



SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

AN AUTONOMOUS INSTITUTION



Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

UNIT 1

Purpose of Database System -- Views of data – Data models, Database Management system - Three-schema architecture of DBMS, Components of DBMS. Entity –Relationship Model - Conceptual data modeling - motivation, entities, entity types, attributes, relationships, relationship types, E/R diagram notations, Examples

PURPOSE OF DATABASE SYSTEM

The 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.

A file processing system has a number of major disadvantages.

1.Data redundancy and inconsistency:

In file processing, every user group maintains its own files for handling its data processing applications.

Example:

Consider the UNIVERSITY database. Here, two groups of users might be the course registration personnel and the accounting office. The accounting office also keeps data on registration and related billing information, whereas the registration office keeps track of student courses and grades. Storing the same data multiple times is called data redundancy. This redundancy leads to several problems.

- Need to perform a single logical update multiple times.
- Storage space is wasted.
- Files that represent the same data may become inconsistent.

Data inconsistency is the various copies of the same data may no longer Agree.

Example:

One user group may enter a student's birth date erroneously as JAN-19-1984, whereas the other user groups may enter the correct value of JAN-29-1984.

2.Difficulty in accessing data

File processing environments do not allow needed data to be retrieved in a convenient and efficient manner.

Example:

Suppose that one of the bank officers needs to find out the names of all customers who live within a particular area. The bank officer has, now two choices: either obtain the list of all customers and extract the needed information manually or ask a system 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 officer needs to trim that list to include only those customers who have an account balance of \$10,000 or more. A program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory.

3. 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.

4. Integrity problems

The data values stored in the database must satisfy certain types of consistency constraints. Example:

The balance of certain types of bank accounts may never fall below a prescribed amount. Developers enforce these constraints in the system by adding appropriate code in the various application programs.

5. Atomicity problems

Atomic means the transaction must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file processing system.

Example:

Consider a program to transfer \$50 from account A to account B. If a system failure occurs during the execution of the program, it is possible that the \$50 was removed from account A but was not credited to account B, resulting in an inconsistent database state.

6. Concurrent access anomalies

For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

Example: When several reservation clerks try to assign a seat on an airline flight, the system should ensure that each seat can be accessed by only one clerk at a time for assignment to a passenger.

7. Security problems

Enforcing security constraints to the file processing system is difficult.

VIEWS OF DATA

A major purpose of a database system is to provide users with an abstract view of the data. i.e. the system hides certain details of how the data are stored and maintained.

Views have several other benefits.

- Views provide a level of security. Views can be setup to exclude data that some users should not see.
- Views provide a mechanism to customize the appearance of the database.
- A view can present a consistent, unchanging picture of the structure of the database, even if the underlying database is changed.

The ANSI / SPARC architecture defines three levels of data abstraction.

• External level / logical level

•Conceptual level

•Internal level / physical level

The objectives of the three level architecture are to separate each user's view of the database from the way the database is physically represented.

External level

The users' view of the database External level describes that part of the database that is relevant to each user.

The external level consists of a number of different external views of the database. Each user has a view of the 'real world' represented in a form that is familiar for that user. The external view includes only those entities, attributes, and relationships in the real world that the user is interested in.

The use of external models has some very major advantages,

- Makes application programming much easier.
- Simplifies the database designer's task.
- Helps in ensuring the database security.

Conceptual level

The community view of the database conceptual level describes what data is stored in the database and the relationships among the data.

The middle level in the three level architecture is the conceptual level. This level contains the logical structure of the entire database as seen by the DBA. It is a complete view of the data requirements of the organization that is independent of any storage considerations.

The conceptual level represents:

- All entities, their attributes and their relationships
- The constraints on the data
- Semantic information about the data
- Security and integrity information.

The conceptual level supports each external view. However, this level must not contain any storage dependent details. For instance, the description of an entity should contain only data types of attributes and their length, but not any storage consideration such as the number of bytes occupied

Internal level

The physical representation of the database on the computer Internal level describes how the data is stored in the database.

