Module II

RELATIONAL MODEL

Relational Model

- Relational Model
  - Developed by Ted Codd in 1970s
  - Based on mathematical model
  - Set theory

- Relational Model Notations
  - Table – Relation
  - Column – attribute
  - Row – tuple
  - Values of the column- domain

- Important Formulas

- R(A1,A2,...An)
- Dom(A1) –domain of A1
- r(R) –relation state or extension
- At any instance r(R) is a subset of Cartesian product of the domain of attributes of R.
- $r(R) \subseteq dom(A1) \times dom(A2) \times dom(A3) \times \ldots$

- Properties of a relation:
  - No order of tuples
  - No order of Attributes
  - Atomic values of attribute
  - Interpretation of relation

Regno	Name	Prog	Bran	Year
17BIT	CRA	BTech	IT	2017
20MIT	NCS	MTech	SE	2020
21MIT	MPL	MTech	SE	2021
22BIT	GIL	BTech	CSE	2022
23MCA	ERJ	MCA	CA	2023

- Relational Model Constraints
  - Domain
    - Size
    - Data type
    - Check

Datatype	size	check
Varchar(20)	20	Check (name in ('score','site','scope'))
Number(3)	3	Check (rollno between 11 and 20)
date	7	Check (dob like '%jul%')

  - Key
    - Entity Integrity constraint -Primary key should not be null
    - Referential Integrity constraint- FK can be null or subset of PK
    - Key constraint -unique

Relational Model Constraints

- Primary Key: The attribute which uniquely identifies a tuple in a relation.

  → Value of the attribute should not be duplicated or be null.
- Foreign Key: The attribute which refers to the key attribute of another relation.

  → The attribute value should be a subset of the referencing key attribute or can be null.

- Keys
  - Primary key – only one, a column or a set of columns which uniquely identifies a tuple in a relation
  - Foreign Key – any number , a column which refers another table column, which exhibits the relationships
  - Candidate key – any number, a column or a set of columns which uniquely identifies a tuple in a relation
  - Super Key - set of columns which uniquely identifies a tuple in a relation , all columns is the default super key.

- Hence, a key satisfies two properties:

  1. Two distinct tuples in any state of the relation cannot have identical values for (all) the attributes in the key. This first property also applies to a superkey.

  2. It is a *minimal superkey*—that is, a superkey from which we cannot remove any attributes and still have the uniqueness constraint in condition 1 hold. This property is not required by a superkey.

- Hence, a key is also a superkey but not vice versa.
- STUDENT relation …. {Ssn} is a key of STUDENT

  because no two student tuples can have the same value for Ssn.
- Any set of attributes that includes Ssn—for example, {Ssn, Name, Age}—is a superkey.
- However, the superkey {Ssn, Name, Age} is not a key of STUDENT because removing Name or Age or both from the set still leaves us with a superkey.
- In general, any superkey formed from a single attribute is also a key.
- A key with multiple attributes must require *all* its attributes together to have the uniqueness property.

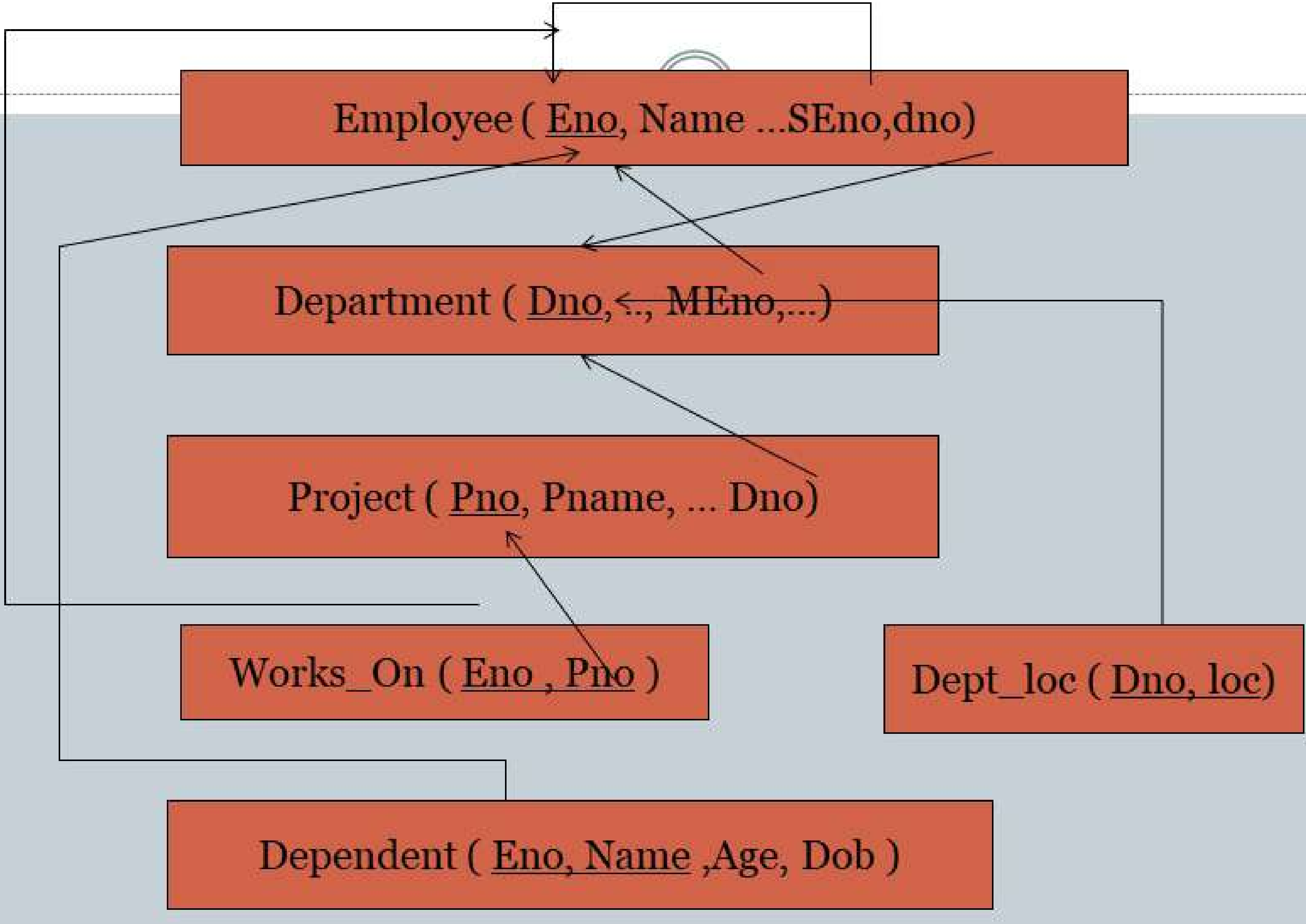
ERPNO	Name	DOB	Desg	PAN	School
10236	Rani	24/09/75	AO1	AHCPR3238G	SCORE
10237	Ajay	17/06/99	AO1	AHCPG8900R	SCORE
10238	Poun	02/02/80	AO1	AHPRG2899B	SCORE
					SCOPE
					SCOPE

