Sid	pid	fid
S1	p1	f1
S1	p2	f2
S2	p2	f2
S2	p3	f1
S3	p1	f1
S3	p2	f2
S3	p3	f1
S4	p1	f1
S4	p3	f1

→ R

→ Retrieve sid of students who are working on both 'p1' and 'p2'.

RE ← R

Temp1 ← R × RE

Temp2 ← σ R.sid = RE.sid and R.pid = p1 and R.pid = p2 (Temp1)

Temp3 ← π (R.pid, RE.pid) (Temp2)

(To check if correct)

$T_1 \leftarrow \rho_A(R) \times \rho_B(R)$

$T_2 \leftarrow \sigma_{A.pid = 'p1' \text{ and } B.pid = 'p2' \text{ and } A.sid = B.sid}(T_1)$

$T_3 \leftarrow \pi_{sid}(T_2)$

Union, Intersection, Division:

- The no. of attributes should be same.
- Domain of corresponding attributes should be same.

R ∪ S

(a b) (c d)

Same domain (a → c)
(b → c)

For previous query,

$T_1 \leftarrow \pi_{sid}(\sigma_{pid = 'p1'}(R))$

$T_2 \leftarrow \pi_{sid}(\sigma_{pid = 'p2'}(R))$

$T_3 \leftarrow T_1 \cap T_2$

- Retrieve all sid's who are working on all of the projects on which S1 is working.

$T_1 \leftarrow \pi_{pid}(\sigma_{sid = 'S1'}(R))$

~~$T_2 \leftarrow \pi_{sid,pid}(R)$~~

$T_2 \leftarrow T_2 \times R$

$T_3 \leftarrow \pi_{sid}(\sigma_{T_2.pid = R.pid}(T_2))$ — wrong as only gives sid who are working on p1 or p2

$T_1 \leftarrow \pi_{pid}(\sigma_{sid = 'S1'}(R))$

$T_2 \leftarrow \pi_{sid,pid}(R)$

Result $\leftarrow T_2 / T_1$ (Division Operator) (÷)

→ Retrieve list of all student who are working on [all] projects mentored by 'f1'.

$T_1 \leftarrow \pi_{pid}(\sigma_{fid = 'f1'}(R))$

$T_2 \leftarrow \pi_{sid,pid}(R)$

Result $\leftarrow T_2 / T_1$

→ Count the no. of student working on 'p1'.

→ Find total no. of hours per week that S1 spends on projects.

Aggregate Opr (ℱ):

Student

sid	lab	duo	TA amt
1	CP	d1	4000
5	DSA	d1	5000
9	DBMS	d2	3000
12	CP	d2	4000
24	DSA	d2	4000
37	CP	d3	1000

sum()
avg()
count()
max()
min()

average
3166.6

→ $\mathcal{F}_{\text{average TAamt}}$ (Student)
(function attribute) (relation name)

→ $\mathcal{F}_{\text{max TAamt, min TAamt, average TAamt}}$ (Student)

max	min	average
5000	1000	3166.6

→ Max and min of department d2.

$\mathcal{F}_{\text{max TAamt, min TAamt}}$ ($\sigma_{\text{duo = 'd2'}}$ (Student))

→ avg TA amt per department

dno	avgamt
d1	4500
d2	3000
d3	1000

$\mathcal{F}$ dno, average TA amt (Student)

grouping basis → dno to show in result

→ Retrieve no. of TA's in each lab.

$_{lab}\mathcal{F}_{count=sid,\ lab}$ (Student)

or

$_{lab}\mathcal{F}_{count=lab,\ lab}$ (Student)

→ department wise find no. of students in each lab.

// Groups within groups

$_{dno,\ lab}\mathcal{F}_{count=sid,\ lab,\ dno}$ (Student)

whatever attributes (extra) we write here, need to be used in grouping on left.

SQL query:

(3) select dno, lab, count(sid)
(1) from student
(2) Group by dno, lab;

If where clause, then, it would execute (2).

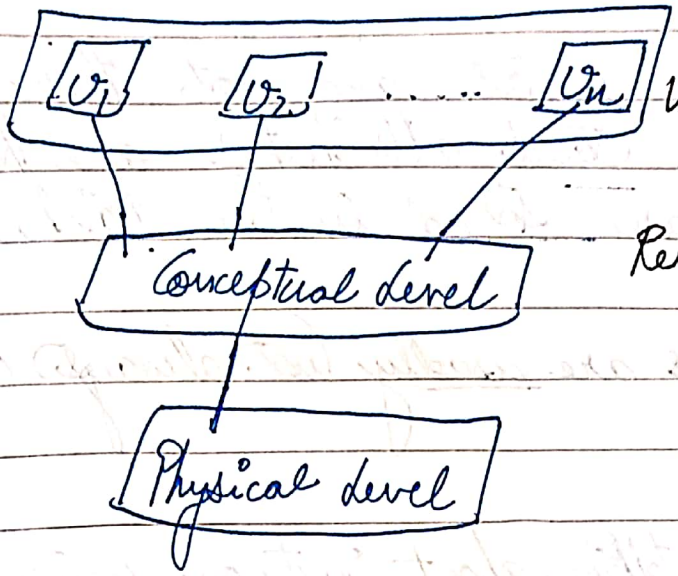
Note: 'where' clause cannot have aggregate functions, we use 'having' clause for this.

