Date / /
Page No.

Unit-1

- Types of database
- kinds of data stored
1) Traditional dbs → textual / numeric
2) NO SQL dbs → unstructured database (Social)
3) Multi media dbs → (in structured) audio, video clips etc
4) Geographical information System (GIS) → Maps, satellite images, weather data etc
5) Data warehouse & OLAP System → data is extracted, helping in decision making
6) real time database → industrial & manufacturing process
7) Collection of related data is called a data base → Something that can be recorded & that can have implicit meaning

- Implicit Properties →
- Represent some aspect of the real world - UOD / mini-world.
- A database is a logically coherent collection of data with some inherent meaning
- A data base designed, built & populated with data for a specific purpose, it has an interactive, in which there users are interested.
- Users can't be direct interact with D.B

* database defination → general purpose software that enables user to create & maintain the db.

* function of DBMS:

DBMS facilates the process of defining Constructing, manupulating & sharing database among various users & applications

i) defining the db
data items
identifying data types
Constraints)
relationship among data.

ii) Constructing
Creating and storing the db on a physical storage device.

iii) Manupulating
Manupulating data base includes functions such as quering db to retrive specific data.

(iv) Sharing → sharing db allows multiple user

Date / /
Page No.

& programmer to access the db simultaniously

(i) Protection
(:) System protection
Hardware & software
Created

Security
:: user protection
- unauthorized
access.

(ii) Maintaining the db
- lifetime of a dp spans several year.
Since its lifetime is long we should maintain
it by regularly updating it.

- file processing Approach
-> for every user have copy of data type.
- drawbacks -> data redancy & maintaining
multiple copies of single data.

- characteristics of a db Approach
(i) self describing nature of the db system.
db + dBMS
operating

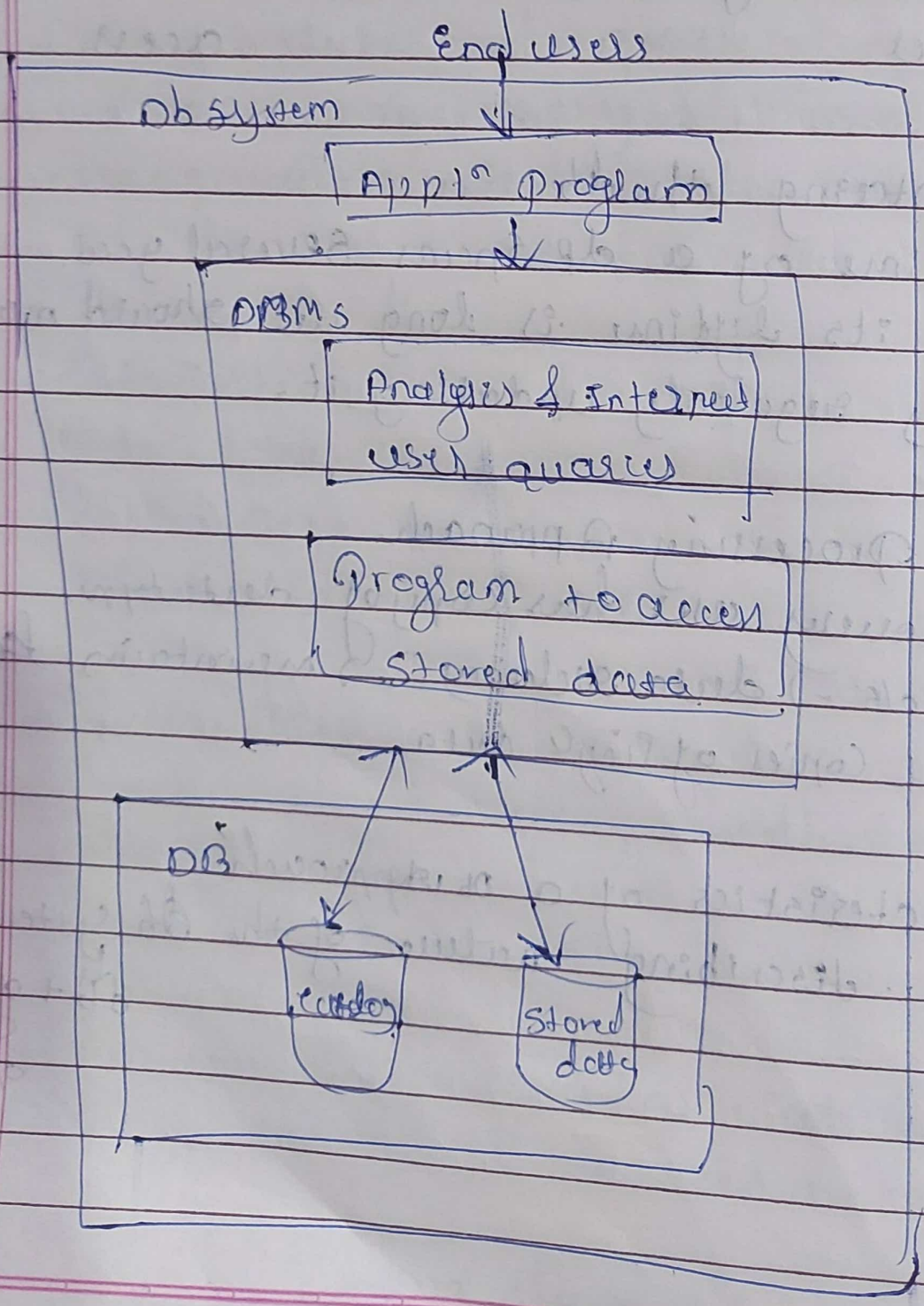
stores the defination of database

catalog → data dictionary → "metadata"

Db system

~~End users~~

The Db system Contains not only db itself but also Complete defination or discription of database structure & Constrains.



(2) insulation bet° programs & data & data abstraction

- Program data independence

→ the ability to change the structure of data without requiring to change the application program that access the data

- Program operation independence

user application programs can operate on the data by invoking the operations (methods) through their names and arguments regardless of how the operations implimented.

(3) Support of multiple views of data.

subset of data

data store in the table not physically

A database typically many users, each of whom may require a different perspective or view of data base.

(4) Sharing of data and multiuser tool transection process

To share a data concurrency control method must be provided by DBMS.

→ transection process.

Strong Entity

Entity that has A key attribute (Primary Key) are called strong Entitys. Single edged rectangle is used for representing a strong entity

Weak entity

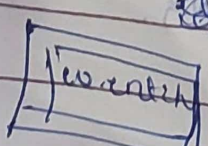
Entity that do not have a primary key of their own are called weak entitys

It has partial key. also called A discriminator

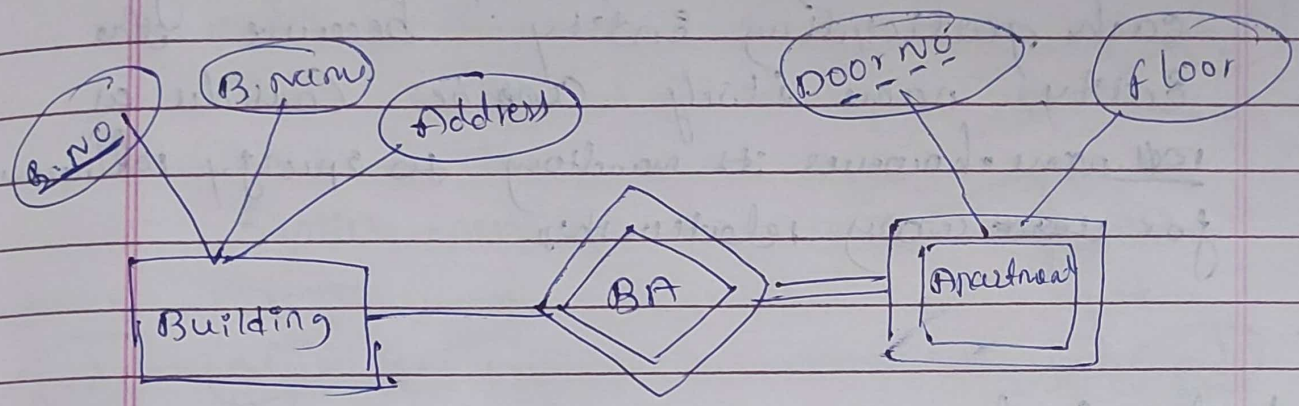
- We can identify weak Entity by attaching Strong to it.
- Weak entity can identified by being related to Specific entities from another entity type (Strong Entity) in Combination with One of their attribute value.
- The entity to which a weak entity is associated is called owner entity (identifying)