The internal level covers the physical implementation of the database to achieve optimal runtime performance and storage space utilization. It covers the data structures and file organizations used to store data on storage devices. The internal level is concerned with

- Storage space allocation for data and indexes.
- Record descriptions for storage
- Record placement.

- Data compression and data encryption techniques.
- Below the internal level there is a physical level that may be managed by the operating system under the direction of the DBMS

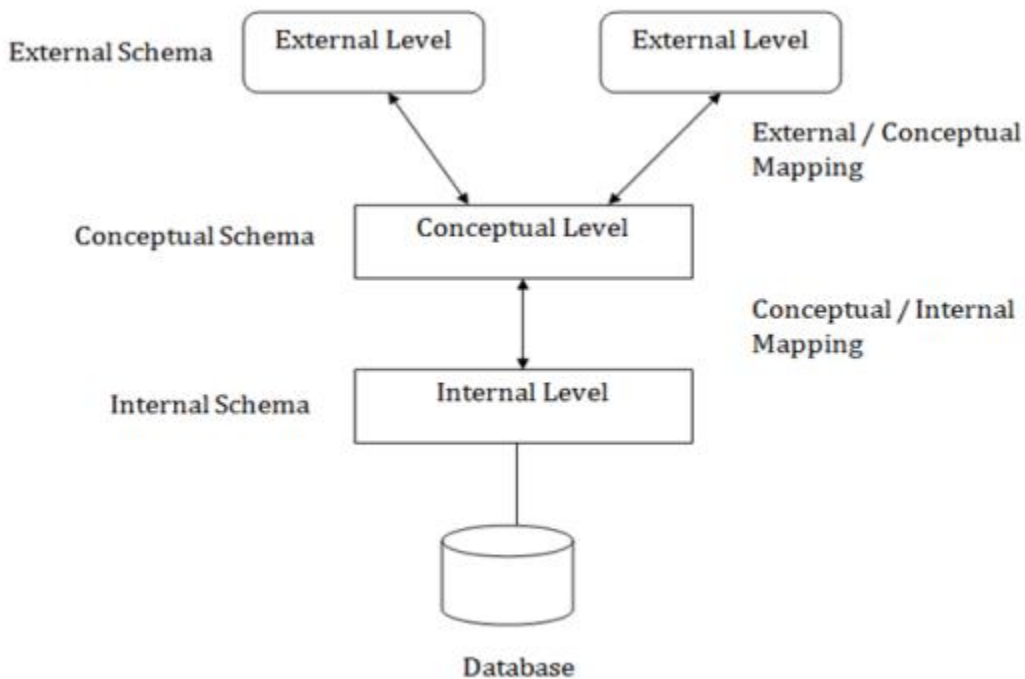
Physical level

- The physical level below the DBMS consists of items only the operating system knows such as exactly how the sequencing is implemented and whether the fields of internal records are stored as contiguous bytes on the disk.

Instances and Schemas

Similar to types and variables in programming languages which we already know, Schema is the logical structure of the database (e.g., the database consists of information about a set of customers and accounts and the relationship between them) analogous to type information of a variable in a program.

Physical schema: database design at the physical level
 Logical schema: database design at the logical level



DATA MODELS

The data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. A data model provides a way to describe the design of a database at the physical, logical and view level.

The purpose of a data model is to represent data and to make the data understandable. According to the types of concepts used to describe the database structure, there are three data models:

1. An external data model, to represent each user's view of the organization.
2. A conceptual data model, to represent the logical view that is DBMS independent
3. An internal data model, to represent the conceptual schema in such a way that it

can be understood by the DBMS.

Categories of data model:

1. Record-based data models

2. Object-based data models

3. Physical-data models.

The first two are used to describe data at the conceptual and external levels, the latter is used to describe data at the internal level.

1. Record -Based data models

In a record-based model, the database consists of a number of fixed format records possibly of differing types. Each record type defines a fixed number of fields, each typically of a fixed length.

There are three types of record-based logical data model.

