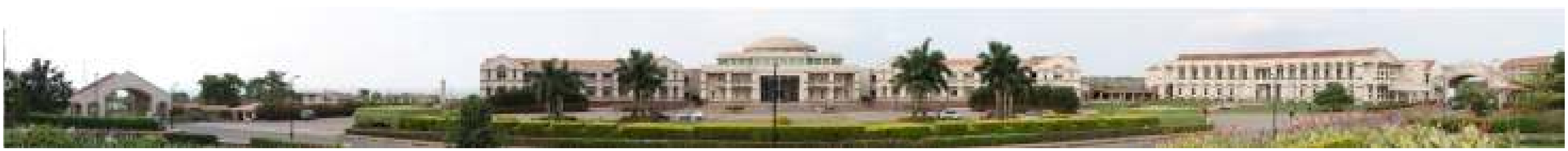
BITS, PILANI – K. K. BIRLA GOA CAMPUS

Database Systems (CS F212)

by

Dr. Mrs. Shubhangi Gawali

Dept. of CS and IS



Relational Algebra

- Six basic operators
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename
- The operators take one or more relations as inputs and give a new relation as a result.

Select Operation – Example

- Relation *r*

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

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

A	*B*	*C*	*D*
α	α	1	7
β	β	23	10

Select Operation

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

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of terms connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each term is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq. <. \leq$

- Example of selection:

$\sigma_{\textit{branch-name=“Perryridge”}}(\textit{account})$

Project Operation – Example

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}\ (r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

- Notation:

$$\Pi_{A1,\ A2,\ \ldots,\ Ak}\ (r)$$

  where A_1, A_2 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 *branch-name* attribute of *account*

$$\Pi_{account\text{-}number,\ balance}\ (account)$$

Union Operation – Example

- Relations *r, s:*

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

$r \cup s$:

A	B
α	1
α	2
β	1
β	3

Union Operation

- Notation: $r \cup s$
- Defined as: $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

For $r \cup s$ to be valid.

1. *r, s* must have the *same arity* (same number of attributes)
2. The attribute domains must be *compatible* (e.g., 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

- E.g. to find all customers with either an account or a loan

$$\Pi_{customer\text{-}name}(depositor) \cup \Pi_{customer\text{-}name}(borrower)$$

Set Difference Operation – Example

Relations

r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r – s:

A	B
α	1
β	1

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between *compatible* relations.
  - r and s must have the *same arity*
  - attribute domains of r and s must be compatible

Cartesian-Product Operation-Example

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

Cartesian-Product Operation

- Notation *r* x *s*
- Defined as:

  $$r \times s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of *r(R)* and *s(S*) are not disjoint, then renaming must be used.

Composition of Operations

- Can build expressions using multiple operations
- Example:

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

- $r\ x\ s$

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

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

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

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.

Example:

$$\rho_{x}(E)$$

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

If a relational-algebra expression *E* has arity *n*, then

$$\rho_{x\,(A1,\,A2,\,\ldots,\,An)}(E)$$

returns the result of expression *E* under the name *X*, and with the

attributes renamed to *A1, A2, …., An*.

Additional Operations

- Set intersection
- Natural join
- Division
- Assignment

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - *r*, *s* have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

Set-Intersection Operation - Example

Relation

r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cap s$

A	B
α	2

Natural Join Operation – 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	$\in$

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

Natural Join – Example

- 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

loan X borrower

loan.loan-number	*branch-name*	*amount*	*borrower.loan-number*	*customer-name*
L-170	Downtown	3000	L-170	Jones
L-170	Downtown	3000	L-230	Smith
L-170	Downtown	3000	L-155	Hayes
L-230	Redwood	4000	L-170	Jones
L-230	Redwood	4000	L-230	Smith
L-230	Redwood	4000	L-155	Hayes
L-260	Perryridge	1700	L-170	Jones
L-260	Perryridge	1700	L-230	Smith
L-260	Perryridge	1700	L-155	Hayes

