23 Sep-2021

Module 3

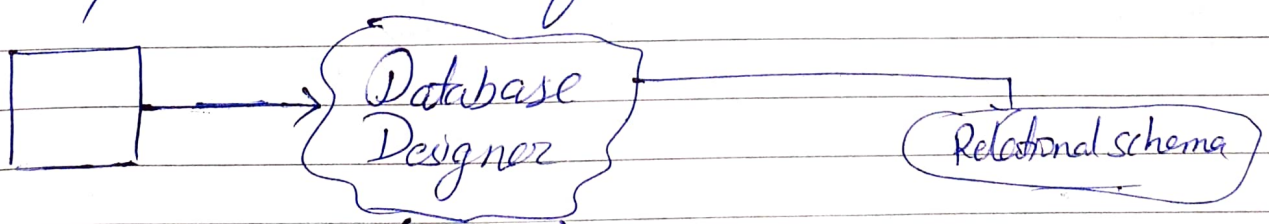
Date ____ Page ____

Entity - Relationship Model.

→ Design Process
→ Data Modelling
→ Modelling of the constraint
→ Entity Relationship diagram.
→ Design issues
→ Weak Entity.
→ Extended Entity Relationship features.
→ Design of a bank database.
→ Reduction to Relational Schema.
→ Database Design.

* Design Process *

- First principle – To understand the actual need of the end user.
- Second principle – Data model needs to represent the enterprise into a set of relational schema.
- Third Principle – Understand the functional requirement of the enterprise.



Two diff. kinds of decisions
- Logical design
- Physical design.

→ Business Decision. (what attributes one needs to store)
→ Computer Science Decision (How we can distribute the attribute across the relational schema.)

Physical design (tells us about the physical layout of the database).

→ Data structure to be used for storage

→ Block size for data transfer and so on...

* Enterprise Description (Bank).

(1) organisational description
- Bank is organised into Branch.
- Each branch is identified by a unique name.
- Bank monitors its assets.

(2) customer details
- Customers are identified by the customer_id value
- Customer may take loans
- Customer is associated with a banker

(3) employee details
- Employees are identified by Emp_id value.
- Other information regarding the employees.

(4) Account details
- Two types of account
  (a) Saving Account
  (b) Current Account.
- Account operation details, a customer can have more than one account.
- Each account is maintained by unique account No.

Date _____ Page _____

Account details.
- Balance & most recend date of use
- Saving account has an interest rate.

Loan details
- ⑤ → A loan can be held by one ore more customers
- Identified by a unique loan_No.
- Bank keeps track of loan amount & loan payment details.
- loan payment No does not uniqually identify, what is called as a particular payment.
- date & payment.

The final goal is to come up with a set of relational schema.

We have to adopt a key design approach. To use a data model for modelling the data.

* Entity Relationship Model

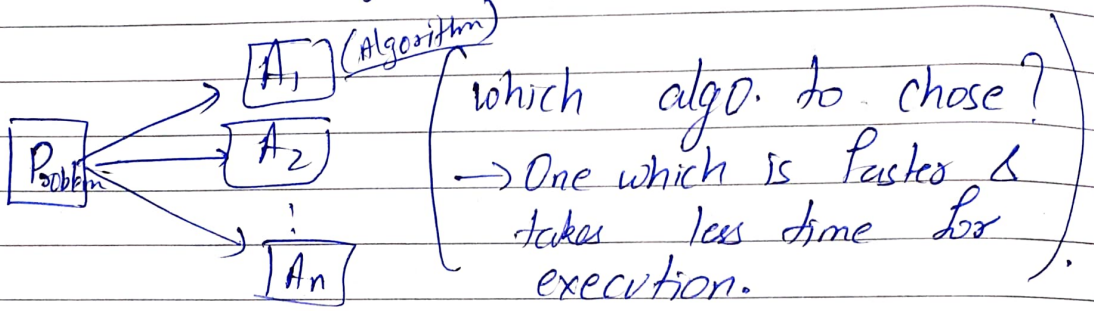
- Models an enterprise as a collection of entities and relationship.

(a) Entities → 'thing' or an 'object' in the enterprise that is distinguishable from the other.
Eg. - Employee, loan, account, customer, branch etc.

→ Each entity is described by a set of attributes

(b) Relationship → It is an association among several entities

This model represents an enterprise at a logical level by a diagram. This is called as Entity-Relationship diagram. (E-R diagram).



1. Entity Set
2. Relationship Set.
3. Attributes.

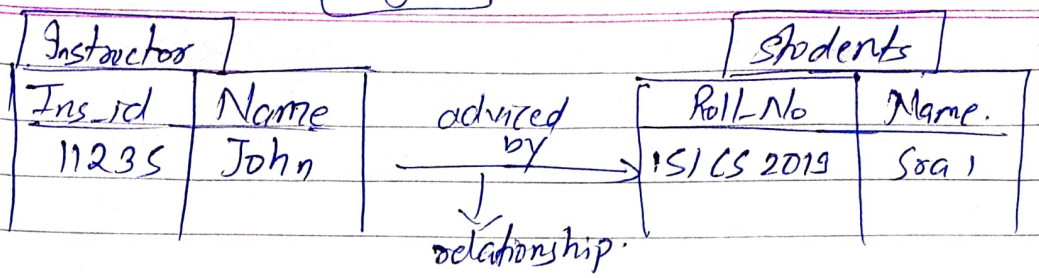
* Entity Set → Set of entities of similar kind.

Eg. Set of Person, Set of company.

Instructor (Ins_id, name, dept_name, salary).
Course (course_id, title, credit)

Fig-1

Date Page



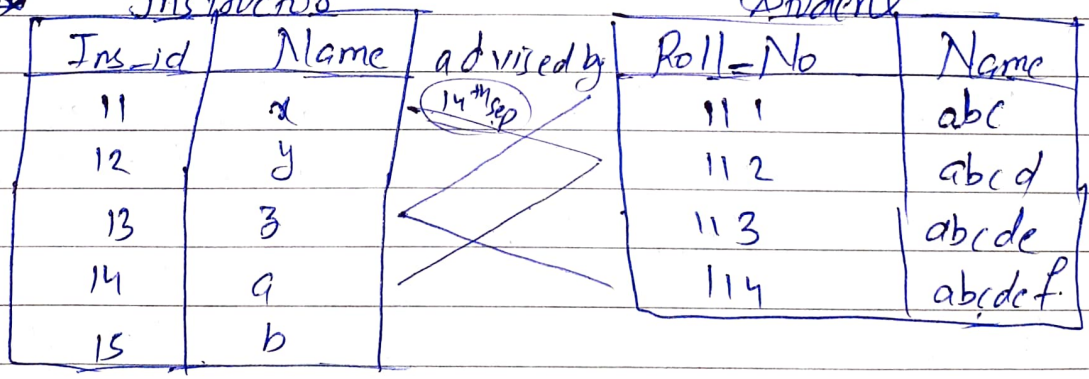
→ "Advised-by" is a relationship associated with two ~~Instr~~ entities Instructor & Students.

A relationship is a mathematical relation with ~~the~~ $n \geq 2$ many entities.

$$\{(e_1, e_2, \ldots e_n) : e_1 \in E_1, e_2 \in E_2 \ldots, e_n \in E_n\}$$

In Fig-1 $(11235, 151CS2019) \in$ Advised by

*



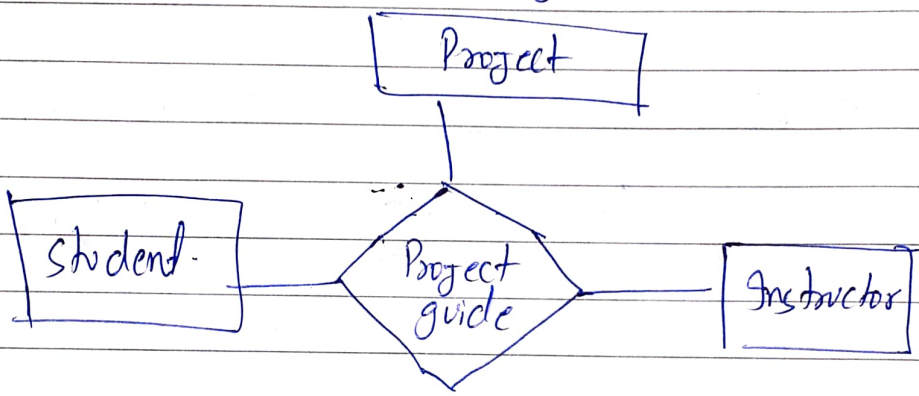
An additional attributes may be associated with relationship.

Date _____ Page _____

- Degree of Relationship → defined as no. of different kinds of entities involved in the relationship.

1) Binary Relationship → Involves two diff. kinds of entities.

2) Ternary Relationship → It is an association among three entities.



Here, (student, project & instructor) are entity and project_guide is relationship.

* Mapping the cardinality Constraints

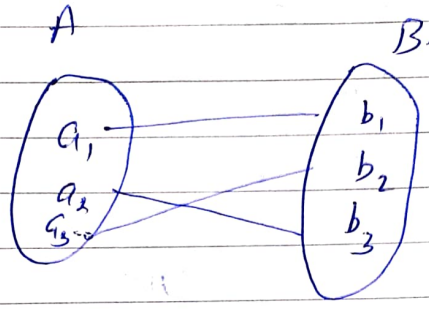
⟹ Expresses the no. of ~~const~~ entities to which another entity can be related / associated via a relationship set. It is useful for describing binary relation.

Date _____ Page _____

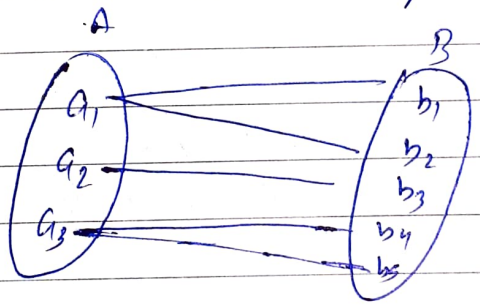
→ Cardinality Constraints.