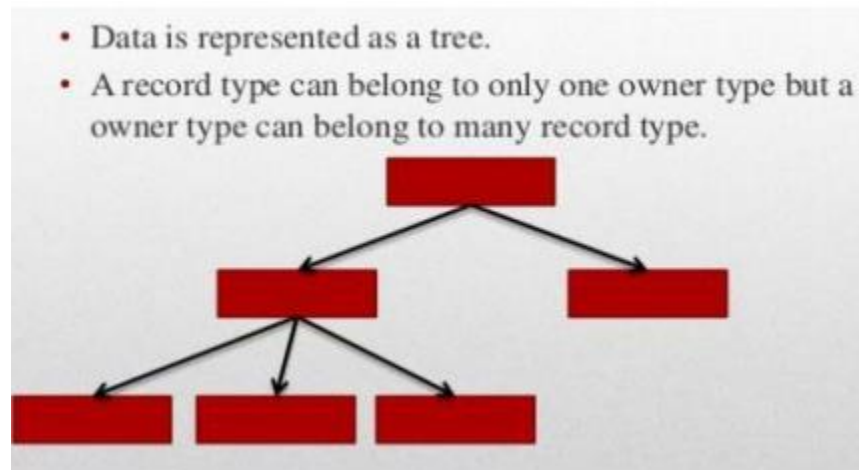
- **Hierarchical data model.**

- **Network data model**

- **Relational data model**

Hierarchical data model

In the hierarchical model, data is represented as collections of records and relationships are represented by sets. The hierarchical model allows a node to have only one parent. A hierarchical model can be represented as a tree graph, with records appearing as nodes, also called segments, and sets as edges.

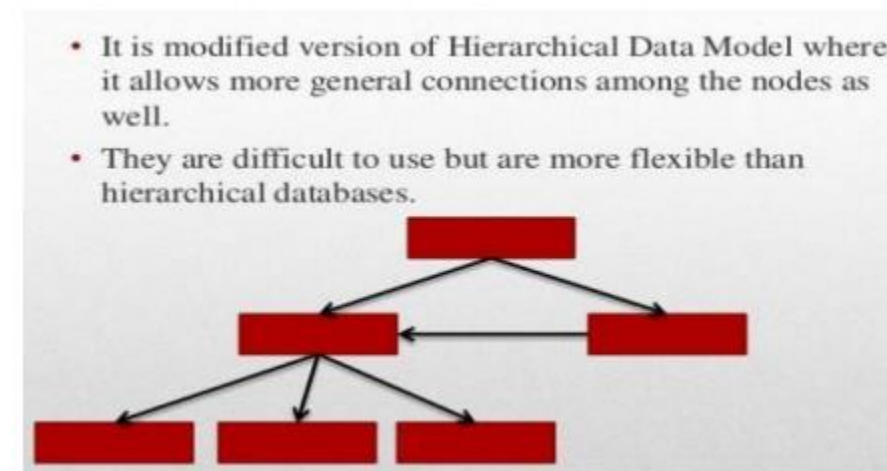


Network data model

In the network model, data is represented as collections of records and relationships are represented by sets. Each set is composed of at least two record types:

- An owner record that is equivalent to the hierarchical model's parent
- A member record that is equivalent to the hierarchical model's child

A set represents a 1:M relationship between the owner and the member.



Relational data model:

The relational data model is based on the concept of mathematical relations. Relational model stores data in the form of a table. Each table corresponds to an entity, and each row represents an instance of that entity. Tables, also called relations are related to each other

through the sharing of a common entity characteristic.

Example

Relational DBMS DB2, oracle, MS SQLserver.

- It is a lower level model that uses a collection of tables to represent both data and relationships among those data.
- Each table has multiple columns, depending on the number of attributes, and each column has a unique name.

