BE23CS405 - DATABASE MANAGEMENT SYSTEMS

Unit: II: Database Design

Entity - Relationship model - E-R Diagrams – ER-to- Relational Mapping – Functional Dependencies – Non-loss Decomposition – First – Second – Third Normal Forms – Dependency Preservation – Boyce/Codd Normal Form – Multi-valued Dependencies and Fourth Normal Form – Join Dependencies and Fifth Normal Form. SQL – Set Operations, Aggregate Functions – GROUPBY – HAVING, Joins, Sub queries, Views, Triggers.

Part I: Entity Relationship Model

Introduction to Entity Relationship Model

Entity Relational model is a model for identifying entities to be represented in the database and representation of how those entities are related.

Let us first understand the design process of database design.

Design Phases

Following are the six steps of database design process. The ER model is most relevant to first three steps

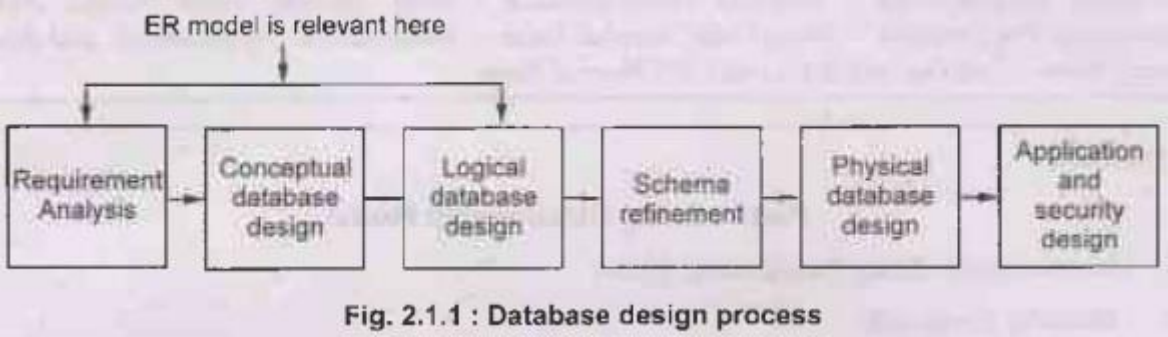


Fig. 2.1.1 : Database design process

Step 1: Requirement analysis:

• In this step, it is necessary to understand what data need to be stored in the database, what applications must be built, what are all those operations that are frequently used by the system.

• The requirement analysis is an informal process and it requires proper communication with user groups.

• There are several methods for organizing and presenting information gathered inthis step.

• Some automated tools can also be used for this purpose.

Step 2: Conceptual database design:

• This is a steps in which E-R Model i.e. Entity Relationship model is built.

• E-R model is a high level data model used in database design.

• The goal of this design is to create a simple description of data that matches with the requirements of users.

Step 3: Logical database design:

• This is a step in which ER model in converted to relational database schema, sometimes called as the logical schema in the relational data model.

Step 4: Schema refinement:

• In this step, relational database schema is analyzed to identify the potential smise problems and to refine it.

• The schema refinement can be done with the help of normalizing and restructuring the relations.

Step 5: Physical database design:

• In this step, the design of database is refined further.

• The tasks that are performed in this step are building indexes on tables and clustering tables, redesigning some parts of schema obtained from earlier design steps.

Step 6: Application and security design:

• Using design methodologies like UML (Unified Modeling Language) the design of the database can be accomplished.

• The role of each entity in every process must be reflected in the application task.

• For each role, there must be the provision for accessing the some part of database and prohibition of access to some other part of database.

• Thus some access rules must be enforced on the application(which is accessing the database) to protect the security features.

ER Model

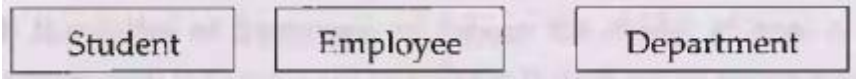
The ER data model specifies enterprise schema that represents the overall logical structure of a database.

The E-R model is very useful in mapping the meanings and interactions of real-world entities onto a conceptual schema.

The ER model consists of three basic concepts -

1) Entity Sets

• **Entity**: An entity is an object that exists and is distinguishable from other objects. For example - Student named "Poonam" is an entity and can be identified by her name. The entity can be concrete or abstract. The concrete entity can be - Person, Book, Bank.The abstract entity can be like - holiday, concept entity is represented as a box.

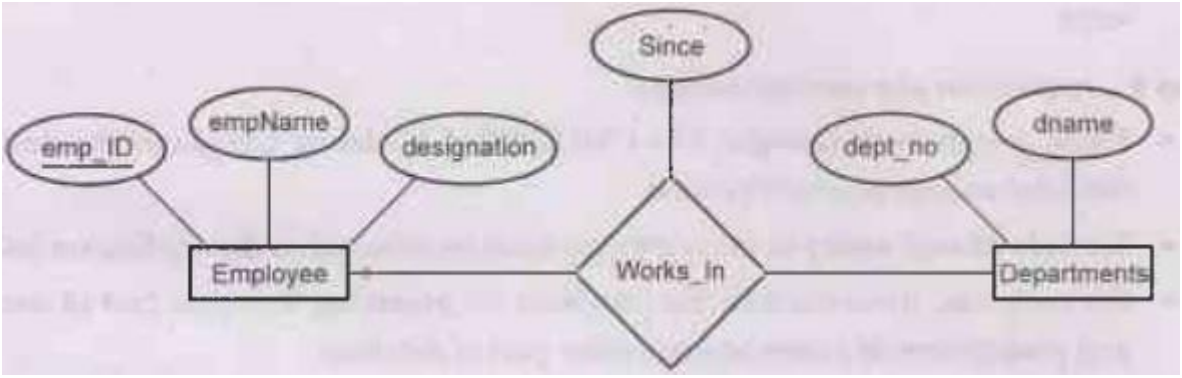


• **Entity set:** The entity set is a set of entities of the same types. For example - All students studying in class X of the School. The entity set need not be disjoint. Each entity in entity set have the same set of attributes and the set of attributes will distinguish it from other entity sets. No other entity set will have exactly the same set of attributes.

2) Relationship Sets

Relationship is an association among two or more entities.

The relationship set is a collection of similar relationships. For example - Following Fig. 2.1.2 shows the relationship works for for the two entities Employee and Departments.



The association between entity sets is called as participation. That is, the entity sets E1, E2,..., En participate in relationship set R.

The function that an entity plays in a relationship is called that entity's role.

3) Attributes

Attributes define the properties of a data object of entity. For example if student is an entity, his ID, name, address, date of birth, class are its attributes. The attributes help in determining the unique entity. Refer Fig. 2.1.3 for Student entity set with attributes - ID, name, address. Note that entity is shown by rectangular box and attributes are shown in oval. The primary key is underlined.

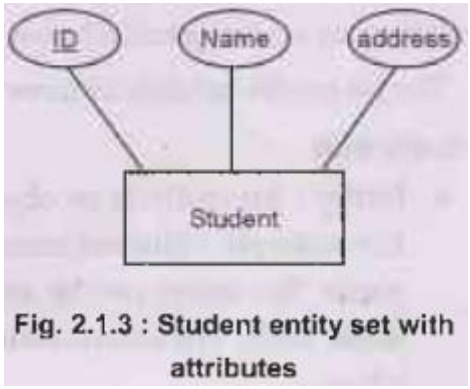


Fig. 2.1.3 : Student entity set with attributes

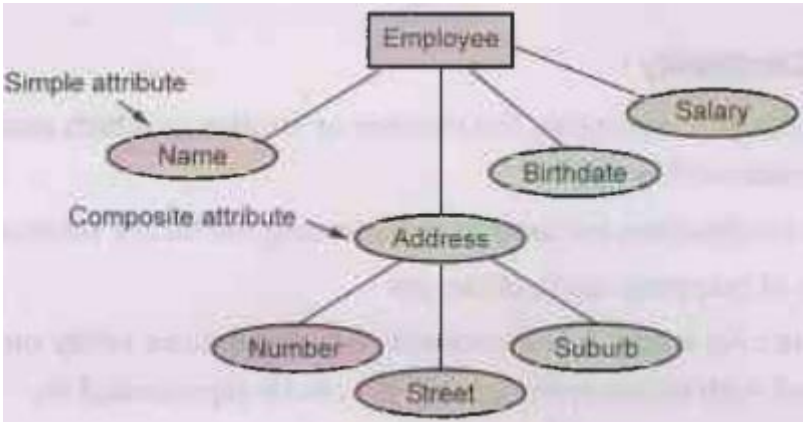
Types of Attributes

1) Simple and Composite Attributes:

1) Simple attributes are attributes that are drawn from the atomic value domains

For example - Name = {Parth}; Age = {23}

1) Composite attributes: Attributes that consist of a hierarchy of attributes For example - Address may consists of "Number", "Street" and "Suburb"→ Address = {59+ 'JM Road' + 'Shivaji Nagar'}



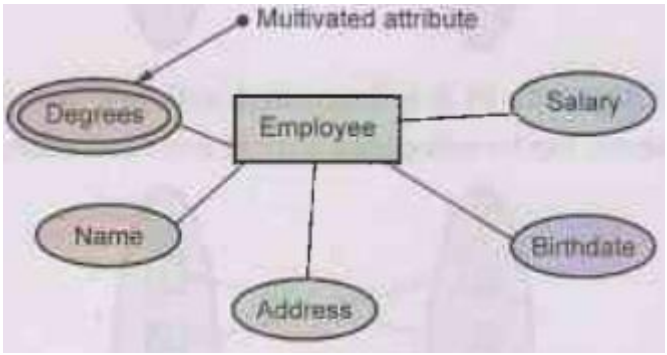
2) Single valued and multivalued:

• There are some attributes that can be represented using a single value. For example

- StudentID attribute for a Student is specific only one studentID.

• Multivalued attributes: Attributes that have a set of values for each entity. It is represented by concentric ovals

For example - Degrees of a person: BSc', 'MTech', 'PhD'



3) Derived attribute:

Derived attributes are the attributes that contain values that are calculated from other attributes. To represent derived attribute there is dotted ellipse inside the solid ellipse. For example Age can be derived from attribute DateOfBirth. In this situation, DateOfBirth might be called Stored Attribute.

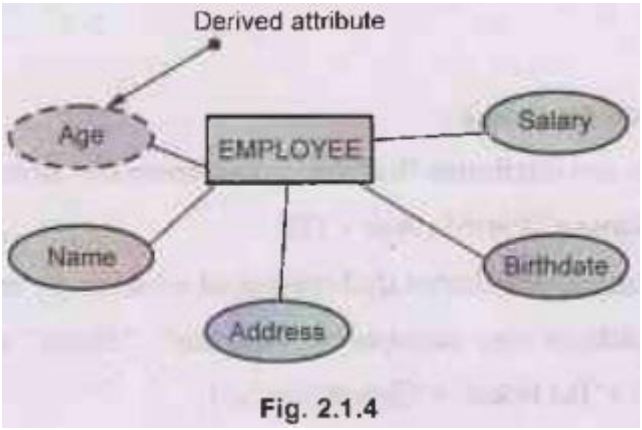


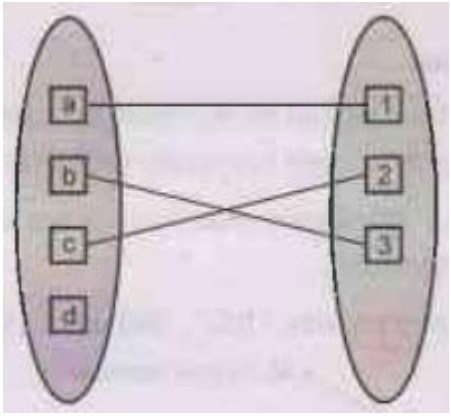
Fig. 2.1.4

Mapping Cardinality

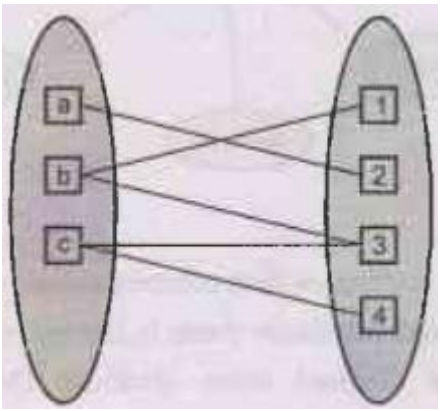
Mapping Cardinality represents the number of entities to which another entity can be associated via a relationship set.

The mapping cardinalities are used in representing the binary relationship sets. Various types of mapping cardinalities are -

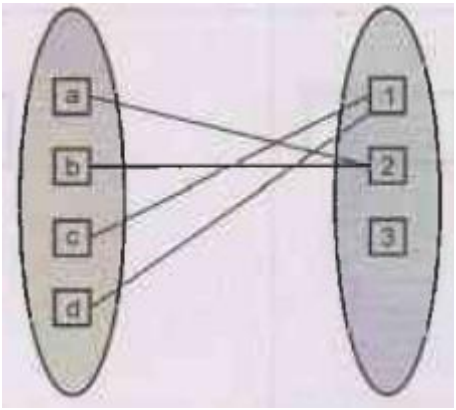
1) One to One: An entity A is associated with at least one entity on B and an entity B is associated with at one entity on A. This can be represented as,



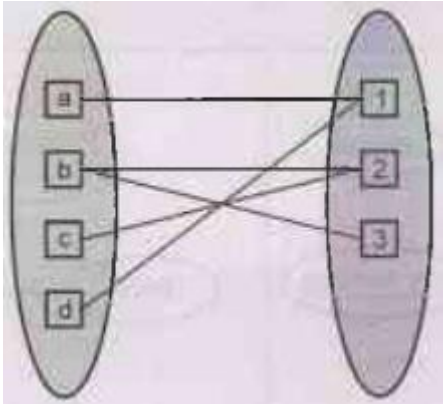
2) One to Many: An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.



3) Many to One: An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A.



4) Many to many: An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



ER Diagrams

An E-R diagram can express the overall logical structure of a database graphically.E-R diagrams are used to model real-world objects like a person, a car, a company and the relation between these real-world objects.

Features of ER model

i) E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).

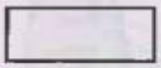
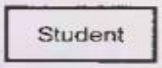
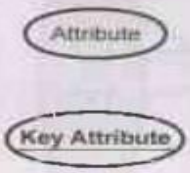
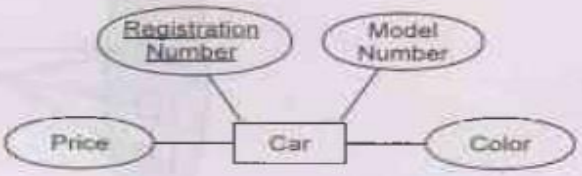
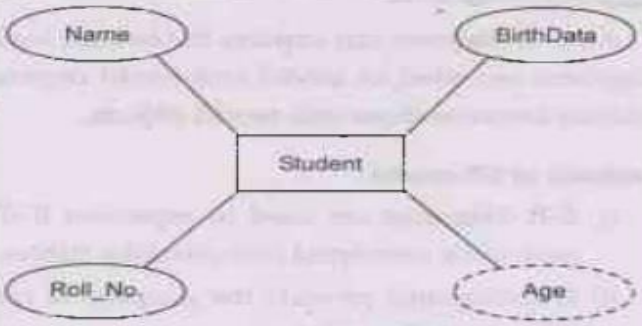
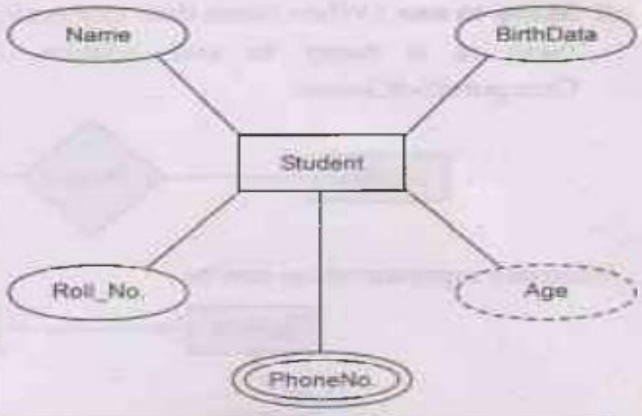
ii) E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.

iii) E-R diagrams require no technical knowledge and no hardware support.

iv) These diagrams are very easy to understand and easy to create even by a naive user.

v) It gives a standard solution of visualizing the data logically.

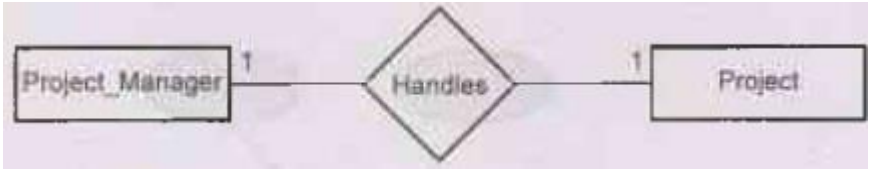
Various Components used in ER Model are-

Component	Symbol	Example
Entity : Any real-world object can be represented as an entity about which data can be stored in a database. All the real world objects like a book, an organization, a product, a car, a person are the examples of an entity.		
Relationship : Rhombus is used to setup relationships between two or more entities.		
Attribute : Each entity has a set of properties. These properties of each entity are termed as attributes. For example, a car entity would be described by attributes such as price, registration number, model number, color etc		
Derived attribute : 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		
Multivalued attribute : An attribute that can hold multiple values is known as multivalued attribute. We represent it with double ellipses in an E-R Diagram. E.g. A person can have more than one phone numbers so the phone number attribute is multivalued.		
Total participation : Each entity is involved in the relationship. Total participation is represented by double lines.		