Primary key of weak entity set =

its own discriminator + primary key of strong entity



Between strong & weak Entity



Building → strong or identifying or owner entity
primary key of building → B.NO.
apartment → weak entity → partial key of apartment - DOOR NO. identifying
BA → Name of the relationship
primary key of apartment = Door NO + B.NO.
partial key owner key

* Composite

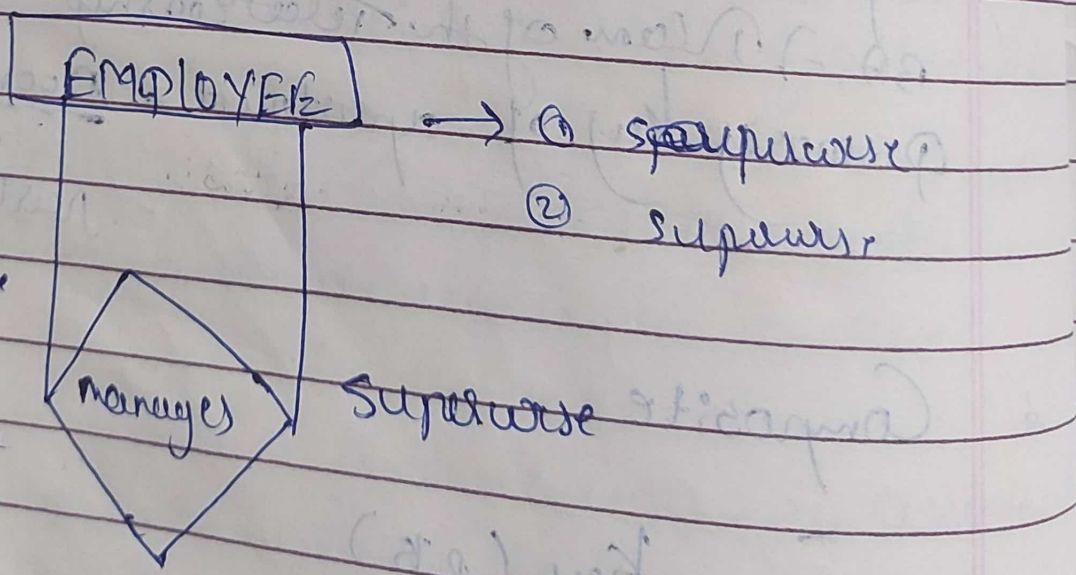
Key (PK)

Simple key
to made up of
only one attribute

Composite key
is a combination of
more than one
attribute

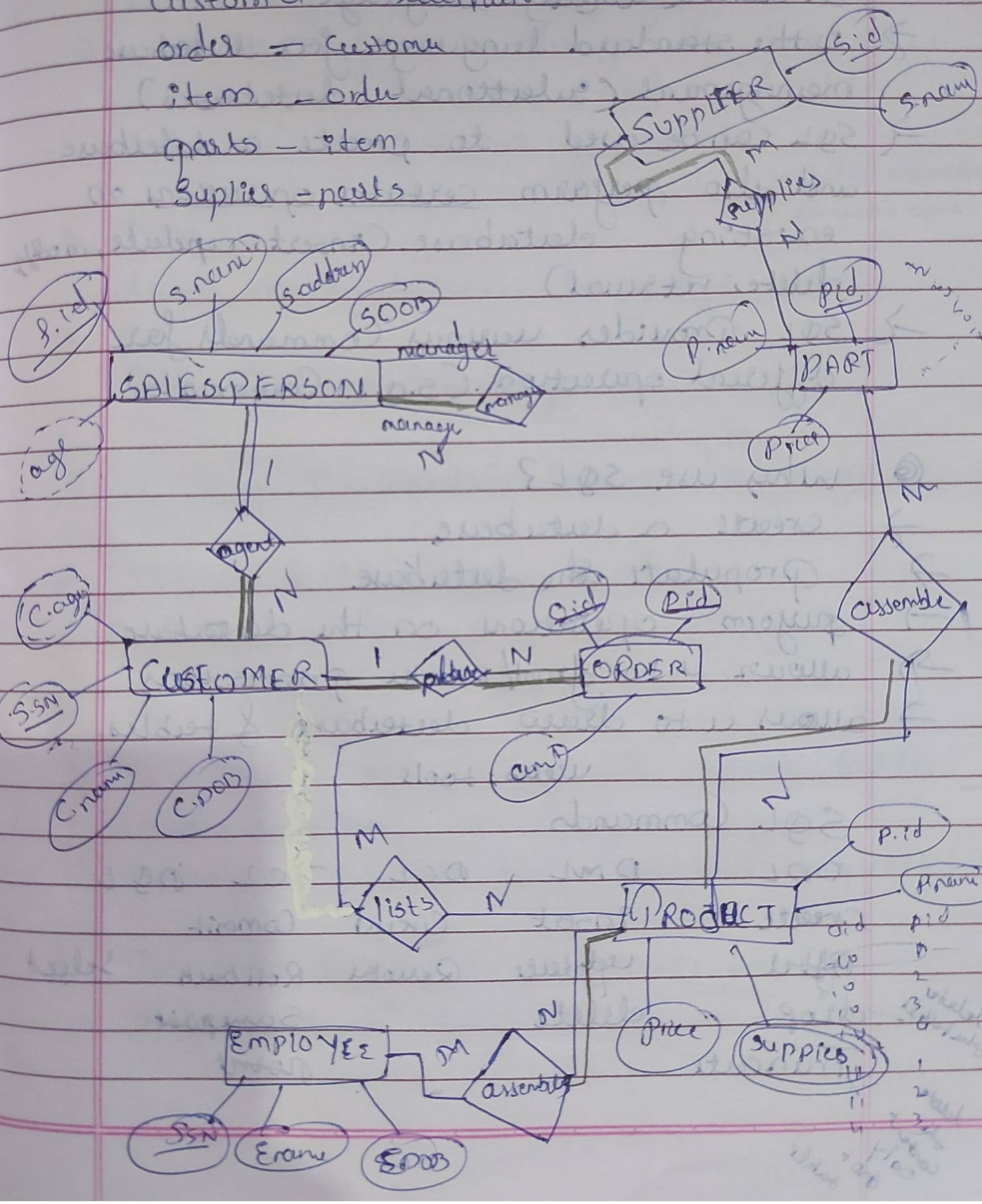
For relationship degree two or more it is not mandatory to specify role name for each participating Entity because the Entity name itself can be Consider as role name. however it is mandatory to specify role name for recursive relationship.

* An Employee is Super



Data / /
Page No.

Salesperson - Salesperson
Customer - Salesperson
order - Customer
item - order
parts - item
Suplies - parts



Unit-4 SQL

* what is SQL?
→ is the standard language for database management (relational databases)
→ SQL can be used to create a database and also perform certain operations on existing database. (Insert, update, delete, retrival)
→ SQL provides various Commands for different operation. (S.Q.L Commands)

* Why use SQL?
→ Create a database
→ Populate the database
→ Perform operations on the database.
→ allows u to grant/revoke permissions
→ allows u to draw databases & tables.

views, tools

SQL Commands

DDL	DML	DCL	TCL	DQL
Create	insert	Grant	Commit	
Alter	update	Revoke	Rollback	Select
drop	delete		Savepoint	
Truncate			Point	

delete the table

delete the content only not table

Date / /
Page No.

