GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

Seshadri Rao Knowledge Village, Gudlavalleru – 521 356.

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

HANDOUT

on

DATABASE MANAGEMENT SYSTEMS

Vision

To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society

Mission

- To impart quality education through well-designed curriculum in tune with the growing software needs of the industry.
- To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society.
- To serve our students by inculcating in them problem solving, leadership, teamwork skills and the value of commitment to quality, ethical behaviour & respect for others.
- To foster industry-academia relationship for mutual benefit and growth.

Program Educational Objectives

PEO1 : Identify, analyze, formulate and solve Computer Science and Engineering problems both independently and in a team environment by using the appropriate modern tools.

PEO2 : Manage software projects with significant technical, legal, ethical, social, environmental and economic considerations.

PEO3 : Demonstrate commitment and progress in lifelong learning, professional development, leadership and Communicate effectively with professional clients and the public.

HANDOUT ON DATABASE MANAGEMENT SYSTEM

II B.Tech – II Semester Year: 2019-20

Branch: **CSE** Credits: **3**

==========

- **Brief History and Scope Of The Subject**

  Database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies. A **database management system** (**DBMS**) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases

- **Pre-Requisites**

  Students need to be aware of different storage mechanisms used for data storage

- **Course Objectives:**

  a. To familiarize the concepts of database systems and different issues involved in the database design.

  b. To introduce how to write SQL for storage,retrieval and manipulation of data in a relational database.

- **Course Outcomes:**

  Upon successful completion of the course, the students will be able to

  CO1: recognize the importance of database system over file processing system.

  CO2: analyze an information storage problem and derive an information model in the form of an entity relationship diagram.

  CO3: write simple and complex queries using structured query Language(SQL) for storage, retrieval and manipulation of data in a relational database.

CO4: employ principles of normalisation for designing a good relational database schema.

CO5: describe the issues and techniques relating to concurrency and database recovery in a multi-user database environment.

Program Outcomes:

Graduates of the Computer Science and Engineering Program will have Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and wit society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

- **Mapping of Course Outcomes with Program Outcomes:**

	1	2	3	4	5	6	7	8	9	10	11	12
CO1		3			2							
CO2	2	2	1	3								
CO3	3	3		2							1	
CO4	2		3		1							
CO5	2		3		1							

3- High **2- Medium** **1-Low**

- **Prescribed Text Books**
  1. Korth & Sudarshan, Database system concept, MH.
  2. Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, MH.
- **Reference Text Books**
  1. Elmasri Navrate, Fundamentals of Database Systems, Pearson Education.
  2. C.J.Date, Introduction to Database Systems, Pearson Education.

3. Peter Rob & C Coronel, Database Systems design, Implementation, and Management, 7th Edition.

- **URLs and Other E-Learning Resources**
  1. **http://www.w3schools.com/sql/**
  2. **http://www.mysqltutorial.org/**
  3. **http://www.java2s.com/Tutorial/Oracle/CatalogOracle.htm**
  4. **http://www.oracle.com/technetwork/tutorials/index.html**
- **Digital Learning Materials:**
  1. https://www.youtube.com/watch?v=rbwXdTsCk2c
  2. https://www.youtube.com/watch?v=EUzsy3W4I0g
- **Lecture Schedule / Lesson Plan**

TOPIC	No. of Periods	
	Theory	Tutorial
UNIT –1: INTRODUCTION TO DATABASE		
Advantages of using DBMS	1	2
Data Models, Schema and instances	2	
Levels of abstraction	1	
Entity- Relationship Model- Attributes and Keys	1	
Relationship Types,Weak Entity set, Strong Entity Set	1	
Enhanced E–R Modeling- specialization,generalization	1	
database design for banking enterprise	1	
reduction to relational schemas	3	
UNIT – 2: RELATIONAL MODEL & SQL		
Relational model concepts	1	2
constraints, keys, relational algebra	1	
SQL- DDL, DML	1	
Set operations, Aggregate Functions	1	
Null values, Nested queries	2	
Defining different constraints on a table	2	
Group by, having clauses	1	

UNIT – 3: DATABASE DESIGN		
Functional Dependencies: partial, full, transitive dependencies	2	3
Axioms	1	
attribute closure	2	
Lossless join, dependency preserving decomposition	2	
Normal forms-First, second third normal forms	2	
Boyce- Codd normal form	1	
UNIT – 4: TRANSACTION MANAGEMENT		
Transaction Management- Transaction concept	1	2
ACID properties, transaction state diagram	2	
Schedules: serial, concurrent, serializable	2	
serializability of schedules	3	
recoverability	2	
UNIT – 5: CONCURRENCY CONTROL		
Concurrency control	1	3
Concurrent execution of transactions, anomalies	2	
Lock-based protocols:2PL, strict 2PL, rigorous 2PL	1	
Timestamp-based protocols	2	
Thomas write rule, Deadlock handling: deadlock prevention	2	
Deadlock detection and recovery	1	
UNIT – 6: CRASH RECOVERY		
Failure classification	1	2
Different types of Recovery techniques-deferred update, immediate update	3	
Shadow paging, Check points	3	
Total No. of Periods:	**56**	**14**

- **Seminar Topics**
  - Serializability
  - Normalization
  - Transaction Management
  - Lock-based protocols
  - Crash recovery techniques

UNIT-1
DATABASE MANAGEMENT SYSTEMS

Objectives:

To provide the basic knowledge about the database management system

Syllabus:

Introduction to Database Purpose of Database Systems, Data models, Schema and instances, DBMS architecture, Entity- Relationship model- Attributes and Keys, Relationship types, Weak Entity set, Strong Entity Set, enhanced E–R Modelling- Specialization and generalization, Database design for Banking enterprise, reduction to relational schemas.

Learning Outcomes:

At the end of the unit, students will be able to

- differentiate database systems from file systems by enumerating the features provided by the database system.
- describe fundamental elements of a database management system.
- design entity relationship diagrams to represent simple database application scenarios.
- convert entity relationship diagrams into relational tables, populate a relational database and formulate SQL queries on the data.
- criticize a database design and improve the design by normalization.
- interpret the basic issues of transaction processing and concurrency control.

- **Introduction to Database Management Systems**
  - **Database -** Database is a large integrated collection of data
  - **Database Management System (DBMS) –** Database management system is a collection of interrelated data and a set of programs to

access those data. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

- Databases are usually designed to manage large bodies of information. This involves
    - Definition of structures for information storage (data modeling).
    - Provision of mechanisms for the manipulation of information (File and systems structure, query processing).
    - Providing for the safety of information in the database (crash recovery and security).
    - Concurrency control if the system is shared by users.

- **Purpose of Database Systems**
    - To see why database management systems are necessary, let's look at a typical file processing system supported by a conventional operating system.
    - The application is a savings bank:
        - Savings account and customer records are kept in permanent system files.
        - Application programs are written to manipulate files to perform the following tasks:
            - Debit or credit an account.
            - Add a new account.
            - Find an account balance.
            - Generate monthly statements.
    - Development of the system proceeds as follows:

- New application programs must be written as the need arises.
- New permanent _les are created as required

- But over a long period of time files may be in different formats, and Application programs may be in different languages.
- So we can see there are problems with the straight file processing approach:
- **Data redundancy and inconsistency**
  - Same information may be duplicated in several places.
  - All copies may not be updated properly.
- **Difficulty in accessing data**
  - May have to write a new application program to satisfy an unusual request.
  - E.g. find all customers with the same postal code.
  - Could generate this data manually or need to write an application program for getting the desired information. Both these activities are very difficult.
- **Data isolation**
  - Data in different files.
  - Data in different formats.
  - Difficult to write new application programs to access the data
- **Integrity problems**
  - Data may be required to satisfy constraints.
  - E.g. no account balance below $25.00.

- Again, difficult to enforce or to change constraints with the file-processing approach.

- **Atomicity Problems**
  - A computer system, like any other mechanical or electrical device, is subject to failure.
  - In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.
  - E.g. Consider a program to transfer $500 from account A to account B. If a system failure occurs during the execution of the program, it is possible that the $500 was debited from account A but not credited to account B, resulting in an inconsistent database state.
  - Clearly, it is essential to database consistency that either both credit or debit occurs or that neither occurs. i.e. the funds transfer must be atomic – it must happen in its entirety or not at all.
  - It is difficult to ensure atomicity in a conventional file processing system.

- **Concurrent Access anomalies**
  - Want concurrency for faster response time.
  - Need protection for concurrent updates.
  - E.g. two customers withdrawing funds from the same account at the same time - account has $500 in it, and they withdraw $100 and $50. The result could be $350, $400 or $450 rather the correct value of $350.

- It is very difficult to guard against concurrent executions in file processing system because data may be accessed by many different application programs that have not been coordinated previously.

- **Security problems**
  - Every user of the system should be able to access only the data they are permitted to see.
  - E.g. payroll people only handle employee records, and cannot see customer accounts; tellers only access account data and cannot see payroll data.
  - Difficult to enforce this with application programs.

These problems and others led to the development of database management systems.

- **Data Abstraction**
  - The major purpose of a database system is to provide users with an abstract view of the system. The system hides certain details of how data is stored and maintained.
  - Complexity should be hidden from database users.
  - There are three levels of abstraction:

Physical Level

- Deals with how the data are stored.
- E.g. index, B-tree, hashing.
- Lowest level of abstraction.
- Complex low-level structures described in detail.

Conceptual Level

- Next highest level of abstraction.
- Describes what data are stored.
- Describes the relationships among data.
- Database administrator level.

View Level

- Highest level.
- Describes part of the database for a particular group of users.
- Can be many different views of a database.
- E.g. tellers in a bank get a view of customer accounts, but not of payroll data.

Figure 1.1 illustrates three levels of abstraction.

