DBMS: Basic Concepts

What is data:

Data is the known facts or figures that have implicit meaning. It can also be defined as it is the representation of facts ,concepts or instruction in a formal manner, which is suitable for understanding and processing. Data can be represented in alphabets(A-Z, a-z),in digits(0-9) and using special characters(+,-.#,$, etc)
e.g: 25, “ajit” etc.

Information:

Information is the processed data on which decisions and actions are based. Information can be defined as the organized and classified data to provide meaningful values.

Eg: “The age of Ravi is 25”

File:
File is a collection of related data stored in secondary memory.

File Oriented approach:
The traditional file oriented approach to information processing has for each application a separate master file and its own set of personal file. In file oriented approach the program dependent on the files and files become dependent on the files and files become dependents upon the programs

Disadvantages of file oriented approach:

1) Data redundancy and inconsistency:
The same information may be written in several files. This redundancy leads to higher storage and access cost. It may lead data inconsistency that is the various copies of the same data may longer agree for example a changed customer address may be reflected in single file but not else where in the system.

2) Difficulty in accessing data :
The conventional file processing system do not allow data to retrieved in a convenient and efficient manner according to user choice.

3) Data isolation :
Because data are scattered in various file and files may be in different formats with new application programs to retrieve the appropriate data is difficult. **4) Integrity Problems:**
Developers enforce data validation in the system by adding appropriate code in the various application program. How ever when new constraints are added, it is difficult to change the programs to enforce them.

5) Atomicity:

It is difficult to ensure atomicity in a file processing system when transaction failure occurs due to power failure, networking problems etc.
(atomicity: either all operations of the transaction are reflected properly in the database or non are)

6) Concurrent access:

In the file processing system it is not possible to access a same file for transaction at same time

7) Security problems:

There is no security provided in file processing system to secure the data from unauthorized user access.

Database:

A database is organized collection of related data of an organization stored in formatted way which is shared by multiple users.
The main feature of data in a database are:

1. It must be well organized
2. it is related
3. It is accessible in a logical order without any difficulty
4. It is stored only once

for example:
consider the roll no, name, address of a student stored in a student file. It is collection of related data with an implicit meaning.
Data in the database may be persistent, integrated and shared.

Persistent:
If data is removed from database due to some explicit request from user to remove. **Integrated:**
A database can be a collection of data from different files and when any redundancy among those files are removed from database is said to be integrated data. **Sharing Data:**
The data stored in the database can be shared by multiple users simultaneously with out affecting the correctness of data.

Why Database:

In order to overcome the limitation of a file system, a new approach was required. Hence a database approach emerged. A database is a persistent collection of logically related data. The initial attempts were to provide a centralized collection of data. A database has a self describing nature. It contains not only the data sharing and integration of data of an organization in a single database.

A small database can be handled manually but for a large database and having multiple users it is difficult to maintain it, In that case a computerized database is useful. The advantages of database system over traditional, paper based methods of record keeping are:

- **compactness**:
  No need for large amount of paper files
- **speed**:
  The machine can retrieve and modify the data more faster way then human being
- **Less drudgery**: Much of the maintenance of files by hand is eliminated ● **Accuracy:** Accurate,up-to-date information is fetched as per requirement of the user at any time.

Database Management System (DBMS):
A database management system consists of collection of related data and refers to a set of programs for defining, creation, maintenance and manipulation of a database.

Function of DBMS:

1. **Defining database schema**: it must give facility for defining the database structure also specifies access rights to authorized users.
2. **Manipulation of the database:** The dbms must have functions like insertion of record into database updation of data, deletion of data, retrieval of data 3. **Sharing of database:**The DBMS must share data items for multiple users by maintaining consistency of data.
4. **Protection of database:** It must protect the database against unauthorized users. 5. **Database recovery:** If for any reason the system fails DBMS must facilitate data base recovery.

Advantages of dbms:

Reduction of redundancies:

Centralized control of data by the DBA avoids unnecessary duplication of data and effectively reduces the total amount of data storage required avoiding duplication in the elimination of the inconsistencies that tend to be present in redundant data files.

Sharing of data:

A database allows the sharing of data under its control by any number of application programs or users.

Data Integrity:

Data integrity means that the data contained in the database is both accurate and consistent. Therefore data values being entered for storage could be checked to ensure that they fall with in a specified range and are of the correct format.

Data Security:

The DBA who has the ultimate responsibility for the data in the dbms can ensure that proper access procedures are followed including proper authentication schemas for access to the DBS and additional check before permitting access to sensitive data.

Conflict resolution:

DBA resolve the conflict on requirements of various user and applications. The DBA chooses the best file structure and access method to get optional performance for the application.

Data Independence

Data independence is usually considered from two points of views; physically data independence and logical data independence.
Physical data Independence allows changes in the physical storage devices or organization of the files to be made without requiring changes in the conceptual view or any of the external views and hence in the application programs using the data base.
Logical data independence indicates that the conceptual schema can be changed without affecting the existing external schema or any application program.

Disadvantage of DBMS:
1. DBMS software and hardware (networking installation) cost is high 2. The processing overhead by the dbms for implementation of security, integrity and sharing of the data.
3. centralized database control
4. Setup of the database system requires more knowledge, money, skills, and time. 5. The complexity of the database may result in poor performance.