* DDL Commands
data definition language
⇒ used to define or change the data base structure or schema / table.
⇒ database is collection of tables

-) auto committed
no undo option

* DML
data manipulation language

* DCL
data Control language

* TCL ⇒ Transaction Control language.

* DQL -) data query language
⇒ It is used to fetch the data as it is.

* Create table
⇒ is a DDL statement used to create tables in the database.
Column name, data type, Constraints, key attributes

* The basic Syntax of the Create table statement is as follows.

Alter table tablename → sure →

ADD column datatype

to

Modify

DROP Column - colum

Rename

CREATE TABLE table-name (

column1 datatype,

column2 datatype,

column3 datatype,

.....

columnN datatype,

PRIMARY KEY (one or more c

);

INT / INTEGER / SMALLINT

REAL / FLOAT / DOUBLE

BOOLEAN / BOOL

Strings — fixed length - CHAR(n) / CHARAC

varying length

VARCHAR(n) /

VARYINGCHAR

Strings " " x ' ' ←

⇒ DESCRIBE name

↳ it gives the structure of the table

INSERT

two ways → direct values or column name

Date / /
Page No.

1) only values → Commas in between

INSERT INTO table_name
VALUES (value1, value2, value3, ...);

(2) Insert data only in specified columns
it is also possible to only insert data in specific columns

INSERT INTO Table_name (column1, column2, column3, ...columnN)
VALUES (value1, value2, value3 ... valueN);

⇒ SELECT * FROM table_name → to see added info

* Specifying Constrains in SQL
- NOT NULL Constrain → Not NULL mentioned that we have to define the value
→ if u donot want a columns to have null value then u need to define the not null Constrain on all such columns
⇒ The NOT NULL Constrain is always implicitly specify for the attribute that are designated as primary key
⇒ Columns level
- if we declare NOT NULL then we cant use NULL in that column

Unique Constraint (column level)

The unique Constrain ensures that all values in a column are different

- Secondary key
- check Constraint
- while adding the values check Constraint check the Conditions
- column level → One check
  More check → column or table level

Constraint - check

Default Constraint

→ column level

Primary key Constraint

→ Primary key must Contain unique values & Can't Contain Null values

- Either tables or column
  - Composite primary key
  - Simple key
- If this is no need to declare Primary key as NOT NULL & Unique

whenever using multiple use 2 constraint (primary key checks)

Date / /
Page No.

```
CREATE TABLE Player (
    ID
```

* The Foreign key Constraint
  -> is a key used to link two tables together
  -> referencing key
  -> it must refer to the Primary key of another table
  - an attribute in one table that refers to the primary key of another table is called Foreign key
  - The table with the foreign key is child table & table with primary key that is being referenced is called parent table.
  - data type of both has to same.

* basic difference bet^n Primary Foreign

Primary	Foreign key
* ensures uniqueness	* link two tables together
* any table can have only one primary key	* table can have any number of foreign keys
* Values for a primary key must be unique	* Can have duplicate values

- Any value, as long as it is unique, can be assigned
- must be one of the values of the primary key in the parent table.

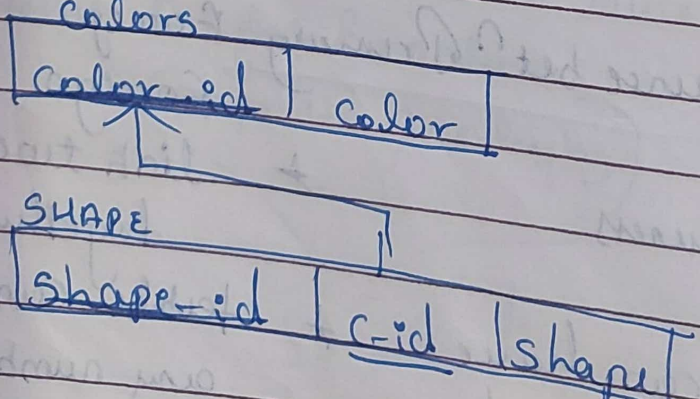
Syntax for foreign key

Column_name datatype references (keyword) name_of_parent_table(Cpk) // any key

→ Column level or table level

- SQL - Commands

Consider the following relations schema. write the create table statement for the same



(21-Jul-20

Date / /
Page No.

```
create table colors
(   color_id integer PRIMARY KEY,
    color varchar(12) NOT NULL
);
```

single shape can have multiple color
single color can have multiple shapes

```
create table shape
(
    shape_id integer PRIMARY key,
    c_id integer NOT NULL,
    Foreign key (c_id) references colors
    shape varchar(12) NOT NULL
);
```

```
Create table shape
(
  shape_id integer,
  c_id integer,
  shape varchar(12),
  Primary key ( shape_id , c_id),
  Foreign key (c_id) references colors (colorid)
);
```

@ table level

Create table shapes
(
shape_id integer,
c_id integer reference colors(c
shape var.char(12),
Primary key (shape_id)
);

Column level

1. red 2. pink 3. Green 4. Blue 5. white

shape id = 1. square
= 2. circle
3. triangle
4. rectangle

→ while entering foreign key we need to focus on f. key value must be same value from primary key of parent table

→ Composite key → the combinations must be different (4, 1) (5, 2) may be same value in same column but must be different combinations

Date / /
Page No.

→ if only simple key present then that must be have different values.

id),

⑥ Select statement

Result set or result table it Contents ~~of~~ of records (rows) return by the select statement.

Basic Syntax

SELECT < attribute list >
FROM < table list > / expression
(optional) WHERE < condition >;
↓ must return boolean value

upper case keyword

→ Select *
→ Select c.id, sharp-id, sname
→ From → table-name

→ attribute specified in select must be present in from.

→ Select statement terminate with ;

Expression → attributes operator. constant value/ attribute value

→ if our table has 5 column and we want only 3 then we can use Select Statement

- the optional where clause is used to specify the Condition while fetching the data from a single or multiple tables

- if the specified or given Condition specified then that row will appear in the result set or it is filtered out
- a Condition is specified by using Comparison or logical operators
  =, <, <=, >, >=, and <> → not equal to

- where WHERE [condition.1] AND [condition
  the SQL and or operator use to Combine multiple Conditions in where Clause

insert into
select from

Date / /
Page No.

* The Delete Command

Delete is a DML Command used for to delete existing records from a table.

→ delete clause have two clauses

```
DELETE FROM table_name
[WHERE [condition];] optional
```

* Update Command.

The update statement is used to update the data of an existing table in database.

```
UPDATE table_name
SET Column1 = expression1,
    column2 = expression2,
    =
    =
[WHERE Columns];
```

* create table
* delete from
* insert into
* Alter table
* Update table
* Select from

