MC 5403- ADVANCED DATABASES AND DATA MINING

UNIT I: RELATIONAL MODEL

MC5403- ADBDM UNIT - I : Relational Model

Syllabus:

UNIT I RELATIONALMODEL 9

Data Model – Types of Data Models: – Entity Relationship Model – Relational Data Model – Mapping Entity Relationship Model to Relational Model – Structured Query Language – Database Normalization – Transaction Management.

Table of Contents

Total Pages: 95

1. INTRODUCTION

1.1 Introduction Data

1.1.1 Data and Information

- Data are plain facts.
- The word "data" is plural for "datum."
- Data is nothing but facts and statistics stored or free flowing over a network, generally it's raw and unprocessed.
- When data are processed, organized, structured or presented in a given context so as to make them useful, they are called Information.
- It is not enough to have data (such as statistics on the economy).
- Data themselves are fairly useless, but when these data are interpreted and processed to determine its true meaning, they becomes useful and can be called Information.

For example: When you visit any website, they might store you IP address, that is data, in return they might add a cookie in your browser, marking you that you visited the website, that is data, your name, it's data, your age, it's data.

1.1.2 Database

A Database is a collection of related data organized in a way that data can be easily accessed, managed and updated. Database can be software based or hardware based, with one sole purpose, storing data

What is Database?

- A database is a data structure that stores organized information.
- Most databases contain multiple tables, which may each include several different fields.
  - For example, a company database may include tables for products, employees, and financial records.
  - Each of these tables would have different fields that are relevant to the information stored in the table.

***Definitions* of Database :**

- An organized body of related information.
- In computing, a database can be defined as a structured collection of records or data that is stored in a computer so that a program can consult it to answer queries.

- The records retrieved in answer to queries become information that can be used to make decisions.
- An organized collection of records presented in a standardized format searched by computers. Web Pals, ID Weeks Library's Online Catalog, is a database.
- The periodical indexes available through the library are also databases.
- A collection of data organized for rapid search and retrieval by a computer.

1.1.3 DBMS (or) Database Management System

- A DBMS is a software that allows creation, definition and manipulation of database, allowing users to store, process and analyse data easily.
- DBMS provides us with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the database and a lot more.
- DBMS also provides protection and security to the databases. It also maintains data consistency in case of multiple users.

What is DBMS?

- Collection of interrelated data
- Set of programs to access the data
- DBMS contains information about a particular enterprise
- DBMS provides an environment that is both *convenient* and *efficient* to use.

Here are some examples of popular DBMS used these days:
MySql, Oracle, SQL Server, IBM DB2, PostgreSQL, Amazon SimpleDB (cloud based) etc.

1.1.4 Database Applications:

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades

- Sales: customers, products, purchases
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions
- Databases touch all aspects of our lives

1.1.5 Characteristics of Database Management System

- Data Consistency
- Support Multiple user and Concurrent Access
- Query Language
- Security

1.1.6 Purpose of Database System

• In the early days, database applications were built on top of file systems
• Drawbacks of using file systems to store data:
– Data redundancy and inconsistency
• Multiple file formats, duplication of information in different files – Difficulty in accessing data
• Need to write a new program to carry out each new task
– Data isolation — multiple files and formats
– Integrity problems
• Integrity constraints (e.g. account balance > 0) become part of program code
• Hard to add new constraints or change existing ones
– Atomicity of updates
• Failures may leave database in an inconsistent state with partial updates carried out
• E.g. transfer of funds from one account to another should either complete or not happen at all
– Concurrent access by multiple users
• Concurrent accessed needed for performance
• Uncontrolled concurrent accesses can lead to inconsistencies
– E.g. two people reading a balance and updating it at the same time
– Security problems
• Database systems offer solutions to all the above problems

1.1.7 Types of Database Management Systems

There are four structural types of database management systems:

- Hierarchical databases.
- Network databases.
- Relational databases.
- Object-oriented databases

1.1.8 Advantages of DBMS

- Segregation of applicaion program.
- Minimal data duplicacy or data redundancy.
- Easy retrieval of data using the Query Language.
- Reduced development time and maintainance need.
- With Cloud Datacenters, we now have Database Management Systems capable of storing almost infinite data.
- Seamless integration into the application programming languages which makes it very easier to add a database to almost any application or website.

1.1.9 Disadvantages of DBMS

- It's Complexity
- Except MySQL, which is open source, licensed DBMSs are generally costly.
- They are large in size.

2. DATA MODEL

2.1 Introduction

- A model is a representation of reality, 'real world' objects and events, associations.
- A collection of tools for describing
  - data
  - data relationships
  - data semantics
  - data constraints

What is Data Model?

- Data models define how the logical structure of a database is modeled.
- Data Models are fundamental entities to introduce abstraction in a DBMS.

- Data models define how data is connected to each other and how they are processed and stored inside the system.

2.2 Characteristics of Data Models

- The very first data model could be flat data-models, where all the data used are to be kept in the same plane.
- Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.
- It is an abstraction that concentrates on the essential, inherent aspects an organization and ignores the accidental properties.
- A data model represents the organization itself. It should provide the basic concepts and notations that will allow database designers and end users unambiguously and accurately to communicate their understanding of the organizational data.
- Data Model can be defined as an integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization.
- A database model is the theoretical foundation of a database and fundamentally determines in which manner data can be stored, organized, and manipulated in a database system.
- It thereby defines the infrastructure offered by a particular database system.

2.3 Uses of Data Models

- These models can be used in database design.
- It provides useful concepts that allow us to move from an informal description to precise description.
- This model was developed to facilitate database design by allowing the specification of overall logical structure of a database.
- It is extremely useful in mapping the meanings and interactions of real world enterprises onto a conceptual schema.
- These models can be used for the conceptual design of database applications.

2.4 Components of a Data Model

A data model comprises of three components:

1) A **structural part**, consisting of a set of rules according to which databases can be constructed.
2) A **manipulative part**, defining the types of operation that are allowed on the data (this includes the operations that are used for updating or retrieving data from the database and for changing the structure of the database).
3) Possibly a **set of integrity rules**, which ensures that the data is accurate.

Thus a data model can help the data designers, DB managers, DB administrators to conceptualize, organize, design and Develop databases and also provides mechanism to manage to the extent that will be useful for entire data science world.

****

3. TYPES OF DATA MODEL

3.1 Introduction

- The entire structure of a database can be described using a data model.
- A data model is a collection of conceptual tools for describing data.
- The benefits of these data models do not limit its helping hand to the time bound and situation bound bond with its depended peoples.

3.2 Categories of data models

Data models can be classified into following types.

1. Object Based Logical Models.

2. Record Based Logical Models.

3. Physical Models.

1) Object Based Logical Models:

These models can be used in describing the data at the logical and view levels.

These models are having flexible structuring capabilities classified into following types.

a) The entity-relationship model.

b) The object-oriented model.

c) The semantic data model.

d) The functional data model.

2) Record Based Logical Models:

These models can also be used in describing the data at the logical and view levels.

These models can be used for both to specify the overall logical structure of the database and a higher-level description.

These models can be classified into

a) Relational model.

b) Network model.

c) Hierarchal model.

3) Physical Models:

These models can be used in describing the data at the lowest level, i.e. physical level. These models can be classified into

a) Unifying model

b) Frame memory model

The most popular example of a database model is the relational model.

4. ENTITY – RELATIONSHIP MODEL (E-R MODEL)

4.1 Introduction

- The entity relationship model is a collection of basic objects called entities and relationship among those objects.
- Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them.
- While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.
- ER Model is best used for the conceptual design of a database.

ER Model is based on −

a) **Entities** and their *attributes.*

b) **Relationships** among entities.

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

- Entity-relationship model is a model used for design and representation of relationships between data.
- The main data objects are termed as Entities, with their details defined as attributes, some of these attributes are important and are used to identity the entity, and different entities are related using relationships.

4.2 Basics of ER Model

There are two techniques used for the purpose of data base designing from the system requirements and they are:

a) Top down Approach known as Entity-Relationship Modeling
b) Bottom Up approach known as Normalization.

- The Entity-Relationship (ER) model is a top down approach of designing database.
- It is a graphical technique, which is used to convert the requirement of the system to graphical representation, so that it can become well understandable.

- It also provides the framework for designing of database.
- The Entity-Relationship (ER) model was originally proposed by Peter in 1976 as a way to unify the network and relational database views.
- Simply stated, the ER model is a conceptual data model that views the real world as entities and relationships.
- A basic component of the model is the Entity-Relationship diagram, which is used to visually represent data objects.

For the database designer, the utility of the ER model is:

• It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.

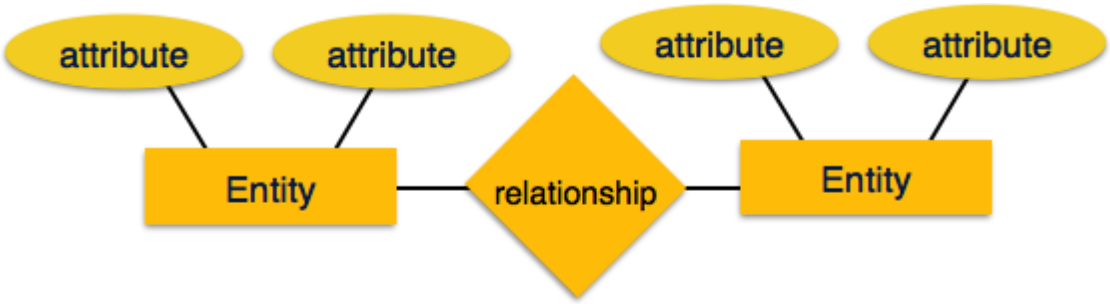
• It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.

• In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

4.3 Elements of E-R Model

The major elements or components of a ERD are the participating elements while creating it.

These concepts are explained below.



The ER elements are:

a) Entity and Entity Set
b) Attributes And Types of Attributes.
c) Keys

d) Relationships

a) **Entity** − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**.

- For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- **Entity Representation** : A Simple rectangular box represents an Entity.

An Entity is generally a real-world object which has characteristics and holds relationships in a DBMS.
If a Student is an Entity, then the complete dataset of all the students will be the **Entity Set**

Entity set: The set of all entities of the same type is termed as an entity set.

Entity type:
An entity type defines a collection of entities that have the same attributes.

Example:
For a **School Management Software**, we will have to store **Student** information, **Teacher** information, **Classes**, **Subjects** taught in each class etc.

Considering the above example, **Student** is an entity, **Teacher** is an entity, similarly, **Class**, **Subject**etc are also entities.

b) **Relationship** − The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways.

c) Mapping cardinalities define the number of association between two entities.

Mapping cardinalities −

- one to one

- one to many
- many to one
- many to many

ER- Diagram Notations

ER- Diagram is a visual representation of data that describe how data is related to each other.

- **Rectangles:** This symbol represent entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

ER Model: Attributes

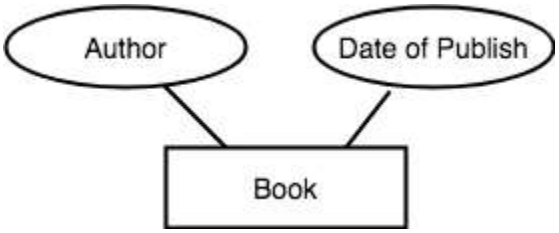
Attributes

An **Attribute** describes a property or characterstic of an entity.
An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set.

Example:
A possible attributes of customer entity are customer name, customer id, Customer Street, customer city.

For example, **Name**, **Age**, **Address** etc can be attributes of a **Student**. An attribute is represented using eclipse.

Attributes for any Entity
Ellipse is used to represent attributes of any entity. It is connected to the entity.



Key Attribute
Key attribute represents the main characterstic of an Entity.
It is used to represent a Primary key.
Ellipse with the text underlined, represents Key Attribute.

Single valued and multi valued attributes
Single valued attributes: attributes with a single value for a particular entity are called single valued attributes.
Multi valued attributes : Attributes with a set of value for a particular entity are called multivalued attributes.

stored and derived attributes
Stored attributes: The attributes stored in a data base are called stored attributes.
Derived attributes: The attributes that are derived from the stored attributes are called derived attributes.

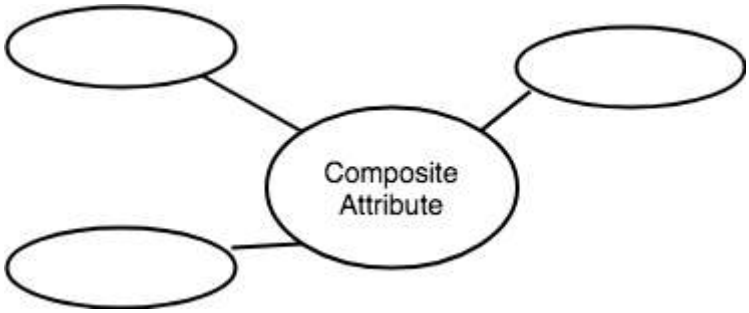
Example :

If a Student is an Entity, then student's **roll no.**, student's **name**, student's **age**, student's **gender** etc will be its attributes.
An attribute can be of many types, here are different types of attributes defined in ER database model:

1. **Simple attribute:** The attributes with values that are atomic and cannot be broken down further are simple attributes. For example, student's **age**.
2. **Composite attribute:** A composite attribute is made up of more than one simple attribute. For example, student's **address** will contain, **house no.**, **street name**, **pincode** etc.

Composite Attribute for any Entity

A composite attribute is the attribute, which also has attributes.



3. **Derived attribute:** These are the attributes which are not present in the whole database management system, but are derived using other attributes. For example, *average age of students in a class*.

Derived Attribute for any Entity

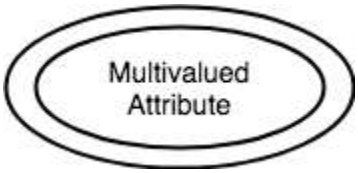
Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth.

To represent a derived attribute, another dotted ellipse is created inside the main ellipse.

4. **Single-valued attribute:** As the name suggests, they have a single value.
5. **Multi-valued attribute:** And, they can have multiple values.

Multivalued Attribute for any Entity

Double Ellipse, one inside another, represents the attribute which can have multiple values.



Key Attribute for any Entity

To represent a Key attribute, the attribute name inside the Ellipse is underlined.

Relationships

- A relationship is an association among several entities.
- When an Entity is related to another Entity, they are said to have a relationship.

Example: A depositor relationship associates a customer with each account that he/she has.

