282

Double side 14

Scoring Indicators

COURSE NAME: DATABASE MANAGEMENT SYSTEMS

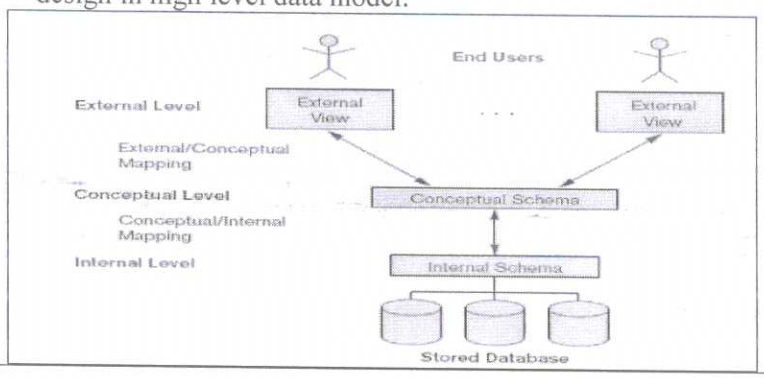
COURSE CODE:3133 **QID: 2109240085-A**

Q. No.		Scoring Indicators	Split Score	Sub Total	Total Score
PART A					**9**
I	1	Database schema	1	1	
I	2	Tuple	1	1	
I	3	Menu-Based Interfaces for web clients or browsing, Forms-Based Interfaces, Graphical User Interfaces, Natural Language interfaces, Speech input and output, Interfaces for parametric users, Interfaces for the DBA. (Any two).	½ + ½	1	
I	4	INSERT	1	1	
I	5	COUNT()	1	1	
I	6	Weak entity.	1	1	
I	7	Rectangle.	1	1	
I	8	4NF	1	1	
I	9	Atomicity	1	1	
PART B					**24**
II	1	Network Model - Entities are organized in a graph, in which some entities can be accessed through several paths. - It supports many-to-many relationships. - Instead of only allowing each child to have one parent, this model allows each child to have multiple parents (children are called members and the parents are ***owners***). - Disadvantages : - It is more difficult to implement and maintain. - The programmer still has to understand the data structure well in order to make the model efficient.	Any 3 points: 1 mark each	3	
II	2	Data Definition Language - A database system provides a data-definition language to specify or define the database schema (Schema includes name of data item, type of data , constraints, functions etc.) - The definitions are actually data about data (also known as metadata) - Database system stores the definitions in a place known as data dictionary. - The major DDL commands in SQL are: CREATE, ALTER, DROP.	2 1	3	
II	3	- The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values. - Here, RegistrationNumber can be used as primary key for the relation CAR(Name, RegistrationNumber, Model, Color).	1 mark each	3	

		- RegistrationNumber is the only unique value in the given list of attributes.			
II	4	i) SELECT DISTINCT Category FROM PRODUCT; ii) SELECT Prod_Id, Name, Price, Category FROM PRODUCT WHERE Prod_Id = 1012;	1 ½ + 1 ½	3	
II	5	View - A view is a virtual table based on the result-set of an SQL statement. - A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. - Syntax CREATE VIEW <view name> [(attribute1, attribute2, ...)] AS SELECT <fieldlist / expression list> FROM<table list> WHERE<condition> GROUP BY<expression> ; - DROP VIEW command is used to dispose an existing view DROP VIEW <view name>;	1 1 1	3	
II	6	Entity - An entity is referred to as an object or thing that exists in the real world. For example, student, customer, book, etc. All these entities have some attributes or properties that give them their identity. Relationship - The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships. - An entity is a unique object that exists, and a relationship is the connection between entities.	1 ½ 1 ½	3	
II	7	**Normalization** - Normalization is a process of analyzing the given relation schemas based on their functional dependencies (FDs) and primary keys to achieve the following desirable properties: • minimize redundancy and • Minimize the insertion, deletion and update anomalies. - Normalization process performs a series of decomposition operations on relations until the relation achieves certain normal forms. - The normal form of a relation indicates the degree to which it has been normalized. - Various levels of Normal forms are 1NF, 2NF, 3NF, BCNF, 4NF and 5NF.	2 1	3	
II	8	Concurrency Control - Concurrency Control in Database Management System is a procedure of managing simultaneous operations without conflicting with each other.	Any 3 points 1 mark each	3	

