DATABASE MANAGEMENT SYSTEM

Course Objectives:

● To understand the fundamental concepts of Database Management Systems
● To acquire the knowledge of database query languages and transaction processing
● To understand systematic database design approaches
● To acquire the skills to use a powerful, flexible, and scalable general-purpose databases to handle Big Data
● To be familiar with advances in databases and applications

Course Outcomes:

On completion of the course, learners should be able to
CO1: Analyze and design Database Management System using ER model
CO2: Implement database queries using database languages
CO3: Normalize the database design using normal forms
CO4: Apply Transaction Management concepts in real-time situations

CO5: Use NoSQL databases for processing unstructured data
CO6: Differentiate between Complex Data Types and analyze the use of appropriate data types

Unit I	**Introduction to Database Management Systems and ER Model**	**06 Hours**

Introduction, Purpose of Database Systems, Database-System Applications, View of Data, Database Languages, Database System Structure, Data Models. **Database Design and ER Model**: Entity, Attributes, Relationships, Constraints, Keys, Design Process, Entity-Relationship Model, ER Diagram, Design Issues, Extended E-R Features, converting ER and EER diagram into tables.

Introduction to Database Management System

As the name suggests, the database management system consists of two parts.

They are:

1. Database and 2. Management System

What is a Database?

To find out what database is, we have to start from data, which is the basic building block of any DBMS.

Data: Facts, figures, statistics etc. having no particular meaning (e.g. 1, ABC, 19 etc).

Record: Collection of related data items, e.g. in the above example the three data items had no meaning. But if we organize them in the following way, then they collectively represent meaningful information.

Roll Name Age 1 ABC 19

Table or Relation: Collection of related records.

Roll Name Age 1 ABC 19 2 DEF 22 3 XYZ 28 The columns of this relation are called Fields, Attributes or Domains.

The rows are called Tuples or Records.

Database: Collection of related relations.

Introduction

The information storage and retrieval has become very important in our day-to-day life. The old era of manual system is no longer used in most of the places. For example, to book your airline tickets or to deposit your money in the bank the database systems may be used. The database system makes most of the operations automated. A very good example for this is the billing system used for the items purchased in a super market. Obviously this is done with the help of a database application package. Inventory systems used in a drug store or in manufacturing industry are some more examples of database. We can add similar kind of examples to this list.

Apart from these traditional database systems, more sophisticated database systems are used in the Internet where a large amount of information is stored and retrieved with efficient search engines. For instance, http://www.google.com is a famous web site that enables users to search for their favorite information on the net. In a database we can store starting from text data to very complex data like audio, video, etc.

1.1 Database Management Systems (DBMS)

A database is a collection of related data stored in a standard format, designed to be shared by multiple users. A database is defined as “A collection of interrelated data items that can be processed by one or more application programs”.
A database can also be defined as “A collection of persistent data that is used by the application systems of some given enterprise”. An enterprise can be a single individual (with a small personal database), or a complete corporation or similar large body (with a large shared
database), or anything in between.
Example: A Bank, a Hospital, a University, a Manufacturing company

Data

Data is the raw material from which useful information is derived. The word data is the plural of Datum. Data is commonly used in both singular and plural forms. It is defined as raw facts
or observations. It takes variety of forms, including numeric data, text and voice and images.
Data is a collection of facts, which is unorganized but can be made organized into useful
information. The term Data and Information come across in our daily life and are often
interchanged.

Example: Weights, prices, costs, number of items sold etc.

Information

Data that have been processed in such a way as to increase the knowledge of the person who
uses the data. The term data and information are closely related. Data are raw material
resources that are processed into finished information products. The information as data that has been processed in such way that it can increase the knowledge of the person who uses it.
In practice, the database today may contain either data or information.

Data Processing

The process of converting the facts into meaningful information is known as data processing.
Data processing is also known as information processing.
Metadata Data that describe the properties or characteristics of other data

UNIT-1

Introduction to Database Management System

As the name suggests, the database management system consists of two parts. They are:

1. Database and
2. Management System

What is a Database?

To find out what database is, we have to start from data, which is the basic building block of any DBMS.

Data: Facts, figures, statistics etc. having no particular meaning (e.g. 1, ABC, 19 etc).

Record: Collection of related data items, e.g. in the above example the three data items had no meaning. But if we organize them in the following way, then they collectively represent meaningful information.