Example

For example, A **Class** Entity is related to **Student** entity, because students study in classes, hence this is a relationship.
Depending upon the number of entities involved, a **degree** is assigned to relationships.
For example, if 2 entities are involved, it is said to be **Binary relationship**, if 3 entities are involved, it is said to be **Ternary** relationship, and so on

There are three types of relationship that exist between Entities.

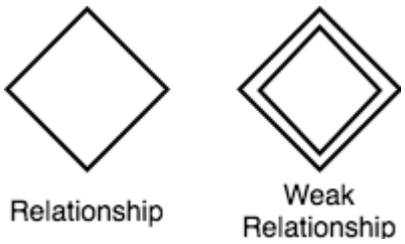
1. Binary Relationship
2. Recursive Relationship
3. Ternary Relationship

Relationship set

Relationship set : The set of all relationships of the same type is termed as a relationship set.

Relationships between Entities - Weak and Strong

Rhombus is used to setup relationships between two or more entities.



D**egree of relationship set**

The degree of relationship type is the number of participating entity types.

i) Key attribute

ii) Value set

Key attribute : An entity type usually has an attribute whose values are distinct from each individual entity in the collection. Such an attribute is called a key attribute.

Value set: Each simple attribute of an entity type is associated with a value set that specifies the set of values that may be assigned to that attribute for each individual entity.

Cardinality

Mapping cardinalities or cardinality ratios express the number of entities to which another entity can be associated. Mapping cardinalities must be one of the following:

• One to one
• One to many
• Many to one
• Many to many

- While creating relationship between two entities, we may often need to face the cardinality problem.
- This simply means that how many entities of the first set are related to how many entities of the second set.

Cardinality can be of the following three types.

a) <u>One-to-One</u>

- Only one entity of the first set is related to only one entity of the second set. E.g. *A teacher teaches a student.*

- Only one teacher is teaching only one student.

This can be expressed in the followingdiagram as:

b) <u>One-to-Many</u>

- Only one entity of the first set is related to multiple entities of the second set.
- E.g. *A teacher teaches students.* Only one teacher is teaching many students.

This can be expressed in the following diagram as:

c) <u>Many-to-One</u>

- Multiple entities of the first set are related to multiple entities of the second set. E.g. *Teachers teach a student.*
- Many teachers are teaching only one student.

This can be expressed in the followingdiagram as:

d) Many-to-Many

- Multiple entities of the first set is related to multiple entities of the second set. E.g. *Teachers teach students.*
- In any school or college many teachers are teaching many students.
- This can be considered as a two way one-to-many relationship.

This can be expressed in the following diagram as:

Weak and strong entity sets

Weak entity set: entity set that do not have key attribute of their own are called weak entity sets. Strong entity set: Entity set that has a primary key is termed a strong entity set.

- Based on the concept of foreign key, there may arise a situation when we have to relate an entity having a primary key of its own and an entity not having a primary key of its own.
- In such a case, the entity having its own primary key is called a strong entity and the entity not having its own primary key is called a weak entity.
- Whenever we need to relate a strong and a weak entity together, the ERD would change just a little.

Example:

- Say, for example, we have a statement "*A Student lives in a Home.*" STUDENT is obviously a strong entity having a primary key Roll.
- But HOME may not have a unique primary key, as its only attribute Address may be shared by many homes (what if it is a housing estate?).
- HOME is a weak entity in this case.

The ERD of this statement would be like the following

As you can see, the weak entity itself and the relationship linking a strong and weak entity must have double border.

4.3 How to Prepare an ERD(Entity Relationship Diagram) ?

There are 3 major steps for preparing an E-R Diagram (ERD)

Step 1: Prepare a written Document

Let us take a very simple example and we try to reach a fully organized database from it.

Let us look at the following simple statement:

A boy eats an ice cream.

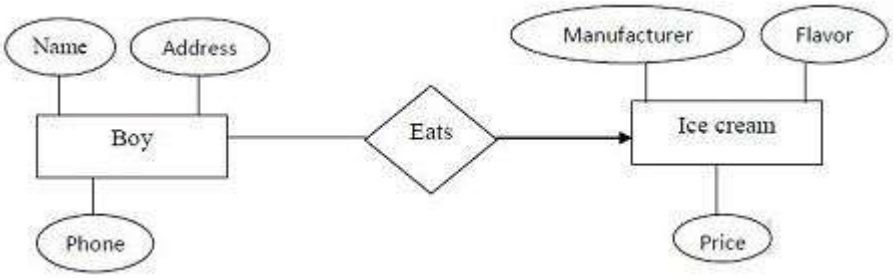
This is a description of a real word activity, and we may consider the above statement as a written document (very short, of course).

Step 2 : P**repare the ERD**

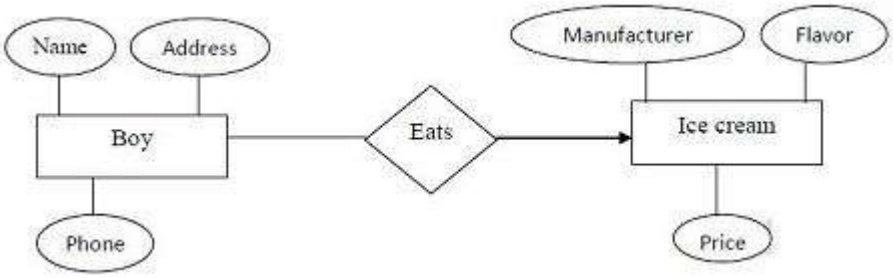
- Now we have to prepare the ERD.
- Before doing that we have to process the statement a little.

- We can see that the sentence contains a subject (*boy*), an object (*ice cream*) and a verb (*eats*) that defines the relationship between the subject and the object. Consider the nouns as entities (*boy* and *ice cream*) and the verb (*eats*) as a relationship.
- To plot them in the diagram, put the nouns within rectangles and the relationship within a diamond.
- Also, show the relationship with a directed arrow, starting from the subject entity (*boy*) towards the object entity (*ice cream*).

- Well, fine. Up to this point the ERD shows how *boy* and *ice cream* are related.
- Now, every boy must have a name, address, phone number etc. and every ice cream has a manufacturer, flavor, price etc. Without these the diagram is not complete.
- These items which we mentioned here are known as attributes, and they must be incorporated in the ERD as connected ovals.



- But can only entities have attributes? Certainly not.
- If we want then the relationship must have their attributes too.
- These attribute do not inform anything more either about the *boy* or the *ice cream*, but they provide additional information about the relationships between the *boy* and the *ice cream.*



Step 3: Convert Entity into Table

We are almost complete now. If you look carefully, we now have defined structures for at least three tables like the following:

Boy

Name	Address	Phone

Ice Cream

Manufacturer	Flavor	Price

Eats

Date	Time

- However, this is still not a working database, because by definition, database should be "collection of related tables."
- To make them connected, the tables must have some common attributes.

If we chose the attribute Name of the ***Boy*** table to play the role of the common attribute, then the revised structure of the above tables become something like the following.

Boy

Name	Address	Phone

Ice Cream

Manufacturer	Flavor	Price	Name

Eats

Date	Time	Name

- This is as complete as it can be.
- We now have information about the boy, about the ice cream he has eaten and about the date and time when the eating was done.

4.4 Advantages and Disadvantages of E-R Data Model

4.4.1 Advantages of E-R Data Model

Following are advantages of an E-R Model:

a) ***Straightforward relation representation*:** Having designed an E-R diagram for a database application, the relational representation of the database model becomes relatively straightforward.
b) ***Easy conversion for E-R to other data model:*** Conversion from E-R diagram to a network or hierarchical data model can· easily be accomplished.
c) ***Graphical representation for better understanding:*** An E-R model gives graphical and diagrammatical representation of various entities, its attributes and relationships between entities. This is turn helps in the clear understanding of the data structure and in minimizing redundancy and other problems.

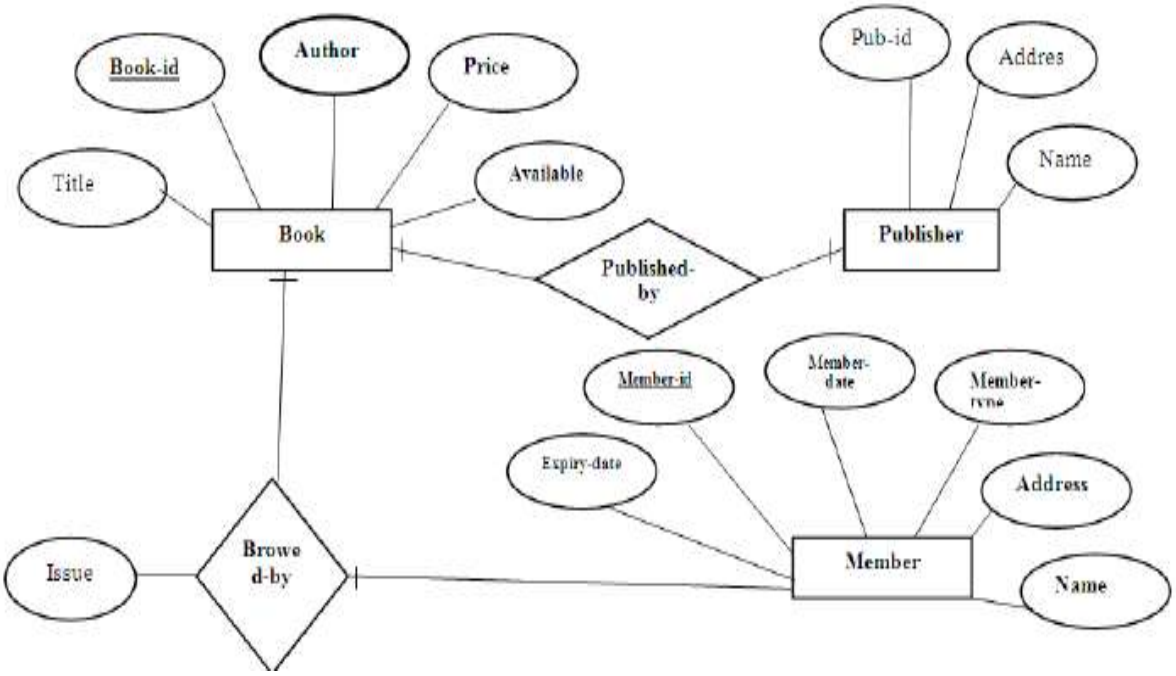
4.4.2 Disadvantages of E-R Data Model

Following are disadvantages of an E-R Model:

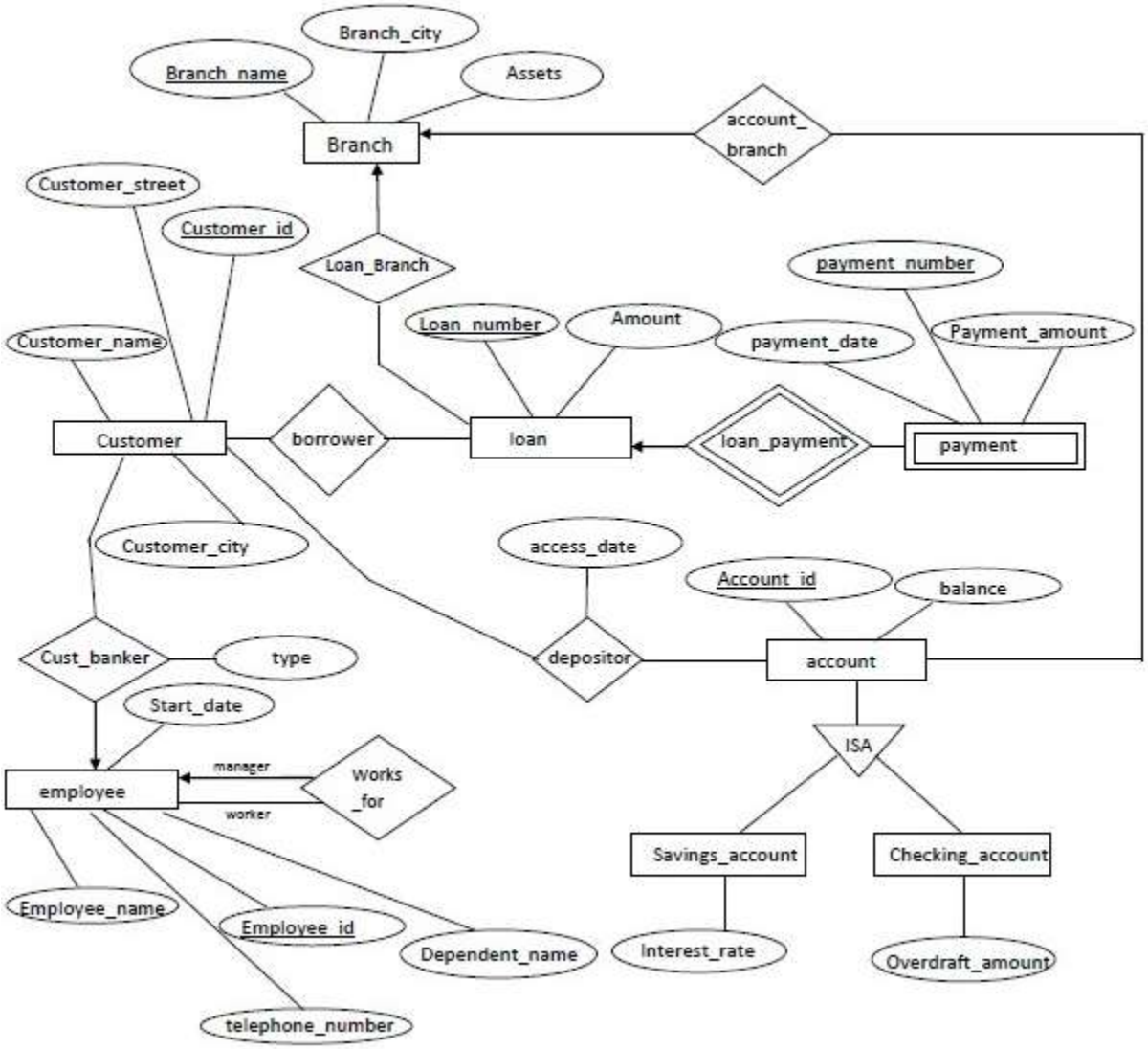
a) **No industry standard for notation:** There is no industry standard notation for developing an E-R diagram.
b) **Popular for high-level design**: The E-R data model is especially popular for high level.

