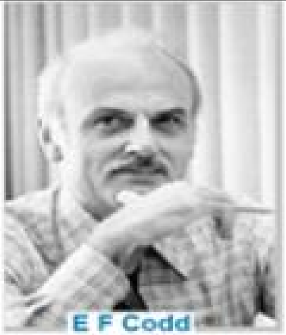
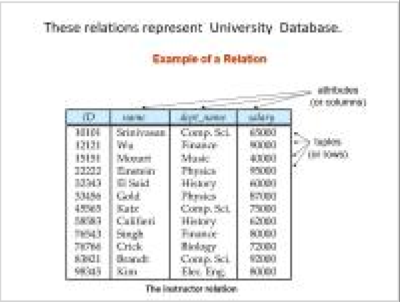
DATABASE MANAGEMENT SYSTEM – MCA
I Semester
AUG 2022

Relational Model

Introduction To Relational Model



- A data model is a collection of conceptual tools.
- **Relational Database Model** is the most common model in industry today.
- A relational database is based on the relational model developed by Edgar. F. Codd (12 rules).
- The relational model- collection of tables and the relationships among those data.
- A relational database consists of a collection of **tables,** each table is assigned with unique name**.**
- Correspondence between the concept of **table** and the mathematical concept of **relation**


These relations represent University Database.

Example of a Relation

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

The instructor relation

Properties of a relation

- Each relation contains only one record type.
- Each relation has a fixed number of columns that are explicitly named. Each attribute name within a relation is unique.
- No two rows(tuples) in a relation are the same.
- Each item or element in the relation is atomic.
- Rows have no ordering associated with them.
- Columns have no ordering associated with them.

Relational Terminology

Terms	Definition
Relation	Set of rows(tuples), each row therefore has the same columns(attributes).
Tuple	It is a row in the relation.
Attribute	It is a column in the relation.
Degree of a relation	Number of columns in the relation
Cardinality of a relation	Number of rows in the relation
N-ary relation	Relation with degree N.
Domain	Set of allowed values for each attribute.

These relations represent University Database.

Example of a Relation

attributes (or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples (or rows)

The instructor relation

Example of a Relation

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

The course relation.

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

The *prereq* relation

Some Terms

Relation Instance: A specific instance of a relation, i.e. set of rows in a relation at an instance.

In general, a relation schema consists of a list of Attributes and their corresponding domains.

Database schema, which is the logical design of the database.

Database Instance, which is a snapshot of the data in the Database at a given instant in time.

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute.
  - A valid range value for a Marks attribute may be 0-100
  - Marks Domain is {0,1,2,..50,51,..100}
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain.
- The **null** value **causes** complications in the definition of many operations.

Relation Schema and Instance

- *If* $A_1, A_2, \ldots, A_n$ are **attributes**, *then*
- $R = (A_1, A_2, \ldots, A_n)$ is a **relation schema**

  Example: *instructor* = (*ID, name, dept_name, salary*)

  Let $D_1, D_2, \ldots. D_n$ *be the Domains of* $A_1, A_2, \ldots, A_n$ *respectively.*

- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is a **subset of** $D_1 \times D_2 \times \ldots \times D_n$ ***(cartesian product)***

  Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

Relation :

- A row in a table represents a relationship among a set of values.
- A table is a collection of such relationships, there is a close correspondence between the concept of table and the mathematical concept of relation.

Ex:

Cours_IDs set A={ BIO_301,BIO_399,CS_190,...} (set of all valid Course_ID)

Prereq_IDs set B={ BIO_101,CS_101,..} (set of all valid Prrereq_ID)

A x B ={ (BIO_301, BIO_101),(BIO_301,CS_101),..
(BIO_399,BIO_101), (BIO_399,CS_101),..
(CS_190, BIO_101),(CS_190,CS_101),..
}

A set (table in previous slide) *Prereq* is a subset of A x B

***Prereq* ={(BIO_301, BIO_101), (BIO_399,BIO_101), (CS_190, CS_101),...} is a Relation.**

Compare it with ***Prereq*** relation

A x B ={ (BIO_301, BIO_101),(BIO_301,CS_101),..
…,(BIO_399,BIO_101), (BIO_399,CS_101),..
….,(CS_190, BIO_101),(CS_190,CS_101),..
}

A x B gives all possible combinations of domain values, which involve real world facts- such as (BIO_301, BIO_101) i.e. for Course BIO-301, BIO-101 is Prerequisite course. Also A x B is having information such as (BIO_301,CS_101) which is not a real world fact.

A relation such as **Prereq** is the **subset of A x B** which represents real world fact.