Mapping Cardinality Representation using ER Diagram

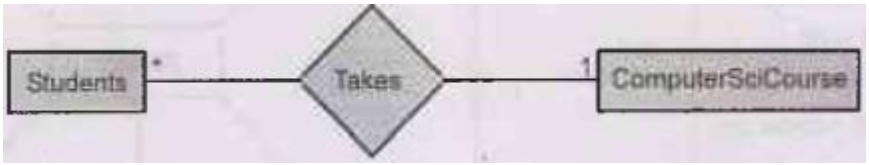
There are four types of relationships that are considered for key constraints.

i) One to one relation: When entity A is associated with at the most one entity B then it shares one to one relation. For example - There is one project manager who manages only one project.

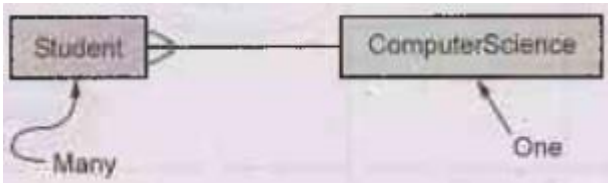


ii) One to many : When entity A is associated with more than one entities at a time then there is one to many relation. For example - One customer places order at a time.

iii) Many to one : When more than one entities are associated with only one entity then there is is many to one relation. For example – Many student take a ComputerSciCourse



Alternate representation can be

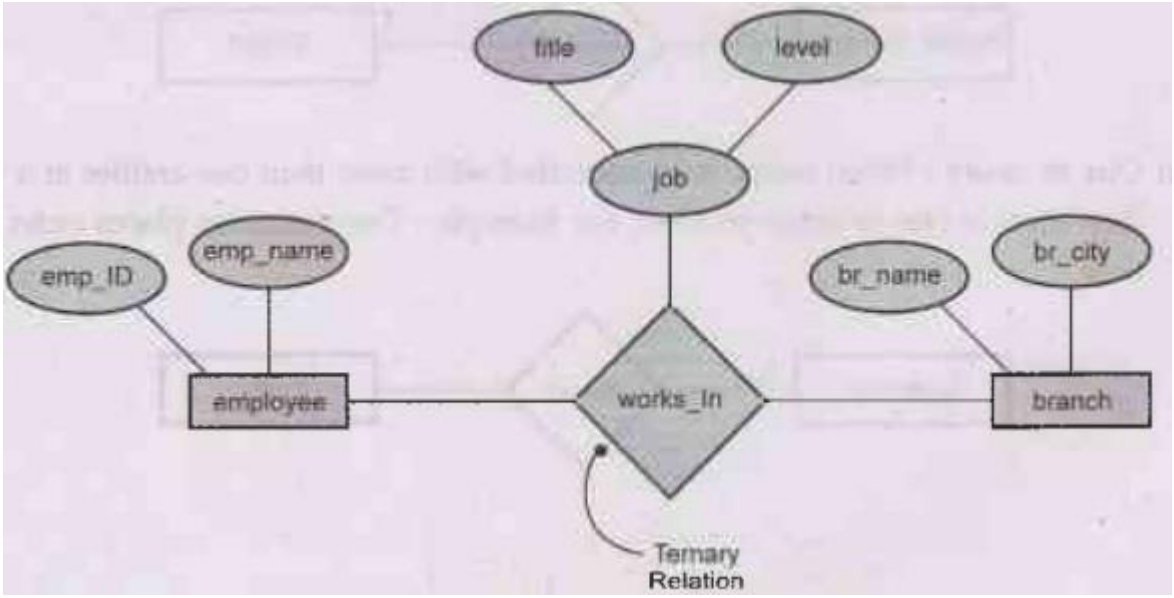


iv) Many to many: When more than one entities are associated with more than one entities. For example - Many teachers can teach many students.

Alternate representation can be

Ternary Relationship

The relationship in which three entities are involved is called ternary relationship. For example -



Binary and Ternary Relationships

• Although binary relationships seem natural to most of us, in reality it is sometimes necessary to connect three or more entities. If a relationship connects three entities, it is called ternary or "3-ary."

• Ternary relationships are required when binary relationships are not sufficient to accurately describe the semantics of an association among three entities.

• For example - Suppose, you have a database for a company that contains the entities, PRODUCT, SUPPLIER, and CUSTOMER. The usual relationships might be PRODUCT/ SUPPLIER where the company buys products from a supplier - a normal binary relationship. The intersection attribute for PRODUCT/SUPPLIER is wholesale_price

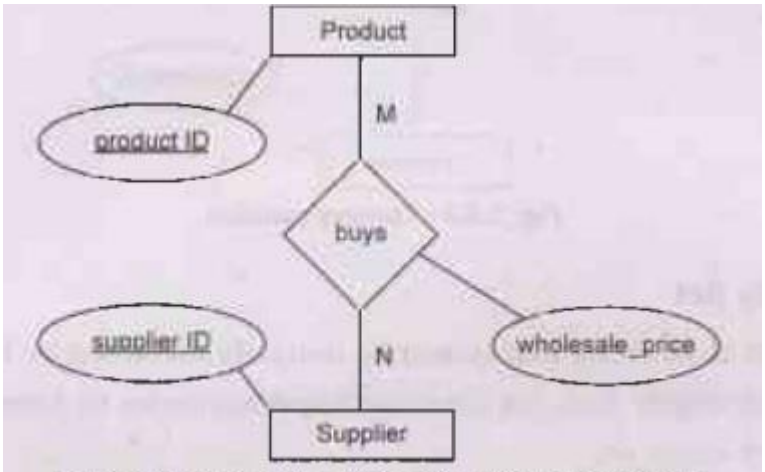


Fig. 2.3.1 : A binary relationship of PRODUCT and SUPPLIER and an intersection attribute, wholesale_price

• Now consider the CUSTOMER entity, and that the customer buys products. If all customers pay the same price for a product, regardless of supplier, then you have a simple binary relationship between CUSTOMER and PRODUCT. For the CUSTOMER/PRODUCT relationship, the intersection attribute is retail_price.

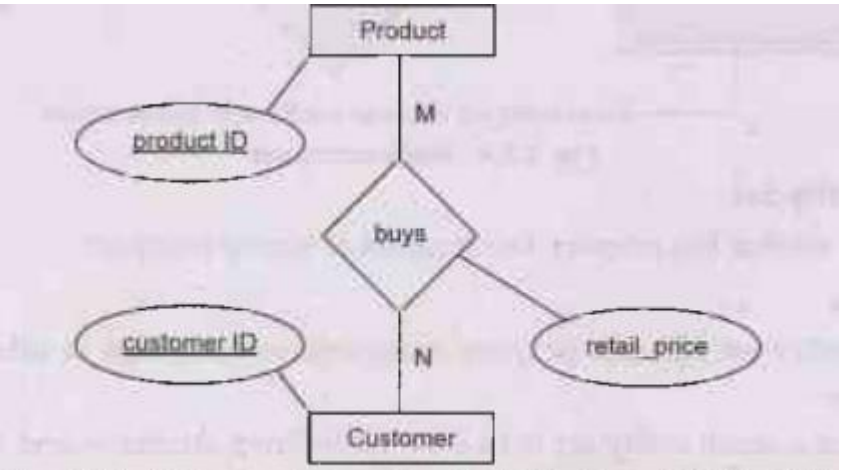


Fig. 2.3.2 : A binary relationship of PRODUCT and CUSTOMER and an Intersection attribute, retail_price

• **Single ternary relation:** Now consider a different scenario. Suppose the customer buys products but the price depends not only on the product, but also on the supplier. Suppose you needed a customerID, a productID, and a supplierID to identify a price. Now you have an attribute that depends on three things and hence you have a relationship between three entities (a ternary relationship) that will have the intersection attribute, price.

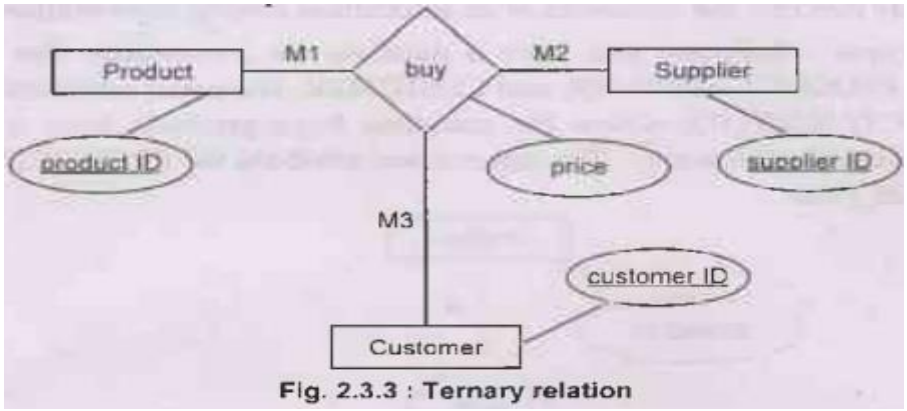


Fig. 2.3.3 : Ternary relation

Weak Entity Set

• A weak entity is an entity that cannot be uniquely identified by its attributes alone. The entity set which does not have sufficient attributes to form a primary key is called as weak entity set.

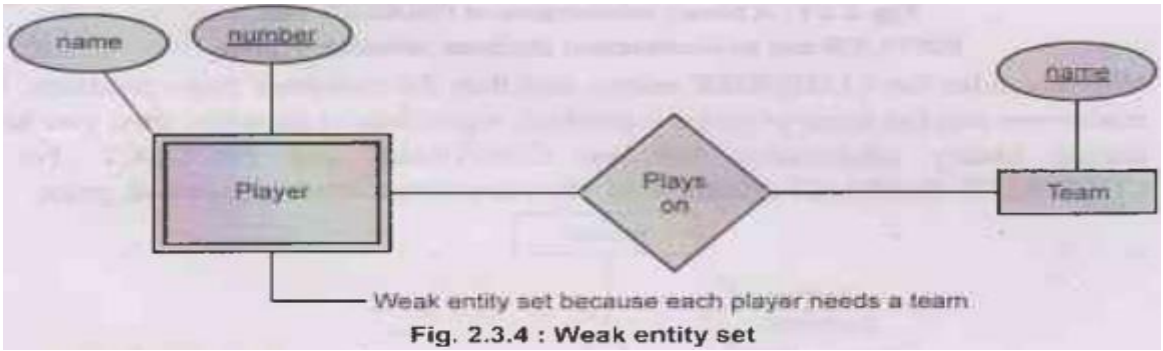


Fig. 2.3.4 : Weak entity set

• **Strong Entity Set**

The entity set that has primary key is called as strong entity set

Weak entity set because each player needs a team

Weak entity rules

• A weak entity set has one or more many-one relationships to other (supporting) entity sets.

• The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets. For example - player-number and team-name is a key for Players.

Difference between Strong and Weak Entity Set

Sr. No.	Strong entity set	Weak entity set
1.	It has its own primary key.	It does not have sufficient attribute to form a primary key on its own.
2.	It is represented by rectangle	It is represented by double rectangle.
3.	It represents the primary key which is underlined.	It represents the partial key or discriminator which is represented by dashed underline.
4.	The member of strong entity set is called as dominant entity set	The member of weak entity set is called subordinate entity set.
5.	The relationship between two strong entity sets is represented by diamond symbol.	The relationship between strong entity set and weak entity set is represented by double diamond symbol.
6.	The primary key is one of the attributes which uniquely identifies its member.	The primary key of weak entity set is a combination of partial key and primary key of the strong entity set.

Examples based on ER Diagram

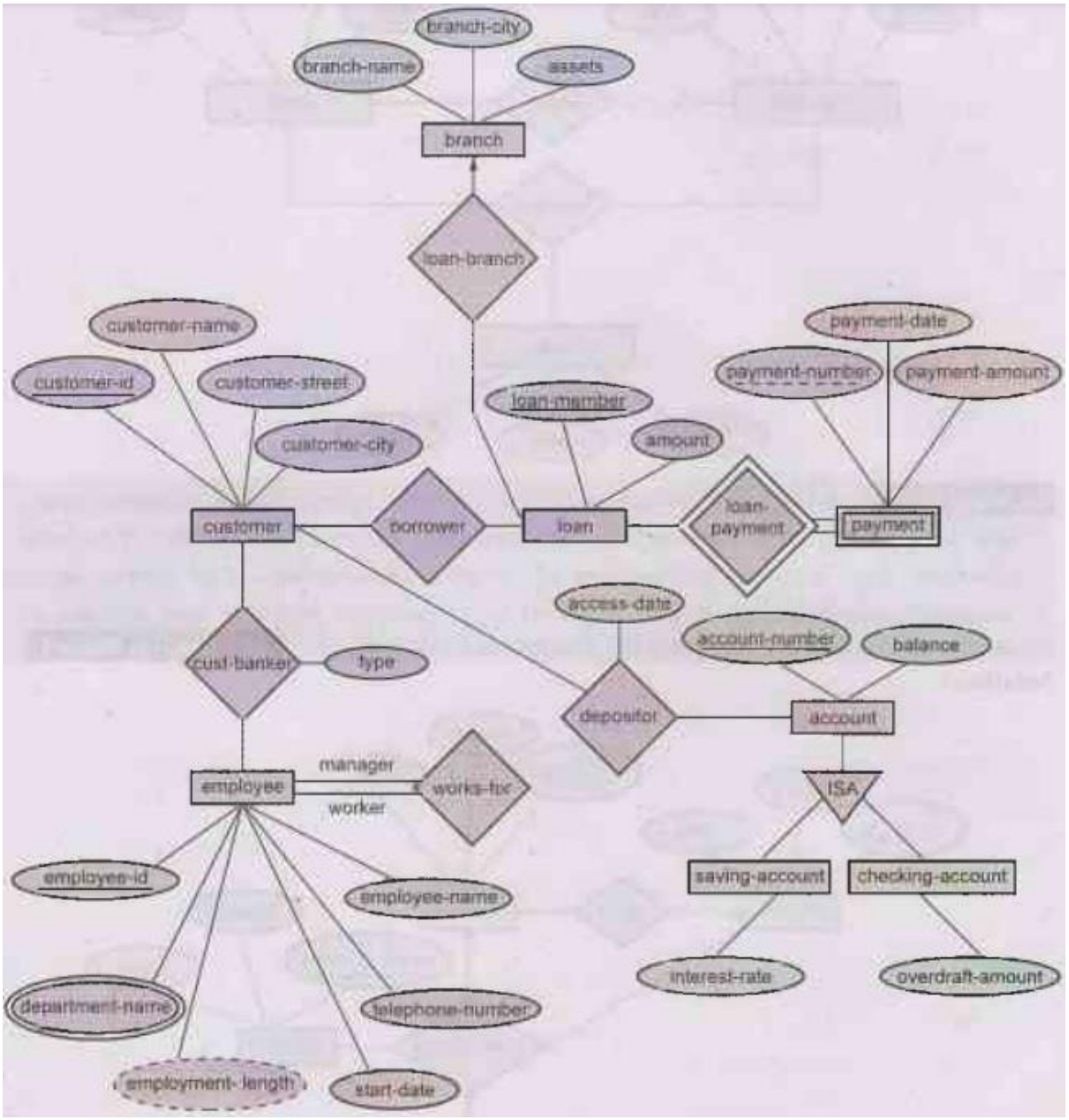
An E-R diagram can express the overall logical structure of a database graphically.

Example 2.5.1 Draw the ER diagram for banking systems (home loan applications). **AU: Dec.-17, Marks 8**

OR Draw an ER diagram corresponding to customers and loans. AU: May.-14, Marks 8

OR Write short notes on: E-R diagram for banking system. AU: Dec.-14, Marks 8

Solution:



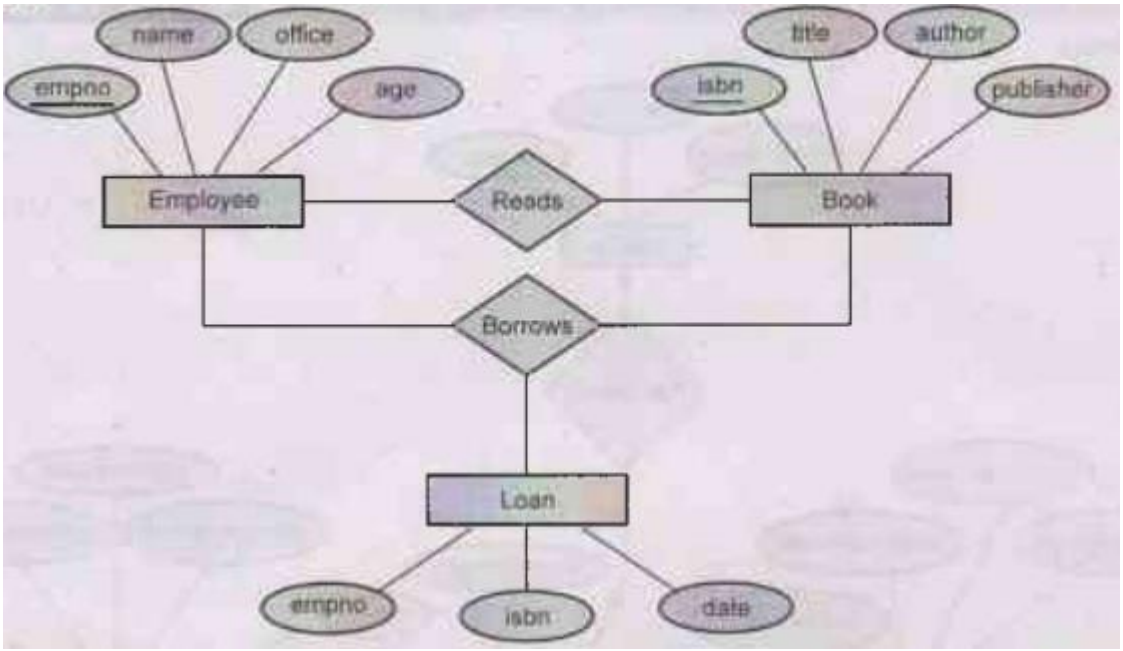
Example 2.5.2 Consider the relation schema given in Figure. Design and draw an ER diagram *that capture the information of this schema.*AU: May-17, Marks 5

