ADBMS_IA2_QB

(2M)

1. **What are 3 different types of relationships the subclasses corresponding to a superclass can have… explain with examples**
   Mapping of Binary 1:1 Relationship Types
   a. For each binary 1:1 relationship type
      i. Identify relations that correspond to entity types participating in *R*
   b. Possible approaches:
      i. Foreign key approach
      ii. Merged relationship approach
      iii. Cross-reference or relationship relation approach

- Mapping of Binary 1:*N* Relationship Types
  – For each regular binary 1:*N* relationship type
    - Identify relation that represents participating entity type at *N*-side of relationship type
    - Include primary key of other entity type as foreign key in *S*
    - Include simple attributes of 1:*N* relationship type as attributes of *S*
  – Alternative approach
  – Use the relationship relation (cross-reference) option as in the third option for binary 1:1 relationships

- Step 5: Mapping of Binary *M*:*N* Relationship Types
  – For each binary *M*:*N* relationship type
    - Create a new relation *S*
    - Include primary key of participating entity types as foreign key attributes in *S*
  - Include any simple attributes of *M*:*N* relationship type

2. **What are inabilities of ER Model how it can be resolved**

No relationship may be defined between an entity type and a relationship type •No relationship may be defined between an entity type and a collection of entity types

•No relationship may be defined between an entity type and a collection of entity types from which any one type may participate –e.g. Entity type1 : POLICY-HOLDER may be an individual, multiple individuals, one organization, or many organizations–Entity type2 : POLICY

Cannot incorporate Set-subset Relationships

• Cannot Incorporates Generalization Hierarchies

• Cannot define Constraints: like :–Coverage Constraints: partial vs. total–Disjointedness Constraint: disjoint vs. overlapping

It cannot talk about inheritance

Difficulties exist in defining complex relations of GIS, mulmedia databases

3. **Why Normalization is Needed**

- Normalization theory provides a mechanism for analyzing and refining the schema produced by an E-R design
- Normalization aids in Designing a schema that can be explained easily relation by relation. The semantics of attributes should be easily interpreted after normalization
- The complexities of relations will be easily sampleable
- Mixing attributes of multiple entities may cause problems and Information is stored redundantly wasting storage ….normalization easily aids in resolving these issues
- Update, Delete and Insert anomalies could be easily resolved using appropriate decomposition using normalization rules

4. **What is functional Dependency**

Definition: A functional dependency(FD) on a relation schema Ris a constraint X → Y, where X and Y are subsets of attributes of R, such that for every pair of tuples, tand s, in an instance of R, if every pair of tuples, tand s, in an instance of R, if t and s agree on all attributes in X then they must agree on all attributes in Y

This means that the values of the Y component of a tuple in r depend on, or are determined by, the values of the X component; alternatively, the values of the X com ponent of a tuple uniquely (or functionally) determine the

values of the Y compo nent. We also say that there is a functional dependency from X to Y, or that Y is functionally dependent on X

Key constraint is a special kind of functional dependency where all other l attributes are dependent or their values are uniquely defined by the key attribute . If they don’t depend on a single attribute and if nested dependencies exist then it will lead to redundant data. That will be resolved by decomposition.

• SSN →SSN, Name, Address

Schema *(R, F)* where

$R = \{SSN, Name, Address, Hobby\}$

$F = \{SSN \rightarrow Name, Address\}$

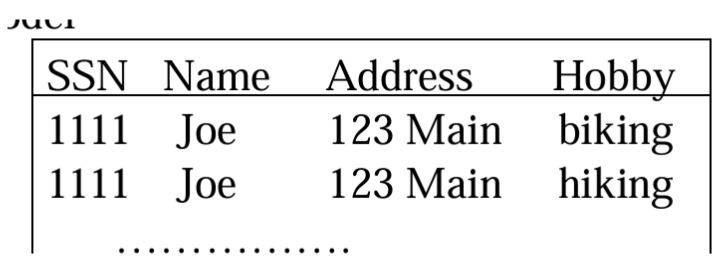
can be decomposed into

$R_1 = \{SSN, Name, Address\}$

$F_1 = \{SSN \rightarrow Name, Address\}$

In the above example hobby is not uniquely identified by ssn (it is multi valued) so a decomposition lead to more stable relation set.

5. Is decomposition a Solution to various kinds of anomalies that data might possess



SSN	Name	Address	Hobby
1111	Joe	123 Main	biking
1111	Joe	123 Main	hiking
....			

Redundancy leads to anomalies:–Update anomaly: A change in Address must be made in several places–Deletion anomaly: Suppose a person gives up all hobbies. Do we: 9 all hobbies. Do we: •Set Hobby attribute to null (no, since Hobby is part of key) •Delete the entire row (no, since we lose other information in the row) –Insertion anomaly: Hobby value must be supplied for any inserted row (since Hobby is part of key)

Solution: use two relations to store

Person information–Person1 (SSN, Name, Address)–

Hobbies (SSN, Hobby)

• The decomposition is more general: people • The decomposition is more general: people with hobbies can now be described •No update anomalies:–Name and address stored once–A hobby can be separately supplied or deleted

Thus decomposition could be one strong solution to anamolies

6. Define Select and Project Operations in Relational Algebra using Set Notation

Select operator (σ) selects tuples that satisfy a given predicate.

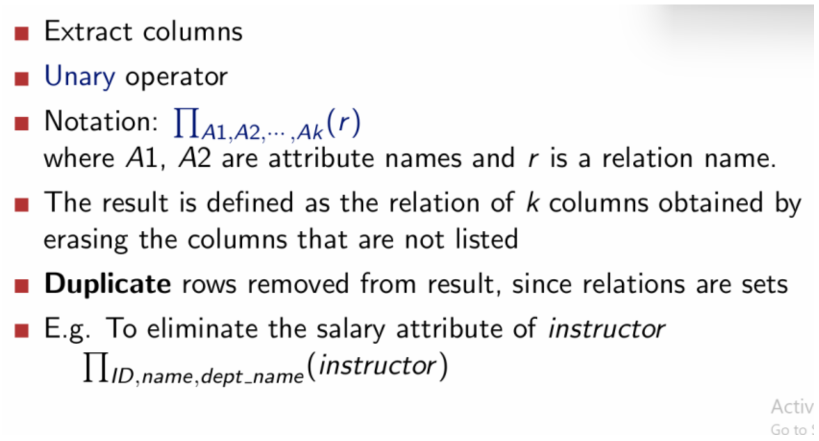
Notation: $\sigma_p(r)$ p is called the selection predicate

Definition : $\sigma_p(r) = \{ t \mid t € r \text{ and } p(t) \}$

P is a formula in propositional calculus consisting of terms connected by : Λ (and), v (or), ┐ (not)

$$\sigma_{dept\_name = \text{“}Physics\text{''} \wedge salary > 90000}(instructor)$$