		- If many transactions try to access the same data, then inconsistency arises. - Concurrency control is required to maintain consistency in data. - Concurrency control is provided in a database to: • enforce isolation among transactions. • preserve database consistency through consistency preserving execution of transactions. • resolve read-write and write-read conflicts			
II	9	- Decompose EMPLOYEE table into two tables as follows. - EMPLOYEE Emp_Id | Name | Dept_Id - DEPARTMENT Dept_Id | Dept_Name | Dept_Head	1 ½ + 1 ½	3	
II	10	**Advantages of Mobile Databases** Some advantages of mobile databases are: - The data in a database can be accessed from anywhere using a mobile database. - It provides wireless database access. - The main database systems are synchronized with mobile databases and multiple users can access the data with seamless delivery process. - Mobile databases require very little support and maintenance. - The mobile database can be synchronized with multiple devices such as mobiles, computer devices, laptops etc.	Any 3 points 1 mark each	3	
		PART C			42
III		**Database Users** - In a large organization, many people are involved in the design, use and maintenance of their databases. ▪ **Database Administrator (DBA)** ▪ **Database Designer** ▪ **Various End Users** ▪ **System Analysts** ▪ **Application Programmers** **Duties of Database Administrator (DBA)** **Database administrators (DBA)** are responsible for: - Administering the resources such as database, DBMS and related software. - Authorizing access to the database (grant, revoke) - Coordinating and monitoring database use - Acquiring software and hardware resources - DBA is responsible to database security and performance - Responsible for regular backup & restore procedures **Responsibilities of Database Designer** **Database designers** are responsible for: - Identifying the data to be stored in the database	2 1 1	7	

		- Choosing appropriate structures to represent and store this data - Communicate with all prospective database users to understand their requirements - Develop views of the database that meet the requirements **Various End Users** - End users are people whose jobs require access to the database for querying, updating and generating reports. - There are several categories of end users: ▪ **Casual end users**: They access database for different types of information occasionally. ▪ **Naive** or **parametric end users:** They constantly querying and updating the database using canned transactions which have been carefully programmed and tested. (Eg: Bank Clerks, Réservation agents for Airlines etc.) ▪ **Sophisticated end users**: This include engineers, scientists, business analysts etc, who thoroughly study the facilities of DBMS in order to implement their own applications with complex requirements ▪ **Standalone users** : They maintain personal databases by using ready-made programs **Duties of System Analysts & Programmers** - **System analysts :** System analyst determine requirements of end users, especially naive & parametric end users, and develop specifications for canned transactions that meet their requirements - **Application programmers:** Programmers implement these specifications as programs, then they test, debug, document, and maintain these canned transactions. - System Analysts and programmers are also referred to as software developers or software engineers.	1 2		
IV		**Three Schema Architecture** In this architecture, schema can be defined at **three** levels with proper mapping between levels 1. **Internal level** 2. **Conceptual Level** 3. **3. External or view level** **Internal Level (Internal Schema)** - This level has internal schema which describes the physical storage structure of the database. - Internal schema uses a physical data model and describes the complete details of data storage and access paths for the database. **Conceptual Level (Conceptual Schema)** - This level has a **conceptual schema**. - It describes the structure of the whole database. - Conceptual schema hides the details of physical storage structure and concentrates on describing entities, relationships, user operations and constraints. - A representational data model is used to describe the conceptual schema.	Diagram: 4 + Explanation: 3	7	