(1) One to One
(2) One to Many
(3) Many to One
(4) Many to many.

* One to One

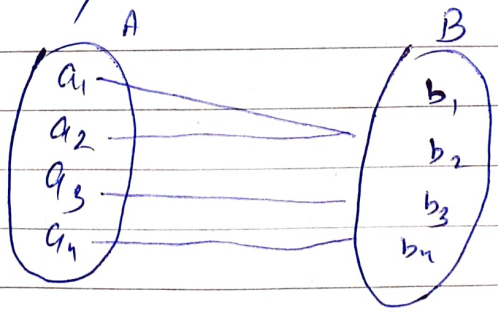


* One to Many.

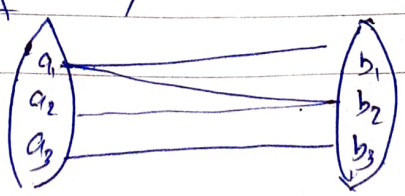


(In this relationship one occurence of an entity relates to many occurences in another entity)

* Many to One



* Many to Many.

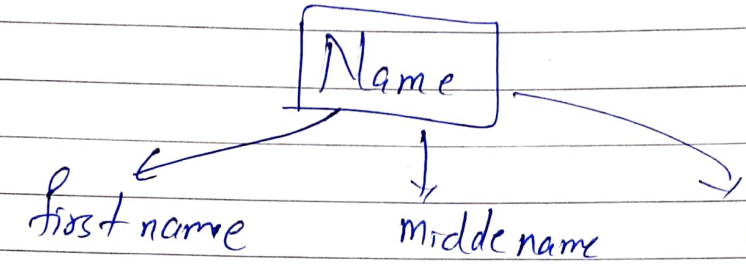


Date _____ Page _____

* Attributes :-

1) Simple v/s Composite Attribute.
2) Single valued v/s Multivalued Attribute
3) Derived Attributes.

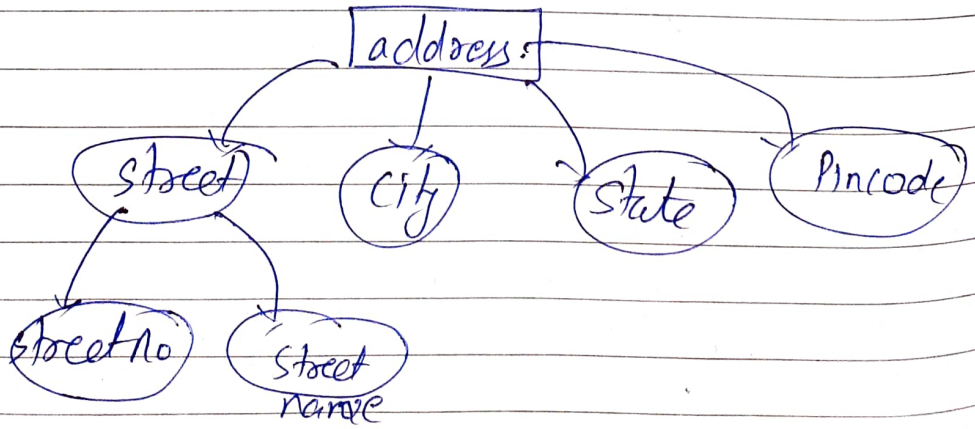
- Composite Attributes can be splitted into simple attributes.



→ It can be splitted into components.

- Simple Attribute → An attribute that can not be subdivided into components.

* Heirarchical Attributes

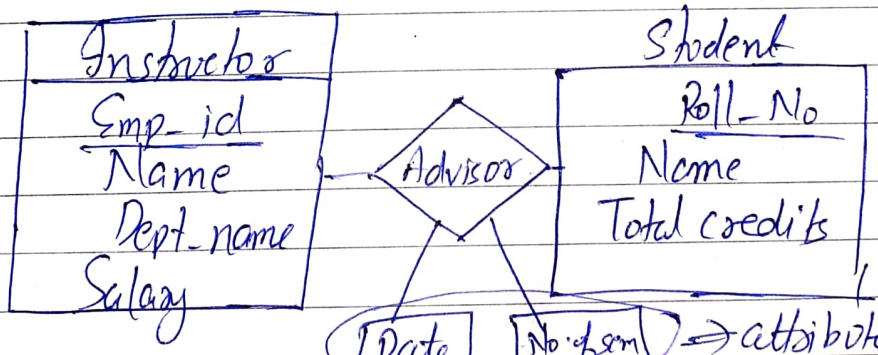


Date Page

* Derived Attributes → That can be derived from other attributes.
  eg. total & average marks of student.
* Single Valued → Attributes which take up only a single value for each entity instance. Eg. age of student
* Multi-valued attribute → Attributes which take up more than a single value for each entity instance.
  Eg. Phone Number of a student.

* Entity Relationship Diagram

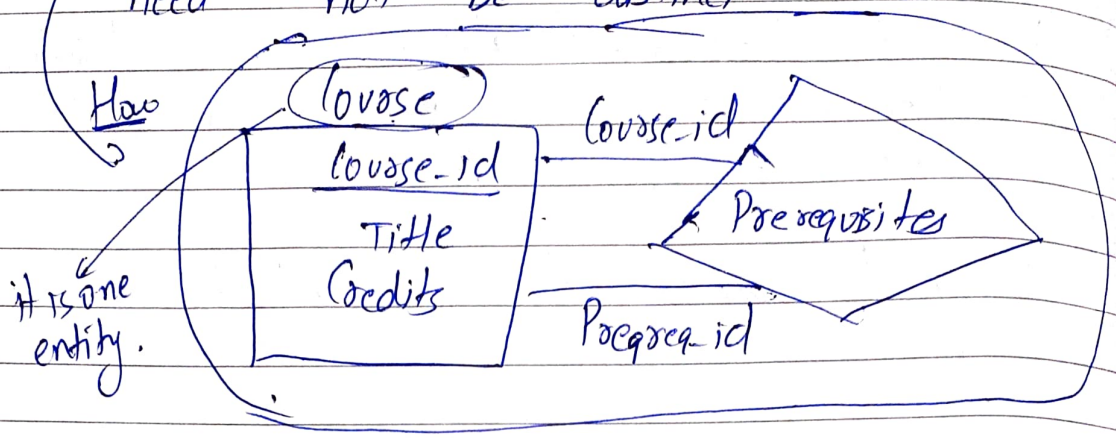
* Entity sets → Entities are represented as or by rectangles. Their attributes are listed on in each line.



Primary key will be mentione with underline (__)

Relationship :→ It is represented by diamond.

Entities involved in a relationship need not be distinct.



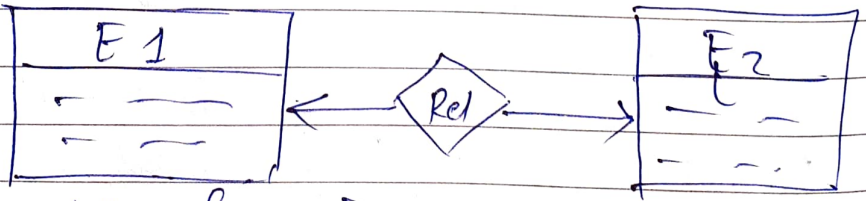
This type of ~~entities~~ relationship are called unary. ~~binary~~ relationship.

* Cardinality Constraints (Sign).

⊙ ⟶ (one).

⊙ — (many).

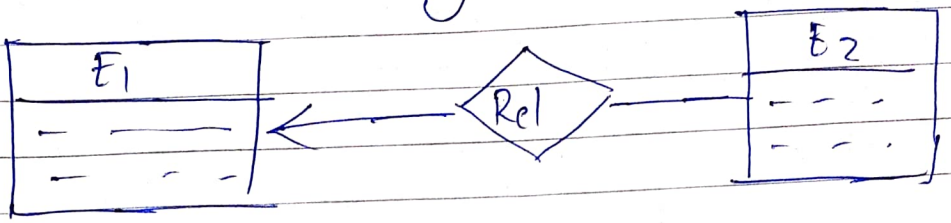
* 1) One to One.



One entity from E_1 is involved with one entity from E_2 and vice versa.

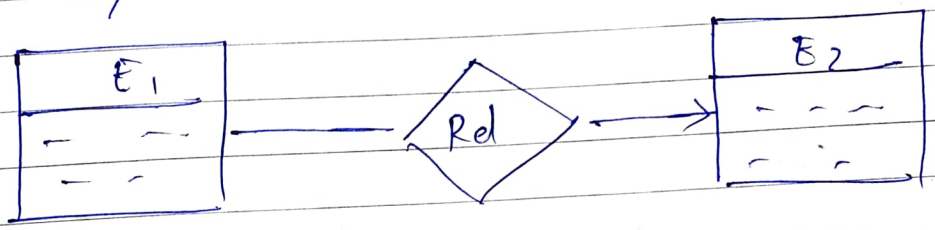
Date _____ Page _____

2) One to Many.

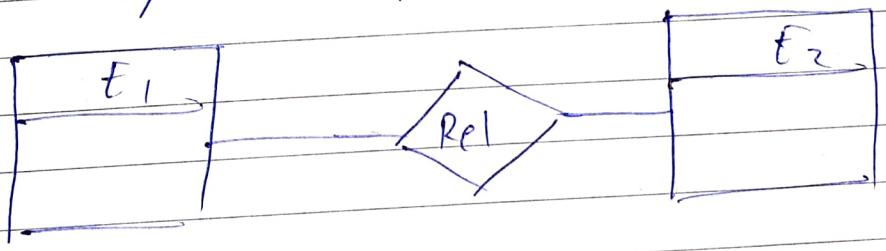


One Entity from E_1 is involved with many entity from E_2.

3) Many to One



4) Many to Many.