Employee(empno,name,office,age)

Books(isbn,title,authors,publisher)

Loan(empno,isbn,date)

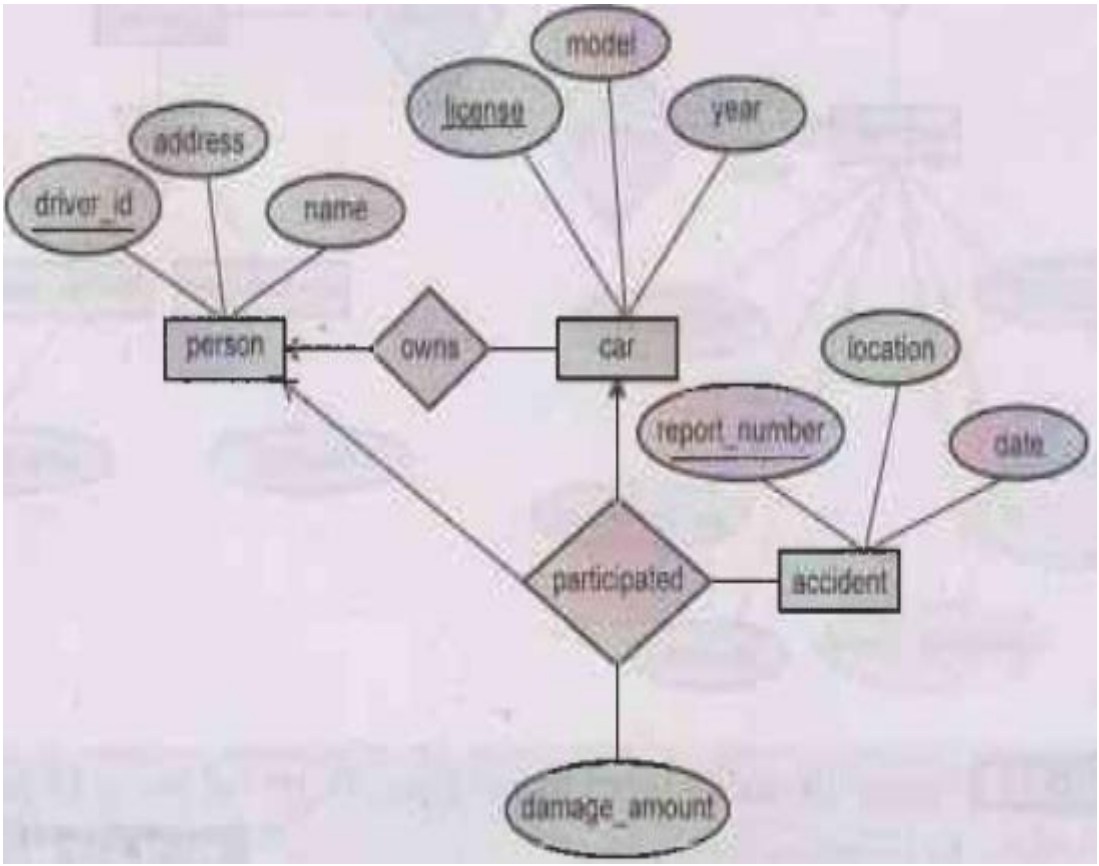
Solution:



Example 2.5.3 *Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for particular period of time and has an associated due date and date when the payment was received*

AU: Dec.-16, Marks 7

Solution:



Example 2.5.4 *A car rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number license number, manufacturer, model, date of purchase and color. Special data are included for certain types of vehicles.*

Trucks: Cargo capacity

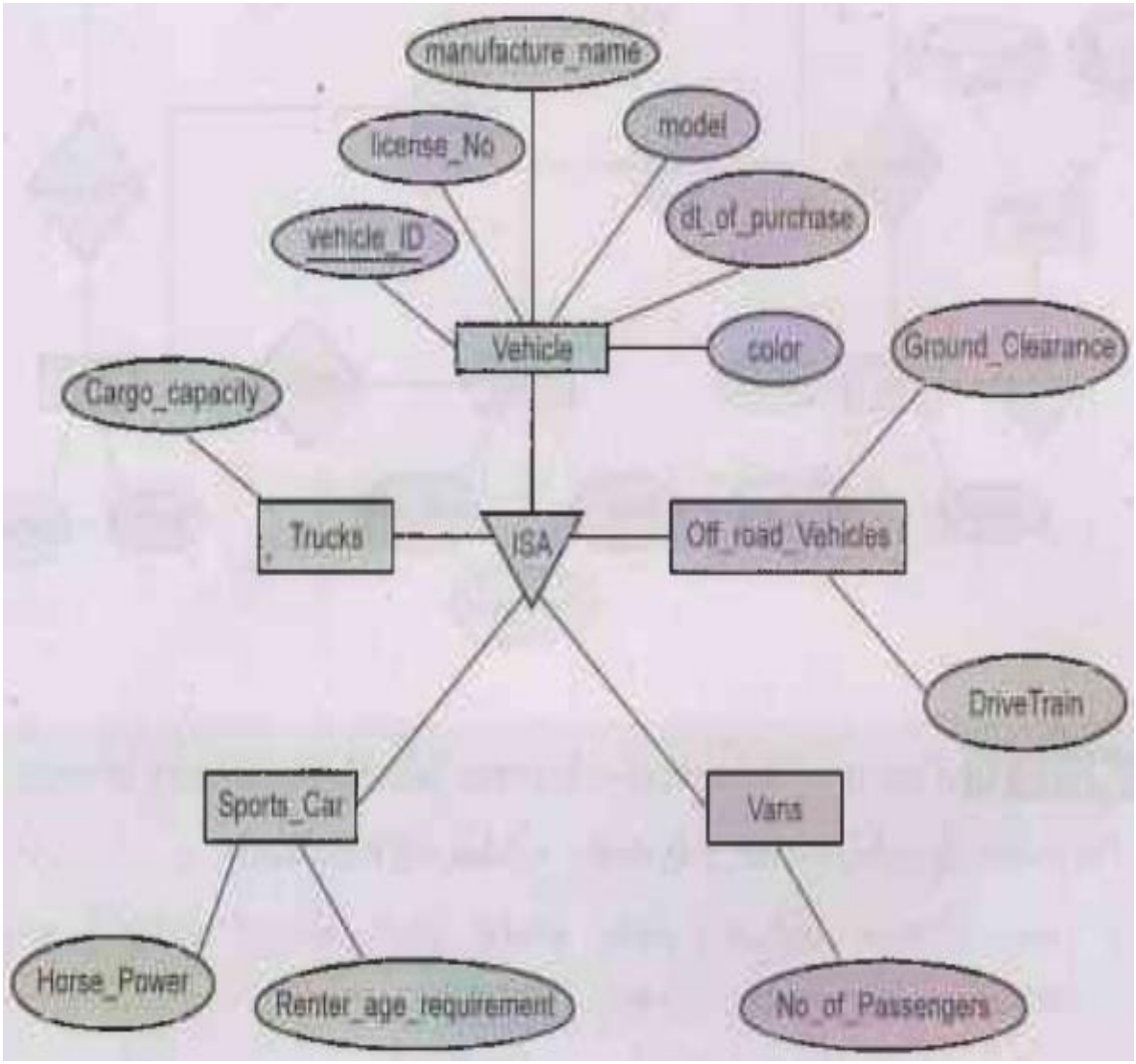
Sports cars: horsepower, renter age requirement

Vans: number of passengers

Off-road vehicles: ground clearance, drivetrain (four-or two-wheel drive)

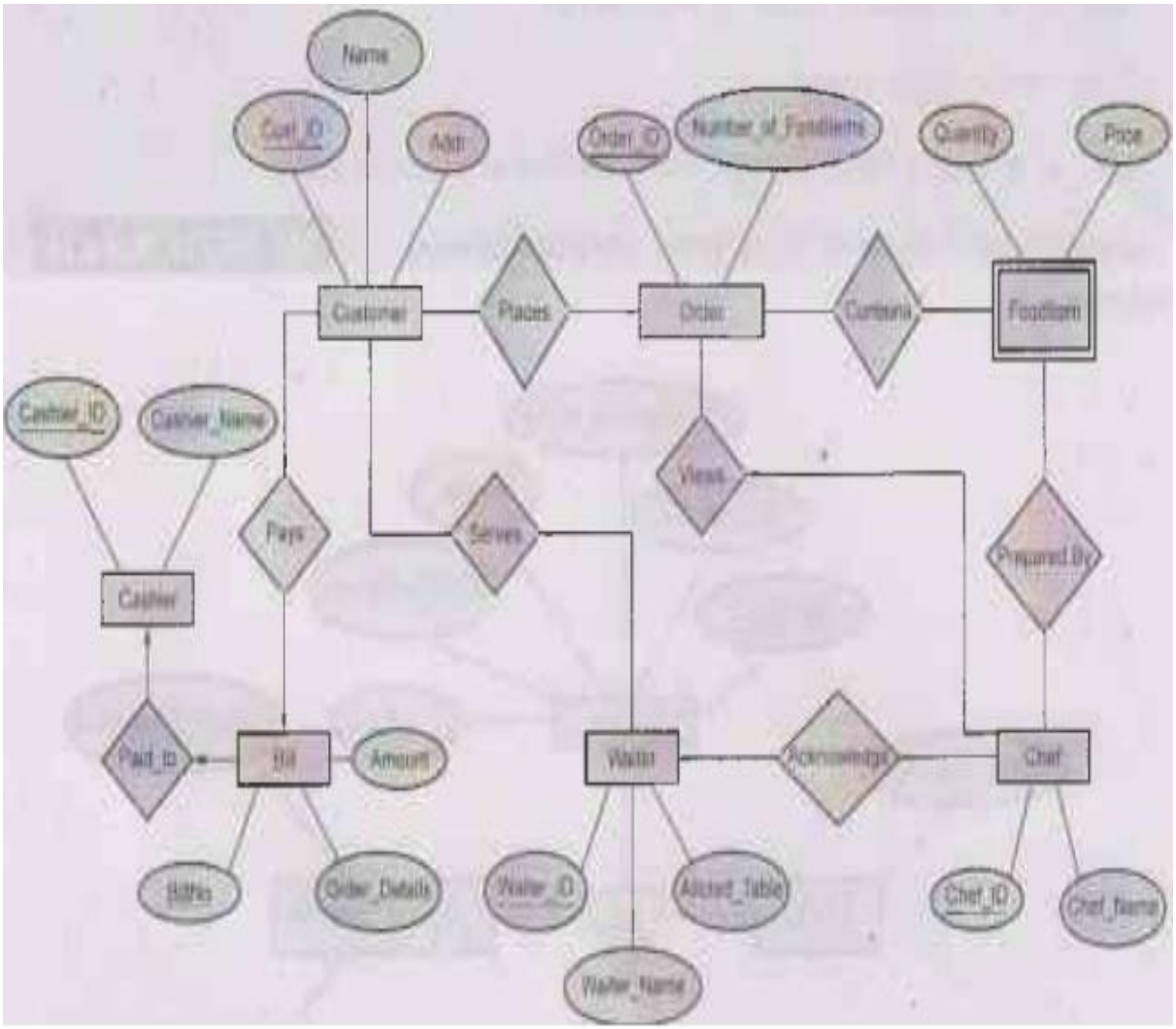
*Construct an ER model for the car rental company database."***AU: Dec.-15, Marks 16**

Solution:



Example 2.5.5 *Draw E-R diagram for the "Restaurant Menu Ordering System", which will facilitate the food items ordering and services within a restaurant. The entire restaurant scenario is detailed as follows. The customer is able to view the food items menu, call the waiter, place orders and obtain the final bill through the computer kept in their table. The Waiters through their wireless tablet PC are able to initialize a table for customers, control the table functions to assist customers, orders, send orders to food preparation staff (chef) and finalize the customer's bill. The Food preparation staffs (chefs), with their touch-display interfaces to the system, are able to view orders sent to the kitchen by waiters. Duringpreparation they are able to let the waiter know the status of each item, and can send notifications when items are completed. The system should have full accountability and logging facilities, and should support supervisor actions to account for exceptional circumstances, such as a meal being refunded or walked out on.*

Solution:



Example 2.5.6 *A university registrar's office maintains data about the following entities:*

(1) courses, including number, title, credits, syllabus, and prerequisites;

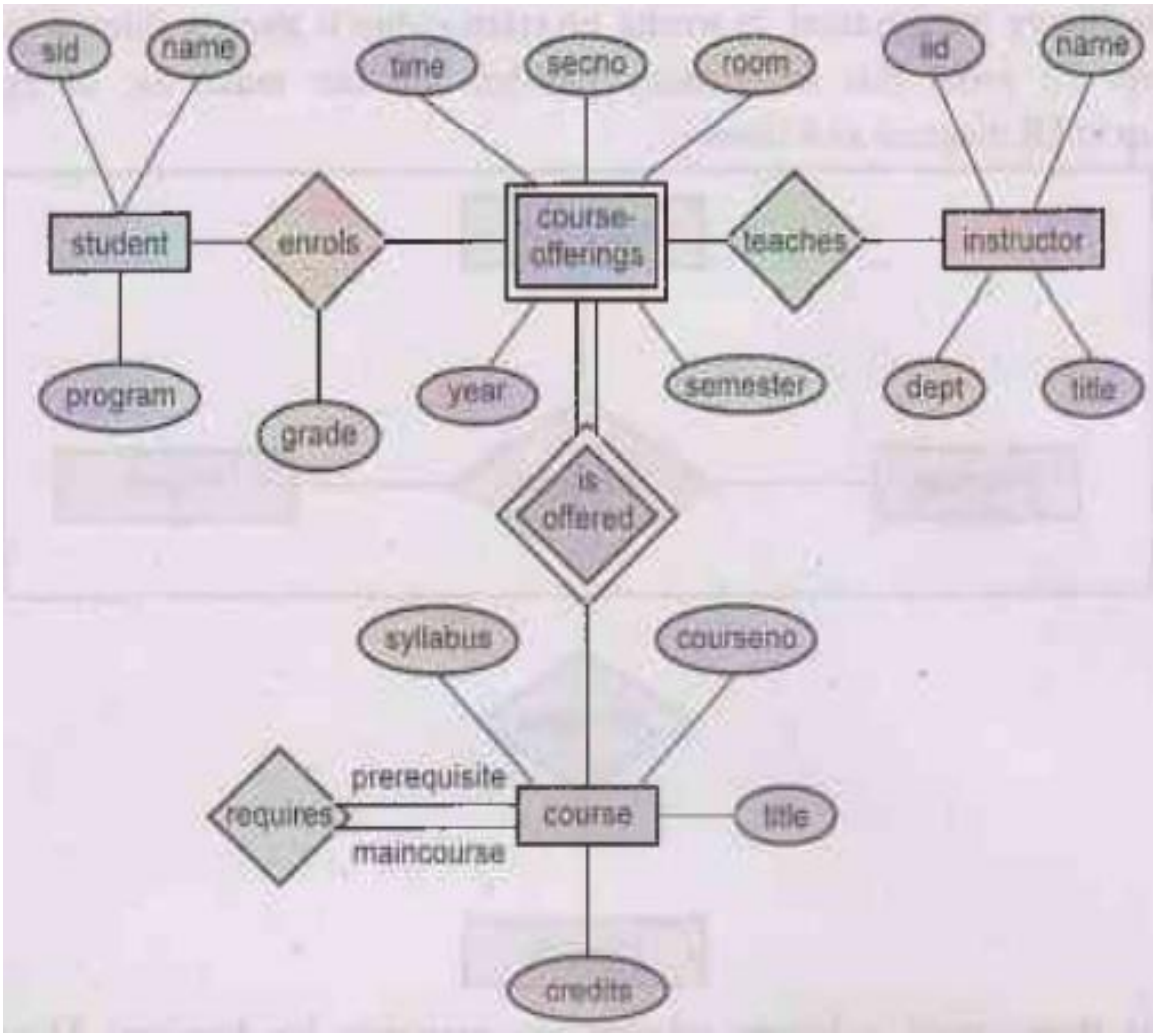
(2) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;

(3) students, including student-id, name, and program; and

(4) instructors, including identification number, name, department, and title.