sid	sname	Standard
A-101	Ramesh	11
A-102	Kriti	10
A-103	Laxmi	12

2. Object -Based Data Models

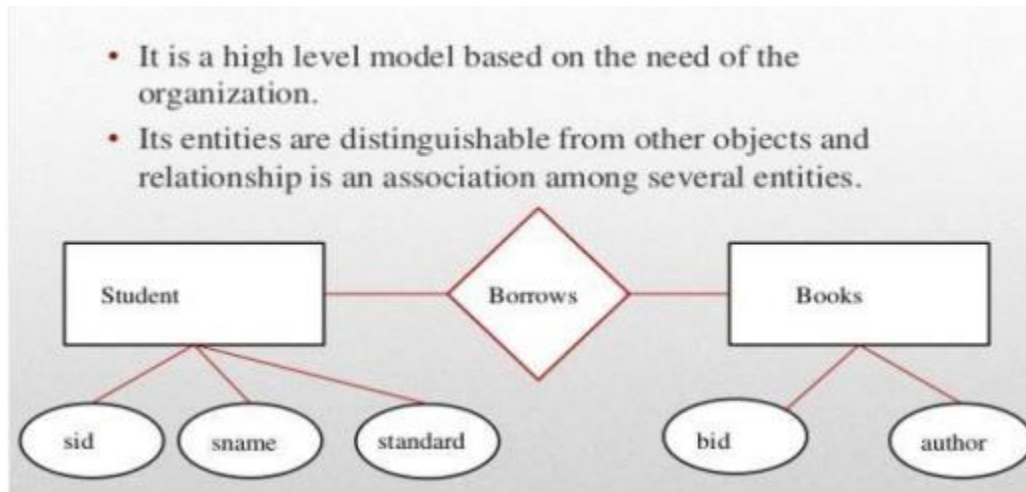
Object-based data models use concepts such as entities, attributes, and relationships. An entity is a distinct object in the organization that is to be represented in the database. An attribute is a property that describes some aspect of the object, and a relationship is an association between entities. Common types of object-based data model are:

- **Entity -Relationship model**
- **Object -oriented model**
- **Semantic model**

Entity Relationship Model:

The ER model is based on the following components:

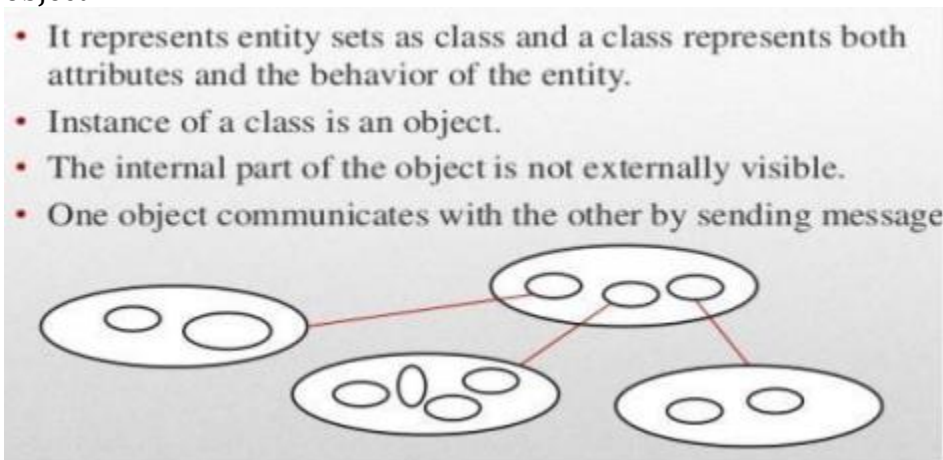
- Entity: An entity was defined as anything about which data are to be collected and stored. Each row in the relational table is known as an entity instance or entity occurrence in the ER model. Each entity is described by a set of attributes that describes particular characteristics of the entity.



Object oriented model:

In the object-oriented data model (OODM) both data and their relationships are contained in a single structure known as an object. An object is described by its factual content. An object includes information about relationships between the facts within the object, as well as information about its relationships with other objects. Therefore, the facts within the object are given greater meaning. The OODM is said to be a semantic data model because semantic indicates meaning. The OO data model is based on the following components:

An object is an abstraction of a real-world entity. Attributes describe the properties of an object.