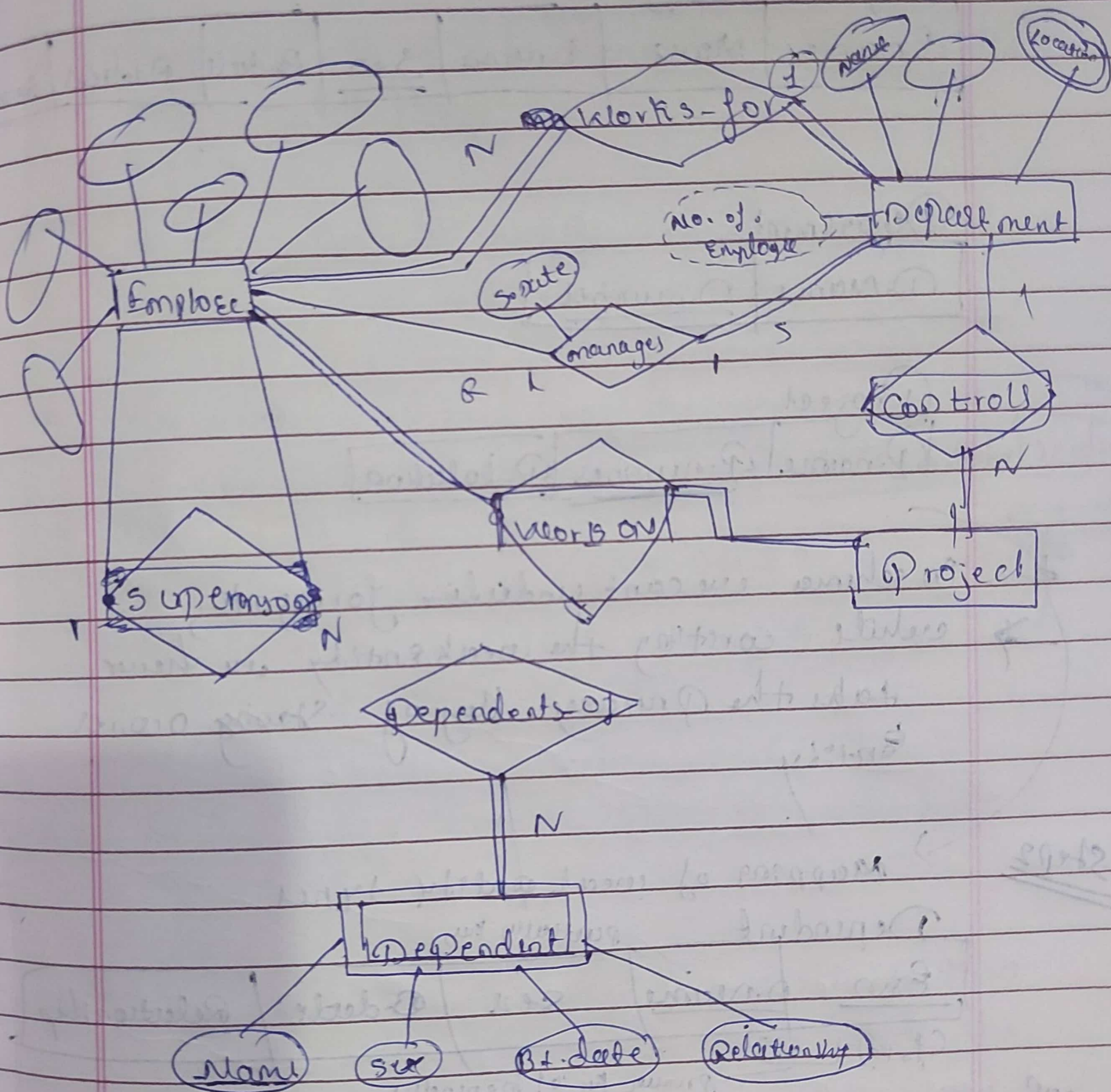
→ Types of Aggregate

- Count
- Sum
- Avg
- Min
- Max

→ Select Count(column) From table_name;
gives the number of rows in column mentioned

→ Select Count(column) From table_name
where (condition);

- Select min(column) From table_name;
- " Max " "
  FROM Column;
  Avg


Works-for
1
N
Name
Location
No. of. Employee
Department
Employee
Start date
manages
S
1
1
Controls
N
Works on
Project
Supervision
1
N
Dependents-of
N
Dependent
Name
Sex
Birth-date
Relationship

Step 1: Mapping of Regular Entity Types

only the entity which has strong entity

Employee

Fname	Minit (middle init)	Lname	Ssn	Bdate	Address

Department

Dname	Dnumber

Project

Pname	Pnumber	Plocation

→ in schema we can't underline foreign key while creating the weak entity we have take the Primary key of owner Entity

Step 2 → Mapping of weak entity types

Dependent

Essn (fork)	Dname (partially key)	Sex	Bdate	Relationship

Primary key of dependent

we can change the entity attribute & the Primary key owner Entity that become foreign key of weak Entity.

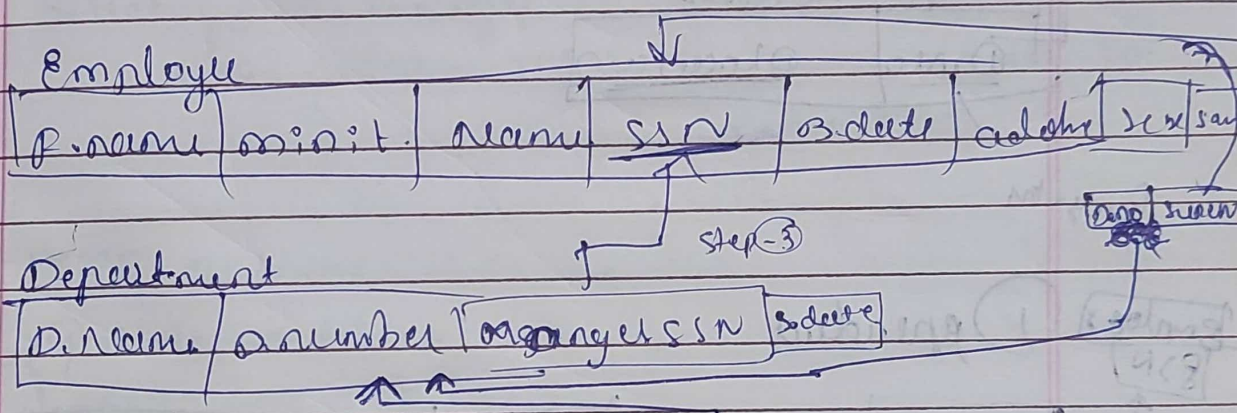
Date / /
Page No.

Step 3

Mapping of Binary 1:1 Relation type.

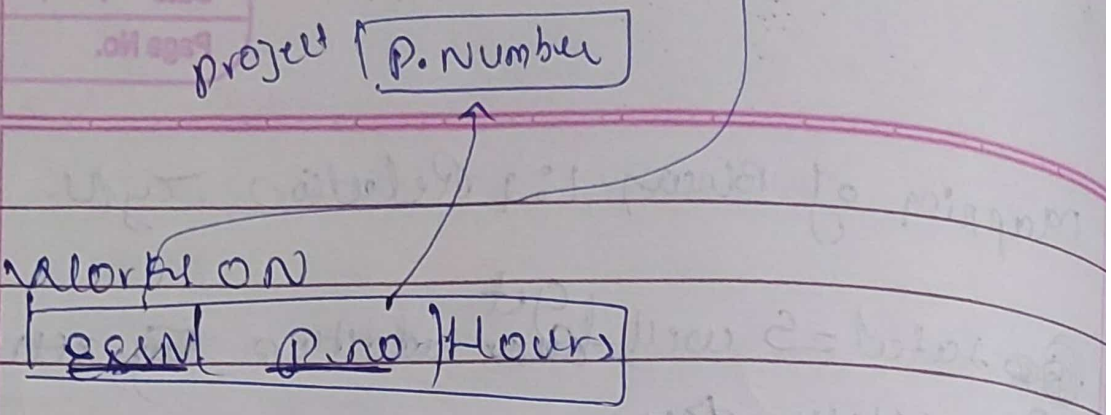
Total = S will get additional attribute.
Partially =

from this choosing total side we have to add one more entity which act as foreign key