Prereq =**{(BIO_301, BIO_101), (BIO_399,BIO_101), (CS_190, CS_101),…}**

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

Keys

- A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.
- Let **K** $\subseteq$ **R**, **K** is a **superkey** of ***R*** if values for **K** are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID , name} are both super keys of *instructor.*
  - ***Instructor(ID, Name, Dept_Name, Salary)***
  - *i.e* ***R={ID, Name, Dept_Name, Salary}*** ***& assume K= {ID, name}***
- A superkey may contain **extraneous attributes**.
  - *Example: In {ID, Name} ,* ***name*** *is a* ***extraneous attribute****, which not really required to identify a row uniquely, in other words, only ID is enough to identify rows uniquely.*

Keys

- **Minimal super key** is called candidate key.
- Super key **K** is a **candidate key** if **K** is minimal.
- Minimal Super key means-Minimum number of attribute of K required to identify every row uniquely.

  Example: K= **{***ID, Name***}** is a Super key, but ***Name*** attribute is not necessary to identify each row uniquely. **Name** is **extraneous**

  Hence ID is minimum required attribute to identify every row uniquely

Therefore ID is candidate key for *Instructor*

- **Primary key** is a term used by the database designer to denote a candidate key.

Keys…

- Answer the following by understanding the requirements given below.
- **CUSTOMER(Custid, Name,Mid_Name,LastName, City, phone, email)**

  ACCOUNT(AccNo, CustId, Intr_CustId, AccType, Branch)
- Is **(Phone, Email)** is a Super Key for CUSTOMER? If yes, is it a minimal Super key ?

In Bank every customer will have Unique **CustomerID**. **Intr_CustId** is the Customer Id of customer who is introducing a new customer to the Bank.

A customer can have multiple accounts such as SB, Current, Loan etc. Every **Accno** is unique. **Name , Mid_name and Last_Name** information about a customer must be distinguishable from other customers. **Phone** - phone number of the customer. **Email-** Email Id of the customer. Every Customer has a unique phone number and email id.

Keys

- Is **(Phone, Email)** is a Super Key , if yes is it a minimal Super key ?

- **K= (Phone, Email), Super key –YES**
- **IS K minimal Super Key?**
  - **K - Phone ={Email} Email alone can be used to identify every tuple uniquely, hence K is not minimal.**
  - **Email is minimal Super key & hence it is a Candidate key.**
  - Another possibility is
  - **K - Email ={ Phone}**
  - **Phone is minimal Super key & hence it is a Candidate key.**
  - **In this case Phone, Email & CustId , (Name, Mid_Name, Last_Name) is also Candidate Key.**

- Is **(AccNo, CustId)** a Super Key in ACCOUNT relation ? if yes, is it a minimal Super key ?

- IS (Custid, Name, Mid_Name, LastName) a Super Key in CUSTOMER Relation?
  - **I**f Yes, is it minimal Super key ?
- List Possible minimal Super keys (Candidate Keys) in

ACCOUNT(AccNo, CustId, Intr_CustId, AccType, Branch)

CUSTOMER(Custid, Name, Mid_Name, LastName, City, phone, email)

A Relation may have multiple minimal Super Keys (Candidate Key)

Keys

- A Relation may have **multiple minimal Super Keys** (Candidate Key)

- One of them may be considered as Primary Key
  - Ex: **Cust_Id** in Customer may be Primary Key
- Remaining all Candidate Keys are called as **Alternate Keys.**
- There can be only **ONE Primary Key** for a relation- but it may be Simple or Composite primary key.
- Ex: Stud(RegNo, Course_Id, Grade) -simple

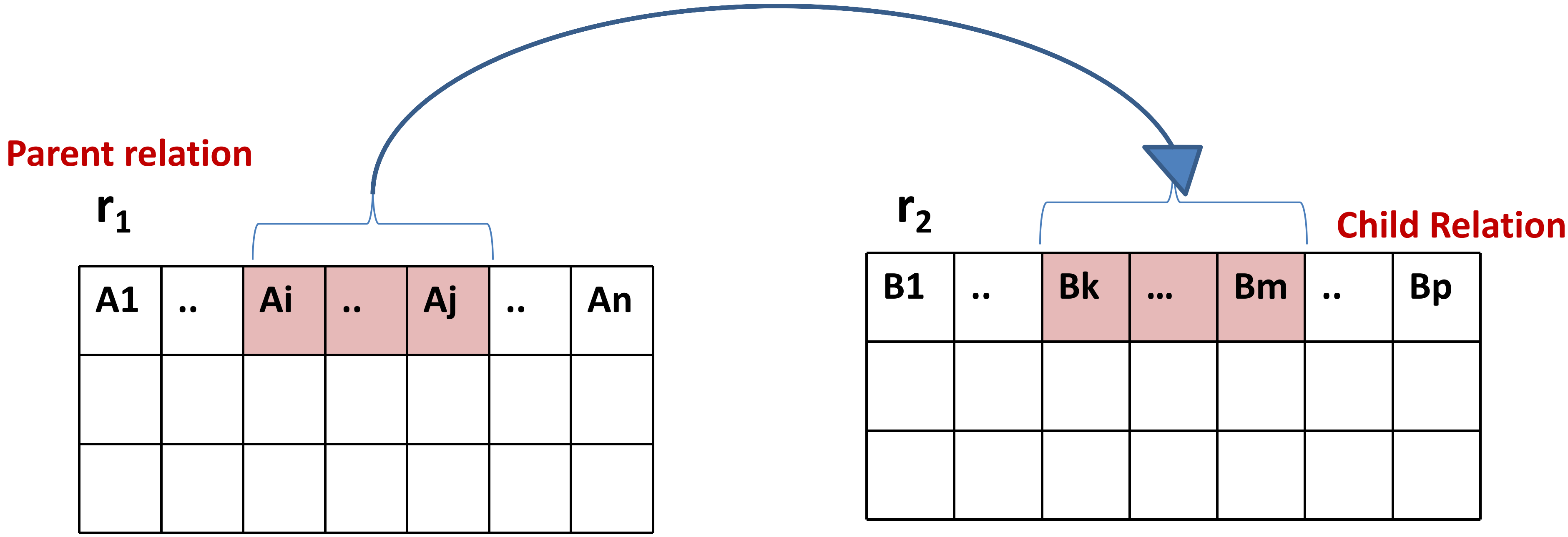
  Person(Name, Mname, Lname, Age) -composite