<u>Database Basics:</u>

Data item:
The data item is also called as field in data processing and is the smallest unit of data that has meaning to its users.
Eg: “e101”,”sumit”

Entities and attributes:

An entity is a thing or object in the real world that is distinguishable from all other objects
Eg:
Bank,employee,student
Attributes are properties are properties of an entity.
Eg:
Empcode,ename,rolno,name

Logical data and physical data :

Logical data are the data for the table created by user in primary memory.
Physical data refers to the data stored in the secondary memory.

Schema and sub-schema :

A schema is a logical data base description and is drawn as a chart of the types of data that are used . It gives the names of the entities and attributes and specify the relationships between them.

A database schema includes such information as :

- Characteristics of data items such as entities and attributes .
- Logical structures and relationships among these data items .
- Format for storage representation.
- Integrity parameters such as physical authorization and back up policies.

A *subschema* is derived schema derived from existing schema as per the user requirement. There may be more then one subschema create for a single conceptual schema.

Three level architecture of DBMS

A database management system that provides three level of data is said to follow three level architecture .

- External level
- Conceptual level
- Internal level

External level :

The external level is at the highest level of database abstraction . At this level, there will be many views define for different users requirement. A view will describe only a subset of the database. Any number of user views may exist for a given global or subschema.

for example , each student has different view of the time table. the view of a student of Btech (CSE) is different from the view of the student of Btech(ECE).Thus this level of abstraction is concerned with different categories of users.
Each external view is described by means of a schema called schema or schema.

Conceptual level :
At this level of database abstraction all the database entities and the relationships among them are included . One conceptual view represents the entire database . This conceptual

view is defined by the conceptual schema.

The conceptual schema hides the details of physical storage structures and concentrate on describing entities , data types, relationships, user operations and constraints.

It describes all the records and relationships included in the conceptual view . There is only one conceptual schema per database . It includes feature that specify the checks to relation data consistency and integrity.

Internal level :

It is the lowest level of abstraction closest to the physical storage method used . It indicates how the data will be stored and describes the data structures and access methods to be used by the database . The internal view is expressed by internal schema. The following aspects are considered at this level:

1. Storage allocation e.g: B-tree,hashing
2. access paths eg. specification of primary and secondary keys,indexes etc 3. Miscellaneous eg. Data compression and encryption techniques,optimization of the internal structures.

Database users :

Naive users :

Users who need not be aware of the presence of the database system or any other system supporting their usage are considered naïve users . A user of an automatic teller machine falls on this category.

Online users :

These are users who may communicate with the database directly via an online terminal or indirectly via a user interface and application program. These users are aware of the database system and also know the data manipulation language system.

Application programmers :

Professional programmers who are responsible for developing application programs or user interfaces utilized by the naïve and online user falls into this category.

Database Administration :

A person who has central control over the system is called database administrator .
The function of DBA are :

1. creation and modification of conceptual Schema definition
2. Implementation of storage structure and access method.
3. schema and physical organization modifications .
4. granting of authorization for data access.
5. Integrity constraints specification.
6. Execute immediate recovery procedure in case of failures

7. ensure physical security to database

Database language :

1) Data definition language(DDL) :

DDL is used to define database objects .The conceptual schema is specified by a set of definitions expressed by this language. It also give some details about how to implement this schema in the physical devices used to store the data. This definition includes all the entity sets and their associated attributes and their relation ships. The result of DDL statements will be a set of tables that are stored in special file called data dictionary.

2) Data manipulation language(DML) :

A DML is a language that enables users to access or manipulate data stored in the database. Data manipulation involves retrieval of data from the database, insertion of new data into the database and deletion of data or modification of existing data.

There are basically two types of DML:

- **procedural**: Which requires a user to specify what data is needed and how to get it.
- **non-Procedural:** which requires a user to specify what data is needed with out specifying how to get it.

3) Data control language(DCL):

This language enables user to grant authorization and canceling authorization of database objects.

Elements of DBMS:

DML pre-compiler:

It converts DML statement embedded in an application program to normal procedure calls in the host language. The pre-complier must interact with the query processor in order to generate the appropriate code.

DDL compiler:

The DDL compiler converts the data definition statements into a set of tables. These tables contains information concerning the database and are in a form that can be used by other components of the dbms.

File manager:

File manager manages the allocation of space on disk storage and the data structure used to represent information stored on disk.

Database manager:

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

The responsibilities of database manager are:

1. **Interaction with file manager**: The data is stored on the disk using the file system which is provided by operating system. The database manager translate the the different DML statements into low-level file system commands. so The database manager is responsible for the actual storing,retrieving and updating of data in the database.
2. **Integrity enforcement:**The data values stored in the database must satisfy certain constraints(eg: the age of a person can't be less then zero).These constraints are specified by DBA. Data manager checks the constraints and if it satisfies then it stores the data in the database.
3. **Security enforcement:**Data manager checks the security measures for database from unauthorized users.
4. **Backup and recovery:**Database manager detects the failures occurs due to different causes (like disk failure, power failure,deadlock,s/w error) and restores the database to original state of the database.
5. **Concurrency control:**When several users access the same database file simultaneously, there may be possibilities of data inconsistency. It is responsible of database manager to control the problems occurs for concurrent transactions.

query processor:

The query processor used to interpret to online user's query and convert it into an efficient series of operations in a form capable of being sent to the data manager for execution. The query processor uses the data dictionary to find the details of data file and using this information it create query plan/access plan to execute the query.