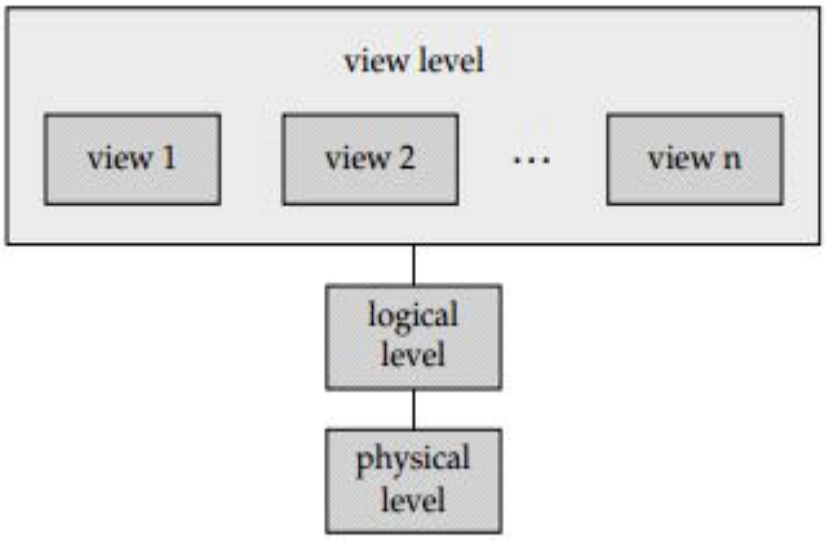


Figure 1.1 The three levels of data abstraction.

- **Data models**
  - Data models are a collection of conceptual tools for describing data, data relationships, data semantics and data constraints. Some of the popular data models are:

Entity-Relationship model

- In this, data is represented as a collection of entities and relationships among the entities.
- An entity is a thing or object in the real world that is distinguishable from other objects.

- Each entity is described by set of attributes.
- E.g. ***Customer*** entity is described by the attributes *customer-id*, *customer-name*, *customer-street,* and *customer-city*
- A relationship is an association among several entities
- E.g. A ***borrower*** relationship associates a Customer entity with Account entity
- The set of all entities of the same type is called as the entity set.
- Similarly, the set of all relationships of the same type is called as the relationship set.
- The overall logical structure of a database can be expressed graphically by an E-R diagram where:
  - **rectangles:** represent entity sets.
  - **ellipses:** represent attributes.
  - **diamonds:** represent relationships among entity sets.
  - **lines:** link attributes to entity sets and entity sets to relationships.

Figure 1.2 illustrates a sample ER diagram.

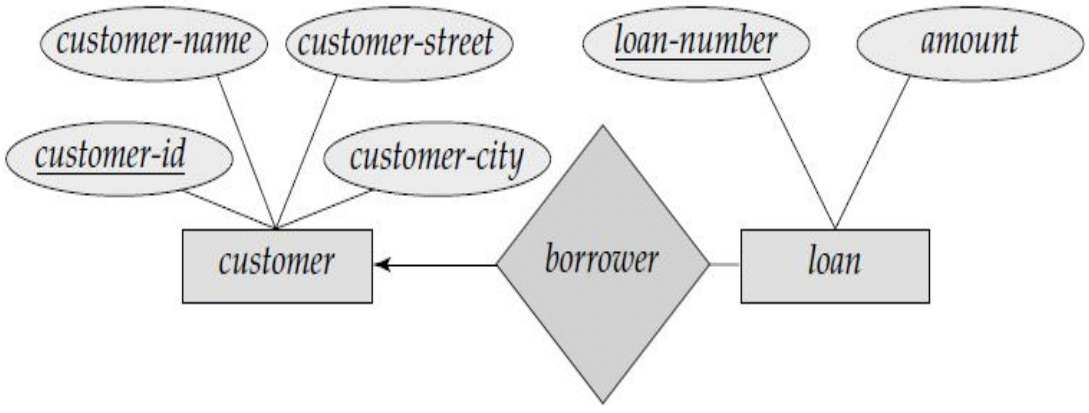


Fig. 1.2: a sample ER diagram

Relational Model

- In this, data is represented as a collection of tables also known as relations. E.g. customer table, account table
- Each table has a number of columns with unique names. E.g. customer table has *customer-id*, *customer-name*, *customer-street*, and *customer-city columns*.
- Similarly, loan table has *loan-number* and *amount* columns.
- Figure 1.3 shows a sample relational model.

customer_id	*customer_name*	*customer_street*	*customer_city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account_number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

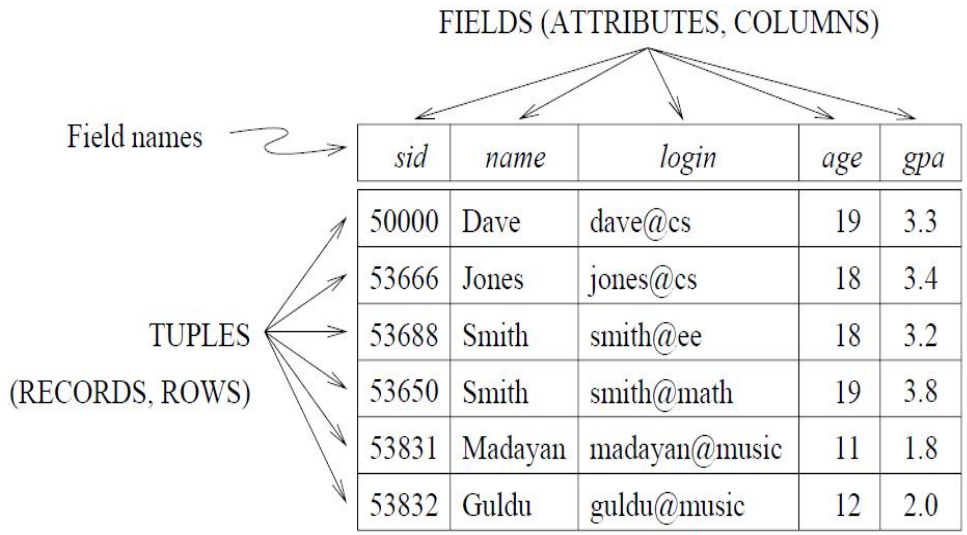
(b) The *account* table

customer_id	*account_number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Figure 1.3: A sample Relational Model.

- The degree, also called arity, of a relation is the number of fields in it.
- The cardinality of a relation is the number of tuples in it.
- For example, for the relation shown in figure 1.4, the degree of the relation is 5, and cardinality is 6.



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Fig. 1.4: student relation

Network Model

- In this, data are represented by collections of records.
- Relationships among data are represented by links.
- Collection of records are organised as an arbitrary graph.
- Figure 1.5 shows a sample network model.

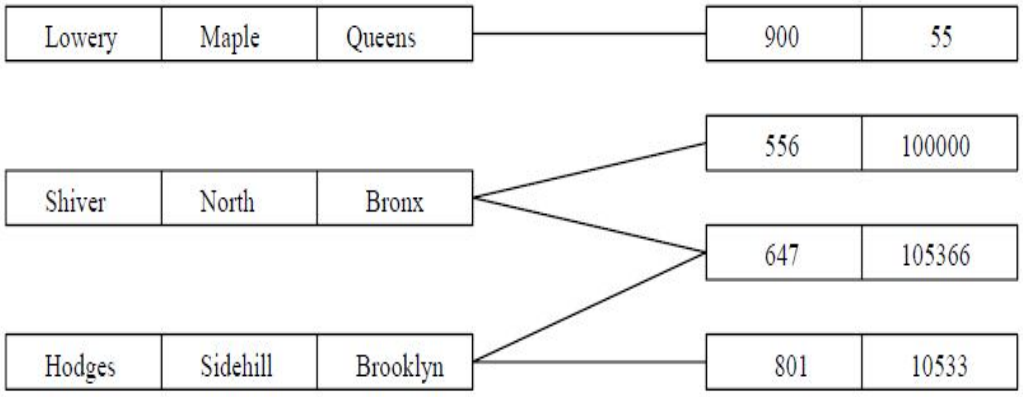


Fig. 1.5: A sample Network Model

Hierarchical Model

- It is similar to the network model.
- Organizes the collection of the records as trees, rather than arbitrary graphs.
- Figure 1.6 shows a sample hierarchical database.

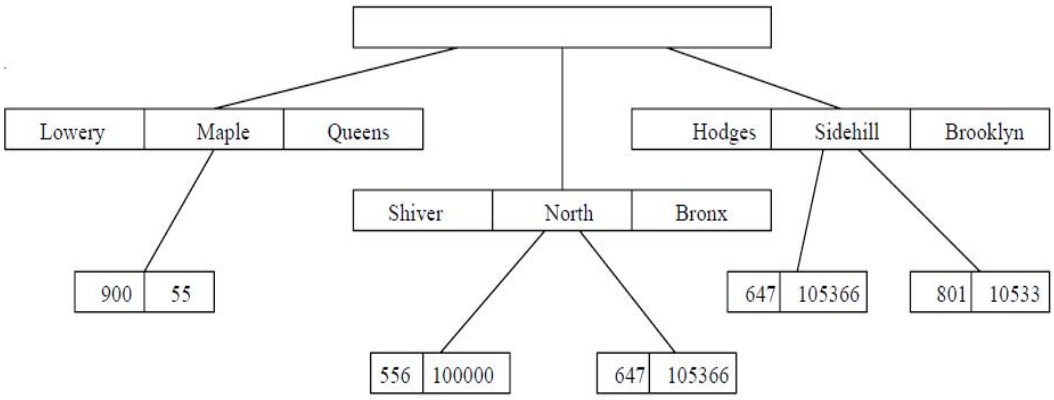


Fig. 1.6: A sample Hierarchical Model.

Object Relational Model

- Combines the features of the object-oriented data model and relational data model.

- **Instances and Schemas**
  - Databases change over time.
  - The information in a database at a particular point in time is called an instance of the database.
  - The overall design of the database is called the database schema. In other words, a database schema consists of collection of relational schemas.
  - The overall structure of the relation (or table) is known as relation schema. E.g. student relation schema is shown as

    student (sid, name, login, age, gpa)

- The information in a relation (or table) at a particular point in time is called an instance of the relation. E.g. figure 1.7 shows instance of the *student relation.*

sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Fig. 1.7: Instance of *student relation*

Fig. 1.8: Database system structure

- **Entity-Relationship Model**
  - In this, data is represented as a collection of entities and relationships among the entities.
  - An entity is a thing or object in the real world that is distinguishable from other objects.
  - Each entity is described by set of attributes.
  - E.g. ***Customer*** entity is described by the attributes *customer-id*, *customer-name*, *customer-street,* and *customer-city*
  - A relationship is an association among several entities
  - E.g. A ***borrower*** relationship associates a Customer entity with Account entity
  - The set of all entities of the same type is called as the entity set.