Foreign Keys-Referential Constraint

- Some attributes of a relation $\mathbf{r_2(B_1,B_2,..B_p)}$ shares domains and derives values from primary key attributes of another(or same also possible) relation $\mathbf{r_1(A_1,A_2,..A_n)}$.
- Such attributes of $\mathbf{r_2}$ is called a **foreign key** referencing $\mathbf{r_1}$.
- The relation $\mathbf{r_2}$ is called **referencing(Child) relation for** the foreign key dependency.
- The relation $\mathbf{r_1}$ is called **referenced(Parent) relation** for the foreign key.

Foreign key can also be – Simple or Composite

Foreign Keys-Referential Constraint



If **Bk,..Bm** attributes of **r2** derive values from primary key **Ai,..,Aj** of **r1** say, then

(Bk,…,Bm) forms **Foreign key** (child columns), **r2** (child table) is **referencing relation**.

(Ai,..,Aj) forms parent columns, **r1** (Parent table) is **referenced relation** for the foreign key.

(Bk,…,Bm) derives values from **(Ai,..,Aj).**

Existence of (Bk,…,Bm) values Depends on the existence of (Ai,..,Aj) values

Example: Foreign Keys-Referential Constraint

SID	Name	Age
S101	Ram	
S102	Akshay	
S103	Santosh	

Students (parent table, Students (SID) is parent column)

SID	CNo	Year	Grade
S101	C10	2012	
S101	C11	2013	
S103	C11	2013	
S103	C10	2012	
~~S120~~	~~C13~~	2012	

Enrollment (child)

CNo (Child Column) –Foreign Key referencing CID in Courses Table (parent column)

SID (Child Column) –Foreign Key referencing SID (Parent Column) of **Students Table**

CID	C_Name	Credits	Duration
C10	E.Maths	4	
C11	CSc	4	
C12	Electronics	4	

Courses (parent table, Courses(CID) is Parent column)

Properties:

A Foreign key can contain-

- Only values present in the corresponding Parent Column/s.
- **NULL** values (unless additional **NOT NULL** constraint imposed)