Excercises:

1) **Draw a E-R Diagram for Library Management System.**



2) **Draw a E-R Diagram for a Banking System**



E-R Diagram for Banking System

5. RELATIONAL DATA DMODEL

5.1 Introduction

- The Relational Model is a depiction of how each piece of stored information relates to the other stored information.
- It shows how tables are linked, what type of links are between tables, what keys are used, what information is referenced between tables.
- It's an essential part of developing a normalized database structure to prevent repeat and redundant data storage.
- The basic idea behind the relational model is that a database consists of a series of unordered tables (or relations) that can be manipulated using non-procedural operations that return tables.
- This model was in vast contrast to the more traditional database theories of the time that were much more complicated, less flexible and dependent on the physical storage methods of the data..
- The RELATIONAL database model is based on the Relational Algebra, set theory and predicate logic.

It is commonly thought that the word relational in the relational model comes from the fact that you relate together tables in a relational database.

- Relational model stores data in the form of tables.
- This concept purposed by Dr. E.F. Codd, a researcher of IBM in the year 1960s.

What is Relational Model?

The relational model uses a collection of tables to represent both data and the relationships among those data.

- Relational Model represents how data is stored in Relational Databases.
- A relational database stores data in the form of relations (tables).

- The relational model represents the database as a collection of relations.
- A relation is nothing but a table of values. Every row in the table represents a collection of related data values.

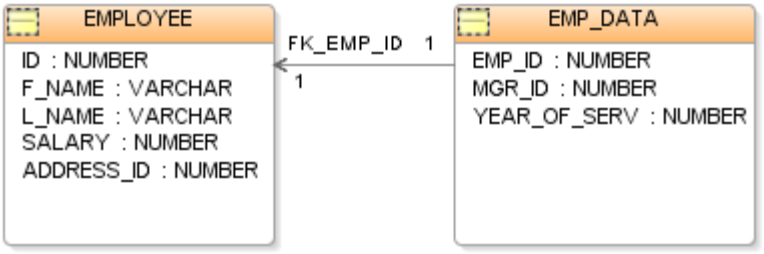
- These rows in the table denote a real-world entity or relationship.
- The table name and column names are helpful to interpret the meaning of values in each row.
- The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.
- Attribute, Tables, Tuple, Relation Schema, Degree, Cardinality, Column, Relation instance, are some important components of Relational Model
- Relational Integrity constraints are referred to conditions which must be present for a valid relation
- Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type
- Insert, Select, Modify and Delete are operations performed in Relational Model
- The relational database is only concerned with data and not with a structure which can improve the performance of the model

Basic rules on Relational

- Data need to be represented as a collection of relations
- Each relation should be depicted clearly in the table
- Rows should contain data about instances of an entity
- Columns must contain data about attributes of the entity
- Cells of the table should hold a single value
- Each column should be given a unique name
- No two rows can be identical
- The values of an attribute should be from the same domain

Example:

Consider a set of Employees in a company. Each employee has 2 –levels of information



- The various tables in the database have a set of tuples.
- The columns enumerate the various attributes of the entity (the employee's name, address or phone number, for example), and a row is an actual instance of the entity (a specific employee) that is represented by the relation.
- As a result, each tuple of the employee table represents various attributes of a single employee.

Tuple and attribute.

- **Attributes:** column headers
- **Tuple** : Row

5.2 Components of Relational Model

The relational model consists of three major components:

1. The set of relations and set of domains that defines the way data can be represented (data structure).

2. Integrity rules that define the procedure to protect the data (data integrity).

3. The operations that can be performed on data (data manipulation).

Relational Database

A rational model database is defined as a database that allows you to group its data items into one or more independent tables that can be related to one another by using fields common to each related table.

Characteristics of Relational Database

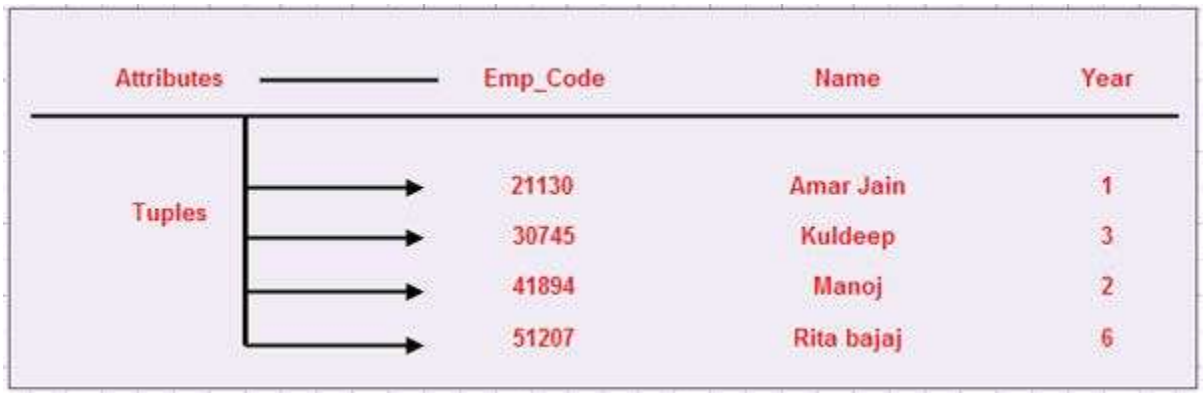
Relational database systems have the following characteristics:

a) The whole data is conceptually represented as an orderly arrangement of data into rows and columns, called a relation or table.
b) .All values are scalar. That is, at any given row/column position in the relation there is one and only one value.
c) . All operations are performed on an entire relation and result is an entire relation, a concept known as closure.

- Dr. Codd, when formulating the relational model, chose the term "relation" because it vas comparatively free of connotations, unlike, for example, the word "table".
- It is a common misconception that the relational model is so called because relationships are established between tables.
- In fact, the name is derived from the relations on whom it is based.
- Notice that the model requires only that data be conceptually represented as a relation, it does not specify how the data should be physically implemented.
- A relation is a relation provided that it is arranged in row and column format and its values are scalar.
- Its existence is completely independent of any physical representation.

5.3 Basic Terminology used in Relational Model

The figure shows a relation with the. Formal names of the basic components marked the entire structure is, as we have said, a relation.



a) Tuples of a Relation

Each row of data is a tuple. Actually, each row is an n-tuple, but the "n-" is usually dropped.

b) **Cardinality of a relation:** The number of tuples in a relation determines its cardinality. In this case, the relation has a cardinality of 4.

c) **Degree of a relation:** Each column in the tuple is called an attribute. The number of attributes in a relation determines its degree. The relation in figure has a degree of 3.

d) **Domains:** A domain definition specifies the kind of data represented by the attribute.

- More- particularly, a domain is the set of all possible values that an attribute may validly contain.
- Domains are often confused with data types, but this is inaccurate.
- Data type is a physical concept while domain is a logical one. "Number" is a data type and "Age" is a domain.
- To give another example "StreetName" and "Surname" might both be represented as text fields, but they are obviously different kinds of text fields; they belong to different domains.
- Domain is also a broader concept than data type, in that a domain definition includes a more specific description of the valid data.

For example, the domain Degree A warded, which represents the degrees awarded by a university.

In the database schema, this attribute might be defined as Text [3], but it's not just any three-character string, it's a member of the set {BA, BS, MA, MS, PhD, LLB, MD}.

Of course, not all domains can be defined by simply listing their values. Age, for example, contains a hundred or so values if we are talking about people, but tens of thousands if we are talking about museum exhibits.

In such instances it's useful to define the domain in terms of the rules, which can be used to determine the membership of any specific value in the set of all valid values.

For example, Person Age could be defined as "an integer in the range 0 to 120" whereas Exhibit Age (age of any object for exhibition) might simply by "an integer equal to or greater than 0."

Body of a Relation: The body of the relation consists of an unordered set of zero or more tuples.

There are some important concepts here.

a) First the relation is unordered. Record numbers do not apply to relations.
b) Second a relation with no tuples still qualifies as a relation.
c) Third, a relation is a set.

The items in a set are, by definition, uniquely identifiable.

Therefore, for a table to qualify as a relation each record must be uniquely identifiable and the table must contain no duplicate records.

Keys of a Relation

It is a set of one or more columns whose combined values are *unique* among all occurrences in a given table.

A key is the relational means of specifying uniqueness. Some different types of keys are:

a) Primary key is an attribute or a set of attributes of a relation which posses the properties of uniqueness and irreducibility (No subset should be unique).

For example: Supplier number in S table is primary key, Part number in P table is primary key and the combination of Supplier number and Part Number in SP table is a primary key

b) Foreign key is the attributes of a table, which refers to the primary key of some another table.

- Foreign key permit only those values, which appears in the primary key of the table to which it refers or may be null (Unknown value).
- For example: SNO in SP table refers the SNO of S table, which is the primary key of S table, so we can say that SNO in SP table is the foreign key.
- PNO in SP table refers the PNO of P table, which is the primary key of P table, so we can say that PNO in SP table is the foreign key.
- The database of Customer-Loan, which we discussed earlier for hierarchical model and network model, is now represented for Relational model as shown.
- In can easily understood that, this model is very simple and has no redundancy.

- The total database is divided in to two tables. Customer table contains the information about the customers with CNO as the primary key.
- The Cutomer_Loan table stores the information about CNO, LNO and AMOUNT.
- It has the primary key combination of CNO and LNO.
- Here, CNO also acts as the foreign key and refers to CNO of Customer table.
- It means, only those customer number are allowed in transaction table Cutomer_Loan that have their entry in the master Customer table.

Customer Table

CNO	NAME	ADDRESS	CITY
C1	Rahal	Thapar campus	Patlala
C2	Ruhi	Tagore Nagar	Jalandhar
C3	Chachat	Dharampura	Qadian
C4	Pooja	GNDU	Amritsar

Customer_Loan Table

CNO	LNO	AMOUNT
C1	L1	10000
C2	L1	10000
C3	L2	15000
C3	L3	25000
C4	L4	35000

Relationat Model of Custmor-Loan database

Relational View of Sample database

Let us take an example of a sample database consisting of supplier, parts and shipments tables. The table structure and some sample records for supplier, parts and shipments tables are given as Tables as shown below:

The Supplier Records

Sno	Name	Status	Status
S1	Suneet	20	Qadian
S2	Ankit	10	Amritsar
S3	Amit	10	Qadian

The Part Records

Pno	Name	Status	Weight	City
P1	Nut	Red	12	Qadian
P2	Bolt	Green	17	Amritsar
P3	Screw	Blue	17	Jalandhar
P4	Screw	Red	14	Qadian

The Shipment Records

Sno	Name	Qty
S1	P1	250
S1	P2	300
S1	P3	500
S2	P1	250
S2	P2	500
S3	P2	300

- We assume that each row in Supplier table is identified bya unique SNo (Supplier Number), which uniquely identifies the entire row of the table.Likewise each part has a unique PNo (Part Number).
- Also, we assume that no more thanone shipment exists for a given supplier/part combination_in the shipments table.
- Note that the relations Parts and Shipments have PNo (Part Number) in common andSupplier and Shipments relations have SNo (Supplier Number) in common.
- The Supplier andParts relations have City in common.
- For example, the fact that supplier S3 and part P2 arelocated in the same city is represented by the appearance of the same value, Amritsar, in thecity column of the two tuples in relations.

5.4 Operations in Relational Model

The four basic operations in Relational Models and they are as follows:

a) Insert
b) Update
c) Delete
d) Retrieve

The four operations are shown below on the sample database in relational model:

a) Insert Operation:

- Suppose we wish to insert the information of supplier who does not supply any part, can be inserted in S table without any anomaly e.g. S4 can be inserted in Stable.
- Similarly, if we wish to insert information of a new part that is not supplied by any supplier can be inserted into a P table.
- If a supplier starts supplying any new part, then this information can be stored in shipment table SP with the supplier number, part number and supplied quantity.
- So, we can say that insert operations can be performed in all the cases without any anomaly.

b) Update Operation:

- Suppose supplier S1 has moved from Qadian to Jalandhar.
- In that case we need to make changes in the record, so that the supplier table is up-to-date.
- Since supplier number is the primary key in the S (supplier) table, so there is only a single entry of S 1, which needs a single update and problem of data inconsistencies would not arise.
- Similarly, part and shipment information can be updated by a single modification in the tables P and SP respectively without the problem of inconsistency.
- Update operation in relational model is very simple and without any anomaly in case of relational model.

c) Delete Operation:

- Suppose if supplier S3 stops the supply of part P2, then we have to delete the shipment connecting part P2 and supplier S3 from shipment table SP.
- This information can be deleted from SP table without affecting the details of supplier of S3 in supplier table and part P2 information in part table.
- Similarly, we can delete the information of parts in P table and their shipments in SP table and we can delete the information suppliers in S table and their shipments in SP table.

d) Record Retrieval:

Record retrieval methods for relational model are simple and symmetric which can be clarified with the following queries:

Query1: ***Find the supplier numbers for suppliers who supply part P2.***

Solution: In order to get this information we have to search the information of part P2 in the SP table (shipment table). For this a loop is constructed to find the records of P2 and on getting the records, corresponding supplier numbers are printed.

Algorithm

do until no more shipments;

get next shipment where PNO=P2;

print SNO;

end;

Query2: ***Find part numbers for parts supplied by supplier 52.***

Solution: In order to get this information we have to search the information of supplier S2 in the SP table (shipment table). For this a loop is constructed to find the records of S2 and on getting the records corresponding part numbers are printed.

Algorithm

do until no more parts;

get next shipment where SNO=S2;

print PNO;

end;

Since, both the queries involve the same logic and are very simple, so we can conclude that retrieval operation of this model is simple and symmetric.

Advantages and Disadvantages of Relational Model

Advantages of using Relational model

a) **Simplicity**: A relational data model is simpler than the hierarchical and network model.
b) **Structural Independence**: The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
c) **Easy to use**: The relational model is easy as tables consisting of rows and columns is quite natural and simple to understand
d) **Query capability**: It makes possible for a high-level query language like SQL to avoid complex database navigation.
e) **Data independence**: The structure of a database can be changed without having to change any application.
f) **Scalable**: Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Additional advantages of the relational model are:

a) ***Structural independence:***

- In relational model, changes in the database structure do not affect the data access.
- When it is possible to make change to the database structure without affecting the DBMS's capability to access data, we can say that structural independence has been achieved. So, relational database model has structural independence.

b) ***Conceptual simplicity:***

- We have seen that both the hierarchical and the network database model were conceptually simple.
- But the relational database model is even simpler at the conceptual level.
- Since the relational data model frees the designer from the physical data storage details, the designers can concentrate on the logical view of the database.

c) Design, implementation, maintenance and usage ease:

The relational database model\ achieves both data independence and structure independence making the database design, maintenance, administration and usage much easier than the other models.

d) Ad hoc query capability:

- The presence of very powerful, flexible and easy-to-use query capability is one of the main reasons for the immense popularity of the relational database model.
- The query language of the relational database models structured query language or SQL makes ad hoc queries a reality.
- SQL is a fourth generation language (4GL). A 4 GL allows the user to specify what must be done without specifying how it must be done.
- So, sing SQL the users can specify what information they want and leave the details of how to get the information to the database.