loan |X| borrower

loan.loan-number	*branch-name*	*amount*	*borrower.loan-number*	*customer-name*
L-170	Downtown	3000	L-170	Jones
L-170	Downtown	3000	L-230	Smith X
L-170	Downtown	3000	L-155	Hayes X
L-230	Redwood	4000	L-170	Jones X
L-230	Redwood	4000	L-230	Smith
L-230	Redwood	4000	L-155	Hayes X
L-260	Perryridge	1700	L-170	Jones X
L-260	Perryridge	1700	L-230	Smith X
L-260	Perryridge	1700	L-155	Hayes X

loan |X| borrower

loan.loan-number	*branch-name*	*amount*	*borrower.loan-number*	*customer-name*
L-170	Downtown	3000	L-170	Jones
L-230	Redwood	4000	L-230	Smith

loan-number	*branch-name*	*amount*	*customer-name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

Division Operation – Example

Relations *r, s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
$\in$	6
$\in$	1
β	2

r

B
1
2

s

$r \div s$:

A
α
β

Another Division Example

Relations *r, s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

$r \div s$:

A	*B*	*C*
α	a	γ
γ	a	γ

Assignment Operation

- The assignment operation ($\leftarrow$) provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable.
- Example: Write $r \div s$ as

$$temp1 \leftarrow \prod_{R-S}(r)$$

$$temp2 \leftarrow \prod_{R-S}((temp1 \times s) - \prod_{R-S,S}(r))$$

$$result = temp1 - temp2$$

  - The result to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$.
  - May use variable in subsequent expressions.

Extended Relational-Algebra-Operations

- Generalized Projection
- Outer Join
- Aggregate Functions

Generalized Projection

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\Pi_{\mathrm{F1,\,F2,\,\ldots,\,Fn}}(E)$$

- E is any relational-algebra expression
- Each of F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of E.
- Given relation *credit-info(customer-name, limit, credit-balance),* find how much more each person can spend:

$$\Pi_{\textit{customer-name, limit – credit-balance}}\ (\textit{credit-info})$$

Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single 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

$$_{G1,\, G2,\, \ldots,\, Gn}\, \mathcal{G}\, _{F1(A1),\, F2(A2),\ldots,\, Fn(An)}\, (E)$$

  – E is any relational-algebra expression
  – $G_1, G_2 \ldots, G_n$ is a list of attributes on which to group (can be empty)
  – Each F_i is an aggregate function
  – Each A_i is an attribute name

Aggregate Operation – Example

- Relation *r*

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

$\mathcal{G}_{\textbf{sum(c)}}\,(\mathrm{r})$

sum-C
27

Aggregate Operation – Example

- Relation *account* grouped by *branch-name*:

branch-name	*account-number*	*balance*
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$$_{branch\text{-}name}\,\mathcal{G}_{\,sum(balance)}\,(account)$$

branch-name	*balance*
Perryridge	1300
Brighton	1500
Redwood	700

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist

Outer Join – Example

- 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

Outer Join – Example

- **Inner Join**

loan $\bowtie$ *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*

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

- **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

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations are expressed using the assignment operator.

Deletion

- A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database.
- Can delete only whole tuples; cannot delete values on only particular attributes
- A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$

where r is a relation and E is a relational algebra query.

Deletion Examples

- Delete all account records in the Perryridge branch.

$$account \leftarrow account - \sigma_{branch\text{-}name = \text{“Perryridge”}}(account)$$

- Delete all loan records with amount in the range of 0 to 50

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50}(loan)$$

- Delete all accounts at branches located in Needham.

$$r_1 \leftarrow \sigma_{branch\text{-}city = \text{“Needham”}}(account \bowtie branch)$$

$$r_2 \leftarrow \Pi_{branch\text{-}name,\ account\text{-}number,\ balance}(r_1)$$

$$r_3 \leftarrow \Pi_{customer\text{-}name,\ account\text{-}number}(r_2 \bowtie \text{depositor})$$

$$account \leftarrow \text{account} - r_2$$

$$depositor \leftarrow \text{depositor} - r_3$$

Insertion

- To insert data into a relation, we either:
  - specify a tuple to be inserted
  - write a query whose result is a set of tuples to be inserted
- in relational algebra, an insertion is expressed by:

$$r \leftarrow r \cup E$$

  where *r* is a relation and *E* is a relational algebra expression.
- The insertion of a single tuple is expressed by letting *E* be a constant relation containing one tuple.