Example: Primary key and Foreign key relationship (**recursive**) in same table

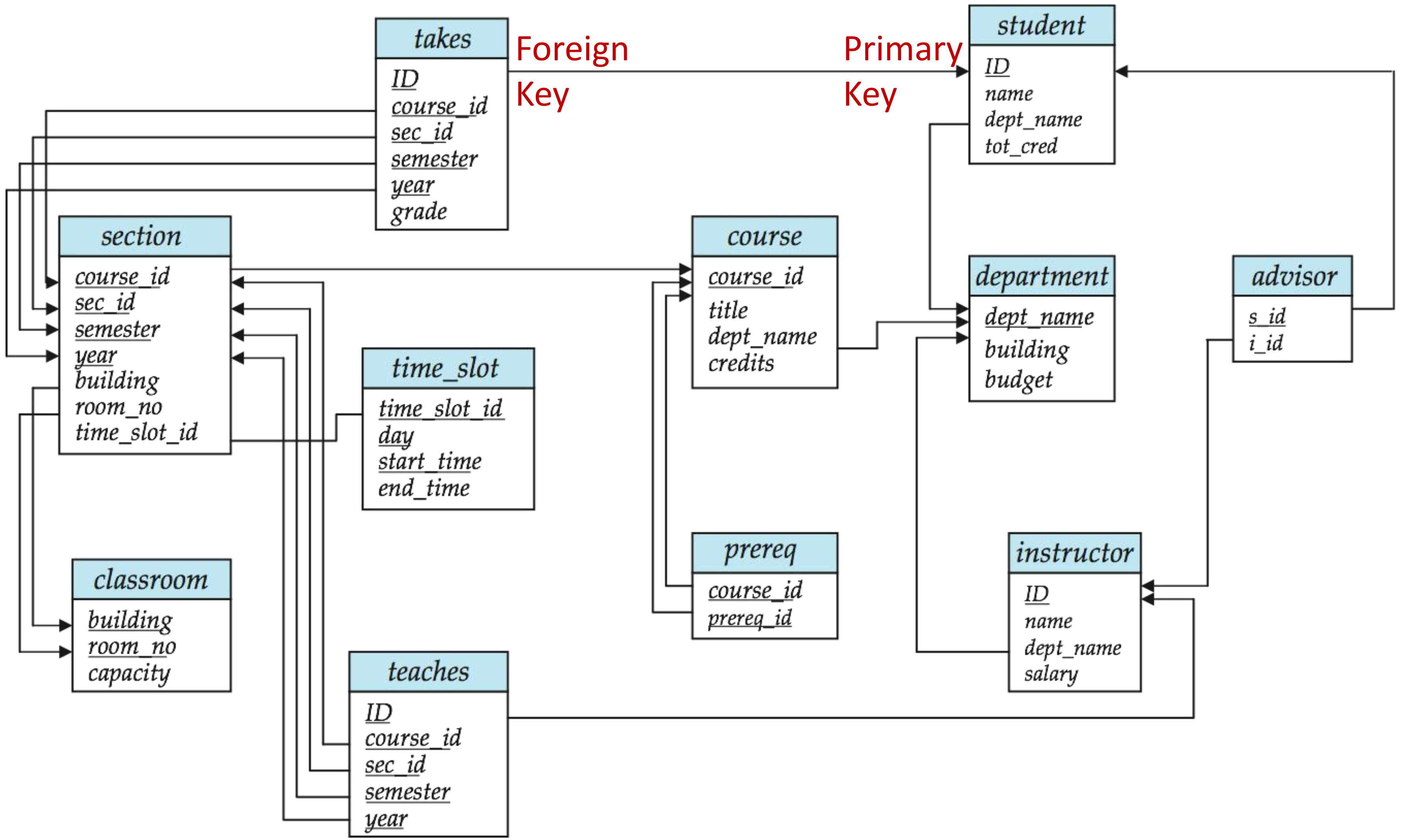
EMP table

EMPNO	ENAME	MGRNO
100		103
101		100
103		104
104		104
105		

MGRNO is the Employee number of Manger. Employee with EMpno 103 is the Manger for Employee with Empno 100. Therefore MGRNO is Foreign Key Referencing EMPNO

Insert / update / Delete should not violate primary key & foreign key relationship constraints

Schema Diagram for University Database



Visit: Later in ER model chapter

Relational Query Languages

- ***Query languages:*** Allow manipulation and retrieval of data from a database.

- Query Languages **!=** programming languages
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.

Formal Relational Query Languages

- Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:
  - *Relational Algebra*: More operational(procedural), very useful for representing execution plans.
  - *Relational Calculus*: Lets users to describe what they want, rather than how to compute it. (Non-operational, *declarative*.)

Relational Algebra

Query Language-

Procedural & Non-Procedural

There are a number of “pure” query languages:

The relational algebra is procedural,

The tuple relational calculus and domain relational calculus are nonprocedural.

The relational algebra consists of a **set of operations** that take one or two relations as input and produce a new relation as their result.

They **illustrate the fundamental techniques for extracting data** from the database.

SELECT (σ) : Selection of tuples(rows)

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

comparisons operators =, ≠, <, ≤, >

connectives *and* ($\wedge$), *or* ($\vee$), and n($\neg$).

- Select tuples with **A=B** and **D > 5**

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Quiz Q1:

$\sigma_{A<>B \text{ OR } D < 7}$ (r) has (1) 1 tuple (2) 2 tuples (3) 3 tuples (4) 4 tuples

Example: Selection of tuples(rows) (σ)

Instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Instructor

Result

ID	*name*	*dept_name*	*salary*
12121	Wu	Finance	90000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
83821	Brandt	Comp. Sci.	92000

Result of Instructors having salary more than $85000

$$\sigma_{Salary > 85000}(\text{Instructor})$$

PROJECT (π): Selection of Columns (Attributes)-

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- Select A and C attributes
- Projection
- $\pi_{A,\,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

removes duplicates

Quiz Q2:

The projection operation (1) removes duplicates (2) does not remove duplicates

Example: PROJECT

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Instructor

ID	*name*
22222	Einstein
12121	Wu
32343	El Said
45565	Katz
98345	Kim
76766	Crick
10101	Srinivasan
58583	Califieri
83821	Brandt
15151	Mozart
33456	Gold
76543	Singh

Result of Projection on ID and Name columns of Instructors relation.

$$\pi_{ID,\ Name}(Instructor)$$

Project Operation always- Discards duplicates, retains only one copy.