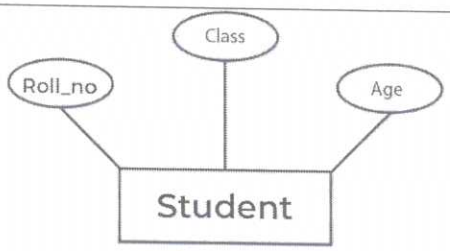
		- Conceptual schema implementation is based on a conceptual schema design in a high level language model (Eg: SQL). **External Level (External Schema)** - It is also known as View Level - It includes a number of external schemas or user views. - Each external schema describes the part of the database interested by a user group. - It hides the rest of the database from the user group. - It is implemented by a representational data model based on a design in high level data model.  			
V		Integrity Constraints in SQL NOT NULL - A constraint NOT NULL may be specified if NULL is not permitted for a particular attribute. - Example: AdmsnNo INT NOT NULL, Name VARCHAR(30) NOT NULL DEFAULT <value> - It defines a default value for an attribute by appending the clause DEFAULT <value> to an attribute definition. Default value is taken, if explicit value is not provided. - Example: Course VARCHAR(20) DEFAULT 'Diploma' CHECK - CHECK clause can restrict attribute or domain values by specifying conditions. - Example : Mark INT NOT NULL CHECK(Mark> 0 AND Mark<100) PRIMARY KEY - PRIMARY KEY column uniquely identifies tuples (rows) in a relation. - It can be a single column or combination of more than one column. By default, PRIMARY KEY is NOT NULL. - Example1: regno INT PRIMARY KEY Example 2 : PRIMARY KEY(regno, course_id) FOREIGN KEY - Referential integrity is specified via. FOREIGN KEY clause. - A referential integrity constraint can be violated when tuples are inserted or deleted or when a primary key or foreign key attribute value is modified. - Example: CONSTRAINT empdepfk FOREIGN KEY(DeptNo) REFERENCES department(Dno)	List: 1 + Any three: 2 marks each	7	

		ON DELETE SET DEFAULT ON UPDATE CASCADE. UNIQUE - UNIQUE clause specifies alternate (secondary) keys. - UNIQUE key can also be specified directly for a secondary key if the secondary key is a single attribute. - Eg : Dname VARCHAR(15) UNIQUE			

VI		TRIGGERS
		- A database trigger is a procedural code that is automatically executed in response to certain events on a particular table in a database.
		- Trigger is a database object that is associated with a table.
		- A trigger is activated when a particular event occurs for the table.
		- A trigger is defined to activate when a statement INSERTS, UPDATES, or DETELTES rows in the associated table. These row operations are trigger events. A trigger can be set to activate either BEFORE of AFTER the trigger event.
		- Syntax:
		
		CREATE TRIGGER<trigger-name>
		<trigger-time> <trigger-event>
		ON<table-name>
		FOR EACH ROW
		[<trigger-order>]
		<trigger-body> ;
		
		▪ <trigger-time> - { BEFORE | AFTER }
		▪ <trigger-event> - { INSERT | UPDATE | DELETE }
		▪ <trigger-order> - { FOLLOWS | PRECEDES }
		<other_trigger-name>
		- <trigger-body> is the statements to execute when the trigger activates. To execute multiple statements, use BEGIN ... END compound statement construct.
		- OLD.col-name refers to a column of an existing row before itis updated or deleted
		- NEW.col-name refers to the column of a new row to be inserted or an existing row after it is updated.
		
		- Example: (Any similar example)
		- Trigger to update total employees in the Department table whenever a new data added to Employee table.
		
		CREATE TRIGGER After_Insert_Employee
		AFTER INSERT ON Employee
		FOR EACH ROW
		BEGIN
		UPDATE Department
		SET TotalEmployees = TotalEmployees + 1
		WHERE Dept_Id = NEW.Department_Id;
		END;
			2
			
			3
			
			2
				7	

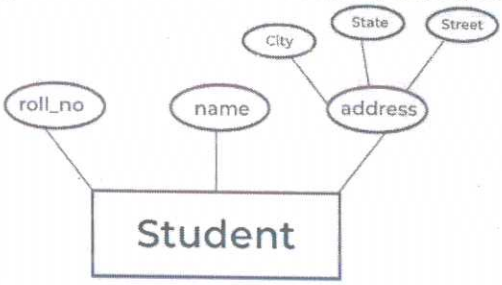
VII		i) CREATE TABLE STUDENT(Stud_Id INT NOT NULL, Stud_Name VARCHAR(30) NOT NULL, DOB DATE NOT NULL, Gender CHAR(10),	3	7	

