- Relational Algebra

It is a basic set of operations for relational model

These operations enable a user to specify basic retrieval query

A sequence of relational algebra operation forms a relational Algebra Expression

i) Unary Relational Operations

- SELECT (Symbol : σ (sigma))
- PROJECT (Symbol : π (pi))
- RENAME (Symbol : ρ (rho))

ii) Relational Algebra Operation from Set Theory
- Union ($\cup$), Intersection ($\cap$), Difference (or MINUS $-$)
- Cartesian product ($\times$)

iii) Binary Oper^n
- JOIN ($R3 \leftarrow R1 \bowtie_{condition} R2$)

Unary Relan opera^n.

i) select.

Selection condition acts as filter.

1) Write a Relational algebra query. (Sql). to retrieve all employee detail who work for. Dept 5

Select Row filtering

$\sigma_{DNO=5}$ (Employee)

General Syntax : Select Operation of Rela^n algebra

$$\sigma_{\langle condition \rangle} (R)$$

PROJECT column filter

→ Write a Relational algebra to retrive, 1st name, Last name SSN of employee

General Syntax :

$$\Pi_{\langle attribute1, attribute2 \ldots \rangle} (R).$$

$$\Pi_{Fname, Lname, SSN} (EMPLOYEE)$$

→ Write RAE to Retrieve First name, Lname, SSN of all employee those who work for dept=04

$$R_1 \leftarrow \sigma_{DNO=4} (EMPLOYEE)$$

$$R_2 \leftarrow \Pi_{fname, Lname, SSN} (R1)$$

→ WHTE RAE to dependent table details

$$\Pi_{Essn, Dependent\_name, Sex, Bdate, Relationship} (DEPENDENT)$$

→ Write RAQ to retrieve all dept details

$\Pi_{Dname, Dnumber, mgr\text{-}ssn, mgr\text{-}start\text{-}date}$ (DEPARTMENT)

→ Write Query to Retrieve name of department which is having dept no = 4.

* $R1 \leftarrow \sigma_{DNO=4}$ (DEPARTMENT)

* $R2 \leftarrow \Pi_{Dname}(R1)$

OR $R2(Dept\text{-}name) \leftarrow \Pi_{Dname}(R1)$.

R1, ↳ For Rename

Administration	4	98765432	1995-01-01

R2,

Dname
Administration

→ Write RAQ to Retrieve details of Employee those who work for Research dept,

General Syntax for JOIN, → [cartesian product followed by Rows filtering]

$R3 \leftarrow R1 \bowtie R2$ (Join condition)

↳ Foreign key, Primary key

DNO = Dnumber

For Join, (All the Employee)

$R3 \leftarrow$ EMPLOYEE $\bowtie$ DEPARTMENT,

DNO = DNUMBER.

For Research,

$R4 \leftarrow \sigma_{DName = Research}(R3)$.

For Fname, Lname, SSN

$R5 \leftarrow \Pi_{Fname, Lname, SSN}(R4)$

→ Write RAQ to Retrieve Employee Fname, Lname of all managers.

$R_1 \leftarrow$ EMPLOYEE ⋈ DEPARTMENT.
SSN = mgr-SSN

$R_2 \leftarrow \Pi_{Fname, Lname}(R1)$

→ Write a RAQ Department details which is located in Houston

$R_1 \leftarrow \sigma_{Dlocation = Houston}(Dept\_location)$

Dept full table

Dnumber	Dloc
1	Houston
5	Houston

$R_2 \leftarrow R_1 \bowtie_{Dnumber = Dnumber.}$ Department [Equi Join — Dno. colum appears twice]

Natural Join,
Represented Using *.
$R_3 \leftarrow R_1 * Dept$ [Dnumber column appears Once]

→ Write a RAQ to Retrieve DName which is Located in Mangalore.

$R_1 \leftarrow \sigma_{Dlocation = mangalore}(Dept\_location)$

→ Write RAG to Retrieve employee Fname, Lname who works on Project no 2.

$R_1 \leftarrow \sigma_{PNO=2}(WORKS\text{-}ON)$

$R_2 \leftarrow \pi_{ESSN}(R_1)$. $R_2(SSN) \leftarrow \pi_{ESSN}(R_1)$

$R_3 \leftarrow R_2 \bowtie EMPLOYEE$. $R_3 \leftarrow R_2 * EMPLOYEE$

$Result \leftarrow \pi_{fname,lname}(R_3)$

→ Write a RAG to employee Fname, Lname who works on Project located in Houston

$R_1 \leftarrow \sigma_{Plocation=Houston}(PROJECT)$.

$R_2 \leftarrow \pi_{Pnumber}(R_1)$.

$R_3 \leftarrow R_2 \bowtie_{Pnumber=Pno} WORKS\text{-}ON$.

$R_4 \leftarrow \pi_{ESSN}(R_3)$.

$R_5 \leftarrow R_4 \bowtie_{ESSN=SSN} EMPLOYEE$.

$R_6 \leftarrow \pi_{Fname,Lname}(R_5)$.

→ Write a RAG to retrieve Employee Fname, Lname who works on Project no 2 and 1

$R1 \leftarrow \sigma_{PNO=1}(WORKS\text{-}ON)$

$R2 \leftarrow \pi_{ESSN}(R1)$

$R3 \leftarrow \sigma_{PNO=2}(WORKS\text{-}ON)$

$R4 \leftarrow \pi_{ESSN}(R3)$

$R5(SSN) \leftarrow R2 \cap R4$

$R6 \leftarrow R5 * EMPLOYEE$

$R7 \leftarrow \pi_{Fname, Lname}(R6)$

→ Write a RAQ to Retrieve maximum salary given to employee.