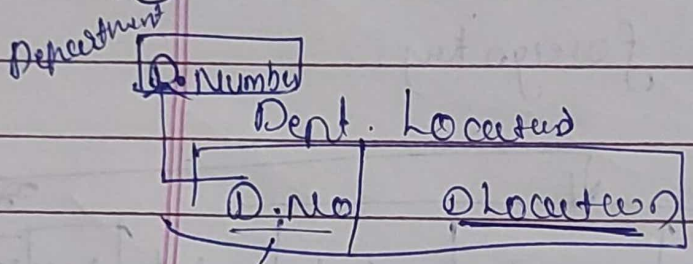


Step 4 N side will get additional attribute

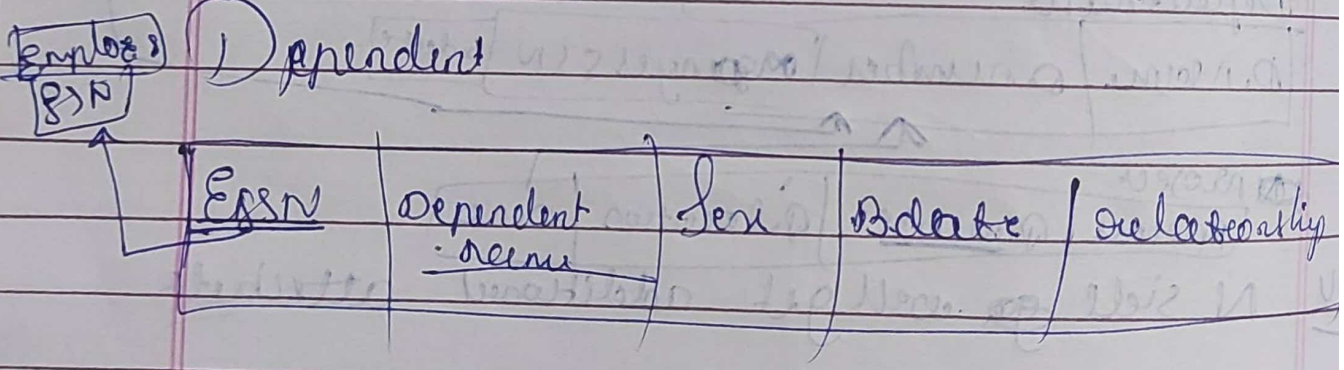
Step-5 M:N → the relation if it has attribute then we have to take them one along with the Primary key of those entity which are having M:N relation



Step (6) Multivalued attribute



Foreign key



retrive Count of no of flights bet' Bengalore &

- Retrieve details of flights origin from Bangalore
- Retrieve Count of no of flight bet' bengalore & frankfart
- Retrieve aircraft whose cruising range bet' 4000 & 15000

Date / /
Page No.

* "Between AND"

where Range Between 1000 AND 1500.

* Value of an attribute from within a specified list IN operator

Eg Select * from employee
where salary IN (3000, 4000, 500, 600);
value ∈ {}

* LIKE operator
⇒ Pattern matching
two
% → matches zero/more chars
_ → exactly one char

Ex Select aid, aname
from aircraft
where aname LIKE 'B%';

any string that matches with upper case B

tuple → row
relation - table

Group By clause

→ It is not command. it is a clause

→ Group By clause is used with the SQL Select statement

Imp → the Group By clause specifies the grouping attributes which should also appear in the Select clause.

Syntax

```
Select <column-list> ([column1, col2, function name
From <table-name> (( one-more tables
Where <condition>
GROUP BY <column>
[HAVING] <condition>
```

function name : SUM(), AVG() name of function

- display number of employees work in each department
- Display total salary paid to employees work in each department.
- display number of employees, total salary paid to employees work in each department

Group By (id)

(1) Select *from dno id, count(
from employee
Group by (did)

Select d.id count(*) "no. of employee" ← rename
from employee
Group by (d.id);

(2) Select Sum.(salary), d-id
from employee,
Group by (d-id.);

(3) Select Sum (salary), d-id.; count(*) ← no of employee
from employee
Group by (d-id);

SQL Group By one more than One columns
The group by clause can include the names of more than one columns as grouping attributes

constraint (No comma separator)
Select column → comma separator
Group by (comma separator)

* Display the department id, job id, total salary paid to in each employee in each department

Rename

```
Select depard_id "department_code"
    job_id, sum(salary) "total salary"
    from employees
Group By department_id, job_id;
```

* SQL Group by with WHERE clause
- display the department code, total salary paid to employees group by department id if manager_id = 103

```
Select did "department code", sum(sal)
"Total Salary"
    from employees
    where manger_id = 103
    Group by department_id
```

The WHERE clause is used to specify the condition on columns. if we want to place condition on Groups use the Having clause.

Having clause (Optional)

1. Display the d_id, number of employee of those groups that have more than 2 employee.

```
Select d_id, count(*) "No of employee"
    from employees
    Group by (d_id)
    // Having (count(*) > 2) ;
    Having count(*) > 2 ;
```

Color & shape

i) retrieve the no of shapes in each color
ii) " " " " " having more than one color.
iii) retrieve the no. of circles or squares in each color.
iv) retrieve c_id of shape for those shapes that is not a circle, square, polygon
v) retrieve shape_id, shape, c_id of shapes whose c_id is 1, 2, 3, 4
vi) Retrieve s_id of shape. sort the output in descending order
vii) Select distinct c_id that is available from shapes.

DISTINCT ⇒ It eliminates duplicates from output.

iii) Select DISTINCT (c-id) from shapes;

ii) Select s.id, shape
from shape
ORDER BY ~~Shape-id~~ s.id DESC;

v) Select s-id, c-id, shape
from shapes
Group by (c-id)
having (c-id <= 4);

iv) Select c-id, shape
from shapes
Where IN ('circle', 'rectangle', 'polygon')

Date / /
Page No.

Joins

join operation is used to Combine data from multiple relation so that related information can represented in a single table.

→ join is a Combination of Cartesian Product followed by selection Process

$R \bowtie S$
condition

$R \bowtie_{<\text{join condition}>} S$

m attributes, n attributes

(m+n) attributes

Cartesian $\Rightarrow R \times S$

Selection from $m \times n$ rows.

→ There are two types

1. Inner join
   - Natural join
   - Theta join
   - Equi join
2. outer join
   - left outer join
   - Right " "
   - Full " "

1) inner join

In this type of join only those tuples which full fill or satisfy the join condition are selected. all those they dont satisfy the condition are excluded

i) natural join (⋈)

Both relations should have atleast one common attribute having same name & domain [datatype] [Rules]

ii) ~~theta join~~ Equi join

In an Equi join we use only the equality operator.

$$R \bowtie_{n=n} S$$

num of attributes

iii) theta join

$$R_1 \bowtie_\theta R_2$$

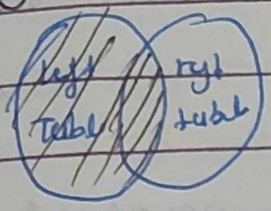
This join uses all types of comparision operators $R_1 \& R_2$ all relation such that they don't have any common attribute.

Commit ;

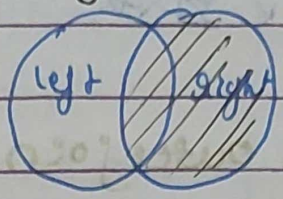
Date / /
Page No.

Join operations (Venn diagrams) types

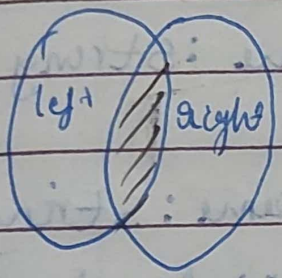
left join



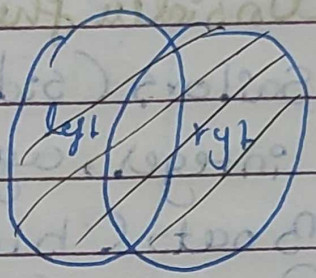
Right join



Inner join



Full join



2) Outer join

in an outer join, along with tuples that satisfy matching criteria, also includes some or all tuples that don't satisfy the criteria.

i) left outer join

$R_1 \bowtie R_2$

R1 C.shirts	R2 C.pants
yellow	red
green	green
blue	blue

$R_1 \bowtie_{R_1.C.shirt = R_2.C.pants} R_2$

condition

ii) Right outer join

$R_1 \bowtie R_1.shirtcolor = R_2.pantcolor R_2$

iii) Full outer join

$R_1 \bowtie R_1.shirtcolor = R_2.pantcolor R_2$

* Consider the following relational schema

Sailors (sid: integer, sname: string, rating: integer, age: real)

Boats (bid: integer, bname: string, color(str))

Reserves (sid: integer, bid: integer, day: date)

Sailors

sid	Sname	Rating	Age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	40
85	Art	3	25.5
95	Bob	3	63.5

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Date / /
Page No.