		Email VARCHAR(20), Course VARCHAR(20) DEFAULT 'Diploma', PRIMARY KEY(Stud_Id)); ii) UPDATE STUDENT SET Email = 'name@gmail.com' WHERE Stud_Id = 2310; iii) SELECT Stud_Id AS 'Student Id', Stud_Name AS 'Name', DOB, Gender, Email, Course FROM STUDENT WHERE UPPER(Stud_Name) LIKE 'A%';	2 2		
VIII		i) Create table Publisher first. CREATE TABLE Publisher(PublisherId INT NOT NULL, Name VARCHAR(30), Address VARCHAR(50), PRIMARY KEY(PublisherId)); Then create Book table. CREATE TABLE Book(Book_Id INT NOT NULL, Title VARCHAR(50) NOT NULL, PublisherId INT NOT NULL, Price DECIMAL(10,2), PRIMARY KEY(Book_Id), FOREIGN KEY(PublisherId) REFERENCES Publisher (PublisherId)); ii) SELECT Book_Id as 'Book ID', title as 'Title', Publisher.Name as 'Publisher', Publisher.Address as 'Address of Publisher', price as 'Price' FROM Book INNER JOIN Publisher ON Book.PublisherId = Publisher.PublisherId;	4 3	7	
IX		**Types of Attributes** - There are different types of attributes as: • Simple Attribute • Composite Attribute • Single-Valued Attribute • Multi-Valued Attribute • Derived Attribute • Complex Attribute • Stored Attribute • Key Attribute • Null Attribute • Descriptive Attribute **1. Simple Attribute** - An attribute that cannot be further subdivided into components is a simple attribute. - **Example:** The roll number of a student, the ID number of an employee, gender, and many more.	Name of any 3: 1 mark + Any 3: 2 marks each	7	



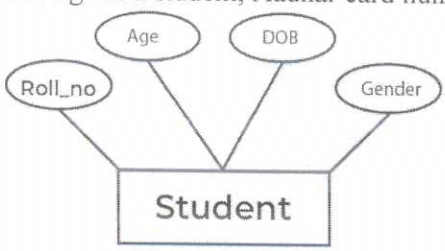
2.Composite Attribute

- An attribute that can be split into components is a composite attribute.
- **Example:** The address can be further split into house number, street number, city, state, country, and pin code, the name can also be split into first name middle name, and last name.



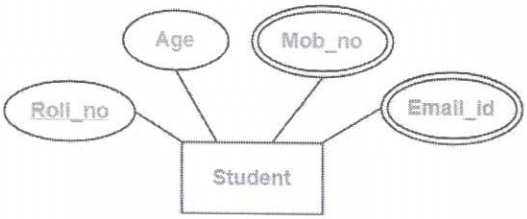
3. Single-Valued Attribute

- The attribute which takes up only a single value for each entity instance is a single-valued attribute.
- **Example:** The age of a student, Aadhar card number.



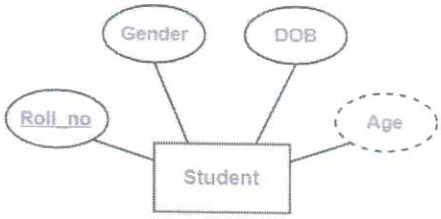
4. Multi-Valued Attribute

- The attribute which takes up more than a single value for each entity instance is a multi-valued attribute. And it is represented by double oval shape.
- **Example:** Phone number of a student: Landline and mobile.



5. Stored Attribute

- The stored attribute are those attribute which doesn't require any type of further update since they are stored in the database.
- **Example:** DOB(Date of birth) is the stored attribute.

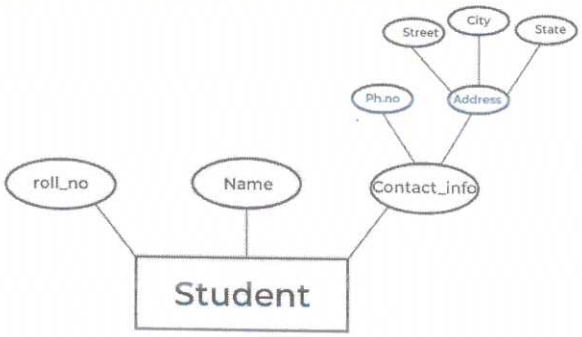


6. Derived Attribute

- An attribute that can be derived from other attributes is derived attributes. And it is represented by dotted oval shape.
- **Example:** Total and average marks of a student, age of an student that is derived from date of birth.
  (see the above picture)

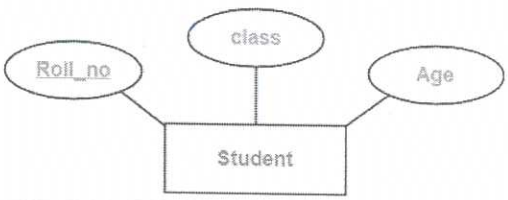
7. Complex Attribute

- Those attributes, which can be formed by the nesting of composite and multi-valued attributes, are called "*Complex Attributes*".
- **Example:** Address because address contain composite value like street, city, state, PIN code and also multi-valued because one people has more than one house address.