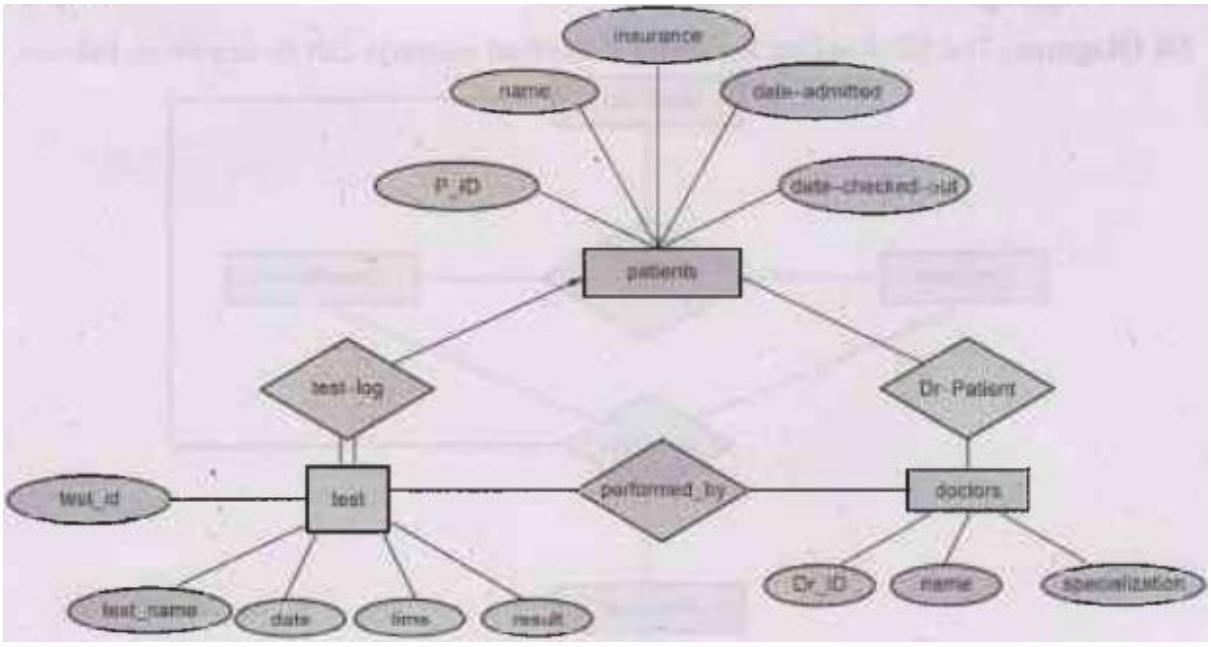
*Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.***AU: Dec.-13, Marks 10**

Solution:



Example 2.5.8 *Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted. AU: Dec.-07, Marks 8*

Solution:



Example 2.5.9 *Consider the following information about a university database:*

i) Professors have an SSN, a name, an age, a rank and a research specialty.

ii) *Projects have a project number, a sponsor name, (e.g. NSF), a starting date, an ending date and a budget.*

iii) *Graduate students have an SSN, a name, an age and a degree program (e.g. M.S. or Ph.D.).*

iv) *Each project is managed by one professor (known as the project's principal investigators)*

v) *Each project is worked on by one or more professors (known as the project's co- investigators).*

vi) *Professors can manage and/or work on multiple projects.*

vii) E*ach project is worked on by one or more graduate students (known as the project's research assistants).*

viii) *When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.*

ix) *Departments have a department number, a department name and a main office.*

x) *Departments have a professor (known as the chairman) who runs the department.*

xi) *Professors work in one or more departments and for each department that they work in, a time percentage is associated with their job,*

xii) *Graduate students have one major department in which they are working on their degree.*

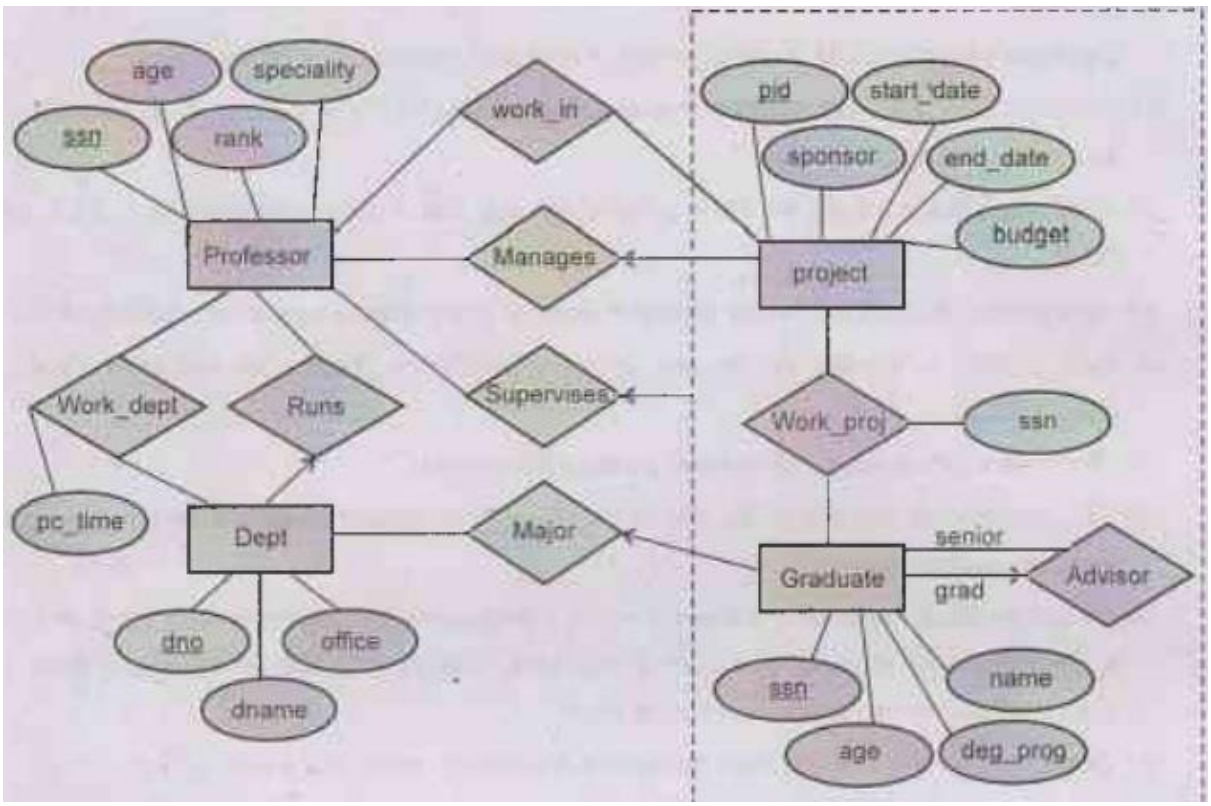
xii) *Each graduate student has another, more senior graduate student (known as a student advisor) who advises him or her on what courses to take.*

xiii) *Design and draw an ER diagram that captures the information about the university.*

Use only the basic ER model here; that is entities, relationship and attributes.

Be sure to indicate any key and participation constraints.

Solution:



ER to Relational Mapping

AU: May-17, Dec.-19, Marks 13

In this section we will discuss how to map various ER model constructs to Relational Model construct.

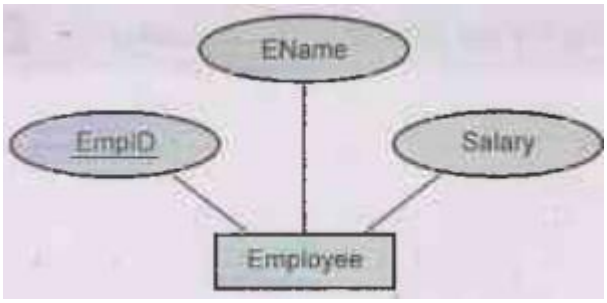
Mapping of Entity Set to Relationship

• An entity set is mapped to a relation in a straightforward way.

• Each attribute of entity set becomes an attribute of the table.

• The primary key attribute of entity set becomes an entity of the table.

• For example - Consider following ER diagram.



The converted employee table is as follows –

EmpID	EName	Salary
201	Poonam	30000
202	Ashwini	35000
203	Sharda	40000

The SQL statement captures the information for above ER diageam as follows -

CREATE TABLE Employee(EmpID CHAR(11),

EName CHAR(30),

Salary INTEGER,

PRIMARY KEY(EmpID))

Mapping Relationship Sets(Without Constraints) to Tables

• Create a table for the relationship set.

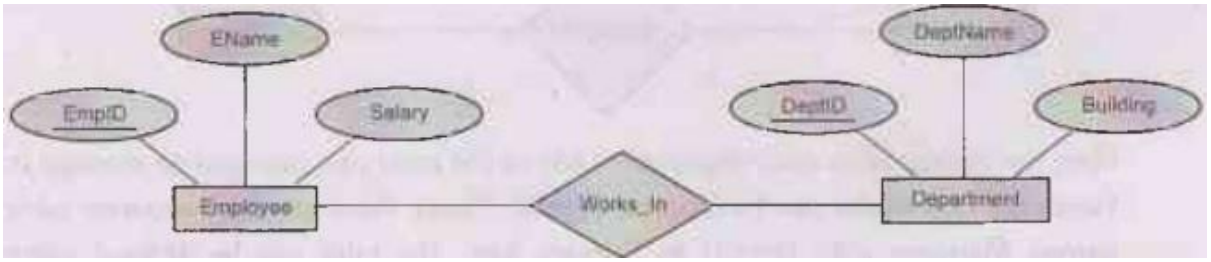
• Add all primary keys of the participating entity sets as fields of the table.

• Add a field for each attribute of the relationship.

• Declare a primary key using all key fields from the entity sets.

• Declare foreign key constraints for all these fields from the entity sets.

For example - Consider following ER model



The SQL statement captures the information for relationship present in above ER diagram as follows -

CREATE TABLE Works In (EmpID CHAR(11),

DeptID CHAR(11),

EName CHAR(30),

Salary INTEGER,

DeptName CHAR(20),

Building CHAR(10),

PRIMARY KEY(EmpID,DeptID),

FOREIGN KEY (EmpID) REFERENCES Employee,

FOREIGN KEY (DeptID) REFERENCES Department

)

Mapping Relationship Sets(With Constraints) to Tables

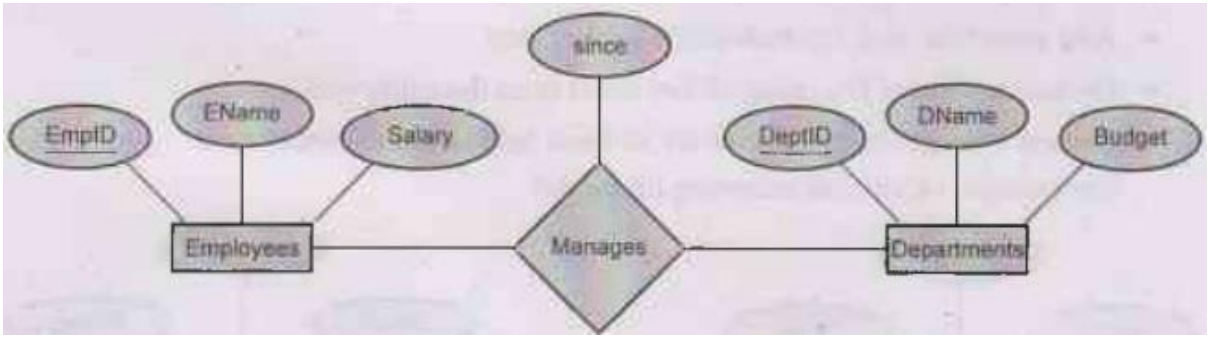
• If a relationship set involves n entity sets and some m of them are linked via arrows in the ER diagram, the key for anyone of these m entity sets constitutes a key for the relation to which the relationship set is mapped.

• Hence we have m candidate keys, and one of these should be designated as the primary key.

• There are two approaches used to convert a relationship sets with key constraints into table.

• **Approach 1:**

• By this approach the relationship associated with more than one entities is separately represented using a table. For example - Consider following ER diagram. Each Dept has at most one manager, according to the key constraint on Manages.



Here the constraint is each department has at the most one manager to manage it. Hence no two tuples can have same DeptID. Hence there can be a separate table named Manages with DeptID as Primary Key. The table can be defined using following SQL statement

CREATE TABLE Manages (EmpID CHAR(11),

DeptID INTEGER,

Since DATE,

PRIMARY KEY (DeptID),

FOREIGN KEY (EmpID) REFERENCES Employees,

FOREIGN KEY (DeptID) REFERENCES Departments)

Approach 2:

• In this approach, it is preferred to translate a relationship set with key constraints.

• It is a superior approach because, it avoids creating a distinct table for the relationship set.

• The idea is to include the information about the relationship set in the table corresponding to the entity set with the key, taking advantage of the key constraint.

• This approach eliminates the need for a separate Manages relation, and queries asking for a department's manager can be answered without combining information from two relations.

• The only drawback to this approach is that space could be wasted if several departments have no managers.

• The following SQL statement, defining a Dep_Mgr relation that captures the information in both Departments and Manages, illustrates the second approach to translating relationship sets with key constraints:

CREATE TABLE Dep_Mgr (DeptID INTEGER,

DName CHAR(20),

Budget REAL,

EmpID CHAR (11),

since DATE,

PRIMARY KEY (DeptID),

FOREIGN KEY (EmpID) REFERENCES Employees)

Mapping Weak Entity Sets to Relational Mapping

• A weak entity can be identified uniquely only by considering the primary key of another (owner) entity. Following steps are used for mapping Weak Entity Set to Relational Mapping

• Create a table for the weak entity set.

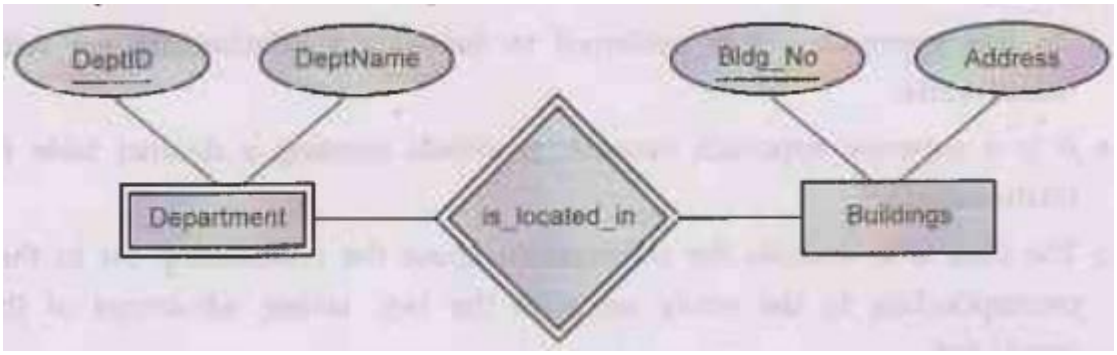
• Make each attribute of the weak entity set a field of the table. AI baris M

• Add fields for the primary key attributes of the identifying owner.

• Declare a foreign key constraint on these identifying owner fields.

• Instruct the system to automatically delete any tuples in the table for which there are no owners

For example - Consider following ER model,



Following SQL Statement illustrates this mapping

CREATE TABLE Department (DeptID CHAR(11),

DeptName CHAR(20),

Bldg No CHAR(5),

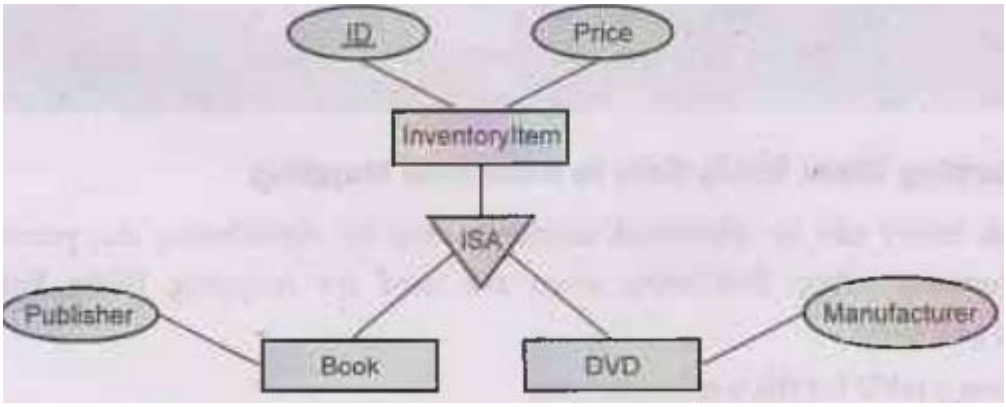
PRIMARY KEY (DeptID,Bldg_No),

FOREIGN KEY(Bldg_No) References Buildings on delete cascade

)

Mapping of Specialization / Generalization (EER Construct)to Relational Mapping

The specialialization/Generalization relationship (Enhanced ER Construct) can be mapped to database tables(relations) using three methods. To demonstrate the methods, we will take the - InventoryItem, Book, DVD



Method 1: All the entities in the relationship are mapped to individual tables

InventoryItem(ID, name)

Book(ID,Publisher)

DVD(ID, Manufacturer)

Method 2: Only subclasses are mapped to tables. The attributes in the superclass are duplicated in all subclasses. For example -

Book(ID,name, Publisher)

DVD(ID, name, Manufacturer)

Method 3: Only the superclass is mapped to a table. The attributes in the subclasses are taken to the superclass. For example -

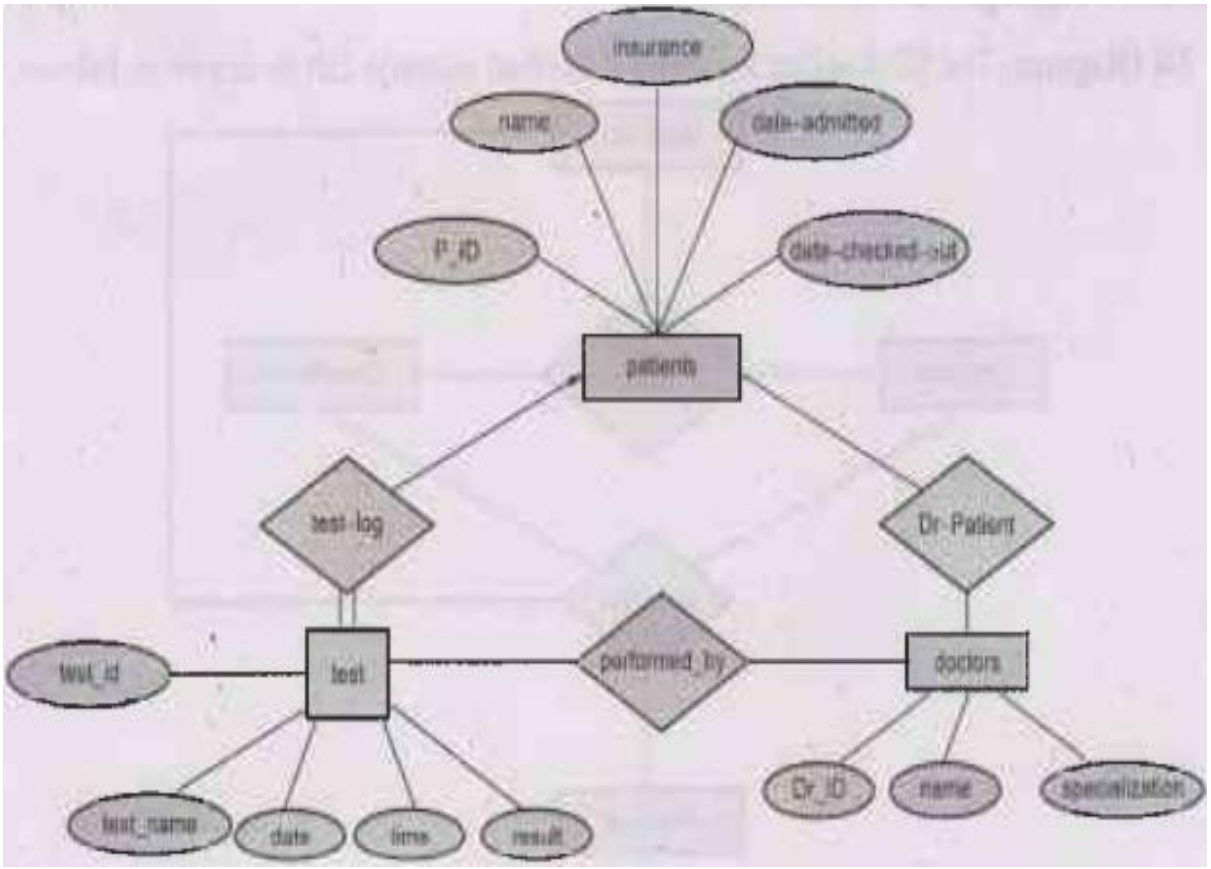
InventoryItem(ID, name, Publisher, Manufacturer)

This method will introduce null values. When we insert a Book record in the table, the Manufacturer column value will be null. In the same way, when we insert a DVD record in the table, the Publisher value will be null.