Code	Name	Dean	Location
SITE	School of Information Technology and Engineering	Dr.Sumathy	SJT
SCOPE	School of computer Science and Engineering	Dr.Ramesh Babu	SJT
SMEC	School of Mechanical Engineering	Dr. Devandranath Ramkumar	MB

- Constraint violations
  - Insert
    - EIC – when PK has null value
    - RIC – when FK is not subset of PK
    - Key constraint – when a unique col gets duplicate
    - Domain – data type mismatch or size violated
  - Delete
    - RIC – when referred PK is deleted
  - Update – Insert +delete
    - So all

Converting an ER Model to Relational Model

- **Strong entity** – Create new relation, Key attribute (Primary key)
- **Weak entity** - Create new relation, Partial key attribute + Parent key attribute (Primary key)
- ***1:1 Relationship type*** – Add the key attribute as foreign key to total participation side.
- **1:N Relationship type** - Add the key attribute of 1 side as foreign key to N side.
- **M:N Relationship type** – Create a new relation, add M and N side key attributes as a composite primary key.
- **Multivalued or complex attribute** – Create a new relation, add entity type's key attribute and multivalued attribute as primary key



Employee (Eno, Name ...SEno,dno)
Department (Dno, .., MEno,...)
Project (Pno, Pname, ... Dno)
Works_On (Eno , Pno)
Dept_loc (Dno, loc)
Dependent (Eno, Name ,Age, Dob)

Bank Relational Schema

Bank(code, name,address)

PK

Bank_branch(address, branchno, code)

FK

Account(Accno, balance, type, brno, code)

FK

Loan(Loanno, balance, type,brno,code)

FK

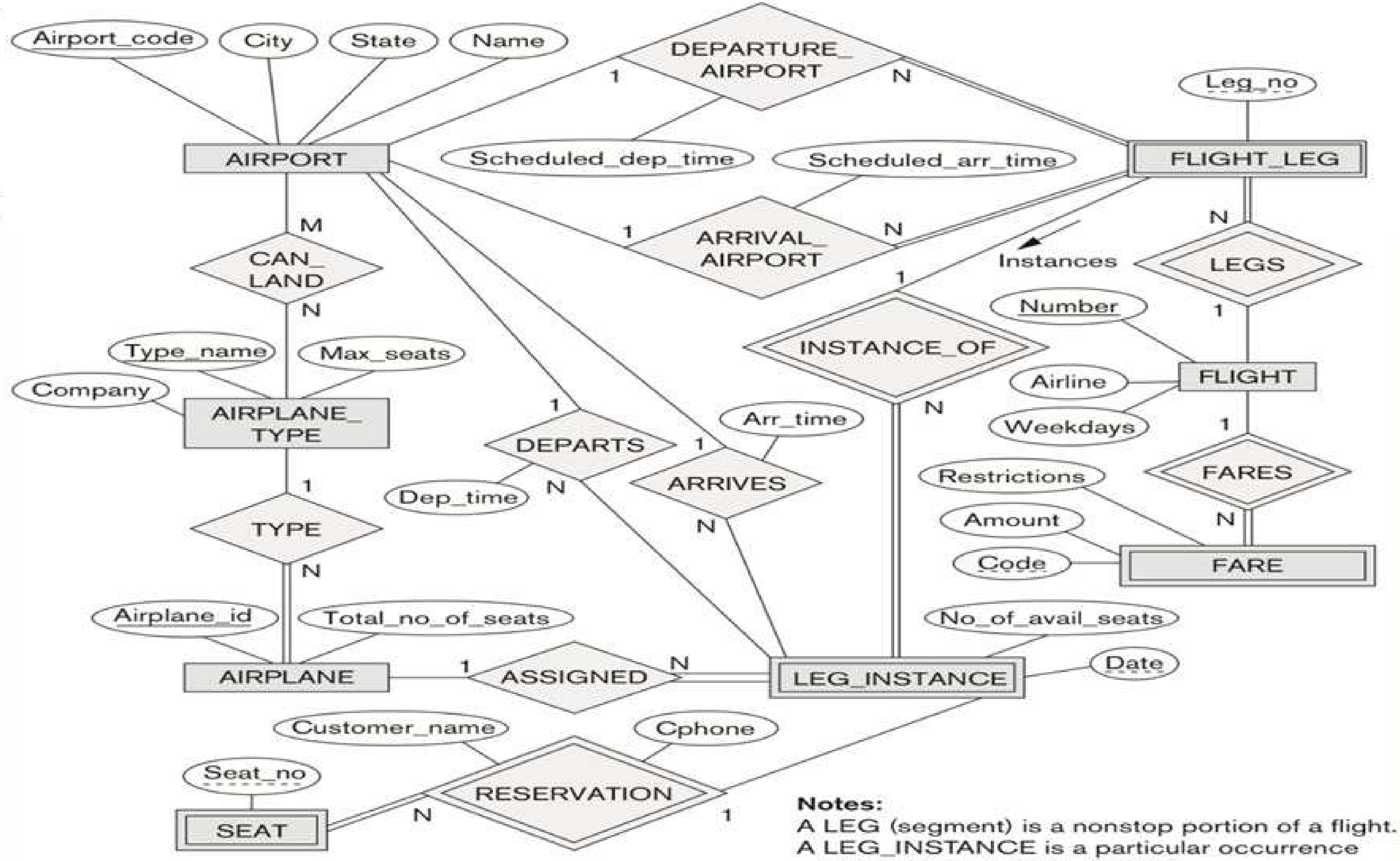
Customer(SSN,Phno,Name, Address)

Cus_Acc(SSN, Accno)

FK FK

Cus_Loan(SSN,Lno)

FK FK



Notes:
A LEG (segment) is a nonstop portion of a flight.
A LEG_INSTANCE is a particular occurrence of a LEG on a particular date.

