2/08/18

DBMS (Data Base Management System)

Collection of inter-related data + set of programs to access data

Applications - Banking
: Healthcare
- Tele-communications
- Sales

Book: Silber Shatz Korth

Demerits of File System

? Physical & Logical partitioning.

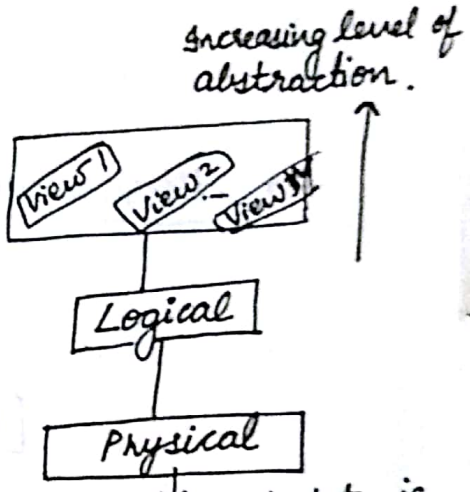
Files + App programs

① Data redundancy and inconsistency.
② Difficult to access data.
③ Atomicity problems.
④ Data isolation.
⑤ Concurrent access.
⑥ Security.

VIEW of DATA

↳ Level of abstraction

1) Physical.
2) Logical.
3) View.



DATA INDEPENDENCE

① Physical independence — change in DB instance but DB schema remains unaffected

② Logical independence — Change in DB schema but DB instance remains unaffected

Data Base instance + Data Base Schema (Design)

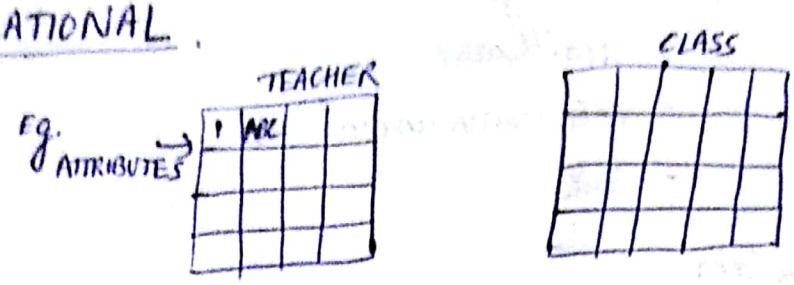
data stored in a given point of time in data base

what type of variables, relation between it...

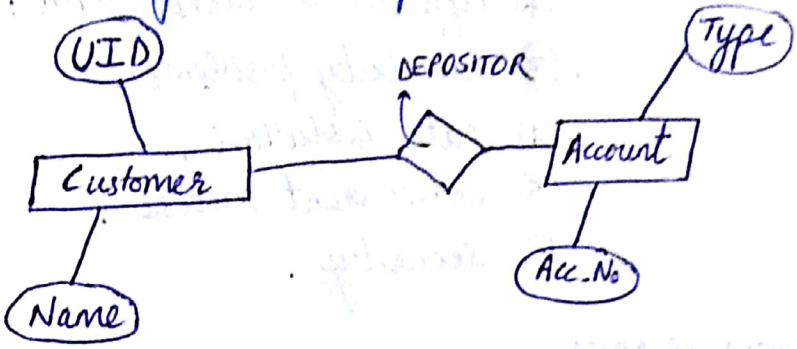
3/08/18

DATA MODELS

→ RELATIONAL.



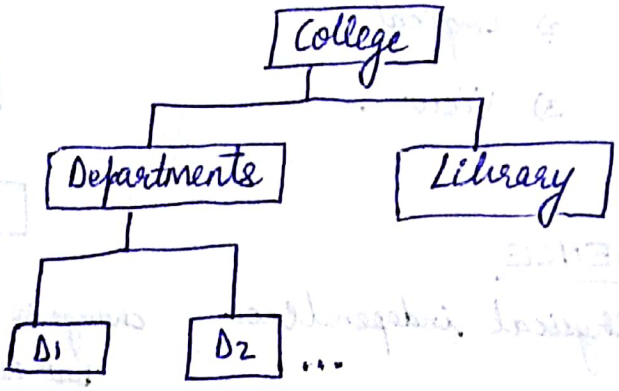
→ E R Model (Entity - Relationship)