Aggregate Function (F

FMAX Salary (EMPLOYEE) retrieves the maximum salary value from the employee relation

FMIN Salary (EMPLOYEE) retrieves the minimum ~~maxin~~ salary value from the employee Relation

FSum Salary (EMPLOEE) retrieves the Sum of Salary from the EMPLOYEE Relation

→ Write RA to retrieve maximum salary paid in each department

$R_1 \leftarrow {}_{DNO}\mathcal{F}_{MAX\ salary}(EMPLOYEE)$ retrieves the maximum salary value from the Employee relation

→ Write RA to retrieve minimum salary given in each department.

$R_2 \leftarrow {}_{DNO}\mathcal{F}_{MIN\ salary}(EMPLOYEE)$ retrieves the minimum salary value from the EMPLOYEE relation

→ Query to Retrives total no of employee.

$R_1 \leftarrow {}_{}\mathcal{F}_{count\ SSN}(EMPLOYEE)$

$R_1 \leftarrow {}_{}\mathcal{F}_{count\ SSN}(EMPLOYEE)$ computes the count (number) of employees.

count SSN	salary
8	281000

→ Write RA Retrieve employee Fname and Lname those who are having dependent

$R_1 \leftarrow \Pi_{ESSN}(DEPENDENT)$ $R_1(SSN) \leftarrow \Pi_{ESSN}(DEPEN)$

$R_2 \leftarrow R_1 \boxtimes EMPLOYEE$ $R_2 \leftarrow R_1 \times$ Employer

$ESSN = SSN$.

Ra,	Employee	
Essn	ssn	
333445555	123456789	333445555
333445555	333445555	98765432
333445555	999887777	123456789
987654321	987654321	
123456789	66688444	
123456789	45345345	
123456789	987987987	
	888665555	

union compatibility

Write a Query to Retrieve Fname Lname of employee those who have not dependent

$R1 \leftarrow \Pi_{ESSN}(DEPENDENT)$

$R2 \leftarrow \Pi_{SSN}(EMPLOYEE)$

$R3 \leftarrow R2 - R1$

$R4 \leftarrow R3 \underset{SSN=SSN}{|X|} (EMPLOYEE)$

$R5 \leftarrow \Pi_{Fname, Lname, SSN}(R4)$

$R1 \leftarrow \Pi_{ESSN}(Dept)$

ESSN
333445555
333445555
333445555
987654321
123456789
123456789
123456789

$R2 \leftarrow \Pi_{Fname, Lname, SSN}(Employee)$

Fname	Lname	SSN
John	Smith	123456789
Franklin	Wong	333445555
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Ramesh	Narayan	666884444
Joyce	English	453453453
Ahmad	Jabbar	987987987
James	Borg	888665555

ESSN	Fnamc	Lnamc	SSN
333445555	John	Smith	123456789
333445555 →	Franklin	Wong	333445555
333445555	Alicia	Zelaya	999887777
333445555	Jennifu	Wallamce	98765432l
333445555	Ramesh	Narayan	66 6884444
333445555	Joyce	English	4534 53453
333445555	Ahmad	Jabbar	98798787
333445555	Jamcs	Borg	888665555

333445555	John	Smith	123456789
333445555	Franklin	Wong	333445555
333445555	Alicia	Zelaya	999887777
333445555	Jenifer	Wallance	98165432l
333445555	Ramesh	Narayan	66688444
333445555	Joycec	English	453453
33344455555	Ahmad	Jabbar	98798787
33344455555	Jamcs	Borg	888665555

333445555	John	Smith	123456789
333445555	Franklin	Wong	333445555
333445555	Alicia	Zelaya	999887777
333445555	Jenifer	Wallance	999887777
333445555	Ramesh	Narayan	666884444
333445555	Joyce	English	453455453
333445555	Ahmad	Jabbar	987987987
333445555	James	Borg	888665555

987654321	John	Smith	123456789
987654321	Franklin	Wong	333445555
987654321	Alicia	Zelaya	999887777
987654321	Jennifer	Wallace	987654321
987654321	Ramesh	Narayan	666884444
987654321	Joyce	English	453453453
987654321	Ahmad	Jabbar	987987987
987654321	James	Borg	888665555

123456789	John	Smith	123456789
123456789	Franklin	Wong	333445555
123456789	Alicia	Zelaya	999887777
123456789	Jennifer	Wallace	987654321
123456789	Ramesh	Narayan	666884444
123456789	Joyce	English	453453453
123456789	Ahmad	Jabbar	987987987
123456789	James	Borg	888665555

123456789	John	Smith	123456789
123456789	Franklin	Wong	333445555
123456789	Alicia	Wallace	999887777
123456789	Jennifer	Zelaya	987654321
123456789	Ramesh	Narayan	666884444
123456789	Joyce	English	453453453
123456789	Ahmad	Jabbar	987987987
123456789	James	Borg	888665555

123456789	John	Smith	123456789
123456789	Franklin	Wong	333445555
123456789	Alicia	Zelaya	999887777
123456789	Jennifer	Wallace	987654321
123456789	Ramesh	Narayan	666884444
123456789	Joyce	English	453453453
123456789	Ahmad	Jabbar	987987987
123456789	James	Borg	888665555

To apply Set Operation,

Both the Relation must be Union compatible condition

i) Same no. of column

ii) Associated datatype corresponding to that datatype same.