AIRPORT

Airport_code	Name	City	State

FLIGHT

Flight_number	Airline	Weekdays

FLIGHT_LEG

Flight_number	Leg_number	Departure_airport_code	Scheduled_departure_time	Arrival_airport_code	Scheduled_arrival_time

LEG_INSTANCE

Flight_number	Leg_number	Date	Number_of_available_seats	Airplane_id	Departure_airport_code	Departure_time	Arrival_airport_code	Arrival_time

FARE

Flight_number	Fare_code	Amount	Restrictions

AIRPLANE_TYPE

Airplane_type_name	Max_seats	Company

CAN_LAND

Airplane_type_name	Airport_code

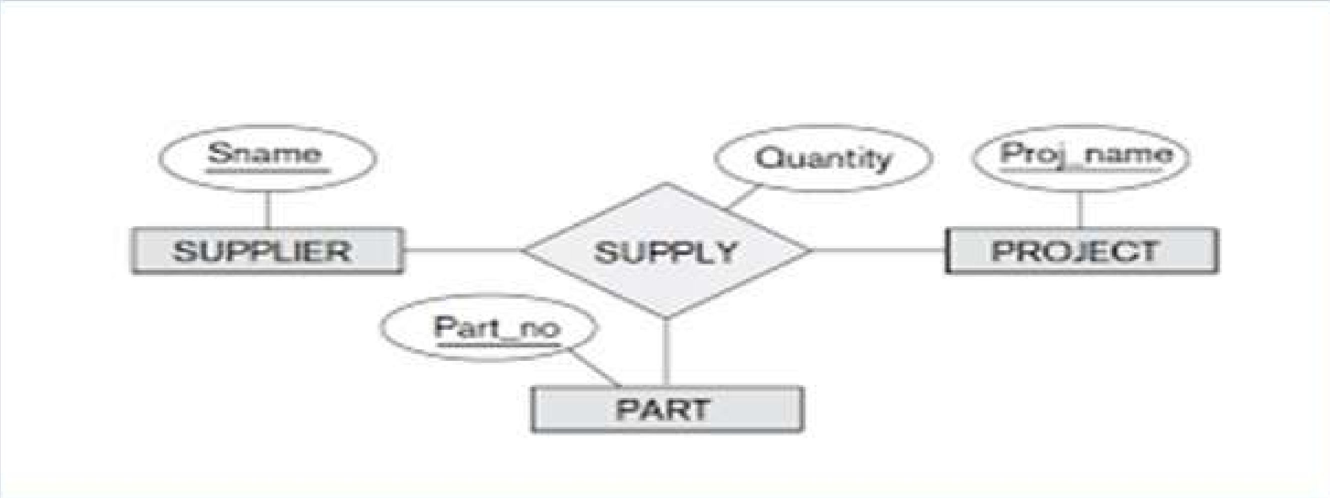
AIRPLANE

Airplane_id	Total_number_of_seats	Airplane_type

SEAT_RESERVATION

Flight_number	Leg_number	Date	Seat_number	Customer_name	Customer_phone

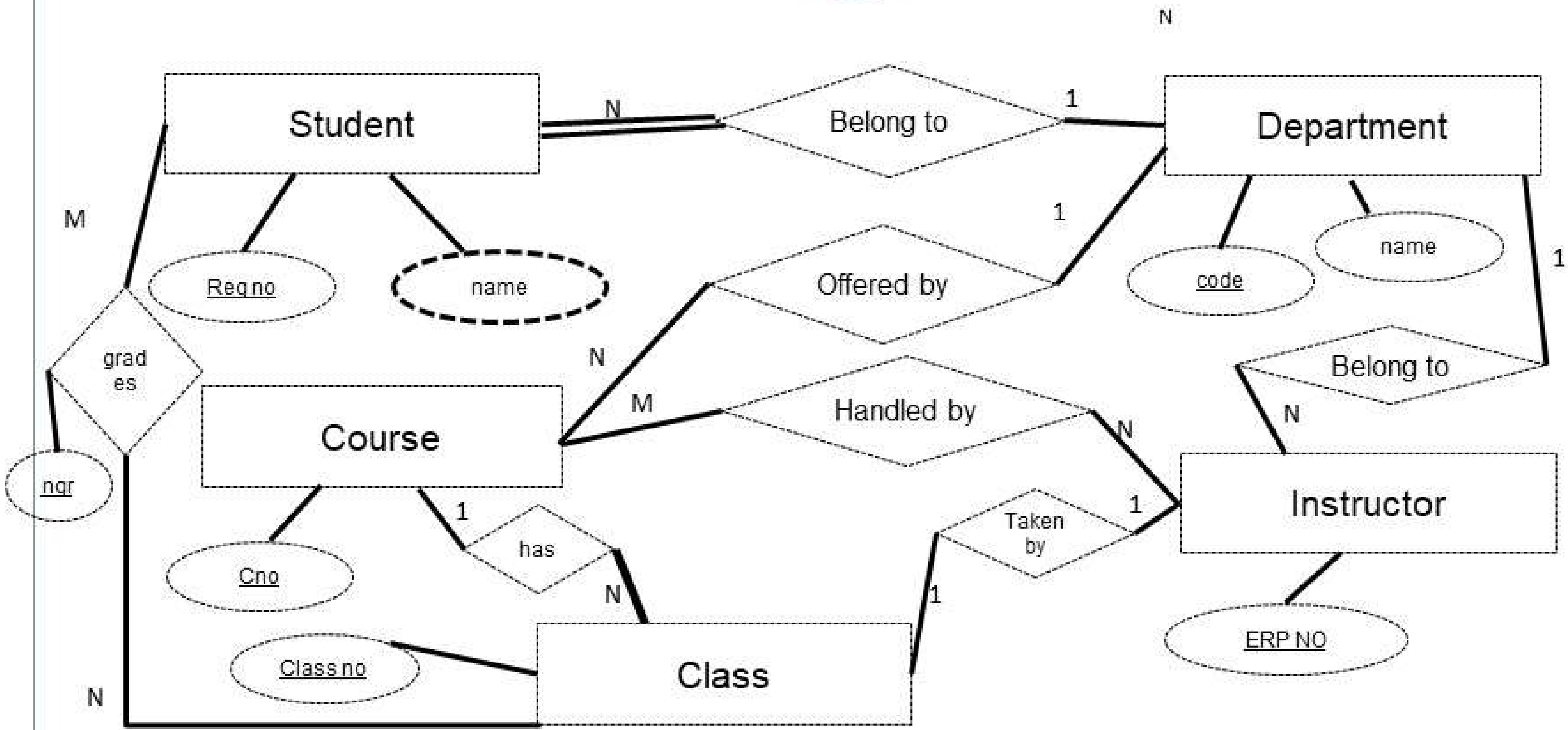
Ternary relationship



Create a new relation, add all side PK as FK. Make the FKs as composite PK. If any attribute in relationship add it.