Roll	**Name**	**Age**
1	ABC	19

Table or **Relation**: Collection of related records.

Roll	**Name**	**Age**
1	ABC	19
2	DEF	22
3	XYZ	28

The columns of this relation are called **Fields**, **Attributes** or **Domains**. The rows are called **Tuples** or **Records**.

Database: Collection of related relations. Consider the following collection of tables:

T1

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

T2

Roll	Address
1	KOL
2	DEL
3	MUM

T3

Roll	Year
1	I
2	II
3	I

T4

Year	Hostel
I	H1
II	H2

We now have a collection of 4 tables. They can be called a "related collection" because we can clearly find out that there are some common attributes existing in a selected pair of tables. Because of these common attributes we may combine the data of two or more tables together to find out the complete details of a student. Questions like "Which hostel does the youngest student live in?" can be answered now, although ***Age*** and ***Hostel*** attributes are in different tables.

A database in a DBMS could be viewed by lots of different people with different responsibilities.

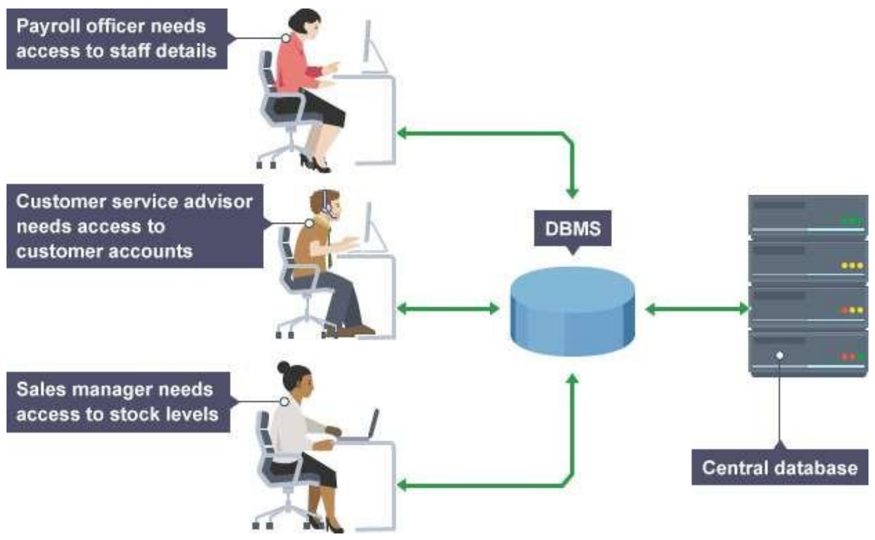

Payroll officer needs access to staff details
Customer service advisor needs access to customer accounts
Sales manager needs access to stock levels
DBMS
Central database

Figure 1.1: Empolyees are accessing Data through DBMS

For example, within a company there are different departments, as well as customers, who each need to see different kinds of data. Each employee in the company will have different levels of access to the database with their own customized **front-end** application.

In a database, data is organized strictly in row and column format. The rows are called **Tuple** or **Record**. The data items within one row may belong to different data types. On the other hand, the columns are often called **Domain** or **Attribute**. All the data items within a single attribute are of the same data type.

What is Management System?

A **database-management system** (DBMS) is a collection of interrelated data and a set of programs to access those data. This is a collection of related data with an implicit meaning and hence is a database. 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*. By **data,** we mean known facts that can be recorded and that have implicit meaning.

The management system is important because without the existence of some kind of rules and regulations it is not possible to maintain the database. We have to select the particular attributes which should be included in a particular table; the common attributes to create relationship between two tables; if a new record has to be inserted or deleted then which tables should have to be handled etc. These issues must be resolved by having some kind of rules to follow in order to maintain the integrity of the database.

Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

Because information is so important in most organizations, computer scientists have developed a large body of concepts and techniques for managing data. These concepts and technique form the focus of this book. This
chapter briefly introduces the principles of database systems.

Database Management System (DBMS) and Its Applications:

A Database management system is a computerized record-keeping system. It is a repository or a container for collection of computerized data files. The overall purpose of DBMS is to allow he users to define, store, retrieve and update the information contained in the database on demand. Information can be anything that is of significance to an individual or organization.