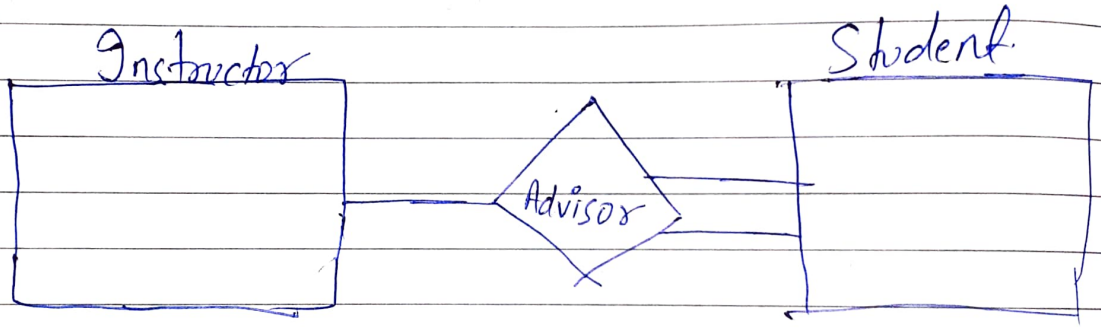
Date _____ Page _____

* How to represent participation of diff. entities in an entity set in ER diagram.

↳ Two types of participation

↓ Total Participation
Every entity is entity set participates in at least one relationship

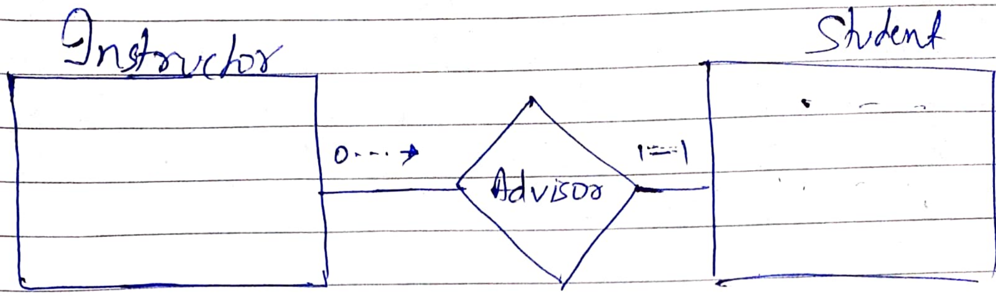
↓ Partial participation.

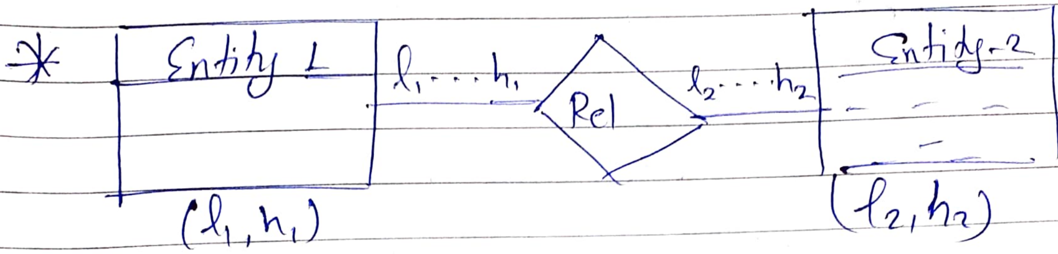


Every student must have an advisor however an instructor may not advise any student

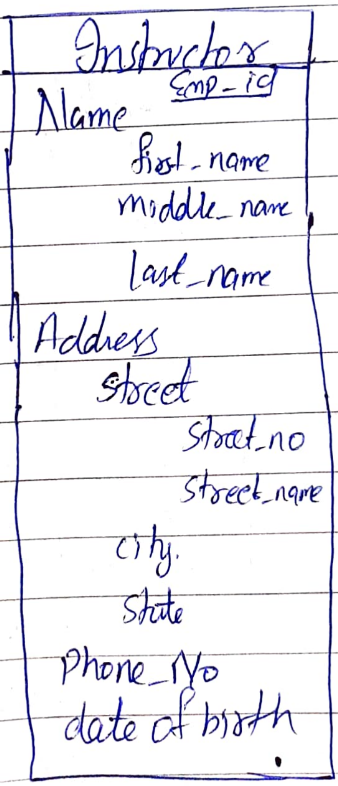
* More complex Constraints.

→ Instructor can advise zero(0) or more student. A student must have one (1) advisor and cannot have multiple advisor.



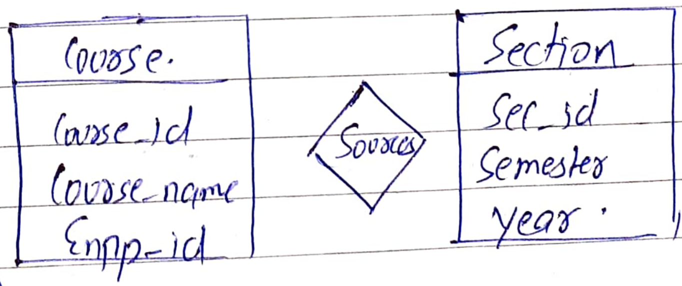


if the upper limit is not known, then it is represented by (*)



Weak Entity Sets.

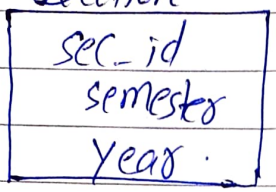
① Course } → (course_id, year,
② Section } (section_id, sem)



Date _____ Page _____

Weak Entity → If the attributes associated with an entity set is not sufficient to uniquely identify an entity, then this is called as weak entity.

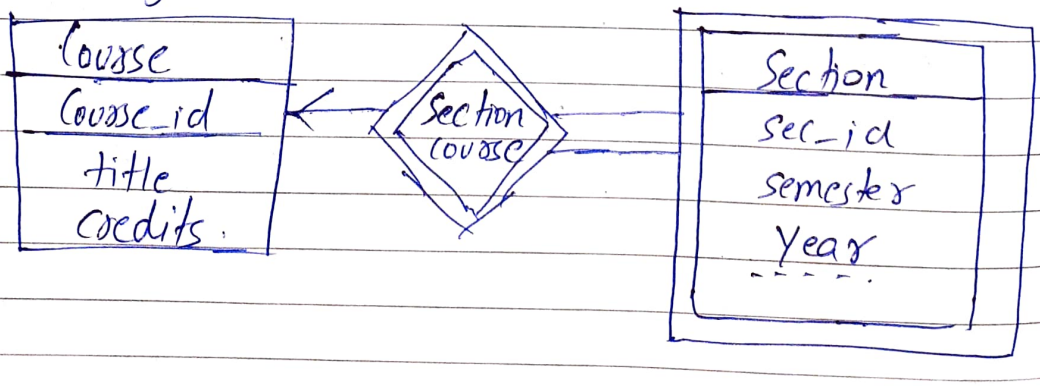
*



* Representation of a weak Entity in E-R diagram.

→ A weak entity is represented via a double rectangle.

→ The relationship of weak entity with identifying strong entity is depicted by double diagonal.



Date _____ Page ____

ER diagram of a Balancing System.

① Bank has got customer
② Bank has got branches
③ Branches are identified by name, branch_no, address
④ Customers are identified by customer_id, phone_no, address
⑤ One customers may have more than one account
⑥ Customers can avail loans
⑦ loans are identified by loan_id, loan_type & amount.
⑧ Accounts & loans are related to a particular branch.

Ans- Step-1: Identify the entities.

Branch, Bank, Customer, loan, Amount.

⇓ these are entities.

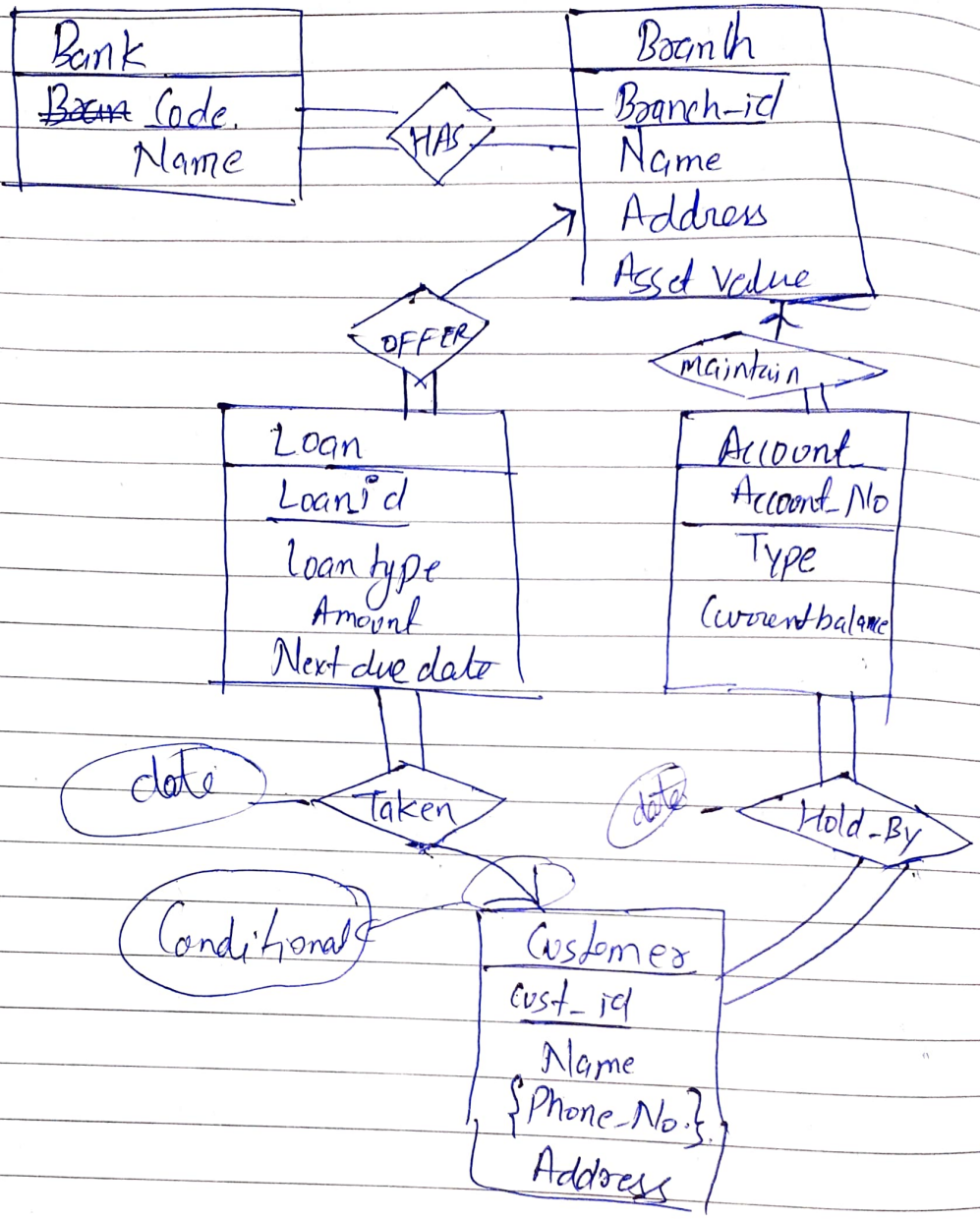
Step 2: Identify the relationship among the entities