Supply(<u>sname, pno, pjname</u>, quantity)

University Database : ER Schema



University Database : Relational Schema

Student(Regno ,Fname, Lname, Dcode)
FK

Department(Code, Name,)

Course(Code, Title, Credits, Offdno)
FK

Instructor(Eno, Name, Qualification, dno)
FK

Class(Cno, slot, Venue, Ino, code)
FK FK

Course_handledby(ccode, Ino)
FK FK

Grades(Srno, Cno, lga, Nga)
FK FK

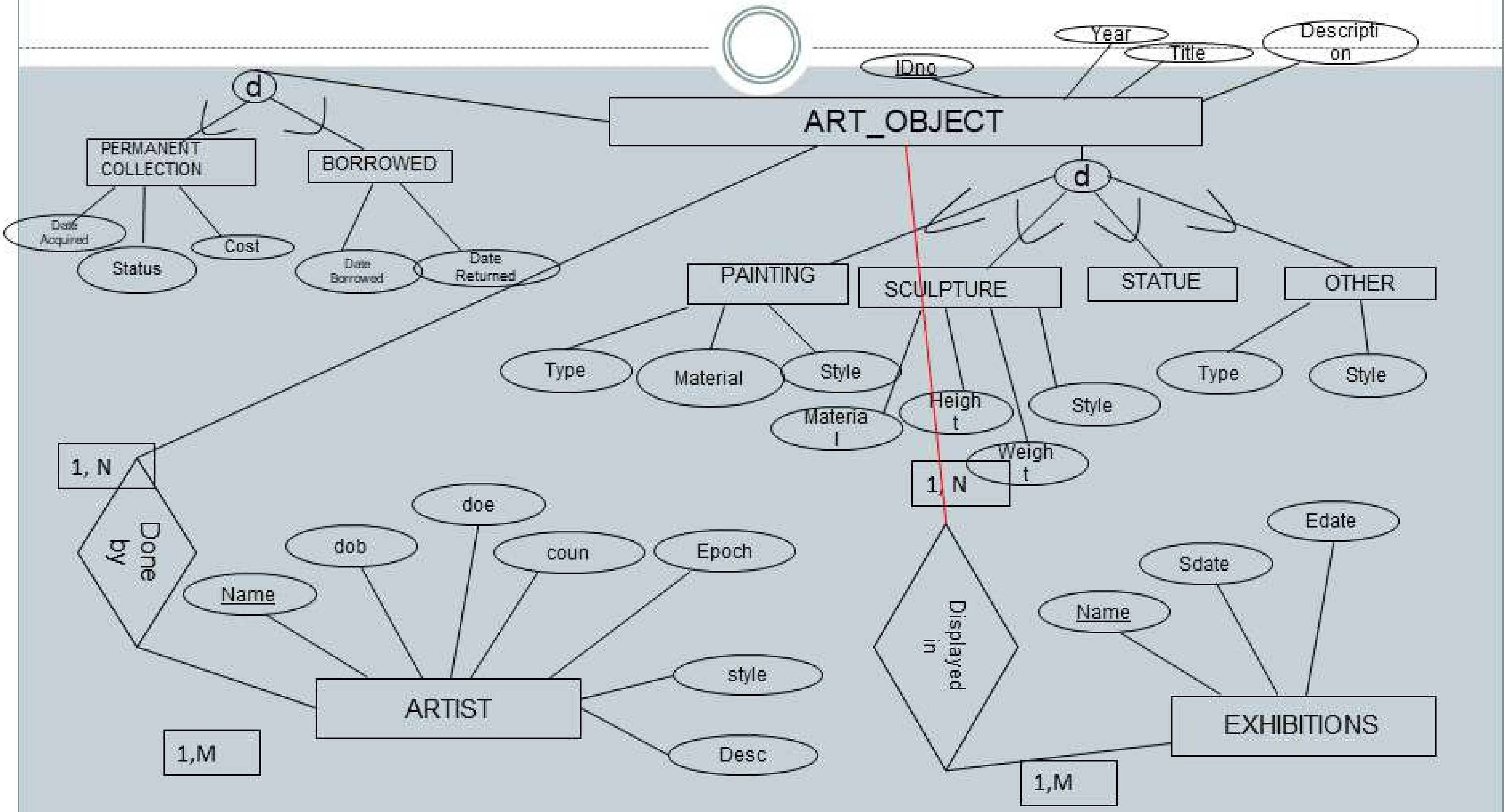
EER to Relational

1. Create a relation for super class and include all the attributes , add a attribute for subclass (if subclass has less attributes or no attribute) Make the key attribute as PK.

Or

1. Create a relation for each subclass and super class and add the attributes . Make the super class key attribute as PK in all the relations of super and subclass.