DATABASE SYSTEM ARCHITECTURE

Transaction Management

A transaction is a collection of operations that performs a single logical function in a database application. Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g. power failures and operating system crashes) and transaction failures. Concurrency-control manager

controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Storage Management

- A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

- The storage manager is responsible for the following tasks:

- Interaction with the file manager

- Efficient storing, retrieving, and Storage Management

- A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

- The storage manager is responsible for the following tasks:

- Interaction with the file manager

- Efficient storing, retrieving, and updating of data

Database Administrator

- Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs:

- Schema definition

- Storage structure and access method definition

- Schema and physical organization modification

- Granting user authority to access the database

- Specifying integrity constraints

- Acting as liaison with users

- Monitoring performance and responding to changes in requirements

Database Users

Users are differentiated by the way they expect to interact with the system.

- **Application programmers:** interact with system through DML calls.

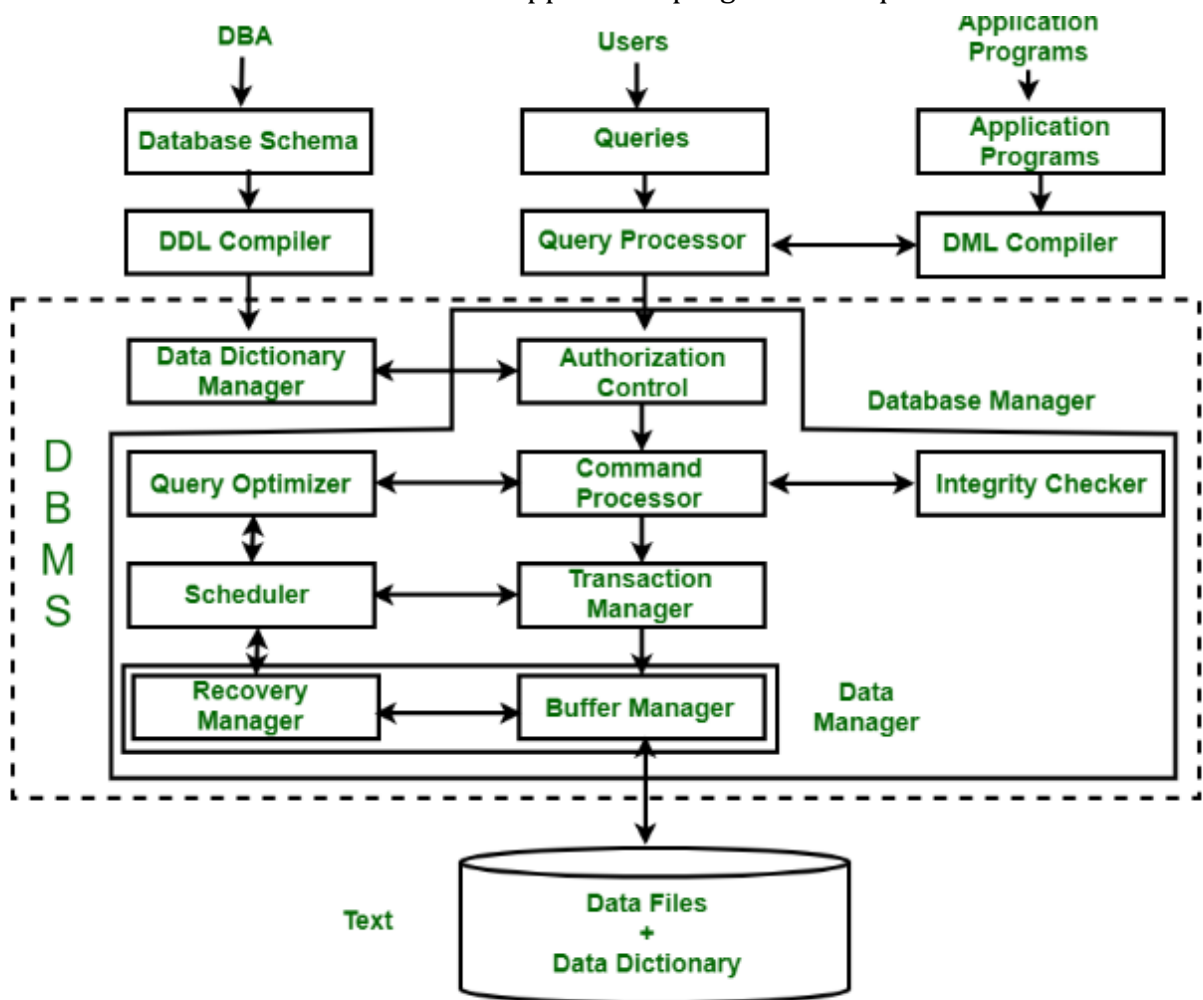
- **Sophisticated users** – form requests in a database query language
- **Specialized users** – write specialized database applications that do not fit into the traditional data processing framework
- **Naive users** – invoke one of the permanent application programs that have been written previously

File manager

manages allocation of disk space and data structures used to represent information on disk.