Eg: having count(sid) > 1.
(executed before select)

Ex: Write SQL query to determine avg TA amt paid to each department. O/P only those rows where avg > 2000.

```
select dno, average(TAamt)
from student
Group by dno
having average(TAamt) > 2000 ;
```

Ch - 4



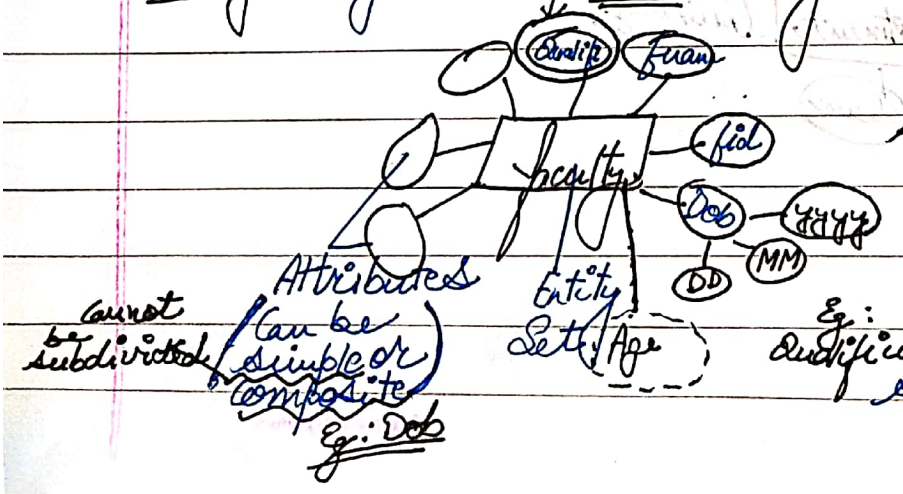
Relational Model
E-R diagram

Three schema architecture of DBMS

→ Data Independence (changes at conceptual does not affect view level & similarly changes at physical level doesn't affects conceptual level.)

Entity Relationship Diagram

Entity : Object Entity Set / Type : class



Attributes can be (Eid)
same value at all times
Single valued or multivalued, can be
Key attributes, can be
stored or derived.

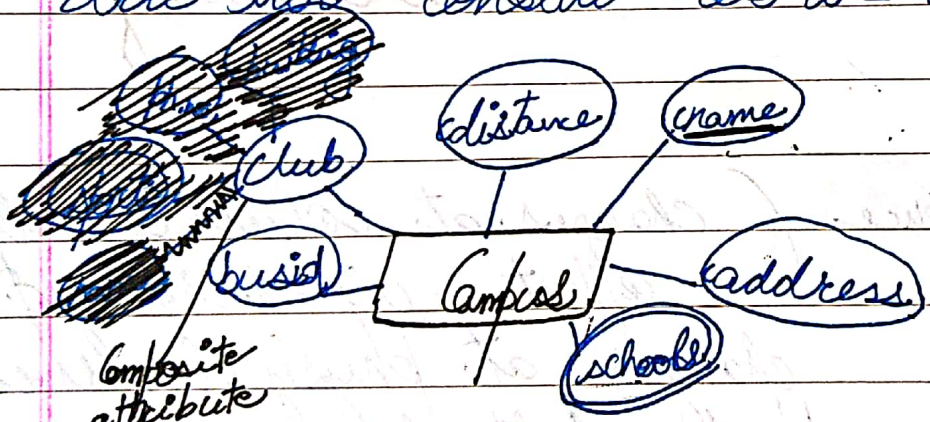
Eg: Age (Current date - do

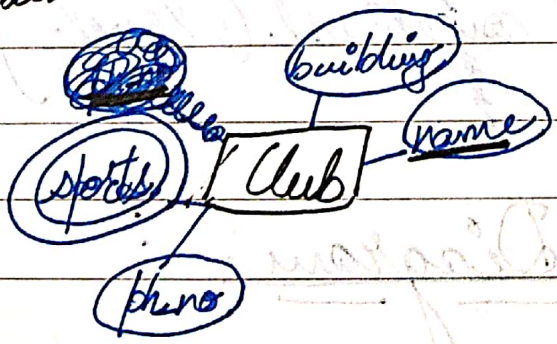
→ We use simple or derived, depends on the queries of client.

→ Derived attributes are not there in the database. Stored attributes are those whose values are stored in the database.

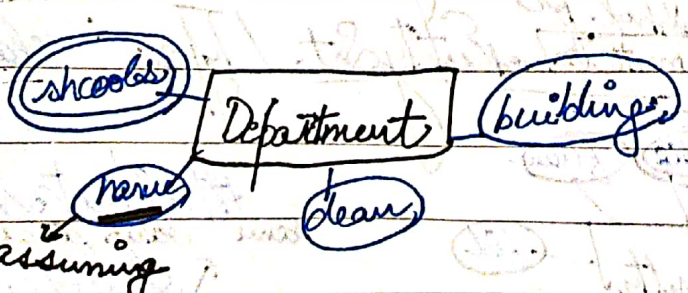
Note: Nouns are usually but not always entity or attributes.

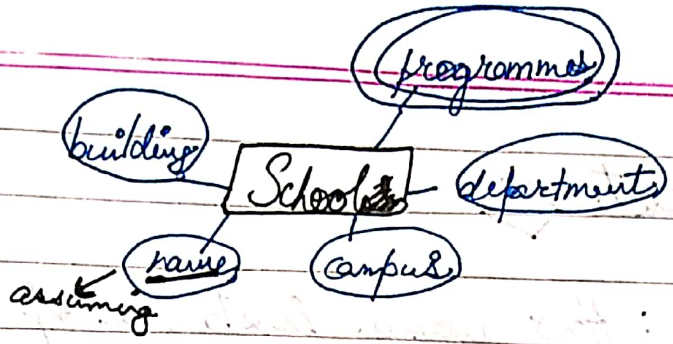
We are talking about just one university, so we will not consider it as an entity set.