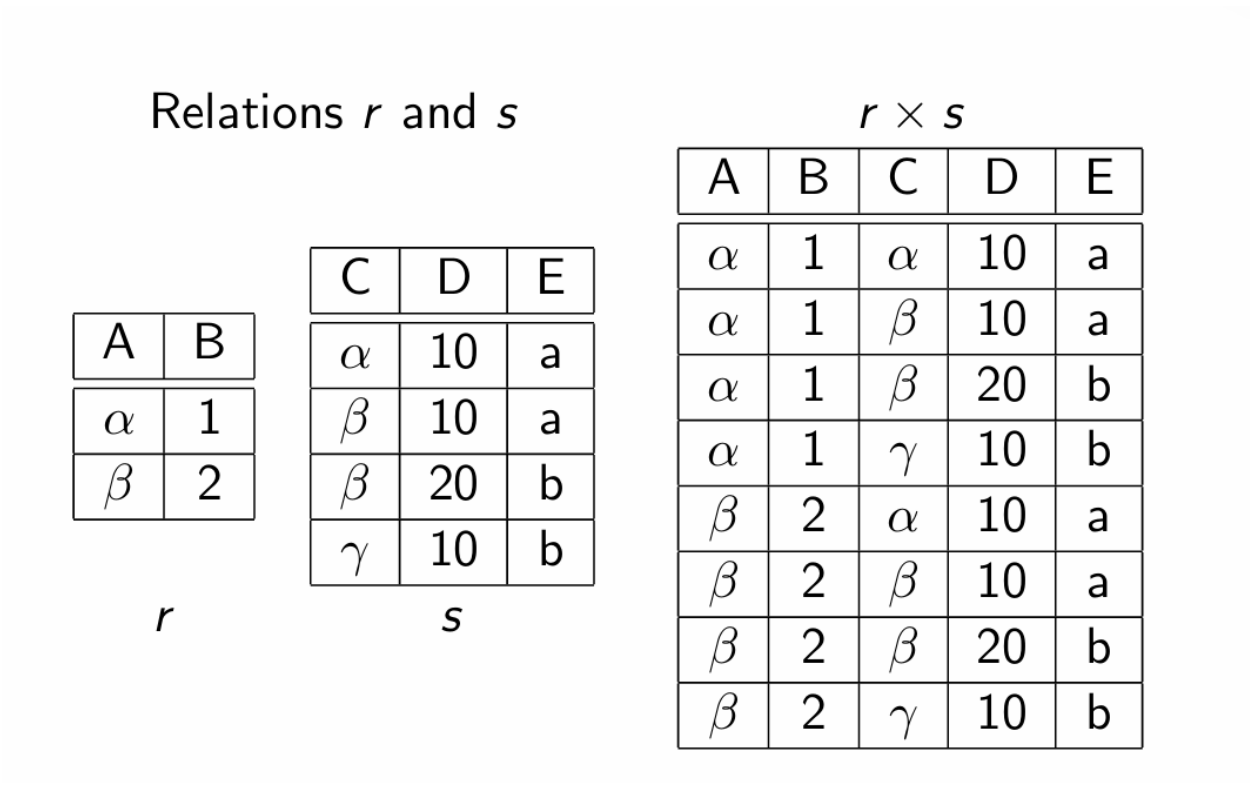
Project operator (∏) is used to



- Extract columns
- Unary operator
- Notation: $\prod_{A1,A2,\cdots,Ak}(r)$
  where $A1$, $A2$ are attribute names and r is a relation name.
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- **Duplicate** rows removed from result, since relations are sets
- E.g. To eliminate the salary attribute of *instructor*
  $\prod_{ID,name,dept\_name}(instructor)$

Activ
Go to

7. What is the difference between join and cartesian product in relational Algebra



Relations r and s

A	B
α	1
β	2

r

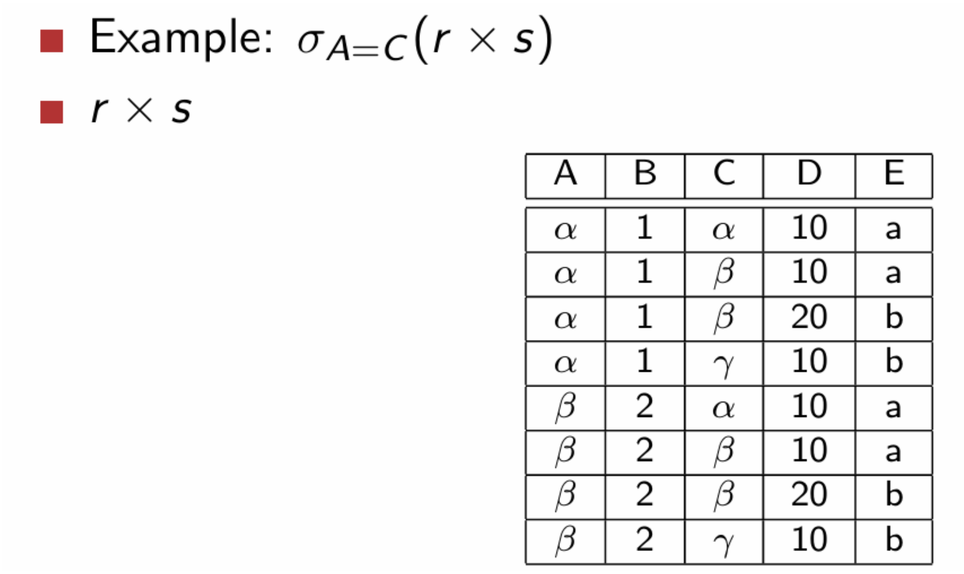
C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

$r \times s$

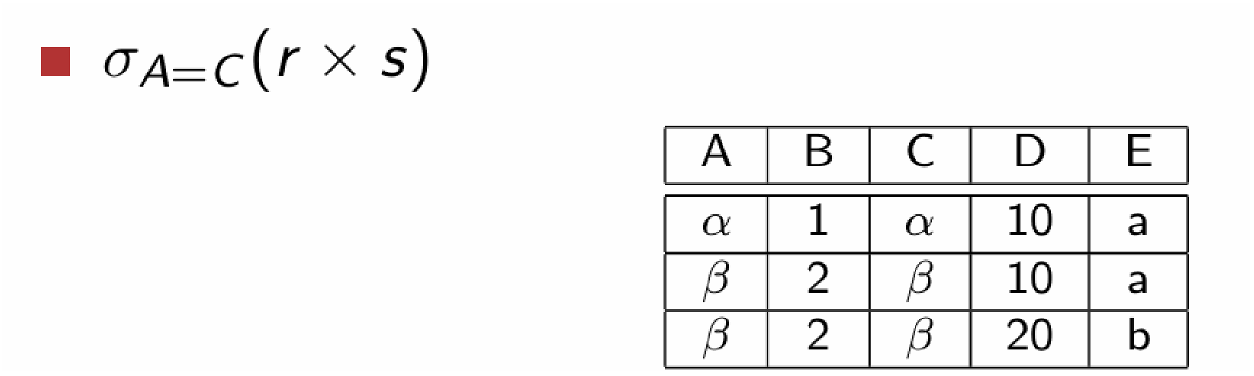
A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Can build expressions using multiple operations