- Similarly, the set of all relationships of the same type is called as the relationship set.
- The overall logical structure of a database can be expressed graphically by an E-R diagram where:
  - **rectangles:** represent entity sets.
  - **ellipses:** represent attributes.
  - **diamonds:** represent relationships among entity sets.
  - **lines:** link attributes to entity sets and entity sets to relationships.
  - **Double ellipses:** represent multivalued attributes.
  - **Dashed ellipses:** denote derived attributes
  - **Double lines:** indicate total participation of an entity in a relationship set (described later).
  - **Double rectangles:** represent weak entity sets (described later)

Figure 1.9 shows a sample ER diagram.

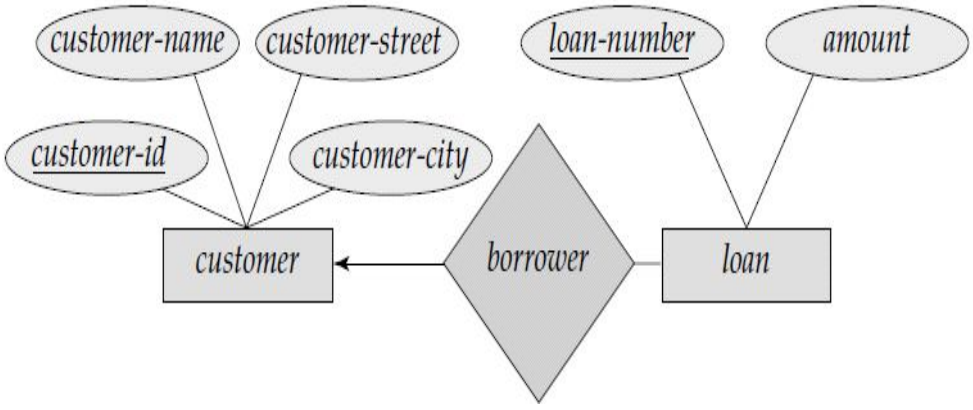


Fig. 1.9: A sample E-R diagram

7.1 Attributes

- Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).

- Attributes are categorized into following types:
- Simple attribute – is an attribute that cannot be divided into sub parts.
  E.g. account-number
- Composite attribute – is an attribute that can be divided into sub parts.
  E.g. customer-name (It can be divided into sub parts as first_ name, middle _name and last_ name).

The following figure (Figure 1.10) shows composite attributes.

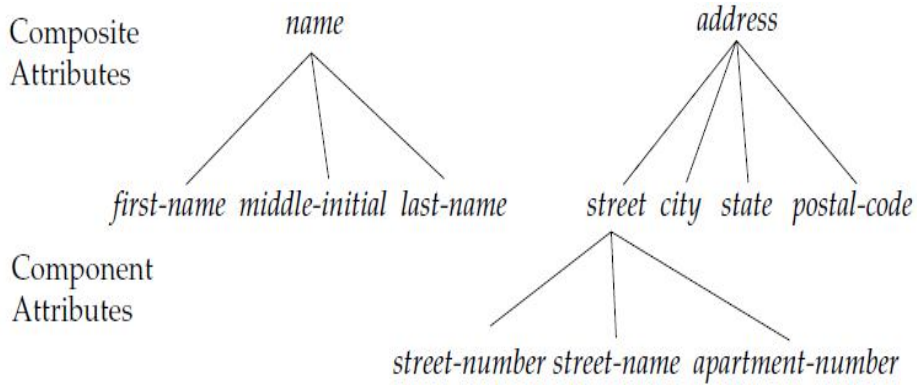


Fig. 1.10: Composite attributes

- **Single valued attribute:** an attribute that takes only one value.
  E.g. rollnumber
- **Multi valued attribute:** an attribute that takes more than one value.
  E.g. phone-number
- **Derived attribute:** attribute can be derived from the values of other related attributes.
  E.g. age is a derived attribute as its value can be derived from date_of_birth attribute.

The following figure (Figure 1.11) shows an ER diagram containing composite, multivalued and derived attributes.

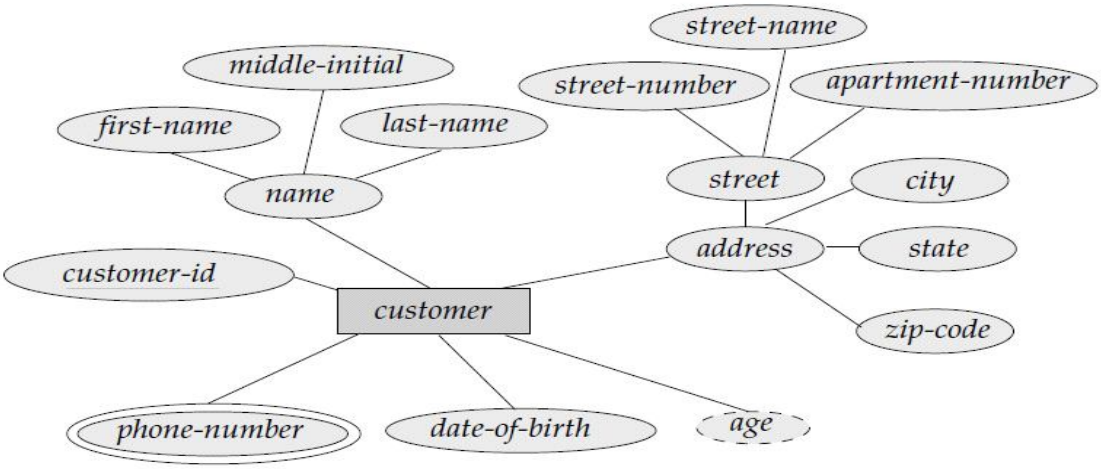


Fig. 1.11: E-R Diagram that shows composite, multivalued and derived attributes.

7.2 Keys

- Key is a set of attributes that uniquely identify an entity set from set of entities.
- E.g. suppose student entity consists of the attributes (*Rollno, Name, DOB, Address*) then the attribute *Rollno* identifies uniquely a student from set of students. Hence, *Rollno* is the key for student entity.

- **Relationship Types**
  - Relationship is an association among two or more entities.
  - There are three kinds of relationships
- **Unary Relationship** – is a relationship that associates an entity with the same entity. E.g. Figure 1.12 shows Unary Relationship(works-for)

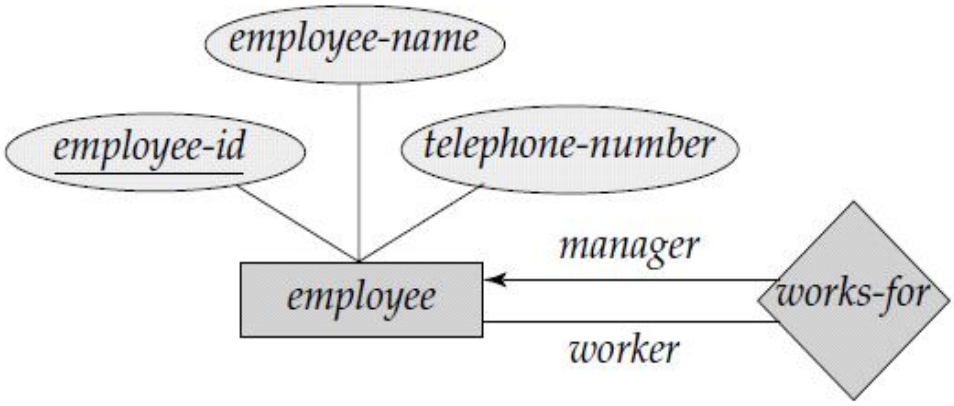


Fig. 1.12: An example Unary Relationship.

- **Binary Relationship** - is a relationship between any two entities.
  E.g. Figure 1.13 shows a binary relationship (depositor) between the entities *customer* and *account*.

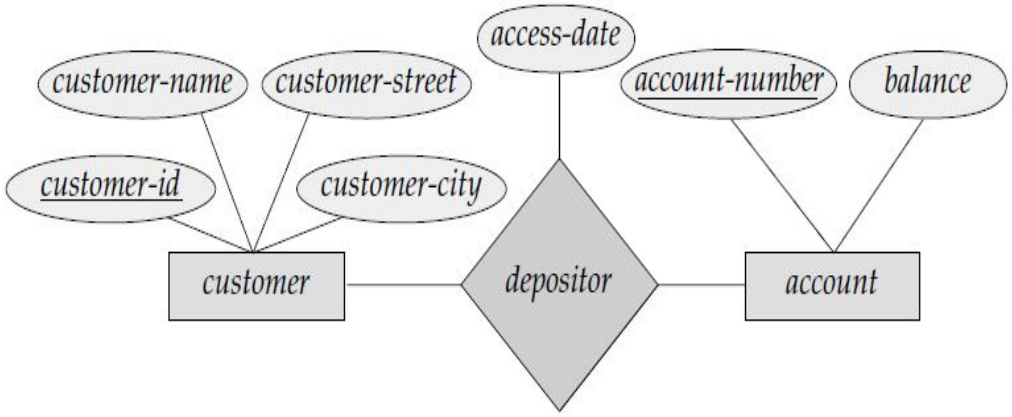


Fig. 1.13: An example Binary Relationship.

- **Mapping Cardinalities** - Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.
- Mapping cardinalities are most useful in describing binary relationship sets.
- For a binary relationship R between entity sets A and B, the mapping cardinalities must be one of the following:
  - **One to one.** An entity in A is associated with at most one entity in B. and an entity in B is associated with at most one entity in A.
  - **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.

  Figure 1.14 shows one-to-one and one-to-many mapping cardinalities.