Entity 1	Entity 2	Relationship.
Bank	Branch.	HAS.
Branch	loan	MAY OFFER
Customer	loan	Taken
Customer	Account	Hold_by
Account	Branch	Maintain

* Add one more column to represent cardinality & participation constraints.

Date _____ Page

Step-3: Apply the rules of ER diagram to connect entities with the relationship.



* ER diagram of a university.

1. Divided into different departments.
2. Department offers courses.
3. Instructors and students belong to different departments.

(10)

2M

Date _____ Page ___

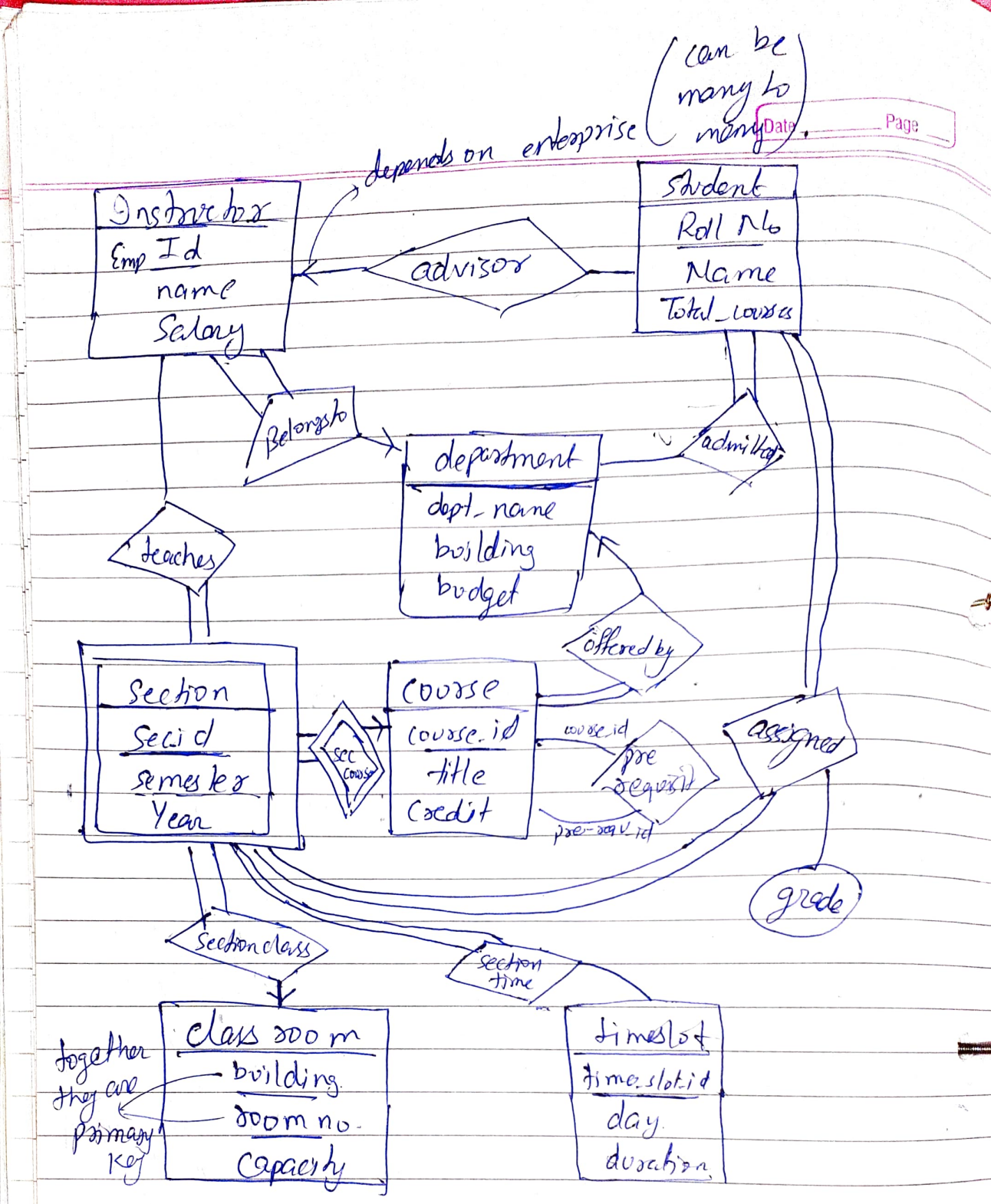
④ classes are divided into different sections
⑤ different sections run into different time slots & class_room.
⑥ different courses have diff. pre-requist requirements.

Ans :- Step 1 : Identify the id entities.

Department, Instructors, students, sections, time_slots, class_room, courses.

Step-2 : Identify the relationship among the entities.

Entity-1	Entity-2	Relationship.
Department	Instructors	Belongs_To
Department	Students	Admitted
Instructor	Students	Advised by
Instructor	Section	teaches.
Section	Course	Sec_course
Department	course	~~Dep~~ offed by.
Sections	class_room	section_class.
section	time_slot	section_time.
Student	Section	Assigned.
course		pre-requist.



* Reduction to Relational Schemas

⇒ A database which confirms to an E|R diagram can be represented by a collection of relational schemas.

Date _____ Page _____

⇒ Each Schema has got a no. of columns which have unique name.

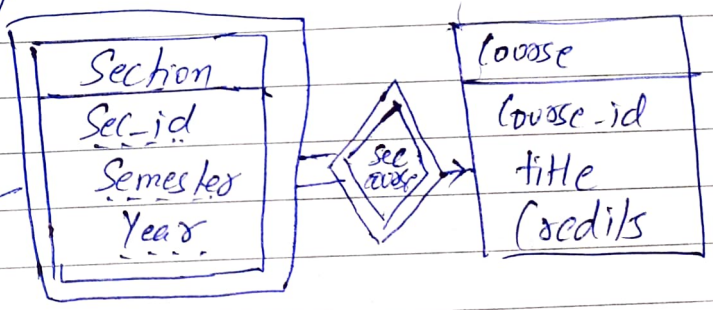
✱ Rules :-

① Each strong Entity presents in the ER diagram reduces to a schema with the same name and same set of attributes.

Student (Roll_No, Name, total_credits).

② Weak Entity set becomes a relational schema that includes a column for the primary key of the identifying strong entity.

Eg.



As section is a weak enity, it will be converted to relational schema as follows :-

Section (Course_id, Sec_id, Semester, Year)

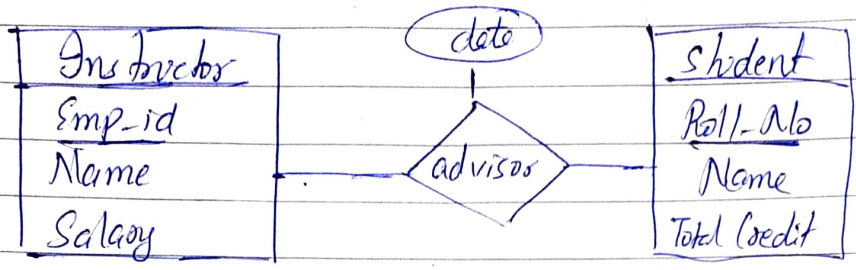
together these 4 are ~~called~~ primary key.

Date Page

Representing Relationship

- A many to many relationship is represented as a schema with the following attributes.
  1. Primary key for both entities.
  2. Attribute associated with the relationship itself

Eg.



Advisor (Emp_id, Roll_No, date)

⇒ How do deal with multi-valued attributes ? → like Phone_No.?