- Example: $\sigma_{A=C}(r \times s)$
- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b



- $\sigma_{A=C}(r \times s)$

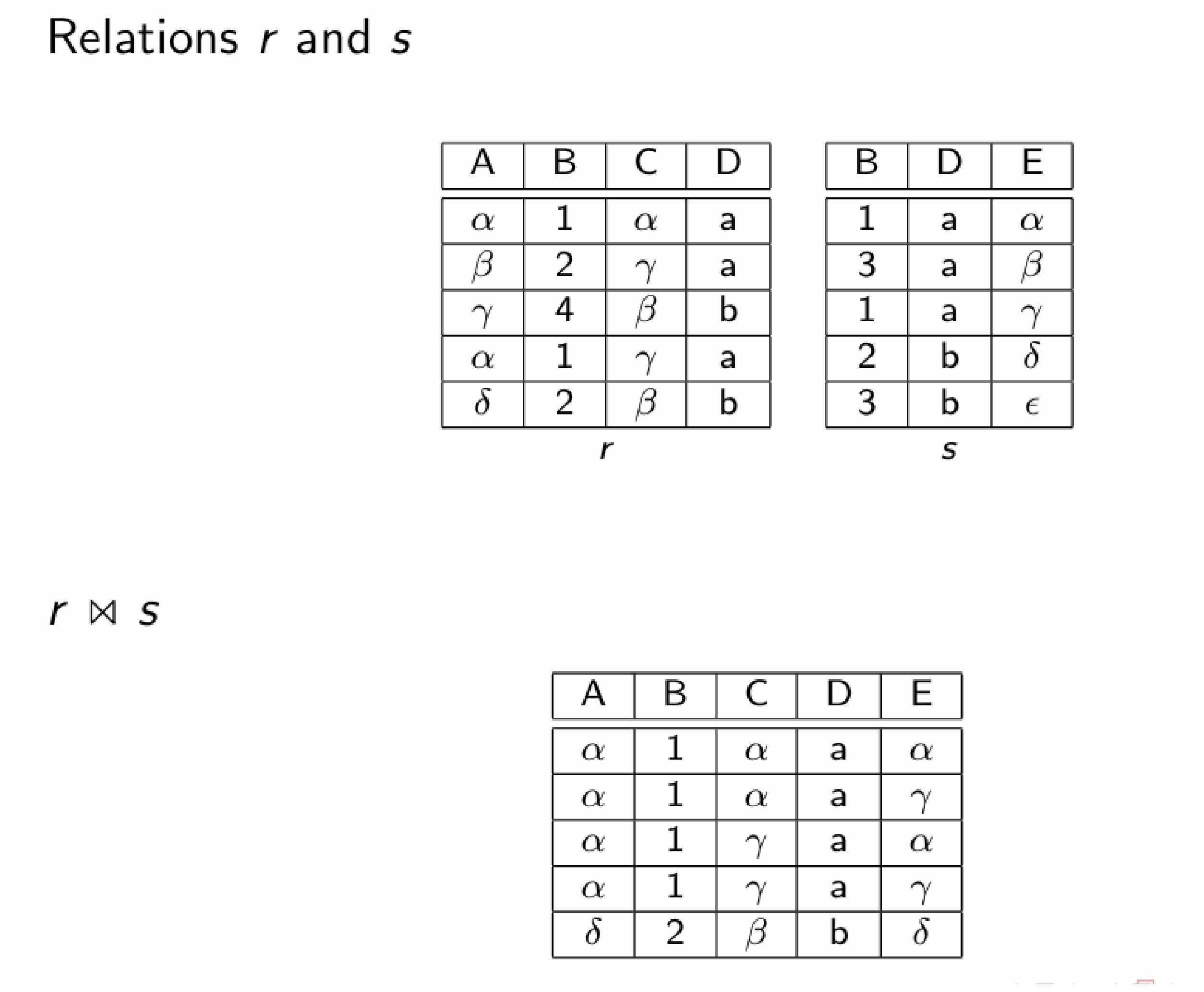
A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Natural-Join Operation

- Let r and s be relations on schemas R and S respectively. Then, $r \bowtie s$ is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
    - t has the same value as t_r on r
    - t has the same value as t_s on s
- Example:

  R = (A, B, C, D)
  S = (E, B, D)
  - Result schema = (A, B, C, D, E)
  - $r \bowtie s$ is defined as:

    $\prod_{r.A, r.B, r.C, r.D, s.E}(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s))$



Relations r and s

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

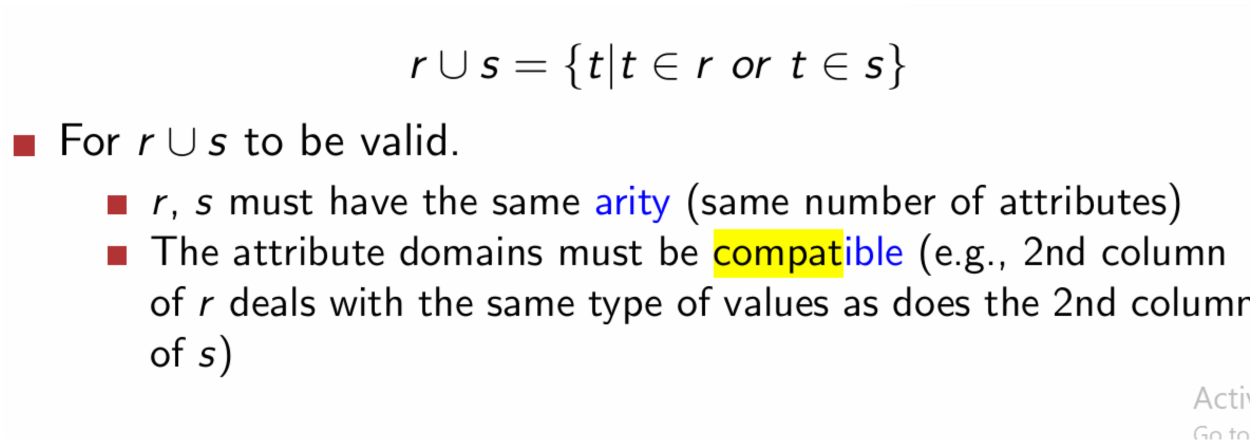
B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

s

$r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

8. What is union compatibility? Why do the UNION, INTERSECTION, and DIFFERENCE operations require that the relations on which they are applied be union compatible



$$r \cup s = \{t | t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.
  - r, s must have the same arity (same number of attributes)
  - The attribute domains must be compatible (e.g., 2nd column of r deals with the same type of values as does the 2nd columr of s)

Acti
Go to

This is called union compatibility

(5M)

1. Design an EER schema for a database application that you are interested in. Specify all constraints that should hold on the database. Make sure that the schema has at least five entity types, four relationship types, a weak entity type, a superclass/subclass relationship, a category