Reserve

sid	bid	day
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-8
22	104	1998-10-7
31	102	1998-11-10
31	103	1998-11-6
31	104	1998-11-12
64	101	1998-9-5
64	102	1998-9-8
74	103	1998-9-8

a) list names of boats

Select bname from Boats;

→ Select distinct (bname) from Boats;

b) list ratings and ages sailors

Select rating, age from sailors;

→ Select distinct (rating, age) From sailors;

c) list names of sailors who are over 21 years old.

Select fname from sailors

where (age > 21);

d) list names of red boats.

Select bname From boats

where (color = 'red');

e) List ids of boats that have been reserved
Select distinct(bid) from reserve;

g) Find name, age, 2nd rating of all sailors whose name starts with A or contains 'ust' as a substring starting from position 2
Select Sname, age, 2nd rating from sailors
where (Sname Like 'A%' OR Sname Like '_ust%');

g) find names, sids of sailors whose rating +5 is less than their age
Select snames, sid from sailors
where ((rating +5) < age);

h) find names, sides of sailors whose names contain the letter u
Select Snames, Sid from Sailors
where (Sname like '%u%');

i) Select sid as sailor-id, rating/10 as normalized-rating from sailors;
Select sid as sailor-id, rating/10 as normalized-rating from sailors;

i) Retrieve all reservations sorted by sid, with ties broken by bid

```
Select * from Reservations
ordered by sid
```

ii) Retrieve all reservations sorted by sid (descending), with ties broken by bid (ascending)

```
Select * from reservations
ordered by sid desc;
```

i) Find the color of a boat named Clipper.

ii) Find all sailors who are 35 or older, sort them by name.

iii) List the names of red boats, sorted by boat id.

iv) Find all boats that are either red or called Interlake

v) Count the number of diff. Sailor names

i) Select color from boats where bname = 'Clipper';

Select

* inner query will execute first and value is the input for outer query

ii) Select Sname from Sailors
where age >= 35
ordered by Sname

iii) Select bname
from boats
where color = 'red' order by bid;

iv) Select * from boats where color = 'red' or bname = 'interlake';

v) Select Count (distinct (sname)) from sailors;

* Calculate avg age of sailors
Select avg(age) from sailors;
* for each rating, find the avg age of sailors at that level of rating
Select avg(age), ratings from sailors
Group by rating;
* Find the avg age of sailors for each rating level that has at least two sailors.

Date / /
Page No.

rating
Select avg (age) from sailors
Group by rating
Having (count(*) >= 2);
primary key

+ find the ID AND Name of boats reserved by sailor whose id is 22
Select Bname, Boats.Bid from Boats, Reserve
Where reserve.SID = 22 ;
AND Boats.Bid = Reserve.Bid ;

+ find the bid of boats and day's for which ~~Bob~~ Dustin has made reservations
Select bid, day from Reserve, sailors
where Sailors.sid = Reserve.sid AND
Sname = Dustin

+ find the colors of boats reserved by ~~Bob~~ Dustin
+ find the names of sailors who have not reserved
+ boat whose name Contains the string "lake" order the names in ascending order.
+ find all sailors id's of sailors who have a rating of at least 8 or reserved boat 103.

Select color., from sailors, boats, reserves
where sailors.sid = reserve.sid
AND boats.bid = reserve.bid
AND Sname = 'Dustin';

* Select Sname from sailors, boats, reserves
where bname like = 'lake%' AND

ordered by bname;

Set Theory

* To specify any of the operation from set theory on two relation R_1 & R_2, the two relation must be union compatible.
* Two relation said to be union and compatible if and only if they have the same number of attributes & data types of the corresponding pair of attributes must be same.

Date / /
Page No.

* Find unique names and ratings of sailors who have reserved a boat named Interlake and whose name contains the letter 'i'.

```
Select distinct(Sname), rating
From Sailors, Boats, Reserve
Where Sname like '%i%' AND
bname = 'Interlake' AND
Sailors.SID = Reserve.SID AND
Boats.bID = Reserve.bID ;
```

* Select find the name and age of oldest sailor.

* Find the age of the younger sailor for each rating level. where age =

```
Select Sname From Sailors ^ Select MAX(Age)
From Sailors.
```

```
Select min(age), rating From Sailors
group by rating ;
```

* Find the age of youngest sailors for each rating level, such that there are atleast 100 sailors

```
Select min(Age), rating From Sailors.
Group by rating
having Count(*) >= 2 ;
```

- find unique SID and names of Sailors who are reserved red & blue booths.
- find the IDs of Sailors who have reserved a red booth or green booth
- find SID of Sailors who have reserved a red not a blue booth

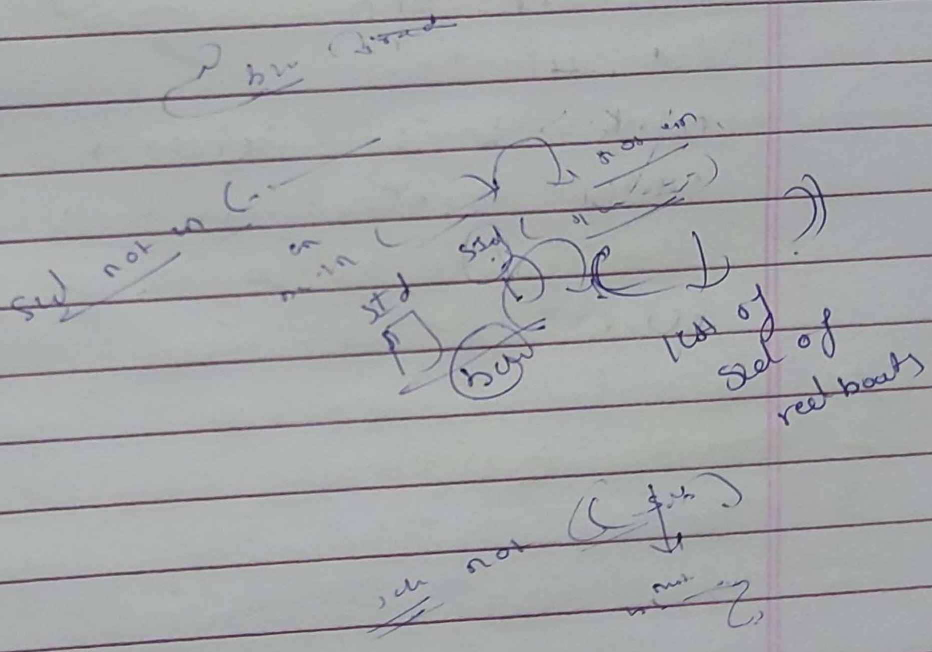
// Select distinct(sname), SID from Sailors S reserve R, boaths b
where b.color = red AND b.color = blue.
AND S.SID = R.SID
AND b.BID = R.BID;

or

Select

- find the names of sailors who have never reserved red boats.
- find SID of sailors who have

Select S.sname from sailors S
where S.sid not in (select R.sid from
Reserves R where R.bid in (select B.bid
from Boats B where B.color = 'red'));



Join
on
Condition

degree: no of selected col

Project operation

- Project operation retrieve certain columns from the table while discarding others

* general Symbol

$$\pi < \text{attribute list} > (R)$$ - Regular relation

* The degree of the resulting relation is equal to the number of attributes in the attribute list.

→ it automatically delete the duplicate tuples. (duplicate elimination)

Then for the result of Project operation a set of distinct tuples

* Commutative does not hold on Project.
* Binary operator
  - i Join
  - ii division

i) Join

Relation Algebra operations from set theory

- i union
- ii intersection
- iii Set difference

Date / /
Page No.

Relational model Constraints

Constraints represents a restriction of some kind on the data that ll be store on the data base

→ Constraints are condition that must hold on all valid relation instances

3 types Relation model Constraints

i) implicit Constraints (Relation model)
ii) Explicit
iii) sigma schematic

↓

imp • Schema - based Constraints

- domain Constraints
- Key "
- Constraints on NULLs,
- entity integrity constraints and
- referential integrity Constraints

• Domain Constraints

if A_i subscript is an attribute in the relation R, the domain of A_i is represented as dom(A_i) or

→ A domain Constraints specify that within each tuple, the value of each attribute A_i

age integer then 1, 22, 24, 25

must be an atomic value from domain

② Keys

A key is a property or hint based on which we can easily select any specific row from many rows.