Ans → A multi valued attribute for an entity (E) is represented as a separate schema. It will be corresponding to primary key of E and an attribute corresponding to multivalue attribute M.

inst_phone = (Emp_id, Phone_Number)

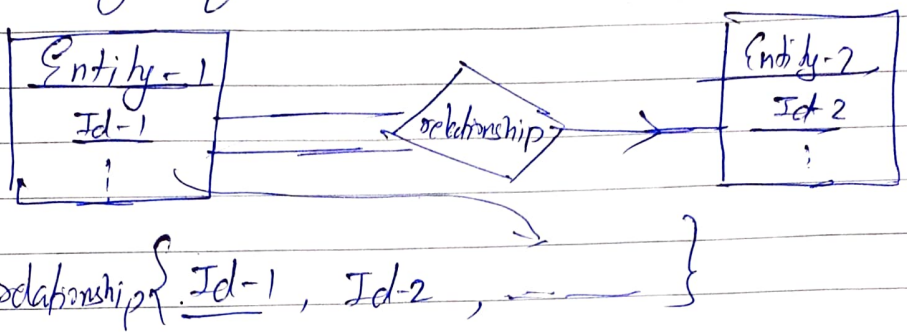
Eg. An instructor with Emp_id 122 has phone No abc & xyz

(122, abc) , (122, xyz) ∈ inst_phone

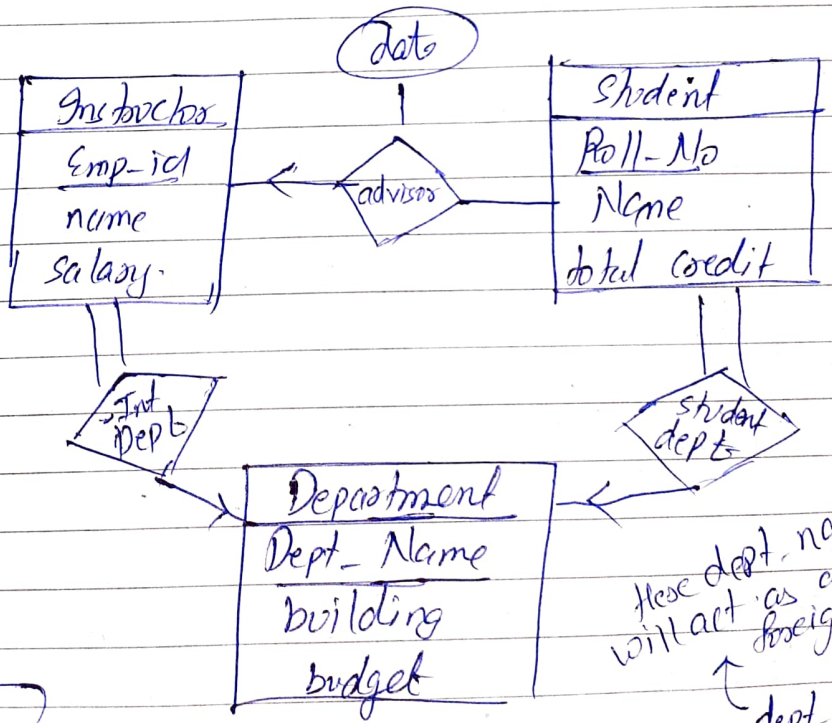
Date Page

- Many to One / One to Many.

⇒ Many to One or (One to Many) relationship that are total on many side can be represented by adding an entire attribute to many side containing the primary key of "one" side.



Eg.



Rule-1

Instructor (Emp_id, Name, Salary, dept_name)

Student (Roll_No, Name, total_credit, dept_name)

Department (Dept_Name, building, budget)

Advisor (Emp_id, Roll_No, date)

* One to One

→ For one to one relationship either side can be chosen as many and act accordingly.

→ An extra attribute can be ~~added~~ added to many side entity.

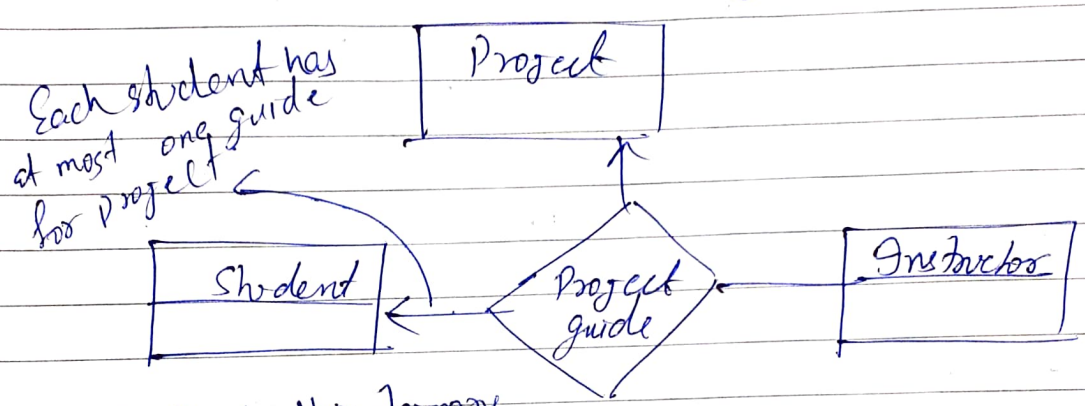
* Draw the relational schemas of university data base.

(1) Student (Roll No, name, total credit, dept_name
(2) Instructor (Emp_id, Name, Salary, dept_name)
(3) Department (Dept_Name, building, budget)
(4) Advisor (Roll_No, Emp_id, Date)
(5) Section (course_id, sec_id, semester, year
(6) Course (course_id, title, credits, dept_name)
(7) teaches (Emp_id, Course_id, Sec_id, Semester, year)
(8) Assigned (Roll_No, course_id, sec_id, semester, year, grade)
(9) time slot (timeslot_id, day, duration)
(10) Prerequisite (course_id, Prerequisite_id)
(11) Class_room (building, Room_No, capacity)

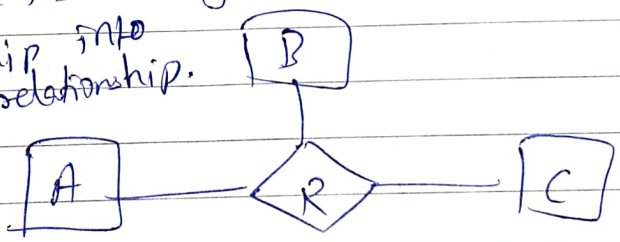
Date _____ Page _____

* Ternary Relationship

Though most of the relationship that we encounter in ER diagram are binary, still in many situations it is more convinent to represent relationship as non-binary (in particular ternary).

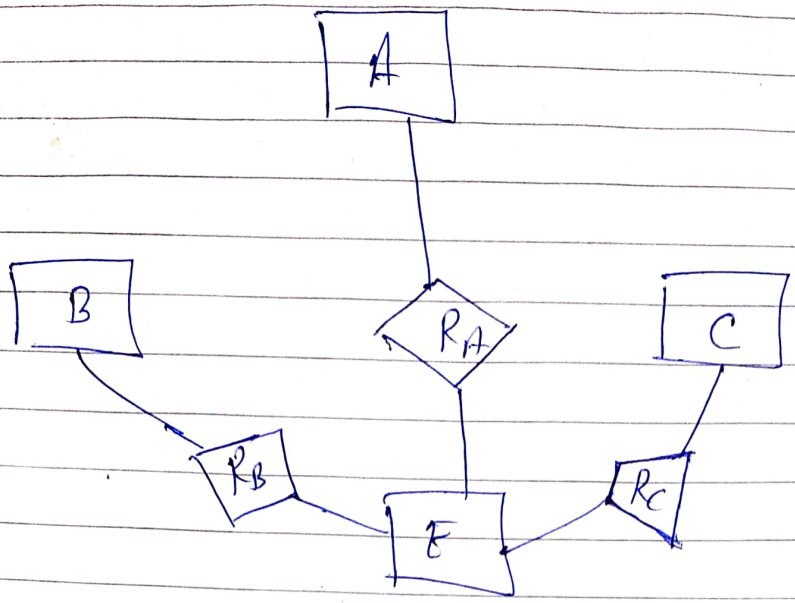


⇒ Break this ternary relationship into binary relationship.



→ Replace R by an entity set E with 3 relationship sets.

- R_A relating E and A
- R_B relating E & B
- R_C relating E & C.



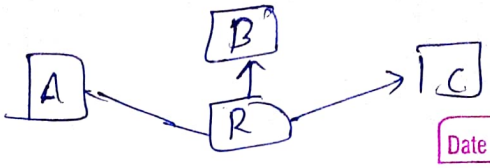
Ternary Relationship is broken into three binary relationship.

→ Steps for conversion.

① Create an identifying attribute for entity E and all the attribute of the Relationship R are added to E

② For each relationship $(a_i, b_i, c_i) \in R$,
- a new entity (e_i) in the entity set E is created
- add $(e_i, a_i) \in R_A$
- add $(e_i, b_i) \in R_B$
- add $(e_i, c_i) \in R_C$

We allow at most one arrow in a ternary relationship to indicate a cardinality constraint.



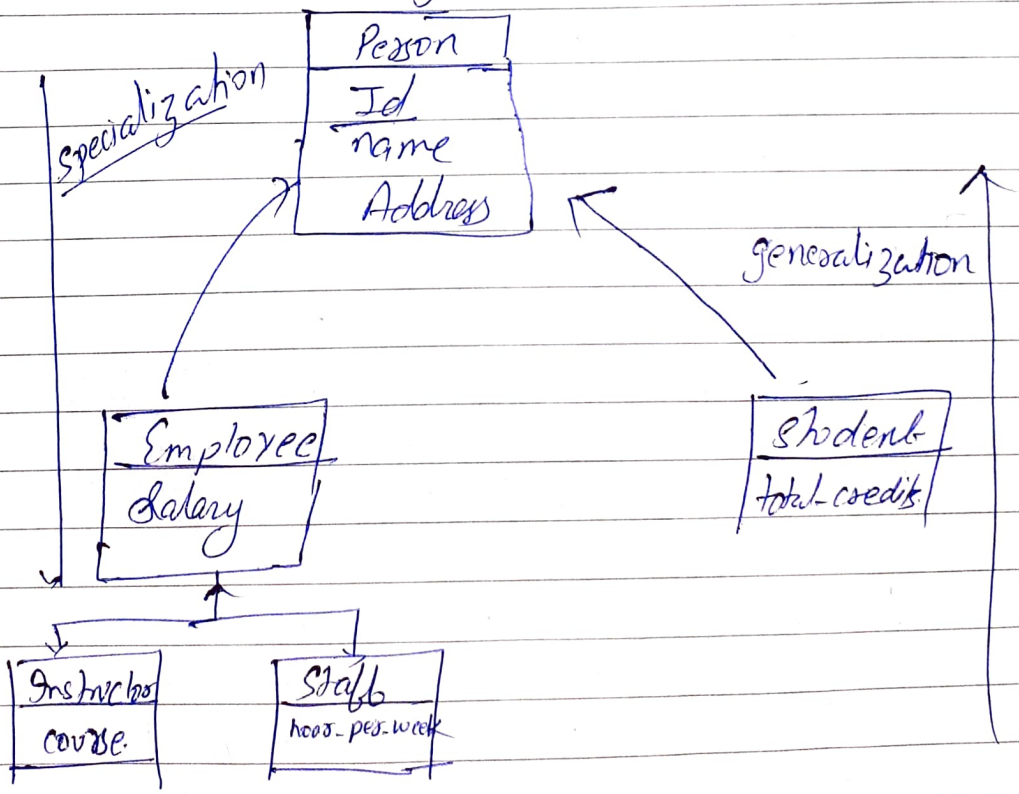
Date _____ Page _____

Interpretation - 1: Each entity is associated with a unique entity from B & C

Interpretation - 2: Each pair of entity from (A, B) is associated with a unique entity from B. Similarly (A, C) is associated with an entity from B.

