CMR INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Approved by AICTE, Accredited by NBA and NAAC with "A++" Grade

ITPL MAIN ROAD, BROOKFIELD, BENGALURU-560037, KARNATAKA, INDIA

Department of Computer Science Engineering

Answer Scheme & Model Solution- IAT1

Sub: Database Management System		Sub Code: 18CS53	Sem/Branch: V / CSE		Sections: A,B,C
			MARKS	CO	RBT
Question	1	Define an entity and an attribute. Explain the different types of attributes that occur in an ER model, with an example of each with their corresponding ER notations	10	CO1	L2
Scheme		Define entity with example. Define attribute with example. Explain types with notation and example	2 2 6		

Solution

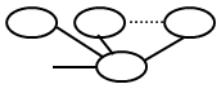
Entities and Attributes

The basic object that the ER model represents is an entity, which is a thing in the real world with an independent existence. An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course).

Attribute - the particular properties that describe an entity. For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.

Several types of attributes occur in the ER model: simple versus composite, single valued versus multi valued, and stored versus derived.

Composite versus Simple (Atomic) Attributes: Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. For example, the Address attribute of the EMPLOYEE entity can be subdivided into Street_address, City, State, and Zip,3 with the values '2311 Kirby', 'Houston', 'Texas', and '77001.' Attributes that are not divisible are called simple or atomic attributes. Composite attributes can form a hierarchy; for example, Street_address can be further subdivided into three simple component attributes: Number, Street, and Apartment_number,

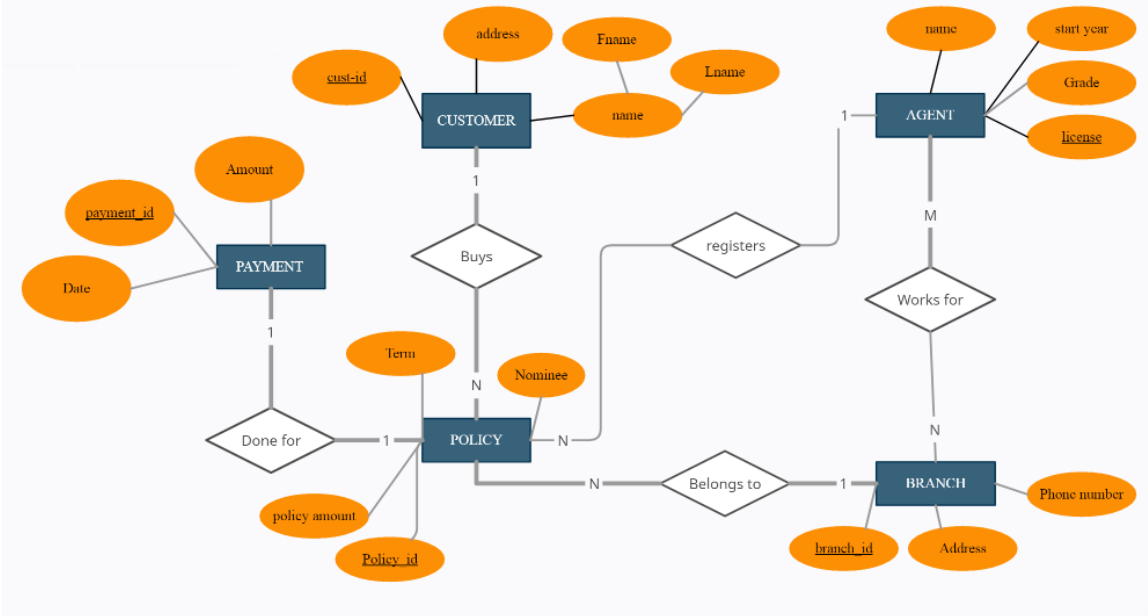
A composite attribute is represented in ER diagram as 

Single-Valued versus Multi valued Attributes: Most attributes have a single value for a particular entity; such attributes are called single-valued. For example, Age is a single-valued attribute of a person. One person may not have a college degree, another person may have one, and a third person may have two or more degrees; therefore, different people can have different numbers of values for the College_degrees attribute. Such attributes are called multi valued. A multi valued attribute may have lower and upper bounds to constrain the number of values allowed for each individual entity.