→ Write a RAQ to Retrieve employe Fname, Lname, SSN those who are having more than one dependent

Next page

→ How many Dependent

R1 ← $\mathcal{F}$ count ESSN (DEPENDENT).

count ESSN
7

$R_1 \leftarrow$ ESSN $\mathcal{F}$ count (Dependent_name (Dependent))

R1

ESSN	count(Dependent_name)
333445555	3
987654321	1
123456789	3

R2

ESSN	count(DEPT_no)
333	1
123	3

$R_2 \leftarrow \sigma_{count(Dependent\_name) > 1}$

$R_3 \leftarrow \Pi_{ESSN}$ (R2)

ESSN
333
123

$R_4 \leftarrow R_3 \underset{ESSN=SSN}{\bowtie}$ (EMPLOYEE)

$R_5 \leftarrow \Pi_{fname, Lname, SSN}$ (R4)

→ Write RA to Retrieve no. of employee who works on different departments

$R_1 \leftarrow \mathcal{F}_{count\ SSN}$ (EMPLOYEE)

$R_2 \leftarrow$ DNO $\mathcal{F}_{count\ SSN}$ (EMPLOYEE)

DNO	count(SSN)
5	4
4	3
1	1

→ Write a RA Q to retrieve ESSN those who works on more than Project.

$R_1 \leftarrow$ ESSN $\mathcal{F}$ count Pno (WORKS-ON)

ESSN	Pno
123456789	2
666884444	1
453453453	2
333445555	4
999887777	2
987987987	2
987654321	2
888665555	1

→ Write Query to Retrieve Employee Fname, Lname, SNN on department which is located in more than table.

$R_1 \leftarrow$ Dnumber $\mathcal{F}$ count DLocation (Dept-Locati

$R_2 \leftarrow \sigma$ count(DLocation) > 1

$R_3 \leftarrow \pi_{DNumber}(R_2)$

$R_4 \leftarrow R_3 * EMPLOYEE$

$R_5 \leftarrow \pi_{Fname, Lname, SSN}(R_4)$.

→ Write a RAQ to Retrieve employee First name, Lname and works on atleast 2 project

$R1 \leftarrow ESSN\ \mathcal{F}_{count\ Pno}(WORKS\text{-}ON)$.

$R2 \leftarrow \sigma_{count(Pno) > 1}$

$R3 \leftarrow \Pi_{ESSN}(R2)$.

$R4(SSN) \leftarrow R3 \bowtie_{ESSN = mgr\text{-}SSN} DEPARTMENT$.

$R5 \leftarrow R4 * EMPLYOEE$

$R5 \leftarrow \Pi_{fname, Lname}(R5)$

Wrong Query

→ Write a RAQ to Retrieve dependent names of Employee who works on Depart no 5 and 4

→ Write a RAQ to Retrieve the Employee Fname, Lname who Relationship as Spouse.

$R1 \leftarrow \sigma_{Relationship = spouse}(DEPENDENT)$

$R2 \leftarrow \Pi_{ESSN}(R1)$

$R3 \leftarrow R2 \bowtie_{ESSN == SSN} EMPLOYEE$

$R4 \leftarrow \Pi_{fname, Lname}(R3)$

→ Write a RAQ to Retrieve average salary of male employes who works for deptno 5

$R_1 \leftarrow \sigma_{SEX=M}(EMPLOYEE)$

$R_2 \leftarrow \sigma_{DNO=5}(R_1)$

$R_3 \leftarrow \pi_{salary}(R_2)$

$R_4 \leftarrow FAvg\ salary(R_3)$

→ Write a Query to Retrieve employee Fname, Lname who is both manager and Supervisor

$R_1 \leftarrow_{(SSN)} \pi_{mgr\text{-}SSN}(DEPARTMENT)$

$R_2 \leftarrow \pi_{sup\text{-}SSN}(EMPLOYEE)$

$R_3 \leftarrow_{(SSN)} R_2 \cap R_1$

$R_4 \leftarrow R_3 * EMPLOYEE$

$R_4 \leftarrow \pi_{fname, Lname.}(R_3)$

Binary Relational operations : DIVISION

Division operation.

- Division Opeation applied to two Relations
- $R(Z) \div S(X)$, where X Subset Z.
  Let $Y = Z - X$ (and hence $Z = X \cup Y$);
  that is, Let Y the set of attributes of R that are not attributes of S.
- The Result of DIVISION relation $T(Y)$ includes tuples in f:

→ Write a RAQ to Retrieve the employee details SSN all employes who works on all project in which john Smith work_on

$R1 \leftarrow \sigma_{Lname=Smith}(EMPLOYEE)$

$R2 \leftarrow \pi_{SSN}(R1)$

$Smith\_PNOS \leftarrow \pi_{Pno}\left(R2 \;|X|\; WORKS\_ON \text{ over } SSN=ESSN\right)$

Smith-PNOS

Pno
1
2

$SSN\_PNOS \leftarrow \pi_{ESSN,PNO}(WORKS\_ON)$

SSNS

SSN
183456789
458453453

$SSNS(SSN) \leftarrow SSN\_PNOS \div Smith\_PNOS$

→ Write a Query to Retrieve Fname, Lname of all employes who works_on all the Project Located at Houston

X $R1 \leftarrow \pi_{Pnumber}(\sigma_{Plocation=Houston}(PROJECT))$

X $R2 \leftarrow R1 \;|X|\; (WORKS\_ON)$ over $Pnumber = Pno$

X $R_3 \leftarrow \Pi_{ESSN}$

$R_4 \leftarrow$

$R_1 \leftarrow \sigma_{Plocation='Houston'}(PROJECT)$

$R_2 \leftarrow \Pi_{pnumber}(R_1)$

$R_3 \leftarrow \Pi_{ESSN, PNO}(WORKS\_ON)$

$R_4 \leftarrow R_3 \div R_2$

$R_5 \leftarrow R_4 \,|X|_{ESSN=SSN}\, (EMPLOYEE)$

$R_6 \leftarrow \Pi_{Fname, Lname}(R_5)$

Natural

Outer Joins

i) Left Outer Join ⟕

ii) Right Outer Join ⟖

Full outer Joins ⟗

Chapter Outline

ER to Relational Mapping Algorithm

Step 1 : Mapping Of Regular Entity type
Step 2 : Mapping of weak Entity Type.
Step 3 : Mapping of Binary 1:1 Relation types
Step 4 : Mapping of Binary 1:N Relationship Types
Step 5 : Mapping of Binary M:N Relationship Types
Step 6 : Mapping of multivalued attributes
Step 7 : Mapping of N-ary Relationship Types.