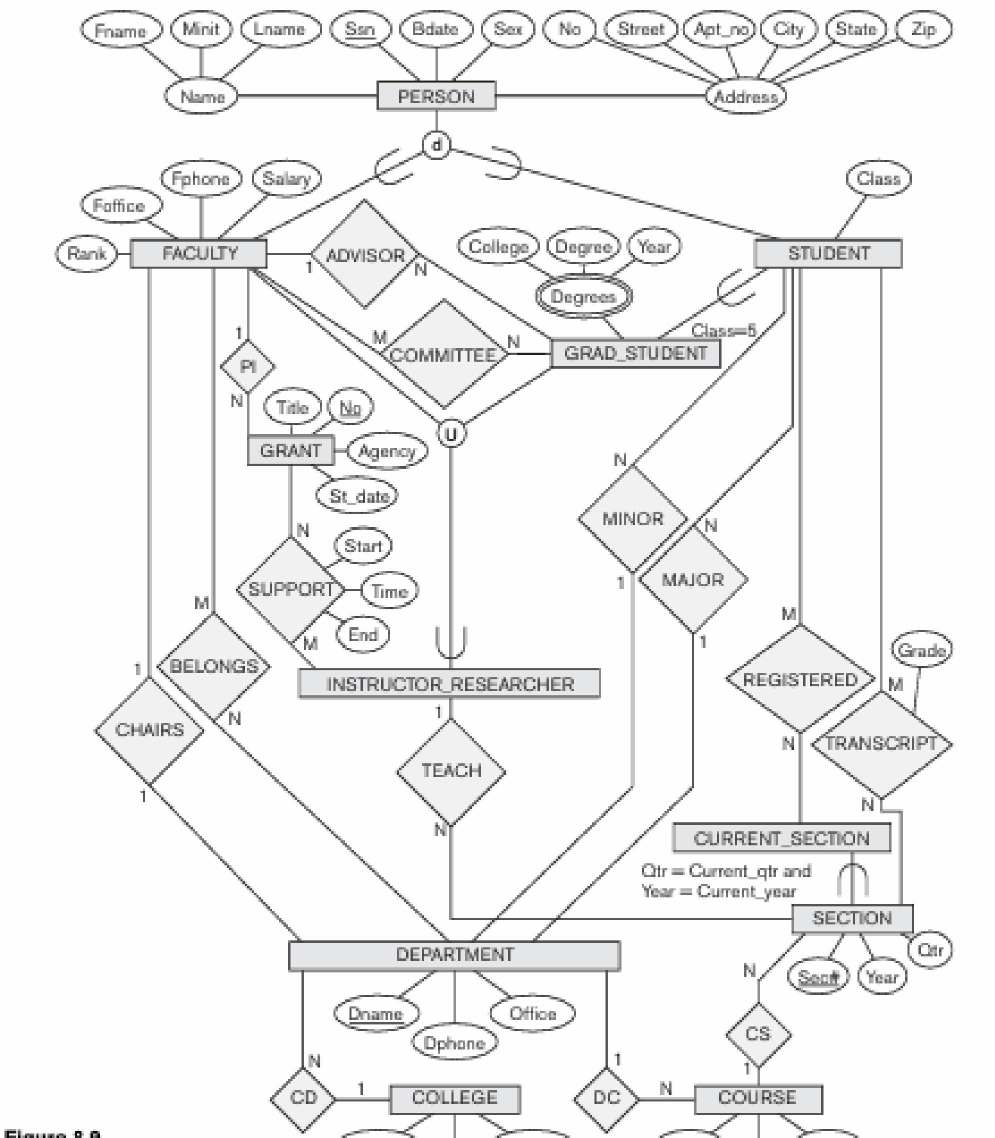


Figure 8.9
An EER conceptual schema
for a UNIVERSITY database.

2. Draw an EER diagram in   for a car dealer. Map the EER schema into a set of relations. For the VEHICLE to CAR/TRUCK/SUV generalization, consider all the three options i.e. disjoint, union and overlapping types describe the corresponding

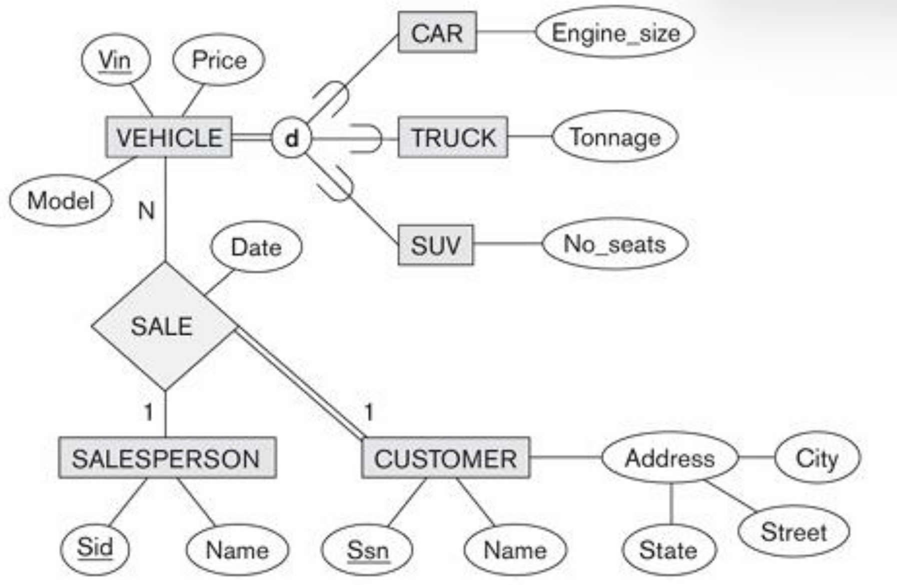
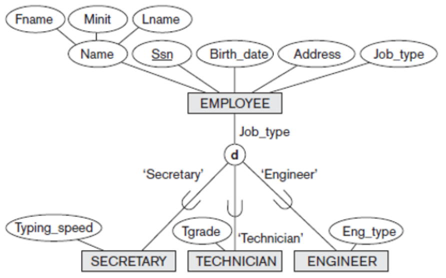
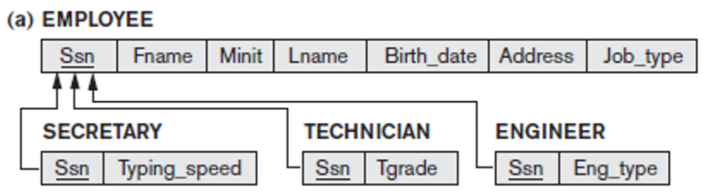


Figure 9.9
EER diagram for a car dealer

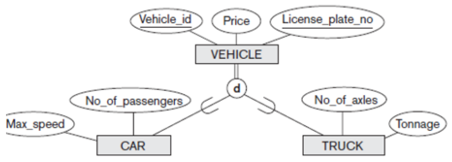
EER Mapping Rules

- Step 8: Options for Mapping Specialization or Generalization (see pages 294-295)
  - **Option 8A: Multiple relations—superclass and subclasses**
    - For any specialization (total or partial, disjoint or overlapping)