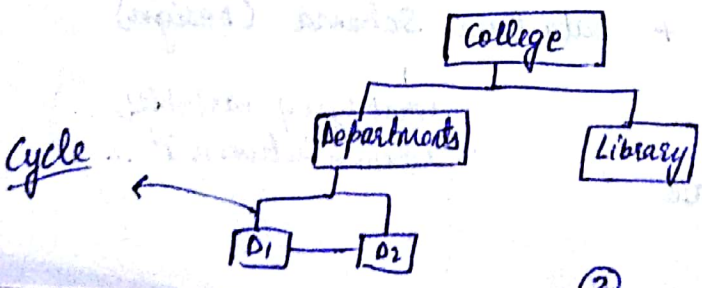
→ Heirarchial Model (Tree Based)

≡

Acyclic Connected graph



→ NETWORK MODEL (GRAPH Based)



→ OBJECT BASED MODEL

DATABASE LANGUAGES

(i) Data Definition → schema

(ii) Data Manipulation ↓ (Sub part) Query language * only for information retrieval

Relational Database (Data stored in tabular format)

Keys :

(VID, Acc-No, Name) could be same Foreg RAM, RAM

1) Super key - set of attributes used to uniquely identify all the tuples/records

MULTIPLE { Minimal super key 2) Candidate key

ONLY ONE ← 3) Primary key - set of attributes currently in use to uniquely identify all the tuples/record

RELATIONAL ALGEBRA

① Select : (σ) (table name)

σ column-name = " "

LOAN

Loan No.	Branch name	Amount	XYZ
L-1	AB	15,000	0
L-2	CD	20,000	0
L-3	EF	16,000	0

PRIMARY KEY

(Loan)

σ branch_name = "EF" ;

> ≥ < ≤

(LOAN)

σ Amount, Amount >= 15000

③

6/08/18

Select σ ROW

Project (π) COLUMN

$\pi_B(T)$

ROW/Rec

Column: All

T

A	B	C

Customer

Cust-name	Cust_city
ABC	Delhi
DEF	Agra
GH	Delhi

$\pi_{\text{cust-name}}(\sigma_{\text{cust\_city} = \text{"Delhi"}}(\text{Customer}))$; (SAME LEVEL)

Table Name

- employee (person_name, street, city); // represents primary key
- works (person_name, company-name, sal);
- company (company_name, city);
- manager (person-name, manager_name);

Q1 name of employees who don't work for X Corporation (company)

$\pi_{\text{person-name}}(\sigma_{\text{company-name} \neq \text{"X Corporation"}}(\text{works}))$;

Query 2 Name of employees having the highest pay in the given company. (AB)

H.W.

$\pi_{\text{person-name}}(\sigma_{\text{company-name} = \text{"AB"}}(\text{works}))$;

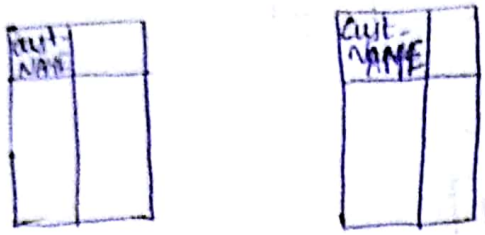
// Don't use MAX

To be finished

$$\pi_{\text{person-name}}\left(\sigma_{\text{sal} > \left(\pi_{\text{sal}}(\text{works})\right)}(\text{works})\right);$$

UNION (U)

T_1 T_2



$$\pi_{\text{cust-NAME}}(T_1) \cup \pi_{\text{cust-NAME}}(T_2)$$

INTERSECTION (∩)

$$\pi_{\text{cust-NAME}}(T_1) \cap \pi_{\text{cust-NAME}}(T_2)$$

SET DIFFERENCE (-)

$$\pi_{\text{cust-NAME}}(T_1) - \pi_{\text{cust-name}}(T_2$$

09/08/18

CARTESIAN PRODUCT

T1

A	B	C
1	2	3
4	5	6

T_2

C	D
2	8
4	6

T1 X T2

A	B	$T_1.C$	$T_2.C$	D
1	2	3	2	8
1	2	3	4	6
4	5	6	2	8
4	5	6	4	6

LOAN

l-No	b-name	amt
L1	B1	900
L2	B2	800
L3	B3	1200
L4	B3	800

BORROWER

cust_name	l-No
Ram	L1
Shyam	L2
Shyam	L3
D	L4

Query → cust. who have loan at B3

$$\Pi_{\text{cust\_name}}\left(\sigma_{\text{b-name} = \text{"B3"}}\left(\text{LOAN} \times \text{BORROWER}\right)\right);$$

H.W.