Database manager

The interface between low level data and application programs and queries.



Architecture of DBMS

Query processor

translates statements in a query language into low-level instructions the database manager understands. (May also attempt to find an equivalent but more efficient form.)

DML precompiler

converts DML statements embedded in an application program to normal procedure calls

in a host language. The precompiler interacts with the query processor.

DDL compiler

converts DDL statements to a set of tables containing metadata stored in a data dictionary. In addition, several data structures are required for physical system implementation:

Data files: store the database itself.

Data dictionary: stores information about the structure of the database. It is used heavily. Great emphasis should be placed on developing a good design and efficient implementation of the dictionary.

Indices: provide fast access to data items holding particular values.

ENTITY RELATIONSHIP MODEL

The entity relationship (ER) data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database. The E-R data model is one of several semantic data models.

The semantic aspect of the model lies in its representation of the meaning of the data. The E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

The ERDs represent three main components: entities, attributes and relationships. Entity sets:

An entity is a thing or object in the real world that is distinguishable from all other objects. Example:

Each person in an enterprise is entity.

An entity has a set of properties, and the values for some set of properties may uniquely identify an entity.

Example:

A person may have a person-id which would uniquely identify one particular person whose value uniquely identifies that person.

An entity may be concrete, such as a person or a book, or it may be abstract, such as a loan, a holiday, or a concept. An entity set is a set of entities of the same type that share the same properties, or attributes.

Example:

Relationship sets:

A relationship is an association among several entities.

Example:

A relationship that associates customer Smith with loan L-16, specifies that Smith is a customer with loan number L-16.

A relationship set is a set of relationships of the same type.

The number of entity sets that participate in a relationship set is also the degree of the relationship set.

A unary relationship exists when an association is maintained within a single entity. Attributes:

For each attribute, there is a set of permitted values, called the domain, or value set, of that attribute. Example:

The domain of attribute customer name might be the set of all text strings of a certain length. An attribute of an entity set is a function that maps from the entity set into a domain.

An attribute can be characterized by the following attribute types:

- Simple and composite attributes.
- Single valued and multi valued attributes.
- Derived attribute.

Simple attribute (atomic attributes)

An attribute composed of a single component with an independent existence is called simple attribute.

Simple attributes cannot be further subdivided into smaller components.

An attribute composed of multiple components, each with an independent existence is called composite attribute.

Example:

The address attribute of the branch entity can be subdivided into street, city, and postcode attributes.

Single-valued Attributes:

An attribute that holds a single value for each occurrence of an entity type is called single valued attribute.

Example:

Each occurrence of the Branch entity type has a single value for the branch number (branch No) attribute (for example B003).