Databases touch all aspects of our lives. Some of the major areas of application are as follows:

1. Banking
2. Airlines
3. Universities
4. Manufacturing and selling

5. Human resources

Enterprise Information

- ***Sales*: For customer, product, and purchase information.
- ***Accounting*: For payments, receipts, account balances, assets and other accounting information.
- ***Human resources*: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
- ***Manufacturing*: For management of the supply chain and for tracking production of items in factories, inventories of items inwarehouses and stores, and orders for items.

Online retailers: For sales data noted above plus online order tracking,generation of recommendation lists, and maintenance of online product evaluations.

Banking and Finance

- *Banking*: For customer information, accounts, loans, and banking transactions.
- *Credit card transactions*: For purchases on credit cards and generation of monthly statements.
- *Finance*: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.
- *Universities*: For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- *Airlines*: For reservations and schedule information.

Airlines were among the first to use databases in a geographically distributed manner.

- *Telecommunication*: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

Purpose of Database Systems

Database systems arose in response to early methods of computerized management of commercial data. As an example of such methods, typical of the 1960s, consider part of a university organization that, among other data, keeps information about all instructors, students, departments, and course offerings. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including programs to:

- ✓ Add new students, instructors, and courses
- ✓ Register students for courses and generate class rosters
- ✓ Assign grades to students, compute grade point averages (GPA), and generate transcripts System programmers wrote these application programs to meet the needs of the university.

New application programs are added to the system as the need arises. For example, suppose that a university decides to create a new major (say, computer science). As a result, the university creates a new department and creates new permanent files (or adds information to existing files) to record information about all the instructors in the department, students in that major, course offerings, degree requirements, etc. The university may have to write new application programs to deal with rules specific to the new major. New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs.

This typical **file-processing system** is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems. Keeping organizational information in a file- processing system has a number of major disadvantages:

Data redundancy and inconsistency. Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department. This redundancy leads to higher storage and access cost. In addition, it may lead to **data inconsistency**; that is, the various copies of the same data may no longer agree. For example, a changed student address may be reflected in the Music department records but not elsewhere in the system.

Difficulty in accessing data. Suppose that one of the university clerks needs to find out the names of all students who live within a particular postal-code area. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of *all* students.

The university clerk has now two choices: either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same clerk needs to trim that list to include only those students who have taken at least 60 credit hours. As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory. The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

Data isolation. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

Integrity problems. The data values stored in the database must satisfy certain types of **consistency constraints**. Suppose the university maintains an account for each department, and records the balance amount in each account. Suppose also that the university requires that the account balance of a department may never fall below zero. Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

Advantages of DBMS:

Controlling of Redundancy: Data redundancy refers to the duplication of data (i.e storing same data multiple times). In a database system, by having a centralized database and centralized control of data by the DBA the unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space.

Improved Data Sharing : DBMS allows a user to share the data in any number of application programs.

Data Integrity : Integrity means that the data in the database is accurate. Centralized control of the data helps in permitting the administrator to define integrity constraints to the data in the database. For example: in customer database we can can enforce an integrity that it must accept the customer only from Noida and Meerut city.

Security : Having complete authority over the operational data, enables the DBA in ensuring that the only mean of access to the database is through proper channels. The DBA can define authorization checks to be carried out whenever access to sensitive data is attempted.

Data Consistency : By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example: is a customer address is stored only once, we cannot have disagreement on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage that results from redundant data storage.

Efficient Data Access : In a database system, the data is managed by the DBMS and all access to the data is through the DBMS providing a key to effective data processing

Enforcements of Standards : With the centralized of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc.

Data Independence : Ina database system, the database management system provides the interface between the application programs and the data. When changes are made to the data representation, the meta data obtained by the DBMS is changed but the DBMS is continues to provide the data to application program in the previously used way. The DBMs handles the task of transformation of data wherever necessary.

Reduced Application Development and Maintenance Time : DBMS supports many important functions that are common to many applications, accessing data stored in the DBMS, which facilitates the quick development of application.

