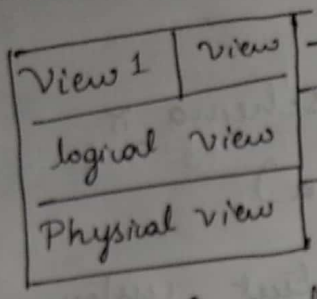
Unit-2 RELATIONAL MODEL



- View → E-R diagram → what data is stored → Relational model
- logical view → How data is stored → file structure
- Physical view

Relational model → It is made up of tables
→ also called as flat-files as each record is represented by flat structure.

Row of table → Relation instance / ~~tuple~~ tuple

column of table → Attributes

cardinality → no. of rows

degree → No. of columns

Table → A relation ~~schema~~ / schema

Eg :-

STUDENT → attribute

USN	Name.	Elective	GPA.
14EC02	Smith	DM	7
14EC08	John	DBMS	9

} cardinality

Degree

Domains :-

→ Set of allowed values of attribute

→ Usually attribute is atomic or individual

→ Eg : account No → is attributes.

account No.s → is not atomic

→ Domain is said to be atomic if all its members are atomic

Relation schema :-

→ $r(R)$

a relation r on relation schema R

Eg: Customer (Customer-schema)

Customer-schema (Customer-~~name~~ street, Customer-city, apt.no)

In general

$R(A_1, A_2, \ldots A_n)$

indicates a relation R of degree n

$A_1 \ldots$ attributes.

Ordering of tuples :

↳ No specific order

	John		
	Smith		

rows can be interchanged.

Relational model Notation :-

→ Q, R, S → Relation Name

→ q, r, s → relation states.

→ dot notation is used to specify attribute of relation

Eg : ~~Student~~ STUDENT.Name

STUDENT.USN

→ t, u, v → tuples

$t = \langle v_1, v_2, v_3 \ldots v_n \rangle$

v_i is value corresponding to attribute A_i

→ t[vi], t[i]

1/2/18

Relation model constraints :-

3 types

1] Implicit constraint → Inherent in database
→ Eg : duplication of tuples.

2] Explicit constraint → Expressed in schema, typically
By specifying in DDL
Eg : Domain constraints
constraint on null value
Key constraints

3] Semantics constraints → Enforced by application prgm.

Domain constraint → Domain should be atomic value
→ data types – Integer, string, character, Boolean

Constraints :-

Key constraint :-

For eg STUDENT Relation

Key → USN

Set of attributes with key is called superkey
→ { USN, Age, Birth date }

Some relation may have more than one key

for eg CAR Relation

License-No.	Vehicle-No.	make	model	Year

candidate keys

One key is called primary key & underlined

Integrity, Referential Integrity & Foreign Key

Entity Integrity Constraint :- EIC

↳ Primary key in a Relation should not have Null values.

Foreign key -> R_1 R_2

if attributes of R_1 is a primary attribute of R_2 then attribute is called Foreign Key

Let FK be attributes in R_1 → Foreign key

Let PK be attribute in R_2 → Primary key

if

① attribute of FK in R_1 should have same domain values as primary attribute PK in R_2

then attribute FK refrence R_2

② $t_1[FK] = t_2[PK]$

then tuple t_1 refrence t_2

EMPLOYEE

FN	MN	LN	SSN	Age	Start_date	Salary	D.No

DEPARTMENT

D.Name	D.No	Location	SSN

5/2/18

Select Operation

* Used to choose subset of tuples from a relation that satisfies condition.
* Similar to filter operation
* denoted with notation 'σ'
* It does horizontal partition

set of tuples that satisfy condition — set of tuples that does not satisfy condition

* Relation algebraic expression of Select

σ condition (R) → relation name

or.

σ select condition (R)

↓

different operations {<, >, ≤, ≥, =, ≠}

* Clauses can be combined using boolean operations such as AND, OR, NOT.

Eg :

EMPLOYEE

F.N	M.N	L.N.	B.D	Salary	Gender	Dept.No
						1 2 3 4 5

Write the attribute expression for selecting employee tuples who are working in department with dept.No : 5

$$\sigma_{\text{Dept.No}=5}(\text{EMPLOYEE})$$

Write down algebraic expression for selecting a Emp. tuple whose salary is greater than 30,000

$\sigma_{salary > 30,000}$ (EMPLOYEE)

PROJECT Operation :-

* Used to choose columns from a relation
* does vertical partition
* If duplicate tuples are there removes → duplication Elimination
* denoted with notation 'TT'
* Algebraic expression

$\Pi_{attributes\ list}$ (R)

Write down algebraic expression to not choose attributes firstname, MN, LN from the relation Employee.

$\Pi_{FN, M.N, L.N}$ (EMPLOYEE)

Write algebraic Expression to get the name of the Employees whose salary is greater than 30,000

$\Pi_{F.N,M.N,L.N}\left(\sigma_{salary > 30,000}\ (Employee)\right)$

Write the Expression for finding the Employees who are working in dept 4 & salary greater than 30,000 OR who are working in dept 5th & salary greater than 20,000

$\left(\left(\sigma_{Salary > 30,000}\ \text{AND}\ \sigma_{dept.No = 4}\ \text{Employee}\right)\right.$

OR

$\left.\left(\left(\sigma_{salary > 20,000}\ \text{AND}\ \sigma_{dept\ No = 5}\right)\ \text{Employee}\right)\right)$

(EMPLOYEE)

$\sigma_{(D.NO = 4\ AND\ Salary > 30,000)\ OR\ (D.NO = 5\ AND\ Salary > 20000)}$

Write statement, algebraic expression for finding Names of female Employees

$\Pi_{F.N, M.N, L.N}\left(\sigma_{gender = Female}\ (Employee)\right)$

Select Employee tuples whose birth date is 1-1-1980

$\sigma_{B.D = 1-1-1980}$ (EMPLOYEE)

1) Draw E-R Diagram for company database.
2) Draw E-R diagram for Banking database using conversions
3) Discuss high level conceptual data model.