EMPLOYEE
DEPARTMENT
PROJECT

Step 1 : Employee.

| Fname | Minit | Lname | SSN | Bdate | Address | Sex | Salary |

Department

| Name | Dnumber | Dlocation |

PROJECT

| Pname | Pnumber | Plocation |

Step 2 : Dependent

| DName | Sex | Bdate | Relationship | ESSN |

Step3: For each binary 1:1 relationship type R in the ER Schema, identify the relations S and T that correspond to entity types participating in R

3.1 Foreign key approach

Avoid Null value in single instance

Department

| Name | Dnumber | Dlocation | mgr-SSN |

2. Merged Relation option

3 Cross reference or relationship Relation option

Third alternative is to set up a third relation R for the purpose of cross referencing primary keys of two relations S and T representing the entity Types.

Step4:

| Fname | minit | Lname | SSN | Bdate | Address | sex | salary | DNO | Supa SNN |

To Avoid Redundancy include foreign key as N side

N side add foreign key

bcz of that 1:N Relations on Supervision

Step 6: Multivalue attribute A; Create a new Relation R

Step 7:

Step 8 Not Th n

Step 1: Customer

Name	CustomerID	phoneno	Address	emailID	BranchID

Branch

Branch-ID	Location	BNO	BName

Bank Staff

Staff-ID	staff-name	mobile	Email-ID	Branch-ID

Accounts

Acc-no	Acc-name	Acc-Type	Balance	Cust-ID	Branch-ID

Step 2: DEPENDENT

DName	Age	phoneno	Address	Relationship	Staff-id

Step 3:

Customer

Step 4:

Name	Customer-ID	Phoneno	Address	emailID	BranchID

Accounts

Acc-no	Acc-name	Acc-Type	Balance	Customer-ID	BranchID

Bankstaff

Staff-ID	Staff-name	mobile	EmailID	Branch-ID

Patients

Name	Patient-ID	phoneno	Address

Doctors

D-ID	name	Specilization	Phoneno	DpID

Medical Staff

S-ID	name	phoneno	DpID

Departments

DName	Dp-ID	phone

Nurses

N-ID	NName	phoneno	DpID

Management Staff

MSID	Name	phone	DpID

Caretaker

Name	Relationship	Age	phoneno	PatientID

consults

Doc-ID	Pat-ID

cont

contains_medical Staff

| M-ID | D-ID |

Chapter 8 :

SQL 99 : Schema Definition, Constraints and Queries and Views →

Foreign key (MYRSSN) References Emp
ON DELETE / UPDATE (ON) CASCADE .
set default.

Retrieval Queries in SQL

SQL relation (table) is a multi set (Something called a bag) of tuples ; It is not a Set of tuples =

Write SQL Query to Retrieve Salary of all employe

Select * (Salary) from Employee;

Select DISTINCT Salary from Employee; →
(Repeatative Value not there)

Bag - Repetition possible

Set - No " " with no order

list - With Repetition with order

General Syntax :

SELECT <attribute list>

-> Write a SQL Query to Retrieve Fname, Lname of all employe Dno=5

Select Fname, Lname from Employee where DNO=5

Select FName, Lname, Address
From EMPLOYEE, DEPARTMENT
where Dname = 'Research' AND DNUMBER = DNO

Wht SQL Query Fname, Lname who works for Project controlled by Dept 5.

Select Fname, Lname From Employee, WORKS_ON, PROJECT
where Project.Dno = 5 AND Project.Pnumber = WORKS_ON.PNO AND WORKS_ON.ESSN = Employee.SSN

Fname, Lname and Salary of all Employees who work in Dept Location in Sugarland

Select Fname, Lname, Salary From Employee, Dept-Location where
Dept-location = Sugarland AND
Dlocation.Dnumber = Employee.Dno

Write SQL Query to Retrieve employee Fname, Lname who works on project located in Houston

only 4 types.

Select Fname, Lname from Employee, Project, works on where Project.Plocation = Houston and Project.Pnumber = WORKS_ON.Pno AND WORKS_ON.ESSN = EMPLOYEE.SSN

→ Write Query For every project located in 'Stafford' list the project number, the controlling department number, and department manage last name and address and Bdate

Select Pnumber, DNum, Lname, BDate, Address From PROJECT, DEPARTMENT, EMPLOYEE where Dnum = Dnumber AND mgrSSN = SSN AND Plocation = 'Stafford'

→ Retrieve name and address of all employee who work for Research department using nested query.

Select Fname, Lname, Address from Employee where Dno IN (Select Dnumber from Department where Dname = 'Research');

→ Retrieve Fname & Lname of all employee who work for dept 4 using nested query.

(Select Fname, Lname from Employee where Dno IN (Select DNO from Employee where Dno = 4));

→ Retrieve name of each employee who has a dependent with same First name as employee.

```
Select E.Fname, E.Lname
From Employee E
where SSN IN
(Select ESSN From Dependent D
 where D.ESSN = E.SSN
 And D.Dependent_name = E.fname);
```

→ Write Query to Retrieve address of all employee who work on project number 30.

```
Select E.Address
From Employee E
(without alias work on)
where SSN IN
 ( Select ESSN From WORKS_ON W
  where W.ESSN = E.SSN
  AND W.Pno = 30) ;
```

If Pname is Newbenefits

```
Select E.Address
From Employee E
where SSN IN(or Exists )
(Select ESSN(Replace with *)
 From worksON W, Project P
AND W.Pno = P.Pnumber
AND P.Pname = 'New Benefits');
```