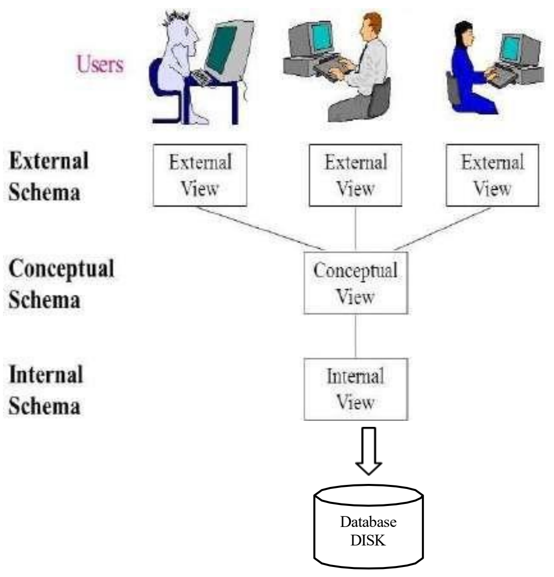


Figure 1.2 : Levels of Abstraction in a DBMS

· **Physical level (or Internal View / Schema)**: The lowest level of abstraction describes *how* the data are actually stored. The physical level describes complex low-level data structures in detail.

· **Logical level (or Conceptual View / Schema)**: The next-higher level of abstraction describes *what* data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as **physical data independence**. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

· **View level (or External View / Schema):** The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database. Figure 1.2 shows the relationship among the three levels of abstraction. An analogy to the concept of data types in programming languages may clarify the distinction among levels of abstraction. Many high-level programming languages support the notion of a structured type

- *department*, with fields *dept_name*, *building*, and *budget*
- *course*, with fields *course_id*, *title*, *dept_name*, and *credits*
- *student*, with fields *ID*, *name*, *dept_name*, and *tot_cred*

At the physical level, an *instructor*, *department*, or *student* record can be described as a block of consecutive storage locations. The compiler hides this level of detail from programmers. Similarly, the database system hides many of the lowest-level storage details from database programmers. Database administrators, on the other hand, may be aware of certain details of the physical organization of the data.

At the logical level, each such record is described by a type definition, as in the previous code segment, and the interrelationship of these record types is defined as well. Programmers using a programming language work at this level of abstraction. Similarly, database administrators usually work at this level of abstraction.

Finally, at the view level, computer users see a set of application programs that hide details of the data types. At the view level, several views of the database are defined, and a database user sees some or all of these views. In addition to hiding details of the logical level of the database, the views also provide a security mechanism to prevent users from accessing certain parts of the database. For example, clerks in the university registrar office can see only that part of the database that has information about students; they cannot access information about salaries of instructors.

· **Relational Model**. The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as **relations**. The relational model is an example of a record-based model.
Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

Entity-Relationship Model. The entity-relationship (E-R) data model uses a collection of basic objects, called

entities, and *relationships* among these objects.

An entity is a "thing" or "object" in the real world that is distinguishable from other objects. The entity- relationship model is widely used in database design.

Database Languages

A database system provides a **data-definition language** to specify the database

schema and a **data-manipulation language** to express database queries and updates. In practice, the data- definition and data-manipulation languages are not
two separate languages; instead they simply form
parts of a single database language, such as the
widely used SQL language.

Data-Manipulation Language

A **data-manipulation language (DML)** is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

Data-Definition Language (DDL)

We specify a database schema by a set of definitions expressed by a special language called a **data-definition language** (**DDL**). The DDL is also used to specify additional properties of the data.

We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a **data storage and definition** language. These statements define the implementation details of the database schemas, which are usually hidden from the users.

The data values stored in the database must satisfy certain **consistency constraints**.

For example, suppose the university requires that the account balance of a department must never be negative. The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated. In general, a constraint can be an arbitrary predicate pertaining to the database. However, arbitrary predicates may be costly to test. Thus, database systems implement integrity constraints that can be tested with minimal overhead.

· **Domain Constraints**. A domain of possible values must be associated with every attribute (for example, integer types, character types, date/time types). Declaring an attribute to be of a particular domain acts as a constraint on the values that it can take. Domain constraints are the most elementary form of integrity constraint. They are tested easily by the system whenever a new data item is entered into the database.

Conceptual Database Design - Entity Relationship(ER) Modeling:

Database Design Techniques

1. ER Modeling (Top down Approach)
2. Normalization (Bottom Up approach)

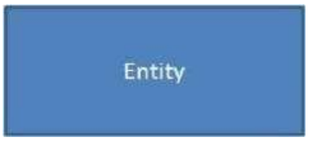
What is ER Modeling?

A graphical technique for understanding and organizing the data independent of the actual database implementation

We need to be familiar with the following terms to go further.