* Specialization & Generalization *

In the top down design process, we often may allow subgrouping within an entity set itself.



Date _____ Page _____

Attribute Inheritance → A lower level entity set inherits all the attribute of the higher level entity set to which it is linked.

* Overlapping & Disjoint

⇒ Employee & Students are overlapping as student who is doing PHD is an employee as well as student

Instructor & staff are disjoint

Q. How to convert generalization & specialization to relationship schema?

Ans → (i) form a schema for the higher level entity
(ii) form a schema for each lower level entity set and include primary key of higher level entity set and the local attribute.

Person (ID, name, address)
Employee (ID, salary)
Student (ID, total credits)
Instructor (ID, course)
Staff (ID, hour_per_week)

Date _____ Page _____

Advantages & Dis advantages

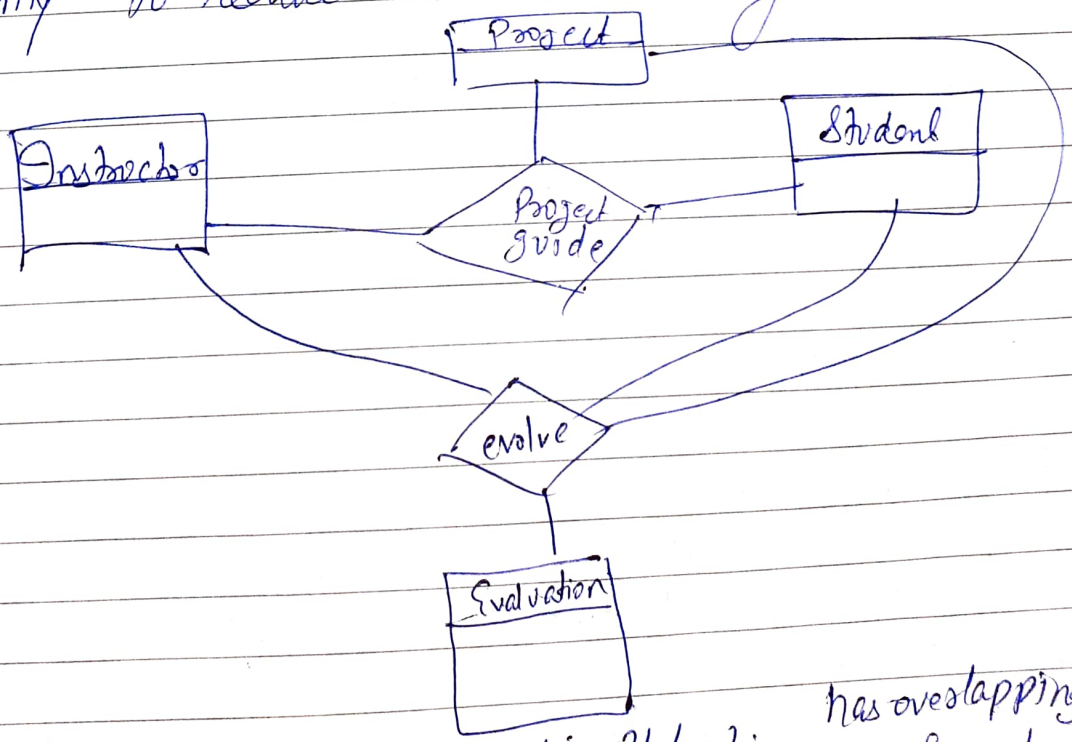
(Easy to handle)

* Draw relationship schema for banking database.

Bank (code, name , address)
Branch (Branch_id , name, address , asset value)
Loan (loan_id , loan_type, amount , ~~branch_id~~)
Account (Acc_No, type, balance, Branch_id)
Customer (cust_id , name, address)
Taken (cust_id , loan_id , date)
Hold By (cust_id , account_No , date)
Has (code, branch_id)

* Aggregation ⇒

Only to reduce data redundancy.



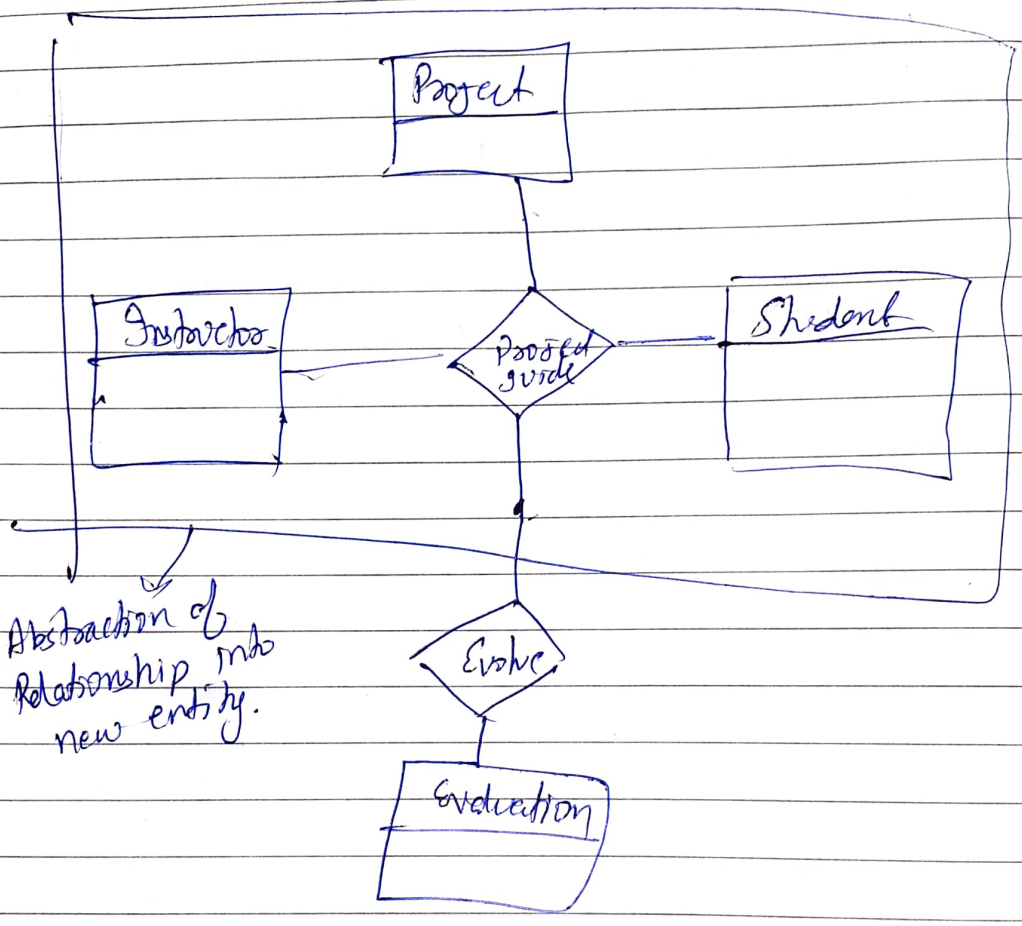
evolve & Projectguide Relationship has overlapping information.

two notes.

Eliminate this redundancy via aggregation

Choose one from the following steps :→ ~~have to be followed.~~ to reduce data redunceancy :→

1) Treat relationship as an abstract entity
2) Allow relationship b/w relationship.
3) Abstraction of Relationship into new entity.

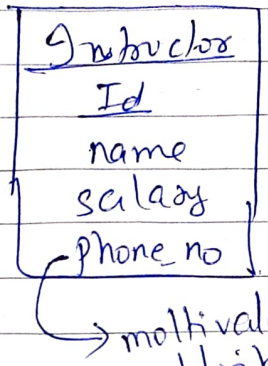


Evolve (Student_id, Inst_id, Proj_id, Ev_id, grade)

Here Project_guide data is redundant.
So we will not draw any schema for project_guide.

Date _____ Page _____

* Entities V/s Attributes.



⇓

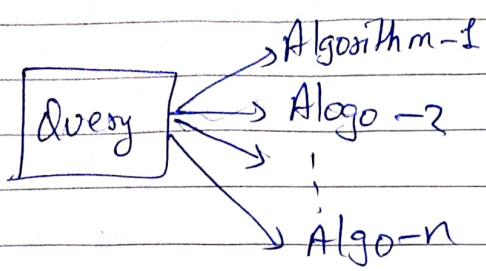
Here we will create one entity of (phone_no & id) to reduce redoncancy.

⇓ Phone_No (Id, Phone_no).

Module-3

Functional Dependency & Database Normalization

Date Page



- Features of good database design.
- Functional dependency (way of expressing constraint into a relational schema).
- ~~Analysis~~ Anomalies (Inserting/Deletion/Update)
- Normal form
- Breaking down Relational Schema.
- Algorithm for decomposition of relation
- Join dependency.
- More normal forms based on join dependency.
- By designing practices.

* Employee (E_name, Aadhar_id, address, D_no)
* Department (D_name, D_no, Manager_Aadhar_id)

Emp_Depart (E_name, Aadhar_id, address, D_no, D_name, Manager_Aadhar_id)

What are advantages & disadvantages

→ Advantages

↳ in operation where we have to merge two tables (Employee & Department) we will be saved from natural join.