Museum DB : EER Schema



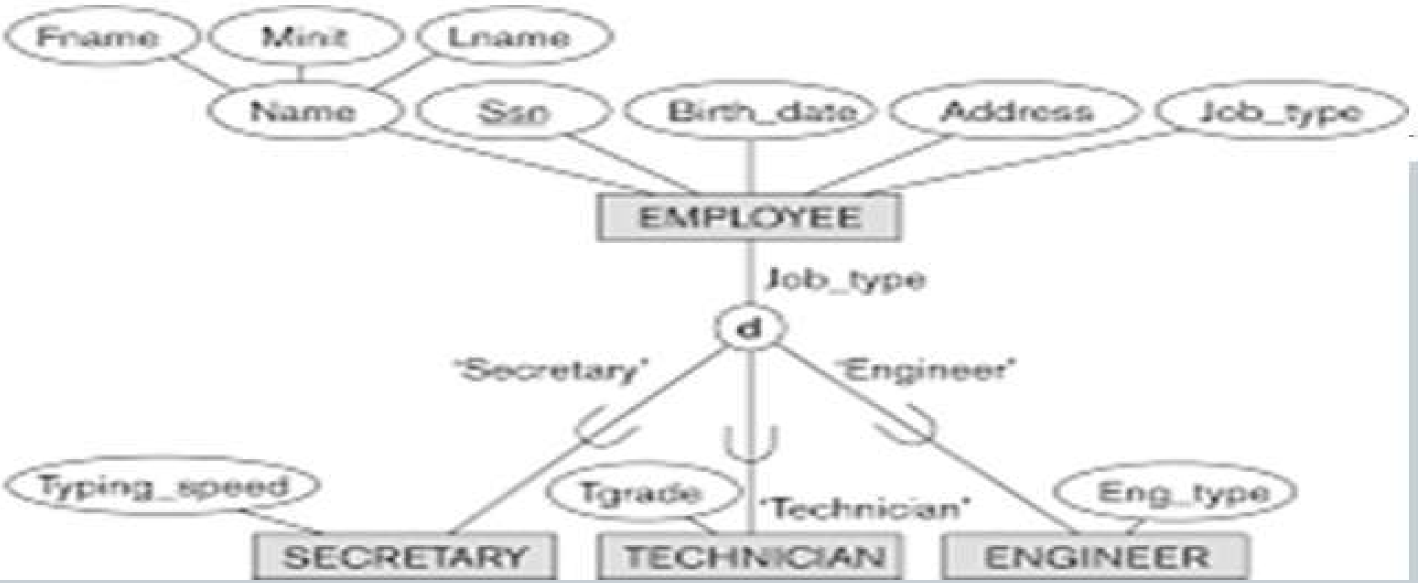
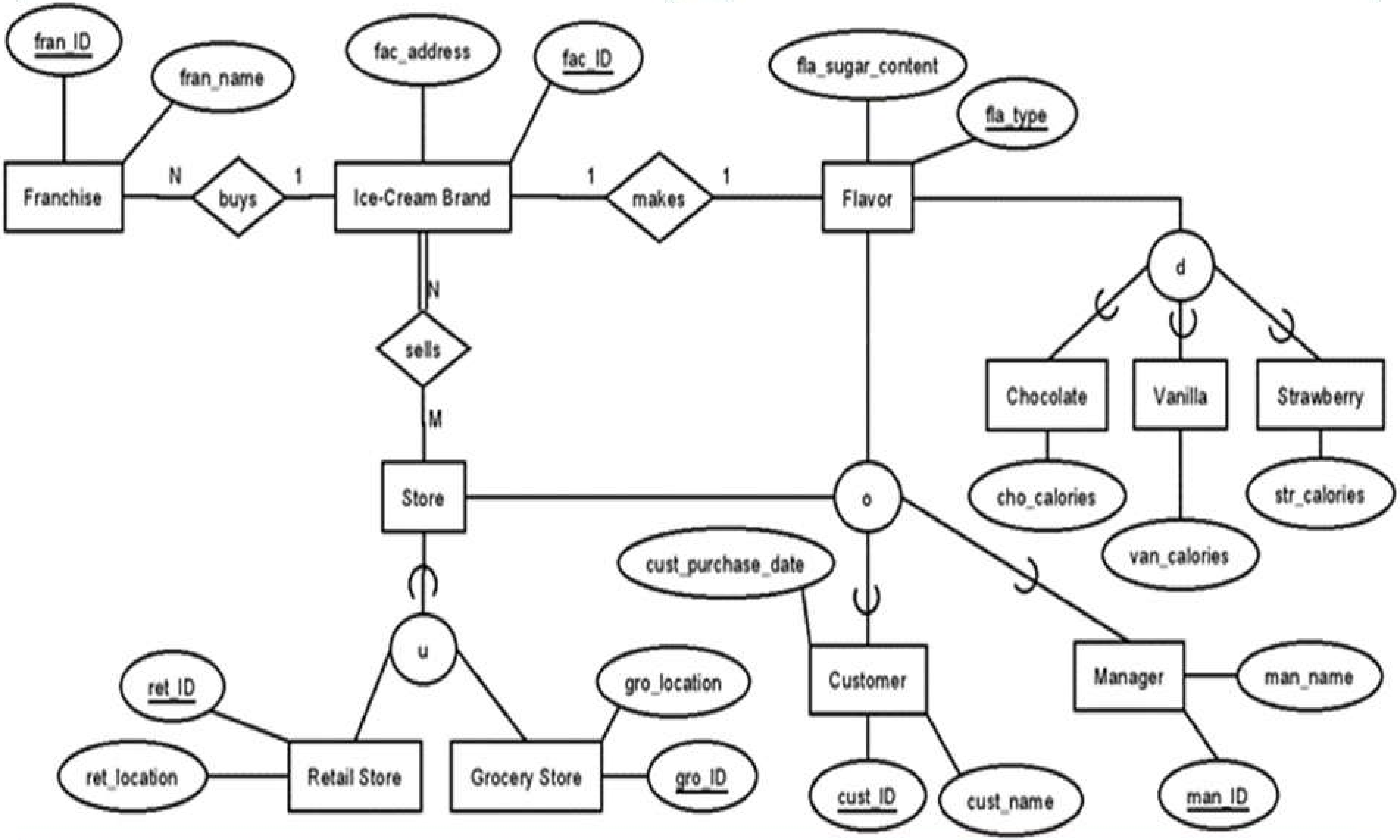


Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.