→ Write query to Retrieve Fname, Lname of all the male Employee who has dependent with same gender.

```
Select E.fname , E.lastname
From Employee E
where E.SSN IN
(Select ESSN
 from Dependent D
 where E.SSN = D.ESSN
 AND E.SEX = D.SEX AND E.sex=M);
```

CORRELATED NESTED QUERIES

→ Write Query to Retrieve Employee Fname, Lname who work on Project located in Houston and has more than one dependent

```
Select E.Fname, Lname From E. Employee
 where E.SSN IN
 (
  Select W.ESSN From W.WORKS_ON
   where W.Pno IN
    (Select P.Pnumber From P.PROJECT where
        P.Plocation = 'Houston')
```

```
Select Fname, Lname from Employee
where SSN IN
( Select ESSN from Dependent grouped by
    ESSN having count(*) > 1)
Intersection
Select ESSN from Project P, workON W
    where P.Plocation = Houston and
P Pnumber = W.Pno));
```

2) Write a Query to Retrieve Dependent name of Employee who is manager.

```
Select Dependent_name from Dependent D where
    SSN IN
    ( Select Mgr_SSN from Department DP
                        where
    DP.Mgr_SSN = D.Essn));
```

```
Select Dependent_name from Dependent D
where EXISTS
    ( Select * from Department DP where D.ESSN=
                    DP.Mgr_SSN) ;
```

Write Query to Retrieve Fname Lname of all Employee who have no dependent

```
Select .Fname, .Lname from .Employee
where NOT EXISTS.
(
  Select * from Dependent where
      SSN = ESSN) ;
```

Retrieve Social Security no of all emp who work on project no 1, 2 or 3

```
Select Distinct ESSN
  from
```

Retrieve the name of all employee who do not have Supervisor

```
Select Fname, Lname
From Employee
where SUPERSSN IS NULL
```

SQL uses IS OR IS NOT to Compare NULLs bcz it considers which NULL value distinct from other NULL values

Joined Relation Feature in SQL2

"Joined Relation" in the FROM clause
(theta Joint (=, <, >).

→ Select E.Fname, E.Lname, S.Fname, S.Lname
FROM Employee E S
where E.SUPERSSN = S.SSN ;

written as

Select E.Fname, E.Lname, S.Fname, S.Lname
From (Employee E Left outer JOIN
EMPLOYEE S ON E.SUPERSSN = S.SSN ;

→ Select Fname, Lname, ADDRESS
From Employee, Department
where DName = 'Research' AND DNumber = Dno;

OR

Select Fname, Lname, Address
From (Employee Join department ON
DNumber = Dno) where Dname = 'Research');

Natural join

Select Fname, Lname, Address
From (Employee Natural JOIN Department
As Dept (DName, Dno, MSSN, MSDate)
where DNAME = 'Research';

Select Pnumber, Dnum, Lname, Bdate, Address
From (Project Join Department ON
Dnum = Dnumber) JOIN
Employee ON MGRSSN = SSN))
where Plocation = 'Stafford');

Select MAX (SALARY), MIN (SALAR), AVG (SALARY)
From Employee.

Select count (*) from Employee, Department
where , DE Dno = Dnumber And
Dname = 'Research';

Each dept. Retrieve Dept no., no. of employee in dept and their average Salary

Select DNO, count (*), AVG (SALARY)
From Employee Group by DNO; same

Having Clause :

Sometimes we want to Retrieve the values of these functions for only those groups that satisfy certain conditions.

Substring Comparision

The Like comparision operator is used to compare partial strings
Reserved character '%' (or '*' in implemen^{n})
and '_' replaces a single arbitary character

→ Retrieve all employees who address is in Houston, Texas. Here the value of address attribute must contain substring 'Houston, TX' in it

```
Select Fname, Lname From Employee
where Address Like '% Houston, TX %.'
```

→
```
Select Fname, Lname
From Employee
where BDATE LIKE '195_______'
                  OR
                '195 %'
```

→ Fname, Lname of employee SSN begins with 9

```
Select Fname, Lname
From Employee
where SSN Like '9_________'
```

→ 10% is 1.1

Update Project where Plocation = '' ''

Order By

Sort the tuples

Retrieve list of employee & projects each work in ordered by employee's department & with each department Ordered alphabetically by employee list

```
SELECT DName, Lname, Fname, Pname
 From Department, Employee.
 Works_on, Project
 where Dnumber=Dno and Ssn=Essn
```

ASC
DESC

General Constraints as Assertions

Specification of more general constraints via assertions

Mechanism : CREATE ASSERSION.

Salary of employee must not be greater than Salary of manager

```
Create Assertion Salary_Constraint
check (NOT Exists (Select *
 from Employee E, Employee M, Department D
 where E.salary > M.Salary And
  E.Dno=D.Number AND D.MGRSSN=E.SSN));
```

SQL Triggers

Objective : to monitor a database and take initiate action when a condition occurs

Syntax Some As Assertion
- Event
- Insert, deleted or Update Operation
- condition
- Action

```
Create Trigger Salary_violation
before Insert or Update of
Salary, Supervisior_ssn ON Employee
For each row
when
```

Views in SQL

"Virtual table derived from other tables"
Allows limited Update Operation

SQL command : Create View

```
V1 : Create View   WORKS_ON1
     As Select     Fname, Lname, Pname, Hours
     From          Employee, Project, Works_ON
     where         Ssn = Essn And Pno = Pnumber;
```

Updating View which refer more than one table cannot update

v2 : Create View Dept-Info(Dept_name, No_of_emps, Total sal)
As Select Dname, Count(*), Sum(Salary)
From Department, Employee
where Dnumber=Dno
Group by Dname;

2) Select Fname, Lname From Works_on1;

Ethent view implementation

- View matchialization
involves physically Creating and keeping a temporary table
Statergy : incremental update

→ Update Dept_INFO Set Total_sal = 10,00,000
where Dept_name = 'Research';