Rename

ρ (Tablename);

col = column-name

ρ (LOAN);

b-name = bank

DIVISION

→ for all

STUDENT

S-name	sub-fail-ject
A	DBMS
A	C
C	DBMS
D	C

SUBJECT

subject
DBMS
C

Query: s-name who have failed in all subjects

$\pi_{student\_name}$ (STUDENT $\div$ SUBJECT);

OR

$\pi_{student\_name}$ (STUDENT $\div$ SUBJECT);

H.W. Cartesian product & JOIN.

10/08/18

Foreign key (Primary key of some other table)

COURSE

C-id	C-name
1	A
2	B
3	C

Parent

STUDENT

SID	Name	C-id
1	ABC	1
2	BCD	1
3	AD	2
4	CD	3

Child

Foreign key may not be unique.

Integrity Constraints

1. Key based - Primary key should be unique & not
2. Domain - Range of values that a particular field can take
3. Referential - Based on foreign key

Cascaded delete. checks for dependencies of foreign key before deleting a value from it as the primary key from the parent table.

JOIN

LOAN

l_no	b_name	amt.
L1	A	100
L2	B	200
L3	C	300
L-4	D	400

BORROWER

cust_name	l_no
AB	L-1
CD	L-2
EF	L-3

Join
- 1. Inner
- 2. Outer

Natural JOIN (Inner)

$\uparrow_{cust\_name}(LOAN \bowtie BORROWER)$

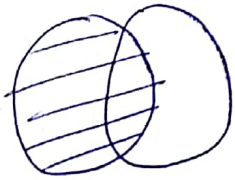
LOAN.loan_no = BORROWER.l_no

$\Downarrow$

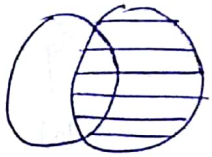
l_no	b_name	amt.	cust_name
L1	A	100	AB
L-2	B	200	CD
L-3	C	300	EF

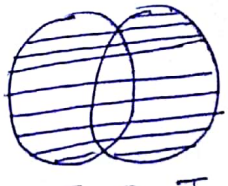
Outer JOIN

LOJ



R.O.J





F.O.J

L O J

l_no	b_name	amt.	cust_name
L-1	A	100	AB
L-2	B	200	CD
L-3	C	300	EF
L-4	D	400	-

L·O·J 

R·O·J 

F·O·J 

Eg.

employee

emp_name	street	city
C	A	E
R	B	F
S	C	G
W	D	H

ft_works

emp_name	b_name	salary
C	P	1500
R	Q	1300
G	R	1200
W	S	1000

L·O·J

emp-name	street	city	b-name	salary
C	A	E	P	1500
R	B	F	Q	1300
S	C	G	–	–
W	D	H	S	100

Natural JOIN

emp-name	street	city	b-name	salary
C	A	E	P	1500
~~bt~~				
R	B	F	Q	1300
W	D	H	S	1000

R·O·J

emp-name	street	city	b-name	salary
C	A	E	P	1500
R	B	F	Q	1300
G	–	–	R	1200
W	D	H	S	1000

F·O·J

emp-name	street	city	b-name	salary
C	A	E	P	1500
R	B	F	Q	1300
S	C	G	–	–
W	D	H	S	1000
G	–	–	R	1200

just for maintaining the order in the table

↳ Does not matter, but it matters which table is listed left.

16/08/18

* CLASS TEST: 24th Aug 2018.
FRIDAY (1, 2)
2 UNITS

SQL (Structured Query Languages)

CREATE TABLE t-name
(field1 type primary key, ...);

DROP TABLE Student/t-name // Schema + Instance removed.

DELETE from t-name // only data gets deleted

ALTER TABLE t-name

Relational SQL
σ_{col} (TABLE) ⟶ SELECT col from TABLE;

SELECT * / all from TABLE; (All rows/columns ? selected)

SELECT distinct (name) from STUDENT;

SELECT name, amt*100 from t-name;

SELECT * from t-name
where x = " "
Condition ;

T1

NAME	

T2

NAME	

SELECT T1.name, T2.name as n from T1, T2
where T1.name = " " * (n)
~~and n = " "~~)
and n like "% am %";

* For printing % use (\%) escape sequence.

Q6 Given: EMPLOYEE DATABASE

employee (employee_name, street, city)
works (employee_name, company_name, salary)
company (company_name, city)
manages (employee_name, manager_name)

underlined keys are primary keys.