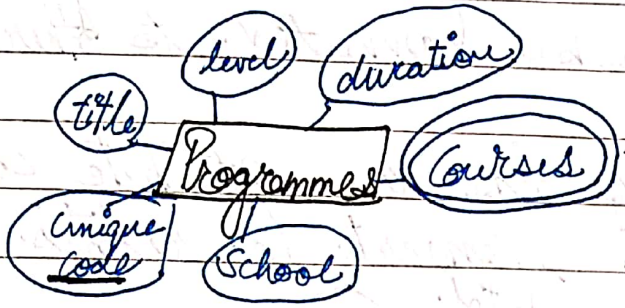


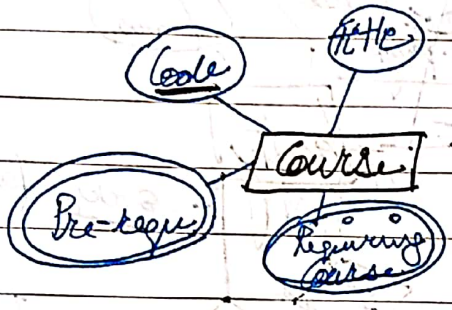


Assuming name is unique.

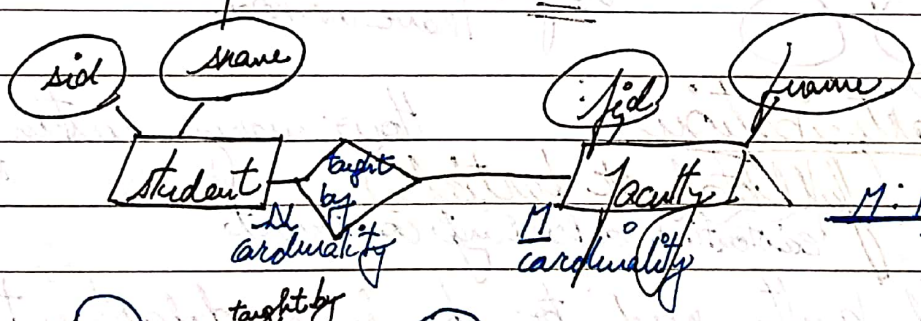


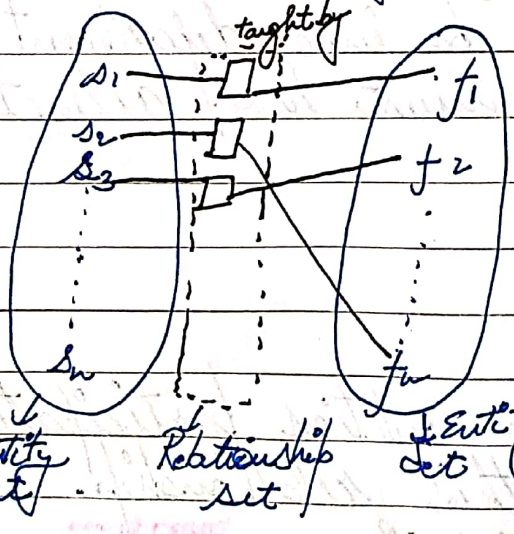






Relationships





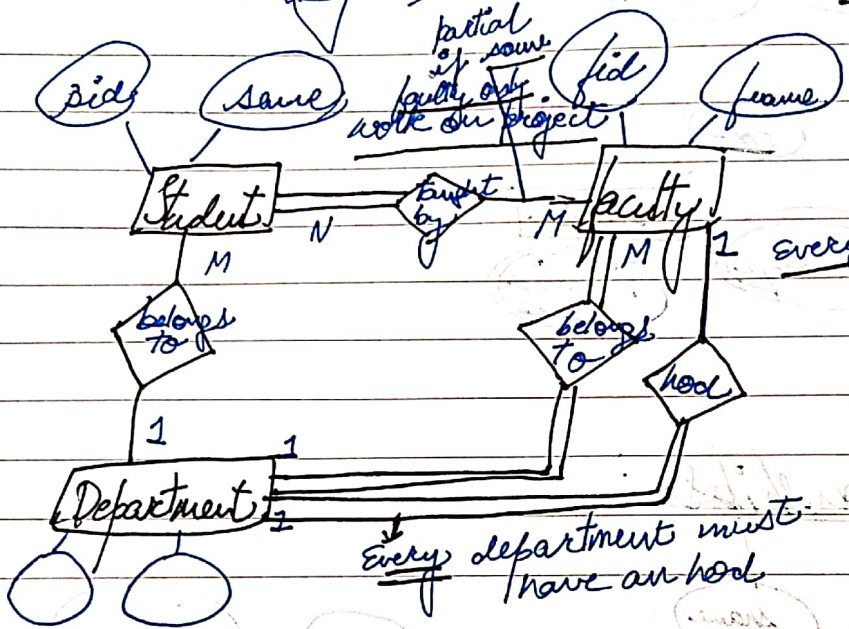
1-1 relation

Can also be one-many, many-one and many-many.

Relationship Set ◇

* ↳ Cardinality : How many instances of one set can be connected to other set.

When a entity has an attribute which is another entity, convert to relation.



1 faculty → 1 department
1 department → M faculty

* Participation : How many instances need to participate?
(Full/Total or partial)
existential depending constraint

All faculty need to be in a department and every department must have atleast one faculty.

both are examples are total participation.