UV1 : Update WORKS_ON1
Set Pname = 'Product 4'
where Lname = 'Smith' AND Fname = 'John'
And Pname = 'Product X';

a) Update works_on
SET Pno = (Select Pnumber
From Project
where Pname = 'Project-y')
where ESSn IN (Select Ssn
From Employee
whoe Lname = 'Smith' And
Fname = 'John')

AND
Pno = (Select Pnumber
From Project
where Pname = 'Product X');

b) Update Project Set Pname = 'Product Y'
where Pname = 'Product X';

Database Programming

To access a database from an application program (as opposed to interactive interfaces)

Database programming Approaches

- Embedded commands
  Database commands are embedded in a general purpose programming languages
- Library database function
  Available to host language for database calls known as an API
- A brand new, full fledged language
  Minimizes impedance mismatch

Impedance mismatch

Incompatibilities b/w a host programming language and database model
- type mismatch and incompatibilities requires a new binding for each language
- set vs record at a time processing
  "CURSOR"
- Need special iterators to loop over query results

Embedded SQL

EXEC SQL or EXEC SQL BEGIN
{

}

END-EXEC or EXEC SQL End or Semicolon

Shared Variable (:)
Represented by using Colon

SQL code is used to communicate errors/exceptions between the database and the program

Two special communication code
i) SQL code
ii) SQL State

SQL code = 0, embedded SQL executed Successfully without error
> 0 no data found
< 0 error found

SQL State,
'0000' no error

INTO clause
special program variable
Pushed into local Variable

inside embedded SQL use colon
in host programming Do not use Colon

Cursor

Cursor is needed to process multiple tuples

Fetch commands move cursor to next tuple

```
prompt ("Enter the Department Name :", dname);
EXEC SQL
    Select Dnumber into :dnumber
    from DEPARTMENT where Dname = :dname;
EXEC SQL DECLARE EMP CURSOR FOR
                    ↳ name of cursor user defined
    Select Ssn, Fname, Minit, Lname, Salary
    from Employee where Dno = :dnumber
    FOR UPDATE OF Salary;
EXEC SQL OPEN EMP; [Points to First tuple)
EXEC SQL FETCH from EMP into :ssn, :fname,
        :minit :lname, :salary
while (SQLCODE == 0) {
    printf ("Employe name is :", Fname, Minit, Lname);
    prompt ("Enter the raise amount :", raise);
    EXEC SQL
        Update EMPLOYEE
        set Salary = salary + :raise
        where CURRENT OF EMP;
```

CURSORS

Syntax:

DECLARE <cursor name> [INSENSITIVE] [SCROLL] CURSOR
[WITH HOLD] FOR <query specification>
[ORDER BY <ordering specification>]
[FOR READ ONLY | FOR UPDATE [OF <attribute list>]];

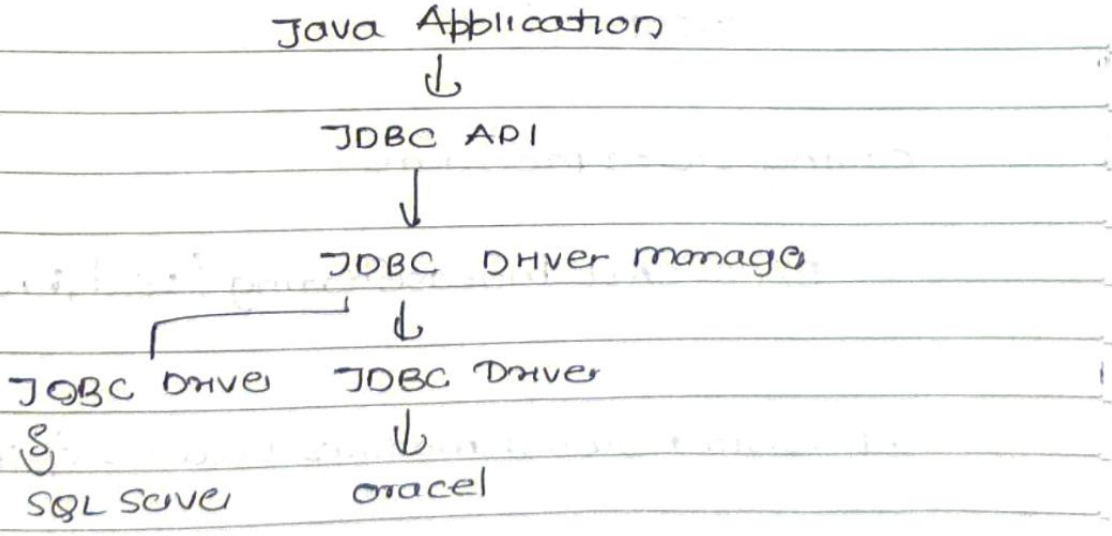
FETCH [[<fetch orientation>] FROM] <cursor name>
INTO <fetch target list>;

NEXT, PRIOR, FIRST, LAST, Absolute i· & RELATIVE i

Dynamic SQL

Update Employee set

JDBC driver Java Database connectivity Driver



Xamp
htdocs
dbmslab

php
?php

w3school

username = "root"
pw = ""

Notepad
save test2.php
save

JDBC - Software component that enables java application to interact with Database

open database Connectivity Drive (ODBC)

Type 1 : JDBC-ODBC Bridge Driver
Type 2 : JDBC Native API
Type 3 : JDBC Net pure Java
Type 4 : 100% Pure Java

Stored Procedures

Program executed through a single SQL Statement
Executed in the process space of the server.