(a) Select employee_name, city from employee, company
Where company_name = "First Bank Corporation" ;

(b) Select employee_name, street, city from employee, works
where company_name = "First Bank Corporation" and
salary > 10000

20/08/18

```
Select * from customer
order by cust_name;          By default ascending
                 ↓
              [ASC/DESC]
```

```
Union
select cust_name from abc,x;          // Removes duplicates
```

```
Union all
select cust_name from abc,x;          // Does not remove duplicates
```

```
-                                     // Set difference
select cust_name from abc, x;
```

```
- all
select cust_name from abc, x;         // Does not remove duplicates
```

```
select * from
emp
avg(sal) > 10000;
```

?

```
select * from emp
salary = max(salary);
```

SCHEMA

branch (b-name, b-city, assets);

customer (cust-name, c-street, c-city);

loan (loan-no, b-name, amount);

borrows (cust-name, loan-no)

account (a-no, b-name, balance)

depositor (cust-name, a-no);

(i) average balance for each customer who lives in Harison city and has atleast 3 accounts

cust-name ~~cust~~ AVG(BALANCE)

TRIAL

```
{ select ~~&~~ from account, customer, depositor
  where c-city = "Harison city" and COUNT (ac NO) >= 3
```

depositor. account. // cannot avoided as only account have has balance

SOLUTION:

```
select cust-name, AVG(balance)
from account, customer, depositor
where c-city = "Harison City" and
COUNT (acc-no) >= 3 ;
cust.c-name = dep.cust-name
and act.a-no = dep.a-no and cust-city = "Har" ;
{ group by cust-name;
{ having count ( distinct (a-no) >= 3 );
```

Note *

→ For joining, there ought to be common attributes

→ Join only provides a view and no original schema is changed.

23/08/18

→ group by → order by

Roll No.	Name	City	Marks
1	A	Delhi	80
2	B	Mumbai	85
3	C	Delhi	95
4	D	Delhi	90
5	E	Mumbai	90

CASE 3

```
select city, Avg(marks)
from student
group by city      /order by city;/
having city = "Delhi" ;
```

CASE 1 — CASE 2 — CASE 4

JOIN :

Emp

Emp_id	Name	Dept_id
1	A	1
2	B	1
3	C	2
4	D	2
5	E	3

Dept

Dept_id	Dept_name
1	COE
2	IT
3	ECE

```
select emp.name, dept.dept_name
from Emp, dept
where Emp.dept_id = Dept_id
      and Dept.Dept_name = "COE";
```

* common column is required to join two tables

Name	Dept_name
A	COE
B	COE

→ output

CARTESIAN PRODUCT

$A \times B$

Rows = $R_1 \times R_2$

Column = $C_1 + C_2$

where, $A\ (R_1, C_1)$

$B\ (R_2, C_2)$

same, done in Relational Algebra.

NATURAL JOIN

```
select Emp.Name, Dept.Dept-name
from Emp natural join Dept;
```

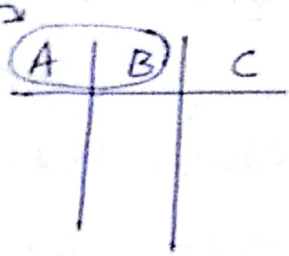
→ same output as in previous page.

EQUI JOIN

```
select Emp.Name, Dept.Dept-name
from Emp equi join Dept;
```

29/10/18

create view as
select A,B
from table;



update table_name
set bal = bal * 0.5;

OR

update table_name
set bal = case
when bal = 100 then bal * 0.5
else bal * 0.7
end;

A inner join B
on A.id = B.id;

- END of SQL
left Advanced SQL
(3 TOPICS)
till 4.3 [Do it yourself]

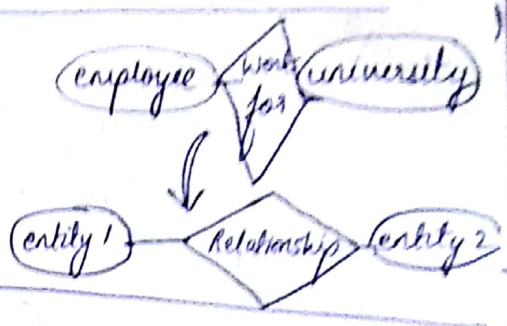
III Database Design

- Data requirements
- Program
- Design

ER Model → 1. Entity eg. employee [Salary] [Name] [Age]

2. Attribute eg. (1.., 2.. and 3..)

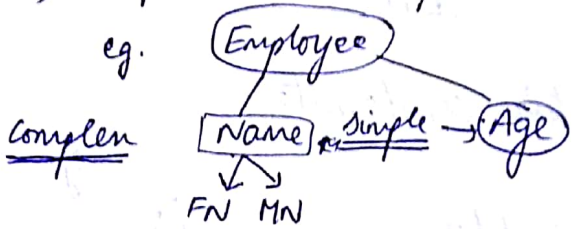
3. Relationship



Entity Set : employee
Ram (collection of entities)
Shyam

ATTRIBUTES : A) Simple vs Composite

eg.