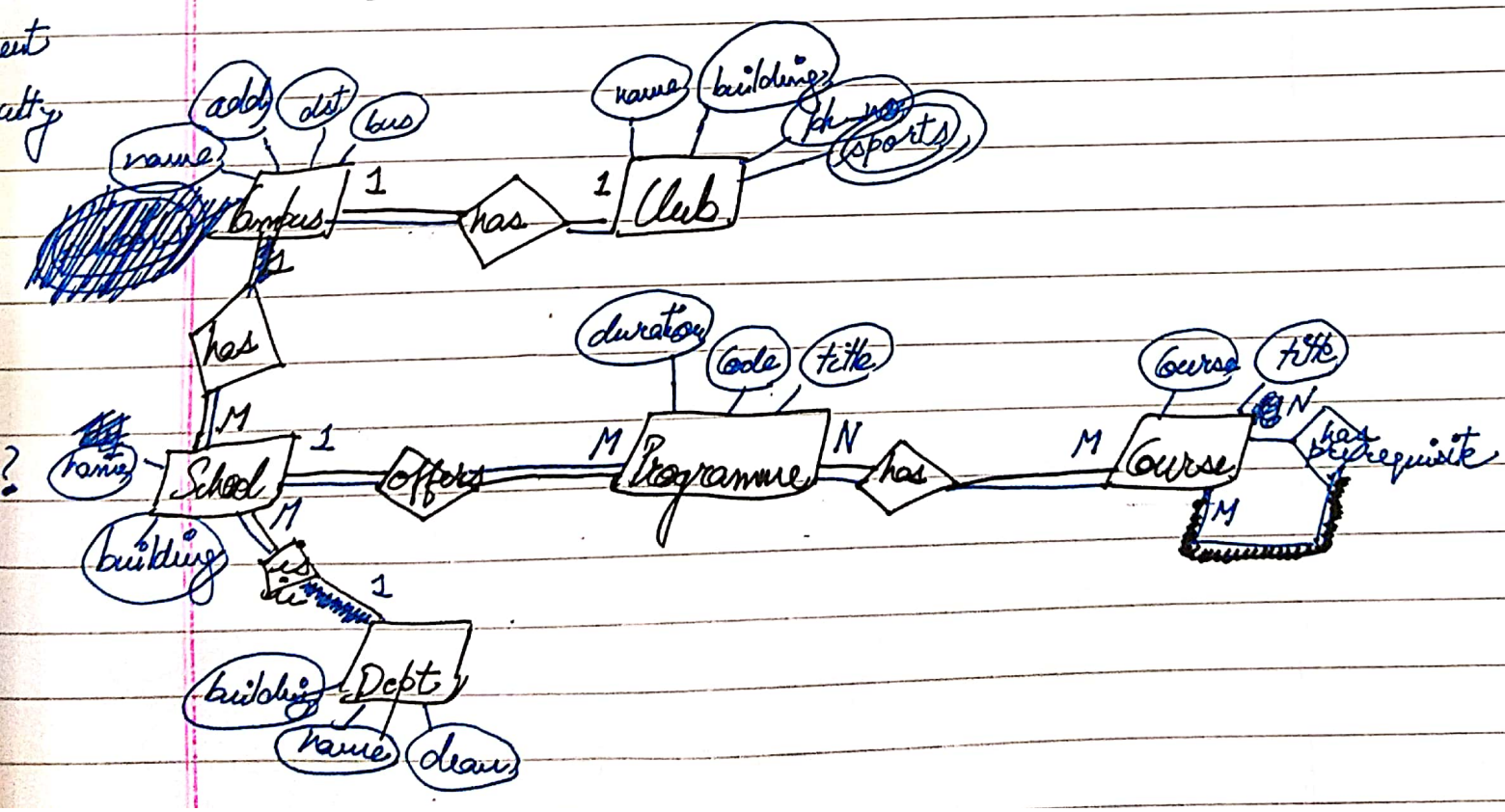
Binary Relations : Two entities participating in this relationship.
degree = 2 //

Relation can be unary or ternary, as well.
degree = 1 (Recursive) degree = 2

There are relations with degree > 3 but they have no name.

→ Relationship name should be such that when you read from left to right or top to bottom, it makes sense.