Can encapsulate Application logic

Stored procedure can have parameters
IN
OUT
INOUT

Calling Stored procedure

EXEC CALL increaseRating (:sid, :rating);

- Persistent Stored modules because of these

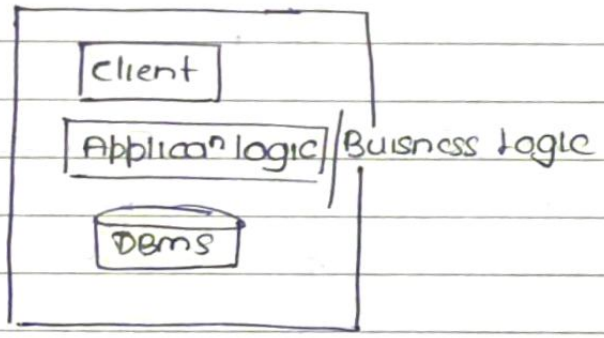
Create Procedure <procedure name> (<parameters>)
<local declaration>
<procedure body> ;

Create Function <function name>(<parameters>)
Returns <return type>
<local declarations>
<function body>

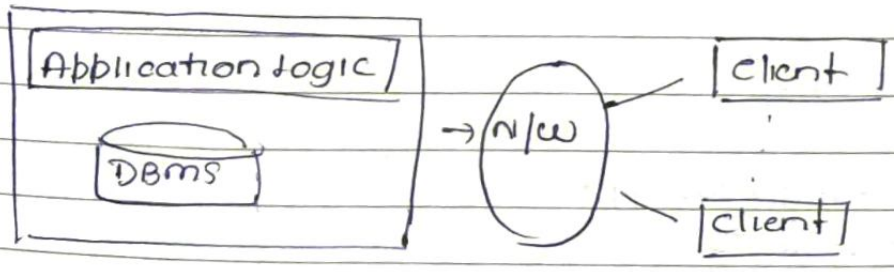
call <procedure or function name)(<argument List>

SQL/PSM.
Persistent Stored Modules.

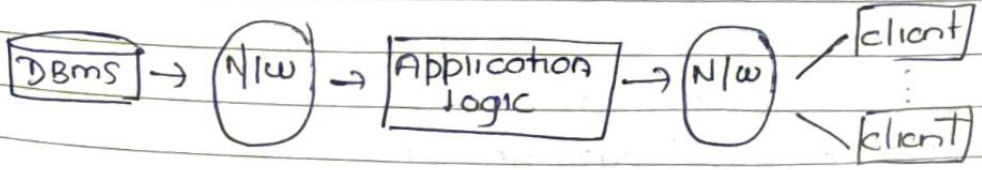
Three Tier Architecture



A Single Tier Architecture.



Two Tier / Think clients



Database web Server client

Scripting programming language

DBMS → N/W → client / Application logic

client / Application logic

Two Tier Architecture : Thick.

Server Side Programming :

Client Program (web Browser)	JavaScript, cookies
↑↓ HTTP	
Application logic (Application Server)	Servlets (Java program), JSP, XSLT
↑↓ JDBC, SQLJ	
Data Storage (Database System)	XML, Stored procedure

Advantages

DOST

1) Heterogenous System
2) Thin clients
3) Integrated Data Access
4) Scalability to Many Clients
5) Software Development Benefits.

Html, CSS, php, perl, ruby, Bootstrap

Client Side programming
Server Side Programming

CGI (Common Gateway Interface)

Disadvantage : Process created, maintain Difficult

To Overcome Servlet

Date
Page

JavaScript.

Dynamic Aspect to client side
Scripting language at client.

Servlets - Java
JSP (Java Server Pages) } Support Cookies
Php } No overload of managing process

Browser Detection
Form Validation

Browser Control: This includes opening pages in customized windows.
Example: Annoying pop-up advertisment that you seen in websites.

CGI - Does not support cookies (temporary Data Stored in System)
Process Creation take time

Pool of Servlet Created
+HTML & XML
XML - Made for Data Transfer

JSP(Servlet embedded in HTML)
Servelet Inside Java HTML is there)

Maintaining State

1) HTTP Statelessness
HTTP request Does know Anything about 1st previous request

ER diagram

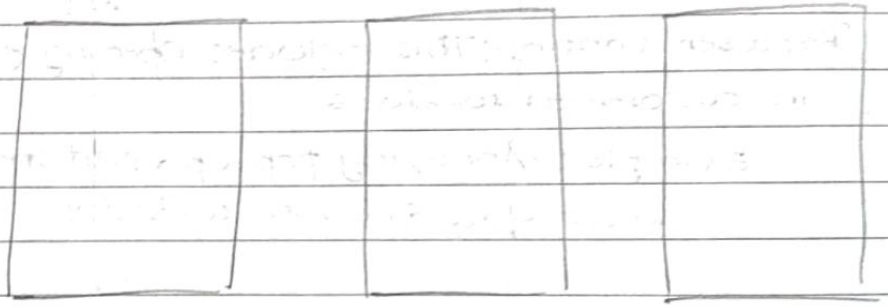
i) ER Diagram [Requirement]
ii) Schema
iii) Functional Requirement
iv) Non Function Requirement

BOOKSHOP

Sign-IN

Search

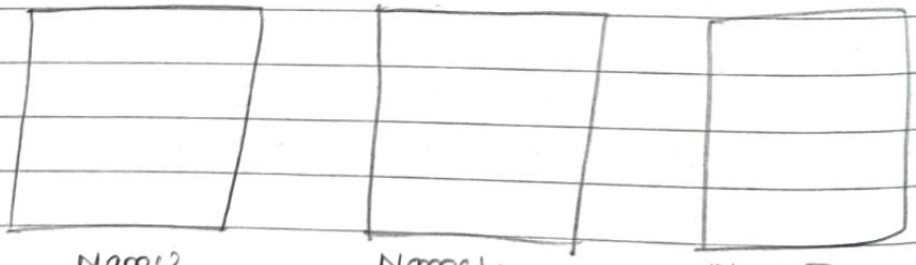
Details About us Purchase contact us



Name1 ♡
Price
Amount

Name2
Price
Amount

Name3
Price
Amount



Name2
Amount

Name4
Price

Name5
Price

ch

No Yet Registered? Sign UP

Functional Requirement	Non Functional Requirement
i) Search	Fast, Reliable
ii) Sign up	Secured
iii) Sign in	Secured
iv) Change (Update)	Robust, reliability

Module 4

chapter 10

Functional Dependencies and Normalization for Relational Database

1) Informal Design Guidelines for Relational Database
   - → Semantics of Relational attribute
   - → Redundant information in Tuple and Update Anomalies [Avoid Duplicate of Data]
   - → Null values in Tuples
   - → Spurious Tuples. [unwanted combination of Tuples]

2) Functional Dependencies (FDS).