Data Dictionary:

Data dictionary is the table which contains the information about database objects. It contains information like

1. external, conceptual and internal database description
2. description of entities , attributes as well as meaning of data elements
3. synonyms, authorization and security codes
4. database authorization

The data stored in the data dictionary is called *meta data.*

ER-MODEL

Data model:
The data model describes the structure of a database. It is a collection of conceptual tools for describing data, data relationships and consistency constraints and various types of data model such as

1. Object based logical model
2. Record based logical model
3. Physical model

Types of data model:

1. Object based logical model
    a. ER-model
    b. Functional model
    c. Object oriented model
    d. Semantic model
2. Record based logical model
    a. Hierarchical database model
    b. Network model
    c. Relational model
3. Physical model

Entity Relationship Model

The entity-relationship data model perceives the real world as consisting of basic objects, called entities and relationships among these objects. It was developed to facilitate data base design by allowing specification of an enterprise schema which represents the overall logical structure of a data base.

Main features of ER-MODEL:

- Entity relationship model is a high level conceptual model
- It allows us to describe the data involved in a real world enterprise in terms of objects and their relationships.
- It is widely used to develop an initial design of a database
- It provides a set of useful concepts that make it convenient for a developer to move from a baseid set of information to a detailed and description of information that can be easily implemented in a database system
- It describes data as a collection of entities, relationships and attributes.

Basic concepts:

The E-R data model employs three basic notions : entity sets, relationship sets and attributes.

Entity sets:

An entity is a “thing” or “object” in the real world that is distinguishable from all other objects. For example, each person in an enterprise is an entity. An entity has a set properties and the values for some set of properties may uniquely identify an entity.

BOOK is entity and its properties(calles as attributes) bookcode, booktitle, price etc .

An entity set is a set of entities of the same type that share the same properties, or attributes. The set of all persons who are customers at a given bank, for example, can be defined as the entity set customer.

Attributes:

An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set.

Customer is an entity and its attributes are **customerid, custmername, custaddress** etc.

An attribute as used in the E-R model , can be characterized by the following attribute types.

a) **Simple and composite attribute:**

simple attributes are the attributes which can't be divided into sub parts eg: customerid,empno
composite attributes are the attributes which can be divided into subparts. eg: name consisting of first name, middle name, last name
address consisting of city,pincode,state

b) **single-valued and multi-valued attribute:**

The attribute having unique value is single –valued attribute
eg: empno,customerid,regdno etc.
The attribute having more than one value is multi-valued attribute
eg: phone-no, dependent name, vehicle

c) **Derived Attribute:**

The values for this type of attribute can be derived from the values of existing attributes
eg: age which can be derived from (currentdate-birthdate)
experience_in_year can be calculated as (currentdate-joindate)

d) **NULL valued attribute:**

The attribute value which is unknown to user is called NULL valued attribute.

Relationship sets:

A relationship is an association among several entities.

A relationship set is a set of relationships of the same type. Formally, it is a mathematical relation on n>=2 entity sets. If E1,E2…En are entity sets, then a relation ship set R is a subset of

{(e1,e2,…en)|e1Є E1,e2 Є E2..,en Є En}

where (e1,e2,…en) is a relationship.

Consider the two entity sets customer and loan. We define the relationship set borrow to denote the association between customers and the bank loans that the customers have.

Mapping Cardinalities:

Mapping cardinalities or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.

Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets.

For a binary relationship set R between entity sets A and B, the mapping cardinalities must be one of the following:

one to one:

An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

Eg: relationship between college and principal

1 1 1 1

college principal has

One to many:

An entity in A is associated with any number of entities in B. An entity in B is associated with at the most one entity in A.

Eg: Relationship between department and faculty

1 M

Department Works Faculty

in

Many to one:

An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.

M 1

Many –to-many:

Entities in A and B are associated with any number of entities from each other.

M M

More about entities and Relationship:

Recursive relationships:

When the same entity type participates more than once in a relationship type in different roles, the relationship types are called recursive relationships.

Participation constraints:

The participation constraints specify whether the existence of any entity depends on its being related to another entity via the relationship. There are two types of participation constraints

Total :

.When all the entities from an entity set participate in a relationship type , is called total participation. For example, the participation of the entity set student on the relationship set must 'opts' is said to be total because every student enrolled must opt for a course.

Partial:

When it is not necessary for all the entities from an entity set to participate ion a relationship type, it is called participation. For example, the participation of the entity set student in 'represents' is partial, since not every student in a class is a class representative.

Weak Entity:

Entity types that do not contain any key attribute, and hence can not be identified independently are called weak entity types. A weak entity can be identified by uniquely only by considering some of its attributes in conjunction with the primary key attribute of another entity, which is called the identifying owner entity.

Generally a partial key is attached to a weak entity type that is used for unique identification of weak entities related to a particular owner type. The following restrictions must hold:

- The owner entity set and the weak entity set must participate in one to may relationship set. This relationship set is called the identifying relationship set of the weak entity set.

- The weak entity set must have total participation in the identifying relationship.

Example:

Consider the entity type dependent related to employee entity, which is used to keep track of the dependents of each employee. The attributes of dependents are : name ,birthrate, sex and relationship. Each employee entity set is said to its own the dependent entities that are related to it. How ever, not that the 'dependent' entity does not exist of its own., it is dependent on the employee entity. In other words we can say that in case an employee leaves the organization all dependents related to without the entity 'employee'. Thus it is a weak entity.

Keys:

Super key:

A super key is a set of one or more attributes that taken collectively, allow us to identify uniquely an entity in the entity set.