UNIVERSITY DOCK

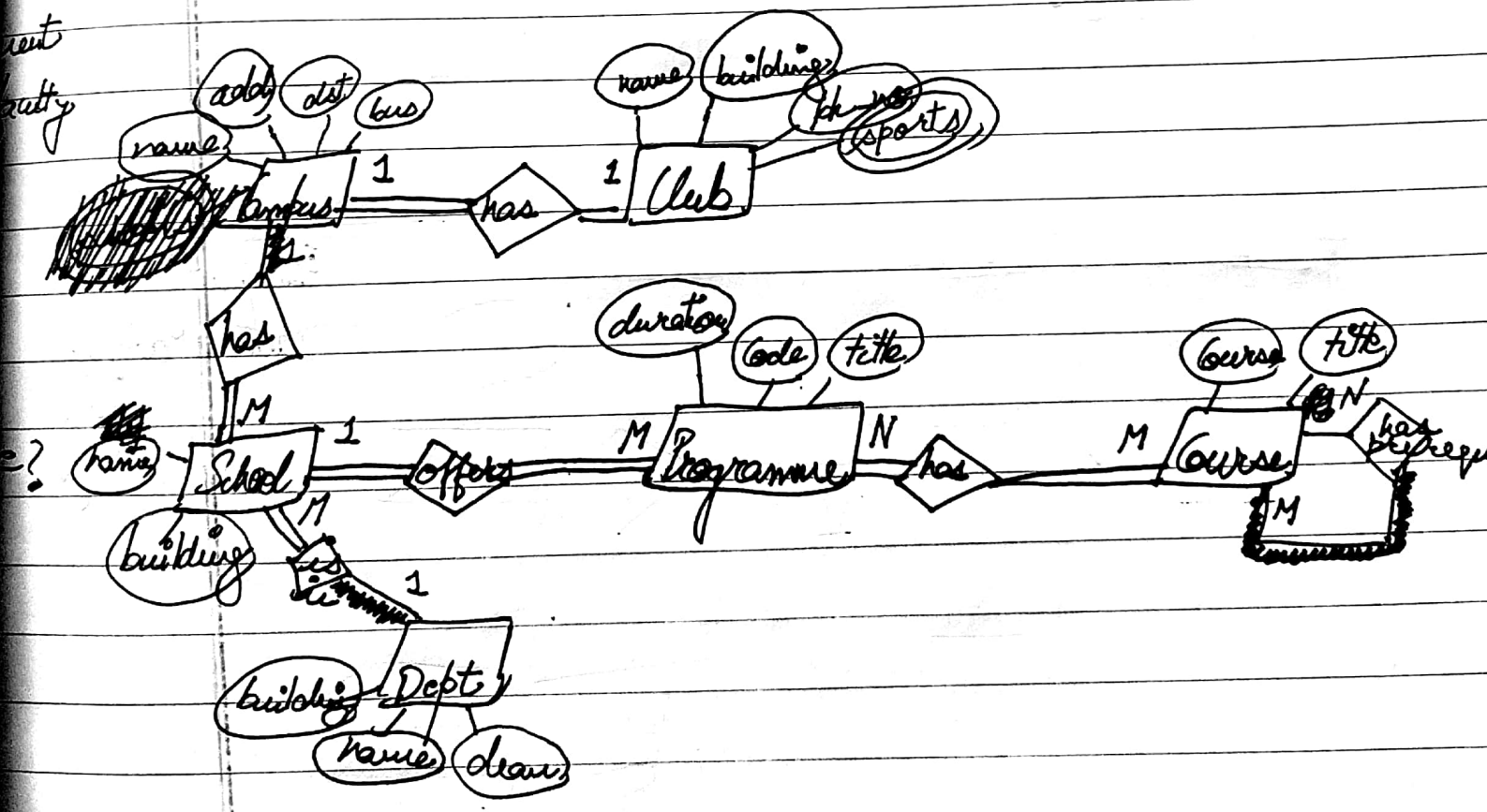


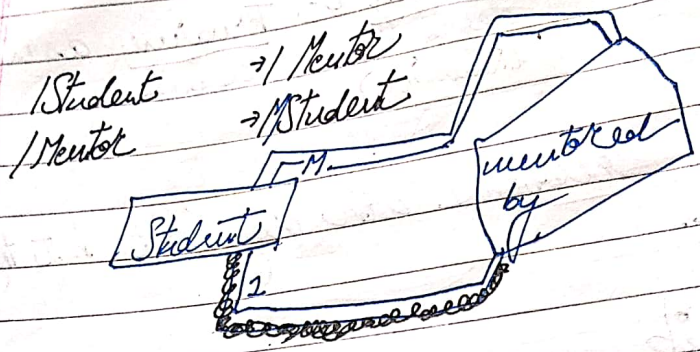
Relation can be unary or ternary, as well.
degree = 1 (Recursive) degree = 2

There are relations with degree > 3 but they have no name.

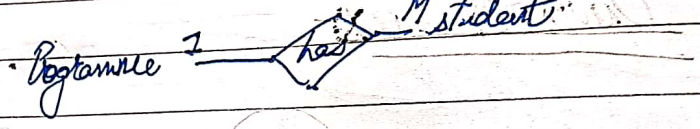
→ Relationship name should be such that when you read from left to right or top to bottom it makes sense.

