IA – 2

18CS53 - DATABASE MANAGEMENT SYSTEM

1.a Discuss Equijoin and Natural Join with suitable examples using relational algebra notations.

A special case of join operation R join S is an equijoin in which equalities arc specified on all fields having the same name in R and S. In this case, we can simply omit the join condition; the default is that the join condition is a collection of equalities on all common fields. We call this special case a natural join, and it has the nice property that the result is guaranteed not to have two fields with the same name.

The equijoin expression S.USN=A.USN is actually a natural join and can simply be denoted as USN, since the only common field is USN. If the two relations have no attributes in common, it is simply the cross-product.

USN	Name	Phone	pincode
100	Abc	123	543567
200	def	543	345345

USN	L_card_no	Dept	Addr
100	32e3	cse	54t
200	34e4	Cse	45h5

The natural join operation operates on two relations and produces a relation as the result.

Concatenates each tuple of the first relation with every tuple of the second, natural join considers only those pairs of tuples with the same value on those attributes that appear in the schemas of both relations.

Consider instructor and *teaches are attributes in a table*, and computing instructor natural join teach considers only those pairs of tuples where both the tuple from instructor and the tuple from teaches have the same value on the common attribute, ID.

1.b Define the following terms with example

Key: A single or set of attributes used to identify a record/tuple from the table.

Super Key: One or more attributes used to identify the record from record set is called super key.
Candidate Key: A Minimal sub-set of super key is called candidate key
Primary Key: The Database Admin can assign any one of the candidate key as ‘Primary key’.
Foreign key: The value in one relation must appear in another relation.

2. a Consider the schema

Sailors(sid, sname, rating, age),
Boats(bid, bname, color),

Reserves(sid, bid, day)
Write relational algebraic queries for the following

i) Find names sailors who have reserved boat number 103.

Π sname((σ$_{bid=103}$ Reserves) ⋈ Sailors

Find names sailors who have reserved a red boat.

ii) **Π$_{sname}$ ((σ$_{color}$= 'red' Boats) ⋈ Reserves ⋈ Sailors**

iii) **Find names of sailors who have reserved a red or green boat.**

p(Tempboats, (σ$_{color}$ ='red' Boats) U (σ$_{color}$='green' Boats))
Πsname(Tempboats ⋈ ReseTves ⋈ Sailors)

Or

p(Tempred, **Π$_{sid}$** ((σ$_{color}$= 'red' Boats) ⋈ Reserves))
p(Tempgreen, **Π$_{sid}$**((σ$_{color}$='green' Boats) ⋈ Reserves))
Πsname((Tempred U Tempgreen) ⋈ Sailors)

iv) **Find names of sailors who have served all boats.**

p(Tempsids, (Π$_{sid,bid}$ Reserves) / (Π$_{bidBoats}$))
Π$_{sname}$(Tempsids ⋈ Sailors)

2.b Explain the data types available for attribute specification in SQL.

Numeric data types include integer numbers of various sizes (INTEGER or INT, and SMALLINT) and floating-point (real) numbers of various precision (FLOAT or REAL, and DOUBLE PRECISION).

Character-string data types are either fixed length—CHAR(n) or CHARACTER(n), where n is the number of characters—or varying length— VARCHAR(n) or CHAR VARYING(n) or CHARACTER VARYING(n), where n is the maximum number of characters.

Bit-string data types are either of fixed length n—BIT(n)—or varying length— BIT VARYING(n), where n is the maximum number of bits. The default for n, the length of a character string or bit string, is 1.