For example , customer-id,(cname,customer-id),(cname,telno)

Candidate key:

In a relation R, a candidate key for R is a subset of the set of attributes of R, which have the following properties:

- *Uniqueness:* no two distinct tuples in R have the same values for the candidate key
- *Irreducible:* No proper subset of the candidate key has the uniqueness property that is the candidate key.

Eg: (cname,telno)

Primary key:

The primary key is the candidate key that is chosen by the database designer as the principal means of identifying entities with in an entity set. The remaining candidate keys if any, are called *alternate key*.

ER-DIAGRAM:

The overall logical structure of a database using ER-model graphically with the help of an ER-diagram.

Advanced ER-diagram:

Abstraction is the simplification mechanism used to hide superfluous details of a set of objects. It allows one to concentrate on the properties that are of interest to the application.

There are two main abstraction mechanism used to model information: **Generalization and specialization:**

. *Generalization* is the abstracting process of viewing set of objects as a single general class by concentrating on the general characteristics of the constituent sets while suppressing or ignoring their differences. It is the union of a number of lower-level entity types for the purpose of producing a higher-level entity type. For instance, student is a generalization of graduate or undergraduate, full-time or part-time students. Similarly, employee is generalization of the classes of objects cook, waiter, and cashier. Generalization is an IS_A relationship; therefore, manager IS_AN employee, cook IS_AN employee, waiter IS_AN employee, and so forth.

Specialization is the abstracting process of introducing new characteristics to an existing class of objects to create one or more new classes of objects. This involves taking a higher-level, and using additional characteristics, generating lower-level entities. The lower-level entities also inherits the, characteristics of the higher-level entity. In applying the characteristics size to car we can create a full-size, mid-size, compact or subcompact car. Specialization may be seen as the reverse process of generalization addition specific properties are introduced at a lower level in a hierarchy of objects.

Generalization Specialization

EMPLOYEE(**empno**,name,dob)
FULL_TIME_EMPLOYEE(**empno**,sa la ry)

PART_TIME_EMPLOYEE(**empno**,type)
Faculty(**empno**,degree,intrest)
Staff(**empno**,hour-rate) Teaching (**empno**,stipend)

AGGREGATION

Aggregation is the process of compiling information on an object, there by abstracting a higher level object. In this manner, the entity person is derived by aggregating the characteristics of name, address, ssn. Another form of the aggregation is abstracting a relationship objects and viewing the relationship as an objec

Conversion of ER-diagram to relational database

Conversion of entity sets:

1. For each strong entity type E in the ER diagram, we create a relation R containing all the single attributes of E. The primary key of the relation R will be one of the key attribute of R.

   STUDENT(rollno (primary key),name, address)
   FACULTY(id(primary key),name ,address, salary)
   COURSE(course-id,(primary key),course_name,duration)
   DEPARTMENT(dno(primary key),dname)

2. for each weak entity type W in the ER diagram, we create another relation R that contains all simple attributes of W. If E is an owner entity of W then key attribute of E is also include In R. This key attribute of R is set as a foreign key attribute of R. Now the combination of primary key attribute of owner entity type and partial key of the weak entity type will form the key of the weak entity type

   GUARDIAN((rollno,name) (primary key),address,relationship)

Conversion of relationship sets:

Binary Relationships:

- **One-to-one relationship:**

For each 1:1 relationship type R in the ER-diagram involving two entities E1 and E2 we choose one of entities(say E1) preferably with total participation and add primary key attribute of another E as a foreign key attribute in the table of entity(E1). We will also include all the simple attributes of relationship type R in E1 if any, For example, the department relationship has been extended tp include head-id and attribute of the relationship.

DEPARTMENT(D_NO,D_NAME,HEAD_ID,DATE_FROM)

- **One-to-many relationship:**

For each 1:n relationship type R involving two entities E1 and E2, we identify the entity type (say E1) at the n-side of the relationship type R and include primary key of the entity on the other side of the relation (say E2) as a foreign key attribute in the table of E1. We include all simple attribute(or simple components of a composite attribute of R(if any) in he table E1)

For example:

The works in relationship between the DEPARTMENT and FACULTY. For this relationship choose the entity at N side, i.e, FACULTY and add primary key attribute of another entity DEPARTMENT, ie, DNO as a foreign key attribute in FACULTY.

FACULTY(CONSTAINS WORKS_IN RELATIOSHIP)
(ID,NAME,ADDRESS,BASIC_SAL,DNO)

• **Many-to-many relationship:**

For each m:n relationship type R, we create a new table (say S) to represent R, We also include the primary key attributes of both the participating entity types as a foreign key attribute in s. Any simple attributes of the m:n relationship type(or simple components as a composite attribute) is also included as attributes of S.
For example:
The M:n relationship taught-by between entities COURSE; and FACULTY shod be represented as a new table. The structure of the table will include primary key of COURSE and primary key of FACULTY entities.

TAUGHT-BY(ID (primary key of FACULTY table),course-id (primary key of COURSE table)

• **N-ary relationship:**

For each n-anry relationship type R where n>2, we create a new table S to represent R, We include as foreign key attributes in s the primary keys of the relations that represent the participating entity types. We also include any simple attributes of the n-ary relationship type(or simple components of complete attribute) as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.

LOAN-SANCTION(cusomet-id,loanno,empno,sancdate,loan_amount
)

• **Multi-valued attributes:**

For each multivalued attribute 'A', we create a new relation R that includes an attribute corresponding to plus the primary key attributes k of the relation that represents the entity type or relationship that has as an attribute. The primary key of R is then combination of A and k.

