I - ASSIGNMENT

(Start Writing From Here)

ANSWER ALL THE QUESTIONS

1. Define the following terms with examples :-
a) Entities b) Generalization c) Specialization d) Aggregation.

Ans: **Entities :**

An entity is an object in the real world that is distinguish-able from other objects.
For example, a Car is an entity.
Other examples include the following : the Green Dragonzod toy, the toy department, the manager of the toy department, the home address of the manager of the toy department.

It is often useful to identify a collection of similar entities, such a collection is called "entity set".

Note that entity sets need not be disjoint: the collection of toy department employees and the collection of appliance department employees may both contain employee John Doe (who happens to work in both departments).

We could also define an entity set called Employees that contains both the toy and appliance department employee sets.

An Entity Type in DBMS is defined by the Name of the Entity. An Entity can be a Tangible Entity (real-world object).

For Example, In the customer example of Entity in Database managed, attributes might include name, address and phone Number.

Generalization :

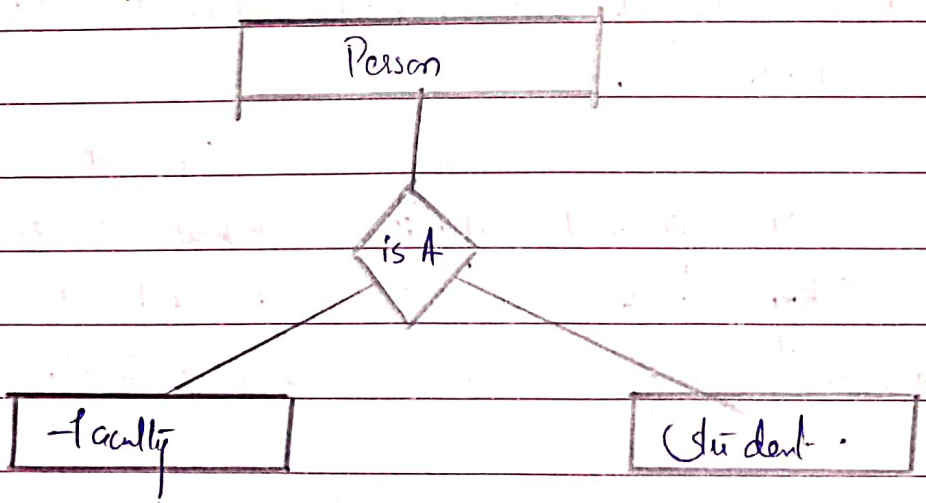
- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.

- In Generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.

- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.

- In Generalization, entities are combined to form a more generalized entity i.e., subclasses are combined to make a superclass.

For Example, faculty and student entities can be generalized and create a higher level entity person.



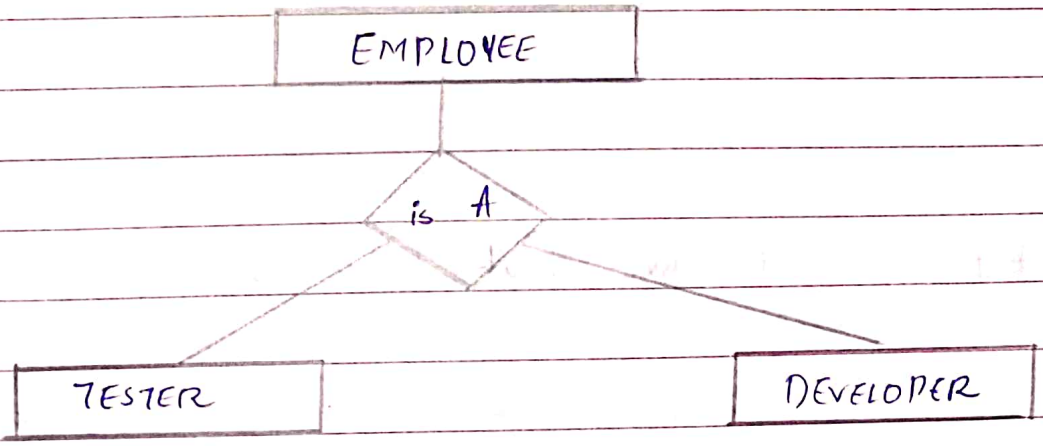
Specialization :-

- Specialization is a top-down approach, and it is opposite to the Generalization. In Specialization, one higher level entity can be broken down into two lower level Entities.

- Specialization is used to Identify the Subset of an Entity set that shares some distinguishing characters.

- Normally, the Superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

for Example : In an Employee Management System, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the Company.