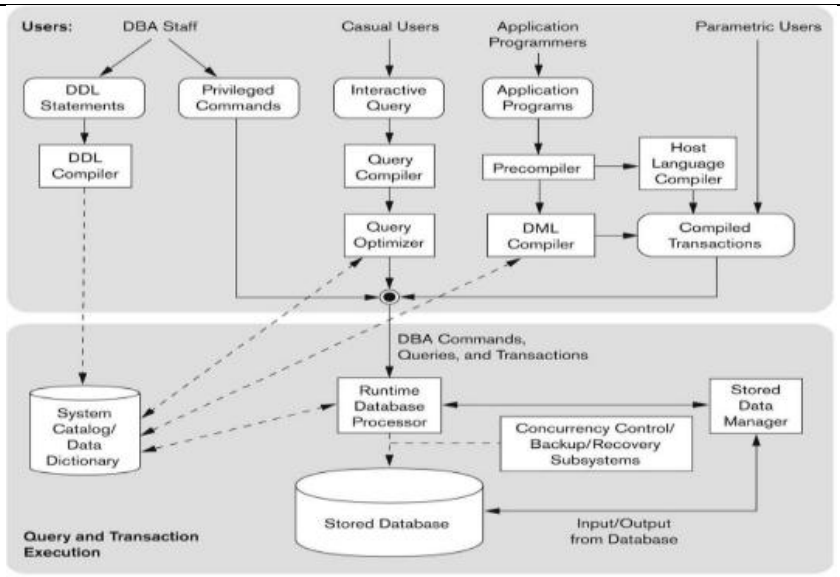
A composite attribute is represented in ER diagram as 

Stored versus Derived Attributes: In some cases, two (or more) attribute values are related, for example, the Age and Birth_date attributes of a person. For a particular person entity, the value

		of Age can be determined from the current (today's) date and the value of that person's Birth_date. The Age attribute is hence called a derived attribute and is said to be derivable from the Birth_date attribute, which is called a stored attribute. A Derived attribute is represented in ER diagram as  Simple, single and stored attributes are represented in ER diagram as 			
Question	2	Considering the following relation for a database of the company: Employee(<u>Eno,</u> Name,Salary,DoB,Dnumber) Department(<u>Dnumber</u>, Dname, MgrNo) DeptLocation(<u>Dnumber</u>, <u>Dlocation</u>) Write SQL statements for: i. Create a COMPANY database. ii. In COMPANY database create tables as given above. Ensure all the required constraints are enforced. iii. Write insert statements to insert at least two rows in a table.	10	CO3	L3
Scheme		Create database statement Create table and alter statements with proper FK declaration Insert and update statements	1 5 4		
Solution		i) Create database COMPANY; use COMPANY; ii) CREATE TABLE Employee(Eno int PRIMARY KEY, Name varchar(10), Salary int, DOB date, Dnumber int); CREATE TABLE Department(Dnumber int PRIMARY KEY, Dname varchar(10), Mgr_No int, Foreign key(Mgr_No) references Employee(Eno)); ALTER TABLE Employee ADD FOREIGN KEY(Dnumber) REFERENCES Department(Dnumber); CREATE TABLE DeptLocation(Dnumber int, Dlocation varchar(10) PRIMARY KEY(Dnumber,Dlocation), Foreign key(Dnumber) references Department(Dnumber)); Insert into Employee(1,'A',10000,'01-01-2000',null); Insert into Employee(2,'B',20000,'11-11-2000',null); Insert into Department(50,'X',1);			

		Insert into Department(51,'Y',2); Update table Employee set Dnumber=50 where Eno=1; Update table Employee set Dnumber=51 where Eno=2; Insert into DeptLocation(50,'Banaglore'); Insert into DeptLocation(50,'Mysore'); Insert into DeptLocation(51,'Banaglore');			
Question	3	Design an ER diagram for an insurance company - Assume suitable entity types like CUSTOMER, AGENT, BRANCH, POLICY, PAYMENT and relationships between them.	10	CO1	L3
Scheme		Relevant Entities Relevant attributes Relevant relationships Relevant constraints Assumptions	2 2 2 2 2		
Solution		 The following are the assumptions made in this ER diagram: • Customer name is assumed to be a composite attribute. • A Customer must buy policies. A policy must have a customer. • A policy must have a payment and a Payment must be associated with a policy. • A Policy must belong to one branch and A branch will have several policies. • An agent must work for at least one branch. A branch can have several agents but there can exist a branch without an agent associated with it. • An agent can register several policies. A policy can be registered by an agent.			
Question	4	Discuss database system environment with its component modules and their interactions with a neat diagram.	10	CO1	L2
Scheme		Diagram Explanation	3 7		