Example 2.6.1 *Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted. Also construct appropriate tables for the ER diagram you have drawn.*

Solution:

ER Diagram - Refer example 2.5.8.



Relational Mapping

Patients (P_id, name, insurance, date-admitted, date-checked-out)

Doctors (Dr_id, name, specialization)

Test (testid, testname, date, time, result)

doctor-patient (P_id, Dr_id)

test-log (testid, P_id) performed-by (testid, Dr_id)

Review Questions

1. Discuss the correspondence between the ER model construct and the relational model constructs. Show how each ER model construct can be mapped to the relational model. Discuss the option for mapping EER construct. **AU: May-17, Marks 13 2.**

2. Discuss in detail the steps involved in the ER to relational mapping in the process of relational database design.

Functional Dependencies

Definition: Let P and Q be sets of columns, then: P functionally determines Q, written P→Q if and only if any two rows that are equal on (all the attributes in) P must be equal on (all the attributes in) Q.

In other words, the functional dependency holds if

T1.P T2.P, then T1.Q=T2.Q

Where notation T1.P projects the tuple T1 onto the attribute in P.

For example: Consider a relation in which the roll of the student and his/her name is stored as follows:

R	N
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 2.8.1 : Table which holds functional dependency i.e. R->N

Here, R->N is true. That means the functional dependency holds true here. Because for every assigned RollNuumber of student there will be unique name. For instance: The name of the Student whose RollNo is 1 is AAA. But if we get two different names for the same roll number then that means the table does not hold the functional dependency. Following is such table –

R	N
1	AAA
2	BBB
3	CCC
1	XXX
2	YYY

Fig. 2.8.2 : Table which does not hold functional dependency

In above table for RollNumber 1 we are getting two different names - "AAA" and "XXX". Hence here it does not hold the functional dependency.

Computing Closure Set of Functional Dependency

The closure set is a set of all functional dependencies implied by a given set F. It is denoted by F^+

The closure set of functional dependency can be computed using basic three rules which are also called as Armstrong's Axioms.

These are as follows -

i) Reflexivity: If XY, then X→ Y

ii) Augmentation: If X→Y, then XZ → YZ for any Z

iii) Transitivity: If X→Y and Y-Z, then X-Z

In addition to above axioms some additional rules for computing closure set of functional dependency are as follows -

• **Union:** If XY and X-Z then XYZ

• **Decomposition:** If X-YZ, then XY and X→ Z

Example 2.8.1 *Compute the closure of the following set of functional dependencies for a relation scheme R(A,B,C,D,E), F={A->BC, CD->E, B->D, E->A)*

Solution: Consider F as follows

A->BC

CD->E

B->D

E->A

The closure can be written for each attribute of relation as follows:

• (A)*= **Step 1:** {A}-> the attribute itself

Step 2: {ABC} as A->BC

Step 3: {ABCD} as B->D

Step 4: {ABCDE} as CD->E

Step 5: {ABCDE} as E->A and A is already present

Hence (A)+ ={ABCDE}

• (B)*=**Step 1:**{B}

Step 2: {BD} as B->D

Step 3: {BD} as there is no BD pair on LHS of F

Hence (B) ={BD}

• (C)*=**Step 1:**{C}

Step 2: {C} as there is no single C on LHS of F

Hence (C)* ={C}

• $(D)^+$ = **Step 1:** {D}

Step 3: {D} as there is no BD pair on LHS of F

Hence (D)* ={D} .

• $(E)^+$ =**Step 1:** {E}

Step 2: {EA} as E->A

Step 3: {EABC) as A->BC

Step 4: {EABCD} as B->D

Step 5: {EABCD} as CD->E and E is already present

By rearranging we get {ABCDE}

Hence $(E)^+$ ={ABCDE}

• (CD)*=**Step 1:**{CD}

Step 2:{CDE}

Step 3: {CDEA}

Step 4:{CDEAB}

By rearranging we get {ABCDE}

Hence (CD)* ={ABCDE}

Example 2.8.2 *Compute the closure of the following set of functional dependencies for a relation scheme R(A,B,C,D,E), F={A->BC, CD->E, B->D, E->A) and Find the candidate key.*

Solution: *For finding the closure of functional dependencies - Refer example 2.8.1.*

We can identify candidate from the given relation schema with the help of functional dependency. For that purpose we need to compute the closure set of attribute. Now we will find out the closure set which can completely identify the relation R(A,B,C,D).

Let, $(A)^+ = \{ABCDE\}$

$(B)^+ = \{BD\}$

$(C)^+ = \{C\}$

$(D)^+ = \{ABCDE\}$

$(E)^+ = \{ABCDE\}$

$(CD)^+ = \{ABCD\}$

Clearly, only (A), (E) and (CD)* gives us (ABCDE) i.e. complete relation R. Hence these are the candidate keys.

Canonical Cover or Minimal Cover

Formal Definition: A minimal cover for a set F of FDs is a set G of FDs such that:

1) Every dependency in G is of the form X->A, where A is a single attribute.

2) The ***closure F**** is equal to the ***closure G*.***

3) If we obtain a set H of dependencies from G by deleting one or more dependencies or by deleting attributes from a dependency in G, then F* H*.

Concept of Extraneous Attributes

Definition: An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies. The formal definition of extraneous attributes is as follows:

Consider a set F of functional dependencies and the functional dependency α→β in F

• Attribute A is extraneous in α if A€ α and F logically implies (F - {α→ B}) U {(α- A) →β}

• Attribute A is extraneous in β if A€ βand the set of functional dependencies (F-{α→β}) U {(α→ (β - A) } logically implies F.

Algorithm for computing Canonical Cover for set of functional DependenciesF

Fc = F

repeat

Use the union rule to replace any dependencies in Fc of the form

α1 →β1 and αl→β2 and αl→β1β2

Find a functional dependency α→β in $F_c$with an extraneous attribute either in α or in β

/* The test for extraneous attributes is done using Fc, not F */

If an extraneous attribute is found, delete it from α→βin Fc.

until (Fc does not change)

Example 2.8.3 *Consider the following functional dependencies over the attribute set R(ABCDE) for finding minimal cover FD = {A->C, AC->D, B->ADE}*

Solution: Step 1: Split the FD such that R.H.S contain single attribute. Hence we get

A->C

AC->D

B->A

B->D

B->E

Step 2: Find the redundant entries and delete them. This can be done as follows –

For A->C: We find (A)* by assuming that we delete A->C temporarily. We get esimab (A)*={A}. Thus from A it is not possible to obtain C by deleting A->C. This means we can not delete A->C

• **For AC->D:** We find (AC)* by assuming that we delete AC->D temporarily. We get (AC)=(AC). Thus by such deletion it is not possible to obtain D. This means we can not delete AC->D

• **For B->A:** We find (B)* by assuming that we delete B->A temporarily. We get (B)*={BDE). Thus by such deletion it is not possible to obtain A. This means we can not delete B->A

• **For B->D:** We find (B)* by assuming that we delete B->D temporarily. We get (B)=(BEACD). This shows clearly that even if we delete B->D we can obtain D. This means we can delete B->A. Thus it is redundant.

• **For B->E:** We find (B) by assuming that we delete B->E temporarily. We get (B)*={BDAC). Thus by such deletion it is not possible to obtain E. This means we can not delete B->E

To summarize we get now

A->C

AC->D

B->A

B->E

Thus R.H.S gets simplified.

Step 3: Now we will simplify L.H.S.

Consider AC->D. Here we can split A and C. For that we find closure set of A and C.

$(A)^+ = \{AC\}$

$(C)^+ = \{C\}$

Thus C can be obtained from both A as well as C. That also means we need not have to have AC on L.H.S. Instead, only. A can be allowed and C can be eliminated. Thus after simplification we get

A->D

To summarize we get now

A->C

A->D

B->A

B->E

Thus L.H.S gets simplified.

Step 3: The simplified L.H.S. and R.H.S can be combined together to form

A->CD

B->AE

This is a minimal cover or Canonical cover of functional dependencies.

Concept of Redundancy and Anomalies

Definition: Redundancy is a condition created in database in which same piece of data is held at two different places.

Redundancy is at the root of several problems associated with relational schemas.

Problems caused by redundancy: Following problems can be caused by redundancy-

i) Redundant storage: Some information is stored repeatedly.

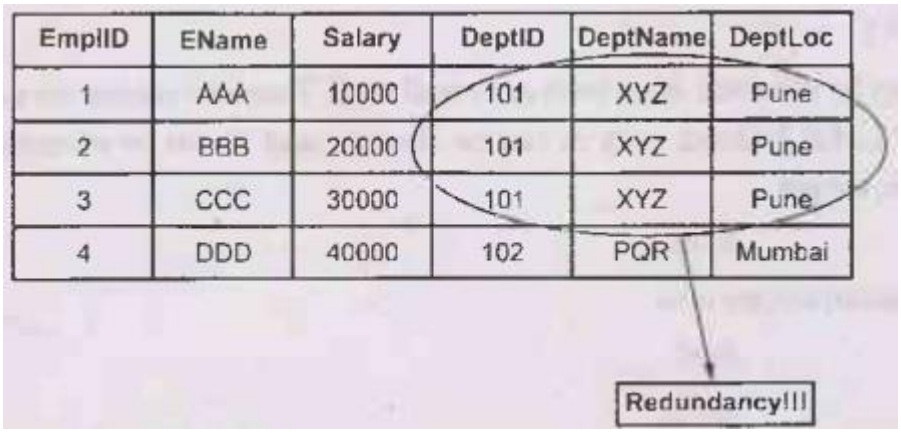
ii) Update anomalies: If one copy of such repeated data is updated then inconsistency is created unless all other copies are similarly updated.

iii) Insertion anomalies: Due to insertion of new record repeated information get added to the relation schema.

iv) Deletion anomalies: Due to deletion of particular record some other important information associated with the deleted record get deleted and thus we may lose some other important information from the schema.

Example: Following example illustrates the above discussed anomalies or redundancy problems

Consider following Schema in which all possible information about Employee is stored.



1) Redundant storage: Note that the information about DeptID, DeptName and DeptLoc is repeated.

2) Update anomalies: In above table if we change DeptLoc of Pune to Chennai, then it will result inconsistency as for DeptID 101 the DeptLoc is Pune. Or otherwise, we need to update multiple copies of DeptLoc from Pune to Chennai. Hence this is an update anomaly**.**

3) Insertion anomalies: For above table if we want to add new tuple say (5, EEE,50000) for DeptID 101 then it will cause repeated information of (101, XYZ,Pune) will occur.

4) Deletion anomalies: For above table, if we delete a record for EmpID 4, then automatically information about the DeptID 102,DeptName PQR and DeptLoc Mumbai will get deleted and one may not be aware about DeptID 102. This causes deletion anomaly.

Decomposition

• Decomposition is the process of breaking down one table into multiple tables.

• Formal definition of decomposition is -

• A decomposition of relation Schema R consists of replacing the relation Schema by two relation schema that each contain a subset of attributes of R and together include all attributes of R by storing projections of the instance.

• For example - Consider the following table

Employee_Department table as follows -

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Hyderabad	25000	D005	Human Resource

We can decompose the above relation Schema into two relation schemas as Employee (Eid, Ename, Age, City, Salary) and Department (Deptid, Eid, DeptName) as follows –

Employee Table

Eid	Ename	Age	City	Salary
E001	ABC	29	Pune	20000
E002	PQR	30	Pune	30000
E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Hyderabad	25000

Department Table

Deptid	Eid	DeptName
D001	E001	Finance
D002	E002	Production
D003	E003	Sales
D004	E004	Marketing
D005	E005	Human Resource

• The decomposition is used for eliminating redundancy.

• **For example:** Consider following relation Schema R in which we assume that the grade determines the salary, the redundancy is caused

Schema R

Name	eid	deptname	Grade	Salary
AAA	121	Accounts	2	8000
AAA	132	Sales	3	7000
BBB	101	Marketing	4	7000
CCC	106	Purchase	2	8000

Redundancy!!!

• Hence, the above table can be decomposed into two Schema S and T as follows:

Schema S

Name	eid	deptname	Grade
AAA	121	Accounts	2
AAA	132	Sales	3
BBB	101	Marketing	4
CCC	106	Purchase	2

Schema T

Grade	Salary
2	8000
3	7000
4	7000
2	8000

Problems Related to Decomposition:

Following are the potential problems to consider:

1) Some queries become more expensive.

2) Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!

3) Checking some dependencies may require joining the instances of the decomposed relations.

4) There may be loss of information during decomposition.

Properties Associated With Decomposition

There are two properties associated with decomposition and those are -

1) Loss-less Join or non Loss Decomposition: When all information found in the original database is preserved after decomposition, we call it as loss less or non loss decomposition.

2) Dependency Preservation: This is a property in which the constraints on the original table can be maintained by simply enforcing some constraints on each of the smaller relations.

Non-loss Decomposition or Loss-less Join

The lossless join can be defined using following three conditions:

i) Union of attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.

$Att(R1) \cup Att(R2) = Att(R)$

ii) Intersection of attributes of R1 and R2 must not be NULL.

$Att(R1)\ Att(R2) \neq \Phi$

iii)Common attribute must be a key for at least one relation (R1 or R2)

Att(R1) Att(R2) -> Att(R1)

Att (R1) Att (R2) -> Att (R2)

Example 2.10.1 *Consider the following relation R(A,B,C,D)and FDs A->BC, is the decomposition of R into R1(A,B,C), R2(A,D). Check if the decomposition is lossless join or not.*

Solution:

Step 1: Here Att(R1) U Att(R2) = Att(R) i.e R1(A,B,C) u R2(A,D)=(A,B,C,D) i.e R.

Step 2: Here R1 R2={A}. Thus Att(R1) Att(R2)≠Φ. Here the second condition gets satisfied.

Step 3: Att(R1) n Att(R2) -> {A}. Now (A)*={A,B,C) ϵ attributes of R1. Thus the third condition gets satisfied.

This shows that the given decomposition is a lossless join.

Example 2.10.2 *Consider the following relation R(A,B,C,D,E,F) and FDs A->BC, C->A, D- >E, F->A, E->D is the decomposition of R into R1(A,C,D), R2(B,C,D), and R3(E,F,D). Check for lossless.*

Solution:

Step 1: R1UR2UR3 = R. Here the first condition for checking lossless join is satisfied as (A,C,D)U (B,C,D) U (E,F,D) = {A,B,C,D,E,F) which is nothing but R.

Step 2: Consider R1 ∩ R2 = {CD) and R2 R3 = {D}. Hence second condition of intersection not being gets satisfied.

Step 3: Now, consider R1(A,C,D) and R2(B,C,D). We find R1/R2 = {CD}

(CD)* = {ABCDE} = attributes of R1 i.e.{A,C,D). Hence condition 3 for checking lossless join for R1 and R2 gets satisfied.

Step 4: Now, consider R2(B,C,D) and R3(E,F,D). We find R2 R3 = {D}. (D)*={D,E} which is neither complete set of attributes of R2 or R3.[Note that F is missing for being attribute of R3].

Hence it is not lossless join decomposition. Or in other words we can say it is a lossy decomposition.

Example 2.10.3 *Suppose that we decompose schema R=(A,B,C,D,E) into (A,B,C) (C,D,E) Show that it is not a lossless decomposition.*

Solution:

Step 1: Here we need to assume some data for the attributes A, B, C, D, and E. Using this data we can represent the relation as follows -

Relation R

A	B	C	D	E
a	1	x	p	q
b	2	x	r	s

Relation R1 = (A,B,C)

A	B	C
a	1	x
b	2	x

Relation R2 = (C,D,E)

C	D	E
x	p	q
x	r	s

Step 2: Now we will join these tables using natural join, i.e. the join based on common attribute C. We get R1⋈ R2 as

A	B	C	D	E
a	1	x	p	q
a	1	x	r	s
b	2	x	p	q
b	2	x	r	s

Here we get more rows or tuples than original relation R

Clearly R1 ⋈ R2 R. Hence it is not lossless decomposition.

Dependency Preservation

• **Definition:** A Decomposition D = {R1, R2, R3....Rn} of R is dependency preserving for a set F of Functional dependency if - (F1 U F2 U... U Fm) = F.

• If decomposition is not dependency-preserving, some dependency is lost in the decomposition.