Insert Operation :-

↳ Provides list of attribute values for new tuple t that can be inserted in relation table

↳ Possible violations are

* Domain constraint → Data type & attribute value may not be in domain
* Key constraint → try to insert new row in table with SSN which is already existing for other tuple
* Entity Integrity constraint
  ↳ Key should not be null value.
* Referential Integrity constraint tuple →

'x', 'y', 'z', null, 25, 30,000, 1-7-1980, 5

Eg: ~~Eg~~ EMPLOYEE

F.N	M.N	L.N	SSN	Age	Salary	B.D	D.No
			~~77799~~ 77798				

DEPARTMENT

D.Nam	D.No	MgV_SSN	Mgr start date
	1		
	2		
	5		

Delete Operation :-

↳ RIC
Refural Integrity constraint.

Eg: DELETE ~~EMPLOYEE~~ WORKS_ON tuple

ESSN	P.No	Hrs
77799	5	10

× DELETE EMPLOYEE tuple with SSN 77799
↳ not acceptable as it violates RIC as in Relation WORKS_ON we have tuple with same SSN

UPDATE Operation :-

↳ Used to update attribute values

Eg: UPDATE
→ 'x', 'y', 'z', 77799, 25, 25000, 1983, 7

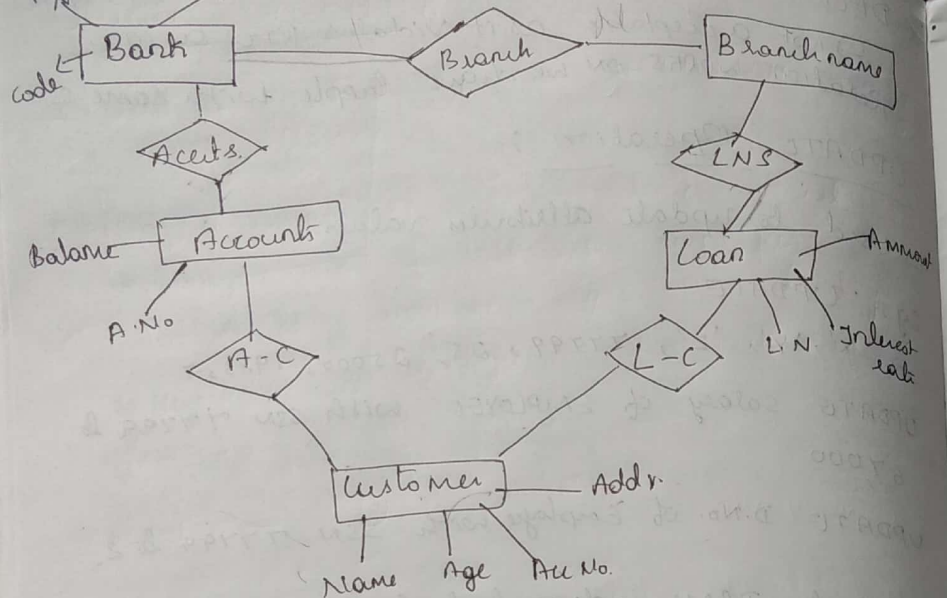
UPDATE salary of EMPLOYEE with SSN 77799 to 67,000

UPDATE D.No. of Employee with SSN 77799 to 2

1) Illustrate DBMS independent & DBMS Specific main phases of data base design with diagram

2) Develop one ER diagram

1) A Bank has many branches & large no. of customers. Customer can open diff kinds of amount with the bank. Bank keeps track of a customer by his SSN, name, addhar & phone no. There are diff. types of loans. Each identified ~~by~~ by a loan no. A customer can take out more than one type of loan & All branches can give loan.



19/2/18

RENAME Operator :-

EMPLOYEE

FN	MN	LN	Age	SSN	Salary	Gender	Super-SSN	dep

→ Eg: Retriving name of emp working in dep.5

→ $\sigma_{D.No=5}$ (EMPLOYEE)

→ $\Pi_{FN, MN, LN}$ ($\sigma_{D.N.=5}$ (EMPLOYEE))

→ EMPdep5 ⟵ $\sigma_{D.No=5}$ (EMPLOYEE)

Result ⟵ $\Pi_{F.N, M.N, LN}$ (EMPdep5)

EMPdep5

						dep.No
						5
						5
						5.

Result :

F.N	M.N	L.N

→ Result (F_N, M_N, L_N) ⟵ $\Pi_{FN,MN,LN}$ (EMPdep5)

Result :-

F_N	M_N	L_N

→ RENAME as unary operator.

Notation → ρ (Rho)

→ 3 types

$\rho_S(R)$ → Rename relation 'R' is renamed as 'S'.

$\rho_{S(B_1, B_2, \ldots B_n)}(R)$ → Sequentially renamed as $B_1, B_2 \ldots, B_n$ renames both relation & attribute.

$\rho_{(B_1, B_2, \ldots B_N)}(R)$ → only attribute will be renamed, relation R will remain same

ρ_{Temp1} ($\sigma_{D.No}$ (EMPLOYEE))

$\rho_{Temp1(F_N, M_N, LN)}$ ($\Pi_{FN, MN, LN}$ (TEMP1))

UNION OPERATOR :-

→ Used to combine the two RELATION having same attribute.