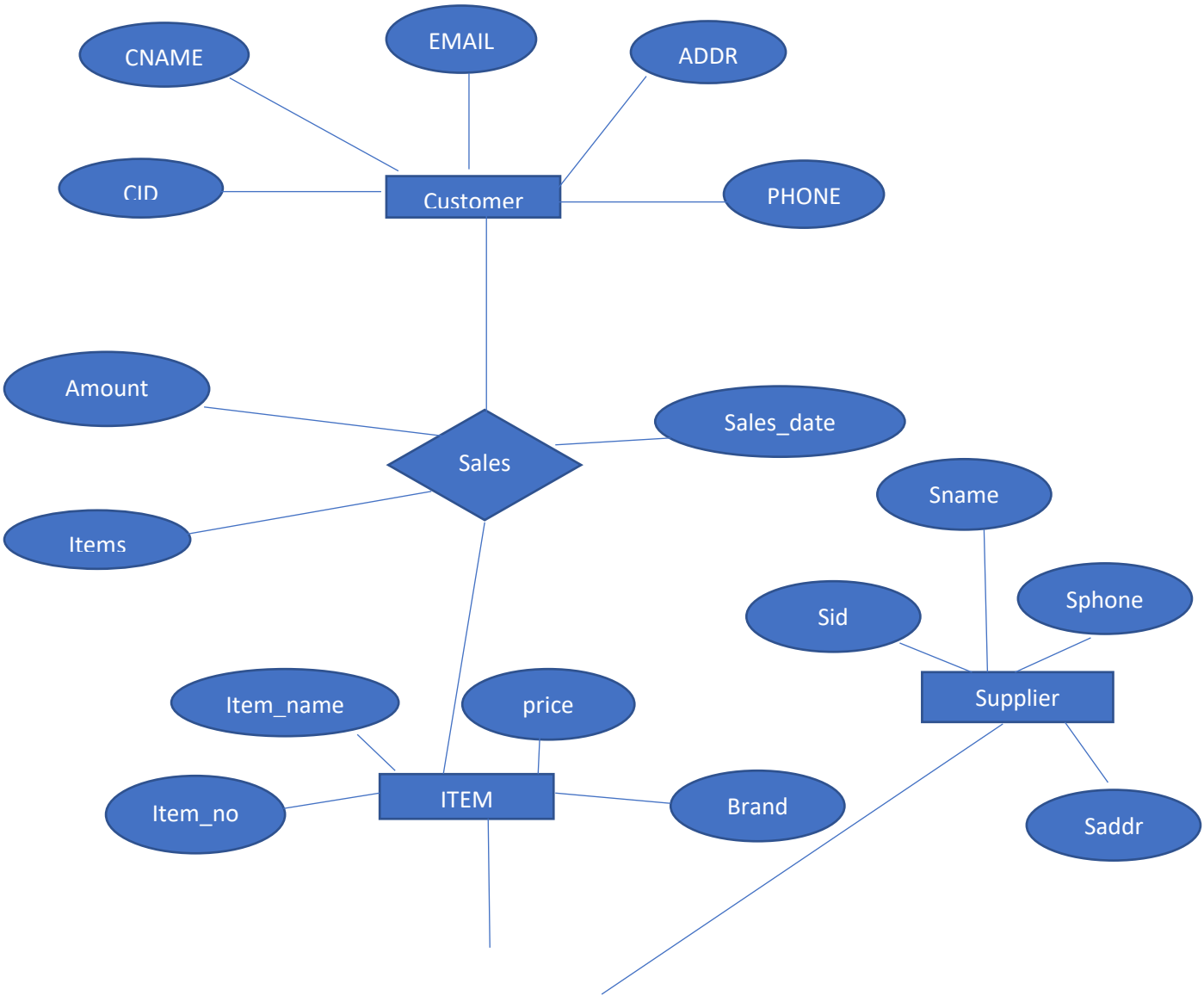
A Boolean data type has the traditional values of TRUE or FALSE.

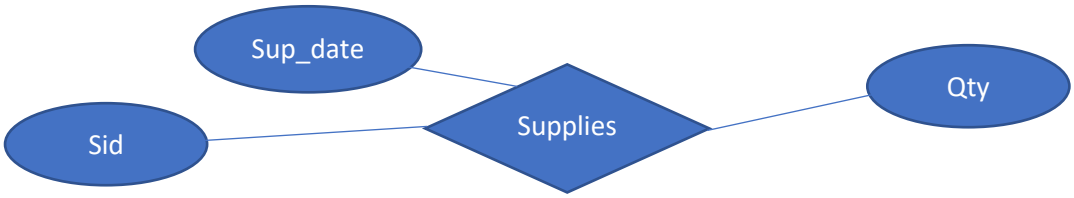
The DATE data type has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD. The TIME data type has at least eight positions, with the components HOUR, MINUTE, and SECOND in the form HH:MM:SS. Only valid dates and times should be allowed by the SQL implementation.

A timestamp data type (TIMESTAMP) includes the DATE and TIME fields, plus a minimum of six positions for decimal fractions of seconds and an optional WITH TIME ZONE qualifier.

3.