Solution



DBMS Component Modules:

- A higher-level stored data manager module of the DBMS controls access to DBMS information that is stored on disk.
- The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.
- The run-time database processor handles database accesses at run time. The query compiler handles high-level queries that are entered interactively.
- The pre-compiler extracts DML commands from an application program written in a host programming language. These commands are sent to the DML compiler for compilation into object code for database access.

Database System Utilities:

1. Loading - loads existing data files (e.g., text files or sequential files) into the database.
2. Backup - this utility provides a backup copy of the database, usually by dumping the entire database onto tape.
3. File reorganization - can be used to reorganize a database file into a different file organization to improve performance.
4. Performance monitoring - monitors database usage and provides statics to the DBA.

Question	5	Discuss how different constrains that can be enforced using SQL DDL statements with examples.	10	CO3	L2
Scheme		Not null, Default, Unique with examples Primary key with example foreign key with example Check constraint with example	4 2 2 2		

Solution

The basic constraints that can be specified when you create a table in SQL are:

- Not Null
- Unique
- Default
- Primary Key
- Foreign Key
- Check

Not Null: By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such constraint on this column specifying that NULL is now not allowed for that column. In the below example, the DBMS will not allow a row which tries to insert NULL value into the Name field.

```
CREATE TABLE EMPLOYEE
(
        SSN NUMBER(10) PRIMARY KEY,
        NAME VARCHAR(10) NOT NULL
)
```

Unique: The UNIQUE Constraint prevents two records from having identical values in a particular column. In the below example, the DBMS will not allow a row which tries to insert duplicate value into the phone number field.

```
CREATE TABLE EMPLOYEE
(
        SSN NUMBER(10) PRIMARY KEY,
        NAME VARCHAR(10) NOT NULL,
        PHONE NUMBER(10) UNIQUE
)
```

Default: The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value. In the below example, the DBMS insert 10000 as a default value into Salary field for a row if no value is specified while inserting that row.

```
CREATE TABLE EMPLOYEE
(
        SSN NUMBER(10) PRIMARY KEY,
        NAME VARCHAR(10) NOT NULL,
        PHONE NUMBER(10) UNIQUE,
        SALARY NUMBER(5) DEFAULT 10000
)
```

Primary Key: A primary key is a single field or combination of fields that uniquely identify a record. The fields that are part of the primary key cannot contain a NULL value and must be Unique. Each table should have a primary key, and each table can have only ONE primary key. In the below example, SSN is a primary key, hence it must be unique and not take null value and the table Employee cannot have any other attribute as primary key as it already has SSN has PK.

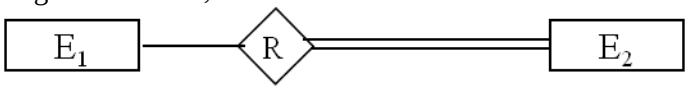
```
CREATE TABLE EMPLOYEE
(
        SSN NUMBER(10) PRIMARY KEY,
        NAME VARCHAR(10) NOT NULL,
        PHONE NUMBER(10) UNIQUE,
        SALARY NUMBER(5) DEFAULT 10000
)
```

Foreign Key: A foreign key is a key used to link two tables together. Foreign Key is a column or a combination of columns whose values match a Primary Key of another table (parent table/referred table). The relationship between 2 rows of two tables matches the Primary Key in one of the parent table with a Foreign Key value of the referenced table. In the below table, DEPTNO is a foreign key referencing to DNO of Department table. The DEPTNO attribute can take value which is present in DNO of department table or NULL value.