Types of keys

- Super key
- Candidate key
- Primary key
- Alternate key
- Foreign key

- Super key
  - every relation will have one default super key
  - Combination of all the attributes.
  - Super key which is the superset of the candidate key is combination of all possible attributes which can uniquely identify tuples in a table.

- Consider the following relation
  student (Roll no, name, Reg no). identify the super key for the given relation.

2^n = Subset

Date / /
Page No.

Step 1) List and call the subsets of attributes
2) Eliminate all the subsets that donot meet the super key's requirement
→ how to check → uniquely distribute attribute key with any other attribute.
$t_1[SK] \neq t_2[SK]$

Candidate key

→ Candidate key is a minimal super key
→ It is the subset of super key
* the keys which not redundant those have to eliminate. [Roll_no, name] = [Roll_no]
* if we have both keys as unique then also those are not Candidate key using one we can identify.

Primary key

* minimum number of attribute have to select
* Select such a key whose value never change
* in worst case we have to select the key one random

Alternate key

the key which is eliminate from Primary key.

Example

Student ID	Student Enroll	Student name	S. Email
S02	4545	Dave	ddd@gmail
S34	4541	Jack	jj@gmail
S22	4555	Mark	mm@gmail

1. Super keys $2^n = 2^4 = 16$ $2 \times 2 \times 2 \times 2$

{Student ID} ✓

{Student Enroll} ✓

{Student name} ✗

{S. email} ✓

{S. ID, Student Enroll} ✗

{Sta name, S. email} ✗

{St. mail, St. ID}

{St. Enroll, St. name}

{St. ID, St. name}

{St. ID, St. email}

{St. Enroll, Student ID}

{Enroll, Email} ✗

{name, ID}

{name, Enroll}

{email, Enroll}

{Student. ID, Enroll, name, email}

{ID, Enroll, name}

{Enroll, name, email}

{ID, name, email}

{ID, email, Enroll}

Date / /
Page No.

* Candidate
{ID}
{enroll}
{email}

* Primary key -> ID
* alternate key -> {Enroll}, {email}

* NULL Constraint
* if we use the Not Null Constraint then we must assign any value to that attribute.
→ the Not Null Constraint makes sure that a column doesnot hold NULL values
* Entity integrity Constraint
Primary key should not be NULL

↓ Referential integrity Constraint
it always specified but two tables
⇒ It is enforced using a foreign key

↓ foreign key
→ foreign key always references a primary key or a path of the primary key in the parent table and has the same domain
⇒ (data type as that of the primary key.

↓ operations of Relational algebra

Table T_1

P	Q	R
10	q	3
15	b	8
25	a	6

Table T_2

A	B	C
10	b	6
25	a	3
10	b	5

degree = no of attributes

↓ Corresponding attribute must be same

P = A - int
Q = B - str
R = C - int

$T_1 = 3$ & degree
$T_2 = 3$

Union & Compatible

Date / /
Page No.

T_1 & T_2 are union compatible thus we can proceed to result of specified operations

$T_1 \cup T_2$

degree same

P	Q	R
10	a	5
15	b	8
25	c	6
10	b	6
25	c	3
10	b	5

$T_1 \cap T_2$

No common row

Null set

$T_1 - T_2$

P	Q	R
10	a	5
15	b	8
25	a	6

$T_2 - T_1$

A	B	C
10	b	6
25	c	3
10	b	5

$T_1 \times T_2$

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	25	c	3
10	a	5	10	b	5
15	b	8	10	b	6
15	b	8	25	c	3

15	b	8	10	b	5
25	a	6	10	b	6
25	a	6	25	c	3
25	a	6	10	b	5

The two tables T_1 & T_2 show the results of the following

a) $T_1 \bowtie \Pi P = T_2 A T_2$

b) $T_1 \forall T_1 q = T_2 B T_2$

c) $T_1 \bowtie T_1 . P = T_2 A T_2$

d) $T_1 \bowtie T_1 q = T_2 B T_2$

Join → table1 ⋈ table2

(6)

a)

P	q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
25	a	6	25	c	3

b) $T_1 \bowtie T_1 q = T_2 B T_2$

P	q	R	A	B	C
15	b	8	10	b	6
15	b	8	10	b	5
~~25~~	~~c~~	~~6~~	~~25~~	~~c~~	~~3~~

Date / /
Page No.

c) $T_1 \bowtie T_1.P = T_2.A\ T_2$

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	NULL		
10	a	5	10	b	5
10	a	5	NULL	NULL	NULL
15	b	8	NULL		
15	b	8	NULL		
15	b	8	NULL		
25	a	6	NULL		
25	a	6	25	c	3
25	a	6	NULL		

b) $T_1 \bowtie T_1.P = T_2.A\ T_2$

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
25	a	6	25	c	3
15	b	8	NULL	NULL	NULL

d) $T_1 \bowtie T_1.Q = T_2.B\ T_2$

P	Q	R	A	B	C
15	b	8	10	b	6
15	b	8	10	b	5
NULL	NULL	NULL	25	c	3

f) $T_1 \bowtie (T_1.P = T_2.A \text{ AND } T_1.R = T_2.C)\ T_2$

P	Q	R	A	B	C
10	a	5	10	b	5

Example

Member (M.ID, Name, date of Birth)

Book (Book.ID, Name, Author)

Borrow (M.ID, BID, B.Date, R.Date)

borrow return

1. ID's of the member who have borrowed books

Select distinct(m.ID) from borrows

$\Pi_{M.ID}$ (Borrows)

2. ID's of the member & ID's of books they have borrowed

Select M.ID, b.ID from borrows

$\Pi_{M.ID, B.ID}$ (Borrows)

3. details of the member who are born in 20/10/1997

horizontal

Select $\sigma_{DOB = '20/10/1997'}$ (Member)

Select * from member

where dob = 20/10/1997

Date / /
Page No.

1. retrieve name & M.ID of member who born in 21/10/1997

$\Pi_{memberID, name}(\sigma_{DOB='21/10/1997'}(Members))$

Select name, MID from member
WHERE DOB = 21/10/1997;

2. borrowing details of the member with ID = 3

$\sigma_{MID=3}(borrows)$

Select * From borrows
WHERE MID = 3

, b.ID = 3 or b.ID = 5

$\sigma_{b.ID=3 \vee b.ID=5}(borrows)$ ∧ - and

3. Name of the books authored by Dan Brown OR Malinda Brown

Select name, from Book
WHERE author = 'Dan' OR author = 'Malinda'

$\Pi_{name}(\sigma_{author=Dan\ Brown \vee malinda\ Brown}(books))$

7 → Name of books authored by ~~Dan Brown~~ AND Malinda Brown

$\Pi_{Bname}(\sigma_{author='DB' \wedge author='MB'}(BOOK))$

8 → ID's & date of birth of member charlie

~~$\Pi_{MID, DOB}(Member)$~~

$\Pi_{MID, DOB}(\sigma_{name='charlie'}(member))$

Select MID, DOB From Member
Where name = charlie ;

V

10 → Name of all members & author

// Select member.name, Book.author
From Member, Book ;

9) Names of members and authors whose ID's are the same

Date / /
Page No.