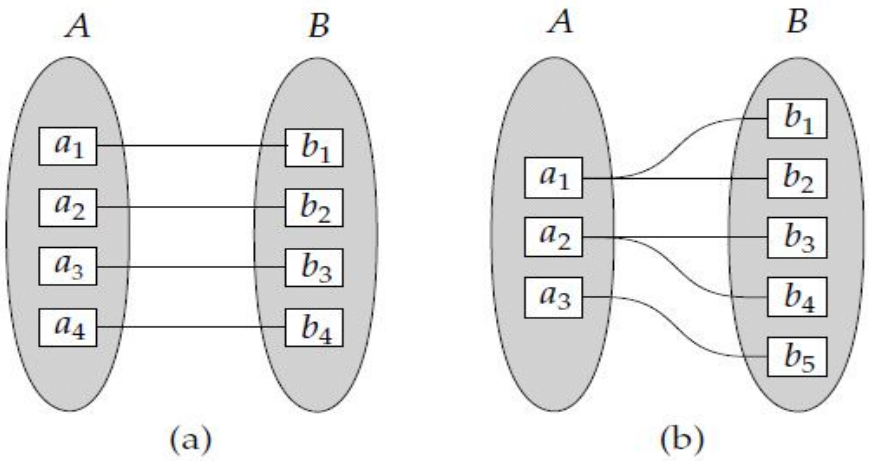


Fig. 1.14: Mapping cardinalities (a) one to one (b) one to many

- **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.

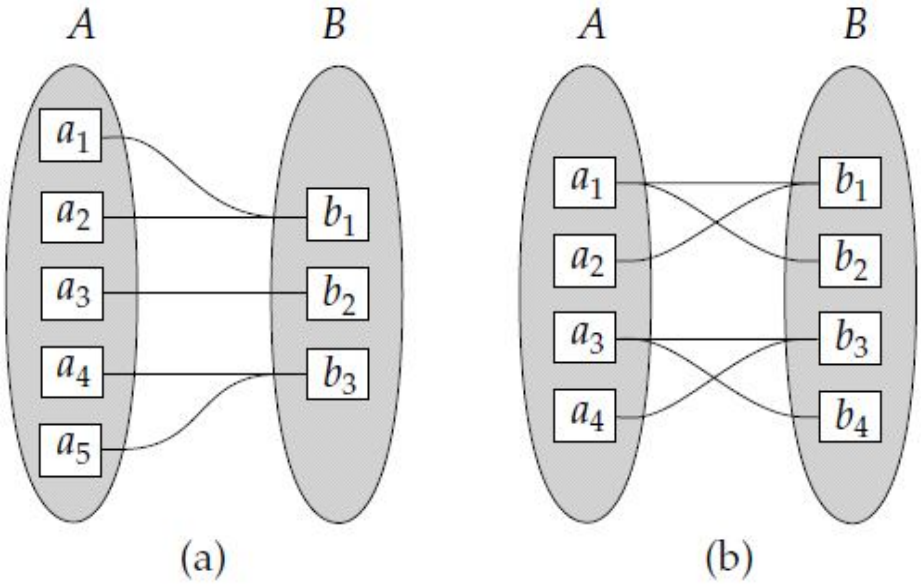


Fig. 1.15: Mapping cardinalities (a) many to one (b) many to many

- **Many to many**. An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

Figure 1.15 shows many-to-one and many-to-many mapping cardinalities.

- **Ternary Relationship** A ternary relationship is one where three entities participate in the relationship. Figure 1.16 shows a ternary relationship.

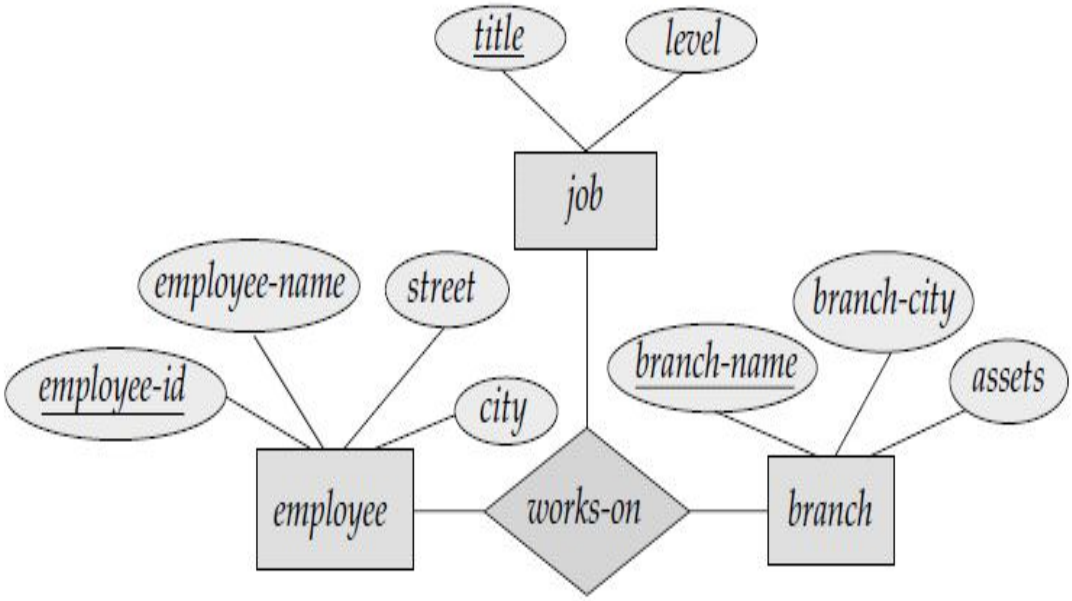


Fig. 1.16: An example ternary relationship.

7.4 Participation constraints.

- Participation constraints describe how the entities are participating in the relationship.
- There are two types of participation constraints.
  - Total Participation – Every entity in the entity set participates in the relationship.
  - Partial Participation – Only some of the entities in the entity set participates in the relationship.

The following diagram (Figure 1.17) shows participation constraints.

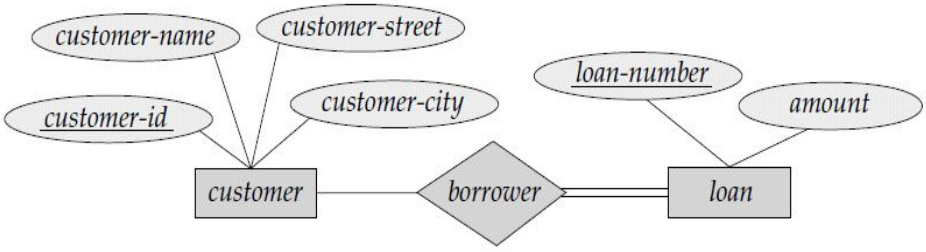


Fig. 1.17: ER diagram showing participation constraints.

- In this diagram, only some of the customers borrow loan hence, customer participation in borrower relationship is partial. Where as for every loan there is a borrower hence, loan participation in borrower relationship is total.

- **Weak entity set –** Entity set without primary key is termed as a weak entity set.
  - A weak entity set can be identified uniquely only by considering some of its attributes in conjunction with the primary key of another entity, called the owner set.
  - Owner entity set and weak entity set must participate in a one-to-many relationship.
  - This relationship set is called the identifying relationship set of the weak entity set. The weak entity set must have total participation in the identifying relationship set.

- **Strong entity set -** An entity set that has a primary key is called as a strong entity set.

  E.g. Figure 1.18 shows weak and strong entity sets. Here, ***payment*** is a weak entity set and ***loan*** is a strong entity set. Primary key for ***loan*** is **loan-number** and primary key for ***payment*** is **{loan-number, payment-number}.**

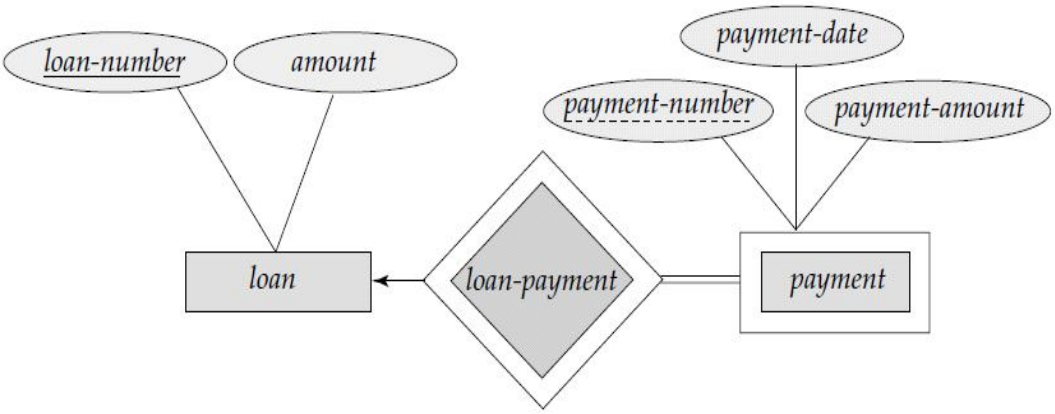


Figure1.18 Weak entity set.

- **Enhanced ER modeling**
  - Enhanced ER modelling includes specialization and generalization.
  - **Specialization** – The process of identifying subsets of an entity set that share some distinguishing characteristics is known as specialization.
  - It is also called as top-down design process.
  - Here, after identifying a general entity, its sub entities are identified that share the common characteristics of the general entity.
  - **Generalization** – The process of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities possessing these common characteristics is known as generalization.
  - It is also called as bottom-up design process.
  - In ER diagram, generalization/specialization is denoted by ISA relationship.
  - We can inherit the properties of super class (general entity) into sub class (specialized entities) by ISA relationship.