8. Key attribute

- Key attributes are those attributes that can uniquely identify the entity in the entity set.
- **Example:** Roll-No is the key attribute because it can uniquely identify the student.
- In diagrammatic notations each key attribute is represented by an underlined attribute inside the oval.



X		**Specialization** - Specialization is the process of defining a set of subclasses of an entity type. The entity type is called the **superclass** of specialization. Specialization is based on some distinguishing characteristics of the entities in the superclass. - **Example :** • The set of subclasses {SECRETARY, ENGINEER, TECHNICIAN } is a specialization of the superclass	3 ½	7	

EMPLOYEE with the distinguishable characteristics - **job_type**.

- The set of subclasses { HOURLY_EMPLOYEE, MONTHLY_EMPLOYEE} is another specialization of EMPLOYEE with distinguishable characteristics - **pay_method**.

- In summary, Specialization process allows to do the following:
  - define a set of subclasses of an entity type
  - define additional attributes with each subclass
  - define additional relationship types between each subclass and other entity types or other subclasses.

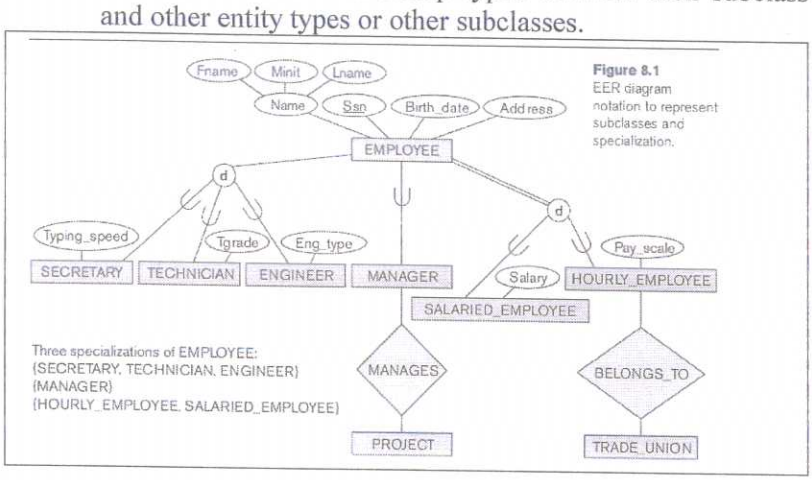


Figure 8.1
EER diagram notation to represent subclasses and specialization.

Generalization

- It is the reverse process of abstraction in which we identify the common features of several entity types and generalize them into a single superclass of which the original entity types are special subclasses. 3 ½
- **Example:-** Consider two entity types – CAR & TRUCK. Both of them have several common features or attributes. Those common features can be generalized into the entity type VEHICLE. VEHICLE is the generalized super class.

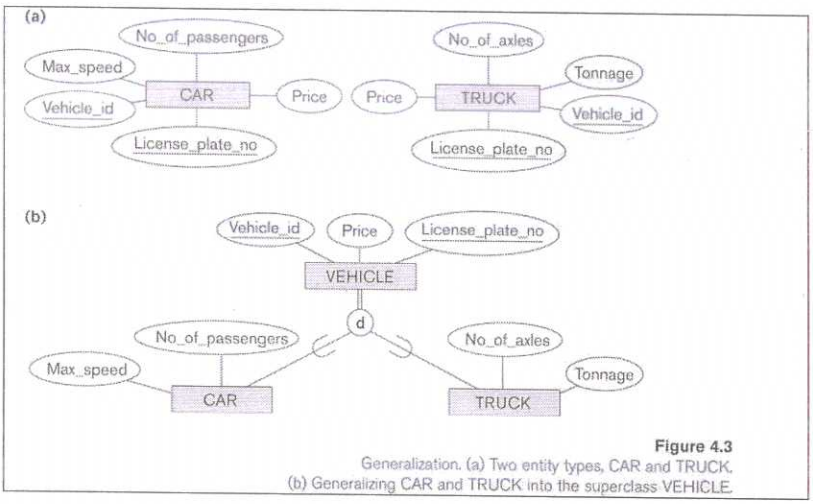
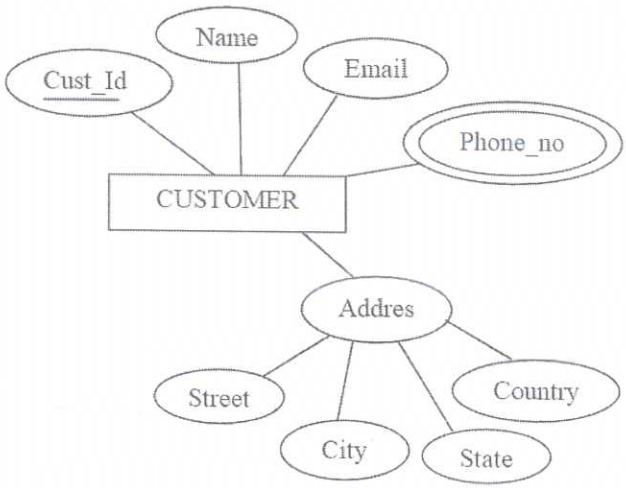