3) Normal Forms Based on Primary keys.
   Normalization

Informal Design Guidelines

Relational database design,
grouping of attribute to form "good" relation Schemas
- → User views (logical)
- → base Relation

Guideline 1:
Design a schema that can be explained easily. [Design Guideline & Update Anamolys.] +

Guideline 2:
Information Stored Redundantly

Consider, Relation
EMP_PROJ (Emp#, Proj#, Ename, Pname, No_hours)

EMP_DEPT

| Ename | ssn | Bdate | Address | Dnumber | Dname | Dmgr-ssn |

| Ename | ssn | Bdate | Address | Dnumbo |

| Dname | Dnumbo | Dmgr-ssn |

Guideline 2.

Design a Schema that Does not Suffer from the insertion

Guideline 3:

Tuples should have few null values as possible

Spurious Tuples

"The lossless Join" property is used to guarantee meaningful results for join Operations.

| SSn | Pnumbo | Hours | Ename |

| Pname | Plocation | Pnumbo | SSN |

Two important properties of Decomposition:

Relation not in proper form, it should kept in proper form by creating Sub Relations

[combing Sub Relation should produce previous table without any unwanted combination]

a) Non-additive or losslessness of Corresponding join

b) Preservation of functional Dependencies

******** Functional Dependencies (FDs)

→ Formal measure of "goodness" of Relational Design

→ keys are used to define normal forms of Relations

A set of attributes X functionally determines a set of attributes Y if the value of X determines a unique value for Y.

$X \rightarrow Y$ holds if whenever two tuples have the same value for X, they must have same value for Y.

$$t1[X] = t2[X] \text{ then } t1[Y] = t2[Y]$$

Normalization of Relations.

The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.

Prime attributes : must be member of some candidate key.
Non prime attribute.

First Normal Form

Disallows
- compositive attributes
- multivalued attributes
- nested relation, attributes whose values for an individual tuple are non atomic

Second Normal Form

Uses Concept of FDS, primary key
→ Prime attributes, - member of prime key.
→ Full functional Dependency
a FD $Y \rightarrow Z$ where removal of any attribute from Y means FD does not hold any more

A relational Schema R is in Second Normal form (2NF) if every non prime attributes A in R is fully functionally dependent on Primary key

If key is made up of more than one attribute then it is in partial functional dependency

Third Normal Form (1)

Transitive Functional dependency

"no non prime attribute A in R is transitively dependent on primary key)

1) Consider the following relation:

CAR-SALE (Car#, Date-sold, Salesperson#, commission%, Discount-amnt)

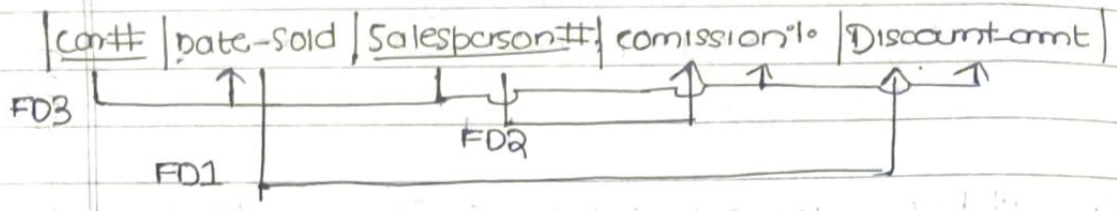
Assume that car may be sold by multiple Salesperson.

(Car#, Salesperson#) primary key

Dependencies

Date-sold → Discount-amnt

Salesman# → commission%.



Given Relation does not contain compositive or multivalued attribute. Hence Given Relation is 1NF

Prime attribute : Car#, Salesperson#

Non prime : Date-Sold, commission, Discount

commission is only dependent on salesperson

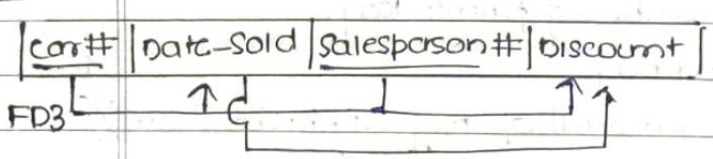
∴ Given Relation does not in second normal form

FD2 violating second normal form because non prime attribute commission is partially dependent on prime attribute of primary key Salesperson

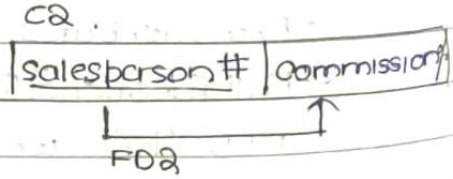
Decompose base Relation into different Relation.

Car-Sale (Car#, Date-Sold, Salesperson#, Discount)

C1



C2



Car#, Salesperson# ⟶ Date-Sold

Date-Sold → Discount

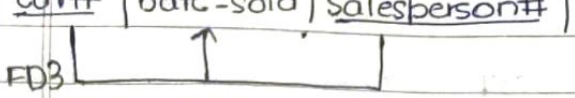
X → Y

Y → Z

Given C1 Relation is not in third normal form, the transitive dependent is found in the FD1 and FD3 [write above statement]

So Normalization of C1

C11



C12