Disadvantages of Relational Model

- The relational model's disadvantages are very minor as compared to the advantages and their capabilities far outweigh the shortcomings
- Also, the drawbacks of the relational database systems could be avoided if proper corrective measures are taken.
- The drawbacks are not because of the shortcomings in the database model, but the way it is being implemented.

Some of the disadvantages are:

a) Hardware overheads:

- Relational database system hides the implementation complexities and the physical data storage details from the users.

- For doing this, i.e. for making things easier for the users, the relational database systems need more powerful hardware computers and data storage devices.
- So, the RDBMS needs powerful machines to run smoothly.
- But, as the processing power of modem computers is increasing at an exponential rate and in today's scenario, the need for more processing power is no longer a very big issue.

b) Ease of design can lead to bad design:

- The relational database is an easy to design and use. The users need not know the complex details of physical data storage.
- They need not know how the data is actually stored to access it.
- This ease of design and use can lead to the development and implementation of very poorly designed database management systems.
- Since the database is efficient, these design inefficiencies will not come to light when the database is designed and when there is only a small amount of data.
- As the database grows, the poorly designed databases will slow the system down and will result in performance degradation and data corruption.

c) 'Information island' phenomenon:

- As we have said before, the relational database systems are easy to implement and use. This will create a situation where too many people or departments will create their own databases and applications.
- These information islands will prevent the information integration that is essential for the smooth and efficient functioning of the organization.
- These individual databases will also create problems like data inconsistency, data duplication, data redundancy and so on.
- But as we have said all these issues are minor when compared to the advantages and all these issues could be avoided if the organization has a properly designed database and has enforced good database standards.

Other Disadvantages of using Relational model

a) Few relational databases have limits on field lengths which can't be exceeded.

b) Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.

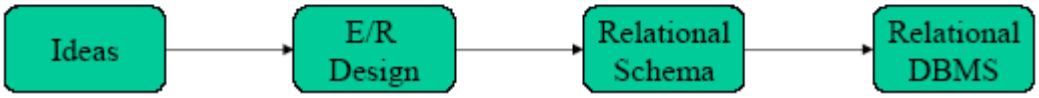
c) Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.

****

6. MAPPING FROM ER MODEL TO RELATIONAL MODEL

6.1 Introduction

- The ER Model can be represented *using ER Diagrams which is a great way of designing and representing the database design in more of a flow chart form.*
- It is ***very convenient to design the database***
  - using the ER Model by creating an ER diagram and
  - later on converting it into relational model to design your tables.
- Not all the ER Model constraints and components can be directly transformed into relational model, but **an approximate schema can be derived**.
- The basic idea on Real world scenario into ER Model and to Relational Model is depicted as follows:
  - Steps in translation:
    - Entity sets to tables
    - Relationships to tables
    - Constraints
    - Weak entity sets



6.2 Basic rules of Conversion of ER diagrams into relational model schema

1) Entity becomes Table
   - Entity in ER Model is changed into tables, or we can say for every Entity in ER model, a table is created in Relational Model.
2) The attributes of the Entity should be converted to columns of the table.

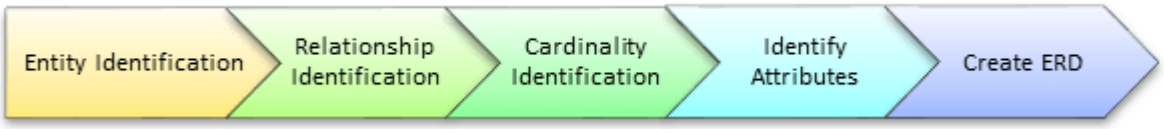
3) The primary key specified for the entity in the ER model, will become the primary key for the table in relational model.

Steps to Create an ERD

Following are the steps to create an ERD:

Step 1) Entity Identification
Step 2) Relationship Identification
Step 3) Cardinality Identification
Step 4) Identify Attributes
Step 5) Create the ERD

The following diagram depicts those 5 sequential steps pictorially:



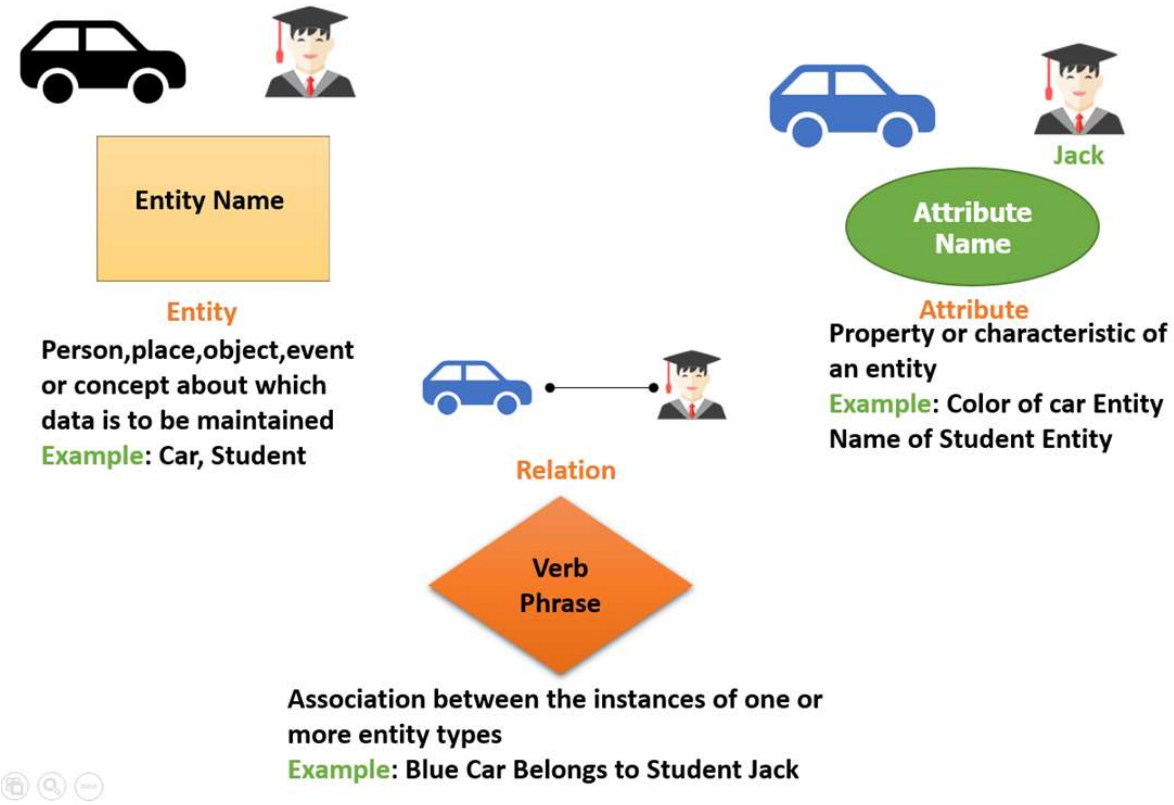
Example:

A real time scenario : UNIVERSITY ENVIRONMENT

In a university, a Student enrolls in Courses. A student must be assigned to at least one or more Courses. Each course is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course

Example

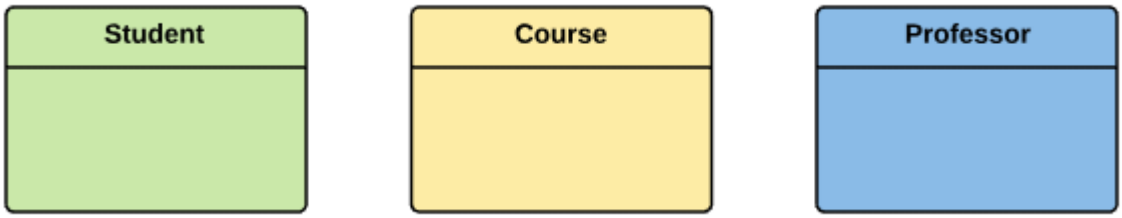
For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.



Step 1) Entity Identification

We have three entities

- Student
- Course
- Professor



Step 2) Relationship Identification

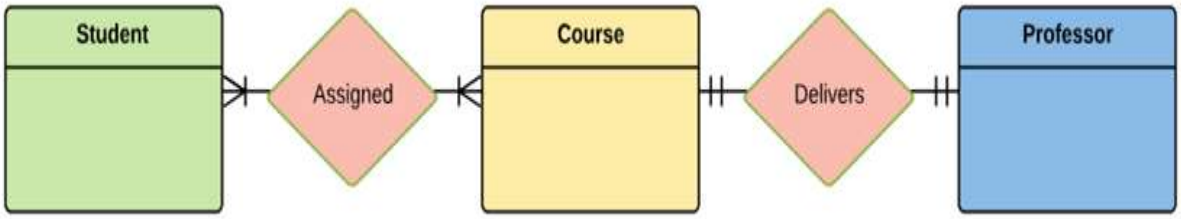
We have the following two relationships

- The student is **assigned** a course
- Professor **delivers** a course

Step 3) Cardinality Identification

For them problem statement we know that,

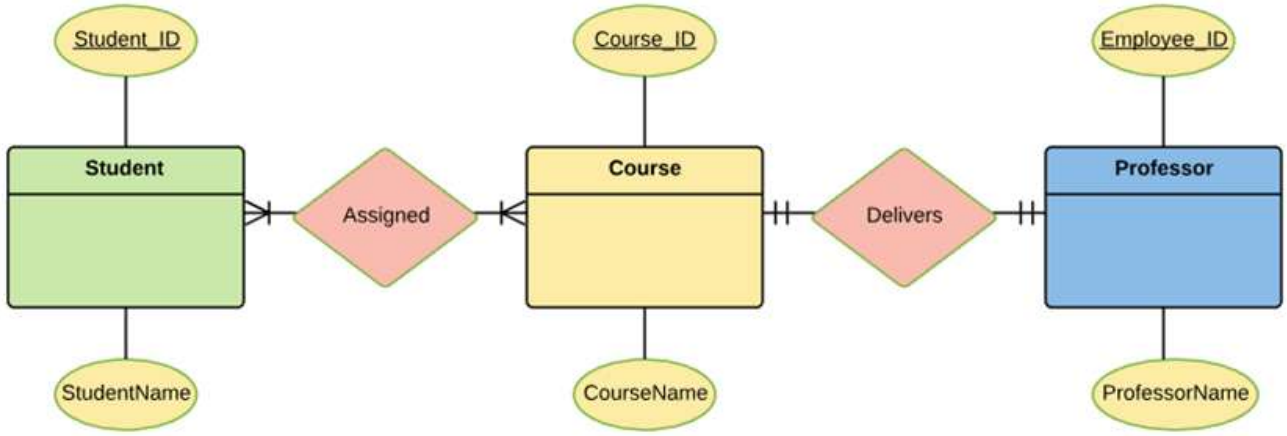
- A student can be assigned **multiple** courses
- A Professor can deliver only **one** course



Step 4) Identify Attributes

- First study the files, forms, reports, data currently maintained by the organization to identify attributes.
- Conduct interviews with various stakeholders to identify entities. Initially, it's important to identify the attributes without mapping them to a particular entity.
- Once, you have a list of Attributes, you need to map them to the identified entities. Ensure an attribute is to be paired with exactly one entity. If you think an attribute should belong to more than one entity, use a modifier to make it unique.
- Once the mapping is done, identify the primary Keys. If a unique key is not readily available, create one.

Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName



For Course Entity, attributes could be Duration, Credits, Assignments, etc. For the sake of ease we have considered just one attribute.

Step 5) Create the ERD

A more modern representation of ERD Diagram

Mapping Process Guidelines

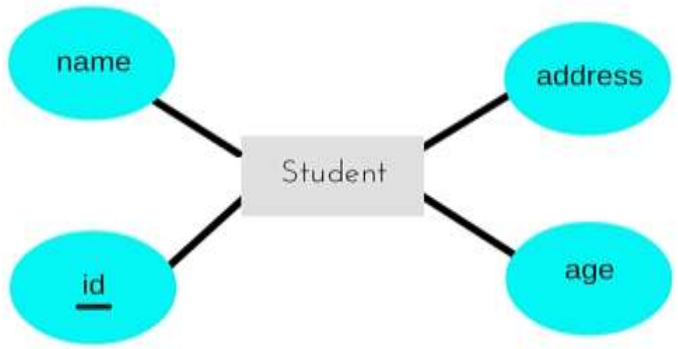
i. Create tables for all higher-level entities.
ii. Create tables for lower-level entities.
iii. Add primary keys of higher-level entities in the table of lower-level entities.
iv. In lower-level tables, add all other attributes of lower-level entities.
v. Declare primary key of higher-level table and the primary key for lower-level table.
vi. Declare foreign key constraints.

6.3 Examples of ER diagrams and convert it into relational model schema

Step 1: Each entity becomes table

Example - 1:

For example, for the below ER Diagram in ER Model,



A table with name Student will be created in relational model, which will have 4 columns, id, name, age, address and id will be the primary key for this table.

Student Table

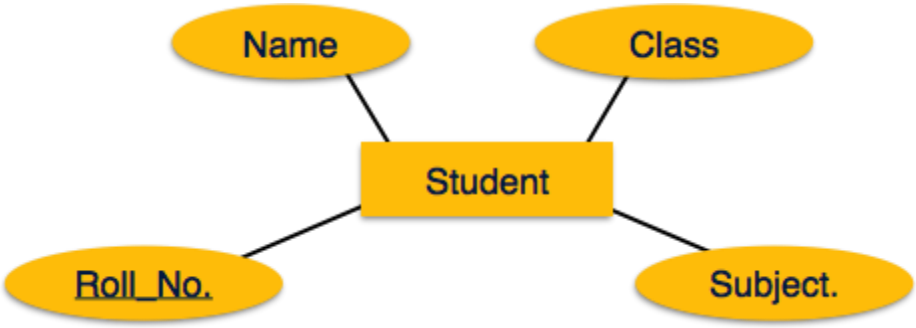
Id	Name	Address	age

Note:

Once the student table is ready, then as many required students' records will be added with no-time.

Example – 2:

An entity is a real-world object with some attributes.