Example 2.10.4 *Consider the relation R (A, B, C) for functional dependency set (A-> B and B-> C) which is decomposed into two relations R1 = (A, C) and R2 = (B, C). Then check if this decomposition dependency preserving or not.*

Solution: This can be solved in following steps:

Step 1: For checking whether the decomposition is dependency preserving or not we need to check following condition

$$F^+ = (F1 \cup F2)^+$$

Step 2: We have with us the F^+ = { A->B and B->C}

Step 3: Let us find $(F1)^+$ for relation R1 and $(F2)^+$ for relation R2

R1(A,C)	R2(B,C)
A->A Trivial	B->B Trivial
C->C Trivial	C->C Trivial
A->C ∵ In $(F)^+$ A->B->C and it is Nontrivial	B->C ∵ In $(F)^+$ B->C and it is Non-Trivial
AC->AC Trivial	BC->BC Trivial
A->B but is not useful as B is not part of R1 set	We can not obtain C->B
We can not obtain C->A	

Step 4: We will eliminate all the trivial relations and useless relations. Hence we can obtain R1 and R2 as,

R1(A,C)	R2(B,C)
A->C Nontrivial	B->C Non-Trivial

$(F1UF2)^+ = \{A->C, B->C) \neq \{A->B, B->C)$ i.e.$(F)^+$

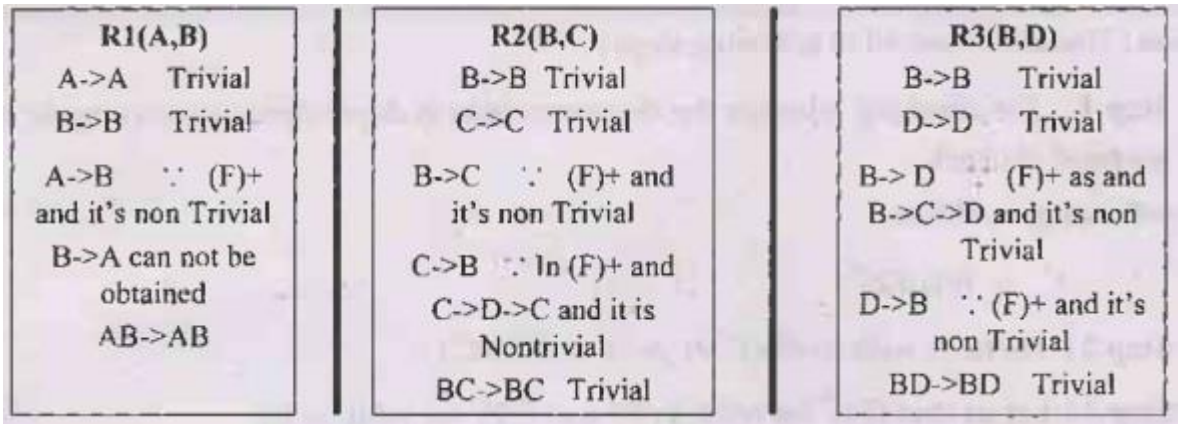
Thus the condition specified in step 1 i.e. $F^+=(F1UF2)^+$ is not true. Hence it is not dependency preserving decomposition.

Example 2.10.5 *Let relation R(A,B,C,D) be a relational schema with following functional dependencies (A->B, B->C,C->D, and D->B). The decomposition of R into (A,B), (B,C) and (B,D). Check whether this decomposition is dependency preserving or not.*

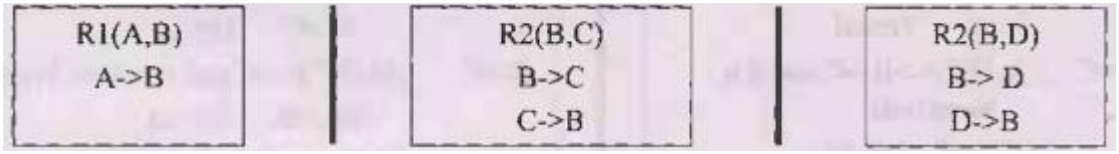
Solution:

Step 1: Let (F) = {A->B, B->C, C->D,D->B}.

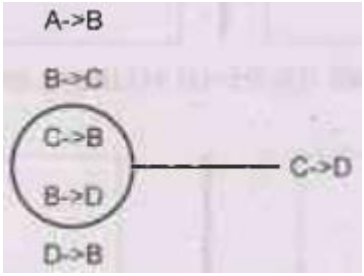
Step 2: We will find $(F1)^+$, $(F2)^+$, $(F3)^+$ for relations R1(A,B), R2(B,C) and R3(B,D) as follows -



Step 3: We will eliminate all the trivial relations and useless relations. Hence we can obtain R1 U R2 U R3 as,



Step 4: As from above FD's we get



Step 5: This proves that $F^+= (F1UF2UF3)^+$. Hence given decomposition is dependency preserving**.**

Example 2.10.6 *Differentiate lossy decomposition and lossless decomposition.*

Solution:

Sr. No.	Lossy decomposition	Lossless decomposition
1.	In this decomposition, there is loss of data.	In this decomposition, there is no loss of any data.
2.	This technique cannot restore the original amount and quality of the data contained in the original file.	This technique can easily restore the original quality and amount of data in an available file after decompression.
3.	The original size of data reduces after lossy data compression.	The original size of data stays intact after lossless data compression.
4.	This technique can hold more data because it does not need to restore them back to their original form.	This technique can hold less data because it restores the original quality and amount of data.

Review questions

Differentiate between lossless join decomposition and dependency preserving decomposition.

Normal Forms

• Normalization is the process of reorganizing data in a database so that it meets two basic requirements:

1) There is no redundancy of data (all data is stored in only one place), and

2) data dependencies are logical (all related data items are stored together)

Need for normalization

1) It eliminates redundant data.

2) It reduces chances of data error.

3) The normalization is important because it allows database to take up less disk space.

4) It also help in increasing the performance.

5) It improves the data integrity and consistency.

First Normal Form

The table is said to be in 1NF if it follows following rules –

i) It should only have single (atomic) valued attributes/columns.

ii) Values stored in a column should be of the same domain

iii) All the columns in a table should have unique names.

iv) And the order in which data is stored, does not matter.

Consider following Student table

Student

sid	sname	Phone
1	AAA	11111 22222
2	BBB	33333
3	CCC	44444 55555

As there are multiple values of phone number for sid 1 and 3, the above table is not in 1NF. We can make it in 1NF. The conversion is as follows -

sid	sname	Phone
1	AAA	11111
1	AAA	22222
2	BBB	33333
3	CCC	44444
3	CCC	55555

Second Normal Form

Before understanding the second normal form let us first discuss the concept of j functional dependency and prime and non prime attributes.

Concept of Partial Functional Dependency

Partial dependency means that a nonprime attribute is functionally dependent on part of a candidate key.

For example: Consider a relation R(A,B,C,D) with functional dependency {AB->CD,A->C}

Here (AB) is a candidate key because

$(AB)^+ = \{ABCD\} = \{R\}$

Hence {A,B) are prime attributes and {C,D) are non prime attribute. In A->C, the non prime attribute C is dependent upon A which is actually a part of candidate key AB. Hence due to A->C we get partial functional dependency.

Prime and Non Prime Attributes

• **Prime attribute:** An attribute, which is a part of the candidate-key, is known as a prime attribute.

• **Non-prime attribute:** An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

• **Example :** Consider a Relation R ={A,B,C,D) and candidate key as AB, the Prime attributes: A, B

Non Prime attributes: C, D

The Second Normal Form

For a table to be in the Second Normal Form, following conditions must be followed

i) It should be in the First Normal form.

ii) It should not have partial functional dependency.

For example: Consider following table in which every information about a the Student is maintained in a table such as student id(sid), student name(sname), course id(cid) and course name(cname).

Student_Course

sid	sname	cid	cname
1	AAA	101	C
2	BBB	102	C++
3	CCC	101	C
4	DDD	103	Java

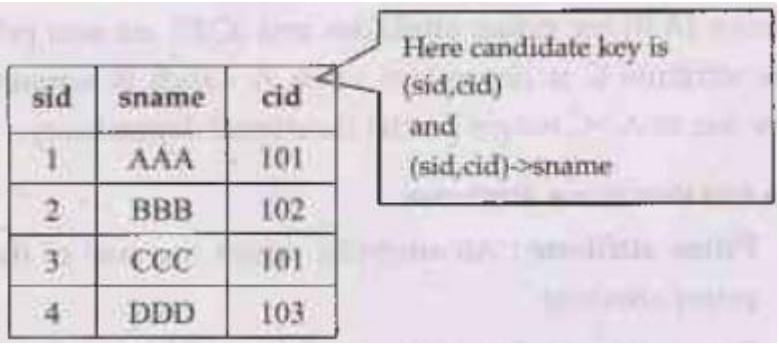
This table is not in 2NF. For converting above table to 2NF we must follow the following steps -

Step 1: The above table is in 1NF.

Step 2: Here sname and sid are associated similarly cid and cname are associated with each other. Now if we delete a record with sid=2, then automatically the course C++ will also get deleted. Thus,

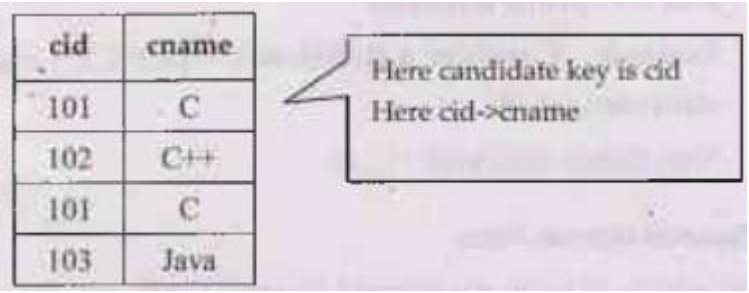
sid->sname or cid->cname is a partial functional dependency, because {sid,cid} should be essentially a candidate key for above table. Hence to bring the above table to 2NF we must decompose it as follows:

Student:



sid	sname	cid
1	AAA	101
2	BBB	102
3	CCC	101
4	DDD	103

Course:



cid	cname
101	C
102	C++
101	C
103	Java

Thus now table is in 2NF as there is no partial functional dependency

Third Normal Form

Before understanding the third normal form let us first discuss the concept of transitive dependency, super key and candidate key.

Concept of Transitive Dependency

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. For example -

X -> Z is a transitive dependency if the following functional dependencies hold true:

X->Y

Y->Z

Concept of Super key and Candidate Key

Superkey: A super key is a set or one of more columns (attributes) to uniquely identify rows in a table.

Candidate key: The minimal set of attribute which can uniquely identify a tuple is known as candidate key.

For example consider following table:

RegID	RollNo	Sname
101	1	AAA
102	2	BBB
103	3	CCC
104	4	DDD

Superkeys

• {RegID}

• {RegID, RollNo}

• {RegID,Sname}

• {RollNo,Sname}

• {RegID, RollNo,Sname}

Candidate Keys

• {RegID}

• {RollNo}

Third Normal Form

A table is said to be in the Third Normal Form when,

i) It is in the Second Normal form.(i.e. it does not have partial functional dependency)

ii) It doesn't have transitive dependency.

Or in other words

In other words 3NF can be defined as: 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 :

i) X is a super key of table

ii) Y is a prime attribute of table

For example: Consider following table Student_details as follows -

sid	sname	zipcode	cityname	state
1	AAA	11111	Pune	Maharashtra
2	BBB	22222	Surat	Gujarat
3	CCC	33333	Chennai	Tamilnadu
4	DDD	44444	Jaipur	Rajastan
5	EEE	55555	Mumbai	Maharashtra

Here

Super keys: {sid}, {sid,sname}, {sid,sname,zipcode}, {sid,zipcode,cityname}... and so on.

Candidate keys:{sid}

Non-Prime attributes: {sname,zipcode,cityname,state}

The dependencies can be denoted as,

sid->sname

sid->zipcode

zipcode->cityname

cityname->state

The above denotes the transitive dependency. Hence above table is not in 3NF. We can convert it into 3NF as follows:

Student

sid	sname	zipcode
1	AAA	11111
2	BBB	22222
3	CCC	33333
4	DDD	44444
5	EEE	55555

Zip

zipcode	cityname	state
11111	Pune	Maharashtra
22222	Surat	Gujarat
33333	Chennai	Tamilnadu
44444	Jaipur	Rajasthan
55555	Mumbai	Maharashtra

Example 2.11.1 *Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F= {{A, B} →C, A →{D, E}, B→F, F→ {G, H}, D →{I, J} }*

1. What is the key for R? Demonstrate it using the inference rules.

2. Decompose R into 2NF, then 3NF relations.

Solution: Let,

A→ DE (given)

A→ D, A→ E

AsD→IJ, A→IJ

Using union rule we get

A → DEIJ

As A→ A

we get A → ADEIJ

Using augmentation rule we compute AB

AB → ABDEIJ

But

AB→C (given)

AB →ABCDEIJ

B→ F (given) F→ GH .. B→ GH (transitivity)

AB →AGH is also true

Similarly AB→ AF B→F (given)

Thus now using union rule

AB → ABCDEFGHIJ

AB is a key

The table can be converted to 2NF as,

R1= (A, B, C)

R2= (A, D, E, I, J) smoot

R3= (B, F, G, H)

The above 2NF relations can be converted to 3NF as follows:

R1= (A, B, C)

R2= (A, D, E)

R1= (D, I, J)

R1= (B, E)

R3= (E, G, H).

Review Questions

*1. What is database normalization? Explain the first normal form, second normal form and third normal form.***AU: May-18, Marks 13; Dec.-15, Marks 16**

2. What are normal forms. Explain the types of normal form with an example. **AU: Dec.-14, Marks 16**

3. *What is normalization? Explain in detail about all Normal forms.* **AU: May-19, Marks 13**

4. Briefly discuss about the functional dependency concepts. **AU: May-19, Marks 13**

Boyce / Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF.

A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

Or in other words,

For a table to be in BCNF, following conditions must be satisfied:

i) R must be in 3rd Normal Form

ii) For each functional dependency (X → Y), X should be a super Key. In simple words if Y is a prime attribute then X can not be non prime attribute.

For example - Consider following table that represents that a Student enrollment for the course -

Enrollment Table

sid	Course	Skill
1	C C++	English German
2	Java	English French

From above table following observations can be made:

• One student can enroll for multiple courses. For example student with sid=1 can enroll for C as well as Java.

• For each course, a teacher is assigned to the student.

• There can be multiple teachers teaching one course for example course C can be taught by both the teachers namely - Ankita and Archana.

• The candidate key for above table can be (sid,course), because using these two columns we can find

• The above table holds following dependencies,

  • (sid,course)->Teacher

  • Teacher->courseann

• The above table is not in BCNF because of the dependency teacher->course. Note NS that the teacher is not a superkey or in other words, teacher is a non prime attribute and course is a prime attribute and non-prime attribute derives the prime attribute.

• To convert the above table to BCNF we must decompose above table into Student and Course tables

Student

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Course

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Now the table is in BCNF

Example 2.12.1 *Consider a relation(A,B,C,D) having following FDs.(AB->C, AB->D, C->A, B->D). Find out the normal form of R.*

Solution:

Step 1: We will first find out the candidate key from the given FD.

$(AB)^+$= {ABCD) = R

$(BC)^+$={ABCD) = R

$(AC)^+$ ={AC} R

There is no involvement of D on LHS of the FD rules. Hence D can not be part of any candidate key. Thus we obtain two candidate keys (AB)* and (BC)*. Hence

prime attributes = {A,B,C)

Non prime attributes = {D}

Step 2: Now, we will start checking from reverse manner, that means from BCNF, so then 3NF, then 2NF.

Step 3: For R being in BCNF for X->Y the X should be candidate key or super key. From above FDs consider C->D in which C is not a candidate key or super key. Inabu Hence given relation is not in BCNF.

Step 4: For R being in 3NF for X->Y either i) the X should be candidate key or super key or ii) Y should be prime attribute. (For prime and non prime attributes refer step 1)

- For AB->C or AB->D the AB is a candidate key. Condition for 3NF is satisfied.
- Consider C->A. In this FD the C is not candidate key but A is a prime attribute. Condition for 3NF is satisfied.
- Now consider B->D. In this FD, the B is not candidate key, similarly D is not a prime attribute. Hence condition for 3NF fails over here.

Hence given relation is not in 3NF.

Step 5: For R being in 2NF following condition should not occur.

Let X->Y, if X is a proper subset of candidate key and Y is a non prime attribute. This is a case of partial functional dependency.

For relation to be in 2NF there should not be any partial functional dependency.

- For AB->C or AB->D the AB is a complete candidate key. Condition for 2NF is satisfied.
- Consider C->A. In this FD the C is not candidate key. Condition for 2NF is satisfied.

• Now consider B->D. In this FD, the B is a part of candidate key(AB or BC), similarly D is not a prime attribute. That means partial functional dependency occurs here. Hence condition for 2NF fails over here. Hence given relation is not in 2NF.

Therefore we can conclude that the given relation R is in 1NF.

Example 2.12.2 *Consider a relation R(ABC) with following FD A->B, B->C and C->A. What is the normal form of R?*

Solution:

Step 1: We will find the candidate key

$(A)^+$ ={ABC} =R

$(B)^+$ = {ABC} =R

$(C)^+$={ABC} =R

Hence A, B and C all are candidate keys

Prime attributes = {A,B,C}

Non prime attribute {}

Step 2: For R being in BCNF for X->Y the X should be candidate key or super key. From above FDs

• Consider A->B in which A is a candidate key or super key. Condition for BCNF is satisfied.

• Consider B->C in which B is a candidate key or super key. Condition for BCNF is satisfied.

• Consider C->A in which C is a candidate key or super key. Condition forBCNF is satisfied.

This shows that the given relation R is in BCNF.

Example 2.12.3 *Prove that any relational schema with two attributes is in BCNF.*

Solution: Here, we will consider R={A,B) i.e. a relational schema with two attributes. Now various possible FDs are A->B, B->A.