Write algebraic expression to retrive social security no.s of all the Employees who either work in dep.5 OR directly supervise a Employee who works in dept.5.

→ Notation → '∪'

$\Pi_{SSN}\left(\sigma_{dep.No=5}(EMPLOYEE)\right)$

$EMPdep5 \leftarrow \sigma_{dp.No=5}(EMPLOYEE)$

$Result_1 \leftarrow \Pi_{SSN}(EMPdep5)$

~~Result 2~~

$Result_2 \leftarrow \Pi_{super-SSN}(EMPdep5)$

$Result_{2(SSN)} \leftarrow \Pi_{super-SSN}(EMPdep5)$

$RESULT \leftarrow Result1 \cup Result2$

SSN
222333
444565
789111
919919

dep5	Super-SSN
5	919919
5	222333
5	222333

Result 1

SSN
222333
444565
789111

Result 2

SSN
919919
222333

RESULT :-

SSN
222333
444565
789111
919919

20/2/18.

INTERSECTION OPERATOR

Notation → '∩'

Eg :-

STUDENT

Name	Branch
X	ECE
Y	ECE
Z	CS
A	EEE

INSTRUCTOR

Name	Branch
C	ECE
X	ECE
B	Civil
D	Mech

STUDENT ∩ INSTRUCTOR

Name	Branch
X	ECE

MINUS OPERATION :-

STUDENT − INSTRUCTOR

=

Name	Branch
Y	ECE
Z	CS
A	EEE

INSTRUCTOR

Name	Branch
C	ECE
B	Civil
D	Mech

CARTESIAN PRODUCT OR CROSS PRODUCT.

→ Let R(A1, A2 ... An)

→ Let S(B1, B2, ... Bm)

→ Q = R × S

Q is the relation with degree n + m

→ Q(A1, A2 ... An, B1, B2 ... Bm)

If R → nR tuples

S → nS tuples

Q → nR × nS tupels

Eg

R

A	B
α	1
β	2

S

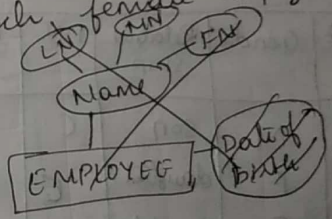
C	D	E
α	10	a
α	20	b
β	30	c
γ	40	d

2 4

A	B	C	D	E
α	1	α	10	a
α	1	α	20	b
α	1	β	30	c
α	1	γ	40	d
β	2	α	10	a
β	2	α	20	b
β	2	β	30	c
β	2	γ	40	d
β				

2 × 4 = 8

Write algebric expression to retrive a list of names of each female employee dependence



EMPLOYEE

FN	MN	LN	SSN	Salary	SuperSSN	Sex	date of birth
			22			M	
			133			F	
			34			F	
			44			M	

DEPENDENCE

ESSN	Dependent name	Gender	Relationship
22	X	M	Son
22	Y	F	daughter
34	Z	F	Daughter
44	A	F	SPOUSE

Temp1 ← $\sigma_{gendu=F}$ (EMPLOYEE)

Temp2 ← $\Pi_{FN,LN,[illegible],SSN}$ (Temp1)

Temp3 ← Temp2 × DEPENDENT

Temp2:-

FN,	MN	LS	SSN
C		D	33
E		M	34

Temp3:-

ESSN	Dependent name	Gendu	Relation ship	FN	LN	SSN	
22	X	M	son	C	D	33	
22	Y	F	daughter	E	M	34	
34	Z	F	Daughter	C	D	33	
44	A	F	Spouse	C	D	33	
22	X	M	son	E	M	34	
22	Y	F	daugh	E	M	34	
34	Z	F	daughter	E	M	34	ESSN = SSN
44	A	F	Spouse	E	M	34	

Temp4 ← $\sigma_{ESSN=SSN}$ (TEMP3)

Result ← $\Pi_{FN,LN,Dependent\ name}$ (Temp4)

21/8

JOIN OPERATION:

* It is used to join tuples of two relation into single longer tuples which satisfies the condition

→ Notation → ⋈

→ R ⋈ condition (s)

↓

operations =, ≥, ≤, <, >

→ other name theta join $A_i \Theta B_j$

Eg:- To Retrive managers name

EMPLOYEE

FN	MN	LN	SSN	
			22	
			⋮	
			33	
			44	

DEPARTMENT

Dname	D.No	Mngr-SSN	Mngr_start date
Research	1	22	
administration	4	33	
office	5	44	

$Temp_1$ ← EMPLOYEE ⋈ $_{SSN = Mngr\_SSN}$ (DEPARTMENT)

Temp1

FN	MN	LN	SSN	D.Name	D.N	Mgr-SSN
			22			22
			33			33
			44			44

(Temp1)

Result <—— $\Pi_{FN, MN, LN}$

Result

FN	MN	LN

Natural JOIN Operation

→ It is used to Join tuples of 2 ~~concat~~ Relation into single longer tuples which satisfies condition

→ notation → *

→ R * S

→ condition is that the join attribute should ~~be~~ have same name

→ If not use rename operator change name then use natural join

Eg:-

EMPLOYEE

FN	MN	LN	SSN

DEPARTMENT

D.Name	D.No	Mgr-SSN	Mgr-strt-date

Result <— ρ (D.Name, D.No. ~~Mgr-SSN~~ SSN, ~~Mgr~~ Mgr-strt-date) (DEPARTMENT)

RESULT <— RESULT * EMPLOYEE.

Result 1 <—— $\Pi_{FN, MN, LN}$ (RESULT).

Result :-

FN	MN	LN

Equijoin = join operator is = sign.