Step 2:Relationship becomes a Relationship Table

- In ER diagram, we use diamond/rhombus to represent a relationship between two entities.
    - In Relational model we create a relationship table for ER Model relationships too.

Apply Mapping Process (Algorithm)

i. Create table for each entity.
ii. Entity's attributes should become fields of tables with their respective data types.

iii. Declare primary key.

Example 3:

In the ER diagram below, we have two entities Teacher and Student with a relationship between them.

From the above extended scenario, there are two tables identified i) Student ii) Teacher from the depicted E-R diagram of two entities such as Teacher and student.

Student Table

Id	Name	Address	Age

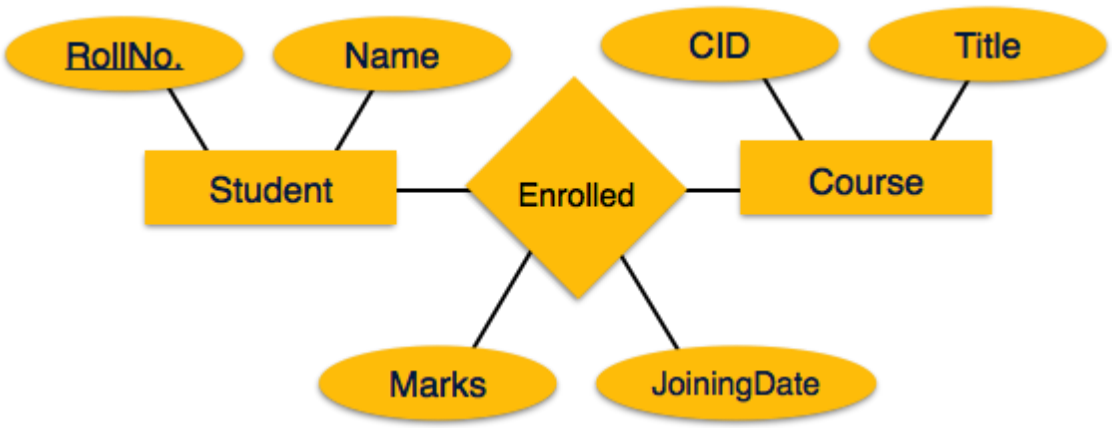
Teacher Table

t_Id	T_Name	t_Address	Id

- When mapping starts the entity gets mapped to table,
  - Hence we will create table for Teacher and a table for Student with all the attributes converted into columns.
- Now, an additional table will be created for the relationship, for example **StudentTeacher** or give it any name you like.
- This table will hold the primary key for both Student and Teacher, in a tuple to describe the relationship, which teacher teaches which student.
- If there are additional attributes related to this relationship, then they become the columns for this table, like subject name.

Example -2: Mapping Relationship

- A relationship is an association among entities.



Step 3: proper foriegn key constraints must be set for all the tables.
A common attribute that links the values / tuples of both Teacher and Student should be identified and it acts as foreign key.

TeacherStudent Table

Id	t_Id	Class_Id	Course

Here, Id + t_Id becomes the primary key for TeacherStudent Table .
The duplicated Id field in the Teacher table becomes foreign key.

Mapping Process for newly created table(s) from relationship mapping process

i. Create table for a relationship.
ii. Add the primary keys of all participating Entities as fields of table with their respective data types.
iii. If relationship has any attribute, add each attribute as field of table.
iv. Declare a primary key composing all the primary keys of participating entities.
v. Declare all foreign key constraints.

Note:

- Similarly we can generate relational database schema using the ER diagram.
- We cannot import all the ER constraints into relational model, but an approximate schema can be generated.
- There are several processes and algorithms available to convert ER Diagrams into Relational Schema.

- Some of them are automated and some of them are manual.

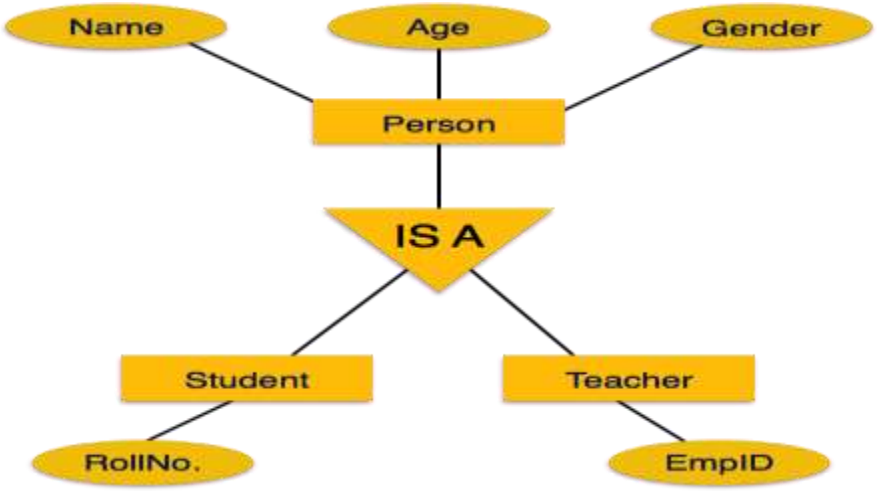
A special scenario

Mapping Process for entity with Weak entities

- Create table for weak entity set.
- Add all its attributes to table as field.
- Add the primary key of identifying entity set.
- Declare all foreign key constraints.

Mapping Hierarchical Entities

ER specialization or generalization comes in the form of hierarchical entity sets.



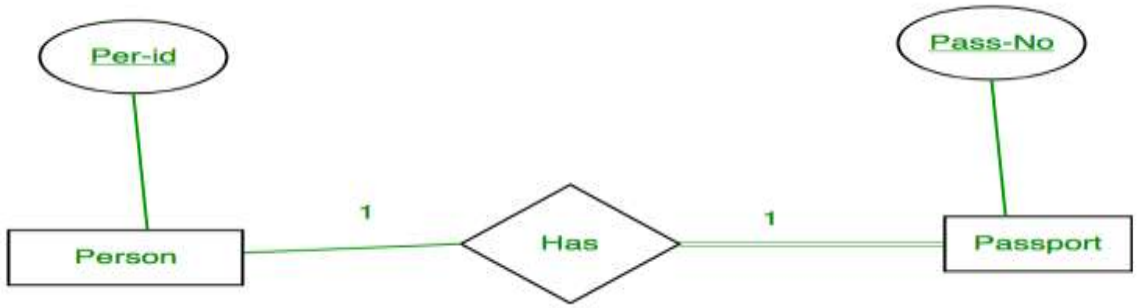
6.4 Points to Remember

Following are some key points to keep in mind while doing so:

a) Entity gets converted into Table, with all the attributes becoming fields(columns) in the table.
b) Relationship between entities is also converted into table with primary keys of the related entities also stored in it as foreign keys.
c) Primary Keys should be properly set.
d) For any relationship of Weak Entity, if primary key of any other entity is included in a table, foriegn key constraint must be defined.

How to convert ER diagram to Relational Model for different scenarios?

Case 1: Binary Relationship with 1:1 cardinality with total participation of an entity



A person has 0 or 1 passport number and Passport is always owned by 1 person. So it is 1:1 cardinality with full participation constraint from Passport.

- First Convert each entity and relationship to tables.
- Person table corresponds to Person Entity with key as Per-Id.
- Similarly Passport table corresponds to Passport Entity with key as Pass-No.
- Has Table represents relationship between Person and Passport (Which person has which passport).
    - So it will take attribute Per-Id from Person and Pass-No from Passport.

Person		Has		Passport	
Per-Id	Other Person Attribute	Per-Id	Pass-No	Pass-No	Other PassportAttribute
PR1	–	PR1	PS1	PS1	–
PR2	–	PR2	PS2	PS2	–
PR3	–				

Table 1

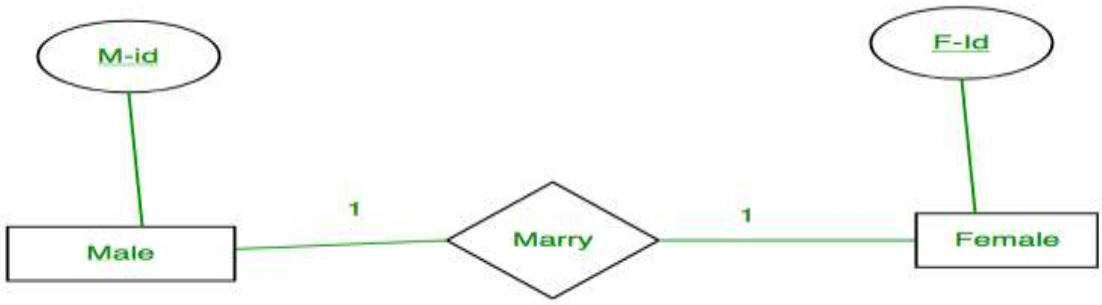
- As we can see from Table 1, each Per-Id and Pass-No has only one entry in Has table.
- So we can merge all three tables into 1 with attributes shown in Table 2.
- Each Per-Id will be unique and not null. So it will be the key.
- Pass-No can't be key because for some person, it can be NULL.

Per-Id	Other Person	Pass-No	Other PassportAttribute

Attribute

Table 2

Case 2: Binary Relationship with 1:1 cardinality and partial participation of both entities



- A male marries 0 or 1 female and vice versa as well. So it is 1:1 cardinality with partial participation constraint from both.
- First Convert each entity and relationship to tables. Male table corresponds to Male Entity with key as M-Id.
- Similarly Female table corresponds to Female Entity with key as F-Id. Marry Table represents relationship between Male and Female (Which Male marries which female). So it will take attribute M-Id from Male and F-Id from Female.

Male		Marry		Female	
M-Id	Other Male Attribute	M-Id	F-Id	F-Id	Other FemaleAttribute
M1	–	M1	F2	F1	–
M2	–	M2	F1	F2	–
M3	–			F3	–

Table 3

- As we can see from Table 3, some males and some females do not marry. If we merge 3 tables into 1, for some M-Id, F-Id will be NULL.
- So there is no attribute which is always not NULL. So we can't merge all three tables into 1. We can convert into 2 tables.
- In table 4, M-Id who are married will have F-Id associated. For others, it will be NULL. Table 5 will have information of all females.
- Primary Keys have been underlined.
-

M-Id	Other Male Attribute	F-Id

Table 4

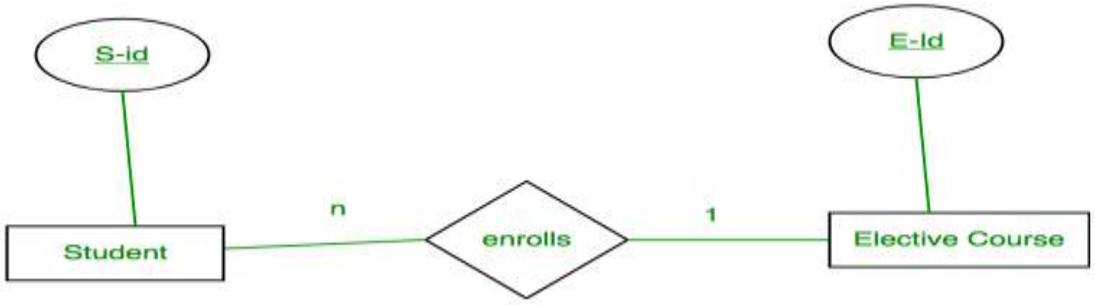
F-Id	Other FemaleAttribute

Table 5

Note:

a) Binary relationship with 1:1 cardinality will have 2 table if partial participation of both entities in the relationship.
b) If atleast 1 entity has total participation, number of tables required will be 1.

Case 3: Binary Relationship with n: 1 cardinality



- In this scenario, every student can enroll only in one elective course but for an elective course there can be more than one student.
- First Convert each entity and relationship to tables. Student table corresponds to Student Entity with key as S-Id.
- Similarly Elective_Course table corresponds to Elective_Course Entity with key as E-Id. Enrolls Table represents relationship between Student and Elective_Course (Which student enrolls in which course).
- So it will take attribute S-Id from and Student E-Id from Elective_Course.

Student		Enrolls		Elective_Course	
S-Id	Other Student Attribute	S-Id	E-Id	E-Id	Other Elective CourseAttribute
S1	–	S1	E1	E1	–
S2	–	S2	E2	E2	–

S3	–	S3	E1	E3	–
S4	–	S4	E1		

Table 6

- As we can see from Table 6, S-Id is not repeating in Enrolls Table. So it can be considered as a key of Enrolls table.
- Both Student and Enrolls Table's key is same; we can merge it as a single table.
- The resultant tables are shown in Table 7 and Table 8.
- Primary Keys have been underlined.

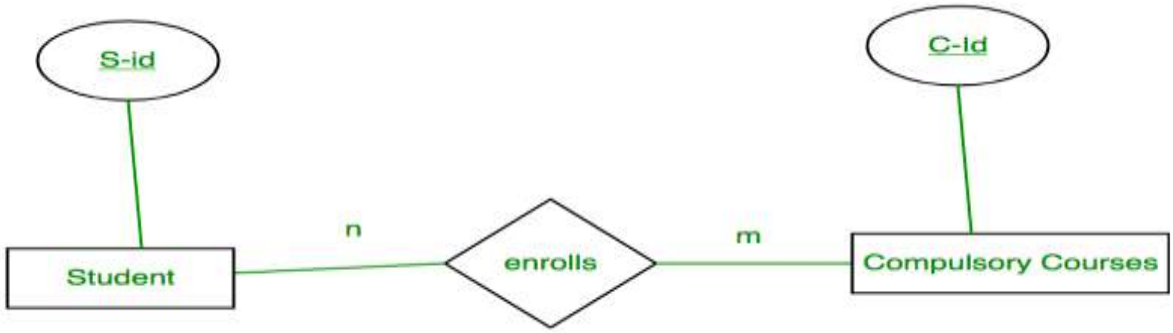
S-Id	Other Student Attribute	E-Id

Table 7

E-Id	Other Elective CourseAttribute

Table 8

Case 4: Binary Relationship with m: n cardinality



- In this scenario, every student can enroll in more than 1 compulsory course and for a compulsory course there can be more than 1 student.
- First Convert each entity and relationship to tables.
- Student table corresponds to Student Entity with key as S-Id. Similarly Compulsory_Courses table corresponds to Compulsory Courses Entity with key as C-Id. Enrolls Table represents relationship between Student and Compulsory_Courses (Which student enrolls in which course).
- So it will take attribute S-Id from Person and C-Id from Compulsory_Courses.