Date Page

→ Disadvantages.

(i) Replication of same information
↳ data redundancy will increase.

(ii) Interpretation of NULL is different & difficult. it can be data not provided, data not available, etc. NULL values also occupy space.

Emp_Depart

E_name	Aadhar_id	address	D_No	D_name	Man_Aadhar_id
John	xxx 3456	Bihar	D1065	Bio-Eng.	xxx abcd
NULL	NULL	NULL	D1064	Arts	xxy 1304

Query. → Suppose a new department has been created. with following details. of (D_No, D_name & Man_Aadhar_id).

→ Since Aadhar_id is a primary key, it can be null so it will be a ("Insertion Anomaly".)

Query :→ if John gets relieved, then the information of Department will be lost as he is the only guy with that department. So this is called det (deletion anomaly.)

(Query: Suppose we have to update manager's Aadhar_id, we have to update everywhere in the table.

→ (updation anomaly)

→ It is highly probable that the database will be in the inconsistent state after updation operation.

→ Solution is we can split the table, such that data reduncany will not happen.

* The only way to avoid data reduncany is to decompose a relational schema into smaller schemas
  The no. of attributes are smaller.

How it can be done?

Employee (ID, name, street, city, salary)

Employee 1 (ID, name)

Employee 2 (name, street, salary, city)

(D104, Ram, Charminar, Behal, 25K)

(D105, Ram, Sadar, Bhopal, 30K)

Employee-1

Id	Name
D104	Ram
D105	Ram

Employee-2

Name	Street	City	Salary
Ram	Charminar	Behal	25K
Ram	Sadar	Bhopal	30K

Date Page

Employee 1 ⋈ Employee 2.

we will get 4 tuples.

Id	Name	Street	City	Salary
D104	Ram	Char minar	Bclw	75k ✓
D104	Ram	Sadar	Bhopal	80k } xx
D105	Ram	Char minar	Bclw	75k
D105	Ram	Sadar	Bhopal	80k ✓

Two additional tuples are appearing when we have done natural Join which is incorrect.

* Let R be a relational schema and R_1, R_2 form a decomposition of R

(1) $R_1 = R_1 \cup R_2$.

this even holds when we divide into more than two schemas:

$\Rightarrow R = R_1 \cup R_2 \cup R_3 \cdots R_n$

if we have decomposed R into $R_1, R_2 \cdots R_n$.

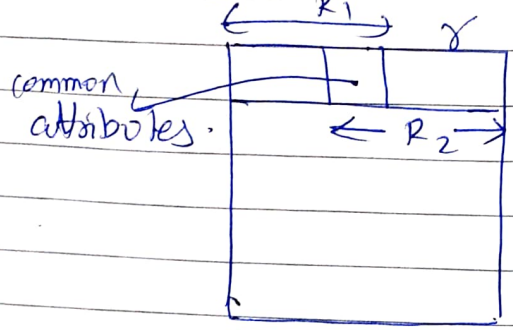
(2) Lossless join property.

→ We say that the decomposition is a lossless decomposition if there is no loss of information

by replacing R with R_1 & R_2.

$\gamma(R)$ → set of attributes
↙ name of schema

$$\Pi_{R_1}(\gamma) \bowtie \Pi_{R_2}(\gamma) = \gamma$$



if this is valid then only we can decompose it.
& it is following lossless join property.

A decomposition is lossy if natural Join of projection, we get a proper & superset of original relation.

formally → $\gamma \subset \Pi_{R_1}(\gamma) \bowtie \Pi_{R_2}(\gamma)$

Eg. $\gamma(A,B,C)$ → $\gamma_1(A,B)$, $\gamma_2(B,C)$

γ

A	B	C
α	1	A
β	2	B

γ_1

A	B
α	1
β	2

γ_2

B	C
1	A
2	B

Date _____ Page _____

So here we will do natural join

$$\Pi_{A,B}(\gamma) \bowtie \Pi_{B,C}(\gamma) = \gamma$$

A	B	C
α	1	Λ
β	2	B

This is an example of lossless join property.

* Normalization theory.

→ To decide whether a particular relation 'r' is in good form or not.
(less amount of redundancy).

→ If this is not then how it can be decomposed into set of relational schema such that
① Each decomposed relation is in good form.
② The decomposition is a lossless decomposition.

→ Normalization theory depends on
① Functional dependency.
② Multivalued dependency.

Date _____ Page _____

① Functional Dependency →

→ There are variety of constraints in real world.

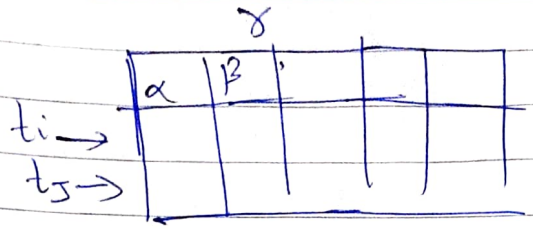
→ Functional dependency gives a mechanism to define a new set of constraints which are not possible using 'domain, not-null, primary key, foreign key etc' ~~are~~ constraints.

→ An instance of relation that satisfies all such real world constraints is called legal instance.

* Primary key : A subset of attributes that can uniquely identify a tuple in a relation.

Let 'R' be a relational schema with 'R' as set of attributes and $\alpha \subseteq R$ & $\beta \subseteq R$. The functional dependency $\alpha \rightarrow \beta$ holds on R if and only if for any legal relation $r(R)$, whenever any tuples t_1 & t_2 of ~~R~~ r agree on the attributes α, they also agree on attributes β.

Date _____ Page _____



where $t_i \neq t_j$.

if $\alpha \rightarrow \beta$ & $t_1[\alpha] = t_2[\alpha]$

then $t_1[\beta] = t_2[\beta]$.

Eg.

γ

A	B
1	4
1	5
3	2

Q. does $A \rightarrow B$ holds?
Ans - No.

Q does $B \rightarrow A$ hold?
Ans - Yes, because we didn't find any two tuples in which A attributes has diff. value.

20/10/2021

* functional dependency :-

$\gamma(R)$, $\alpha \subseteq R$, $\beta \subseteq R$

$\alpha \rightarrow \beta$ holds if for any pair of tuples t_1 & t_2 if $t_1[\alpha] = t_2[\alpha]$ then $t_1[\beta] = t_2[\beta]$.

* $f \rightarrow$ set of given functional dependency

$f = \{ A \rightarrow B , B \rightarrow C \}$.

if there A values are same then there B value also same.

if then B values are same then C value will also be same.

Super key → Uniquely identify a tuple.

Date _____ Page _____

Given a set of functional dependencies, another set of functional dependencies are basically implied.

	A	B	C
t_1	1	4	6
t_2	1	4	6
t_3	2	4	8

⇒ from here we can say that A → C also holds true.

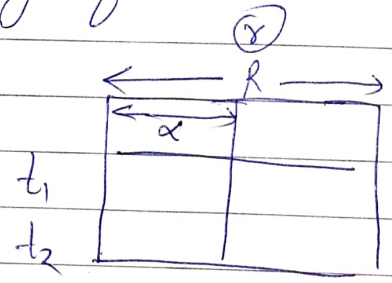
Implied set of functional dependencies = F'
Given set of functional dependencies = F

$$F^+ / F^C = F \cup F'$$

The set of functional dependencies that can be derived from given set of Functional dependencies is called the 'closure of F'.

* Keys in terms of Functional Dependency:-

- $\gamma(R)$, $\alpha \subseteq R$, we call α is the superkey of γ if and only if $\alpha \to R$



Date _____ Page _____

* Candidate key : α is a candidate key for if and only if

① $\alpha \rightarrow R$
② no β C α , $\beta \rightarrow R$

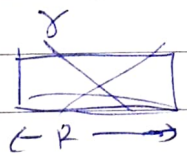
* Functional dependency is a generalised way to impose constraints on a relational schema.

Inst_dept (ID, name, salary, dept_name, building, budget).

dept_name → building. ✓
ID → building. ✓
dept_name → salary ✗✗

* Use of functional dependency :→

. if a relational 'r' is legal under any given set of functional dependency then we can say that r satisfies F.



Eg.

$F = \{A \rightarrow B\}$.

r

A	B	C
1	2	6
1	2	6
2	9	7

if we want to add (2, 10, 11) then it can't be inserted due to functional dependency constraints.

Date _____ Page _____

* FD tests if they are legal under a given set of functional dependency (FD).

A	B	C
1	4	3
3	1	4
2	4	3

$F = \{A \rightarrow B\}$

↳ We can add (5, 4, 6).

By looking at table ($B \rightarrow C$ holds).

A specific ~~name~~ instance of a relational schema may hold/satisfy a FD even if it is not defined.

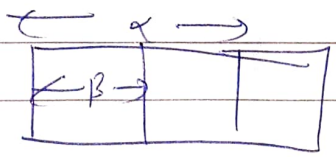
* Trivial functional Dependency

→ A FD is trivial if it is satisfied by all instances of a relation. (if both values are same)

- ID, name ⟶ name.
- name ⟶ name.

Id	name	street	Phone No
D104	Ram		
D104	Ram		

In general for any FD $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$

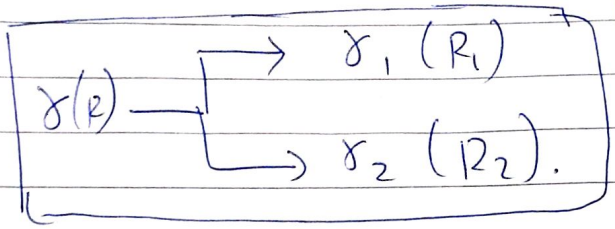


Date Page

Eg. FD ⇒ $\{A, B \rightarrow C\}$

For any tuples if (both value, A & B are same, then their C values will be same).