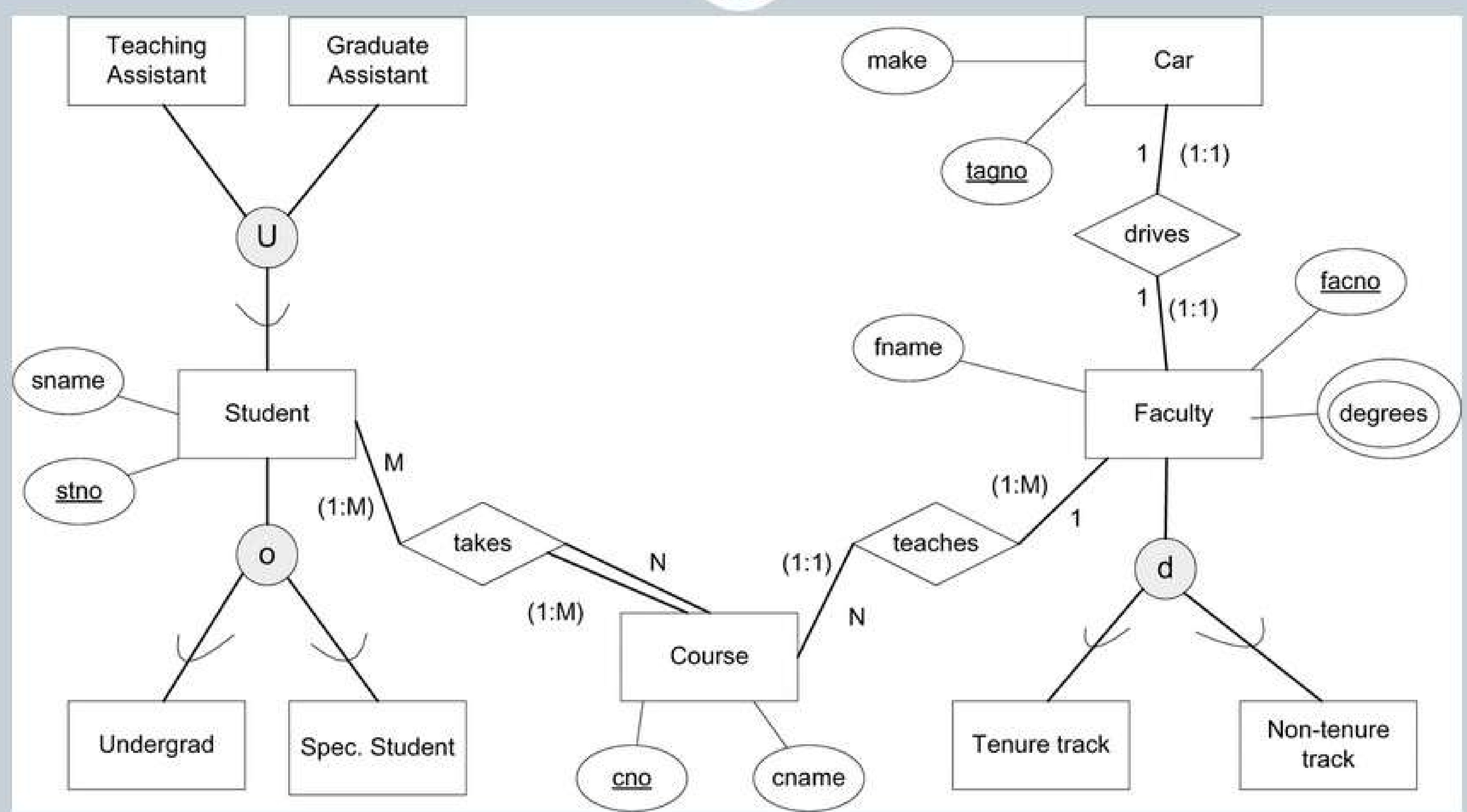
Employee(SSN, Fname, Mint, Lname , DOB, Address, Job_type, typing_speed, Tgrade, Eng_Type)

1. Create one relation for the super class , add the attributes and attributes in subclass.
2. Remaining mapping of weak entity, ratios and other attributes(complex or multivalued) will be same as the ER to Relational steps.
3. The attributes of the subclass will be filled only for the corresponding rows.

Convert EER to relational



Convert EER to Relational



Relational Algebra

C.RANICHANDRA

Outline

- Algebra- Expression for retrieving data from relations
- Expression – collection of operations
- Set Theoretic operations
  - union, intersection, difference, Cartesian product
- Relational operations
  - project, select, rename, join, division

Set Theoretic Operations

- Set Operators
  - union
  - intersection
  - difference
  - Cartesian Product
- Set operators are *binary* and will only work on two relations or sets of data.

Set Theoretic Operators (cont'd)

- Can only be used on **union compatible** sets
  - $R(A_1, A_2, \ldots, A_N)$ and $S(B_1, B_2, \ldots, B_N)$ are union compatible if:
    - degree (R) = degree (S) = N
    - domain (A_i) = domain (B_i) for all i

Union (∪)

- Assuming R & S are union compatible…
  - union: **R** ∪ **S** is the set of tuples in either **R** or **S** or **both**.
  - since it is a set, there are no duplicate tuples

Example of Union

R(fName,age) S(lname,age)

R =

Rani	34
Kumar	32
Raj	21

S =

Kumar	32
Sekar	42

$R \cup S = ???$

$S \cup R = ???$

Rani	34
Kumar	32
Raj	21
Sekar	42

Intersection (∩)

- Assuming that R & S are union compatible:
- intersection: $R \cap S$ is the set of tuples in **both** R and S
- Note that $R \cap S = S \cap R$
- Example: use R and S from before...

Kumar	32

Difference (-)

- Difference: R – S is the set of tuples that appear in R but do not appear in S
- Is (R – S) = (S – R) ???
- Example: R – S

Rani	34
Raj	21

Sekar	42

Cartesian Product (X)

- Cross of two sets
- In general, the result of:

$R(A_1, A_2, \ldots, A_N)$ **X** $S(B_1, B_2, \ldots, B_M)$ is

$Q\ (A_1, A_2, \ldots, A_{N,}\ B_1, B_2, \ldots, B_M)$

- If R has C tuples and S has D tuples, the result is C*D tuples.

Rani	34	Kumar	32
Rani	34	Sekar	42
Kumar	32	Kumar	32
Kumar	32	Sekar	42
Raj	21	Kumar	32
Raj	21	Sekar	42

Example

- Employee(SSN,Name,age)   Dependent(SSN,Name,age)

1	Rani	45
2	Uma	46
3	Maha	55

1	Harrsha	14
1	Brindha	12
2	Akshaya	11

- Find the employee who have dependent
  - Employee(SSN) ∩ Dependent(SSN)
- Find the employee who do not have dependent
  - Employee(SSN) – Dependent(SSN)
- Find all the names from DB
  - Employee(Name) ∪ Dependent(Name)

Relational Operations

- developed specifically for relational databases
- used to manipulate data in ways that set operations can't
  - select
  - project
  - join
  - division

Selection Operation

- Used to select a subset of the tuples
- Selection is based on a "select condition"
- The selection condition is basically a filter

- Notation: $\sigma_{\text{<condition>}}(\text{<Relation>})$

Selection (cont'd)

- To process a selection, we:
  - look at each tuple
  - see if we have a match (based on the condition)
- The degree of the result is the same as the degree of the relation $|\sigma| = |r(R)|$