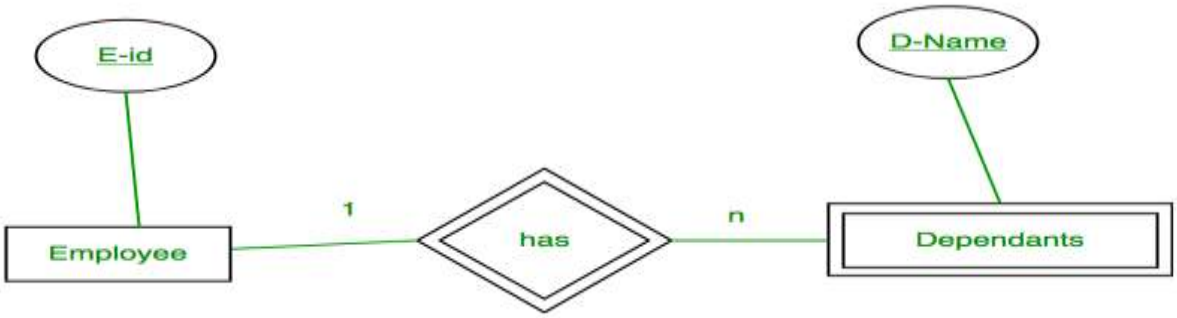
Student		Enrolls		Compulsory_Courses	
S-Id	Other Student	S-Id	C-Id	C-Id	Other Compulsory

	Attribute				CourseAttribute
S1	–	S1	C1	C1	–
S2	–	S1	C2	C2	–
S3	–	S3	C1	C3	–
S4	–	S4	C3	C4	–
		S4	C2		
		S3	C3		

Table 9

- As we can see from Table 9, S-Id and C-Id both are repeating in Enrolls Table.
- But its combination is unique; so it can be considered as a key of Enrolls table.
- All tables' keys are different, these can't be merged.
- Primary Keys of all tables have been underlined.

Case 5: Binary Relationship with weak entity



- In this scenario, an employee can have many dependants and one dependant can depend on one employee.
- A dependant does not have any existence without an employee (e.g; you as a child can be dependant of your father in his company).
- So it will be a weak entity and its participation will always be total. Weak Entity does not have key of its own.
- So its key will be combination of key of its identifying entity (E-Id of Employee in this case) and its partial key (D-Name).
- First Convert each entity and relationship to tables.
- Employee table corresponds to Employee Entity with key as E-Id.

- Similarly Dependants table corresponds to Dependant Entity with key as D-Name and E-Id.
- Has Table represents relationship between Employee and Dependants (Which employee has which dependants). So it will take attribute E-Id from Employee and D-Name from Dependants.

Employee		Has		Dependents	
E-Id	Other Employee Attribute	E-Id	D-Name	E-Id	Other DependentAttribute
E1	–	E1	RAM	E1	–
E2	–	E1	SRINI	E1	–
E3	–	E2	RAM	E2	–
E-Id	–	E3	ASHISH	E3	–

Table 10

- As we can see from Table 10, E-Id, D-Name is key for Has as well as Dependants Table.
- So we can merge these two into 1. So the resultant tables are shown in Tables 11 and 12.
- Primary Keys of all tables have been underlined.

E-Id	Other Employee Attribute

Table 11

D-Name	E-Id	Other DependantsAttribute

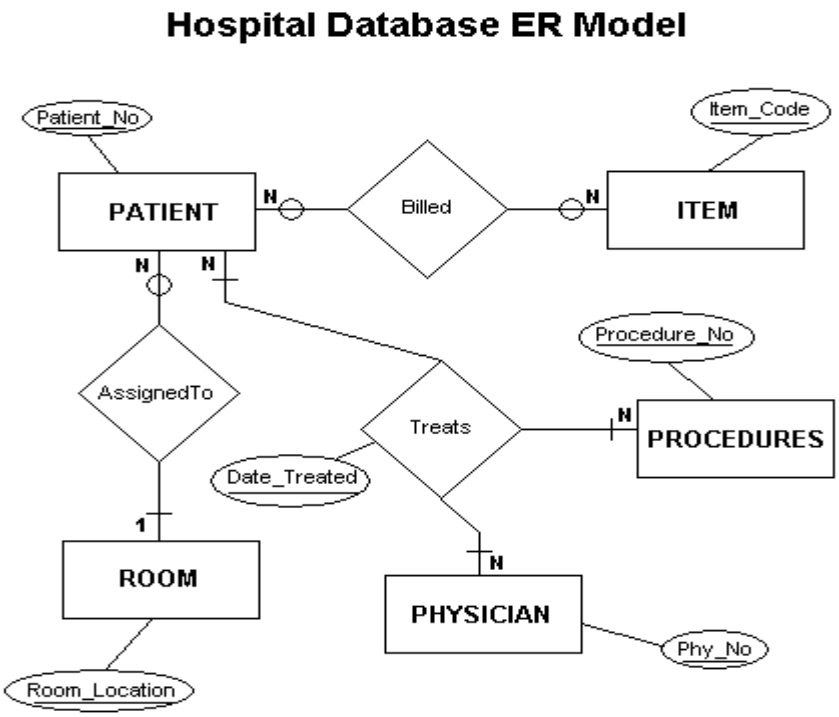
Table 12

Thus mapping from ER Diagram into Relational is achieved through formal step-by-step processes.

6.5 PRACTICAL EXCERCISES

Exercise – 1:

A Hospital real world scenario is depicted in the following ER model:



a) Convert the ER Model into its equivalent Relation model.
(Hint. Give the Table with its structure for the Entities and it's relationships depicted in the above diagram)

b) Give the answers for the Query given below with sample possible data of your own with relevance to the tables of the relational model.

i) Display the **PATIENT_NO**, **ITEM_CODE**, and **CHARGE** and from the **BILLED** table for a specific **PATIENT_NO.**

ii) Display the distinct bill charges from Billed.

Answer:

a)

PATIENT	
PATIENT_NO	**NUMBER(4) - PRI KEY**

DATE_LAST_TREATED	**DATE**
PAT_NAME	**VARCHAR2(50)**
ROOM_LOCATION	**CHAR(4)**

ITEM	
ITEM_CODE	**NUMBER(4) - PRI KEY**
DESCRIPTION	**VARCHAR2(50)**
NORMAL_CHARGE	**NUMBER(7,2)**

PHYSICIANS	
PHY_ID	**NUMBER(4) - PRI KEY**
PHY_PHONE	**CHAR(8)**
PHY_NAME	**VARCHAR2(50)**

ROOM	
ROOM_LOCATION	**CHAR(4) - PRI KEY**
ROOM_ACCOMODATION	**CHAR(2)**
ROOM_EXTENSION	**NUMBER(4)**

PROCEDURES	
PROCEDURE_NO	**NUMBER(4) - PRI KEY**
PROC_DESCRIPTION	**VARCHAR2(50)**

BILLED	
BILL_NO	**NUMBER(5) - PRI KEY**
PATIENT_NO	**NUMBER(9)**
ITEM_CODE	**NUMBER(5)**
CHARGE	**NUMBER(7,2)**

TREATS	
PHY_ID	**NUMBER(4) - PRI KEY**
PATIENT_NO	**NUMBER(4) - PRI KEY**
PROCEDURE_NO	**NUMBER(4) - PRI KEY**
DATE_TREATED	**DATE - PRI KEY**
TREAT_RESULT	**VARCHAR2(50)**

b)

Query 1:

Display the **PATIENT_NO**, **ITEM_CODE**, and **CHARGE** and from the **BILLED** table for a specific **PATIENT_NO.**

Answer :

```
SELECT patient_no, item_code, charge
FROM billed
WHERE patient_no = 1117;
```

Output:

PATIENT_NO	ITEM_CODE	CHARGE
1117	2222	7.54
1117	2255	25

Query 2:

Display the distinct bill charges from Billed.

Answer:

```
SELECT DISTINCT charge
FROM billed;
```

Output:

CHARGE
2.21
4.56
6.68
7.54
7.75
25

Exercise – 2:

A real world scenario related a company has been described as follows:

Scenario

A small company named "ABC Company" which has nominal number of CUSTOMER with selected PRDUCT to sell to its customers in the form receiving ORDER.
As soon as the order is confirmed through the LINE_ITEM generation and then the INVOICE are generated. Then the SHIPMENT process is to be initiated and complete the shipping the products and generate bill of the same.
The minimal information that are included for the ABC SALES PROCESS are as follows:

i) CUSTOMER entity hold the attributes of Customer ID, Name and Address.
ii) PRODUT entity holds Prodcut_ID and its' Description.
iii) ORDER entity holds Order_Number and Order_Date.
iv) INLINE_ITEM entity holds the Ordered Item's Quantity.
v) INVOICE entity holds invoide_Number.
vi) SHIPMENT entity holds Ship_Quantity.

The process of sales for ABS Company is best described as follows:

a) A customer is allowed the place any number of Orders.
b) Once the Order is initiated the identified Products are to be considers as Line Items.
c) As soon as Order confirms an Invoice is to be generated.
d) Then the shipment process starts and includes the possible date of Shipment.

Your are required the draw the E-R diagram based on the information given in the above scenario for the SALES process of ABC Company.

Then convert the ERD into its' equivalent Relational Model by applying the basic mapping rules for ERD TO RELATIONAL model.

(Hint. Carefully read and comprehend the information flow between different entities with the context of Sales business process.)

a) Draw a ER diagram for the given scenario.
b) Convert the given ERD into the relations.

Answer:

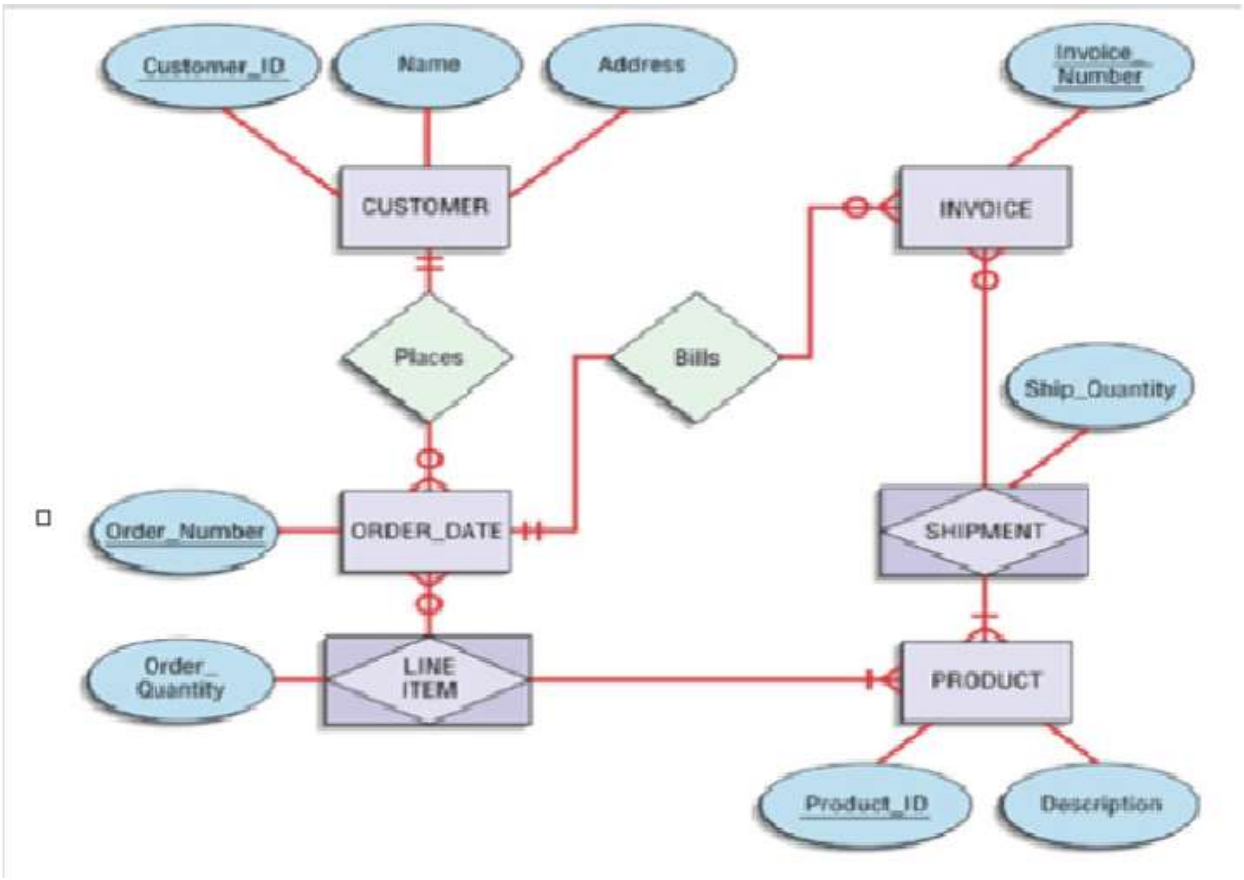
The ABC Company Sales process scenario uses the following set of entities with necessary attributes and it's relationship among the entities are as follows:

List of entities:

i) CUSTOMER
ii) PRODUCT
iii) ORDER
iv) LINE_ITEM
v) INVOICE
vi) SHIPMENT

The Strong entities are: CUSTOMER, PRDOCUT, INVOICE
The weak entities are:

The relationships are : ORDER_DATE , BILLS ,



Answer:

Step 1: Identify the Number of Relations and its' structure.

Relations:
CUSTOMER(Customer_ID,Name,Address)
PRODUCT(Product_ID,Description)
ORDER(Order_Number,Customer_ID,Order_Date)
LINE ITEM(Order_Number,Product_ID,Order_Quantity)
INVOICE(Invoice_Number,Order_Number)
SHIPMENT(Invoice_Number,Product_ID,Ship_Quantity)

Step 2: Design the Tables with its' constraints.

CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount

PRODUCT

Product_ID	Description

ORDER

Order_Number	Customer_ID	Order_Date

LINE_ITEM

Order_Number	Product_ID	Order_Quantity

INVOICE

Invoice_Number	Order_Number

SHIPMENT

Invoice_Number	Product_ID	Ship_Quantity

Note:
Mapping an ER Model into Relational Mode is easy and the vice-versa requires a stuff and care during all phases of it.

Thus a simple rule for converting the ER Model into its's equivalent RELATIONAL model is to strict to the rules of mapping one to another.