B) Single- vs Multi-valued

eg phone numbers — having different values
Multi-valued

C) Stored vs Derived

PRACTICE : student (s_id, name)

registered (s_id, c_id)

Required: s_id, name along with total No. of registered courses

Query → select s_id, name, count(registered.~~count~~(c_id)) as No of regcourses from student inner join on registered.sid = student.sid group by s_id ;

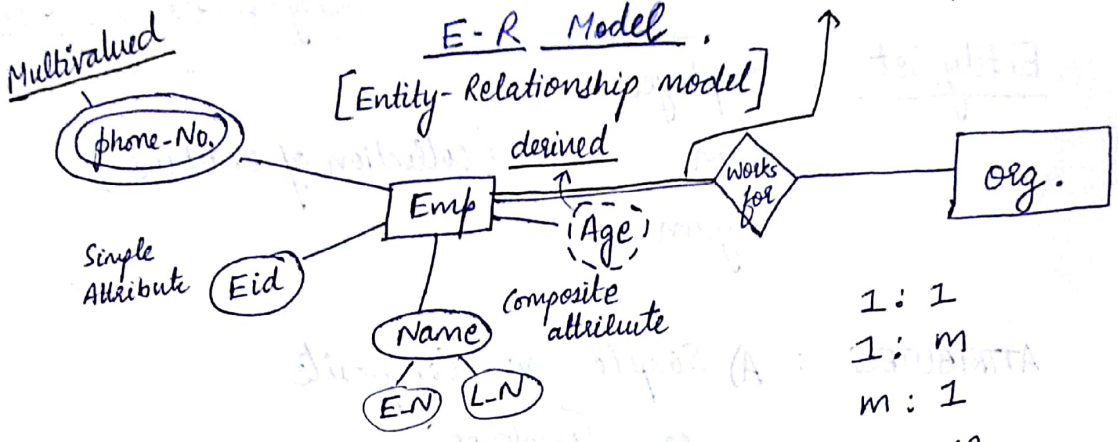
06/09/18

Mid-Term Syllabus

→ First three units

E-R Model.

[Entity-Relationship model]



1 : 1

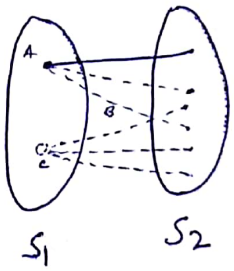
1 : m

m : 1

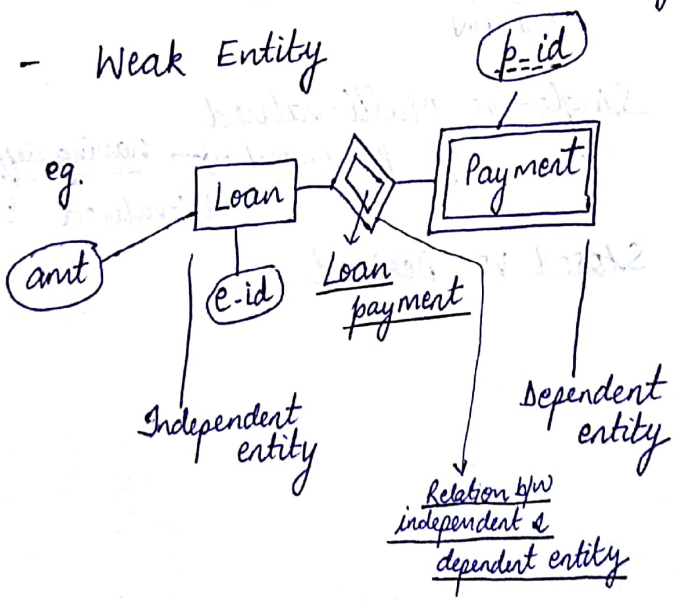
m : m

- participation constraints

eg.