Division Operation :-

R

A	B
a_1	b_1
a_2	b_2
a_3	b_3
a_4	b_4
a_1	b_3
a_2	b_3
a_3	b_3
a_4	b_3
a_1	b_4
a_2	b_4
a_3	b_4

S

A
a_1
a_2
a_3

$T \leftarrow R \div S$

T

b_1
b_4

S

B
b_2
b_4

T

a_2
a_3

$T \leftarrow R \div S$

Query Tree:

* Algebraic operations - represents internal nodes
Relation name → represents leaf nodes

Retrieve the list of project no, controlling dept no & dept mngr, last name, Address & Birthdate for the project located in Stafford.

EMPLOYEE

FN	MN	LN	Salary	SSN	SuperSSN	Dno.

DEPARTMENT

Dname	Dnum	Mgr-SSN	Mgr_start_date
	5	44	

PROJECT:

Pname	Pno	Plocation	Dnumber
X	4	Stafford	5.

TEMP $\leftarrow$ $\sigma_{Plocation = Stafford}$ (PROJECT)

TEMP1 $\leftarrow$ TEMP $\bowtie_{Dnumber = Dnum}$ (DEPARTMENT)

TEMP1.

Pname	Pnumber	Plocation	D. Number	Dnum	mgr_SSN
			5	5	

TEMP2 $\leftarrow$ Temp1 $\bowtie_{mgr\_SSN = SSN}$ (EMPLOYEE)

RESULT $\leftarrow \Pi_{LN, Addr, Birthdate}$ (TEMP2)

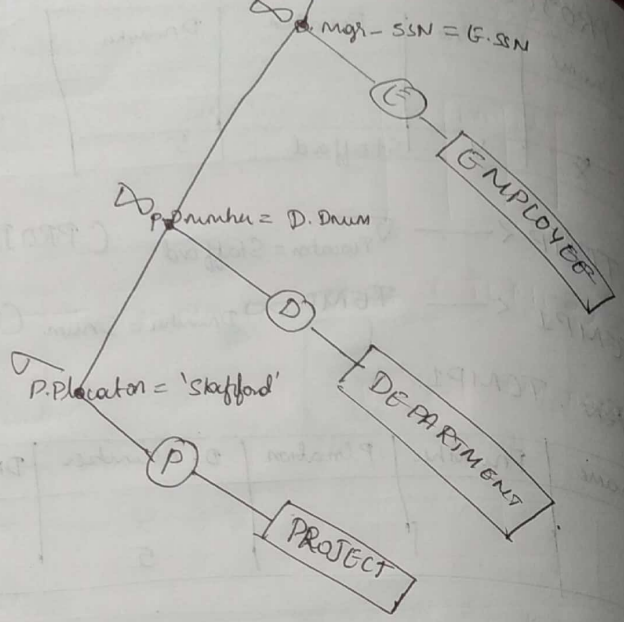
RESULT

LN	Addr	Birth date	Pname	Plocation

$\Pi_{LN, addr, B.d}\Big(\big(\big(\sigma_{Plocation = 'stafford'}(PROJECT)\big) \bowtie_{D.Number}$

$(DEPARTMENT)\big) \bowtie_{mgr\_SSN = SSN} (EMPLOYEE$

4 operations are used 1 - select
1 → project
2 → join



Generalised Projection :-

→ ~~Extended~~ Extends PROJECT Operation to include function attributes.

Eg :- EMPLOYEE

SSN	Salary	Deduction	Service

Net salary = Salary - Deduction

Bonus = 2000 × service

Tax = 0.25 × salary

Temp1 ← $\mathcal{F}_{SSN, netsalary, Bonus, Tax}$ ($\Pi_{SSN, salary, ...}$

2000×service, .25×salary (EMPLO

Aggregate functions & grouping :-

→ Count function → to count tuples & values

→ Sum, Average, Maximum, & minimum value can be calculated.

→ Notation → $\mathcal{F}$ (pronounced script F)

→ <grping attribute> $\mathcal{F}$ <function list>

D.No $\mathcal{F}$ count ssn, salary (EMPLOYEE

D.No.	count-ssn	salary
4	5	. . .
5	3	. . .
1	2	. . .

1] Write the algebraic expression to retrive the ~~state the~~ name & addres of all the employees who work for research department

26/2/18.

EMPLOYEE

Name FN	Name MN	Name LN	SSN	~~Age~~	address	salary	Gender	D.number 5

DEPARTMENT

Dname	D.number	mgr-ssn	mgr-start-date	
Reasearch Administration office.	5	. .	. . . .	

TEMP ← $\sigma_{Dname = Research}$ (DEPARTMEN

TEMP2 ← (TEMP. DEPARTMENT) $\infty_{Dnumber=Dnum}$ (EMPLOYEE)

TEMP2

Dname	Dnum	Mgr_ssn	Mgr_start_date	FN	MN	LN	Addres

RESULT ← $\Pi_{FN, MN, LN, address}$ (TEMP2)

2) Write algebraic expression to list ~~the~~ names of all the employees with two or more dependence.

EMPLOYEE

FN	MN	LN	SSN	Salary	Age	Gender	D.num	SSN
								22 23 24

DEPENDENT

ESSN	Dependent name	Relationship	D.OB
22		son	
22		daughter	
33		son	

aggregate operation.

R_1 (SSN, no.of.depen) ← $_{ESSN}\mathcal{F}_{count\ Dependentname}$ (DEPARTMENT)

R_2 ← $\sigma_{No.of\ dependence \geq 2}$ (R_1)

R_3 ← R_2 * EMPLOYEE

RESULT ← $\Pi_{FN, LN, MN, dept name}$ (R_3)

3) Write algebraic expression to retrive names of employes who have no dependent.