For example, if a STUDENT entity has rollno,name and phone number where phone numer is a multivalued attribute the we will create table PHONE(rollno,phoneno) where primary key is the combination,In the STUDENT table we need not have phone number, instead if can be simply (rollno,name) only.
PHONE(rollno,phoneno)

• **Converting Generalisation /specification hierarchy to tables:** A simple rule for conversion may be to decompose all the specialized entities into table in case they are disjoint, for example, for the figure we can create the two table as:
Account(account_no,name,branch,balance)
Saving account(account-no,intrest)
Current_account(account-no,charges)

Record Based Logical Model

Hierarchical Model:

- A hierarchical database consists of a collection of *records* which are connected to one another through *links.*
- a record is a collection of fields, each of which contains only one data value.
- A link is an association between precisely two records.
- The hierarchical model differs from the network model in that the records are organized as collections of trees rather than as arbitrary graphs.

Tree-Structure Diagrams:

- The schema for a hierarchical database consists of
  - *boxes,* which correspond to record types
  - *lines,* which correspond to links
- Record types are organized in the form of a *rooted tree.*
  - No cycles in the underlying graph.
  - Relationships formed in the graph must be such that only one-to-many or one-to-one relationships exist between a parent and a child.

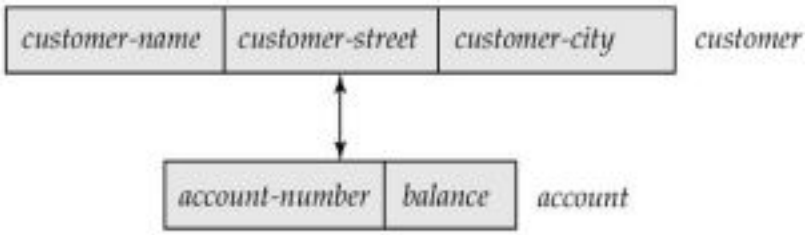
Database schema is represented as a collection of tree-structure diagrams. ・ *single* instance of a database tree
・ The root of this tree is a dummy node
・ The children of that node are actual instances of the appropriate record type

When transforming E-R diagrams to corresponding tree-structure diagrams, we must ensure that the resulting diagrams are in the form of rooted trees.

Single Relationships:

・ Example E-R diagram with two entity sets, *customer* and *account,* related through a binary, one-to-many relationship *depositor.*
・ Corresponding tree-structure diagram has
  - the record type *customer* with three fields: *customer-name, customer street,* and *customer-city.*
  - the record type *account* with two fields: *account-number* and *balance*
  - the link *depositor,* with an arrow pointing to *customer*

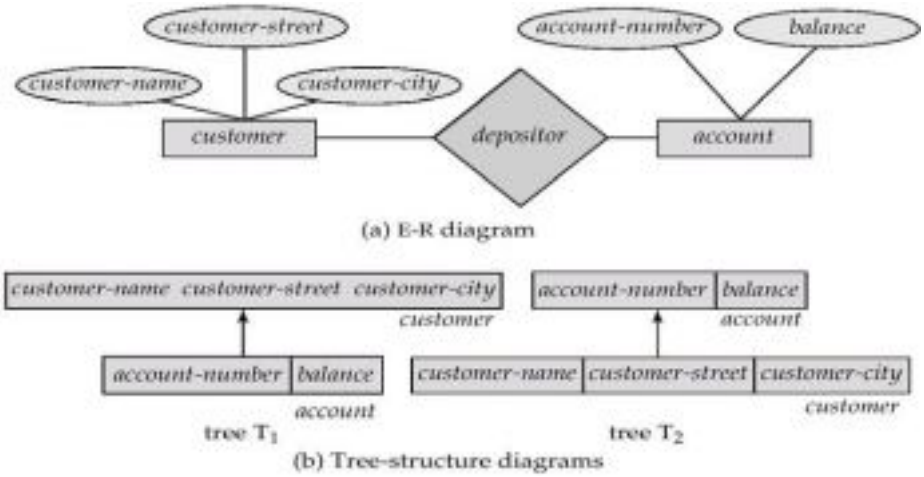
- If the relationship *depositor* is one to one, then the link *depositor* has two arrows.



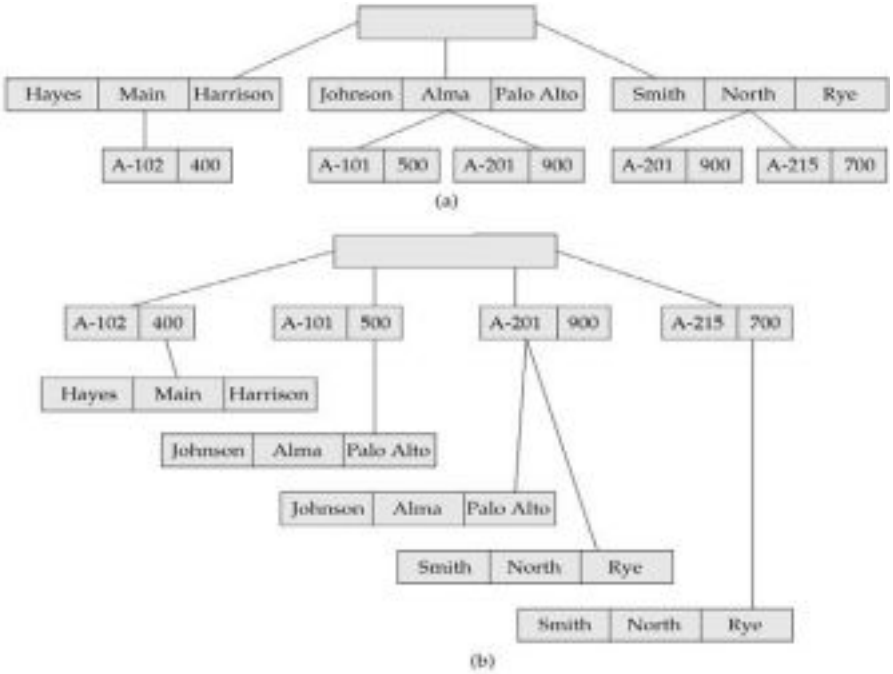
- Only one-to-many and one-to-one relationships can be directly represented in the hierarchical mode.