Joining two relations – Cartesian Product X..

- Relations *r, 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* x *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

Number of Tuples in r X s

r_n – number of tuples in **r**

s_n - number of tuples in **s**

r X s has $r_n * r_s$ **tuples**

Cartesian-product – naming issue.

- Relations *r, s*:

A	*B*
α	1
β	2

r

B	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

Note: Attribute **B** in r & s are from different domains but having same attribute name

- *r* x *s*:

A	*r.B*	*s.B*	*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

Renaming a Table..

- Allows us to refer to a relation, (say **E**) by more than one name.

$$\rho_x(E)$$

returns the expression ***E*** under the name ***X***

- Relations *r*

A	B
α	1
β	2

r

- $r \times \rho_s(r)$

r.A	r.B	s.A	s.B
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Renaming a Table.

- **A second form of the rename operation-**
- $\rho_{X\ (A1, A2, \ldots, An)}\ (E)$
- *Not only **E** is remaned as **X** , **attributes in X are also renamed** to $A_1, A_2, ..A_n$ respectivelly.*
- Relations *r*

A	B
α	1
β	2

r

- *r* **x** $\rho_{s(X,Y)}$ (r)

$r.A$	$r.B$	$s.X$	$s.Y$
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Union, Intersection & Set Difference

- Relations *r, s:*

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

Two conditions
*r and s must be of the **same arity***

*I^{th} **attribute** in r and s must be
from **same domain***

- Union:
r ∪ s

A	B
α	1
α	2
β	1
β	3

- Intersection
r ∩ s:

A	B
α	2

- Set Difference
r - s:

A	B
α	1
β	1

Is this true ?

$r \cap s = r - (r - s)$

tuples which are present only in r (not present in s)

Natural Join – Joining two relations

- Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, the "natural join" of relations *R* and *S* is a relation on schema ***R* ∪ *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***

t_r

A	B	C	D
α	1	α	a
β	2	γ	a

r

t_s

B	D	E
1	a	α
3	a	β

s

t

A	B	C	D	E
α	1	α	a	α

r ⋈ s

Equating attributes of the same name, and Projecting out one copy of each pair of equated attributes

Natural Join Example

- **Relations r, 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

Cartesian product(X) followed by SELECT(σ) operation.

- Cartesian product
- Selection is based on equality on common Attributes in both relations.
- Finally removes duplicate attributes

Natural Join

- r ⋈ s **equivalent to**

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

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

In general, Consider two relations *r*(*R*) and *s*(*S*).

***R* ∩ *S* = {A1, A2, . . .,An}.**

$$r \bowtie s = \Pi_{R \cup S}\left(\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n}\, r \times s\right)$$

Quiz Q3: The natural join operation matches tuples (rows) whose values for common attributes are (1) not equal (2) equal (3) weird Greek letters (4) null

Example: Sample Relation

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Instructor

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Department

Example: Natural Join

Instructor ⋈ Department

ID	*name*	*salary*	*dept_name*	*building*	*budget*
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

Figure 2.12 Result of natural join of the *instructor* and *department* relations.

theta join

The *theta join* operation is a variant of the natural-join operation that allows us to combine a Cartesian product and a selection(based on any kind of condition between attributes) into a single operation.

Consider relations ***r*** (***R***) and ***s***(***S***), and let **ϴ** be a predicate(condition) on attributes in the schema ***R*** ∪ ***S***.

The **theta join** operation on ***r* , *s*** is defined as follows:

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$

It is equivalent to-

- Take the product **r X s**.
- Then apply $\boldsymbol{\sigma_{\theta}}$ to the result.

As for **σ, ϴ** can be any Boolean-valued condition. Historic versions of this operator allowed only A θ B, **where θ is =, <, etc**.; hence the name “theta-join.”

Theta Join Example

EMP

<u>EMPCODE</u>	NAME	Deptno	Salary
100	RAJESH	D1	100000
101	RAVI	D2	120000
102	VIJAY	D1	100000
108	AJAY	D3	140000
110	BHASKAR	D2	120000
106	RAJ	D2	150000
105	MANISH	D3	190000
106	PRSAD	D1	200000

DEPT

<u>Dno</u>	Zone	HeadOffice
D1	North	N.Delhi
D2	West	Mumbai
D3	South	Bangalore
D4	Centre	Nagpur

Note: **Deptno** is foreign key referencing **Dno**

1. Find Name of employees working in West Zone.

$$\pi_{Name}(EMP \bowtie_{Deptno=Dno \wedge Zone='West'} DEPT)$$

2. Find Name, Zone of employees drawing salary more than 150000/-

$$\pi_{name,Zone}(EMP \bowtie_{Deptno=Dno \wedge Salary>150000} DEPT)$$

Theta Join Example

1. Find Name of employees working in West Zone.

EMPCODE	NAME	Deptno	Salary	Zone	HeadOffice
100	RAJESH	D1	100000	North	N.Delhi
101	RAVI	D2	120000	West	Mumbai
102	VIJAY	D1	100000	North	N.Delhi
108	AJAY	D3	140000	South	Bangalore
110	BHASKAR	D2	120000	West	Mumbai
104	RAJ	D2	150000	West	Mumbai
105	MANISH	D3	190000	South	Bangalore
106	PRSAD	D1	200000	North	N.Delhi