- Step 8: Options for Mapping Specialization or Generalization (see pages 294-295)
  - **Option 8B: Multiple relations—subclass relations only**
    - Subclasses are total
    - Specialization has disjointedness constraint



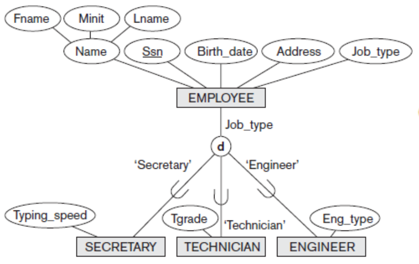
(b) CAR

Vehicle_id	License_plate_no	Price	Max_speed	No_of_passengers

TRUCK

Vehicle_id	License_plate_no	Price	No_of_axles	Tonnage

  - **Option 8C: Single relation with one type attribute**
    - Type or discriminating attribute indicates subclass of tuple
    - Subclasses are disjoint
      - Potential for generating many NULL values if many specific attributes exist in the subclasses

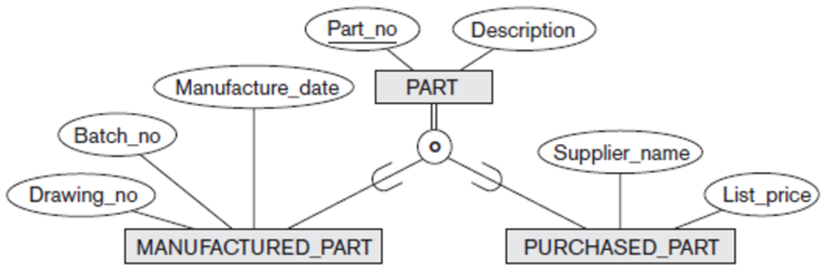


(c) EMPLOYEE

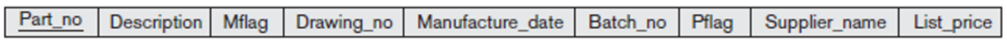
Ssn	Fname	Minit	Lname	Birth_date	Address	Job_type	Typing_speed	Tgrade	Eng_type

– Option 8D: Single relation with multiple type attributes

- Subclasses are overlapping
- Will also work for a disjoint specialization



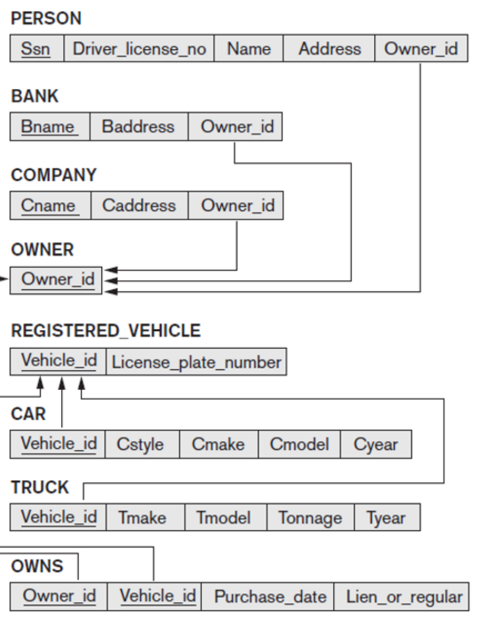
(d) PART



<u>Part_no</u>	Description	Mflag	Drawing_no	Manufacture_date	Batch_no	Pflag	Supplier_name	List_price

- Step 9: Mapping of Union Types (Categories)
  - Defining superclasses have different keys
  - Specify a new key attribute
    - **Surrogate key**

Figure 9.7
Mapping the EER categories (union types) in Figure 8.8 to relations.



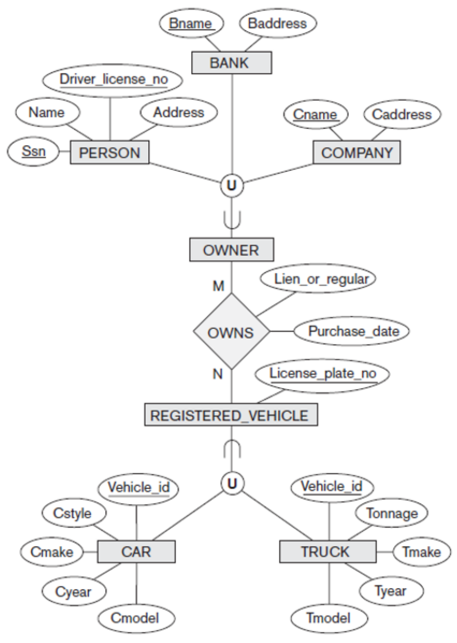


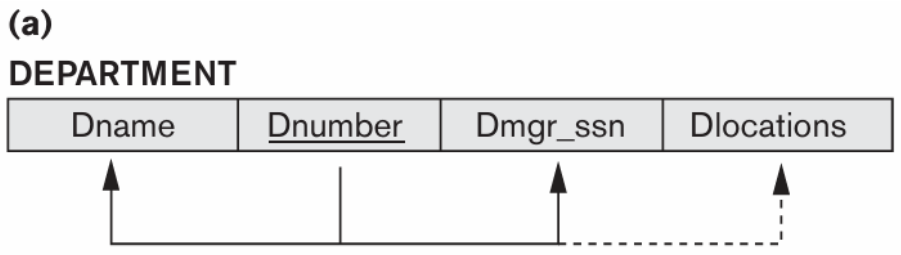
Figure 8.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

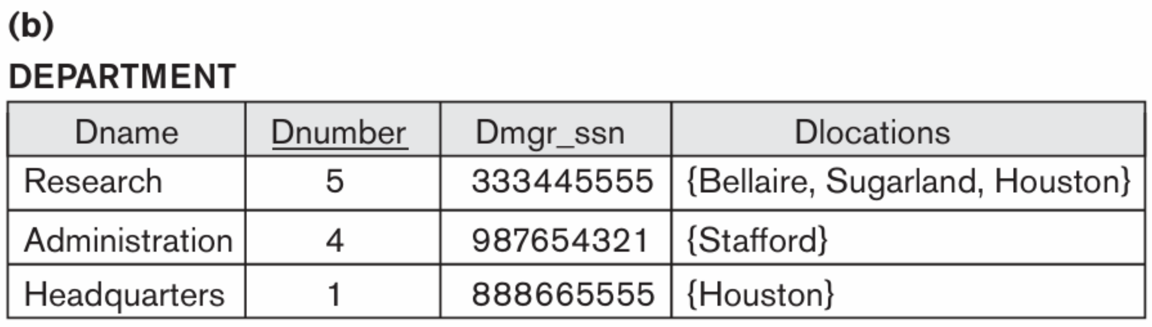
3. Define 1, 2nd Normal Forms Explain with examples

The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized.

Consider the DEPARTMENT relation schema shown in Figure 15.1, whose primary key is Dnumber, and suppose that we extend it by including the Dlocations attribute as shown in Figure 15.9(a).