From the above FDs

• Consider A->B in which A is a candidate key or super key. Condition for BCNF is satisfied.

• Consider B->A in which B is a candidate key or super key. Condition for BCNF is satisfied.

• Consider both A->B and B->A with both A and B is candidate key or super key. Condition for BCNF is satisfied.

• No FD holds in relation R. In this {A,B) is candidate key or super key. Still condition for BCNF is satisfied.

This shows that any relation R is in BCNF with two attributes.

Multivalued Dependencies and Fourth Normal Form

AU: May-14, Dec.-16, 19, Marks 16

Concept of Multivalued Dependencies

• A table is said to have multi-valued dependency, if the following conditions are true,

1) For a dependency A→B, if for a single value of A, multiple values of B exists, then the table may have multi-values dependency.

2) Also, a table should have at-least 3 columns for it to have a multi-valued dependency.

3) And, for a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

• In simple terms, if there are two columns A and B - and for column A if there are multiple values of column B then we say that MVD exists between A and B

• The multivalued dependency is denoted by →

• If there exists a multivalued dependency then the table is not in 4th normal form.

• **For example:** Consider following table for information about student

Student

sid	Course	Skill
1	C C++	English German
2	Java	English French

Here sid =1 leads to multiple values for courses and skill. Following table shows this

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Here sid and course are dependent but the Course and Skill are independent. The multivalued dependency is denoted as :

sid→ Course

sid→ Skill

Fourth Normal Form

Definition: For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1) It should be in the Boyce-Codd Normal Form(BCNF).

2) And, the table should not have any multi-valued dependency.

For example: Consider following student relation which is not in 4NF as it contains multivalued dependency.

Student Table

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Now to convert the above table to 4NF we must decompose the table into following two tables.

Student_Course Table

Key: (sid,Course)

sid	Course
1	C
1	C++
2	Java

Student Skill Table

Key: (sid, Skill)

sid	Skill
1	English
1	German
2	English
2	French

Thus the tables are now in 4NF.

Review Questions

1. Explain first normal form, second normal form, third normal form and BCNF with example. **AU: Dec.-16, Marks 13**

2. Explain Boyce Codd normal form and fourth normal form with suitable example. **AU: May-14, Marks 16**

3. Exemplify the multi-value dependency and fourth normal form 4NF. **AU: Dec.-19, Marks 6**

Join Dependencies and Fifth Normal Form

Concept of Join Dependencies

• Join decomposition is a further generalization of Multivalued dependencies.

• If the join of R1 and R2 over C is equal to relation R, then we can say that a Join Dependency (JD) exists.

• Where R1 and R2 are the decompositions R1(A, B, C) and R2(C, D) of a given relations R (A, B, C, D).

• Alternatively, R1 and R2 are a lossless decomposition of R.

• A JD ⋈ (R1, R2,..., Rn} is said to hold over a relation R if R1, R2,....., Rn is a lossless-join decomposition.

• The *(A, B, C, D), (C, D) will be a JD of R if the join of join's attribute is equal to the relation R.

• Here, *(R1, R2, R3) is used to indicate that relation R1, R2, R3 and so on are a JD of R.

Concept of Fifth Normal Form

The database is said to be in 5NF if -

i) It is in 4th Normal Form

ii) If we can decompose table further to eliminate redundancy and anomalies and when we rejoin the table we should not be losing the original data or get a new record (join Dependency Principle)

The fifth normal form is also called as project join normal form

For example - Consider following table

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Britania	Biscuits

Here we assume the keys as {Seller, Company, Product}

The above table has multivalued dependency as

Seller → {Company, Product). Hence table is not in 4th Normal Form. To make the above table in 4th normal form we decompose above table into two tables as

Seller_Company

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Seller_Product

Seller	Product
Rupali	Cinthol
Sharda	Honey
Sharda	HairOil
Sharda	RoseWater
Sunil	Icecream
Sunil	Biscuits

The above table is in 4th Normal Form as there is no multivalued dependency. But it is not in 5th normal form because if we join the above two table we may get

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Amul	Biscuits
Sunil	Britania	Icecream
Sunil	Britania	Biscuits

Newly added records which are not present in original table

To avoid the above problem we can decompose the tables into three tables as Seller_Company, Seller_Product, and Company Product table

Seller_Company

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Seller_Product

Seller	Product
Rupali	Cinthol
Sharda	Honey
Sharda	HairOil
Sharda	RoseWater
Sunil	Icecream
Sunil	Biscuit

Company_Product

Company	Product
Godrej	Cinthol
Dabur	Honey
Dabur	HairOil
Dabur	RoseWater
Amul	Icecream
Britania	Biscuit

Thus the table in in 5th normal form.

Review Question

1. Exemplify the join dependency and thew fifth normal form. **AU: Dec.-19, Marks 6**

Examples on Normalization

AU: Dec.-19, Marks 9

Example 2.15.1 *Study the relation given below and state what level of normalization can be achieved and normalize it upto that level.*

Order no.	Order Date	Item Lines		
		Item Code	Quantity	Price/Unit
1456	26-12-1999	3687	52	50.4
		4627	38	60
		3214	20	20.00
1886	04-03-1999	4629	45	20.25
		4627	30	60.20
1788	04-04-1999	4627	40	60.20

Solution:

Reason for the given relation being unnormalized

1. Observe order for many items.
2. Item lines has many attributes-called composite attributes.
3. Each tuple has variable length.
4. Difficult to store due to non-uniformity.
5. Given item code difficult to find qty-ordered and hence called Unnormalized relation.

For conversion to First Normal Form -

• Identify the composite attributes, convert the composite attributes to individual attributes.

• Duplicate the common attributes as many times as lines in composite attribute.

• Every attribute now describes single property and not multiple properties, some data will be duplicated.

• Now this is called First normal form (1NF) also called flat file.

Order no.	Order Date	Item Lines		
		Item Code	Quantity	Price/Unit
1456	26-12-1999	3687	52	50.4
1456	26-12-1999	4627	38	60
1456	26-12-1999	3214	20	20.00
1886	04-03-1999	4629	45	20.25
1886	04-03-1999	4627	30	60.20
1788	04-04-1999	4627	40	60.20

Fig. 2.15.1 : Table in first normal form

• The above table has insertion, deletion and update anomalies. For instance - if we delete order no. 1886, then the item code 4629 gets lost. Similarly if we update 4627, then all instances of 4627 need to be changed.

• We need to convert 2NF if it is in 1NF. The non-key attributes are functionally dependent on key attribute and if there is a composite key then no non-key attribute is functionally depend on one part of the key.

• The table can be converted to 2NF as follows -

Orders

OrderNo	OrderDate
1456	26-12-1999
1886	04-03-1999
1788	04-04-1999

Order Details

OrderNo	ItemCode	Qty
1456	3687	52
1886	4629	45
1788	4627	40

Prices

ItemCode	Price/Unit
3687	50.4
4627	60
3214	20
4629	20.25

Example 2.15.2 *A software contract and consultancy firm maintains details of all the various projects in which its employees are currently involved. These details comprise:*

• *Employee Number*

• *Employee Name*

• *Date of Birth*

• *Department Code*

• *Department Name Project Code*

• *Project Description*

• *Project Supervisor*

Assume the following:

• *Each employee number is unique.*

• *Each department has a single department code.*

• *Each project has a single code and supervisor.*

• *Each employee may work on one or more projects.*

• *Employee names need not necessarily be unique.*

• *Project Code, Project Description and Project Supervisor are repeating fields.*

Normalise this data to Third Normal Form.

Solution:

Un-Normalized Form

Employee Number, Employee Name,Date of Birth, Department Code, Department Name, Project Code, Project Description, Project Supervisor

1NF

Employee Number, Employee Name___Date of Birth

Department Code, Department Name

Employee Number. Project Code. Project Description,_Project Supervisor

2NF

Employee Number, Employee Name,_Date of Birth,_Department Code,__Department Name Employee Number, Project Code,

Project Code, Project Description, Project Supervisor

3NF

*Employee Number, Employee Name___Date of Birth,*Department Code*

Department Code, Department Name

Employee Number, Project Code

Project Code, Project Description, Project Supervisor

Example 2.15.3 *What is normalization?Normalize below given relation upto 3NF STUDENT.*

StudID	StudName	City	Pincode	ProjectID	ProjectName	Course	Content
S101	Ajay	Surat	326201	P101	Health	Programming	C++, Java, C
S102	Vijay	Pune	325456	P102	Social	WEB	HTML, PHP, ASP

Solution: For converting the given schema to first normal form, we will arrange it in such a way that have each tuple contains single record. For that purpose we need to split the schema into two tables namely Student and Projects

1NF

Student

StudID	StudName	Pincode	City
S101	Ajay	326201	Surat
S102	Vijay	325456	Pune

Projects

StudID	ProjectID	ProjName	Course	Content
S101	P101	Health	Programming	C++
S101	P101	Health	Programming	Java
S101	P101	Health	Programming	C
S102	P102	Social	WEB	HTML
S102	P102	Social	WEB	PHP
S102	P102	Social	WEB	ASP

2NF

For a table to be in 2NF, there should not be any partial dependency.

Student

StudID	StudName	Pincode	City
S101	Ajay	326201	Surat
S102	Vijay	325456	Pune

Project

StudID	ProjectID	ProjName	CourseID
S101	P101	Health	C101
S101	P101	Health	C102
S101	P101	Health	C103
S102	P102	Social	C104
S102	P102	Social	C105
S102	P102	Social	C106

CourseDetails

CourseID	Course	Content
C101	Programming	C++
C102	Programming	Java
C103	Programming	C
C104	WEB	HTML
C105	WEB	PHP
C106	WEB	ASP

3NF: There was a transitive dependency in 2NF tables because city is associated with student ID and city depends upon zip code. Hence the transitive dependency is removed to covert table into 3NF. The required 3NF schema is as below -

Student

StudID	StudName	Pincode
S101	Ajay	326201
S102	Vijay	325456

Student_Address

Pincode	City
326201	Surat
325456	Pune

Project

StudID	ProjectID	ProjName	CourseID
S101	P101	Health	C101
S101	P101	Health	C102
S101	P101	Health	C103
S102	P102	Social	C104
S102	P102	Social	C105
S102	P102	Social	C106

CourseDetails

CourseID	Course	Content
C101	Programming	C++
C102	Programming	Java
wC103	Programming	C
C104	WEB	HTML
C105	WEB	PHP
C106	WEB	ASP

Example 2.15.4 *Consider table R(A,B,C,D,E) with FDs as A->B, BC->E, and ED->A.The table is in which normal form? Justify your answer.*

Solution:

Step 1: We will first find out the candidate keys for given relation R

$(ACD)^+ = \{A,B,C,D,E\}$

$(BCD)^+ = \{A,B,C,D,E\}$

$(CDE)^+ = \{A,B,C,D,E\}$

Step 2: Let A->B, the ACD is candidate key and A is a partial key, B is a prime attribute(i.e. it is also part of candidate key). Hence A->B is not a partial functional dependency.

Similarly in BC->E and ED->A,

E and A are prime-attributes and hence both are not partial functional dependencies. Hence R is in 2NF.

Step 3: According to 3NF, every non-prime attribute must be dependent on the candidate key.

In the given functional dependencies, all dependent attributes are prime-attributes. Hence the relation R is in 3NF.

Step 4: For R being in BCNF for X->Y the X should be candidate key or super key.

The table is not in BCNF, none of A, BC and ED contain a key.

Example 2.15.5 *Prove the statement "Every relation which is in BCNF is in 3NF but the converse is not true".*

Solution: For a relations to be in 3NF

A table is said to be in the Third Normal Form when,

i) It is in the Second Normal form. (i.e. it does not have partial functional dependency)

ii) It doesn't have transitive dependency.

Or in other words

In other words 3NF can be defined as: 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:

iii) X is a super key of table

iv) Y is a prime attribute of table

For a relation to be in BCNF

(1) It should be in 3NF

(2) A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

For proving that the table can be in 3NF but not in BCNF consider following relation R(Student, Subject, Teacher) . Consider following are FDs

(Subject, Student)-> Teacher

Because subject and student combination gives unique teacher. A

Teacher ->Subject

Because each teacher teaches only Subject.D.

(Teacher, Student)->Subject

• So, this relation is in 3NF as every non-key attribute is non-transitively fully sim noinu grian won Buil functional dependent on the primary key.

• But it is not in BCNF. Because this is a case of overlapping of candidate keys because there are two composite candidate keys:

- (Subject, Student)
- (Teacher, Student)

And Student is a common attribute in both the candidate keys.

So we need to normalize the above table to BCNF. For that purpose we must set Teacher to be a candidate keylot as TME of

The decomposition of above takes place as follows

R1(Student, Teacher)

R2(Teacher, Subject)

Now table is in 3NF, as well as in BCNF.

This show that the relation Every relation which is in BCNF is in 3NF but the converse is not true.

Example 2.15.6 *Consider the relation R = {A, B, C, D, E, F, G, H, I, J) and the set of functional dependencies F = {{A, B} - C, A (D, E), B →F, F{G, H), D→{I, J} }*

1. What is the key for R? Demonstrate it using the inference rules.

2. Decompose R into 2NF, then 3NF relations.

Solution: Let,

A → DE (given)

A → D, A→E (decomposition rule)

As D → IJ, A → IJ

Using union rule we get

A → DEIJ

As A→A

we get A→ ADEIJ

Using augmentation rule we compute AB

AB → ABDEIJ

But

AB→C (given)

AB → ABCDEIJ

B→ F (given) F→GH B→GH (transitivity)

AB→AGH is also true

Similarly AB→AF B → F (given)

Thus now using union rule

AB → ABCDEFGHIJ

AB is a key

The table can be converted to 2NF as

R1=(A, B, C)

R2=(A, D, E, I, J)

R3= (B, F, G, H)

The above 2NF relations can be converted to 3NF as follows:

R1=(A, B, C)

R2 = (A, D, E)

R3=(D, I, J)

R4 = (B, E)

R5= (E, G, H).

Example 2.15.7 *Consider a relation R(ABC) with following FD A->B, B->C and C->A. What is the normal form of R?*

Solution:

Step 1: We will find the candidate key

$(A)^+$={ABC} = R

$(B)^+$= {ABC) = R

$(C)^+$={ABC) = R

Hence A, B and C all are candidate keys

Prime attributes = {A,B,C}

Non prime attribute {}

Step 2: For R being in BCNF for X->Y the X should be candidate key or super key. From above FDs

- Consider A->B in which A is a candidate key or super key. Condition for BCNF is satisfied.
- Consider B->C in which B is a candidate key or super key. Condition for BCNF is satisfied.
- Consider C->A in which C is a candidate key or super key. Condition for BCNF is satisfied.

This shows that the given relation R is in BCNF.

Example 2.15.8 *Consider the Table 2.15.1 and answer to queries given below.*

Userid	U_Email	Fname	Lname	City	State	Zip
MA12	mani@ymail.com	Manish	Jain	Bilaspur	Chhattisgarh	458991
PO45	pujag@gmail.com	Pooja	Magg	Kacch	Gujrat	832212
LA33	lavle98@jj.com	Lavleen	Dhalla	Raipur	Chhattisgarh	853578
CH99	checki9j@ih.com	Chimal	Bedi	Trichy	Tamil Nadu	632011
DA74	danu58@g.com	Dany	James	Trichy	Tamil Nadu	645018

Table 2.15.1 User_personal

1) In this table in first normal form - 1NF? Justify and normalize to 1 NF if needed.

2) Is this table in second normal form - 2NF ? Justify and normalize to 2NF if needed.

3) Is User_personal in third normal form - 3NF? Justify and normalize to 3NF if needed.

Solution:

1) All the rows contain only one atomic value.

Hence table is in 1NF.

2) For identifying if table is in 2NF, we must check two rules -

Rule 1: The table must be in 1NF.

Rule 2: There should not be any partial key dependency.

As we know, that table is in 1NF, Rule 1 is said to be satisfied.

For checking Rule 2, first find out the primary keys.

Assume that, User_id and zip are to primary keys.

F = {User_id → U_Email Fname Lname City State Zip

Zip→ City State

}

Note that Userid can uniquely identify all the attributes of given relation. There is no partial dependency for identifying all the attributes. Hence rule 2 is said to be fulfilled. Therefore table is in 2NF.

3) To verify 3NF, the conditions are -

Rule 1: Table should be in 2NF

Rule 2: There should not be transitive dependency in the table. The table is already in 2NF, hence rule 1 is already satisfied.

Given table shows transitive dependency. It is as follows:

Userid → Zip and Zip→ City State

To bring the relation in 3NF, we have to decompose table into two tables

User_personal (Userid, U_Email, Fname, Lname, Zip)

Address (Zip, City, State)

The underlined fields are primary keys of respective tables. The tables are as follows:

Userid	U_Email	Fname	Lname	Zip
MA12	mani@ymail.com	Manish	Jain	458991
PO45	pujag@gmail.com	Pooja	Magg	832212
LA33	lavle98@jj.com	Lavleen	Dhalla	853578
CH99	checki9j@ih.com	Chimal	Bedi	632011
DA74	danu58@g.com	Dany	James	645018

User_personal table

Zip	City	State
458991	Bilaspur	Chhattisgarh
832212	Kacch	Gujrat
853578	Raipur	Chhattisgarh
632011	Trichy	Tamil Nadu
645018	Trichy	Tamil Nadu

Address Table

Example 2.15.9 What *is the difference between 3NF and BCNF?*

Solution:

Sr. No.	3NF	BCNF
1.	3NF stands for third normal form.	BCNF stands for Boyce Codd normal form.
2.	The table is in 3NF if it is in 2NF and for each functional dependency X->Y at least following condition hold : (i) X is a superkey, (ii) Y is prime attribute of table.	The table is in BCNF if it is in 3rd normal form and for each relation X->Y X should be super key.
3.	3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved in BCNF.
4.	Lossless decomposition can be achieved in 3NF.	Lossless decomposition is hard to obtain in BCNF.
5.	3NF can be achieved without loosing any information from the old table.	For obtaining BCNF we may loose some information from old table.