$EMP \bowtie_{Deptno=Dno}$

EMPCODE	NAME	Dept no	Salary	Zone	HeadOffice
~~100~~	~~RAJESH~~	~~D1~~	~~100000~~	~~North~~	~~N.Delhi~~
101	**RAVI**	**D2**	**120000**	**West**	**Mumbai**
~~102~~	~~VIJAY~~	~~D1~~	~~100000~~	~~North~~	~~N.Delhi~~
~~108~~	~~AJAY~~	~~D3~~	~~140000~~	~~South~~	~~Bangalore~~
110	**BHASKAR**	**D2**	**120000**	**West**	**Mumbai**
104	**RAJ**	**D2**	**150000**	**West**	**Mumbai**
~~105~~	~~MANISH~~	~~D3~~	~~190000~~	~~South~~	~~Bangalore~~
~~106~~	~~PRSAD~~	~~D1~~	~~200000~~	~~North~~	~~N.Delhi~~

$EMP \bowtie_{Deptno=Dno \wedge Zone='West'} DEPT$

NAME
RAVI
BHASKAR
RAJ

$\pi_{Name}(EMP \bowtie_{Deptno=Dno \wedge Zone='West'} DEPT)$

Theta Join Example

2. Find Name, Zone of employees drawing salary more than 150000/-

EMPCODE	NAME	Deptno	Salary	Zone	HeadOffice
100	RAJESH	D1	100000	North	N.Delhi
101	RAVI	D2	120000	West	Mumbai
102	VIJAY	D1	100000	North	N.Delhi
108	AJAY	D3	140000	South	Bangalore
110	BHASKAR	D2	120000	West	Mumbai
104	RAJ	D2	150000	West	Mumbai
105	MANISH	D3	190000	South	Bangalore
106	PRSAD	D1	200000	North	N.Delhi

$EMP \bowtie_{Deptno=Dno}$

EMPCODE	NAME	Deptno	Salary	Zone	HeadOffice
105	MANISH	D3	190000	South	Bangalore
106	PRSAD	D1	200000	North	N.Delhi

$EMP \bowtie_{Deptno=Dno \wedge Salary>1500} DEPT$

$$\pi_{name,Zone}(EMP \bowtie_{Deptno=Dno \wedge Salary>150000} DEPT)$$

NAME	Zone
MANISH	South
PRSAD	North

Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a **single** (**aggregate**) value as a result.

  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

$$\boxed{{}_{G_1, G_2, \ldots, G_n} \mathcal{G}_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)}(E)}$$

E is any relational-algebra expression/ a relation

- $G_1, G_2 \ldots, G_n$ is a list of attribute/s on which to group (can be empty)
- Each F_i is an aggregate function
- Each A_i is an attribute name on which aggregate function applied.

- **Note**: Some books/articles use γ (gamma) instead of $\mathcal{G}$ Calligraphic G)

Aggregate Operation – Example

- **Relation *r*:**

A	***B***	***C***
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}$ **sum(c)** (r)

sum(*c*)
27

- **A** $\mathcal{G}$ **sum(c)** (r)

A	**sum(*c*)**
α	14
β	13

- **What is the result for the following expression ?**

  A,B $\mathcal{G}$ **sum(c)** (r)

Aggregate Operation – Example

- Find the average salary in each department

$$_{dept\_name}\mathcal{G}_{\text{avg}(salary)}(instructor)$$

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregate Operation – Example

- Find the total amount spent by each department as a salary.

$$_{dept\_name}\mathcal{G}_{Sum(salary)}(instructor)$$

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

Dept_Name	Sum(Salary)
Biology	72000
Comp.Sci	232000
Elec.Eng.	80000
Finance	170000
History	122000
Music	40000
Physics	182000

Aggregate Operation – Example

- Find the number of employees working in each department.

dept_name $\mathcal{G}$ ***Count(ID)*** ***(instructor)***

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

⇒

Dept_Name	**Count(ID)**
Biology	1
Comp.Sci	3
Elec.Eng.	1
Finance	2
History	2
Music	1
Physics	2