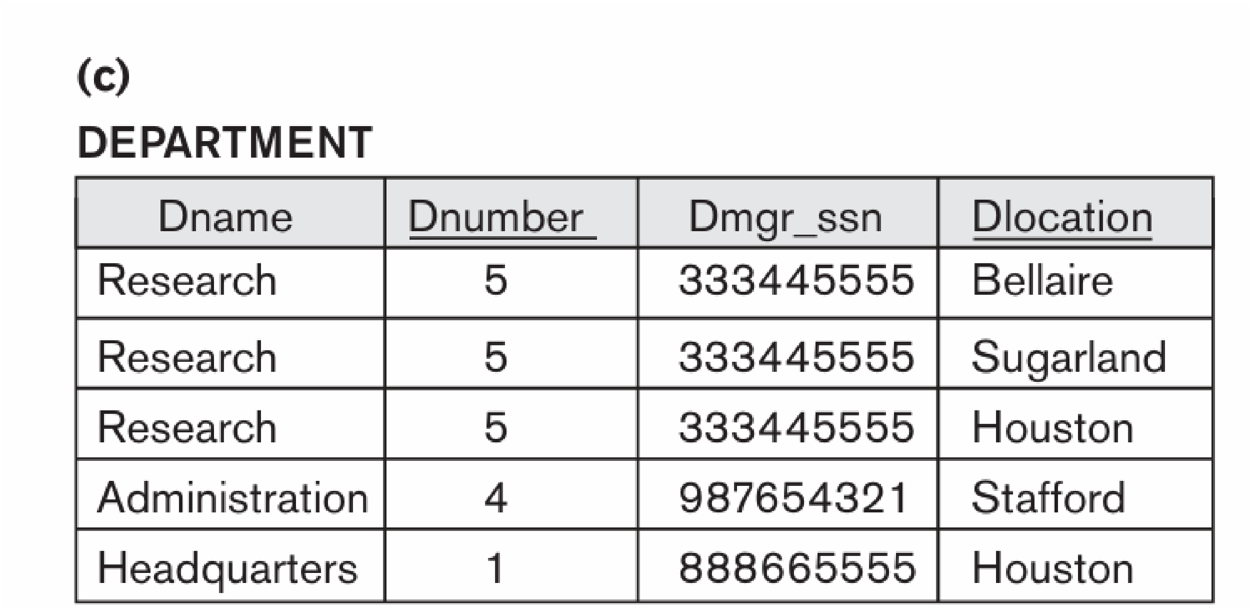
The domain of Dlocations contains atomic values, but some tuples can have a set of these values. In this case, Dlocations is not functionally dependent on the primary key Dnumber




(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}



(c)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

(a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy

The domain of Dlocations contains sets of values and hence is nonatomic. In this case, Dnumber → Dlocations because each set is considered a single mem ber of the attribute domain. The solution to obtain 1st NF is to decompose into 2 different relations. i.e. Remove the attribute Dlocations that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of DEPARTMENT

Thus

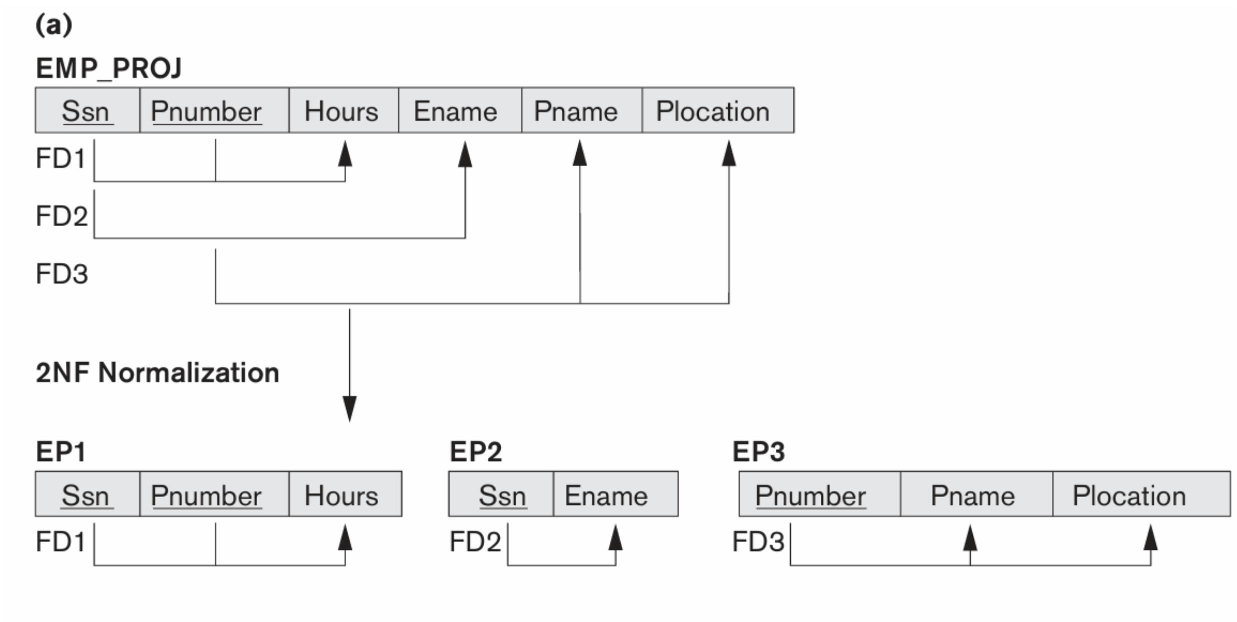
!st NF states that the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple. It states that the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple. The only attribute values permitted by 1NF are single atomic (or indivisible) values.

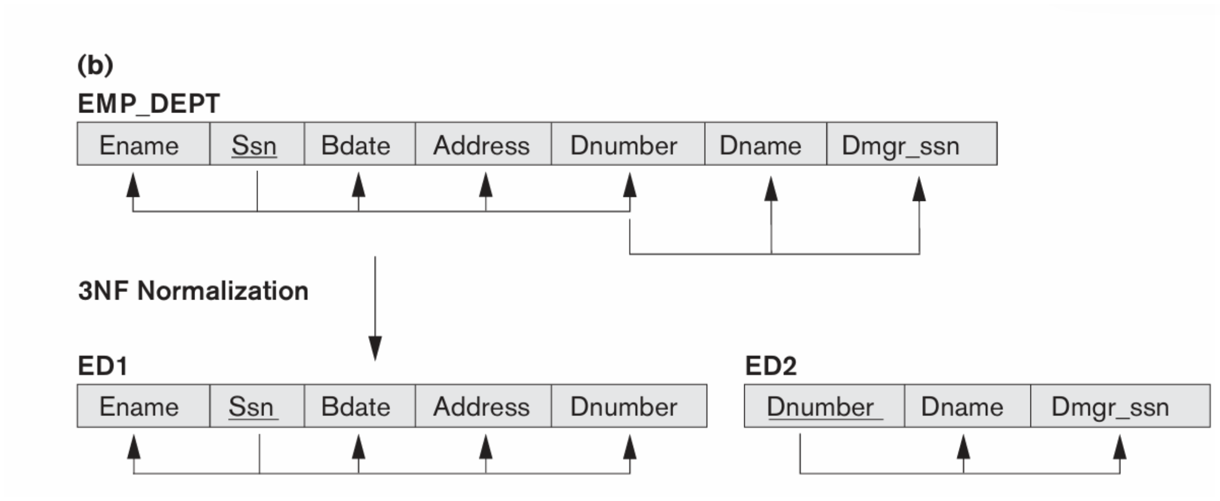
2nd NF

Second normal form (2NF) is based on the concept of full functional dependency. A functional dependency X → Y is a full functional dependency if removal of any attribute A from X means that the dependency does not hold any more;

A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

If a relation schema is not in 2NF, it can be second normalized or 2NF normalized into a number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent. Therefore, the functional dependencies FD1, FD2, and FD3 in Figure 15.3(b) lead to the decomposition of EMP_PROJ into the three relation schemas EP1, EP2, and EP3





a relation schema R is in 3NFif it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key. i.e.