****

7. SQL (Structured Query Language)

7.1 Introduction

- SQL is a programming language for Relational Databases.
- It is designed over relational algebra and tuple relational calculus.
- SQL comes as a package with all major distributions of RDBMS.
- SQL comprises both data definition and data manipulation languages.
- Using the data definition properties of SQL, one can design and modify database schema, whereas data manipulation properties allows SQL to store and retrieve data from database.
- Two classes of languages
    - Procedural – user specifies what data is required and how to get those data
    - Nonprocedural – user specifies what data is required without specifying how to get those data.
- SQL is the most widely used query language.

7.1.1 SQL- What is SQL?

- SQL is a standard language for storing, manipulating and retrieving data in databases.
- SQL, **S**tructured **Q**uery **L**anguage, is a programming language designed to manage data stored in relational databases.
- SQL operates through simple, declarative statements.

7.1.2. Capabilities of SQL

- SQL can
  - execute queries against a database
  - retrieve data from a database
  - insert records in a database
  - update records in a database
  - delete records from a database
  - create new databases
  - create new tables in a database
  - create stored procedures in a database
  - create views in a database
  - set permissions on tables, procedures, and views

7.1.3. SQL at a Glance

- **SQL:** ***widely used non-procedural language***
  - E.g. find ***the name of the customer*** with customer-id 192-83-7465

select ***customer.customer-name***
from ***customer***
where ***customer.customer-id = '192-83-7465'***

  - E.g. find the balances of all accounts held by the customer with customer-id 192-83-7465

select ***account.balance***
from ***depositor, account***
where ***depositor.customer-id = '192-83-7465' and***
depositor.account-number = account.account-number

- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g. ODBC/JDBC) which allow SQL queries to be sent to a database

- This keeps data accurate and secure, and it helps maintain the integrity of databases, regardless of size.
    - SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.
- **SQL used in DBMS are**:
    - MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.

7.1.4 SQL Types

- There are 4 types of SQL statements :
    1) **DDL** (Data Definition Language)
    2) **DML** (Data Manipulation Language)
    3) **TCL** (Transaction Control Language)
    4) **DCL** (Data Control Language)

1) Data Definition Language (DDL)

- **Data Definition Language (DDL)** statements are used to define the database structure or schema.

- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
    - database schema
    - Data *storage and definition language*
        - language in which the storage structure and access methods used by the database system are specified
        - Usually an extension of the data definition language

List of DDL commands:

1) CREATE - to create objects (Database/Table/View …) in the database.
2) ALTER - alters the structure of the database (Database/Table/View …).
3) TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed.
    - deletes all the records in the table not the structure.
4) DROP - delete objects from the database(Database/Table/View …) .

5) COMMENT - add comments to the data dictionary.
   - A Non-executable statement for giving comments
6) RENAME - rename an object.
   - Allows to change the name of table /View.

1) CREATE TABLE

- CREATE TABLE creates a new table in the database.
- It allows you to specify the name of the table and the name of each column in the table.
- Specification notation for defining the database schema
- Also databases, and views from RDBMS.

Syntax :

- CREATE TABLE table_name (column_1 datatype, column_2 datatype, column_3 datatype);

Example 1:

Create database tutorials;
Create table article;
Create view for_students;

Example 2:

Create database bank;

create table *account (*
account-number ***char(10),***
balance ***integer)***

2) ALTER TABLE

- ALTER TABLE table_name ADD column_name datatype;
- ALTER TABLE table_name
  ADD column_name datatypeor

- ALTER TABLE table_name
  DROP COLUMN column_name

3) TRUNCATE TABLE

- TRUNCATE command will delete all the records of a selected table and the structure will be remain.
- TRUNCATE TABLE table_name;

4) DROP TABLE

- DROP command will delete the structure and all the records of a selected table.
- Drops commands, views, tables, and databases from RDBMS.

Syntax :

- DROP TABLE table_name
- Drop object_type object_name;

Example:

Drop database tutorials;
Drop table article;
Drop view for_students;

5) COMMENT

Nal info and meaning to the statements..

Syntax :

COMMENT text;

Example:

COMMENT ‘ EB SYSTEM’;

6) RENAME

To rename or change the name of the given object.

Syntax:

RENAME old_object_name TO new_object_name;

Example:

RENAME emp TO employee;

This statement change the table from emp into employee.

2) Data Manipulation Language

- SQL is equipped with data manipulation language (DML). DML modifies the database instance by inserting, updating and deleting its data.
- DML is responsible for all forms data modification in a database.

Language for accessing and manipulating the data organized by the appropriate data model

- DML also known as query language

- **Data Manipulation Language (DML)** statements are used for managing data within schema objects.

TYPES:

1) SELECT - retrieve data from the a database
2) INSERT - insert data into a tabl

3) UPDATE - updates existing data within a table,
4) DELETE - deletes all records from a table, the space for the records remain

SQL contains the following set of commands in its DML section −

a) SELECT/FROM/WHERE
b) INSERT INTO/VALUES
c) UPDATE/SET/WHERE
d) DELETE FROM/WHERE

These basic constructs allow database programmers and users to enter data and information into the database and retrieve efficiently using a number of filter options.

a) SELECT/FROM/WHERE

The SQL SELECT Statement

- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the result-set.

SELECT Syntax

– SELECT *column1*, *column2*, ...
FROM *table_name*
WHERE *condition*
ORDER BY *column1, column2, ...* ASC|DESC
GROUP BY column_name
HAVING condition;

– Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table,

Use the following syntax for simple query:

– SELECT * FROM *table_name*;

Different Clauses of SELECT Command

a) SELECT clause − This is one of the fundamental query command of SQL. It is similar to the projection operation of relational algebra. It selects the attributes based on the condition described by WHERE clause.

b) FROM from clause − This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given, this clause corresponds to Cartesian product.

c) WHERE clause − This clause defines predicate or conditions, which must match in order to qualify the attributes to be projected.

Example:

Select author_name
From book_author
Where age > 50;

This command will yield the names of authors from the relation book_author whose age is greater than 50.

b) INSERT INTO/VALUES

This command is used for inserting values into the rows of a table (relation).

- The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax

- It is possible to write the INSERT INTO statement in two ways.
- The first way specifies both the column names and the values to be inserted:
  - INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
    VALUES (*value1*, *value2*, *value3*, ...);
    - If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query.
  - INSERT INTO *table_name*
    VALUES (*value1*, *value2*, *value3*, ...);

Example:

INSERT INTO tutorials (Author, Subject) VALUES ("anonymous", "computers");

c) UPDATE/SET/WHERE

This command is used for updating or modifying the values of columns in a table (relation).

- The UPDATE statement is used to modify the existing records in a table.

UPDATE Syntax

- UPDATE *table_name*
  SET *column1 = value1, column2 = value2, ...*
  WHERE *condition*;
- **Note:**
  - Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement.
  - The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

Example:

UPDATE tutorials SET Author="webmaster" WHERE Author="anonymous";

d) DELETE/FROM/WHERE

This command is used for removing one or more rows from a table (relation).
The DELETE statement is used to delete existing records in a table.

DELETE Syntax

- DELETE FROM *table_name [*WHERE *condition]*;
- **Note:**
  - Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement.
  - The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

Example:

DELETE FROM tutorials
WHERE Author="unknown";

3) TCL (Transaction Control Language) Commands

- **Transaction Control (TCL)** statements are used to manage the changes made by DML statements.
- It allows statements to be grouped together into logical transactions.
- A transaction is a sequence of SQL statements that Oracle treats as a single unit.
- Results of Data Manipulation Language (DML) are not permanently updated to table until explicit or implicit COMMIT occurs
- Transaction control statements can:
    - Commit data through COMMIT command
    - Undo data changes through ROLLBACK command

a) COMMIT - save work done
b) SAVEPOINT - identify a point in a transaction to which you can later roll back
c) ROLLBACK - restore database to original since the last COMMIT
d) SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

a) COMMIT

- Explicit COMMIT occurs by executing COMMIT;
- Implicit COMMIT occurs when DDL command is executed or user properly exits system.
- Permanently updates table(s) and allows other users to view changes.
- This statement also erases all savepoints in the transaction and releases the transaction's locks.

Syntax

- COMMIT [WORK]
- Where WORK is supported for compliance with standard SQL.
- The statements COMMIT and COMMIT WORK are equivalent.

b) ROLLBACK

- Used to “undo” changes that have not been committed

- Occurs when:
  - ROLLBACK; is executed
  - System restarts after crash

Syntax

- ROLLBACK [WORK | TO savepoint]
- Where WORK is optional.
- If savepoint name is given, rolls back the current transaction to the specified savepoint. If you omit this clause, the ROLLBACK statement rolls back the entire transaction.

Note:
Using ROLLBACK **without** the TO SAVEPOINT clause performs the following operations:

a) Ends the transaction.
b) Undoes all changes in the current transaction
c) Erases all savepoints in the transaction
d) Releases the transaction's locks

Using ROLLBACK **with** the TO SAVEPOINT clause performs the following operations:

a) Rolls back just the portion of the transaction after the savepoint.
b) Erases all savepoints created after that savepoint. The named savepoint is retained, so you can roll back to the same savepoint multiple times. Prior savepoints are also retained.
c) Releases all table and row locks acquired since the savepoint. Other transactions that have requested access to rows locked after the savepoint must continue to wait until the transaction is committed or rolled back. Other transactions that have not already requested the rows can request and access the rows immediately.

Examples

```
Create table temp_table (t1  number(4));
Rollback;
Describe temp_table
Insert into temp_table (t1)
```

```
Values(10);
Select * from temp_table;
Commit;
```

- A normal exit from most Oracle utilities and tools causes the current transaction to be committed. (by giving Quit SQL * PLUS command).
- If the transaction do not explicitly committed and the program terminates abnormally, the last uncommitted transaction is automatically rolled back.

c) SAVEPOINT

- Identifies a point in a transaction to which you can later roll back.

Syntax

- SAVEPOINT save_point;
- Where save_point is the name of the savepoint to be created.

Example:

- To update BLAKE's and CLARK's salary, check that the total company salary does not exceed 2,7,00, then reenter CLARK's salary,
  enter:

```
UPDATE emp     SET sal = 2000     WHERE ename = 'BLAKE';
SAVEPOINT blake_sal;
UPDATE emp     SET sal = 1500     WHERE ename = 'CLARK';
SAVEPOINT clark_sal;
SELECT SUM(sal) FROM emp;
ROLLBACK TO SAVEPOINT blake_sal;
UPDATE emp     SET sal = 1200     WHERE ename = 'CLARK'; COMMIT;
```

4) Data Control Language (DCL)

- **Data Control Language (DCL)** statements gives permission(s) and if not necessary revokes or collect back those granted permission(s)..

TYPES:-

a) GRANT - gives user's access privileges to database
b) REVOKE - withdraw access privileges given with the GRANT command

a) GRANT statement

This command is related to access right and /or revoking to / from various objects of DBMS.

Syntax:

GRANT privilege_name TO user;

This command gives access right called only CREATE privilege to the user scott.

Example:

GRANT CREATE TO scott;

b) REVOVE statement

This command is related to revoking privilege(s) from various objects of DBMS.

Syntax:

REVOKE privilege_name FROM usert;

This command gives access right called only CREATE privilege to the user scott.

Example:

REVOKE CREATE FROM scott;

Thus the 4 types of SQL statements are used by any DBMS can able to complete its major tasks.

****

8. NORMALIZATION

8.1 Introduction

What is Normalization?

Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

- Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anamolies.
- It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.is a technique of organizing the data in the database.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anamolies.
- It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

What do you mean by normalization?

In relational database design, the process of organizing data to minimize redundancy is called Normolization..

Normalization usually involves dividing a database into two or more tables and defining relationships between the tables.

Normalization is the process of reducing the duplication of the data."NF" refers to "normal form" .

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

The three main **types** of **normalization** are 1NF,2NF,3NF.
Normalization is also known as data **normalization**.
Normalization is a data compression idea. Basically you do not want store duplicated information in a database.

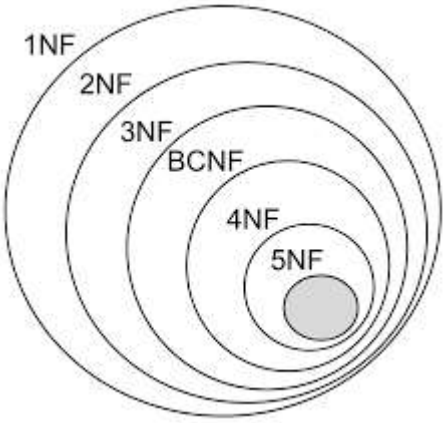
Purpose:

Normalization is used for mainly two purposes,

- Eliminating reduntant(useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

8.2 Types of Normalization

- Normalization usually involves dividing a database into two or more tables and defining relationships between the tables.
- The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.



There are three main normal forms, each with increasing levels of normalization:

a) **First Normal Form (1NF):**
Each field in a table contains different information. For example, in an employee list, each table would contain only one birthdate field.

What is 1NF in DBMS?

First Normal Form (1NF)

Rule : A table is said to be in First Normal Form (**1NF**) if and only if each attribute of the relation is atomic.
That is, Each row in a table should be identified by primary key (a unique column value or group of unique column values) No rows of data should have repeating group of column values.
An attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

For a table to be in the First Normal Form, it should follow the following 4 rules:
1. It should only have single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.
This table is **not in 1NF** as the rule says "each attribute of a table must have atomic (single) values", the emp_mobile values for employees Jon & Lester violates that rule.
To make the table complies with 1NF we should have the data like this:

emp_id	emp_name	emp_address	*emp_mobile*
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123

			8123450987

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

This table is **not in 1NF** as the rule says "each attribute of a table must have atomic (single) values", the emp_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

b) **Second Normal Form (2NF):**

Each field in a table that is not a determiner of the contents of another field must itself be a function of the other fields in the table.

For a table to be in the Second Normal Form,

1) It should be in the First Normal form.
2) And, it should not have Partial Dependency.

In other words,

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.
- For a table to be in the Second Normal form, it should be in the First Normal form and it should not have Partial Dependency.

- Partial Dependency exists, when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key.
- To remove Partial dependency, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

Example:
Suppose a school wants to store the data of teachers and the subjects they teach.
They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}
Non prime attribute: teacher_age

- The table is in 1 NF because each attribute has atomic values.
- However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key.
- This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

To make the table complies with 2NF we can break it in two tables like this:
teacher_details table:

teacher_id	teacher_age
111	38
222	38

333	40

teacher_subject table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	Subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}

Non prime attribute: teacher_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

To make the table complies with 2NF we can break it in two tables like this:

teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	Subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

c) **Third Normal Form (3NF):**

No duplicate information is permitted.