EMPLOYEE

FN	MN	LN	SSN	Salary	age	Gender	D.num	...
			22 23 24					

DEPENDENT

ESSN	Dependentname	Relationship	D.OB
22		son	
22		daughter	
33		son	

$T_{1(SSN)}$ ← Π_{ESSN} (DEPENDENT) or $T_1 \leftarrow \rho_{SSN}\ \Pi_{ESSN}$ (R

T_2 ← Π_{SSN} (EMPLOYEE)

T_3 ← $T_2 - T_1$

RESULT ← $\Pi_{FN, MN, LN}$ (T_3)

4) Find the names of all the employees who works on all the project controlled by dept. 5

WORKS_ON

ESSN	P.No	Hrs
22	1	
33	2	
22	3	
22	8	
22	1, 2, 3	

PROJECT

Pname	Pnumber	Plocation	Dnum
X	1		5
Y	2		5
Z	3		5
A	8		4
B			

$T \leftarrow \sigma_{dept\_num=5}$ (PROJECT)

$T_1 \leftarrow \int_{(Pno)} \Pi_{Pnumber}$ (~~PROJECT~~)

$T_2 \leftarrow \Pi_{ESSN, P.no}$ (WORKS_ON

$T_3 \leftarrow T_2 \div T_1$

T_3:

SSN
22

EMPLOYEE

FN	MN	LN	Salary	Age	SSN	D.num	⊛
					22 33 44		

~~$T \leftarrow$ Dept_num=5 (PROJECT)~~

(EMPLOYEE)

$T_4 \leftarrow T_3 \bowtie_{ESSN=SSN}$

$T_5 \leftarrow \Pi_{FN, LN, MN}$ (T_4)

or

$T_4 \leftarrow \rho_{SSN}\ T_3$

$T_5 \leftarrow T_4 * EMPLOYEE$

$T_6 \leftarrow \Pi_{FN, MN, LN}$ (T_5)

27/2/18. Unit - 3

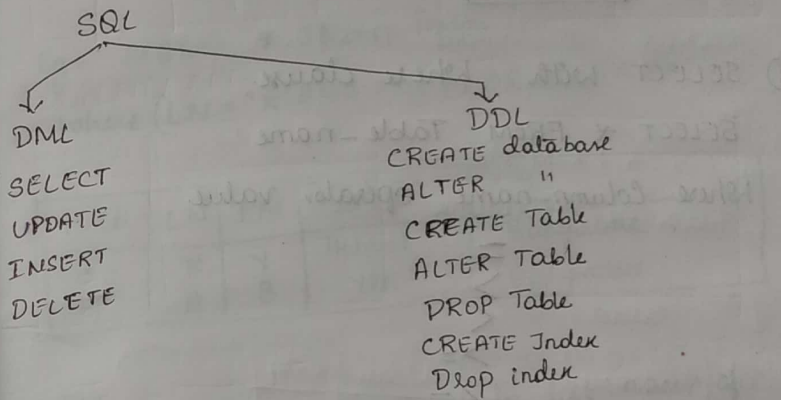
SQL (Structural Query Language)

SQL → Structural Query Language

→ Used to acess & manipulate data

→ American standard

→ Syntax to execute queries, update, Insert



Persons

P_Id	LN	FN	Addres	city
22	X	Y	Hebbal	Banglore Tumkur ~~Bangalore~~
33	A	B	Yelk	
44	C	X	RT Nagar	Bangalore

Records.

① SELECT Statement

SELECT column-name (s) FROM Table-n

SELECT LN, FN, FROM Persons

LN	FN
X	Y
A	B
C	X

SELECT * FROM Persons

↳ to select all columns

28/2

ORDER BY

→ Used to sort data in ascending order & descending order

Syntax :-

```
SELECT column_name (s)
FROM table_name
ORDER BY columnname ASC|DESC
```

OR

```
SELECT * FROM Persons
ORDER BY Lastname DESC
```

Eg :- Persons

P_Id	Lastname	FN	Addrs	City
	K - A			
	G B			
	B G			
	A K			

INSERT INTO

↳ Used to insert new record (row) in the table

Syntax

```
INSERT INTO table_name
VALUES (value1, value2, ....)
```

OR

```
INSERT INTO table_name (column1, column2,...)
VALUES (value1, value2....)
```

Persons

P_ID	LN	FN	Add	City
:				
c	abcd	cccc	Para Road	Banglore

Eg :- INSERT INTO Person

values (u, 'abcd', 'cccc', 'Park Road', 'Bangalore')
P.ID LN FN add city

INSERT INTO Person

UPDATE :-

↳ used to update or modify data

Syntax

```
UPDATE table_name
SET column1 = value, column2 = value2
WHERE some_column = some_value
```

Eg :- UPDATE Persons

SET Addrs = Park Road , city = Bangalore

WHERE Lastname = 'abcd' AND FN = 'cc

Persons

Pid	LN	FN	Addrs	City
:				
u	abcd	cccc	Park Road	Bangalore

DELETE

Syntax:

```
DELETE Column_name
FROM table_name
WHERE Some_column = some_value
```

```
DELETE *
From table_name
```

5/03/2018

TOP:

↳ used to return the number of records

Syntax:

```
SELECT TOP number/Percent Column_name
FROM table_name;
```

OR

```
SELECT TOP number/percent * FROM table_name
```

↓ all columns will be selected

Persons

P_Id	Lastname	Firstname	address	City
1	Kumar	Pranav	Hebbal	Banglore
2	Kumari	Mounitha	YLK	Bangalore
3	Nilsen	Tom	Church road	Tumkur
4	~~Gulabi~~ Gulabi	Shavan	Park road	Tumkur

Eg:- SELECT TOP 2 Lastname * From Persons;

last name
Kumari
Kumar

Eg:- SELECT TOP 2 lastname * Person;

Last name	FN	Address	City	Id
Kumar	Pranav	Hebbal	Banglur	1
Kumari	Mounitha	YLK	Banglur	2

Persons

SELECT TOP 75 percent * FROM PERSON.