Transforming Many-To-Many Relationships:

- Must consider the type of queries expected and the degree to which the database schema fits the given E-R diagram.
- In all versions of this transformation, the underlying database tree (or trees) will have replicated records.



- Create two tree-structure diagrams, *T*1, with the root *customer,* and *T*2, with the root *account.*
- In *T*1, create *depositor,* a many-to-one link from *account* to *customer.* - In *T*2, create *account-customer,* a many-to-one link from *customer* to *account.*



Virtual Records:

- For many-to-many relationships, record replication is necessary to preserve the tree-structure organization of the database.
    - Data inconsistency may result when updating takes place
    - Waste of space is unavoidable
- *Virtual record* — contains no data value, only a logical pointer to a particular physical record.
- When a record is to be replicated in several database trees, a single copy of that record is kept in one of the trees and all other records are replaced with a virtual record.
- Let *R* be a record type that is replicated in *T*1, *T*2, . . ., *Tn*. Create a new virtual record type *virtual-R* and replace *R* in each of the *n* – 1 trees with a record of type *virtual-R.*
- Eliminate data replication in the diagram shown on page B.11; create *virtual customer* and *virtual-account.*
- Replace *account* with *virtual-account* in the first tree, and replace *customer* with *virtual-customer* in the second tree.
- Add a dashed line from *virtual-customer* to *customer*, and from *virtual-account* to *account,* to specify the association between a virtual record and its corresponding physical record.

NETWORK MODEL

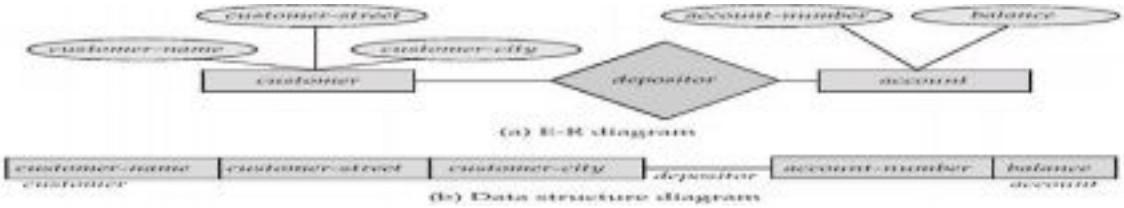
- Data are represented by collections of *records*.
    - similar to an entity in the E-R model
    - Records and their fields are represented as *record type*
- type *customer* = record type *account* = record type *customer-name:*
  string; *account-number:* integer;
  customer-street: string; *balance:* integer;
  customer-city: string;
- end end
- Relationships among data are represented by *links*

- similar to a restricted (binary) form of an E-R relationship
- restrictions on links depend on whether the relationship is many-many, many-to-one, or one-to-one.

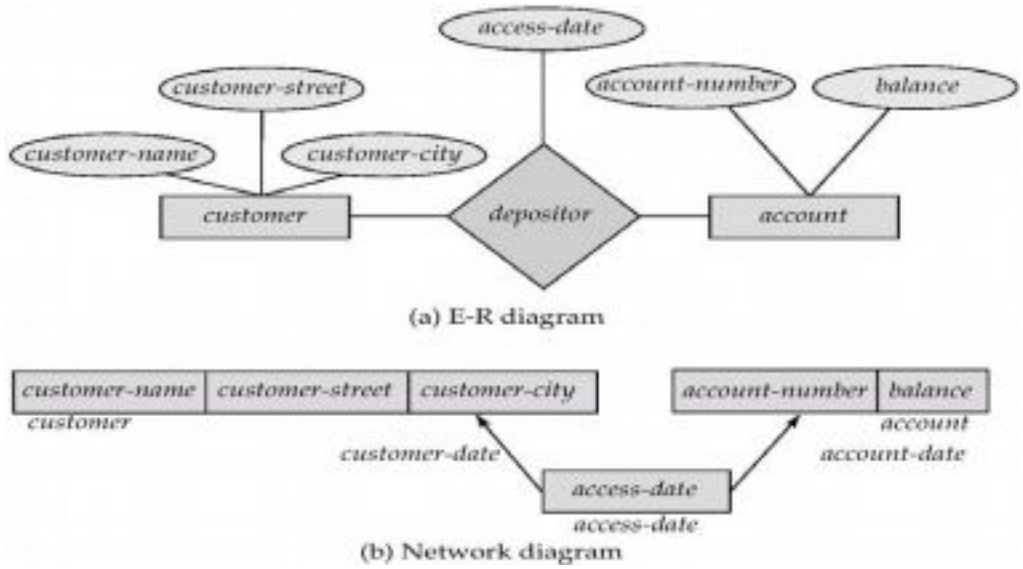
Data-Structure Diagrams:

- Schema representing the design of a network database.
- A data-structure diagram consists of two basic components:
  - Boxes, which correspond to record types.
  - Lines, which correspond to links.
- Specifies the overall logical structure of the database.