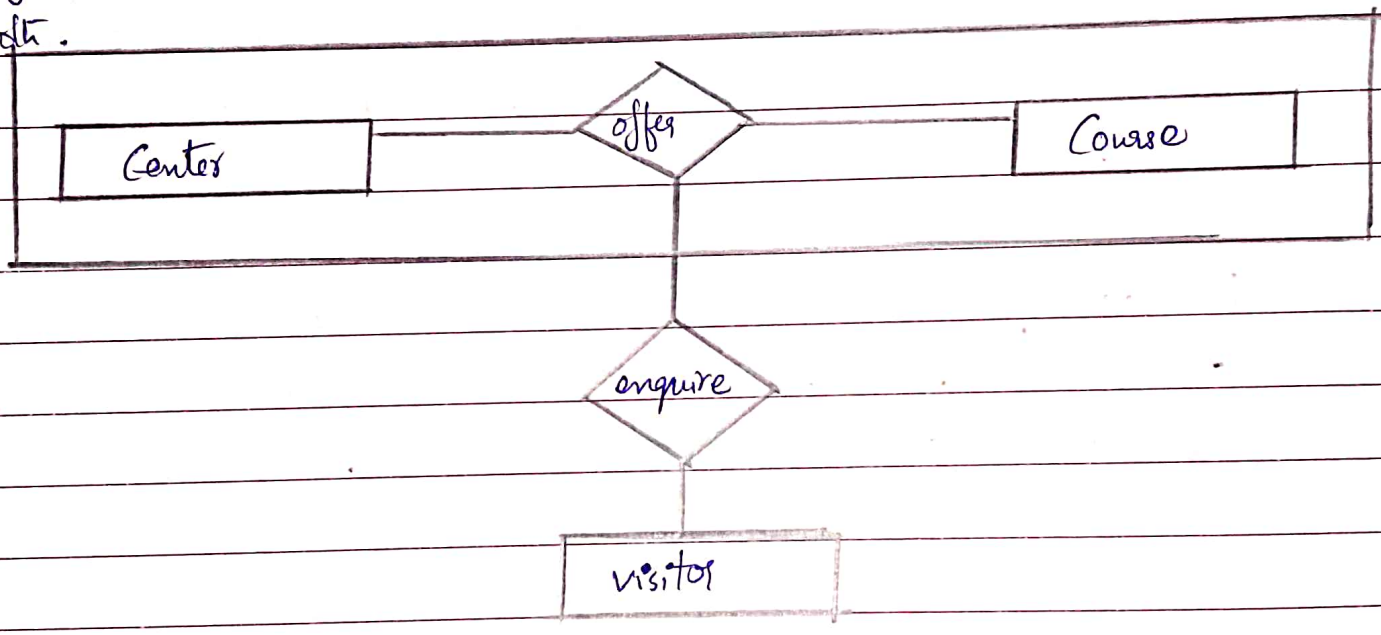


Aggregation :

In aggregation, the relation between two Entities is treated as a

single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

For Example: Center Entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center Instead he will ask the enquiry about both.



2. Explain about Tuple Relational Calculus in detail?

Ans There is an alternate way of formulating queries known as Relational Calculus. Relational Calculus is a Non-procedural query language. In the Non-procedural query language, the user is concerned with the details of how to obtain the end results. The Relational Calculus tells

what to do but new explains how to do. Most commercial relational languages are based on aspects of Relational Calculus including SQL-QBE and QUEL.

Relational Calculus is based on Predicate Calculus, a name derived from branch of Symbolic Language. A Predicate is a truth valued function with arguments. On substituting values for the arguments, the function result in an expression called a proposition. It can be either true or false. It is a tailored version of a subset of the Predicate Calculus to communicate with the relational database.

Types of Relational Calculus :- i) Tuple Relational Model (TRC)
ii) Domain Relational Model (DRC).

1. Tuple Relational Calculus (TRC)

It is a non-procedural Query language which is based on finding a number of tuple variables also known as range variable for which predicate holds true. It describes the detailed information without giving a specific procedure for obtaining that information. The tuple relational Calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation. The result of the relation can have one or more tuples.

Notation : A Query in the tuple relational Calculus is expressed as

$\{T \mid P(T)\}$ or $\{T \mid Condition(T)\}$

Where, "T" is the resulting tuples.

"P(T)" is the condition used to fetch T.

For Example,

$\{T.name \mid Author(T) \text{ AND } T.article = 'database'\}$

OUTPUT: This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (Tuple relation Calculus) can be Quantified. In TRC we can use Existential ($\exists$) and Universal Quantifiers ($\forall$).

Tuple Relational Calculus (TRC) is a formal language used to specify queries on a relational database. It is a non-procedural language that operates on a set of tuples (rows) in one or more relations (tables) and returns a set of tuples that satisfy a given condition. TRC expresses queries in terms of logical formulas, using Variables to refer to tuples and conditions to restrict the result set. The Variables can be constrained using selection and projection operators and the conditions can be combined using logical operators such as AND, OR, and NOT.

- A Tuple relational Calculus is a Non-procedural query language

that specifies to select the tuples in a Relation.