Selection Example

- Faculty (<u>fnum</u>, name, office, salary, rank)

12345	Darcy	Car409	21000	lecturer
23456	Bob	Bac100	23000	associate
34567	Mary	Car301	27000	associate
45678	Jane	Irv342	32000	full

Selection Example (cont'd)

1. $\sigma_{\text{salary} > 27000}(\text{Faculty})$

2. $\sigma_{\text{rank} = \text{associate}}(\text{Faculty})$

3. $\sigma_{\text{fnum} = 34567}(\text{Faculty})$

Selection Example (cont'd)

4. $\sigma_{(salary>26000)\ and\ (rank=associate)}$ (Faculty)

5. $\sigma_{(salary<=26000)\ and\ (rank\ !=associate)}$ (Faculty)

6. $\sigma_{max(salary)}$ (Faculty)

Selection (cont'd)

- For the condition
  - any combination of expressions that can be resolved to a boolean value with the relation is okay
- For the relation
  - any relational expression that resolves to a relation is okay

Project Operation (π)

- A select filters out rows
- A ***project*** filters out columns
  - reduces data (columns) returned
  - reduces duplicate columns created by cross product (why?)
  - creates a new relation

Project (cont'd)

- Notation: $\pi_{\text{<attribute list>}}$ (Relation)
- The degree of the result is the number of attributes in the <attribute list> of the project.
- $| \pi | <= | r(R) |$

Project Example

Faculty (fnum, name, office, salary, rank)

12345	Darcy	Car409	21000	lecturer
23456	Bob	Bac100	23000	associate
34567	Mary	Car301	27000	associate
45678	Jane	Irv342	32000	full

1. $\pi_{\text{name, office}}(\text{Faculty})$
2. $\pi_{\text{fnum, salary}}(\text{Faculty})$

Rename (←)

- Used to give a name to the resulting relation
- Notation to make relational algebra easier to write and understand
- We can now use the resulting relation in another relational algebra expression
- Notation:

<New Name> ← <Relational Expression>

Rename Example

Faculty (<u>fnum</u>, name, office, salary, rank)

12345	Darcy	Car409	21000	lecturer
23456	Bob	Bac100	23000	associate
34567	Mary	Car301	27000	associate
45678	Jane	Irv342	32000	full

Associates $\leftarrow \sigma_{\text{rank = associate}}(\text{Faculty})$

Result $\leftarrow \pi_{\text{name}}$ (Associates)

Join Operation

- Join is a commonly used sequence of operators
  - Take the Cartesian product of two relations
  - Select only related tuples
  - (Possibly) eliminate duplicate columns

Join Example

R =

dcode	Number
COMP	555-1111
HIST	555-2222

S =

code	office
COMP	CAR309
HIST	BAC333

$$R1 \leftarrow R \times S$$

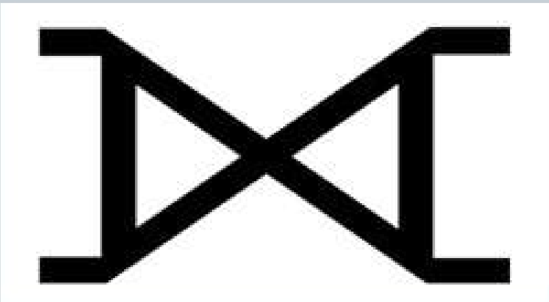
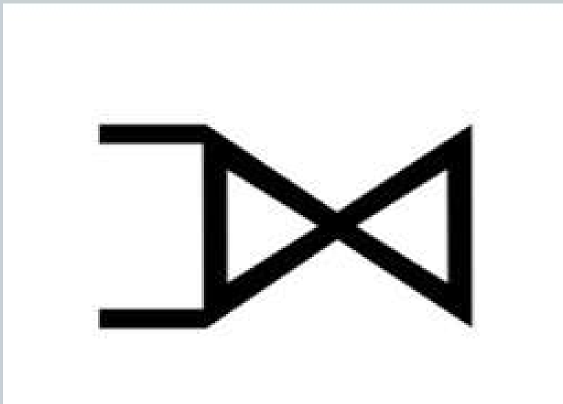
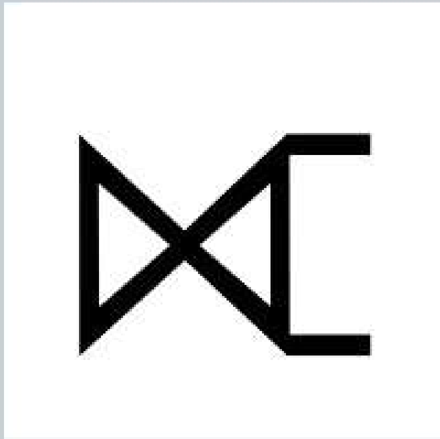
$$R2 \leftarrow \sigma_{\text{dcode = code}}(R1)$$

$$Result \leftarrow \pi_{\text{code, office, number}}(R2)$$

Join Example (cont'd)

- You could do all of that, or you could do a **join**

- $\text{Result} \leftarrow R \bowtie_{\text{dcode = code}} S$

Kinds of Joins

- There are 3 different kinds of joins
  - **Theta join**: A join with some condition specified **R** $\bowtie$ a>b **S**
  - **Equijoin**: A join where the only comparison operator used is "="

    R $\bowtie$ a=b **S**

    - Most common since most joins link together related tuples using a foreign key
    - $\bowtie$

Kinds of Joins (cont'd)

- **Natural join**
  - A Natural join is denoted by (*)
  - Standard definition requires that the columns used to join the tables have the same name

Types of Joins(outer)

- Left Outer Join
  - keep all of the tuples from the “left” relation
  - join with the right relation
  - pad the non-matching tuples with nulls
- Right Outer Join
  - same as the left, but keep tuples from the “right” relation
- Full Outer Join
  - same as left, but keep all tuples from both relations

Left Outer Join

R =

name	phone
Rani	999
Senthil	666

S =

name	email
Rani	cra@vit
Dhana	dan@vit

- If we do a left outer join on R and S, and we match on the first column, the result is:

name	phone	email
Rani	999	cra@vit
Senthil	666	-

Right Outer Join

R=

name	phone
Rani	999
Senthil	666

S=

name	email
Rani	cra@vit
Dhana	dan@vit

- If we do a right outer join on R and S, and we match on the first column, the result is:

name	email	phone
Rani	cra@vit	999
Dhana	dan@vit	---

Full Outer Join

R =

name	phone
Rani	999
Senthil	666

S =

name	email
Rani	cra@vit
Dhana	dan@vit

name	phone	email
Rani	999	cra@vit
Senthil	666	-
Dhana	-	dan@vit

Complete Set of Relational Algebra Operators

- set of operations has been shown that {σ, π, ∪, –, X} is a **complete**.
- Any other relational algebra operations can be expressed as a sequence of operations from this set.