Entity

Any thing that has an independent existence and about which we collect data. It is also known as entity type. In ER modeling, notation for entity is given below.



Entity instance

Entity instance is a particular

member of the entity type.

Example for entity instance : A

particular employee **Regular**

Entity

An entity which has its own key

attribute is a regular entity.

Example for regular entity :

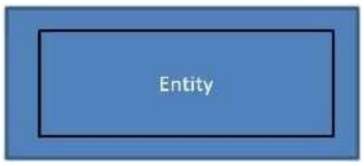
Employee.

Weak entity

An entity which depends on other entity for its existence and doesn't have any key attribute of its own is a weak entity.

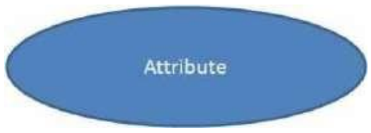
Example for a weak entity : In a parent/child relationship, a parent is considered as a strong entity and the child is a weak entity.

In ER modeling, notation for weak entity is given below.



Attributes

Properties/characteristics which



describe entities are called attributes.

In ER modeling, notation for

attribute is given below.

Domain of Attributes

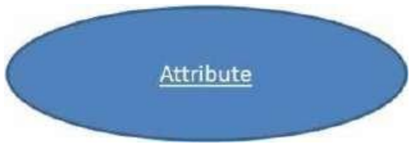
The set of possible values that an attribute can take is called the domain of the attribute. For example, the attribute day may take any value from the set {Monday, Tuesday ... Friday}. Hence this set can be termed as the domain of the attribute day.

Key attribute

The attribute (or combination of attributes) which is unique for every entity instance is called key attribute.

E.g the employee_id of an employee, pan_card_number of a person etc.If the key attribute consists of two or more attributes in combination, it is called a composite key.

In ER modeling, notation for key attribute is given below.



Simple attribute

If an attribute cannot be divided into simpler

components, it is a simple attribute. Example

for simple attribute : employee_id of an

employee.

Composite attribute

If an attribute can be split into components, it is called a composite attribute.

Example for composite attribute : Name of the employee which can be split into First_name, Middle_name, and Last_name.

Single valued Attributes

If an attribute can take only a single value for each

entity instance, it is a single valued attribute. example for

single valued attribute : age of a student. It can take only

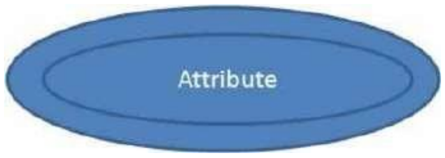
one value for a particular student. **Multi-valued**

Attributes

If an attribute can take more than one value for each entity instance, it is a multi-valued attribute. Multi-valued

example for multi valued attribute : telephone number of an employee, a particular employee may have multiple telephone numbers.

In ER modeling, notation for multi-valued attribute is given below.



Stored Attribute

An attribute which need to be stored

permanently is a stored attribute

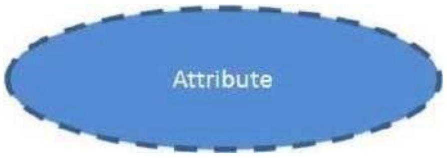
Example for stored attribute : name

of a student

Derived Attribute

An attribute which can be calculated or derived based on other attributes is a derived attribute.

Example for derived attribute : age of employee which can be calculated from date of birth and current date. In ER modeling, notation for derived attribute is given below.

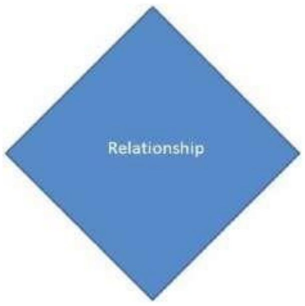


Relationships

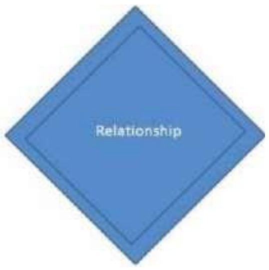
Associations between entities are called relationships

Example : An employee works for an organization. Here "works for" is a relation between the entities employee and organization.

In ER modeling, notation for relationship is given below.



However in ER Modeling, To connect a weak Entity with others, you should use a weak relationship notation as given below



Degree of a Relationship

Degree of a relationship is the number of entity types involved. The n-ary relationship is the general form for degree n. Special cases are unary, binary, and ternary ,where the degree is 1, 2, and 3, respectively.