UNIVERSITY DOCK



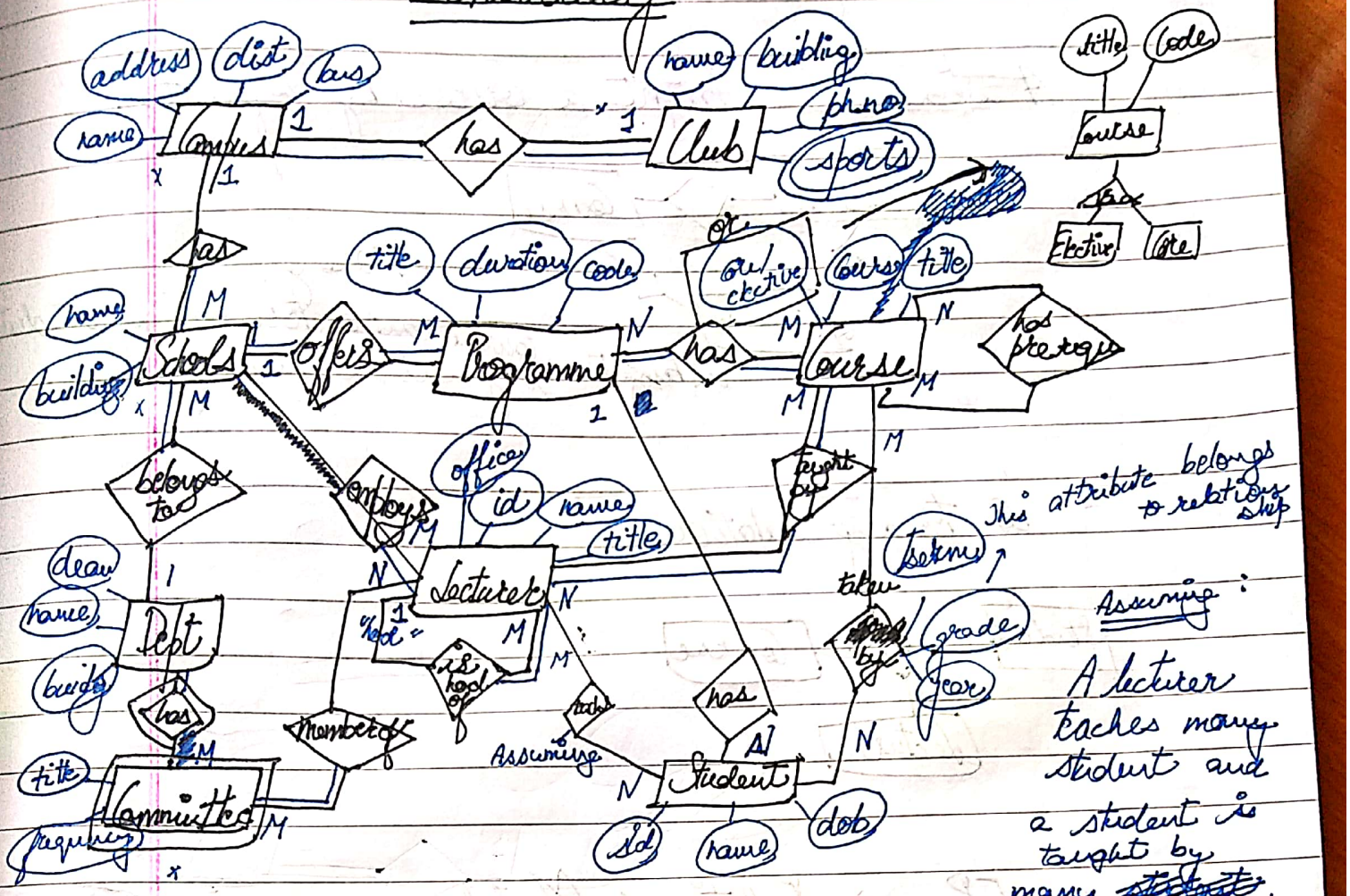


id



4th para
last para (2nd Line)
(frequency)

University :



Participation is a constraint on relationship and not entity.

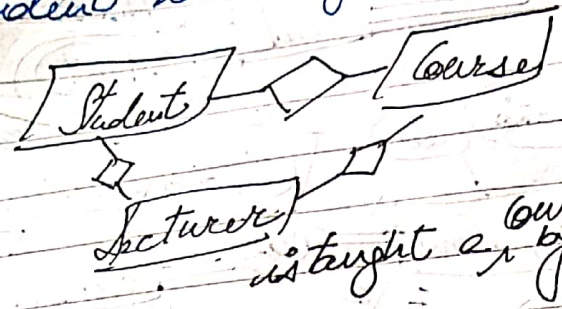
NOTE → Here, Committee doesn't have a key attribute, so called as weak entity sets and entity set with key attribute of their own is known as strong entity sets.

Department along with title (committee) is used to identify individual tuples.

Weak entity set always have full participation on the other side.

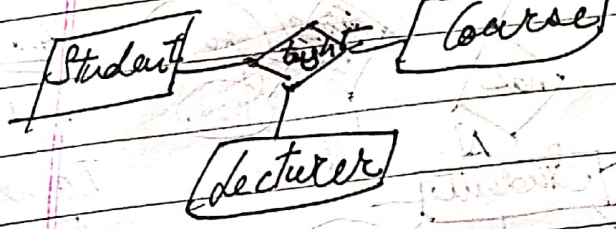
Note: If relation is 1 to M or M to 1, then attribute can go to the entity but if relation is M:N, then attribute belongs to relationship.

A student is taught a course by a faculty.



→ wrong. can't tell which stud.

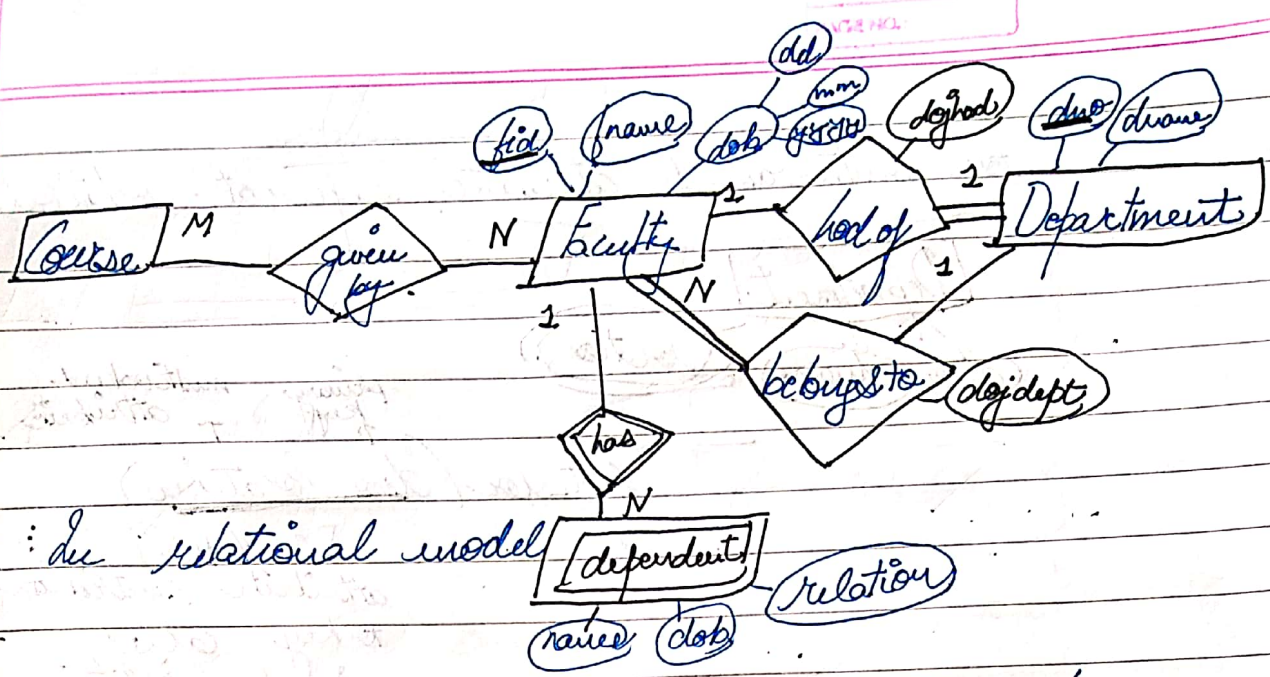
A ternary relation



→ ER → Relational Model

Note: Every strong entity set has a relation in the model.

Lecturer (office_id, name, title)
Course (code, title)
Student (sid, name, dob);
Program (title, duration, code)
Campus (name, address, dist, bus)
Club (name, building, ph_no,
School (name, building)
Department (name, dean, building)
Committee (title, frequency)



Note : In relational model

Wherever foreign key goes, attribute of relation goes with it:

N:1 → foreign key & relation attribute goes to "N" side.