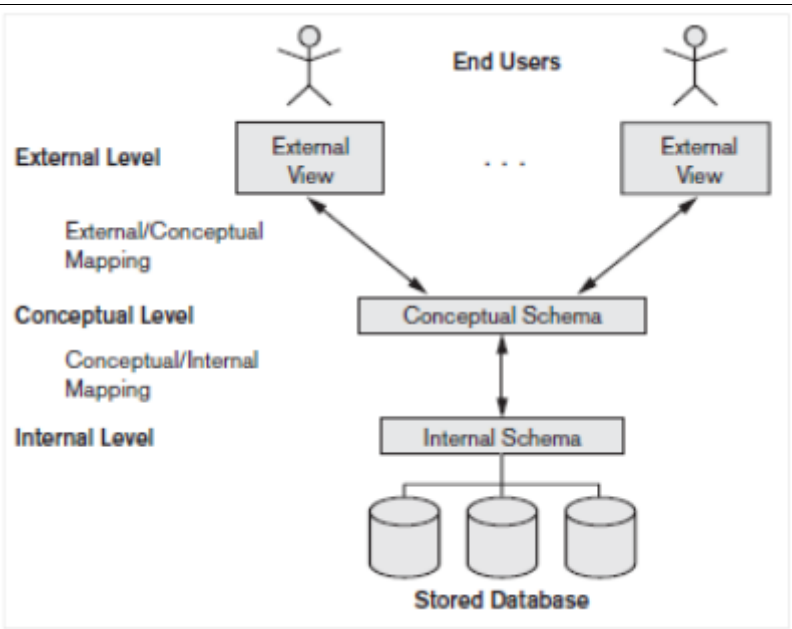
```
CREATE TABLE EMPLOYEE
(
        SSN NUMBER(10) PRIMARY KEY,
        NAME VARCHAR(10) NOT NULL,
        PHONE NUMBER(10) UNIQUE,
        SALARY NUMBER(5) DEFAULT 10000
        DEPTNO NUMBER(2) REFERENCES DEPARTMENT(DNO)
)
```

Check Constraint: Check constraint is used to enforce a condition on an attribute in a table. A record is inserted only if the check constraint is satisfied. If the constraint is not satisfied, the record insertion will be rejected.

```
 CREATE TABLE EMPLOYEE
```

		(SSN NUMBER(10) PRIMARY KEY, NAME VARCHAR(10) NOT NULL, PHONE NUMBER(10) UNIQUE, AGE NUMBER(2) CHECK AGE>16, SALARY NUMBER(5) DEFAULT 10000, DEPTNO NUMBER(2) REFERENCES DEPARTMENT(DNO))			
Question	6a	Explain structural constraints of a relationship-type with examples	5	CO1	L2
Scheme		Cardinality ratio with example Participation with example	2.5 2.5		
Solution		Cardinality ratio and participation constraints, taken together are the structural constraints of a relationship type. The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in. For example, consider a binary relationship type WORKS_FOR between Department and Employee entity types, DEPARTMENT**:**EMPLOYEE is of cardinality ratio 1:N, meaning that each department can be related to numerous employees, but an employee can be related to (work for) only one department. The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N. Cardinality ratio 1:N of a relationship R between E1 & E2 entities is given as below in ER diagrams.  The binary relationship MANAGES which relates a department entity to the employee who manages that department; the cardinality ratio is 1:1. This represents the constraint that an employee can manage only one department and that a department has only one manager. The relationship type WORKS_ON between Employee entity and the Project entity that he works for, is of cardinality ratio M:N, representing that an employee can work on several projects and a project can have several employees. The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. There are two types of participation constraints—total and partial. If a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in a WORKS_FOR relationship instance. Thus, the participation of EMPLOYEE in WORKS_FOR is called total participation, meaning that every entity in "the total set" of employee entities must be related to a department entity via WORKS_FOR. Total participation is also called existence dependency. On the other hand, we do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is partial, meaning that some or "part of the set of" employee entities are related to a department entity via MANAGES, but not necessarily all. In a relationship R, where participation of entity E1 is partial & participation of E2 is total, is represented in ER diagram as below,  			
Question	6b	Explain three schema architecture with a neat diagram.	10	CO1	L2
Scheme		Diagram Explanation	2 3		

Solution



The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system. The goal of the three-schema architecture, is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

a) Internal Level: The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

b) Conceptual level: The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

c) External or view level: The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

Mapping: The three schemas are only descriptions of data; the stored data that actually exists is at the physical level only. In a DBMS based on the three-schema architecture, each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called mappings.