Aggregate Functions

- Summarize column information like min, max, avg, count, sum
- Given by the syntax

  $\mathcal{F}_{<functionname>\ \ <columnnanme>}(\text{Relation})$
- Example:
- $\mathcal{F}_{\text{MIN Salary}}$ (EMPLOYEE)
- $\mathcal{F}_{\text{SUM Salary}}$ (EMPLOYEE)
- $\mathcal{F}_{\text{COUNT eno}}$ ($\sigma_{\text{eno}=4}$(DEPENDENT))

Grouping with Aggregation

- Grouping can be combined with Aggregate Functions
- Example: For each department, retrieve the DNO, number of employees , and AVERAGE SALARY
- A variation of aggregate operation $\mathcal{F}$ allows this:
  - Grouping attribute placed to left of symbol
  - Aggregate functions to right of symbol

Example: $\text{DNO}\ \mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}\ (\text{EMPLOYEE})$

Grouping Example

SSN	Salary	dno
1	900	111
2	700	222
3	200	111
4	100	222
5	250	333
6	600	333
7	500	333

Dno	Avg	Count
111	550	2
222	400	2
333	450	3

Division operator

- Suited to queries that include the phrase "**for all**".
- Let R and S be two relations
  - R (Z), Z = a1,a2,a3...
  - S (X), X= b1,b2,b3..

The result of R ÷ S is a relation

T(Y) , where Y= Z-X X is subset of Z

A tuple t appears in T , if the value in tuple t in R is found with combination of every tuple of S

Example division

R(A,B) ÷ S(A) = Y(B)

A	B
1	P
2	P
3	P
1	Q
2	R
3	S

A
1
2
3

B
P

Division Example

Find the employees who work on all projects that 'John Smith' works

$R1 \leftarrow \pi_{eno}(\sigma_{fname='John' \text{ and } Lname='smith'}(EMPLOYEE))$

$R2 \leftarrow \pi_{pno}(WORKS\_ON) * R1$

$R3 \leftarrow \pi_{pno,eno}(WORKS\_ON)$

$R4 \leftarrow R3 \div R2$

$R5 \leftarrow R4 \bowtie_{eno=E.eno} EMPLOYEE\ E$

RA-Exercise 1

Consider the following relations:

Student (<u>regno</u>, name, address, major)

Course (<u>code</u>, title)

Registered (<u>regno, code</u>)

RA-Exercise 1

1. List the codes of courses in which at least one student is registered.

$$\pi_{code}\ (\text{Course}) \cap \pi_{code}\ (\text{Registered})$$

2. List the titles of registered courses.

$$\pi_{\ title}\ (\text{Registered}\ \bowtie_{\ code=C.code}\ \text{Course C})$$

3. List the codes of courses for which no student is registered.

$$\pi_{code}\ (\text{Course}) - \pi_{code}\ (\text{Registered})$$

4. The titles of courses for which no student is registered

$$\pi_{\ title}\ (\text{R1}\ \bowtie_{\ code=C.code}\ \text{Course C})$$

Exercise-cont..

5.Names of students and the titles of courses they are registered to.

$$\pi_{\text{title, name}} (\text{Student} * \text{Registered} * \text{Course})$$

6. Regno of students who are registered for 'Database Systems' or 'Analysis of Algorithms'.

$$\pi_{\text{regno}} (\sigma_{\text{title} = \text{'DB'} \text{ or title} = \text{'AA'}} (\text{Registered} * \text{Course}))$$

7.Regno of students who are registered for both 'Database Systems' and 'Analysis of Algorithms'.

$$\pi_{\text{regno}} (\sigma_{\text{title} = \text{'DB'} \text{ and title} = \text{'AA'}} (\text{Registered} * \text{Course}))$$

Exercise-cont..

8. List of courses in which all students have registered.

$R1 \leftarrow \pi_{regno}$ (STUDENT)

$R2 \leftarrow \pi_{code,\ regno}$ (REGISTERED)

$R3 \leftarrow R2 \div R1$

R4 <- R3 * Course

9. List of courses in which all 'IT' Major students have registered.

$R1 \leftarrow \pi_{regno} (\sigma_{major='IT'}$ (STUDENT))

$R2 \leftarrow \pi_{code,\ regno}$ (REGISTERED)

$R3 \leftarrow R2 \div R1$

$R4 \leftarrow \pi_{code,\ title}$ $(R3 \bowtie_{code=C.code}$ COURSE C)

10. Find the number of students course code wise.

$_{code}\mathcal{F}$ code, count(*) (Registered)

RA-Exercise 1

11. List of all courses and register number if registered

$$(\text{Course } C \; ⟕_{C.code=R.code} \; \text{Registered } R)$$

12. List the course names registered, and all student details regno, names.

$$R1 \leftarrow (\text{Registered} * \text{Course } C)$$
$$R2 \leftarrow \pi_{\text{title, regno, name}} \; (R1 \ltimes_{\text{regno=regno}} \text{Student})$$

13. List the major wise no. students registered for each course.

$$_{\text{major, code}}\mathcal{F}\; \text{major, code, count(*)} \; (\text{Student*Registered})$$

Exercise 2

Person (pno, name, address)

111	Deepak	Vellore
222	Saras	Chennai
333	Uma	madurai

Car (Regno, year, model)

TN123	2008	I20
TN345	2008	Alto
TN789	2013	Duster

Owns (pno, car_no)

111	TN123
111	TN345
333	TN789

Accident (date, driver, Regno, damage, amount)

12-jan-09	Deepak	TN123	Bannet	15000
23-feb-10	Lalit	TN123	Glass	5000
31-dec-13	Uma	TN789	Bumper	9500

(a) Find the number of accidents in which the cars belonging to 'Deepak' were involved.

(b) Find the total number of people whose cars were involved in accidents in 2010.

(c) Print the accident details done by the owner of the car itself.

(d) Find the result of person left outer join Owns on pno=pno

(e) Find the person names who own all the cars (model) of the year 2008. Write the division result.