The dependency Ssn → Dmgr_ssn is transitive through Dnumber in EMP_DEPT in Figure 15.3(a), because both the dependencies Ssn → Dnumber and Dnumber → Dmgr_ssn hold and Dnumber is nei ther a key itself nor a subset of the key of EMP_DEPT. Intuitively, we can see that the dependency of Dmgr_ssn on Dnumber is undesirable in EMP_DEPT since Dnumber is not a key of EMP_DEPT

4. **What is Lossy and Lossless decomposition Give Examples**

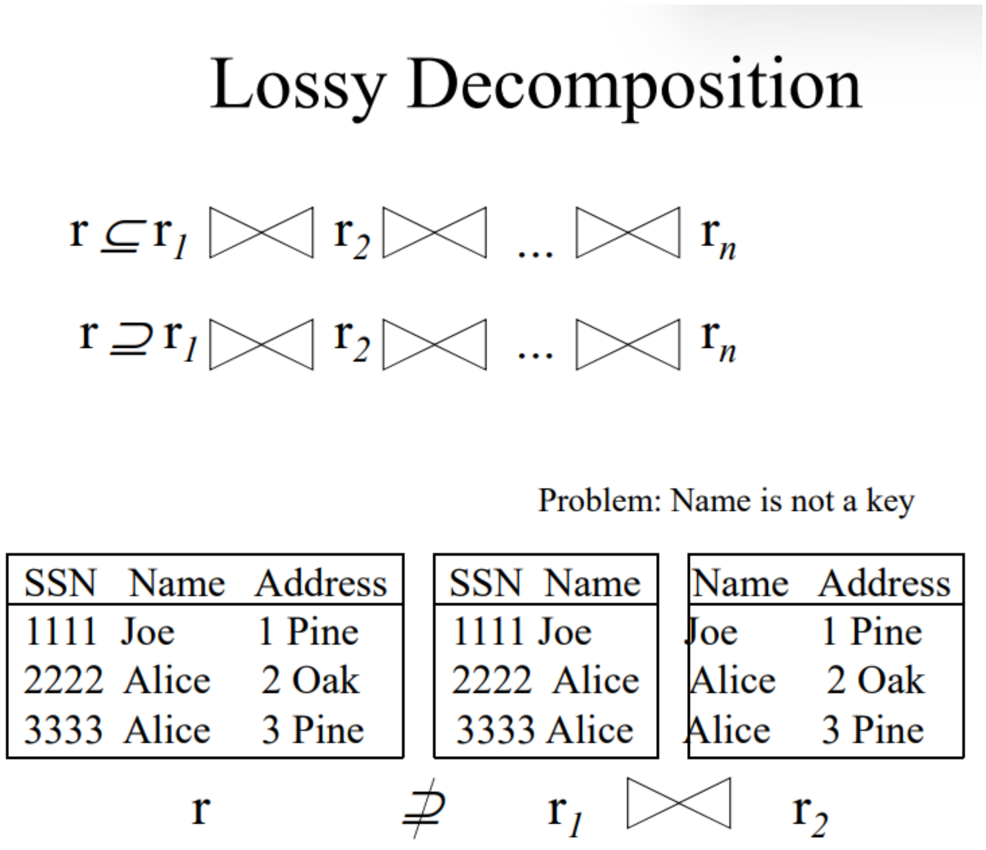
Definition. Formally, a decomposition $D = \{R_1, R_2, ..., R_m\}$ of R has the **lossless (nonadditive) join property** with respect to the set of dependencies F on R if, for *every* relation state r of R that satisfies F, the following holds, where $*$ is the NATURAL JOIN of all the relations in D: $*(\pi_{R_1}(r), ..., \pi_{R_m}(r)) = r$.

lossy design to refer to a design that represents a loss of information
Schema (R, F)where R = {SSN, Name, Address, Hobby} F = {SSN→Name, Address} can be decomposed into into R1= {SSN, Name, Address} F1= {SSN →Name, Address} and R2= {SSN, Hobby}
Schema (R, F) where R = {SSN, Name, Address, Hobby} F = {SSN→ Name, Address} can be decomposed into 38 R1 = {SSN, Name, Address} F1 = {SSN → Name, Address} and R2 = {SSN, Hobby}

A decomposition should not lose any information
• A schema decomposition is lossless if every valid instance, r, of the schema can be reconstructed from its components:
Goal of decomposition is to eliminate redundancy by decomposing a relation into several relations in a higher normal form

Decomposition should be lossless: it must be possible to reconstruct the original relation from the resulting relations

$$r = r_1 \bowtie r_2$$



Lossy Decomposition

$$r \subseteq r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

$$r \supseteq r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

Problem: Name is not a key

SSN	Name	Address
1111	Joe	1 Pine
2222	Alice	2 Oak
3333	Alice	3 Pine

SSN	Name
1111	Joe
2222	Alice
3333	Alice

Name	Address
Joe	1 Pine
Alice	2 Oak
Alice	3 Pine

$$r \not\supseteq r_1 \bowtie r_2$$

Loss lessness test Schema (R, F) where R = {SSN, Name, Address, Hobby} F = {SSN → Name, Address} can be decomposed into R1 = {SSN, Name, Address} 42 F1 = {SSN → Name, Address} and R2 = {SSN, Hobby} F2 = { } Since R1 ∩ R2 = SSN and SSN → R1 the decomposition is lossless