SQL Set Operation

The SQL Set operation is used to combine the two or more SQL SELECT statements.

Types of Set Operation

1. Union
2. UnionAll
3. Intersect
4. Minus

1. Union

- The SQL Union operation is used to combine the result of two or more SQL SELECT queries.
- In the union operation, all the number of datatype and columns must be same in both the tables on which UNION operation is being applied.
- The union operation eliminates the duplicate rows from its resultset.

Syntax

```
SELECT column_name FROM table1
UNION
SELECT column_name FROM table2;
```

Example:

The First table

ID	NAME

1	Jack
2	Harry
3	Jackson

The Second table

ID	NAME
3	Jackson
4	Stephan
5	David

Union SQL query will be:

```
SELECT * FROM First
UNION
SELECT * FROM Second;
```

The resultset table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
4	Stephan
5	David

2. Union All

Union All operation is equal to the Union operation. It returns the set without removing duplication and sorting the data.

Syntax:

```
SELECT column_name FROM table1
UNION ALL
SELECT column_name FROM table2;
```

Example: Using the above First and Second table.

Union All query will be like:

```
SELECT * FROM First
UNION ALL
SELECT * FROM Second;
```

The resultset table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
3	Jackson
4	Stephan
5	David

3. Intersect

- It is used to combine two SELECT statements. The Intersect operation returns the common rows from both the SELECT statements.
- In the Intersect operation, the number of datatype and columns must be the same.
- It has no duplicates and it arranges the data in ascending order by default.

Syntax

```
SELECT column_name FROM table1
INTERSECT
SELECT column_name FROM table2;
```

Example:

Using the above First and Second table.

Intersect query will be:

```
SELECT * FROM First
INTERSECT
SELECT * FROM Second;
```

The resultset table will look like:

ID	NAME
3	Jackson

4. Minus

- It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- It has no duplicates and data arranged in ascending order by default.

Syntax:

```
SELECT column_name FROM table1
MINUS
SELECT column_name FROM table2;
```

Example

Using the above First and Second table.

Minus query will be:

```
SELECT * FROM First
MINUS
SELECT * FROM Second;
```

The resultset table will look like:

ID	NAME
1	Jack
2	Harry

SQL Aggregate Functions

- SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.
- It is also used to summarize the data.

Types of SQL Aggregation Function

1. COUNT FUNCTION

- COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
- COUNT function uses the COUNT(*) that returns the count of all the rows in a specified table. COUNT(*) considers duplicate and Null.

Syntax

```
COUNT(*)
or
COUNT( [ALL|DISTINCT] expression )
```

Sample table:

PRODUCT_MAST

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Cpm1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50
Item10	Com3	4	30	120

Example: COUNT()

```
SELECT COUNT(*)
FROM PRODUCT_MAST;
```

Output:

```
10
```

Example: COUNT with WHERE

```
SELECT COUNT(*)
FROM PRODUCT_MAST;
WHERE RATE>=20;
```

Output:

```
7
```

Example: COUNT() with DISTINCT

```
SELECT COUNT(DISTINCT COMPANY)
FROM PRODUCT_MAST;
```

Output:

```
3
```

Example: COUNT() with GROUP BY

```
SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY;
```

Output:

```
Com1    5
Com2    3
Com3    2
```

Example: COUNT() with HAVING

```
SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING COUNT(*)>2;
```

Output:

```
Com1    5
Com2    3
```

2. SUM Function

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

Syntax

```
SUM()
or
SUM( [ALL|DISTINCT] expression )
```

Example: SUM()

```
SELECT SUM(COST)
FROM PRODUCT_MAST;
```

Output:

```
670
```

Example: SUM() with WHERE

```
SELECT SUM(COST)
FROM PRODUCT_MAST
WHERE QTY>3;
```

Output:

```
320
```

Example: SUM() with GROUP BY

```
SELECT SUM(COST)
FROM PRODUCT_MAST
WHERE QTY>3
```

GROUP BY COMPANY;

Output:

```
Com1    150
Com2    170
```

Example: SUM() with HAVING

```
SELECT COMPANY, SUM(COST)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING SUM(COST)>=170;
```

Output:

```
Com1    335
Com3    170
```

3. AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax

```
AVG()
```

or

```
AVG( [ALL|DISTINCT] expression )
```

Example:

```
SELECT AVG(COST)
FROM PRODUCT_MAST;
```

Output:

```
67.00
```

4. MAX Function

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax

```
MAX()
```

or

```
MAX( [ALL|DISTINCT] expression )
```

Example:

```
SELECT MAX(RATE)
FROM PRODUCT_MAST;
```

```
30
```

5. MIN Function

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

Syntax

MIN()

or

MIN([ALL|DISTINCT] expression)

Example:

```
SELECT MIN(RATE)
FROM PRODUCT_MAST;
```

Output:

```
10
```

SQL Sub Query

A Subquery is a query within another SQL query and embedded within the WHERE clause.

Important Rule:

- A subquery can be placed in a number of SQL clauses like WHERE clause, FROM clause, HAVING clause.
- You can use Subquery with SELECT, UPDATE, INSERT, DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
- A subquery is a query within another query. The outer query is known as the main query, and the inner query is known as a subquery.
- Subqueries are on the right side of the comparison operator.
- A subquery is enclosed in parentheses.
- In the Subquery, ORDER BY command cannot be used. But GROUP BY command can be used to perform the same function as ORDER BY command.

1. Subqueries with the Select Statement

SQL subqueries are most frequently used with the Select statement.

Syntax

```
SELECT column_name  FROM table_name  WHERE column_name expression operator
( SELECT column_name  from table_name WHERE ... );
```

Example

Consider the EMPLOYEE table have the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	John	20	US	2000.00

2	Stephan	26	Dubai	1500.00
3	David	27	Bangkok	2000.00
4	Alina	29	UK	6500.00
5	Kathrin	34	Bangalore	8500.00
6	Harry	42	China	4500.00
7	Jackson	25	Mizoram	10000.00

The subquery with a SELECT statement will be:

```
SELECT *  FROM EMPLOYEE WHERE ID IN (SELECT ID  FROM EMPLOYEE
WHERE SALARY > 4500);
```

This would produce the following result:

ID	NAME	AGE	ADDRESS	SALARY
4	Alina	29	UK	6500.00
5	Kathrin	34	Bangalore	8500.00
7	Jackson	25	Mizoram	10000.00

2. Subqueries with the INSERT Statement

- SQL subquery can also be used with the Insert statement. In the insert statement, data returned from the subquery is used to insert into another table.
- In the subquery, the selected data can be modified with any of the character, date functions.

Syntax:

```
INSERT INTO table_name (column1, column2, column3....)
SELECT *
FROM table_name
WHERE VALUE OPERATOR
```

Example

Consider a table EMPLOYEE_BKP with similar as EMPLOYEE.

Now use the following syntax to copy the complete EMPLOYEE table into the EMPLOYEE_BKP table.

```
INSERT INTO EMPLOYEE_BKP SELECT * FROM EMPLOYEE  WHERE ID IN (SELECT ID FROM E
MPLOYEE);
```

3. Subqueries with the UPDATE Statement

The subquery of SQL can be used in conjunction with the Update statement. When a subquery is used with the Update statement, then either single or multiple columns in a table can be updated.

Syntax

```
UPDATE table  SET column_name = new_value  WHERE VALUE OPERATOR
 (SELECT COLUMN_NAME  FROM TABLE_NAME WHERE condition);
```

Example

Let's assume we have an EMPLOYEE_BKP table available which is backup of EMPLOYEE table. The given example updates the SALARY by .25 times in the EMPLOYEE table for all employee whose AGE is greater than or equal to 29.

```
UPDATE EMPLOYEE SET SALARY = SALARY * 0.25  WHERE AGE IN (SELECT
AGE FROM CUSTOMERS_BKP WHERE AGE >= 29);
```

This would impact three rows, and finally, the EMPLOYEE table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	John	20	US	2000.00
2	Stephan	26	Dubai	1500.00
3	David	27	Bangkok	2000.00
4	Alina	29	UK	1625.00
5	Kathrin	34	Bangalore	2125.00
6	Harry	42	China	1125.00
7	Jackson	25	Mizoram	10000.00

4. Subqueries with the DELETE Statement

The subquery of SQL can be used in conjunction with the Delete statement just like any other statements mentioned above.

Syntax

```
DELETE FROM TABLE_NAME WHERE VALUE OPERATOR (SELECT COLUMN_NAME
FROM TABLE_NAME WHERE condition);
```

Example

Let's assume we have an EMPLOYEE_BKP table available which is backup of EMPLOYEE table. The given example deletes the records from the EMPLOYEE table for all EMPLOYEE whose AGE is greater than or equal to 29.

DELETE FROM EMPLOYEE WHERE AGE IN (SELECT AGE FROM EMPLOYEE_BKP WHERE AGE >= 29);

This would impact three rows, and finally, the EMPLOYEE table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	John	20	US	2000.00
2	Stephan	26	Dubai	1500.00
3	David	27	Bangkok	2000.00
7	Jackson	25	Mizoram	10000.00

Views in SQL

- Views in SQL are considered as a virtual table. A view also contains rows and columns.
- To create the view, we can select the fields from one or more tables present in the database.
- A view can either have specific rows based on certain condition or all the rows of a table.

Advantages of View:

1. **Complexity:** Views help to reduce the complexity. Different views can be created on the same base table for different users.
2. **Security:** It increases the security by excluding the sensitive information from the view.
3. **Query Simplicity:** It helps to simplify commands from the user. A view can draw data from several different tables and present it as a single table.
4. **Consistency:** A view can present a consistent, unchanged image of the structure of the database. Views can be used to rename the columns without affecting the base table.
5. **Data Integrity:** If data is accessed and entered through a view, the DBMS can automatically check the data to ensure that it meets the specified integrity constraints.
6. **Storage Capacity:** Views take very little space to store the data.
7. **Logical Data Independence:** View can make the application and database tables to a certain extent independent.

Disadvantages of View:

The DML statements which can be performed on a view created using single base table have certain restrictions are:

1. You cannot INSERT if the base table has any not null column that do not appear in view.
2. You cannot INSERT or UPDATE if any of the column referenced in the INSERT or UPDATE contains group functions or columns defined by expression.

3. You can't execute INSERT, UPDATE, DELETE statements on a view if with read only option is enabled.
4. You can't be created view on temporary tables.
5. You cannot INSERT, UPDATE, DELETE if the view contains group functions GROUP BY, DISTINCT or a reference to a psuedocolumn rownum.
6. You can't pass parameters to the SQL server views.
7. You can't associate rules and defaults with views.

Sample table:

Student_Detail

STU_ID	NAME	ADDRESS
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

Student_Marks

STU_ID	NAME	MARKS	AGE
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

1. Creating view

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

2. Creating View from a single table

In this example, we create a View named DetailsView from the table Student_Detail.

Query:

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM Student_Details
WHERE STU_ID < 4;
```

Just like table query, we can query the view to view the data.

1. `SELECT * FROM DetailsView;`

Output:

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

3. Creating View from multiple tables

View from multiple tables can be created by simply include multiple tables in the SELECT statement.

In the given example, a view is created named MarksView from two tables Student_Detail and Student_Marks.

Query:

```
CREATE VIEW MarksView AS
SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS
FROM Student_Detail, Student_Mark
WHERE Student_Detail.NAME = Student_Marks.NAME;
```

To display data of View MarksView:

1. `SELECT * FROM MarksView;`

NAME	ADDRESS	MARKS
Stephan	Delhi	97
Kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

4. Deleting View

A view can be deleted using the Drop View statement.

Syntax

```
DROP VIEW view_name;
```

Example:

If we want to delete the View **MarksView**, we can do this as:

```
DROP VIEW MarksView;
```

Significance of Views:

Views are highly significant, as they can provide advantages over tasks. Views can represent a subset of data contained in a table. Consequently they can limit the degree of exposure of the underlying base table to the outer world. They are used for security purpose in database and act as an intermediate between real table schemas and programmability. They act as aggregate tables.

Types of Views:

There are two types of views.

1. **Join View:** A join view is a view that has more than one table or view in its from clause and it does not use any Group by Clause, Rownum, Distinct and set operation.
2. **Inline View:** An inline view is a view which is created by replacing a subquery in the from clause which defines the data source that can be referenced in the main query. The sub query must be given an alias for efficient working.

Trigger in SQL

A **Trigger** in Structured Query Language is a set of procedural statements which are executed automatically when there is any response to certain events on the particular table in the database. Triggers are used to protect the data integrity in the database.

In SQL, this concept is the same as the trigger in real life. For example, when we pull the gun trigger, the bullet is fired.

To understand the concept of trigger in SQL, let's take the below hypothetical situation:

Suppose Rishabh is the human resource manager in a multinational company. When the record of a new employee is entered into the database, he has to send the 'Congrats' message to each new employee. If there are four or five employees, Rishabh can do it manually, but if the number of new Employees is more than the thousand, then in such condition, he has to use the trigger in the database.

Thus, now Rishabh has to create the trigger in the table, which will automatically send a 'Congrats' message to the new employees once their record is inserted into the database.

The trigger is always executed with the specific table in the database. If we remove the table, all the triggers associated with that table are also deleted automatically.

In Structured Query Language, triggers are called only either before or after the below events:

1. **INSERT Event:** This event is called when the new row is entered in the table.
2. **UPDATE Event:** This event is called when the existing record is changed or modified in the table.

3. **DELETE Event:** This event is called when the existing record is removed from the table.

Types of Triggers in SQL

Following are the six types of triggers in SQL:

1. **AFTER INSERT Trigger**

   This trigger is invoked after the insertion of data in the table.

2. **AFTER UPDATE Trigger**

   This trigger is invoked in SQL after the modification of the data in the table.

3. **AFTER DELETE Trigger**

   This trigger is invoked after deleting the data from the table.

4. **BEFORE INSERT Trigger**

   This trigger is invoked before the inserting the record in the table.

5. **BEFORE UPDATE Trigger**

   This trigger is invoked before the updating the record in the table.

6. **BEFORE DELETE Trigger**

   This trigger is invoked before deleting the record from the table.

Syntax of Trigger in SQL

```
CREATE TRIGGER Trigger_Name
[ BEFORE | AFTER ]  [ Insert | Update | Delete]
ON [Table_Name]
[ FOR EACH ROW | FOR EACH COLUMN ]
AS
Set of SQL Statement
```

In the trigger syntax, firstly, we have to define the name of the trigger after the CREATE TRIGGER keyword.

After that, we have to define the BEFORE or AFTER keyword with anyone event.

Then, we define the name of that table on which trigger is to occur.

After the table name, we have to define the row-level or statement-level trigger.

And, at last, we have to write the SQL statements which perform actions on the occurring of event.

Example of Trigger in SQL

To understand the concept of trigger in SQL, first, we have to create the table on which trigger is to be executed.

The following query creates the **Student_Trigger** table in the SQL database:

```
CREATE TABLE Student_Trigger
(
Student_RollNo INT NOT NULL PRIMARY KEY,
Student_FirstName Varchar (100),
Student_EnglishMarks INT,
Student_PhysicsMarks INT,
```

```
1. Student_ChemistryMarks INT,
2. Student_MathsMarks INT,
3. Student_TotalMarks INT,
4. Student_Percentage  );
```

The following query shows the structure of the**Student_Trigger** table:

```
1. DESC Student_Trigger;
```

Output:

Field	Type	NULL	Key	Default	Extra
Student_RollNo	INT	NO	PRI	NULL	
Student_FirstName	Varchar(100)	YES		NULL	
Student_EnglishMarks	INT	YES		NULL	
Student_PhysicsMarks	INT	YES		NULL	
Student_ChemistryMarks	INT	YES		NULL	
Student_MathsMarks	INT	YES		NULL	
Student_TotalMarks	INT	YES		NULL	
Student_Percentage	INT	YES		NULL	

The following query fires a trigger before the insertion of the student record in the table:

```
CREATE TRIGGER Student_Table_Marks
BEFORE INSERT
ON
Student_Trigger
FOR EACH ROW
1. SET new.Student_TotalMarks = new.Student_EnglishMarks + new.Student_PhysicsMarks + new.Student_C
   hemistryMarks + new.Student_MathsMarks,
2. new.Student_Percentage = ( new.Student_TotalMarks / 400) * 100;
```

The following query inserts the record into Student_Trigger table:

```
1. INSERT INTO Student_Trigger (Student_RollNo, Student_FirstName, Student_EnglishMarks, Student_Ph
   ysicsMarks, Student_ChemistryMarks, Student_MathsMarks, Student_TotalMarks, Student_Percentage) VA
   LUES ( 201, Sorya, 88, 75, 69, 92, 0, 0);
```

To check the output of the above INSERT statement, you have to type the following SELECT statement:

```
1. SELECT * FROM Student_Trigger;
```

Output:

Student_RollNo	Student_FirstName	Student_EnglishMarks	Student_PhysicsMarks	Student_chemistryMarks	Student_MathsMarks	Student_TotalMarks	Student_Percentage
201	Sorya	88	75	69	92	324	81

Advantages of Triggers in SQL

Following are the three main advantages of triggers in Structured Query Language:

1. SQL provides an alternate way for maintaining the data and referential integrity in the tables.
2. Triggers helps in executing the scheduled tasks because they are called automatically.
3. They catch the errors in the database layer of various businesses.
4. They allow the database users to validate values before inserting and updating.

Disadvantages of Triggers in SQL

Following are the main disadvantages of triggers in Structured Query Language:

1. They are not compiled.
2. It is not possible to find and debug the errors in triggers.
3. If we use the complex code in the trigger, it makes the application run slower.
4. Trigger increases the high load on the database system.