1:1 → goes to full participation side (if on both sides, can go anywhere or we can join those two tables)

N:N → make a separate table primary key of both

cid	fid
c1	f1
c1	f2
c3	f1
c4	f3
c2	f4

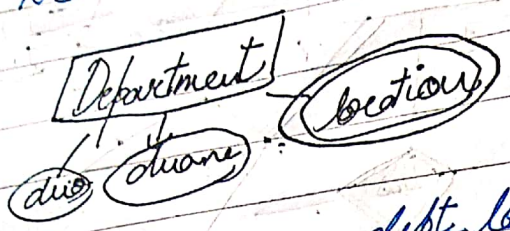
→ name : name of the relation of relation has attributes those also added

givenby (fid, cid)

Course (cid, title, sem, year)

Dept (dno, dname, fid, dojhod)

For multivalued attribute, separate relation.



primary key ↑ T multivalued attribute

dept_loc (dno location)

Since both attribute contribute to key called as key-relation. (not sure check)

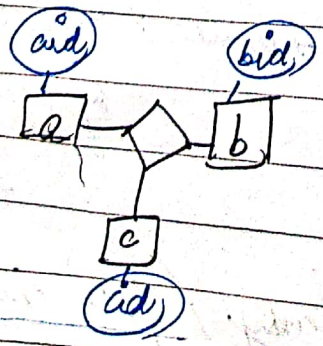
dependent ↑ dependent no.

dependent (fid, dob, name, relation, dep_no)

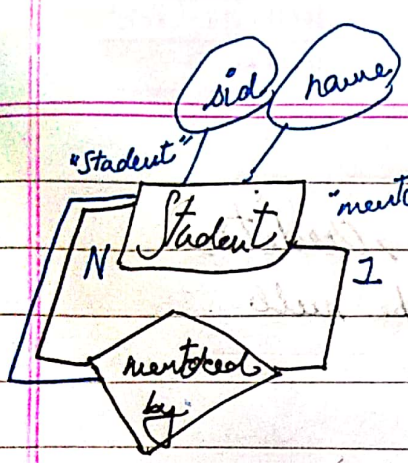
introduced bcz none other comb. can form primary key.

If comb. possible, no need for us to introduce it.

For a relationship of degree greater than 2, a separate relation created for the relationship.



(aid, bid, cid)



student (sid, name, mid)

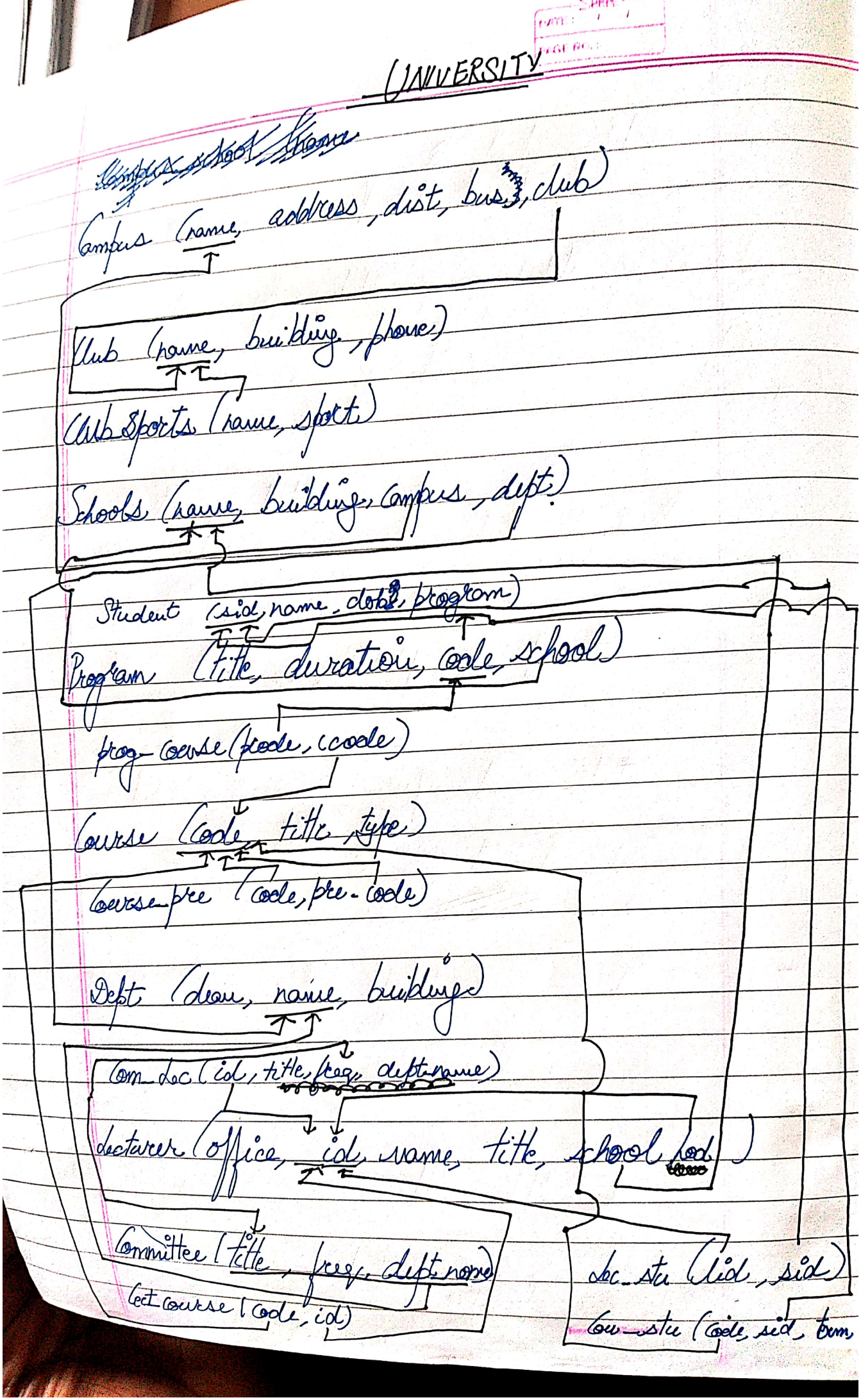
To get name of student & mentor, we join student with itself.