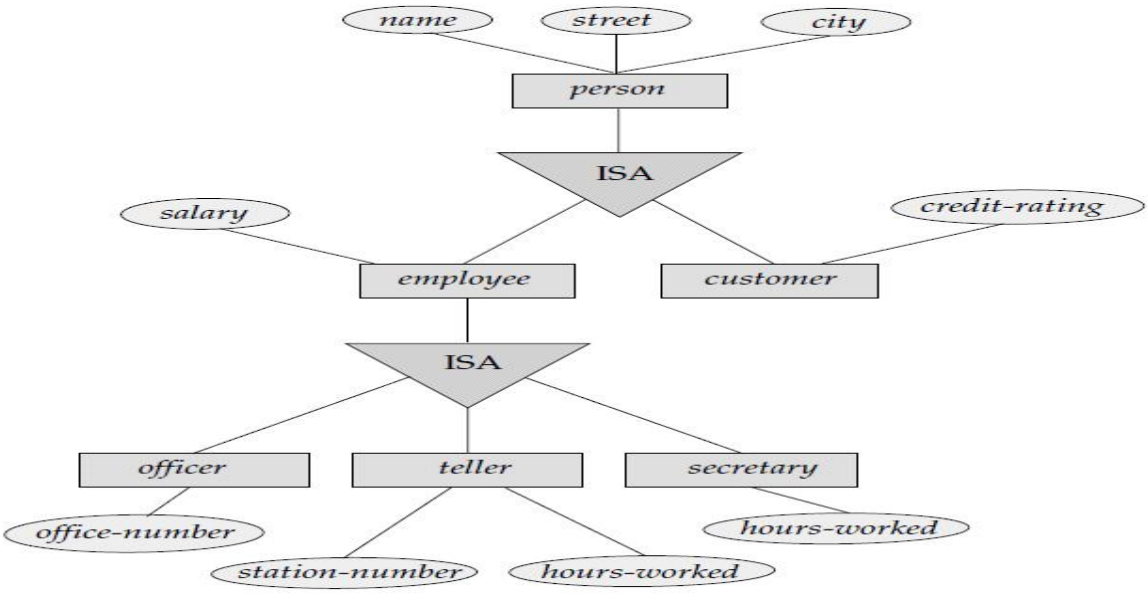


Fig. 1.19: An example of Specialization and Generalization.

- We can specify two kinds of constraints with respect to ISA hierarchies, namely, *overlap* and *covering* constraints. Overlap constraints determine whether two subclasses are allowed to contain the same entity. Covering constraints determine whether the entities in the subclasses collectively include all entities in the superclass.
  E.g. Figure 1.19 shows specialization and generalization.

Aggregation

- The E-R model cannot express relationships among relationships.
- Aggregation is used to express relationships among relationships
- As an example consider the following figure. Here, we treat the relationship set *work* and the entity sets *employee* and *project* as a higher-level entity set called ***work*** and this aggregate component can be related with ***machinery*** entity using ***uses*** relationship. It means that employee working on a project uses machinery.

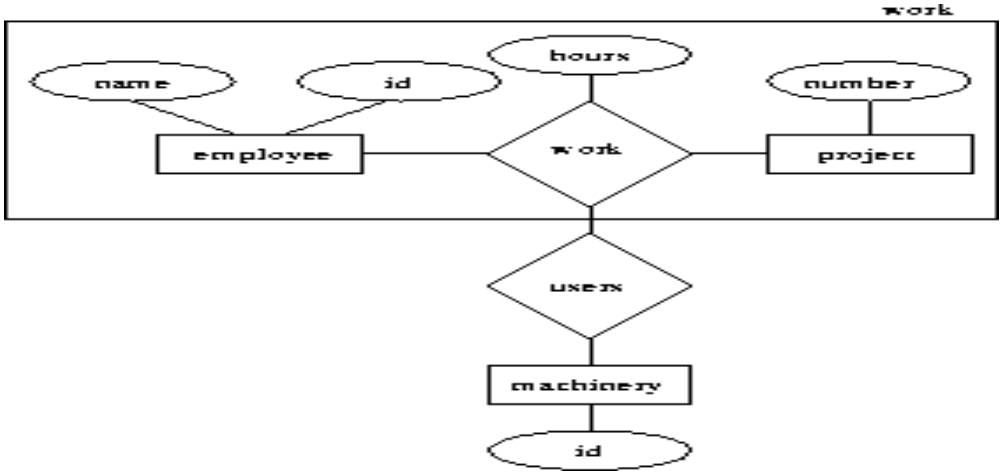


Fig. 1.20**:** E-R diagram with aggregation

- Transforming an E-R diagram with aggregation into tabular form is easy. We create a table for each entity and relationship set as before.
- The table for relationship set *uses* contains a column for each attribute in the primary key of *machinery* and *work*.

- **Database Design For Banking Enterprise**
  - To design the database for Banking Enterprise, first the requirements of Banking Enterprise are gathered and specified. From this

requirement specification, the conceptual design of database is created. Here are the major characteristics of the banking enterprise;

- The bank is organized into branches. Each branch is located in a particular city and is identified by a unique name. The bank monitors the assets of each branch.
- Bank customers are identified by their *customer-id* values. The bank stores each customer's name, and the street and city where the customer lives. Customers may have accounts and can take out loans. A customer may be associated with a particular banker, who may act as a loan officer or personal banker for that customer.
- Bank employees are identified by their *employee-id* values. The bank administration stores the name and telephone number of each employee, the names of the employee's dependents, and the *employee-id* number of the employee's manager. The bank also keeps track of the employee's start date and, thus, length of employment
- The bank offers two types of accounts—savings and checking accounts. Accounts can be held by more than one customer, and a customer can have more than one account. Each account is assigned a unique account number. The bank maintains a record of each account's balance, and the most recent date on which the account was accessed by each customer holding the account. In addition, each savings account has an interest rate, and overdrafts are recorded for each checking account.
- A loan originates at a particular branch and can be held by one or more customers. A loan is identified by a unique loan number. For each loan, the bank keeps track of the loan amount and the loan payments. Although a loanpayment number does not uniquely identify a particular payment among those for all the bank's loans, a payment number does identify a

particular payment for a specific loan. The date and amount are recorded for each payment

- the bank would keep track of deposits and withdrawals from savings and checking accounts.

Keeping in mind the above requirements, the design of the database proceeds as follows;

Step1: Identify the entity sets and their attributes

The entity sets that could be identified for Banking Enterprise are:

- *branch* entity set, with attributes *branch-name*, *branch-city*, and *assets*
- *customer* entity set, with attributes *customer-id*, *customer-name*, *customer-street*, and *customer-city*
- *employee* entity set, with attributes *employee-id*, *employee-name*, *telephone-number*, *salary*, and *manager.* Additional descriptive features are the multi-valued attribute *dependent-name,* the base attribute *start-date,* and the derived attribute *employment-length*
- Two *account* entity sets—*savings-account* and *checking-account*—with the common attributes of *account-number* and *balance*; in addition, *savings-account* has the attribute *interest-rate* and *checking-account* has the attribute *overdraft-amount*
- The *loan* entity set, with the attributes *loan-number*, *amount*, and *originating-branch*
- The weak entity set *loan-payment*, with attributes *payment-number*, *payment-date*, and *payment-amount*

Step2: Identify the relationships and their cardinalities

- *borrower*, a many-to-many relationship set between *customer* and *loan*
- *loan-branch* between *loan* and *branch*, a many-to-one relationship set that indicates in which branch a loan originated
- *loan-payment*, a one-to-many relationship from *loan* to *payment*, which documents that a payment is made on a loan

- *depositor*, with relationship attribute *access-date*, a many-to-many relationship set between *customer* and *account*, indicating that a customer owns an account
- *cust-banker*, with relationship attribute *type*, a many-to-one relationship set expressing that a customer can be advised by a bank employee, and that a bank employee can advise one or more customers
- *works-for*, a relationship set between *employee* entities with role indicators *manager* and *worker*; the mapping cardinalities express that an employee works for only one manager and that a manager supervises one or more employees

Step3: Draw the E-R diagram with the identified entity sets and their relationships

Figure 1.21shows the ER diagram for banking enterprise.