5. Given
instructor (ID, name, dept name, salary, DOJ)
Write the following statements using Relational Algebra Operators

i. Select all instructors from instructor relation who work in physics dept and whose date of joining is earlier than 2025 and whose current salary is > 90000

$$\sigma_{dept\_name = \text{“Physics”} \wedge salary > 90000}(instructor)$$

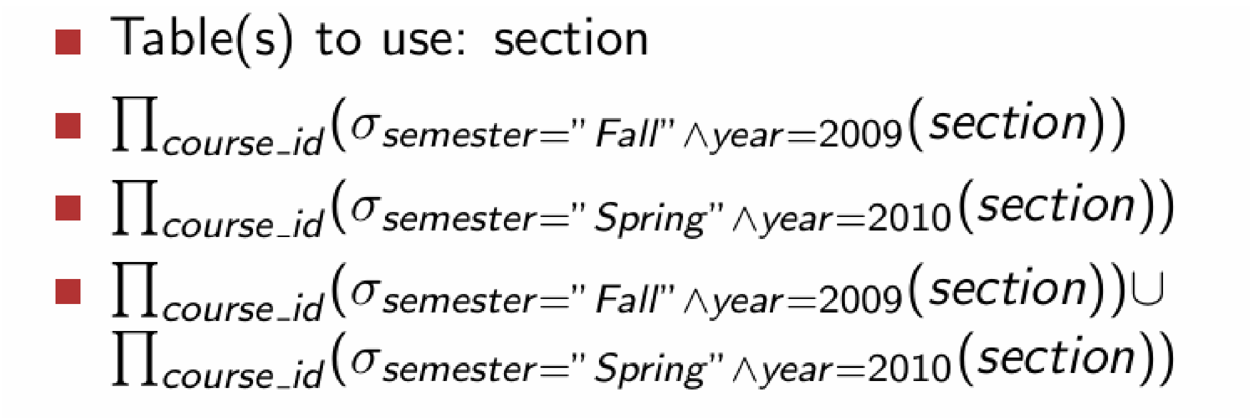
Λ DOJ < “01-01-2025”

ii. List down the ID, Names of All Instructors who work in CS Dept

$\Pi_{ID,name,dept\_name}$ (б(Dept_ID=”CS”) (Instructor))

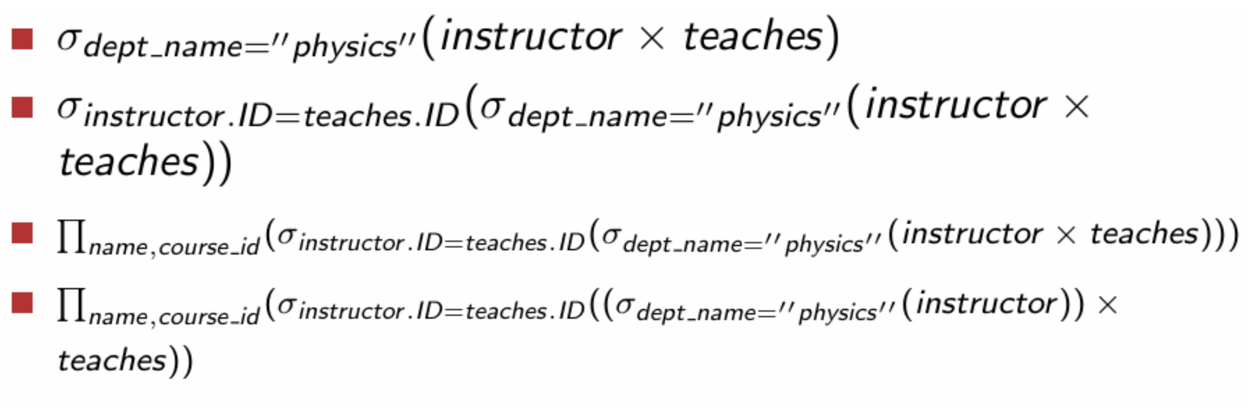
6. course (course id, title, dept name, credits)
section (course id, sec id, semester, year, building, room number, time slot id)
Based on the above schema translate the following statements into RA statement

i. Find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both



- Table(s) to use: section
- $\Pi_{course\_id}(\sigma_{semester=''Fall'' \wedge year=2009}(section))$
- $\Pi_{course\_id}(\sigma_{semester=''Spring'' \wedge year=2010}(section))$
- $\Pi_{course\_id}(\sigma_{semester=''Fall'' \wedge year=2009}(section)) \cup \Pi_{course\_id}(\sigma_{semester=''Spring'' \wedge year=2010}(section))$

ii. Find the names of all instructors in the Physics department together with the course id of all courses they taught. instructor (ID, name, dept name, salary) teaches (ID, course id, sec id, semester



- $\sigma_{dept\_name=''physics''}(instructor \times teaches)$
- $\sigma_{instructor.ID=teaches.ID}(\sigma_{dept\_name=''physics''}(instructor \times teaches))$
- $\prod_{name,course\_id}(\sigma_{instructor.ID=teaches.ID}(\sigma_{dept\_name=''physics''}(instructor \times teaches)))$
- $\prod_{name,course\_id}(\sigma_{instructor.ID=teaches.ID}((\sigma_{dept\_name=''physics''}(instructor)) \times teaches))$

7. Consider the two tables *T*1 and *T*2 shown in Figure 6.15. Show the results the following operations:

a. $T1 \bowtie_{T1.P = T2.A} T2$

b. $T1 \bowtie_{T1.Q = T2.B} T2$

c. $T1 \bowtie_{T1.P = T2.A} T2$

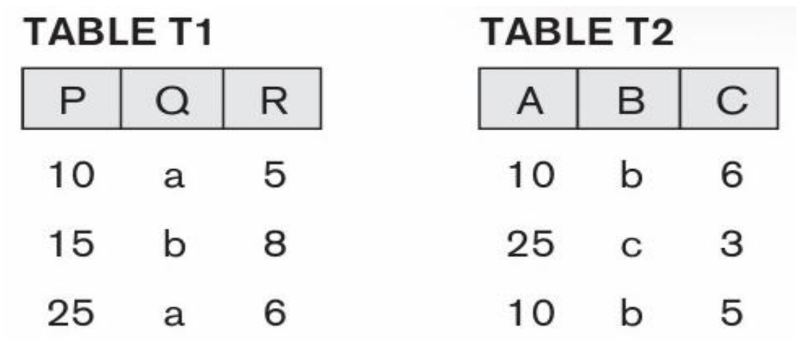


TABLE T1

P	Q	R
10	a	5
15	b	8
25	a	6

TABLE T2

A	B	C
10	b	6
25	c	3
10	b	5

i.

P	Q	R	B	C
10	a	5	b	6
10	a	5	b	5
25	a	6	c	3

ii.

P	Q	R	A	C
15	b	8	10	6
15	b	8	10	5

iii.

(T1 join on T1.r = T2.c   T2)

P	Q	R	A	B
10	a	5	10	b
25	a	6	10	b

8. instructor (ID, name, dept name, salary, DOJ), teaches (ID, course id, sec id, semester, year), course (course id, title, dept name, credits)
For the schema and relations given above write down following SQL statements
   i. Find the total number of instructors who teach a course in the Spring 2010 semester

(*Since Instructor ID, Course ID, Sem and year all appear in teaches relation and we only need to count the instructors and their names are not needed so we only need to use teaches relation no join is required*.)

select count (distinct ID) from teaches where semester = Spring and year = 2010;

(*if you don't use distinct it will count duplicates each instructor teaches multiple courses*)

ii. Write an SQL Statement to Find the names and average salaries of all departments whose average salary is greater than 42000 for the above instructor relation. Explain it with an example

select dept name, avg (salary) from instructor group by dept name;