PId	Lastname	First name	Address	City
1	Kumar	Pranav	Hebbal	Banglur
2	Kumari	Mounitha	YLK	Banglur
3	Nilsen	Tom	churchroad	Tumkur

LIKE:

↳ used to search pattern in a column

```
SELECT Column_name(s)
FROM table_name
WHERE Column name LIKE 'Pattern';
```

OR

```
SELECT * FROM table_name
WHERE Column name LIKE 'Pattern';
```

Eg: SELECT * FROM Persons

WHERE City LIKE 'B%'; '%g' → last 2 rows

↓ wildcards. '%mt%' last 2 values

Persons

P_Id	Lastname	Firstname	Address	City
1				Bangalu
2				Banglu

Eg:- SELECT Lastname FROM Persons
WHERE Firstname LIKE '%v';
* ending with letter v.

Persons

Lastname
Kumar

wild card	description
%	substitute for zero or more characters
_	substitute for single character.
[charlist]	any single char from charlist.
[!charlist]	any single char not from charlist
[^charlist]	" "

SELECT last name FROM Persons WHERE Firstname
LIKE [PMT]%;

Eg SELECT Lastname
FROM Person
WHERE Firstnam [!PMT]%;

Lastname
Gubbi

Eg: SELECT Firstname
FROM Persons
WHERE P_Id LIKE '4';

Fiest name
Sharan

10.

IN Operator:

↳ used to represent multiple value with WHERE Clause.

Syntax:

SELECT Column_name(s) or *
FROM table_name
WHERE Column_name IN (value1, value2,...);

Eg:- SELECT Lastname, Firstname
From Persons
WHERE Addres IN ('Hebbal', 'YLK');

Persons

Lastname	First name
Kumar	Pranav
Kanari	Mounitha

6/3/18

BETWEEN Operator:-

↳ used to select range of data b/w 2 values
↳ values may be text integer, date.

Syntax:-

SELECT Column_name(s)
FROM Table_name
WHERE column_name BETWEEN value1 AND value2;
or
SELECT * FROM table_name WHERE column_name
BETWEEN value1 AND value 2;

Eg :- SELECT * FROM Employee
WHERE Salary BETWEEN 2000 AND 3000 ;

BETWEEN :-

≥ 2000 & ≤ 3000

FN	LN	Add	Age

SELECT FN , LN Employee ~~WHERE~~ FROM Employee
WHERE Salary BETWEEN 2000 AND 3000;

FN	LN

ALIAS Operator

↳ Used to give alias name for both column & table

Syntax for column

SELECT Column_name as alias name
FROM table_name

Syntax for table

SELECT Column_name
FROM table_name as alias name

Eg : Persons

PId	FN	LN	Address	City
	Mounitha	Kumari	X	Y

Order

Or_Id	Order No.	P_Id
1	2233	1
2	4567	1
3	8902	3

Eg :-
SELECT P.FN , P.LN, Po.orders FROM Persons as P, Order as PO
WHERE P.FN = 'Mounitha'
P.LN = 'Kumari' ;

Result :-

FN	LN	Orders
Mounitha	Kumari	2233
Mounitha	Kumari	4567

8/3/18

JOIN Operator :-
↳ used to join 2 tables.

Types
Join → Return rows when there is at least one match in both tables.
left join → Return rows of left table, even if there are no matches in right table
Right join → Return rows of right table, even if there are no matches in left_table
Full join :- Return rows when there is match in one of table

Syntax of JOIN :

```
SELECT Column-names
FROM table_name1
Inner JOIN table_name2
on table_name , column name = table name 2.columnname;
```

Eg : SELECT person.lastname, person.Firstname, Orders.orderno.
FROM Persons Inner JOIN ORDERS
on Persons.P_Id = orders.P_Id

Orders

O_Id	Order No	P_Id
1	223	1
2	445	1
3	112	2
4	337	2

Persons

P_Id	lastname	Firstname	Addr.	city
1	Kumaai	Mounitha	—	—
2	Kumar	Pranav	—	—
3	Gubbi	Sharan	—	

Result :-

Firstname	lastname	Order no.
Mounitha	Kumaai	223
mounitha	Kumaai	445
Pranav	Kumar	112
Pranav	Kumar	337

Left join

LN	FN	Order no
Kum		223
Kumaai		445
Kumar		112
Kumar		337
Kumar		—
Gubbi	Sharan	

Right join

LN	FN	Or. no
K..	..	223
K..	..	445
K..	..	
Kum	-	667.
—	—	

UNION operator :-

→ combines 2 or more SELECT statements
→ Column name should be same
→ Duplicate values will be eliminated

Syntax :-

```
SELECT Column-name FROM table_name 1
        UNION
SELECT Column-name FROM table_name 2 ;-
```

EMP_India

E-Id	E-name
1	Mounitha
2	Pranav
3	Stephen

EMP_USA

E-Id	E-name
1	Kent
2	Stephen
3	Clerk

Result:

E-Name
Mounitha
Pranav
Stephen
Kent
Clerk

```
SELECT Ename FROM EMP_India
UNION
SELECT Ename FROM EMP_USA
```

If you want to allow duplicate value then use UNION ALL. then result will be

E-Name
Mounitha
Pranav
Stephen
Kent
Stephen
Clerk

12/8/18

INTO Statement :-

↳ Used to create backup copies of table

↳ Copies data from one table & insert it into another database also.