Example for unary relationship : An employee ia

a manager of another employee Example for

binary relationship : An employee works-for

department.

Example for ternary relationship : customer purchase item from a shop keeper

Cardinality of a Relationship

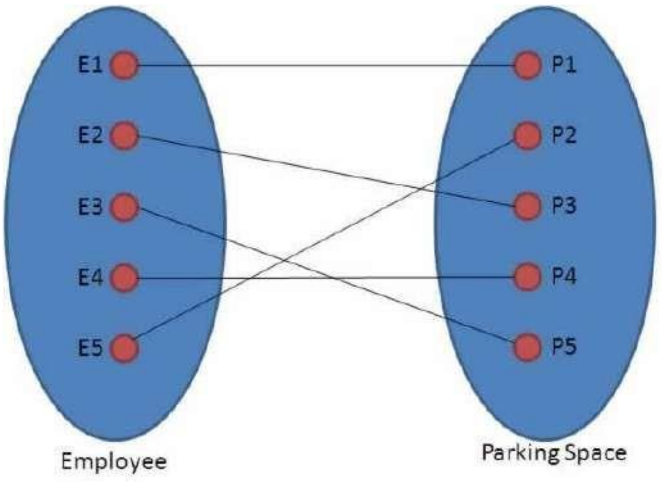
Relationship cardinalities specify how many of each entity type is allowed. Relationships can have four possible connectivities as given below.

1. One to one (1:1) relationship
2. One to many (1:N) relationship
3. Many to one (M:1) relationship
4. Many to many (M:N) relationship

The minimum and maximum values of this connectivity is called the cardinality of the relationship

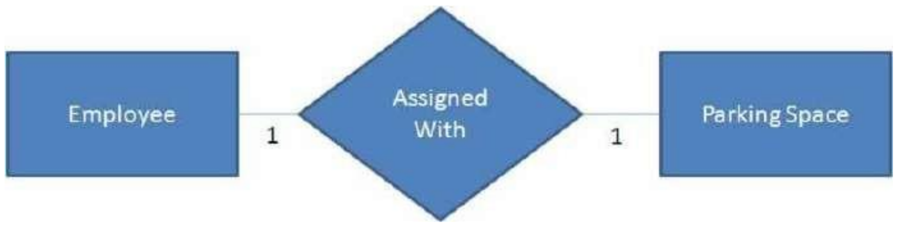
Example for Cardinality – One-to-One (1:1)

Employee is assigned with a parking space.



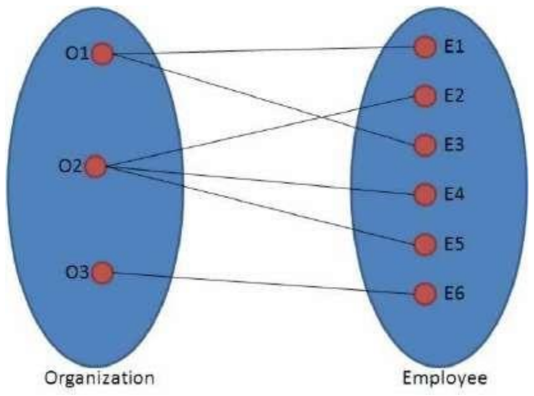
One employee is assigned with only one parking space and one parking space is assigned to only one employee. Hence it is a 1:1 relationship and cardinality is One-To-One (1:1)

In ER modeling, this can be mentioned using notations as given below



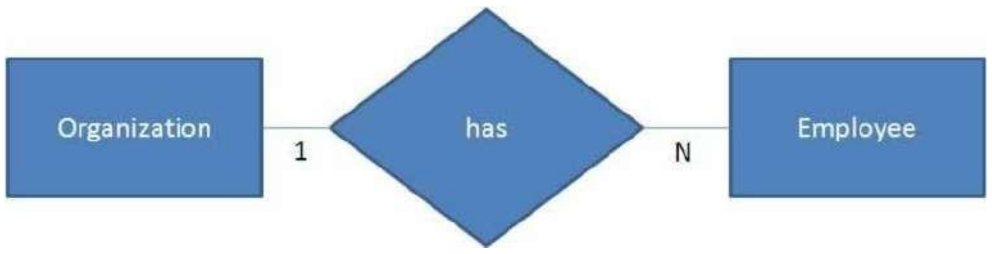
Example for Cardinality – One-to-Many (1:N)