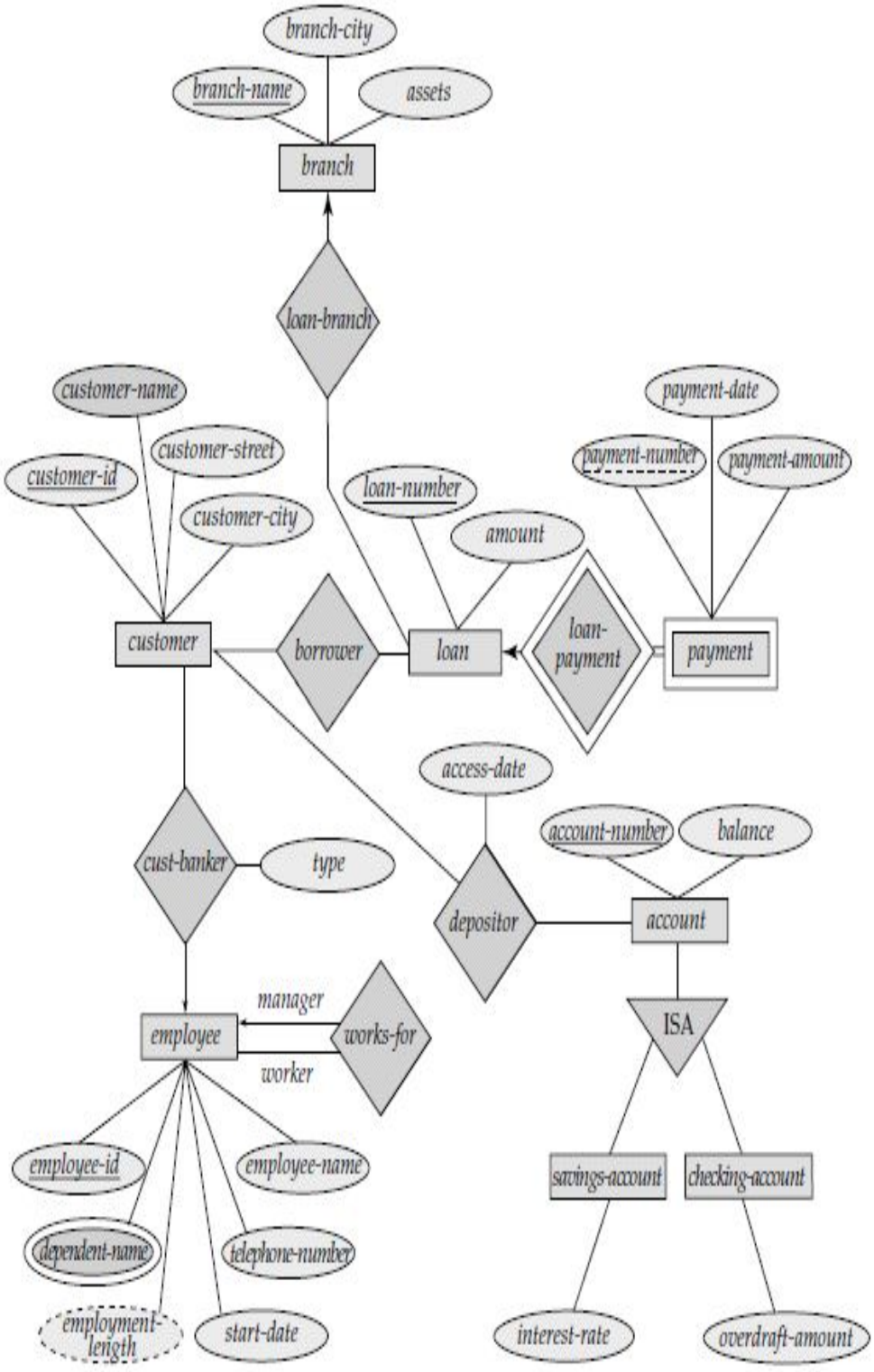


Fig. 1.21: ER diagram for banking enterprise

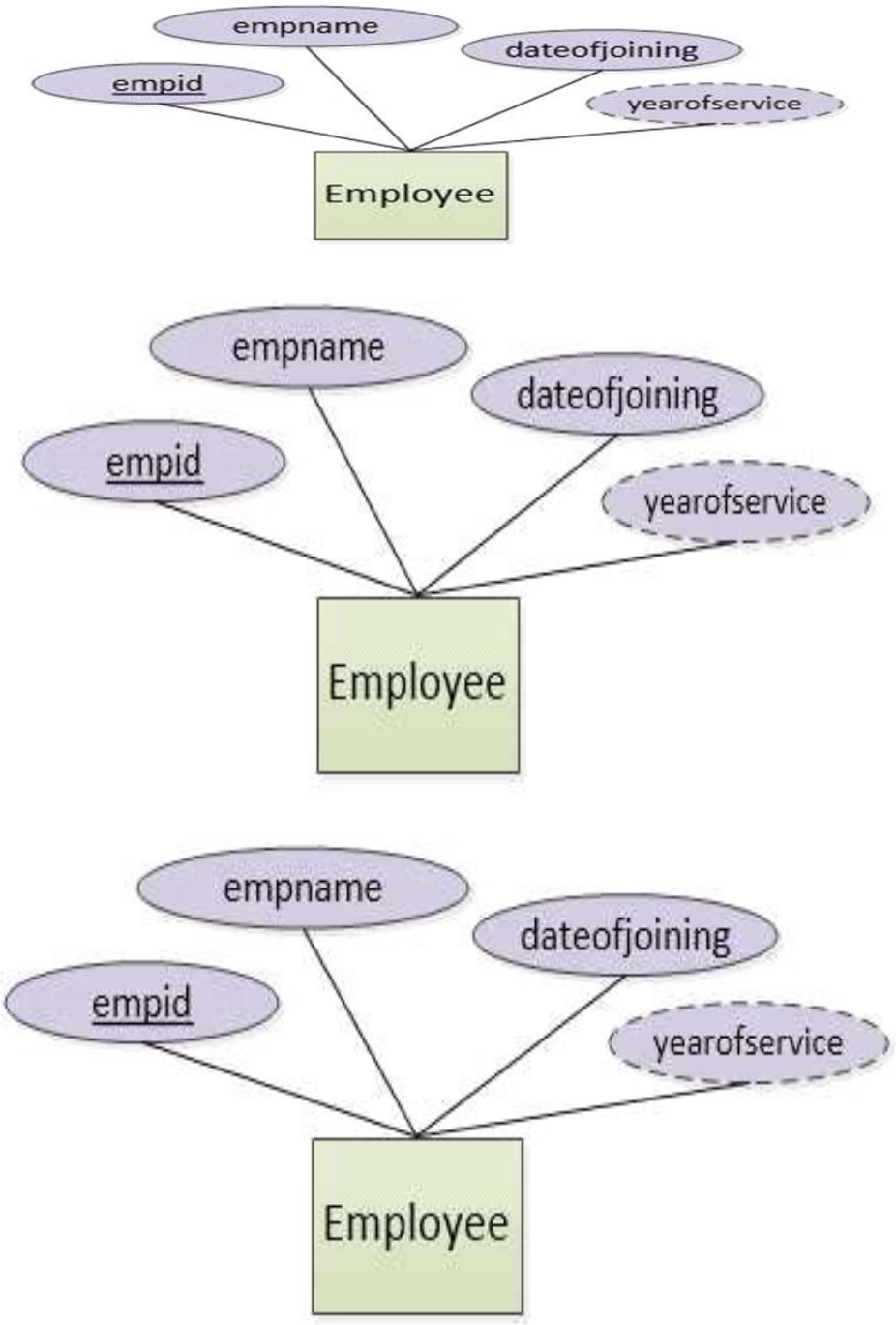
- **Reduction to Relational Schemas**

  Conversion Guidelines

  - Each entity in ER diagram becomes a table in relational schema.
  - Each single-
    valued attribute in ER diagram becomes a column of the table.
  - Derived attributes of entities are ignored.
  - Composite attributes of an entity are represented by its equivalent par ts.
  - Multi-valued attributes are kept in a separate table.
  - The key attribute of an entity is chosen as the primary key of the tabl e.
  - Converting relationships is based on degree and cardinality of relation ship.

- **Conversion of strong entity set**

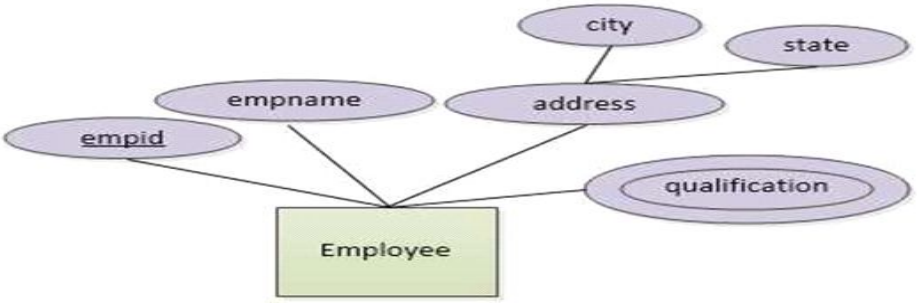
  Example 1:



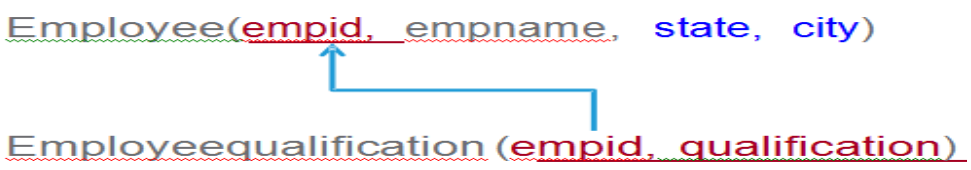
Relational Schema:

Employee(empid, empname, dateofjoining)

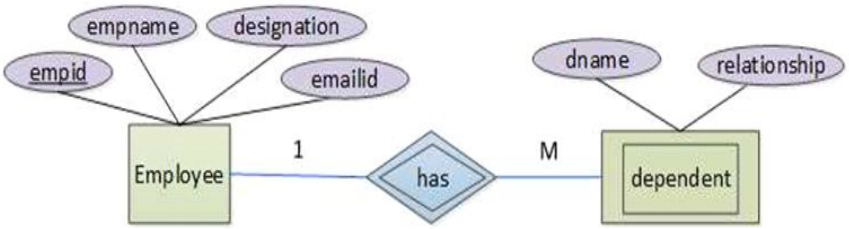
Example 2: Conversion of Strong Entity Set when it has multi-valued attribute



Relational Schema:



Example 3: Conversion of Weak entity set



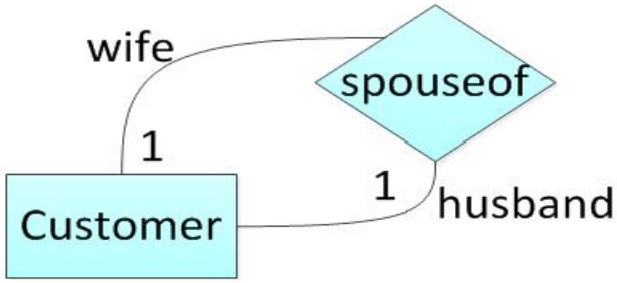
Relational Schema:

employee(empid,empname, designation, emailid)

dependent (empid, dname, relationship)

- **Conversion of Unary Relationship (1:1)**

Example 1:

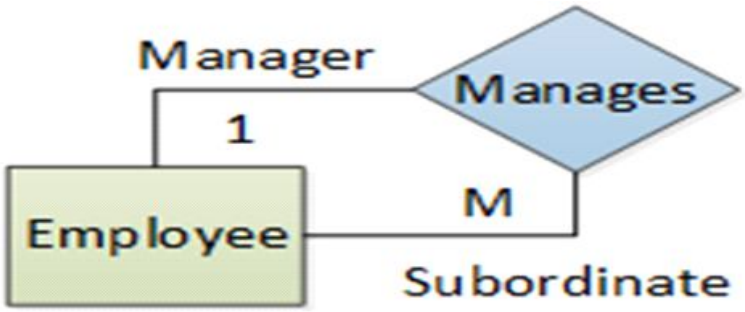


Relational Schema:

customer(customerid, customername,…spouse)

Example 2: Unary Relationship (1:M)

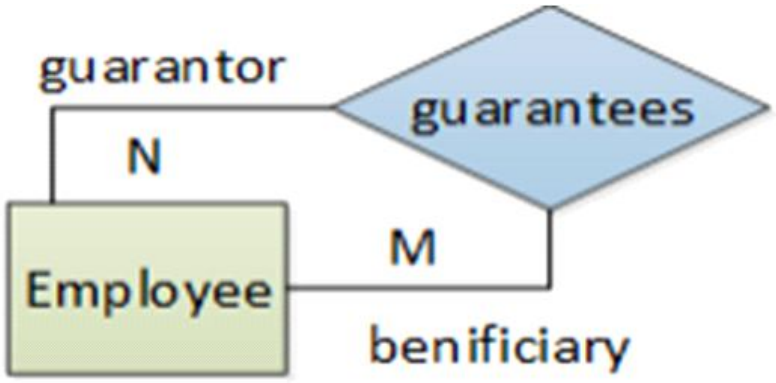
The primary key of the table will itself become foreign key of t he same table



Relational Schema:

Employee(empid, empname, dateofjoining, designation, ……, managerid)

Example 3: Unary Relationship (N:M)



Relational Schema:

employee(empid, empname, designation,....)

guaranty(guarantorid, beneficiaryid)

- **Conversion of Binary Relationship (1:1)**

  Example 1:

  The key attribute of any of the participating entities in a relationship can become a foreign key in the other participating entity.