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

- Generalization process can be viewed as being functionally the inverse of the specialization process.

XI		Key attribute for Primary key Multi-valued attribute Composite attribute 	Entity: 2 Single attribute: 1 Key attribute: 1 Multi-valued: 1 Composite: 2	7	
XII		**Mapping ER Model Constructs to Relations** **Step 1: Mapping of Regular (Strong) Entity Types** - Create a new relation schema R for each regular entity type - The name given to the relation is generally the same as the entity type. - Each simple attribute of the entity type and simple components of composite attributes becomes an attribute of the relation schema R. - Choose one of the key attributes as the primary key for the Relation R. **Step 2: Mapping of Weak Entity Types** - For each weak entity type, create a new relation schema and include all of the simple attributes as attributes of this relation schema. - Then include the **primary key of the identifying relation** (Owner) as a **foreign key** attribute in this new relation schema. - The primary key of the new relation schema is the combination of the primary key of the owner and the partial key of the weak entity type. **Step 3: Mapping of Binary 1:1 Relationships Types** - Identify the relations **S** and **T** participating in the relationship. - There are three possible approaches : • **Foreign key approach** - Choose one of the relations S and include the primary key of T as a foreign key in S. Include all attributes of relationship as attributes of S • **Merged relation approach** - Merge the two entity types and the relationship into a single relation. This is possible when both participations are total	1 mark for each step	7	

		• **Cross-reference or relationship relation approach** - Create a third relation called relationship relation. Include the primary key of both participating entities as foreign keys in the new relation. One of the foreign keys can be used as the primary key for the new relation. **Step 4: Mapping of Binary 1:N Relationship Types.** - Identify the relation **S** , at N-side of the relationship. - Include the primary key of other relation **T** as foreign key in **S.** - Include any simple attributes of 1:N relationship type as attributes of S. **Step 5: Mapping of Binary M:N Relationship Types.** - For each binary M:N relationship type R, create a new relation S (relationship relation) to represent R - Include the primary keys of participating relations as foreign key in the new relation S. Also, include any attributes of relationship type as attributes of S - Primary key of S is the combination of both foreign keys. **Step 6: Mapping of Multivalued attributes.** - For each multivalued attribute A in an entity type E or relationship type R, create a new relation S - Include primary key K of E or R as a foreign key attribute in S. - The primary key of S is the combination of A and K. **Step 7: Mapping of N-ary Relationship Types.** - For each n-ary relationship type R, where n>2. Create a new relation S to represent R - Include the primary keys of participating entity types as foreign key in S - Also include any simple attributes of R as attributes of S - The primary key of S is usually the combination of all foreign keys.			
XIII		FIRST NORMAL FORM (1NF) - It states that domain of an attribute must include only atomic (single, indivisible) values. In other words the value of any attributes in a tuple must be a single value from the domain of that attribute. - All attributes must be single valued or atomic. - Example : (any valid example) Normalization into 1NF. a) The relation schema DEPARTMENT that is not in 1NF. b) Example state of relation DEPARTMENT. c) 1NF version of same relation with redundancy.	3 ½ + 3 ½	7	

(a) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS

(b) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

There are three main techniques to achieve 1 NF.