Customer(cid, cname, email, addr,phone)
Item(item_no, item_name, price, brand)
Sales(cid, item_no, items, amount, sale_date)
Supplier(sid, sname, sphone, saddr)
Supply(sid, item_no, supply_date, qty)





Customer(cid, cname, email, addr,phone)
Item(item_no, item_name, price, brand)
Sales(cid, item_no, items, amount, sale_date)
Supplier(sid, sname, sphone, saddr)
Supply(sid, item_no, supply_date, qty)

i) List the items purchased by the customer 'Prsanth';

select C.cid, C.cname from Customer C, Item I, Sales S
where C.cid=S.cid and S.item_no=I.item_no;

ii) Retrieve items supplied by all suppliers starting from 1st Jan 2019 to 30th Jan 2019.

select item_no from Supply S, Supplier SU
where SU.sid=S.sid and S.supply_date between 1st Jan 2019 and 30th Jan 2019.

iii) Get the details of customers whose total purchase of items worth more than 5000 rupees.

Select cid, cname from Customer C, Sales S
where C.cid=S.cid and S.amount>5000;

iv) List total sales amount, total items, and average sell amount of all item.

select sum(amount), count(items), avg(amount) from sales;

v) Display customers who have not purchase any items.

select cid, cname from Customer where cid not in
(select cid from Sales)

CUSTOMER

CID	CNAME	EMAIL	ADDR	PHONE

ITEM

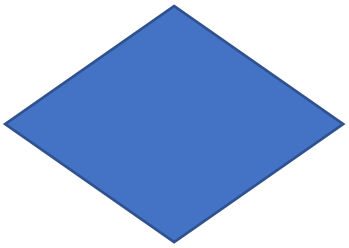
ITEM_NO	ITEM_NAME	PRICE	BRAND

SALES

CID	ITEM_NO	ITEMS	SALE_DATE

SUPPLIER

ITEM_NO	ITEM_NAME	PRICE	BRAND



4. a. Constraints include key and referential integrity constraints, restrictions on attribute domains and NULLs, and constraints on individual tuples within a relation using the CHECK clause.

NULLs as attribute values, a constraint NOT NULL may be specified if NULL is not permitted for a particular attribute.

It is also possible to define a default value for an attribute by appending the clause DEFAULT to an attribute definition. The default value is included in any new tuple if an explicit value is not provided for that attribute.

Another type of constraint can restrict attribute or domain values using the CHECK clause.

Eg., Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);

The CHECK clause can also be used in conjunction with the CREATE DOMAIN statement. For example, we can write the following statement:

CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM > 0 AND D_NUM < 21);

The PRIMARY KEY clause specifies one or more attributes that make up the primary key of a relation

Dnumber INT PRIMARY KEY

The UNIQUE clause can also be specified directly for a unique key if it is a single attribute, as in the following example: Dname VARCHAR(15) UNIQUE.

4. b. Some queries require that existing values in the database be fetched and then used in a comparison condition. Such queries can be conveniently formulated by using nested queries, which are complete select-from-where blocks within another SQL query. That other query is called the outer query.

These nested queries can also appear in the WHERE clause or the FROM clause or the SELECT clause or other SQL clauses as needed.

```
SELECT DISTINCT Pnumber
FROM PROJECT
WHERE Pnumber IN
        ( SELECT Pnumber
        FROM PROJECT, DEPARTMENT, EMPLOYEE
        WHERE Dnum = Dnumber AND Mgr_ssn = Ssn AND Lname = 'Smith' )
        OR
         Pnumber IN
        ( SELECT Pno
        FROM WORKS_ON, EMPLOYEE
        WHERE Essn = Ssn AND Lname = 'Smith' );
```

If a nested query returns a single attribute and a single tuple, the query result will be a single (scalar) value. In such cases, it is permissible to use = instead of IN for the comparison operator. In general, the nested query will return a table (relation), which is a set or multiset of tuples.

Aggregate Function:

Aggregate functions are used to summarize information from multiple tuples into a single-tuple summary. Grouping is used to create subgroups of tuples before summarization. Grouping and aggregation are required in many database applications, and we will introduce their use in SQL through examples.

There are five aggregate functions

1. Sum
2. Count
3. Avg
4. Max
5. Min

SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary) FROM EMPLOYEE;

Group By

In many cases we want to apply the aggregate functions to subgroups of tuples in a relation, where the subgroups are based on some attribute values.