Multi-valued Attribute

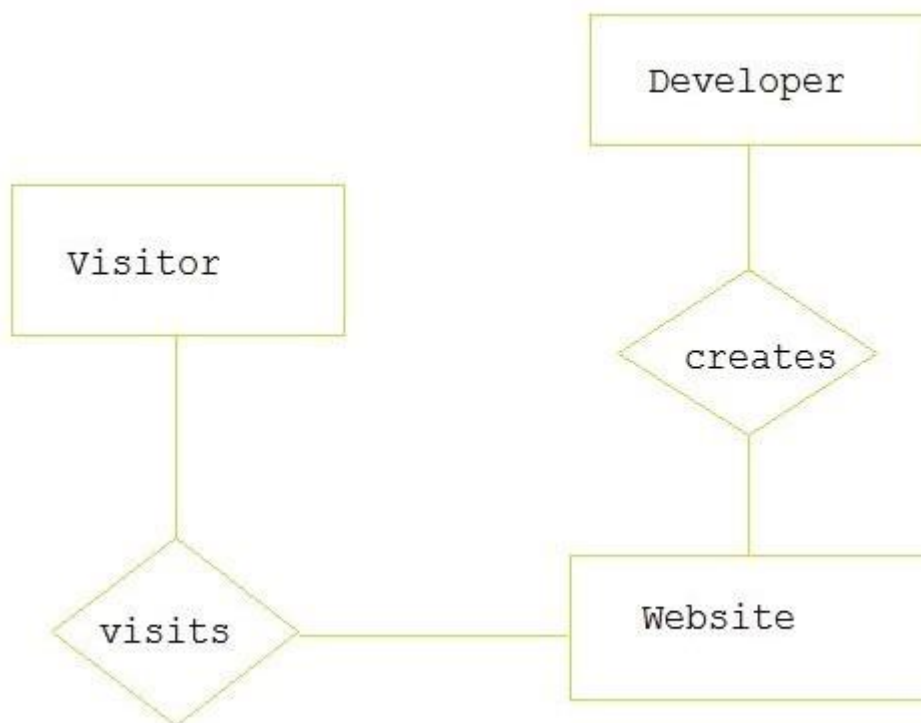
An attribute that holds multiple values for each occurrence of an entity type is called multi-valued attribute.

Example:

Each occurrence of the Branch entity type can have multiple values for the telNo attribute (for example, branch number B003 has telephone numbers 0141-339-2178 and 0141-339-4439).

Derived attributes

An attribute that represents a value that is derivable from the value of a related attribute or set of attributes, not necessarily in the same entity type is called derived attributes.



Here in this ER diagram the entities

are 1. Visitor

2. Website

3. Developer



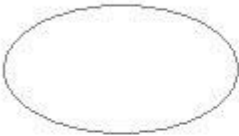

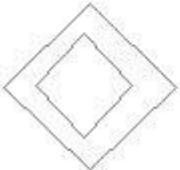

r

Relationships

are 1. visits

2. creates

E-R DIAGRAM REPRESENTATIONS

	represents	
	_____	Entity
	_____	relationship
	_____	attribute
	_____	weak entity
	_____	weak entity relationship
	_____	Multivalued attribute

Keys:

A super key of an entity set is a set of one or more attributes whose values uniquely determine each entity.

A candidate key of an entity set is a minimal super key.

–social-security is candidate key of customer

–account-number is candidate key of account

Although several candidate keys may exist, one of the candidate keys is selected to be the primary key.

The combination of primary keys of the participating entity sets forms a candidate key of a relationship set.

- **must** consider the mapping cardinality and the semantics of the relationship set when selecting the primary key.

– (social-security, account-number) is the primary key of depositor

E-R Diagram Components

- Rectangles represent entity sets.
- Ellipses represent attributes.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Double ellipses represent multivalued attributes.
- Dashed ellipses denote derived attributes.
- Primary key attributes are underlined.

Weak Entity Set
An entity set that does not have a primary key is referred to as a weak entity set. The existence of a weak entity set depends on the existence of a strong entity set; it must relate to the strong set via a one-to-many relationship set. The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set. The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set's existence depends, plus the weak entity set's discriminator. A weak entity set is depicted by double rectangles

Specialization

This is a Top-down design process designate subgroupings within an entity set that are distinctive from other entities in the set.

These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set. Depicted by a triangle component labeled ISA (i.e., savings-account "is an" account

Generalization:

A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.

Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.

Attribute Inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

Design Constraints on Generalization:

Constraint on which entities can be members of a given lower-level entity set.

- condition-defined

- user-defined

-Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.

- disjoint

- overlapping

-Completeness constraint – specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.

- total

- partial Joins in Aggregation

- Treat relationship as an abstract entity.

- Allows relationships between relationships.

- Abstraction of relationship into new entity.

-Without introducing redundancy, the following diagram represents that:

- A customer takes out a loan

- An employee may be a loan officer for a customer-loan pair