For every E-R diagram, there is a corresponding data-structure diagram.

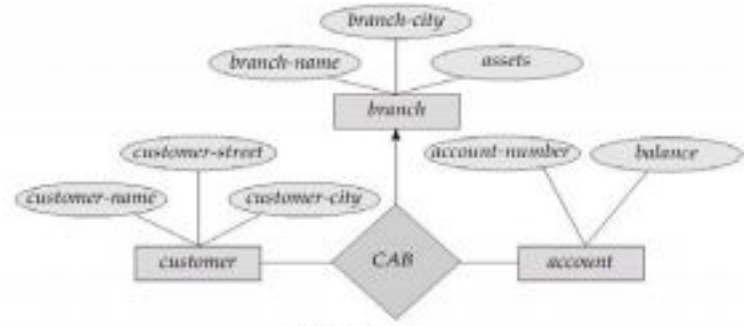


Since a link cannot contain any data value, represent an E-R relationship with attributes with a new record type and links.

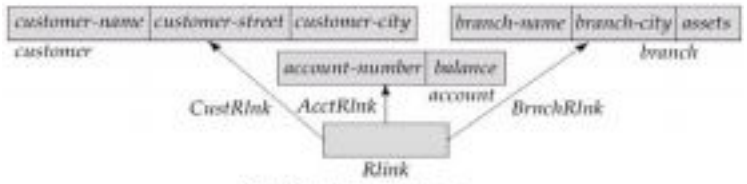


To represent an E-R relationship of degree 3 or higher, connect the participating record types through a new record type that is linked directly to each of the original record types.

1. Replace entity sets *account, customer,* and *branch* with record types *account, customer,* and *branch,* respectively.
2. Create a new record type *Rlink* (referred to as a *dummy* record type).
3. Create the following many-to-one links:
   - *CustRlink* from *Rlink* record type to *customer* record type
   - *AcctRlnk* from *Rlink* record type to *account* record type
   - *BrncRlnk* from *Rlink* record type to *branch* record type



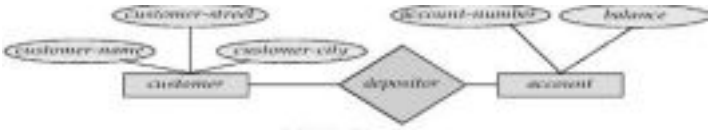
(a) E-R diagram



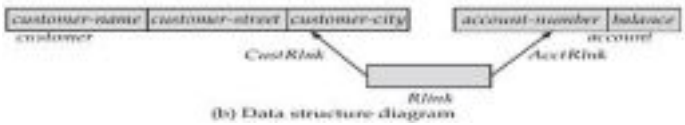
(b) Data structure diagram

The DBTG CODASYL Model:

- All links are treated as many-to-one relationships.
- To model many-to-many relationships, a record type is defined to represent the relationship and two links are used.



(a) E-R diagram



(b) Data structure diagram

DBTG Sets:

- The structure consisting of two record types that are linked together is referred to in the DBTG model as a *DBTG set*
- In each DBTG set, one record type is designated as the *owner,* and the other is designated as the *member*, of the set.
- Each DBTG set can have any number of *set occurrences* (actual instances of linked records).
- Since many-to-many links are disallowed, each set occurrence has precisely one owner, and has zero or more member records.
- No member record of a set can participate in more than one occurrence of the set at any point.
- A member record can participate simultaneously in several set occurrences of *different* DBTG sets.

RELATIONAL MODEL

Relational model is simple model is simple model in which database is represented as a collection of "relations" where each relation is represented by two-dimensional table.

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

The relational model was founded by E.F.Codd of the IBM in 1972.The basic concept in the relational model is that of a relation.

Properties:

- It is column homogeneous. In other words, in any given column of a table, all items are of the same kind.
- Each item is a simple number or a character string. That is a table must be in first normal form.
- All rows of a table are distinct.
- The ordering of rows with in a table is immaterial.

- The column of a table are assigned distinct names and the ordering of these columns in immaterial.

Domain, attributes tuples and relational:

Tuple:

Each row in a table represents a record and is called a tuple .A table containing 'n' attributes in a record is called is called n-tuple.

Attributes:

The name of each column in a table is used to interpret its meaning and is called an attribute.Each table is called a relation.

In the above table, account_number, branch name, balance are the attributes. **Domain:**

A domain is a set of values that can be given to an attributes. So every attribute in a table has a specific domain. Values to these attributes can not be assigned outside their domains.

Relation:

A relation consist of

- **Relational schema**
- **Relation instance**

Relational schema:

A relational schema specifies the relation' name, its attributes and the domain of each attribute. If R is the name of a relation and A1,A2,… and is a list of attributes representing R then R(A1,A2,…,an) is called a relational schema. Each attribute in this relational schema takes a value from some specific domain called domain(Ai).

Example:

PERSON(PERSON_IDinteger,NAME: STRING,AGE:INTEGER,ADDRESS:string)

Total number of attributes in a relation denotes the degree of a relation.since the PERSON relation schemea contains four attributes ,so this relation is of degree 4.