- Weak Entity



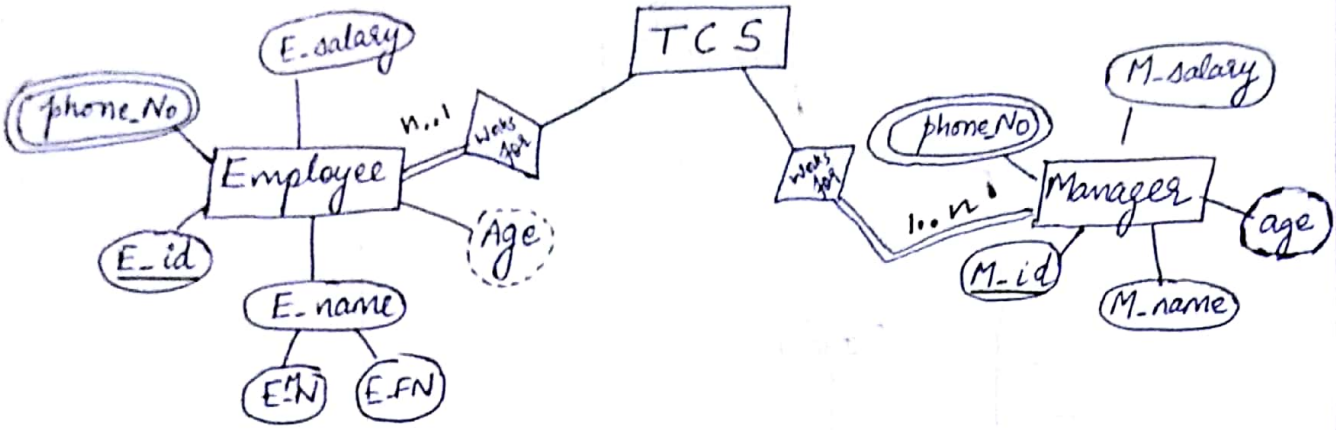
* payment is totally dependent on loan.

- primary key (xyz)

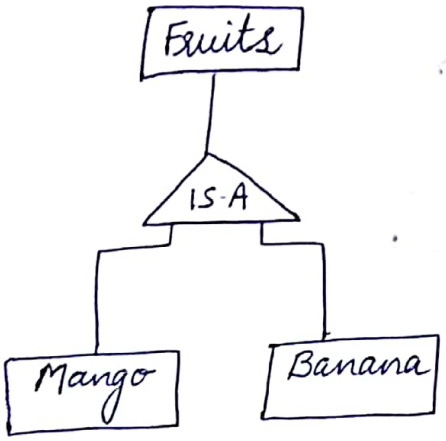
- weak_id (pqr)

(p_key + weak_id) used for uniquely identify each payment.
↓
Independent

Eg. TCS
↓
Employee, Managers, Salary management module



SPECIALIZATION



↓ Top to Bottom approach

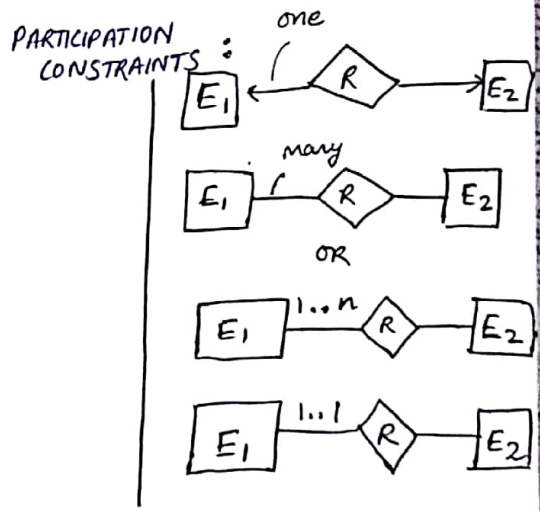
* Hand written file of Q01 & Q02 to be submitted on practical period

H.W./Assignment
→ E-R diagram of Bank database
* eg. BANK from book
* Draw using pencil neatly during examination

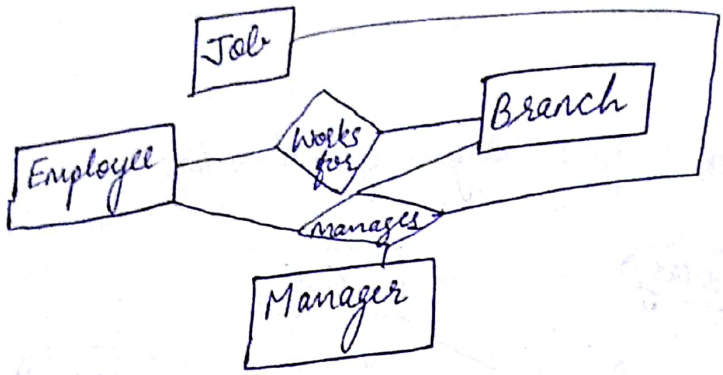
* Generalization is opposite of specialization

↑ Bottom to up approach.