Relational Schema:

employee(empid, empname, designation, ..., salary)

retailoutlet (retailoutletid, retailoutletlocation, retailouletmanagerid)

OR

employee(empid, empname, designation,.....,salary, retailoutletid)

retailoutlet (retailoutletid, retailoutletlocation)

Example 2: Binary Relationship (1: M)

The key attribute of entity on the "1" side of the relationship becomes a foreign key of entity towards the "M" side.

Relational Schema:

supplier (supplierid, suppliername, suppliercontactno, supplieremailid)

quotation (quotationid, itemcode, quotedprice, supplierid)

Example 3: Binary Relationship (N: M)

Relational Schema:

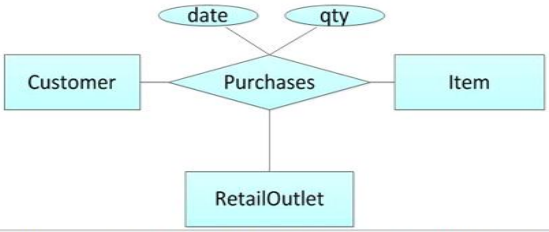
customer (customerid,customertype,customername,emailid,contactno,address)

retailoutlet (retailoutletid, retailoutletlocation)

purchasesfrom (customerid, retailoutletid)

- **Conversion of Ternary relationship**

Example 1:



Relational Schema:

customer (customerid,customertype,customername,emailid,contactno,address)

retailoutlet (retailoutletid, retailoutletlocation)

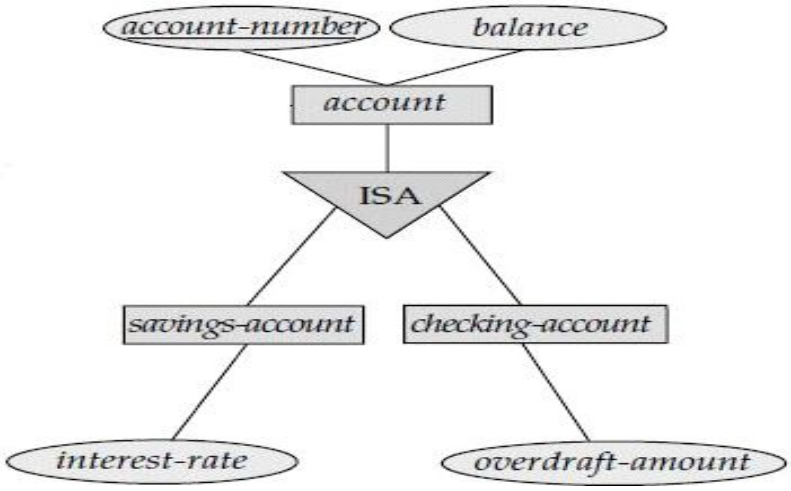
item (itemid, description, reorderlevel, itemclass)

purchases(customerid, retailoutletid, itemcode, date, qty)

- **Conversion of Generalization**

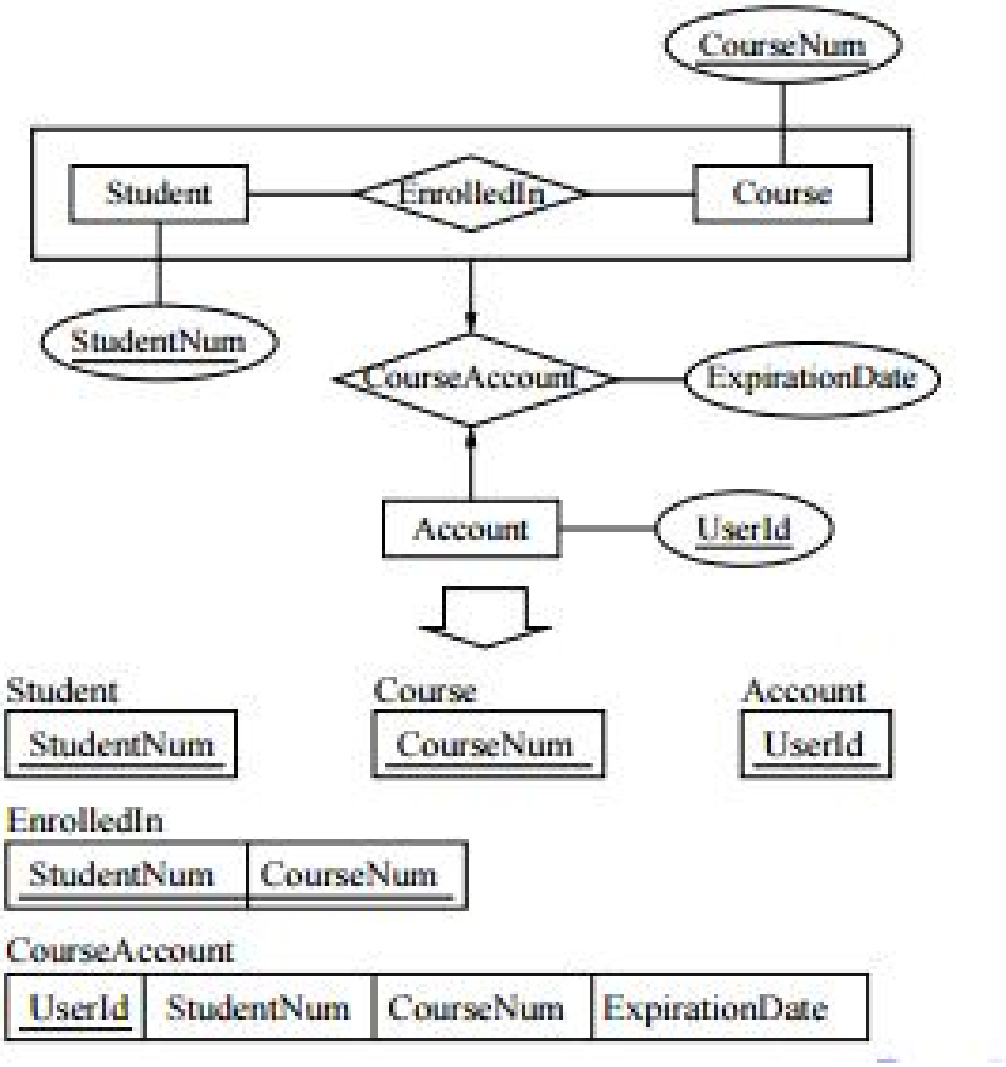
There are two different methods for transforming generalization to tabular form.

- Create a table for the generalized entity set. For each specialized entity set, create a table that includes a column for each of the attributes of that entity set plus a column for the primary key of the generalized entity set. Thus, for the E-R diagram shown below, we have three tables:
  - *account*, with attributes *account-number* and *balance*
  - *savings-account*, with attributes *account-number* and *interest-rate*
  - *checking-account*, with attributes *account-number* and *overdraft-amount*
- An alternative representation is possible, if the generalization is disjoint and complete. Then, we have two tables.
  - *savings-account*, with attributes *account-number*, *balance*, and *interest-rate*
  - *checking-account*, with attributes *account-number*, *balance*, and *overdraft-amount*



- **Conversion of Aggregation**

Conversion of aggregation into tables is illustrated as shown below;



UNIT-I
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. Database applications were built directly on top of file system to overcome the following drawbacks of using file-systems [ ]

   (a) Data redundancy and inconsistency

   (b) Difficulty in accessing data

   (c) Data isolation

   (d) Integrity problems

   (A) (a) (B) (a) and (d)

   (C) (a), (b) and (c) (D) (a), (b), (c) and (d)

2. Data model is a collection of conceptual tools for describing [ ]

   (A) Data and data relationships (B) Semantics

   (C) Consistency constraints (D) All of the above

3. Identify the correct statement related to Physical level Abstraction:
   (A) describes how a record is stored [ ]
   (B) describes how schema is stored in a data base
   (C) hides details of data types
   (D) describes the data relationships in a data base

4. Choose the most appropriate choice with respect to conceptual design.

(A) Conceptual design is a documentation technique. Once the relation schemes are defined one can draw E-R diagrams from the relation schemes for documentation [ ]

(B) Conceptual design needs data volumn and processing frequencies to determine the size of the database

(C) Output of any conceptual design is an E-R diagram

(D) Conceptual design involves modelling the data requirements independent of the DBMS, operating system and the hardware.

5. An ER Model includes

   I. An ER diagram portraying entity types.

   II. Attributes for each entity type

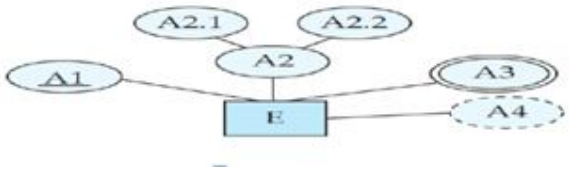
   III. Relationships among entity types.

   IV. Semantic integrity constraints that reflects the business rules about data not captured in the ER diagram. [ ]

(A) I, II, III & IV (B) I&IV

(C) I, II & IV (D) I & III

6. Refer the diagram below, where the attributes of relation E are characterized [ ]



(A) A4 is weak attribute
(B) A3 is multi valued attribute
(C) A2 is a derived attribute
(D) A3 is a foreign key attribute)

7. For a weak entity set to be meaningful, it must be associated with another entity set in combination with some of their attribute values is called as: [ ]

(A) Neighbour set (B) Strong entity set

(C) Owner entity set (D) Weak entity set

8. Which of the following statement is FALSE about weak entity set?

(A)Weak entities can be deleted automatically when their strong entity is deleted. [ ]

(B)Weak entity set avoids the data duplication and consequent possible inconsistencies caused by duplicating the key of the strong entity.

