of records (not unique) are known as Secondary Key.

E.g. of Secondary Key can be Name, Address, Salary, Department_ID etc. as they can identify the records but they might not be unique.