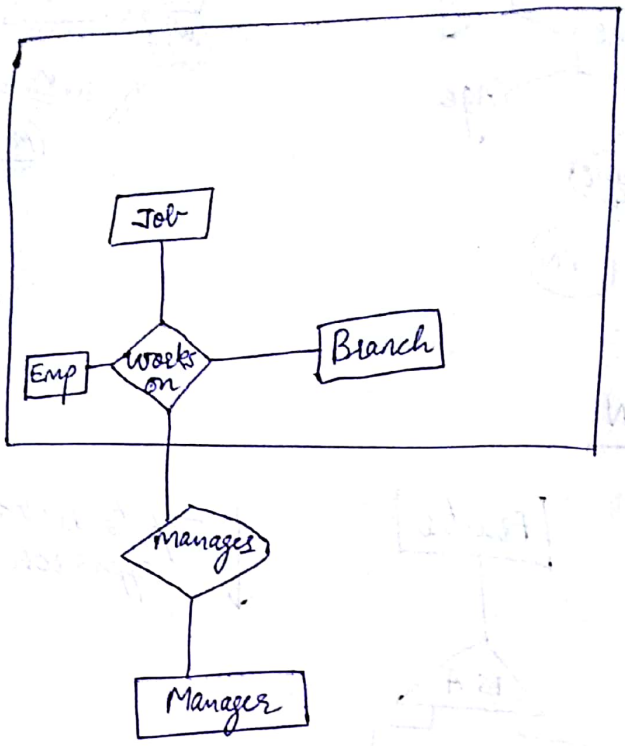
PARTICIPATION CONSTRAINTS :



Aggregation



using aggregation



07/09/18

Mid-Sem Syllabus

I Intro of DB system
- File vs DBMS system
- Level of Abst
- Data models & comparison Hierarchial, network

Ind.

II Relational Data model
- Relational DB
- Relational algebra
- Integrity constraints
- Enforcing " "
- Rel algebra & calculus
- SQL

III DB design
- Top-Down Approach (E-R Model)
- Participation constraints
- Specialization Generalization Aggregation
- Normalization
  Normal forms based on primary key till (BCNF)
- Transformation of ER diagram to Relational DB Model

→ 10Q from each unit subdivided into 2 parts each

7/09/18

** Functional Dependencies

α	β
a	1
b	2
c	3
d	4

* one to many : Not a function

if $(x_1 = x_2)$
$\Rightarrow f(x_1) \neq f($

$fd : \alpha \rightarrow \beta$

$f(\alpha) \rightarrow \beta$

if ($t_1[\alpha] = t_2[\alpha]$ and $t_1[\beta] \neq t_2[\beta]$) (tuple!) then functional dependency does not exist.

eg.

α	β
a	1
b	1
c	2

(many to 1) ✓ VALID

Types of Functional Dependencies

→ $\alpha \rightarrow \beta$

$\beta \subseteq \alpha$

A) Trivial Dependency

eg. $AB \rightarrow A$

→ $\beta \not\subseteq \alpha$

$A \rightarrow B$

B) Non-Trivial Dependency

eg.

A	B	C	D	E
a	2	3	4	5
2	a	3	4	5
a	2	3	6	5
a	2	3	6	6

Functional Dependency

a) $A \rightarrow BC$ ✓

b) $DE \rightarrow C$ ✓

c) $C \rightarrow DE$ ✗ (3,4 5 / 3,6 5)

d) $BC \rightarrow A$ ✓

Closure

$A \rightarrow B$

$B \rightarrow C$

Symbol Notation: $\{A\}^+$ is $\{A\ B\ C\}$

$A \rightarrow C$

Q. $R(A\ B\ C\ D\ E\ F)$

$A \rightarrow B$

$C \rightarrow DE$

$AC \rightarrow F$

$D \rightarrow AF$

$E \rightarrow CF$

Find $\{AC\}^+$ ie closure of AC

Ans: $\{B\ D\ E\ F\ A\ F\ C\}$

Hence, AC is the primary key

candidate key to become the

* Minimal super key is a candidate key

AXIOM — Any statement that is said/considered to be always true.