Syntax

```
SELECT *
INTO newtablename
FROM oldtable name
SELECT column(s)
INTO newtablename
FROM oldtablename
```

Eg :-

```
SELECT *
INTO PersonsBackup
FROM Persons
```

```
SELECT Lastname, Columnname
INTO PersonsBackup
FROM Persons.
```

first

```
SELECT Lastname, Firstname Columnname
INTO Persons Backup
FROM Persons
WHERE City = Bangalore
```

one)

Persons

P_Id	LN	FN	Address	City
1				Bangl
2				Bangl
3				
4				

Persons Backup

LN	FN

Result:

Persons Backup

LN	FN

Result :-

```
SELECT * INTO personsbackup
IN 'Add.mdb' FROM Persons
```

CREATE DATABASE

↳ used to create new database

Syntax

```
CREATE DataBase databasename
```

Eg.

```
CREATE DATEBASE Result
  CREATE TABLE
```

↳ used to create table

Syntax

```
CREATE TABLE table_name
(column_name1 datatype;
 column_name2 datatype,
 column_name3 datatype,
 ....
)
```

Ex

```
CREATE TABLE Persons
(
 P_Id INT
 LN varchar (255);
 FN Varchar (255),
 Addr verchar (255),
 city varchar (255),
);
```

SQL Constraints :-

1. NOTNull
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY

1. NOT NULL

↳ By default column can hold null values.

↳ If this constraint is used then it restricts that column should contain some values

Persons

P_Id	LN	FN	Addrss	city
1	X			
2	Y			
3	Z			
4	A			

one)

~~PRIMARY Key (P_Id);~~

```
CREATE TABLE Persons
(
 P_Id INT NOT Null,
 LN varchar (255) Not Null,
 FN varchar (255),
 Addr varchar (255),
 city varchar (255)
 PRIMARY Key (P_Id));
```

2. UNIQUE

↳ uniquely identifies each record in the table

↳ primary key automatically impose unique constraint.

↳ There should be only one primary key.

4. FOREIGN Key :-

Persons

P_Id	LN	FN	Addr	city
1				
2				
3				
4				

~~Old~~ ORDERS

O_Id	Order No	P_Id
01	223	2
02	445	3
03	628	4
04	910	1

↙ Foreign key for order

↳ Foreign Key refers to Primary key in another table

↳ Eg: P_Id in Orders is Foreign key where P_Id is primary key in persons.

```
CREATE TABLE Orders
(
    O_Id int NOT NULL,
    Order No. int NOT null,
    P_Id int NOT Null,
    PRIMARY KEY (O_Id),
    FOREIGN KEY (P_Id) References
    Persons (P_Id);
```

13/3

TRUNCATE TABLE:

↳ Used to detect all data in the lable & to retain lable.

Syntax:

```
TRUNCATE TABLE table-name
```

Eg.

```
TRUNCATE TABLE Persons
```

ALTER

↳ Used to add, modify, delete, data in column.

Syntax: to add column ALTER TABLE tablename. Add columnname data type.

Eg:

```
ALTER TABLE Persons
Add DateofBirth Date. to modify
```

Syntax

```
ALTER TABLE tablename
ALTER COLUMN Columnname Data type.
```

Eg:-

↓ add the DOB

PId	LN	FN	Addr	city	Date of Birth

↳ Eg: ALTER TABLE Persons
ALTER COLUMN Date of birth Year

To drop → column

Syntax

```
~~ALERT~~ ALTER TABLE tablename
DROP COLUMN column-name
```

Eg:-

```
ALTER TABLE Persons
DROP COLUMN Date of Birth
```

Result

P_Id	LN	FN	Address	city

CREATE VIEW :

↳ used the create virtual table

↳ column from two or more tables can be added in virtual table

Syntax

CREATE VIEW view_name as SELECT column_name(s) FROM table name(s)

where

Eg:-

CREATE VIEW WORKS-ON as SELECT LN, FN, Pnumber, Hrs, FROM Employee, Project works on where SSN = ESSN AND Pnumber = Pno.

WORKS ON

LN FN Pnumber Hrs

SELECT LN, FN, FROM WORKS ON ORDER BY LN

EMPLOYEE

FN	LN	SSN	Gendu	Salary

Project

Pnumber	Pname	location	Dno

Works on

Essn	Pno	Hrs

WORKS ON

LN	FN

DROP VIEW view_name

Aggregate Functions

↳ used to return single value after calculating values in column

AVG ( ) → returns avg value of column

COUNT ( ) → returns no. of values in column or number of rows in table.

MAX ( ) → returns maximum value

MIN ( ) → returns min value

FIRST ( ) → returns first value

LAST ( ) → returns last value

SUM ( ) → return total sum of values in column

↓ column name

1) AVG ( )

Syntax

SELECT AVG (columnname) FROM table_name

Eg SELECT AVG (Orderprice) FROM ORDER

SELECT AVG (order price) as avgprice FROM ORDER

avg price = 200

ORDER

O_ID	orderdate	orderprice	customer
1			X
2			Y
3			Z
4			A

14/3/18.

To find a customer that have an order price value higher than avg order price value

ORDER

O_Id Orderdate orderprice customer.