PREVIOUS EXAMPLE AT BACK (University)

cno	cname	dno	dname	date of establishment
c1	a	d1	cse	2009
c2	b	d2	cse	2009
c3	a	d1	cse	2009
c4	c	d2	ece	2009
c5	d	d3	mme	2012

Formal Guidelines for a good relational model

UNIVERSITY



Formal Guidelines for a good relational model

Normalization, Functional Dependency (FD):

Sid	Sname	City	cid	cname	credits
S1	Ramesh	Jaipur	C1	CP	3
S1	"	"	C2	DSA	4
S2	Ram	Agra	C1	CP	3
S2	"	"	C2	DSA	4
S2	"	"	C3	OS	4
S3	Mohan	Jaipur	C1	CP	3
S4	Rohan	Agra	C2	DSA	3

R(sid, sname, city, cid, cname, credits)
both cannot be null

Student (sid, sname, city)
Course (cid, cname, credits)
Registration (cid, sid)

Issues: Redundancy
→ Same piece of info repeated multiple times (sid, sname, city repeated)
→ wasted memory
→ may have problem with update (Eg: credit of CP changes from 3→5)
(An update is made in only one row and rest rows will have a contradiction with it)

Insertion
Eg: A new course available
cs abc 3

Inserting it in table
null null null cs abc 3
sid can't be null as part of primary key

Similarly a new student who has not yet registered in a course can't be inserted.

:Deletion

If we want to remove a course, we can't do it unless we delete sid, sname & city with it too. We can't delete some columns of a row only, the entire row has to be deleted.

$X \longrightarrow Y$ (X: determinant, Y: dependent)

functionally

We say that Y is dependent on X or X determines Y if and only if for a given value of x we can determine exactly one value for Y.

If two rows agree on their x value, they also agree on their Y value.

It represents M:1 relation b/w X and Y.

1. Sid → Sname
2. {Sid, City} → Sname
3. Cid → Cname
4. Cid → Credit
5. Cname → Credit
6. Cname → Cid
7. {Cid, Cname} → Credit
8. {Cid, Credit} → Cname
9. ~~Sid, Cname~~ {Sid, Cname} → Cid
10. {Sid, Cname} → Credits
11. Cid → {Cname, Credit}
12. {Sid, Cid} → Cname

Sid → City

{Sid, Sname} → Sname

B {sid, city} → credit

If sid → sname matches
{sid, ...} → {sname, ...} also matches

Note: 1-1 is a special case of M-1

Armstrong's Axioms

1. Reflexivity: if B ⊂ A then A → B
   Eg: Sid → Sid
   {Sid, cid} → Sid

2. Augmentation: A → B ⇒ AC → BC

3. Transitivity: A → B & B → C ⇒ A → C

~~Sid → Sname~~
~~Sname →~~

4. Self Determination: A → A

5. Decomposition: A → BC
   ⇒ A → B and A → C

6. Union: A → B, A → C ⇒ A → BC

7. Composition: A → B C → D

Closure of Functional Dependency:

$R(A, B, C, D, E, F)$

$F: \{ A \to BC$
$B \to E$
$CD \to EF \}$

Now $AD \to F$

$A \to BC \Rightarrow A \to B$ and $A \to C$ (Decomposition)
and given $B \to E$
also $CD \to E$ and $CD \to F$ (Decomposition)

~~$\Rightarrow A \to E$ (Transitivity)~~
~~$\Rightarrow AF \to EF$ (Augmentation)~~

$A \to C \Rightarrow AD \to CD$
and $CD \to F$

(Transitivity)

$\Rightarrow AD \to F$

Closure: what all functional dependencies can be proved / derived or from given dependencies.

Closure of Attributes:

$A \to BC$
$B \to E$
$CD \to EF$

If A present, $(AD)^+ = \{A, D, B, C, E, F\}$
add B and C

If both CD present, then add E & F

$AD \to F$?

Since, here all attributes can be derived from AD, it is one of the candidate keys.

Usually, attribute of candidate key is on left side.

Let's check if A is candidate Key,

$(A)^+ = \{A, B, C, E\}$ x as D, F not present

$(B)^+ = \{B, E\}$

$(CD)^+ = \{C, D, E, F\}$

$(ACD)^+ = \{A, C, D, B, E, F\}$ ✓ (But not a candidate key)

It is a super key as CD : subset exists & a candidate key

AD : Primary key

Irreducible Set of FD

Right Side should be only 1 attribute

$A \to BC$

$B \to C$

$A \to B$

$AB \to C$

$AC \to D$

$\Rightarrow$

Remove duplicate or any other which can be derived

$A \to B$

~~$A \to C$~~

$B \to C$

~~$A \to B$~~

$AB \to C$

$AC \to D$

$\longrightarrow$

Convert left irreducible

$A \to B$

$B \to C$

$AB \to C$

$AC \to D$

$\longrightarrow$

$A \to B$

$B \to C$

~~$A \to C$~~

$A \to D$

$\rightarrow$

$A \to B$

$B \to C$

$A \to C$

Irreducible Set of FD's

Same (so we can drop B)

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

$(A)^+ = \{A, B, C, D\}$

$(B)^+ = \{B, C\}$

$(C)^+ = \{C\}$

Same (so we can drop C)

$(AC)^+ = \{A, C, B, D\}$