* Lossless Decomposition in terms of FD.



$$\gamma = \Pi_{R_1}(\gamma) \bowtie \Pi_{R_2}(\gamma).$$

A decomposition of R into (R_1, R_2) is lossless if atleast one of the following dependencies is in F^+.

F^+ : Closure of F

① $R_1 \cap R_2 \rightarrow R_1$

② $R_1 \cap R_2 \rightarrow R_2$

Eg. $\gamma(A, B, C)$ → $\gamma_1(A, B)$, $\gamma_2(B, C)$

$F = \{A \rightarrow B, B \rightarrow C\}$ → then $A \rightarrow C$ also holds.

Date _____ Page _____

$F^C = \{ A \to B,\ B \to C,\ A \to C \}$.

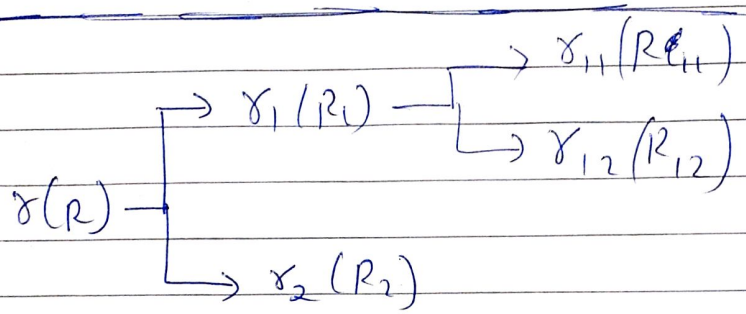
$R_1 \cap R_2 = B$.

$B \to C$ holds as it is given.

$B \to AB$: $(R_1 \cap R_2) \to R_1$

$B \to BC$: $(R_1 \cap R_2) \to R_2$

This is a lossless decomposition



* Functional dependency theory Road Map

How to compute F^+ for some given F?

Armstrong's Axioms

- Reflexive rule: If $\beta \subseteq \alpha$ then $\alpha \to \beta$ hold (Trivial functional dependency)
- Augmentation Rule: If $\alpha \to \beta$ hold & γ is another set of attribute then $\gamma\alpha \to \gamma\beta$ also holds.
- Transitivity Rule: If $\alpha \to \beta$ hold & $\beta \to \gamma$ holds, then $\alpha \to \gamma$ also hold.

Date Page

These rules are sound. Means they generate only functional dependencies that actually hold.

These rules are complete. Means they generate all functional dependencies.

* Let $\gamma(R)$.

$F \longrightarrow F^+$

Let F^A Actual functional dependencies which are holding on γ for given F.

Sound does not mean this. $F^+ \subset F^A$ ✗✗

Complete also does not mean this F

Eg. $\gamma(A, B, C, G, H, I)$

$F = \{A \to B,\ A \to C,\ CG \to H,\ CG \to I,\ B \to H\}$

Sol:→ $A \to B$ ... (1)

$A \to C$ ... (2)

$CG \to H$ ... (3)

$CG \to I$ ... (4)

$B \to H$ ... (5)

· Applying transitivity on (1) & (5)

$A \to H$ also holds → (6)

* Augement G on both sides

$AG \rightarrow CG$ ⟹ (7)

* Applying transitivity on (7) & (4)

$AG \rightarrow I$ ⟹ (8)

* Applying transitivity on (7) & (3)

$AG \rightarrow H$ --- (9)

* Augementing G on both sides of (4)

$CG \rightarrow CGI$ ..... (10)

* Augementing I on both sides of (3)

$CGI \rightarrow HI$ ..... (11)

* Applying transitivity on (10) & (11)

$CG \rightarrow HI$ ..... (12)

Additional Rules :-

- Union Rule → if $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds then $\alpha \rightarrow \beta\gamma$ holds.

Date Page

- Decomposition Rule → if $\alpha \rightarrow \beta\gamma$ holds, then $(\alpha \rightarrow \beta)$ and $(\alpha \rightarrow \gamma)$ holds.

- Pseduo transitivity Rule → if $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds.

* Proof of Union Rule.

* $\alpha \rightarrow \beta$ holds.
So augementing α on both sides.

$\alpha \rightarrow \alpha\beta$ ..... (1*)

* $\alpha \rightarrow \gamma$ holds.
So augementing β on both sides.

$\alpha\beta \rightarrow \beta\gamma$ ..... (2*)

Applying transitivity on (1*) & (2*)

$\boxed{\alpha \rightarrow \beta\gamma}$ also holds

* Proof of decomposition Rule

(3**) ... $\alpha \rightarrow \beta\gamma$ holds {given}.

(3*) ... $\beta\gamma \rightarrow \beta$ also holds {by reflixive}

(4*) ... $\beta\gamma \rightarrow \gamma$ also holds {by reflexive}.

from transitivity. from (3** & 3*) & (3** & 4*)

$\alpha \rightarrow \beta$ holds & $\alpha \rightarrow \gamma$ also holds.

Date Page

* Proof of Pseudo transitivity

$\alpha \rightarrow \beta$ holds $\rightarrow$ (1) {given}

$\gamma\beta \rightarrow \delta$ holds ... (2) {given}

To show $\alpha\gamma \rightarrow \delta$

Sol:→ Augmenting γ on both sides of (1)

$\alpha\gamma \rightarrow \beta\gamma$ ... (3)

Transitivity (or) (2) & (3)

$\alpha\gamma \rightarrow \delta$

Input: $r(R)$ & $F \rightarrow$ given set of FD

Output: F^c

For each FD, $F \in F^+$, we will apply reflexivity & augementation rule on it. F and update the resend list with new set of Functional dependencies

For each pair of FD, $F_1, F_2 \in F^c$

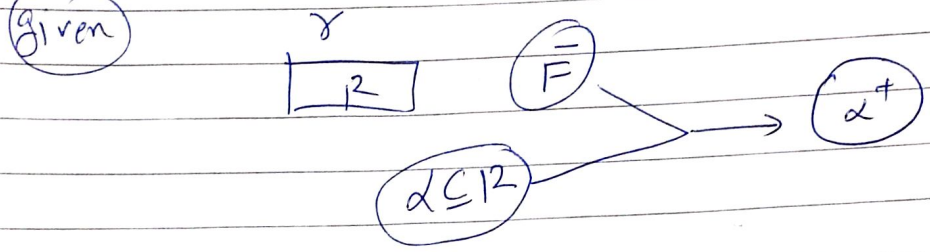
if F_1 & F_2 can be combined using transitivity then add the resultant FD to F^+

↳ Repeat the process until F^+ does not change.

27/10/2021

* Closure of Attribute Set

Given a set of attributes α and a set of FD, the closure of α under $F(\alpha^+)$ as the set of attribute that are functionally determined by α under F.



Given a set of attributes α, the closure of α with respect to functional dependency F is denoted as set of attributes that are functionally determined by α.

The closure of α is denoted by α^+

Eg. γ (A, B, C, G, H, I)

R.

$$F = \{ A \to B,\ A \to C,\ CG \to H,\ CG \to I,\ B \to H \}$$

$\alpha = \{A, G\}$, Now we want to compute α^+.

Date _____ Page _____

Sol:→

$\alpha = \{A, G\} \longrightarrow \alpha^+ = \{A, G\}$

$\downarrow A \to B,\ A \to C$

$\alpha^+ = \{A, B, C, G\}$

$\downarrow B \to H$

$\alpha^+ = \{A, B, C, G, H\}$.

$\downarrow CG \to I$

$\alpha^+ = \{\underbrace{A, B, C, G, H, I}_{R}\}$

$\alpha^+ = R$

The set of attributes in α^+ is equal to set of attributes of F.

Q. is AG a candidate key?

A G	B C H I

Sol:- Let us take closure of A^+

$A^+ = \{A\}$,

$\downarrow A \to B, A \to C$

$= \{A, B, C\}$

$\downarrow B \to H$

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

Date Page

$A^+ \neq R$

$\therefore A^+ = \{A\}$

$\therefore$ AG is a candidate key

Candidate key → Minimal super key is candidate key.

* Algorithm for computing Attribute closure.

Input: $\gamma(R)$, $\alpha \subseteq R$ & F

Output: α^+

code

```
result ← α
while (changes to result) do.
    for each β → γ ∈ F
        if β ⊆ result then
            result ← result ∪ γ
    endl endl
end;
```

* Use of attribute closure

① Testing for superkey :-

: To test α is a superkey we compute α^+ & checks whether it is equal to R or not. if it is equal to R, then it is a superkey.

② Testing for functional dependency :-

$\gamma(R)$ F $\alpha \rightarrow \beta$

→ Check whether $\beta \subseteq \alpha^+$ ↳ compute α^+

Check whether $\alpha \rightarrow \beta$ holds on γ or not.

③ Computing closure of $F(F^+)$

For each $\gamma \subseteq R$, we just find the γ^+ and to each $S \subseteq \gamma^+$, we get FD, $\gamma \rightarrow S$.

* Canonical cover of given set of FD

Database (at t=0) → tuples → after updation → Database (at t=1)

if F is very very large, can we reduce F??

$F' \subseteq F$

F' → canonical cover of F

$(F')^+ = (F^+)$

Here F' is of smaller size

Date Page

* Extraneous Attribute. (in F)

→ An attribute of FD, is said to be extraneous if we can remove that attribute without changing F^+.

$R(R), F$.

$R(A, B, C, D)$.

$F.D = \{ AB \rightarrow C, A \rightarrow C \}$. strong FD.

if $AB \rightarrow C$ holds, it does not imply $A \rightarrow C$ holds.

Q.

$F = \{ \underbrace{AB \rightarrow C}_{①}, \underbrace{A \rightarrow D}_{②}, \underbrace{D \rightarrow C}_{③} \}$

By applying transitivity on ② & ③.
$A \rightarrow C$.

these is no need of B in $AB \rightarrow C$.

So we can say that B is an extraneous attribute.

* Removing an attribute from left part of FD means that FD becomes stronger.