SELECT Customer from

ORDER WHERE

order price > (Select avg (order price) from Orders)

700	X
~~500~~	Y
1000	Z
2000	A
300	

avg 1000

Result > 1000

customer
Z

Syntax

Select Count (columnname) from tablename

Eg Select count (customer) as customer No from ORDER

Customer No
3

Select count (*) as No of order from order

No. Of order
4

Write SQL statement,

To count no. of orders from customer x

ORDER

O_Id	Orderdate	orderprize	customer
1			X
2			Y
3			Z
4			X
5			Y
6			Y

Select count (customer as customer from ORDER WHERE Customer='x'

Customer x
2

MAX→

SELECT MAX (Order prize) as largest order prize FROM Order

largest order prize
500

MIN

SELECT MIN (Orderprize) as smallest orderprize

Smallest order prize
25

FIRST→

SELECT FIRST (columnname) FROM table name

SLECT FIRST (customer) as FIRSTname . FROM order

First name
X

SELECT SUM (ordeprice) as totalorder
FROM Order

total order
975

SQL statement to find customer whose ordeprice is less than total sum order price

15/2/19

GROUP By Operator:-
Used with aggregate function

Syntax

SELECT Column name, aggregate function(Column name)
FROM table-name
GROUP By Column name

Eg: SELECT Customer, sum (ordeprice)
FROM Order
GROUP BY Customer

Order :-

O_Id	Order No.	order price	customer
1		300	X
2		1000	X
3		2000	Y
4		500	Y
5		700	Z

Customer	orderprice
X	1300
Y	2500
Z	700

Customer	Orderprice
X	4500
X	4500
Y	4500
Y	4500
Z	4500

Having clause :-

Used with agg function

SELECT Columnname, aggregate function (Column name) FROM table_name
GROUP By Columnname
Having aggregate function (columnname) operator value;

Eg SELECT Customer, Sum (orderprice) as orderprice,
FROM Order
GROUP By customer
Having Sum (Orderprice > 1000;

Ord_Id	ordno.	orderprice	Custom
1		300	X
2		1000	X
3		200	Y
4		700	Y
5		700	Z

Customer	orderprice
X	1300

Eg :- SELECT customer Sum (orderprice) as orderp
where customer = 'x' OR customer = 'y'
Group by customer
Having Sum (orderprice) : > 1000;

Scalar functions :-

UCASE ( ) → convert text to uppercase
LCASE ( ) → " " " lowercase
MID ( ) → Extract characters from text
LEN ( ) → Return length of text.
ROUND ( ) → Round numeric value to specified decimal
NOW ( ) → Returns current system data
FORMAT ( ) → format the data.

Persons

P_Id	LN	FN	Addr	City
1	Kumari		Vpura	Banglore
2	Kumar		Hebbal	Bangalore
3	Sharan		Yelk	

Syntax of UCASE
SELECT UCASE (Columnname) FROM tablename

Eg
SELECT UCASE (LN) as lastname FROM Persons

Lastname
KUMARI
KUMAR
SHARAN

LCASE :-
Select LCASE (Columnname) from tablename

SELECT LCASE (LN) as lastname from Persons

lastname
Kumari
Kumar
Sharan

Syntax of MID.
SELECT MID Columnname, start [length]) from Tablename,
Columnname → Name of column to extract text
start → starting position from where to extrac

char.

length = no. of char in string

Eg :-

SELECT MID(colq, 1, (4)) as Small colq

From Persons

Small colq
Jang
Bang
Tuan

SELECT MID(Colq 3,[2]) as Small colq from persons

Small colq
ng
ng
nk

Syntax of LEN :-

SELECT LEN (columnname) FROM Tablename

Eg :

SELECT LEN (Addr) as length of addressfrom Persons

length of addr
5
6
3

1) Retrieve Birthdate & address of employee whose name is John B Smith

SELECT Bdate, Address
FROM Employee
WHERE FN='John' AND MN='B' AND LN='Smith'

2) Retrieve name & address of all employee who work for research department

Employee

Lname	Fname	SSN	Address	Gender	salary	Dno

Department

Dname	Dno	mgr. SSN	mgr start date

SELECT Fname, Address
FROM EMPLOYEE, DEPARTMENT
WHERE Dname = Dno AND Dname = 'Research'

SELECT E.LN, E.FN, E.Address
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE D.Dnumber = E.Dnumber AND D.Dname = 'Research'

3) For each employee retrieve employee's LN & last name & FN & LN of his or her immediate supervisor.

Employee

FN	LN	Addr	Gender	SSN	D.Number	Super_SSN
				22		55
				33		55
				44		33
				55		55

```
SELECT E.FN, E.LN, S.LN, S.FN
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE S.SSN = E.Super_SSN.
```

4) Retrive all Employees working for department 5.

```
SELECT *
FROM EMPLOYEE
Where Dnumber = 5
```

5) Retrive salary of every Employee

```
SELECT * All Salary
FROM EMPLOYEE
```

Salary
2000
3000
2000
4000
5000

```
SELECT DISTINCT Salary
FROM EMPLOYEE
```

Salary
2000
3000
4000
5000

6) Make a list of project number for projects that involve an employee whose last name is Smith either as a worker or as a Manager of the department that controls the project.

Works on

ESSN	P.No	Hrs
22	1	
33	3	
22	1	
44	2	
55		

Project

Project number	P.Name	P.Location	D.No

```
SELECT Project number DISTINCT Pnumber
FROM Project
WHERE Pnumber IN (SELECT Pnumber FROM
EMPLOYEE, Department, Project, Department,
Employee
WHERE Dno = Dnumber
AND Mgr_SSN = SSN AND LN = 'Smith';

OR

WHERE Pnumber IN ( SELECT Pnumber
FROM WorksOn, EMPLOYEE
Where ESSN = SSN AND LN='Smith';
```