Relation Instance:

A relational instance denoted as r is a collection of tuples for a given relational

schema at a specific point of time.
A relation state r to the relations schema R(A1,A2…,An) also denoted by r® is a set of n-tuples
R{t1,t2,…tm}
Where each n-tuple is an ordered list of n values
T=<v1,v2,….vn>
Where each vi belongs to domain (Ai) or contains null values.
The relation schema is also called 'intension' and the relation state is also called 'extension'.
Eg:

Relation schema for student:
STUDENT(rollno:strinhg,name:string,city:string,age:integer)

Relation instance:
Student:
Rollno Name City Age 101 Sujit Bam 23 102 kunal bbsr 22

Keys:
Super key:
A super key is an attribute or a set of attributes used to identify the records uniquely in a relation.

For example , customer-id,(cname,customer-id),(cname,telno)
Candidate key:
Super keys of a relation can contain extra attributes . candidate keys are minimal super keys. i.e, such a key contains no extraneous attribute. An attribute is called extraneous if even after removing it from the key, makes the remaining attributes still has the properties of a key.

In a relation R, a candidate key for R is a subset of the set of attributes of R, which have the following properties:

- *Uniqueness:* no two distinct tuples in R have the same values for the candidate key
- *Irreducible:* No proper subset of the candidate key has the *uniqueness property that is the candidate key.*
- *A candidate key's values must exist. It can't be null.*
- *The values of a candidate key must be stable. Its value can not change outside the control of the system.*

Eg: (cname,telno)
Primary key:
The primary key is the candidate key that is chosen by the database designer as the principal means of identifying entities with in an entity set. The remaining candidate keys if any are called ***alternate key*.**

RELATIONAL CONSTRAINTS:

There are three types of constraints on relational database that include
- DOMAIN CONSTRAINTS
- KEY CONSTRAINTS
- INTEGRITY CONSTRAINTS

DOMAIN CONSTRAINTS:

It specifies that each attribute in a relation an atomic value from the corresponding domains. The data types associated with commercial RDBMS domains include:

- Standard numeric data types for integer
- Real numbers
- Characters
- Fixed length strings and variable length strings

Thus, domain constraints specifies the condition that we to put on each instance of the relation. So the values that appear in each column must be drawn from the domain associated with that column.

Rollno Name City Age 101 Sujit Bam 23 102 kunal bbsr 22

Key constraints:

This constraints states that the key attribute value in each tuple msut be unique .i.e, no two tuples contain the same value for the key attribute.(null values can allowed)

Emp(empcode,name,address) . here empcode can be unique

Integrity constraints:

There are two types of integrity constraints:
- Entity integrity constraints
- Referential integrity constraints

Entity integrity constraints:

It states that no primary key value can be null and unique. This is because the primary key is used to identify individual tuple in the relation. So we will not be able to identify the records uniquely containing null values for the primary key attributes. This constraint is specified on one individual relation.

Referential integrity constraints:

It states that the tuple in one relation that refers to another relation must refer to an existing tuple in that relation. This constraints is specified on two relations . If a column is declared as foreign key that must be primary key of another table.

Department(deptcode,dname)

Here the deptcode is the primary key.

Emp(*empcode*,name,city,*deptcode*).

Here the deptcode is foreign key.

CODD'S RULES

Rule 1 : The information Rule.

"All information in a relational data base is represented explicitly at the logical level and in exactly one way - by values in tables."

Everything within the database exists in tables and is accessed via table access routines.

Rule 2 : Guaranteed access Rule.

"Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name."

To access any data-item you specify which column within which table it exists, there is no reading of characters 10 to 20 of a 255 byte string.

Rule 3 : Systematic treatment of null values.

"Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type."

If data does not exist or does not apply then a value of NULL is applied, this is understood by the RDBMS as meaning non-applicable data.

Rule 4 : Dynamic on-line catalog based on the relational model.

"The data base description is represented at the logical level in the same way as-ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data."

The Data Dictionary is held within the RDBMS, thus there is no-need for off-line volumes to tell you the structure of the database.

Rule 5 : Comprehensive data sub-language Rule.

"A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all the following items

- Data Definition
- View Definition
- Data Manipulation (Interactive and by program).
- Integrity Constraints
- Authorization.

Every RDBMS should provide a language to allow the user to query the contents of the RDBMS and also manipulate the contents of the RDBMS.

Rule 6 : .View updating Rule

"All views that are theoretically updateable are also updateable by the system."

Not only can the user modify data, but so can the RDBMS when the user is not logged-in. Rule 7 : High-level insert, update and delete.

"The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data."

The user should be able to modify several tables by modifying the view to which they act as base tables.

Rule 8 : Physical data independence.

"Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods."

The user should not be aware of where or upon which media data-files are stored Rule 9 : Logical data independence.

"Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit un-impairment are made to the base tables."

User programs and the user should not be aware of any changes to the structure of the tables (such as the addition of extra columns).

Rule 10 : Integrity independence.

"Integrity constraints specific to a particular relational data base must be definable in the relational data sub-language and storable in the catalog, not in the application programs."

If a column only accepts certain values, then it is the RDBMS which enforces these constraints and not the user program, this means that an invalid value can never be entered into this column, whilst if the constraints were enforced via programs there is always a chance that a buggy program might allow incorrect values into the system.

Rule 11 : Distribution independence.

"A relational DBMS has distribution independence."

The RDBMS may spread across more than one system and across several networks, however to the end-user the tables should appear no different to those that are local.

Rule 12 : Non-subversion Rule.

"If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity Rules and constraints expressed in the higher level relational language (multiple-records-at-a-time)."