(C) A weak entity set has no primary key unless attributes of the strong entity set on which it depends are included.

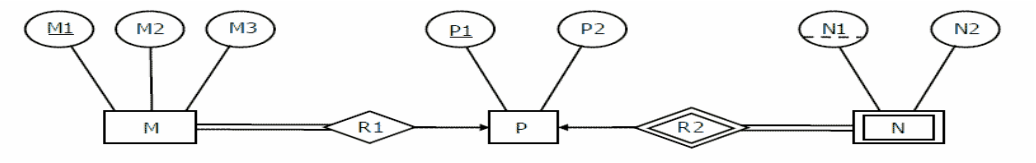
D) Tuples in a weak entity set are not partitioned according to their relationship with tuples in a strong entity set.

9. Every weak entity set can be converted into a strong entity set by:

(A) Using generalization [ ]
(B) Adding appropriate attributes
(C) Using aggregation

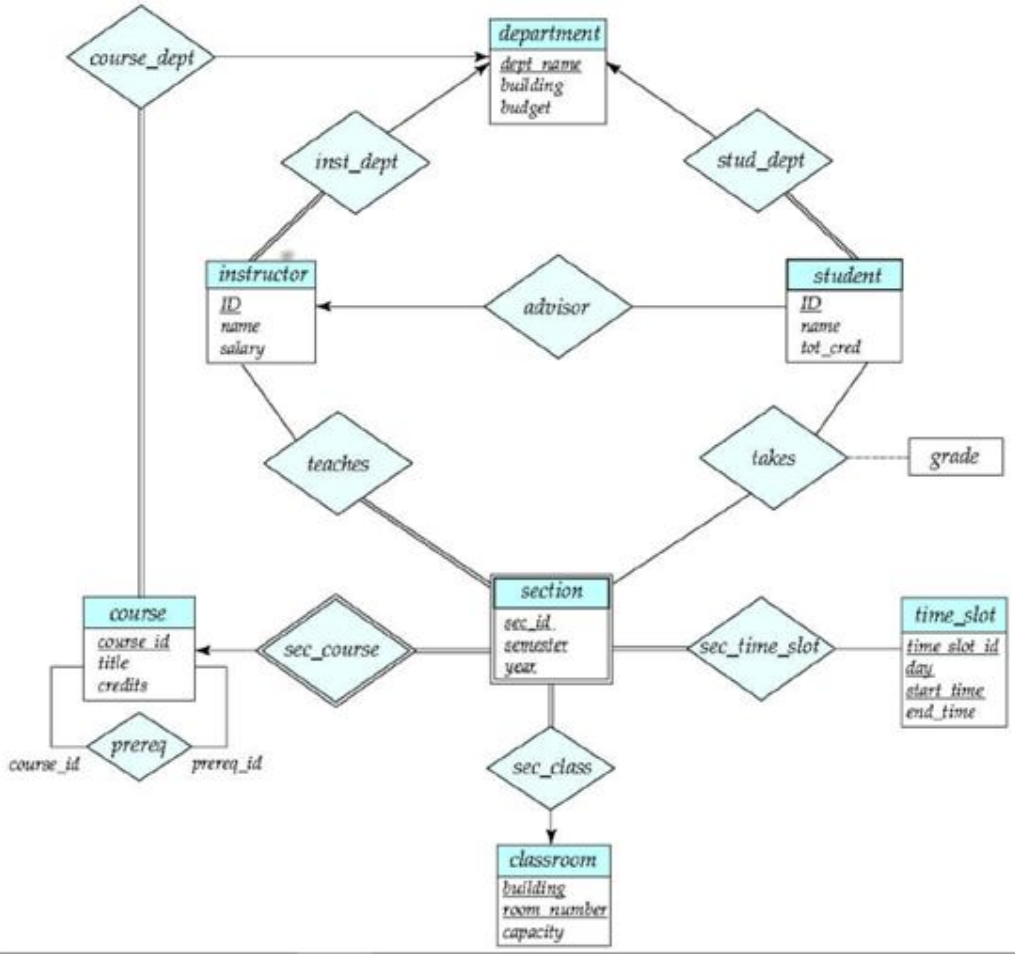
(D) None of the above

10. Consider the following ER diagram. The minimum number of tables needed to represent M, N, P, R1, R2 is [ ]



(A) 2 (B) 3 (C) 4 (D) 5

11.Consider the ER diagram given below:



Identify the correct statement(s) based on the ER model of the university. [ ]

(A) inst_dept is a relation set connects weak entity instructor with department.

(B) department is a strong entity.
(C) (time_slot_id, day) is the primary key for time_slot entity.
(D) course_id, prereq_id are the fields of prereq relation set.

12. Given the basic ER and relational models, Which of the following is INCORRECT? [ ]

(A) An attribute of an entity can have more than one value
(B) An attribute of an entity can be composite
(C) In a row of a relational table, an attribute can have more than one value
(D) In a row of a relational table, an attribute can have exactly one value or a NULL value

SECTION-B

SUBJECTIVE QUESTIONS

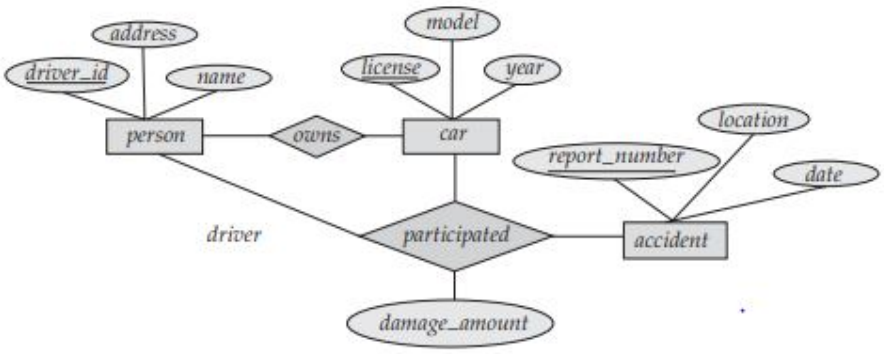
1) Define DBMS and explain why would choose a database system instead of simply storing data in operating system files?
2) What is a data model? Explain different types of data models with an example.
3) Briefly explain schema and instance with suitable example.
4) Describe purpose of ER diagrams and describe how entity, entity sets, Relationships, relationship sets are represented with an example.
5) What are the different types of attributes and keys used in ER model?
6) Quote suitable example to represent a strong and weak entity set through ER diagram.
7) Distinguish between weak entity set and strong entity set.
8) Outline the importance of EER modelling specialization and generalization with an example.
9) Illustrate with an example how to generate a relational-database schema from an ER model.
10) Illustrate with an example translation of relationship sets with participation constraints of an ER diagram to relational model.
11) Design a database for banking enterprise using ER Model.
12) Construct an ER diagram for college admission office section. The office maintains data about each class, including the instructor, the enrolment

and the time and place of the class meetings. For each student class pair a grade is recorded. Determine the entities and relationships.

13) Design a university level database for maintaining the student details of different colleges in the university. Only consider the personal details and the college and branch details of the student belong. represent the same using an ER diagram.

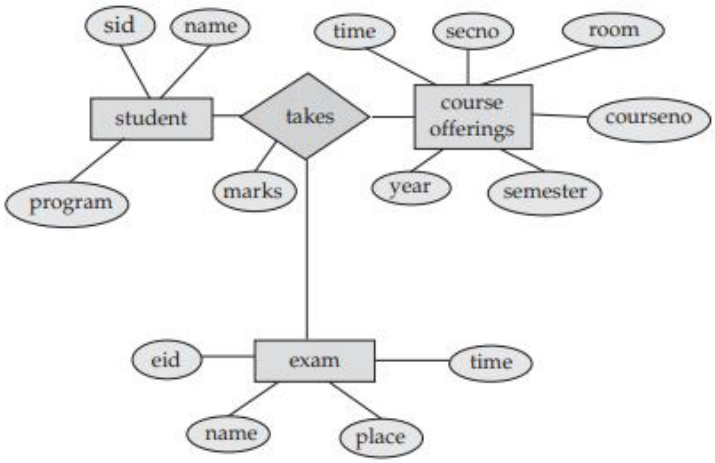
14) A company database needs to store information about employees(ssn, name, designation, salary, address, phone), departments(dno, dname, budget) and children of employees(name, age). Employee works in departments; each department is managed by an employee, a child must be identified uniquely by name when the parent(who is an employee; assume that only one parent works for the company) is known. We are not interested in information about child once the parent leaves the organization. Draw an ER diagram that captures this information.

15) Convert the following ER diagram into Relational tables.



E-R diagram for a car insurance company.

16) Convert the following ER diagram into Relational tables.

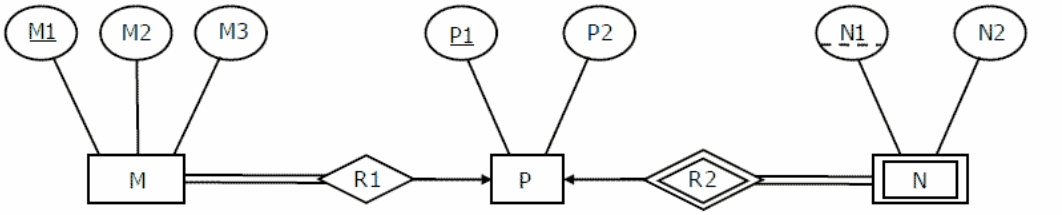


E-R diagram for marks database.

SECTION-C

QUESTIONS AT THE LEVEL OF GATE

1. Which of the following is a correct attribute set for one of the tables generated from below ER diagram. [ ]



a) {M1, M2, M3, P1}

b) {M1, P1, N1, N2}

c) {M1, P1, N1}

d) {M1, P1}

2. Let E1 and E2 be two entities in an E/R diagram with simple single-valued attributes. R1 and R2 are two relationships between E1 and E2, where R1 is one-to-many and R2 is many-to-many. R1 and R2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model? [ ]

a) 2 b) 3

b) 4 d) 5