SELECT Dno, COUNT (*), AVG (Salary) FROM EMPLOYEE GROUP BY Dno;

For each project, retrieve the project number, the project name, and the number of employees who work on that project.

SELECT Pnumber, Pname, COUNT (*) FROM PROJECT, WORKS_ON WHERE Pnumber = Pno GROUP BY Pnumber, Pname;

Sometimes we want to retrieve the values of these functions only for groups that satisfy certain conditions. Here, we can use Having Clause.

HAVING provides a condition on the summary information regarding the group of tuples associated with each value of the grouping attributes. Only the groups that satisfy the condition are retrieved in the result of the query.

SELECT Pnumber, Pname, COUNT (*) FROM PROJECT, WORKS_ON WHERE Pnumber = Pno GROUP BY Pnumber, Pname HAVING COUNT (*) > 2;

5.a The entity integrity constraint states that no primary key value can be NULL. This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that we cannot identify some tuples.

Key constraints and entity integrity constraints are specified on individual relations. The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

To define referential integrity more formally, first we define the concept of a foreign key. The conditions for a foreign key, given below, specify a referential integrity constraint between the two relation schemas R1 and R2. A set of attributes FK in relation schema R1 is a foreign key of R1 that references relation R2 if it satisfies the following rules:

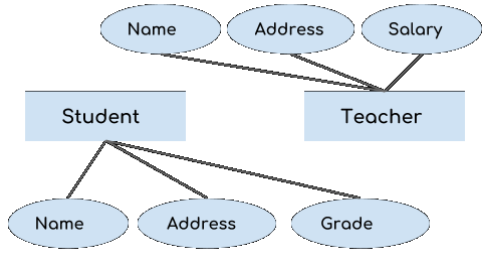
1. The attributes in FK have the same domain(s) as the primary key attributes PK of R2; the attributes FK are said to reference or refer to the relation R2.

2. A value of FK in a tuple t1 of the current state r1(R1) either occurs as a value of PK for some tuple t2 in the current state r2(R2) or is NULL. In the former case, we have t1[FK] = t2[PK], and we say that the tuple t1 references or refers to the tuple t2.

In this definition, R1 is called the referencing relation and R2 is the referenced relation. If these two conditions hold, a referential integrity constraint from R1 to R2 is said to hold. In a database of many relations, there are usually many referential integrity constraints.

5. b

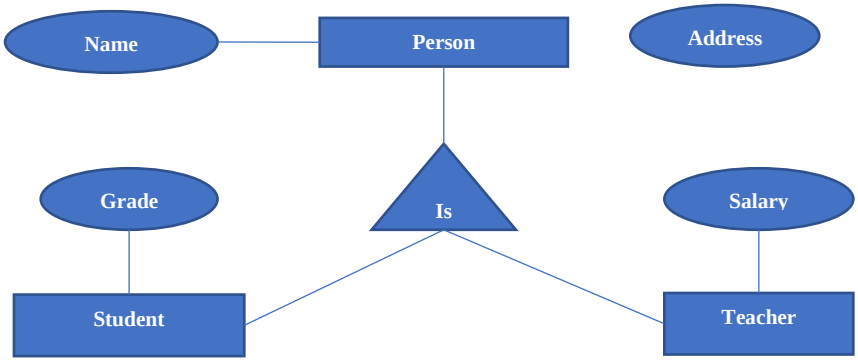
Generalization is a process in which the common attributes of more than one entities form a new entity. This newly formed entity is called generalized entity.



Lets say we have two entities Student and Teacher. Attributes of Entity Student are: Name, Address & Grade Attributes of Entity Teacher are: Name, Address & Salary

These two entities have two common attributes: Name and Address, we can make a generalized entity with these common attributes. Lets have a look at the ER model after generalization.