Organization has employees



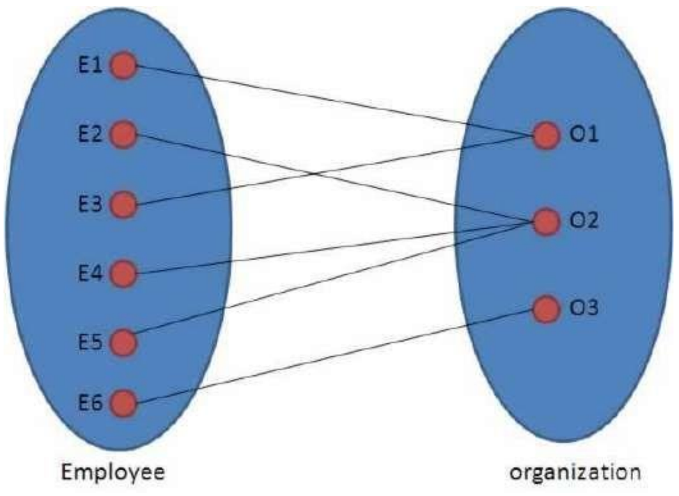
One organization can have many employees , but one employee works in only one organization. Hence it is a 1:N relationship and cardinality is One-To-Many (1:N)

In ER modeling, this can be mentioned using notations as given below



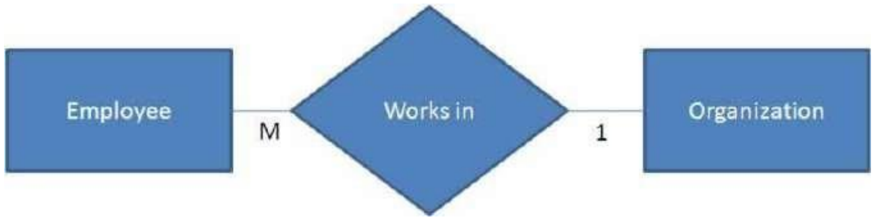
Example for Cardinality – Many-to-One (M :1)

It is the reverse of the One to Many relationship. employee works in organization



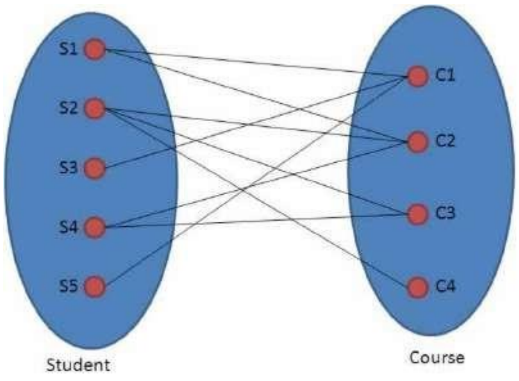
One employee works in only one organization But one organization can have many employees. Hence it is a M:1 relationship and cardinality is Many-to-One (M :1)

In ER modeling, this can be mentioned using notations as given below.



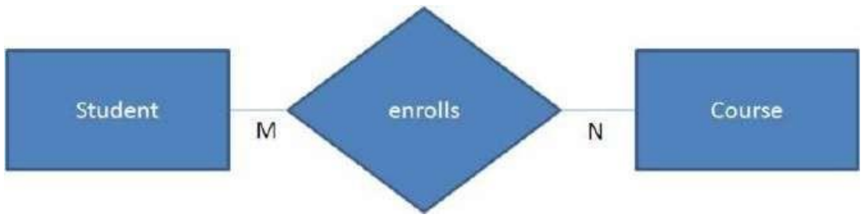
Cardinality – Many-to-Many (M:N)

Students enrolls for courses



One student can enroll for many courses and one course can be enrolled by many students. Hence it is a M:N relationship and cardinality is Many-to-Many (M:N)

In ER modeling, this can be mentioned using notations as given below



In SQL, keys are fundamental components of relational databases that serve to uniquely identify rows within a table and establish relationships between tables, ensuring data integrity and efficient data retrieval.

Types of Keys in SQL:

- **Primary Key:** A column or set of columns that uniquely identifies each row in a table. It must contain unique values and cannot contain NULL values. A table can have only one primary key.

```
CREATE TABLE Customers
(
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(255)
 );
```