Primary

① Transitivity

$X \to Y$
$Y \to Z$
$X \to Z$

② Reflexive

$Y \subseteq X$
$X \to Y$

③ Augmentation

$X \to Y$
$XZ \to YZ$

Secondary

* ① Pseudo Transitivity

$X \to Y$
$WY \to Z$
$\Rightarrow WX \to Z$

② UNION

$X \to Y$
$X \to Z$
$X \to YZ$

③ Decomposition

$X \to YZ$
$X \to Y$
$X \to Z$

④ Composition

$X \to Y$
$Z \to V$
$XZ \to YV$

eg. for pseudo transitivity

X	Y	W	Z
1	1	a	1
2	2	b	p
3	3	c	q
4	4	d	r

10/09/18

NORMALIZATION

STUDENT

NAME	S_ID	D_ID	HOD_NAME	ADDRESS
A	1	1	AB	—
B	2	1	AB	—

* process of removing redundancy from a table.

two tables:

STUDENT ✓ DEPARTMENT ✓

NORMAL FORMS :

① FIRST NORMAL FORM : Tuples / Values — attributes

(1NF) * ATOMIC ^ (only one valued attribute)

↳ X Multivalued attributes

S_ID	NAME	COURSE_OPTED
1	A	A, B, C
2	B	A, B

↓

S_ID	NAME	COURSE_OPTED
1	A	A
1	A	B
1	A	C
2	B	A
2	B	B

Attributes are of two types :

A) PRIME → * Part of Candidate key

B) NON-PRIME (Which are not prime)

for eg. R (A B C)

$AB \rightarrow C$

$\{AB\}^+ = \{A, B, C\}$

Prime = A, B

N-Prime = C

PARTIAL DEPENDENCY

eg: R (A B C)

$AB \rightarrow C$

$B \rightarrow C$ [Partial Dependency]

eg: R (A B C D)

1:→ AB → Candidate key

P = A, B

N-P = C, D

$AB \rightarrow C$

$B \rightarrow C$ partial dependency

$C \rightarrow D$

(NPrime N Prime)

② 2 SECOND-NORMAL FORM (2NF)

* does not allow even the presence of partial dependency.

eg. for 1:→

A	B	C	D

A	D

B	C

③ THIRD NORMAL FORM (3NF)

eg. R (A B C D)

2:→ AB → C
C → D

1NF ✓
2NF ✓

but Third Normal Form ✗

* Transitive dependency should not exist
* Partial dependency should also not exist

→ Primary key should be NOT NULL

A	B	C	D
1	2	3	4
2	3	4	6
1	3	5	7

For 2:→ R_1 (A B C)

R_2 (C D)

④ BCNF (BINARY CODED NORMAL FORM)

$\alpha \longrightarrow \beta$

α → P / N·P

Earlier conditions were applied only on α

These cases should not exist ✗ (α → P/N·P $\longrightarrow \beta_{prime}$)

eg. 3NF

AB → C
D → B

(N·P) (P)

This should not exist for BCNF

Solution $R(ABC)$

$R(D, B)$

Q1 $R(A\ B\ C\ D\ E\ F)$

$AB \rightarrow C$

$DC \rightarrow AE$

$E \rightarrow F$

1) Find all possible candidate keys

(i) ABD

(ii) BDC

2) Prime and Non-Prime segregation

Prime → A, B, C and D

Non-prime → E, F

3) Determine Normal Forms

1NF ✓

2NF ✗ ∵ $CD \rightarrow A$

$(CD \rightarrow E)$ ✗ NA allowed

↓

N.Prime

$R_1(A, B\ C\ E\ F)$

$R_2(A\ C\ D\ E)$

** partial key → Prime is allowed in 2NF

but partial key → N.Prime is not allowed in 2NF

also, NP → NP is allowed in 2NF

Q.2 R (A B C D)

$AB \rightarrow CD$

$D \rightarrow A$

$BC \rightarrow DE$

1) Find all possible candidate keys

AB (BD)

(BC), complete key

2) Prime and Non-prime segregation

Prime → A, B, C, D

Non-Prime → ~~Ø~~ E

1NF ✓

2NF ✓

* Not a partial key — (BC) → E ✓ allowed.

[~~A~~ BC is not a partial key, rather it is a full fledged key]

3NF ✓

BCNF ✗

(D → A)

↓

PRIME

[NP → P ✗]

partial key → NP ✗

Transitivity ✗

X → Y prime [Exception for BCNF]

↓

super key

Q3 Emp (emp_id, emp_nationality, emp_dept, dept_type, dept_no_of_emp)

FD

(i) (emp_id → {emp_nationality})

(ii) (emp_dept → {dept_type, dept_no_of_emp})

1) {emp_id, emp_dept} CANDIDATE KEY

2) Prime - emp_id, emp_dept

Non Prime - emp_nationality, dept_type, dept_no_of_emp

3) 1NF ✓

2NF ✗ ∵[partial key → Non-prime]

4) Conversion to BCNF

R1 (emp_id, emp_nationality) } data loss

R2 (emp_dept, dept_type, dept_no_of_emp)

* R3 (emp_id, emp_dept)

→ Conversion of ER diagrams to Relational tables

→ Read yourself

→ NOT in exam coming!