1. Remove the multivalued attribute DLOCATIONS along with the primary key DNUMBER of DEPARTMENT ie …. after normalization, there will be two 1NF relations :
   - DEPARTMENT (DNAME, DNUMBER, DMGREID)
   - DEPT_LOCATIONS (DNUMBER, DLOCATION)
2. Keep a separate tuple corresponds to each location of the department in the original DEPARTMENT relation itself. Then the primary key becomes { DNUMBER , DLOCATION } . This relation has but redundancy (As shown above).
3. If the maximum number of values is known for the attribute, we can replace the DLOCATIONS attribute as atomic attributes like DLOCATION1, DLOCATION2, …. etc.

SECOND NORMAL FORM (2NF)

- 2NF is based on the concept of full functional dependency.
- A functional dependency X → Y is full functional dependency if removal of any attribute A from X does not hold the dependency any more.
- A functional dependency X → Y is partial dependency if some attribute A can be removed from X and the dependency still holds.
- Example : In EMP_PROJ relation,

{ SSN, PNUMBER } → HOURS is a full dependency

{ SSN, PNUMBER } → ENAME is partial because

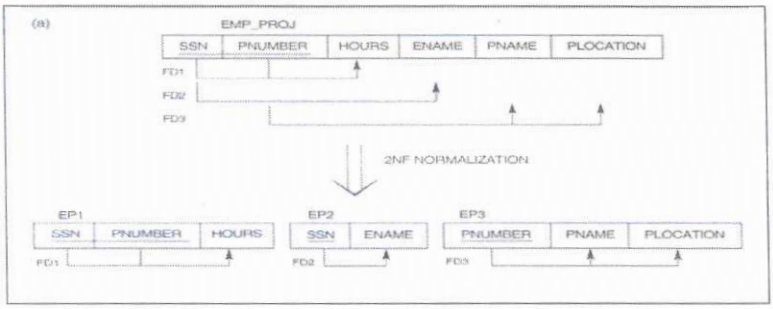
SSN → ENAME holds

Definition of 2NF

- A relation schema R is in 2NF if it holds two conditions
  - It must be in first normal form
  - Every nonprime attribute A in R is fully functionally dependent on the primary key of R.
- Example: (any valid example)
  Normalizing EMP_PROJ into 2NF relations.
- In the below fig (a), the existing FDs are :
  - {SSN , PNUMBER} → HOURS (FD1)
  - SSN → ENAME (FD2) , partial dependency to non-prime

attribute

- PNUMBER → {PNAME , PLOCATION} (FD3, this also partial)



- So, EMP_PROJ is not in 2NF.
- We can decompose this relation into two or more 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent.
- Here, EMP_PROJ can be decomposed as three 2NF relations as follows :

  EP1 (SSN , PNUMBER, HOURS)
  EP2 (SSN , ENAME)
  EP3 (PNUMBER , PNAME, PLOCATION)

XIV

Transaction State Diagram

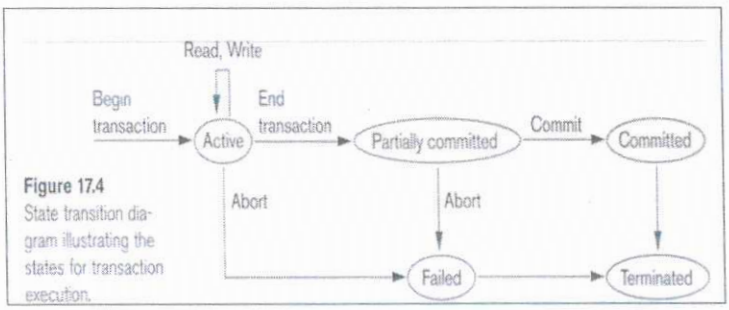


Figure 17.4
State transition diagram illustrating the states for transaction execution.

- **begin transaction**: This marks the beginning of transaction execution.
- **read or write**: These specify read or write operations on the database items that are executed as part of an active transaction.
- **end transaction**: This specifies that read and write transaction operations have completed and marks the end limit of transaction execution. At this point, the database state is partially committed. DBMS can abort the transaction in case of any concurrency violation reported
- **commit transaction** : This signals a successful end of the transaction so that any changes (updates) executed by the transaction can be safely committed to the database and will not be undone.
- **rollback (or abort)**: This signals that the transaction has ended unsuccessfully, so that any changes or effects that the transaction may have applied to the database must be undone.

Diagram: 4

Explanation: 3

7

****