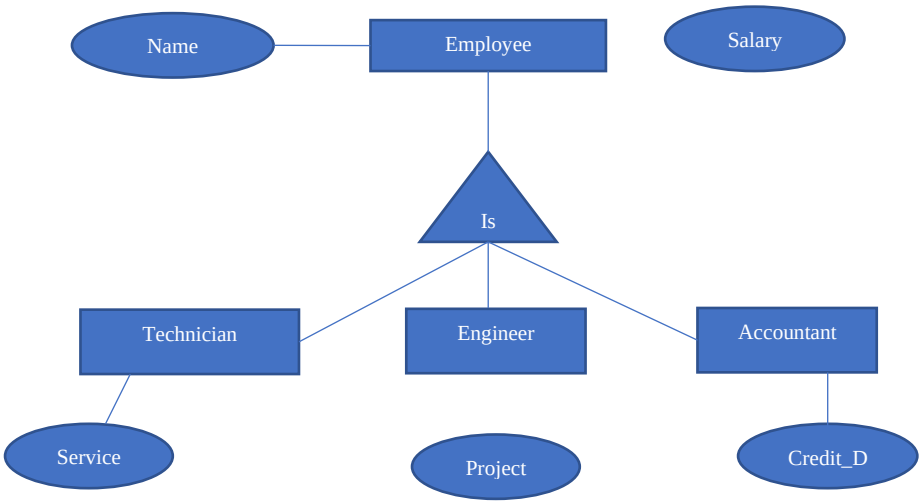
We have created a new generalized entity **Person** and this entity has the common attributes of both the entities. As you can see in the given ER Diagram that after the generalization process the entities Student and Teacher only has the specialized attributes Grade and Salary respectively and their common attributes (Name & Address) are now associated with a new entity Person which is in the relationship with both the entities (Student & Teacher)



1. Generalization uses bottom-up approach where two or more lower-level entities combine together to form a higher-level new entity.

2. The new generalized entity can further combine together with lower-level entity to create a further higher-level generalized entity.

Specialization is a process in which an entity is divided into sub-entities. You can think of it as a reverse process of generalization, in generalization two entities combine together to form a new higher-level entity. Specialization is a top-down process.



Specialization is to find the subsets of entities that have few distinguish attributes. For example – Consider an entity employee which can be further classified as sub-entities Technician, Engineer & Accountant because these sub entities have some distinguish attributes.

6. a

Trigger is invoked by SQL engine automatically whenever a specified event occurs.

Trigger is stored into database and invoked repeatedly, when specific condition match.

Triggers are stored programs, which are automatically executed or fired when some event occurs.

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
```

```
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

A typical trigger has **three components:**
Event: *When this event happens, the trigger is activated*
Condition (optional): *If the condition is true, the trigger executes, otherwise skipped*
Action: *The actions performed by the trigger*
The **action is to be executed automatically if the condition is satisfied when event occurs.**

Here,
CREATE [OR REPLACE] TRIGGER trigger_name: It creates or replaces an existing trigger with the trigger_name.
{BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The **INSTEAD OF** clause is used for creating trigger on a view.
{INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
[OF col_name]: This specifies the column name that would be updated.
[ON table_name]: This specifies the name of the table associated with the trigger.

Triggers are written to be executed in response to any of the following events.

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).
- Triggers could be defined on the table, view, schema, or database with which the event is associated.

Advantages of Trigger

- Trigger generates some derived column values automatically
- Enforces referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Eg.,

```
create or replace trigger std11
before insert on student
for each row
begin
   :new.total:=:new.m1+:new.m2;
 :new.avg:= :new.total/2;
 if :new.avg<30 then
  :new.grade:='fail';
 elsif :new.avg>30 and :new.avg<60 then
  :new.grade:='pass';
```

```
  elsif :new.avg>60 and :new.avg<90 then
   :new.grade:='first';
  else
   :new.grade:='super';
end if;
end;
/
```

6. b.

- A view is a single table that is derived from one or more base tables or other views
- Views neither exist physically nor contain data itself, it depends on the base tables for its existence
- 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.

The command to specify view is **CREATE VIEW.**

The view is given a **virtual table name or view name**, a list of attribute names, and a query to specify the contents of the view.

Syntax

CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition

Example:

CREATE VIEW WORKS_ON1
AS SELECT Fname, Lname, Pname, Hours
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE Ssn=Essn AND Pno=Pnumber ;

Advantages of view:

To simply the specification of certain queries.
A view always shows **up-to-date**
If we **modify the tuples in the base tables on which the view is defined, the view must automatically reflect these changes**

If we do not need a view any more, we can use the DROP VIEW command

•DROP VIEW WORKS_ON1;

View Update

- **Immediate update:** Update the view as soon as the base table are changed
- **Lazy view:** update the view when needed by a view query.
- **Periodic Update:** Update the view periodically.

- Updating of views is complicated and can be ambiguous
- An update on view defined on a single table without any aggregate functions can be mapped to an update on the underlying base table under certain conditions.

- View involving joins, an update operation may be mapped to update operations on the underlying base relations in multiple ways.