Names of authors whose books have been borrowed

$\Pi_{Book.author}(Book \bowtie Borrow)$

Select Book.author from Book, Borrow
where Book.ID = Borrow.ID;

Names of members who have borrowed book

$\Pi_{name}(Book \bowtie Borrow)$ (natural join)
(Member.name)

BookID of Books borrowed by Chare and Mehek

Member ID of Members who have borrowed both the books "Fences and Inheritance"

Member ID of members who have never borrowed books

Relational Algebra the division operation

- Division operation is the binary operation
- It is the Extended operation of relational algebra.
- notation for division $A \div B$ A / B

A & B are relations

Attributes of B is Proper Subset of Attributes of A

$B \subset A$

- for all & for every

degree of resulting relation = Number of attributes in A − Number of B

- the relation returned division operator which Contains those tuples from A which are associated to every B's tuples

A

SNO	PNO
S_1	P_1
S_1	P_2
S_1	P_3
S_1	P_4
S_2	P_1
S_2	P_2
S_3	P_2
S_4	P_2
S_4	P_4

B1

PNO
P_2

B2

PNO
P_2
P_4

B3

PNO
P_1
P_2
P_4

$B_1 = \{Pno\}$

$A = \{Sno, Pno\}$

A/B1 → B's attribute (Sno)

A/B1 =

Sno
S1
S2
S3
P4

P2 associated with Sno

A/B2 =

Sno
S1
S4

P2 → S1 S2 S3 S4

P4 → S1, S4

A/B3

Sno
S1

P1 →

P2 → S1 S2 S3 S4

P1 → S1, S2

P4 → S1, S4

* Division R ÷ S
  defines a relation over the attributes C that consists of set of tuples from R that match combination of every tuple in S
* Expressed using basic operations

$T_1 \leftarrow \Pi_C(R)$

$T_2 \leftarrow \Pi_C((S \times T_1) - R)$

$T \leftarrow T_1 - T_2$

Consider the following schema

Suppliers (sid: integer, sname: string, address: str)
parts (pid: integer, pname: string, color: str)
catalog (sid: integer, pid: integer, cost: real)

Find the name of suppliers who supply some red or green parts

$R_1 \leftarrow \sigma_{color = red}(parts)$
(pid, pname, color)

$R_2 \leftarrow (catalog \bowtie R_1)$
sid, pid, cost, pname, color

$R_3 \leftarrow (suppliers \bowtie R_2)$
sid, sname, add, pid, cost, pname, color

$R_4 \leftarrow \Pi_{sname}(R_3)$

or

$\Pi_{sname}(\sigma_{color = red}(S \bowtie P \bowtie C))$

Date / /
Page No.

find the sids of suppliers who supply every part

Suppliers
sid, Sname, add

⇒ $R_1 \leftarrow \pi$ sid, pid Catlog
$R_2 = \pi$ pid parts
R_1 / R_2 → give all the required (s)# of sids of suppliers who supply every part

* Find the sids of suppliers who supply every red part

$R_1 \leftarrow \pi$ sid, pid catlog
$R_2 \leftarrow \pi$ pid ⌐ color = 'red' (parts)
so required answer is R_1 / R_2

* Find the sids of suppliers who supply every redor green part

$R_1 \leftarrow \pi$ sid, pid Catlog
$R_2 \leftarrow \pi$ pid ⌐ color = 'red' ∨ color = green (part)

R_1 / R_2

Date / /
Page No.

Update operations & dealing with Constraints violations

Relational model

Retrieval

update operation
- insert
- Delete
- update

1) Insert
- Domain Constraint
- Key
- Entity integrity Constraint
- referential integrity Constraint

Action by DBMS
when one or more Constraints violated then
- Reject the insertion
- Accept the insertion after modification

2) Update
- domain Constrain (if the id is integer as of declared and we r updating it as str)
- Key
- Entity integrity Constraint.
- referential

Action by dBms
Reject the 'insertion'
Accept the insertion after modification

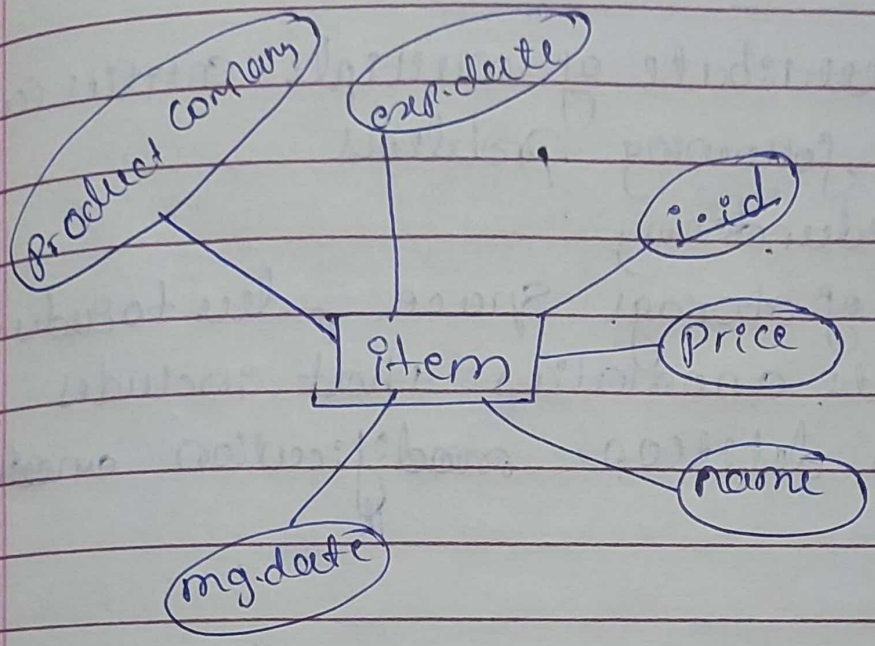
2) DELETE
→ Referential integrity Constraint.
if u r deleting the foreign key value which is referencing the primary key of the another table

Actions
1) On delete set null (when the primary get deleted then the corresponding foreign key get the value null)
2) on delete cascade
3) on delete no action (default action) while deleting the primary key
4) on delete SET default. (value default)

on delete cascade - if die delete Parent table value then Corresponding foreign key also get delete

On delete No action)
default action is taken by dBms

III. Data base Design & transaction processing Concepts



Normalization is the formal way of analyzing which one grouping of attributes into a relational schema may be better than another

Informal Design Guidness

used to determine the quality of (goodness) relations schema design

Guideline 1.

- Imparting Clear Semantics to Attributes or Relations

design a relation schema so that it is easy to explain its meaning. do not combine attributes from multiple entity & relationship

type into a single relation

mixing attribute of multiple entities can cause the following problems

1) data dedundency
2) wastage of storage space due to redundency
3) update anomalies that includes insertion, deletion, modification anomalies

Guidline 2

design the schema that doesnot suffer from update anomali. if there are any present note them down so that application can be made to take them into account

Guidline -3

Null values in tuples

* Problems arising due to Null values in Tuples.
→ wasted of storage space
→ multiple or different interpretation
→ results of select and join operation involving Comparisions may become unpredictable
→ Aggregate donot account for Null values

Date / /
Page No.

Generation of Spurious Tuples

Emp-Locs

Ename	Plocation

Emp-Proj:

SSn	Pnumber	Hours	Pname	Plocation

Wedding-y

design the relation schema they can regain equality condition that are appropriate related. in such a way that it guarantee no spurious tuples are generated

* formal → Functional dependencies
* Functional dependencies along with keys are used to define normal forms for relation — which are formal measures of goodness of relational designs.
* FD is a constraint between two attributes or two sets of attributes and typically exists bet^n the primary key attribute and non-Primary key attributes.

$X \rightarrow Y$

(determinate) (dependent)

X is functionally determines Y
y is functionally dependent on X

- There is FD from the primary key to each of the other attributes in table
- "legal instance of a relation" which satisfies the all the FD.

Example

rollno	name	dept-name	dept-building
u2	abc	CD	A4
u3	pqr	IT	A3
u4	xyz	CD	A4
u5	xyz	IT	A3
u6	mno	EC	B2
u7	jkl	ME	B2

rollno → {name, dept-name, dept-blg}
rollno → name
rollno → dept-name
rollno → dept-building
dept name → dept building

A	B	C	Tuples
10	b_1	c_1	1
10	b_2	c_2	2
11	b_4	c_1	3
12	b_3	c_4	4
13	b_1	c_1	5
14	b_3	c_4	6

i) $A \rightarrow B$ NO $A[\,] \neq [B]$

ii) $B \rightarrow C$ yes

iii) $C \rightarrow B$ NO

iv) $B \rightarrow A$ NO

v) $C \rightarrow A$ NO