Outer Join – Base relations- Loan , Borrower

- Relation *loan*

loan_number	*branch_name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation borrower

customer_name	*loan_number*
Jones	L-170
Smith	L-230
Hayes	L-155

Join & Left Outer Join – Example


Outer Join – Base relations- Loan , Borrower

- Relation loan
- Relation borrower

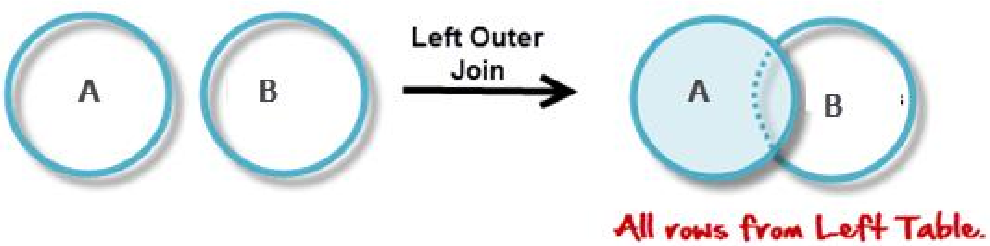
- **Join**

  loan ⋈ borrower

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

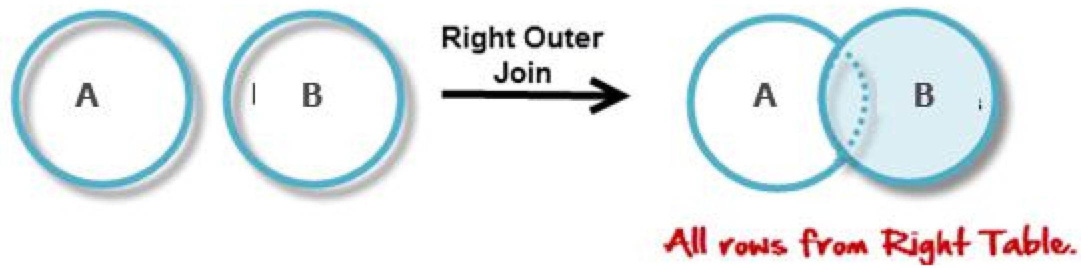
- **Left Outer Join**

  loan ⟕ borrower



loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Right Outer Join & Full Outer Join – Example



- **Right Outer Join**

  loan ⟖ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

Outer Join – Base relations- Loan , Borrower

- Relation *loan*

[illegible]	[illegible]	[illegible]
L-170	Downtown	3000
[illegible]	Redwood	[illegible]
[illegible]	Perryridge	[illegible]

- Relation borrower

[illegible]	[illegible]
Jones	L-170
Smith	L-230
Hayes	L-155

- **Full Outer Join**

  loan ⟗ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

Composition of Relational Operations

The result of a relational-algebra operation is of the same type (relation) as its inputs(relations).

Resultant_Relation ← Realtion_1 **R.Algbraic_Operation** Relation_2

Many different relational-algebra operations can be composed together into a **relational-algebra expression.**

"Find the names of all instructors in the Physics department."

$$\Pi_{name}(\sigma_{dept\ name = \text{“Physics”}}(instructor))$$

Exercise: Consider a relations **A(<u>ID</u>, Name, Age) , B(<u>ID</u>, Phone, Area) , C(<u>ID</u>, Salary, DeptNo)**

- Write a relational Algebraic expression to find ID, Name, Phone of Employees.

Assignment Operator

- The assignment operation, denoted by $\leftarrow$, works like assignment operator **=** in a programming language.
- The result to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$
- While writing a relational-algebra expression, it is convenient to assigning parts of it to temporary relation variables.
- **Eg.** $temp1 \leftarrow \boldsymbol{R} \times \boldsymbol{S}$
  - *temp1* relation variable may used in subsequent expressions.

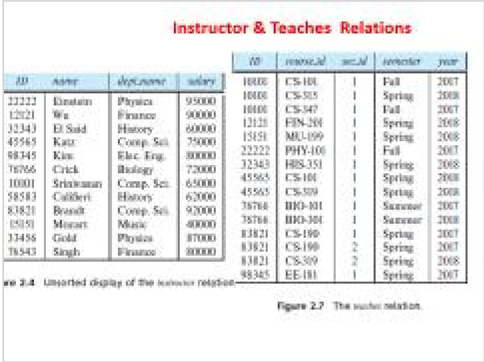
    $temp2 \leftarrow \sigma_{r.A1 = s.A1 \wedge r.A2 = s.A2 \wedge \ldots \wedge r.An = s.An}(\textbf{temp1})$

Example: **"Find the names of all instructors in the Physics department."**

$$Temp\_phy \leftarrow (\sigma_{dept\ name = \text{“Physics”}}(instructor))$$

$$\Pi_{name}(Temp\_phy)$$

Equivalent Queries



Instructor & Teaches Relations

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

re 2.4 Unsorted display of the instructor relation.