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
Candidate Keys: {emp_id}
Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.
Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.
To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:
employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008

1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
Candidate Keys: {emp_id}

Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.

Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

c) Boyce Codd normal form (BCNF)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in **more than one**

department.
They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}
The table is not in BCNF as neither emp_id nor emp_dept alone are keys.
To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
Stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.

While normalization makes databases more efficient to maintain, they can also make them more complex because data is separated into so many different tables.

****

8. TRANSACTION MANAGEMENT

8.1 Transaction Concepts

- A ***transaction*** is a *unit* of program execution that accesses and possibly updates various data items.

A *transaction is a collection of operations that performs a single logical* function in a database application

- A transaction must see a consistent database.

- During transaction execution the database may be inconsistent.
- When the transaction is committed, the database must be consistent.

- If the transaction aborted, the DB must be restored to its prior state. Means such transaction must be undone or rolled back
- Two main issues to deal with:
  - **Failures of various kinds**, such as hardware failures and system crashes
  - **Concurrent execution** of multiple transactions

8.1.1 Transaction Management

- *A **transaction** is the DBMS's abstract view of a user program:* a series of reads/writes of database objects
- Users submit transactions, and can think of each transaction as executing by itself
  - The concurrency is achieved by the DBMS, which interleaves actions of the various transactions.
- *Issues:*
  - Interleaving transactions, and
  - Crashes!

•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.

8.2 Transaction Management Goal: The ACID properties

- **Atomicity**: Either all actions are carried out, or none are
- **Consistency**: If each transaction is consistent, and the database is initially consistent, then it is left consistent

- **Isolation**: Transactions are isolated, or protected, from the effects of other scheduled transactions
- **Durability**: If a transactions completes successfully, then its effects persist

Atomicity

- A transaction can
  - **Commit** after completing its actions, or
  - **Abort** because of
    - Internal DBMS decision: restart
    - System crash: power, disk failure, …
    - Unexpected situation: unable to access disk, data value, …
- A transaction interrupted in the middle could leave the database inconsistent
- DBMS needs to remove the effects of partial transactions to ensure **atomicity**: either all a transaction's actions are performed or none

- A DBMS ensures atomicity by **undoing** the actions of partial transactions
- To enable this, the DBMS maintains a record, called a **log**, of all writes to the database
- The component of a DBMS responsible for this is called the **recovery manager**

Consistency

- Users are responsible for ensuring transaction consistency
  - when run to completion against a consistent database instance, the transaction leaves the database consistent
- For example, consistency criterion that my inter-account-transfer transaction does not change the total amount of money in the accounts!
- **Database consistency** is the property that every transaction sees a consistent database instance.
  - It follows from transaction atomicity, isolation and transaction consistency

Isolation

- Guarantee that even though transactions may be interleaved, the net effect is identical to executing the transactions **serially**
- For example, if transactions T1 and T2 are executed concurrently, the net effect is equivalent to executing
  - T1 followed by T2, **or**

  - T2 followed by T1
- NOTE: The DBMS provides **no guarantee of effective order of execution**

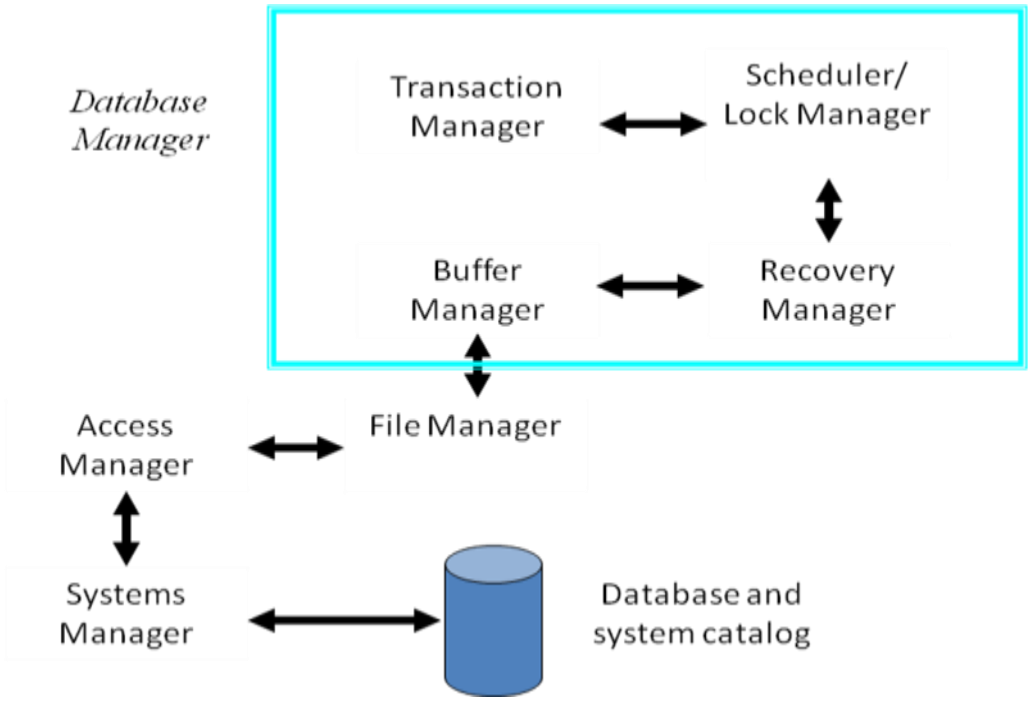
Durability

- DBMS uses the log to ensure durability
- If the system crashed before the changes made by a completed transaction are written to disk, the log is used to remember and restore these changes when the system is restarted
- Again, this is handled by the recovery manager

8.3 Transaction Management with SQL

- **COMMIT** statement – *ends the SQL trans*.; effects permanently recorded within DB
- **ROLLBACK** statement – DB is *rolled back to its previous consistent state and all the changes are aborted*
- Reach end of the program successfully – similar to COMMIT
- Program abnormally terminated – similar to ROLLBACK

DBMS Transaction Subsystem



- ***Trans. Mgr.*** coordinates transactions on

behalf of application program. It communicates with scheduler.

- ***Scheduler*** implements a strategy for concurrency control.
- If any failure occurs, ***recovery manager*** handles it.
- ***Buffer manager*** in charge of transferring data between disk storage and main memory.

- ***File manager*** manipulates the underlying storage files and manages the allocation of storage space on disk.
- File manager does not directly manage the physical input and output of data, rather it passes the requests on to the ***access manager***.
- Appropriate access method is used to either read or write data into the ***system manager***.

Transactions and schedules

- A transaction is seen by the DBMS as a series, or list, of actions
  - Includes read and write of objects
  - We'll write this as R(o) and W(o) (sometimes $R_T(o)$ and $W_T(o)$)
- For example

T1: [R(a), W(a), R(c), W(c)]
T2: [R(b), W(b)]

- In addition, a transaction should specify as its final action either **commit**, or **abort**

Schedules

- A **schedule** is a list of actions from a set of transactions
  - A well-formed schedule is one where the actions of a particular transaction T are in the same order as they appear in T
- For example
  - $[R_{T1}(a), W_{T1}(a), R_{T2}(b), W_{T2}(b), R_{T1}(c), W_{T1}(c)]$ is a well-formed schedule
  - $[R_{T1}(c), W_{T1}(c), R_{T2}(b), W_{T2}(b), R_{T1}(a), W_{T1}(a)]$ is not a well-formed schedule
- A **complete schedule** is one that contains an abort or commit action for every transaction that occurs in the schedule
- A **serial schedule** is one where the actions of different transactions are not interleaved

Serialisability

- A **serialisable schedule** is a schedule whose effect on any consistent database instance is identical to that of some complete serial schedule
- NOTE:
  - All different results assumed to be acceptable
  - It's more complicated when we have transactions that abort
  - We'll assume that all 'side-effects' of a transaction are written to the database

Anomalies with interleaved execution

- Two actions on the same data object **conflict** if at least one of them is a write
- Consider three ways in which a schedule involving two consistency-preserving transactions can leave a consistent database inconsistent.

Transaction Log

- Keep track of all transactions that update the DB
- If failure occurs, information that was stored here will be used for recovery
- It is triggered by ROLL BACK statement, program abnormal termination, or system failure
- It stotes before-and-after data of the DB and the tables, rows and attribute values that participated in the transaction

- The transaction log is subject to dangers such as disk full conditions and disk crashes
- It has to be managed like other DBs
- Transaction log will increase the processing overhead – but it is worthwhile

Example of Fund Transfer

- **Transaction to transfer $50 from account *A* to account *B*:**

1. read(*A*)
2. *A* := *A* – 50
3. write(*A*)
4. read(*B*)
5. *B* := *B* + 50

6. write(*B*)

- **Consistency requirement – the sum of *A* and *B* is unchanged by the execution of the transaction.**
- **Atomicity requirement — if the transaction fails after step 3 and before step 6, the system should ensure that its updates are not reflected in the database, else an inconsistency will result.**

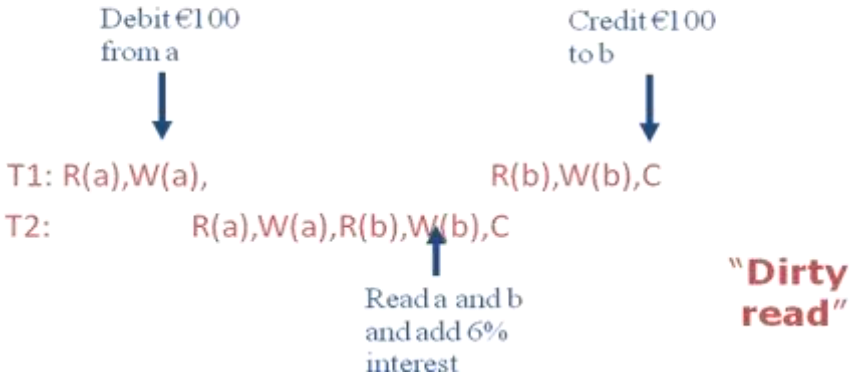
- **Durability requirement — once the user has been notified that the transaction has completed (i.e., the transfer of the $50 has taken place),**
  - **the updates to the database by the transaction must persist despite failures.**
- **Isolation requirement — if between steps 3 and 6, another transaction is allowed to access the partially updated database, it will see an inconsistent database (the sum *A + B* will be less than it should be).**
  Can be ensured trivially by running transactions *serially*, that is one after the other.
  - **However, executing multiple transactions concurrently has significant benefits (this is not covered in WXES2103)**

A transaction's action(Mgnt.)

- In General,
  - A transaction includes:
    - One or more Read (R) operation(s)
    - One or more Write (W) operation(s)
    - A combination of R & W operations
- During a multiple action in an transaction, a major issues / conflict is that :
  - **RW (Read , Write) conflicts**
  - **WR (Write, Read) conflicts**
  - **WW (Write, Write) conflicts**

WR conflicts

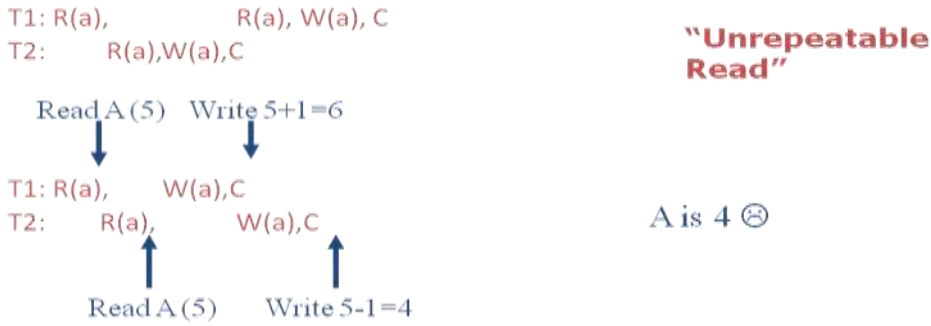
- Transaction T2 reads a database object that has been modified by T1 which has not committed



RW conflicts

RW conflicts

- Transaction T2 could change the value of an object that has been read by a transaction T1, while T1 is still in progress



27

WW conflicts

WW conflicts

- Transaction T2 could overwrite the value of an object which has already been modified by T1, while T1 is still in progress

T1: [W(Britney), W(gmb)] "Set both salaries at £1m"
T2: [W(gmb), W(Britney)] "Set both salaries at $1m"

- But:

T1: W(Britney), W(gmb)
T2: W(gmb), W(Britney)

"Blind Write"
gmb gets £1m
Britney gets $1m
☺

28

Serialisability and aborts

Serialisability and aborts

- Things are more complicated when transactions can abort

Deduct €100 from a

T1:R(a), W(a), **Abort**
T2: R(a),W(a),R(b),W(b),C

Add 6% interest to a and b

Can't undo T2 It's committed ☹

29

Solution 1: Strict two-phase locking

- DBMS enforces the following locking protocol:
  - Each transaction must obtain an S (**shared**) lock before reading, and an X (**exclusive**) lock before writing
  - All locks held by a transaction are released when the transaction completes
  - If a transaction holds an X lock on an object, no other transaction can get a lock (S or X) on that object
- Strict n2PL allows only serialisable schedules

Solution 2: More refined locks

- Some updates that seem at first sight to require a **write (X)** lock, can be given something weaker
  - Example: Consider a seat count object in a flights database
  - There are two transactions that wish to book a flight – get **X** lock on seat count
  - *Does it matter in what order they decrement the count?*
    - They are **commutative actions!**
    - Do they need a write lock?

User's / System's : Aborting action

- If a transaction Ti is aborted, then all actions must be undone
  - Also, if Tj reads object last written by Ti, then Tj must be aborted!
- Most systems avoid **cascading aborts** by releasing locks only at commit time (strict protocols)
  - If Ti writes an object, then Tj can only read this after Ti finishes
- In order to undo changes, the DBMS maintains a **log** which records every write

Recommended Solution:

The log

- The following facts are recorded in the log
  - **"Ti writes an object**": store new and old values
  - **"Ti commits/aborts**": store just a record
- Log records are chained together by transaction id, so it's easy to undo a specific transaction
- Log is often duplexed and archived on stable storage (it's important!)

Recommended Solution:

Connection to Normalization

- The more redundancy in a database, the more locking is required for (update) transactions.
  - **Extreme case:** so much redundancy that all update transactions are forced to execute serially.

- In general, less redundancy allows for greater concurrency and greater transaction throughput.

The Fundamental Tradeoff of Database Performance Tuning

- De-normalized data can often result in faster query response
- Normalized data leads to better transaction throughput

Thus a transaction Management is used to handle transactions effectively.

****