- It can select the tuples with a range of values or tuples for certain attribute values, etc.
- The resulting relation can have one or more tuples.
- It uses first-order logic to generate the Result.
- All the conditions used in the tuple expression are called a well-formed formula (WFF).
- All the conditions in the Expression are combined by using Logical Operators like AND, OR and NOT and quantifiers like 'FOR ALL' (∀) and or 'there exists' (∃).
- If the tuple variables are all Bound variables in a WFF is called a closed WFF.
- In an Open WFF, we will have atleast one free variable.

Tuple Relational Calculus Examples:

Tuple Relational Calculus (TRC) is a query language that is used to retrieve data from a relational database by specifying conditions that the desired tuples must satisfy. Here are a few examples of TRC:

- Retrieve all the employees who work in the "Sales" Department :-

$\{t \mid Employee(t) \wedge t.department = "Sales"\}$

- Retrieve all the names of the Employees who earn more than $50,000 :

$\{t.name \mid Employee(t) \wedge t.salary > 50000\}$

- Retrieve the names of all the Employees who work in the "Sales" department and earn more than $50,000 :

$\{t.name \mid Employee(t) \wedge t.department = "Sales" \wedge t.salary > 50000\}$

3. Explain Unique Key, Not Null, Check constraints with an Example?

Ans

Unique key in DBMS :

The word "unique" defines a thing which is unique from other things. A Unique Key in DBMS is a key that is able to identify all the records of a table uniquely.

Here, i

What is a Unique Key :

A Unique Key in DBMS is used to uniquely identify a tuple in a table and is used to prevent duplicity of the values in a table.

Role of Unique key :-

A Unique key is used to remove the duplicity of values in a Table. However, the usage of a primary key is the same, but there is a difference between both keys. A Primary key cannot take a NULL value, but a unique key can have one NULL value as its value.

NOTE: The SQL standards believe that the unique key does not satisfy or guarantees the uniqueness of those rows having NULL as its value. However, other RDBMS does not follow the SQL standards.

Implementing Unique key: Below Syntax shows the implementation of the unique key on "CREATE" Table -

```
>Create TABLE student (Sid int NOTNULL, Sname varchar(100)
  NOT NULL, Course varchar (120), Age int, UNIQUE (Sid));
```

NOT NULL Constraints:

NOT NULL constraints prevent null values from being entered into a column. The Null value is used in databases to represent an unknown state. By default, all of the built-in-datatypes provided with the database manager support the presence of null values. However, some Business rules might dictate that a value must always be provided (for example, every employee is required to provide emergency contact information). The NOT NULL constraint

used to ensure that a given column of a table is never assigned the null value. Once a NOT NULL constraint has been defined for a particular column, any insert or update operation that attempts to place a Null value in that column will fail.

Because constraints only apply to a particular table, they are usually defined along with a table's attributes, during the table creation process.

The NOT NULL Constraint enforces a column to NOT accept NULL values

This Enforces a field to always contain a Value, which means that you cannot insert a new record, or update a record without adding a Value to this field.

for Example: The following SQL ensures that the "ID", "LastName" and "FirstName" Columns will NOT accept NULL values when the "Persons" table is created:

```
SQL> CREATE TABLE persons (ID int NOTNULL, LastName varchar(255)
     NOT NULL, FirstName varchar(255) NOT NULL, Age int);
```

SQL NOT NULL on ALTER TABLE

To create a NOTNULL Constraint on the "Age" Column when the "Person" table is already created, use:- ALTER TABLE persons

MODIFY COLUMN Age int NOT NULL;

SQL CHECK CONSTRAINT : The check constraint is used to limit the value range that can be placed in a column.
If you define a "check" constraint on a column it will allow only certain values for this column.
If you define a "check" constraint on a table it can limit the values in certain based on values in other columns in the row.

SQL CHECK on CREATE TABLE : The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The check constraint ensures that the age of a person must be 18 or older.

```
SQL > CREATE TABLE Persons ( ID int NOTNULL, LastName Varchar
      (255) NOT NULL, firstName varchar (255), Age int,
      CHECK (Age >= 18 ));
```

To allow naming of a check constraint, and for defining a check constraint on multiple columns, use the syntax :-

```
SQL > CREATE TABLE Persons ( ID int NOT NULL, LastName varchar(255
      NOT NULL, firstName varchar(255), Age int, city varchar(255
      CONSTRAINT CHK_PERSON CHECK (Age >=18 and City = 'Sandnes
```

SQL CHECK on ALTER TABLE : To create a check constraint on the "Age" column when the table is already created, use the SQL

```
SQL > ALTER TABLE Persons
      ADD CHECK (Age >=18);
```

To allow naming of a Check constraint, and for defining a check constraint on multiple columns, use the following SQL Syntax -

```
SQL > ALTER TABLE Persons
      ADD CONSTRAINT CHK_PersonAge
      CHECK (Age >=18 AND City = 'Sandnes');
```

DROP a CHECK Constraint : To drop a Check Constraint, use the following SQL :-

```
SQL > ALTER TABLE Persons
      DROP CONSTRAINT CHK_PersonAge;
```

(or)

```
SQL > ALTER TABLE Persons
      DROP CHECK CHK_PersonAge;
```