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2007
10101	CS-315	1	Spring	2008
10101	CS-347	1	Fall	2007
12121	FIN-201	1	Spring	2008
15151	MU-199	1	Spring	2008
22222	PHY-101	1	Fall	2007
32343	HIS-351	1	Spring	2008
45565	CS-101	1	Spring	2008
45565	CS-319	1	Spring	2008
76766	BIO-101	1	Summer	2007
76766	BIO-301	1	Summer	2008
83821	CS-190	1	Spring	2007
83821	CS-190	2	Spring	2007
83821	CS-319	2	Spring	2008
98345	EE-181	1	Spring	2007

Figure 2.7 The teaches relation.

Find information about courses taught by instructors in the Physics department

$$\sigma_{dept\ name=\text{“Physics”}}(instructor \bowtie_{instructor.ID=teaches.ID} teaches)$$

Both queries are equivalent

$$(\sigma_{dept\ name=\text{“Physics”}}(instructor)) \bowtie_{instructor.ID=teaches.ID} teaches$$

Example:

Find the name of Employee and their Manger Names.

EMP

<u>EMPCODE</u>	NAME	Deptno	Salary	MGR_NO
100	RAJESH	D1	100000	102
101	RAVI	D2	120000	102
102	VIJAY	D1	100000	105
108	AJAY	D3	140000	105
110	BHASKAR	D2	120000	106
104	RAJ	D2	150000	105
105	MANISH	D3	190000	106
106	PRSAD	D1	200000	

EMP1

<u>EMPCODE</u>	NAME	Deptno	Salary	MGR_NO
100	RAJESH	D1	100000	102
101	RAVI	D2	120000	102
102	VIJAY	D1	100000	105
108	AJAY	D3	140000	105
110	BHASKAR	D2	120000	106
104	RAJ	D2	150000	105
105	MANISH	D3	190000	106
106	PRSAD	D1	200000	

Note: MGR_NO must be foreign key referencing EMPCODE

$$\rho_{EMP1}(EMP)$$

$$\pi_{EMP.name, EMP1.Name}(EMP \bowtie_{MGR\_NO=EMPCODE} EMP1)$$

Aggregate Operation – Example

EMP

EMPCODE	NAME	Deptno	Salary
100	RAJESH	D1	100000
101	RAVI	D2	120000
102	VIJAY	D1	100000
108	AJAY	D3	140000
110	BHASKAR	D2	120000
106	RAJ	D2	150000
105	MANISH	D3	190000
106	PRSAD	D1	200000

DEPT

Dno	Zone	HeadOffice
D1	North	N.Delhi
D2	West	Mumbai
D3	South	Bangalore
D4	Centre	Nagpur

Note: **Deptno** is foreign key referencing **Dno**

1. Find number of employees in Centre, West Zones respectively.

$$Temp1 \leftarrow (EMP \bowtie_{Deptno=Dno \wedge (Zone='West' \vee Zone='Centre')} DEPT)$$

$$_{Zone}\mathcal{G}_{Count(ID)}(Temp1)$$

Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary} >= 85000}(\textit{instructor})$ Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}(\textit{instructor})$ Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product)	*instructor* x *department* Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
∪ (Union)	$\Pi_{name}(\textit{instructor}) \cup \Pi_{name}(\textit{student})$ Output the union of tuples from the *two* input relations.
– (Set Difference)	$\Pi_{name}(\textit{instructor}) - \Pi_{name}(\textit{student})$ Output the set difference of tuples from the two input relations.
⋈ (Natural Join)	*instructor* ⋈ *department* Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

Exercise

Consider a relations **A(ID, Name, Age) , B(EID, Phone, City) ,**

C(ID, Salary, DeptName)

Note: EID and ID derived from Same domain

Write a relational Algebraic expression –

- To find ID, Name, Phone of Employees.
- To find Name of Employees having **Salary>90000**
- To find DeptName and total salary of each department.
- To find Name of Employees who from city - **Manipal**

Solution: **A(ID, Name, Age) , B(EID, Phone, City) , C(ID, Salary, DeptName)**

Note: EID and ID derived from Same domain

- To find ID, Name, Phone of Employees.
  - **PROJECT$_{\text{ID, Name, Phone}}$ (A Theat$_{\text{ID=EID}}$ B)**
- To find Name of Employees having **Salary>90000**
  - **PROJECT $_{\text{Name}}$ (SELECT $_{\text{Salary>90000}}$ (A N.JOIN C))**

  other way is

  - **PROJECT $_{\text{Name}}$ ((SELECT $_{\text{Salary>90000}}$ (C)) N.JOIN A)**
- To find DeptName and total salary of each department.
  - **$_{\text{DeptName}}$ G $_{\text{SUM(Salary)}}$ (C)**
- To find Name of Employees who from city - **Manipal**
  - **PROJECT** $_{\text{Name}}$ (A **Theta** $_{\text{(ID=EID AND City='Manipal')}}$ B)

END

Instructor & Teaches Relations

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

re 2.4 Unsorted display of the *instructor* relation.

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

Figure 2.7 The *teaches* relation.

Instructor & Department Relations

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

jure 2.4 Unsorted display of the *instructor* relation.

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Figure 2.5 The *department* relation.