Insertion Examples

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch.

$$account \leftarrow account \cup \{(\text{“Perryridge”}, \text{A-973}, 1200)\}$$

$$\text{depositor} \leftarrow depositor \cup \{(\text{“Smith”}, \text{A-973})\}$$

- Provide as a gift for all loan customers in the Perryridge branch, a $200 savings account. Let the loan number serve as the account number for the new savings account.

$$r_1 \leftarrow (\sigma_{branch\text{-}name\ =\ \text{“Perryridge”}}(borrower \bowtie \text{loan}))$$

$$account \leftarrow account \cup \Pi_{branch\text{-}name,\ account\text{-}number, 200}\ (r_1)$$

$$\text{depositor} \leftarrow depositor \cup \Pi_{customer\text{-}name,\ loan\text{-}number}(r_1)$$

Updating

- A mechanism to change a value in a tuple without changing *all* values in the tuple
- Use the generalized projection operator to do this task

$$r \leftarrow \prod_{F1,\, F2,\, \ldots,\, Fl,} (r)$$

- Each F_i is either
  - the *i*th attribute of *r*, if the *i*th attribute is not updated, or,
  - if the attribute is to be updated F_i is an expression, involving only constants and the attributes of *r*, which gives the new value for the attribute

Update Examples

- Make interest payments by increasing all balances by 5 percent.

$$account \leftarrow \Pi_{AN,\ BN,\ BAL\ *\ 1.05}\ (account)$$

where *AN*, *BN* and *BAL* stand for *account-number*, *branch-name* and *balance*, respectively.

- Pay all accounts with balances over \$10,000 6 percent interest and pay all others 5 percent

$$account \leftarrow \Pi_{AN,\ BN,\ BAL\ *\ 1.06}\ (\sigma_{BAL\ >\ 10000}\ (account)) \cup \Pi_{AN,\ BN,\ BAL\ *\ 1.05}\ (\sigma_{BAL\ \leq\ 10000}\ (account))$$

Relational Schema

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q1. List the name of all Employees

$$\Pi_{name} (\text{Employee})$$

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q2.List the name & telno of all junior engineers.

$$\Pi_{\text{name , telno}}(\sigma_{\textbf{post = 'junior engineer'}}(\text{Employee}))$$

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q3. List the name & post of those employees who live in mumbai & have salary > 10,000.

$$\Pi_{name\,,\,post}(\sigma_{city\ \mathbf{=\ 'Mumbai'\ \wedge\ sal>10{,}000}}(Employee))$$

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q4. Find the names of employees where projno=123;

ANS 1

$\Pi_{name}(\sigma_{pno=123 \wedge Assigned.eno=Employee.eno}$ (Assigned X Employee))

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q4. Find the names of employees where projno=123;

ANS 2

$\Pi_{name}(\sigma_{Assigned.eno=Employee.eno}(Employee \; X \; (\sigma_{pno=123}(Assigned))$

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q5. Find the location where employee no 5 has worked.

ANS 1

$\Pi_{\text{location}} (\sigma_{\text{eno=5} \wedge \text{Assigned.pno=project.pno}} (\text{Project X Assigned}))$

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q5. Find the location where employee no 5 has worked.

ANS 2

$\Pi_{location}(\sigma_{Assigned.pno=project.pno}(Project \; X \; (\sigma_{eno=5}(Assigned))))$

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q6. Find the city of the managers of all the projects located at Dadar. Assume Managerid field contains eno.

$\Pi_{city} (\sigma_{location = \text{'Dadar'} \; \mathbf{\Lambda\ Eno= Managerid}} (Project \times Employee))$

- Employee(eno,Name,Telno,post,DOJ, sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q7. Find the eno who are assigned the pno 123 after the manager of the project joined the organisation.

$$\Pi_{eno} (\sigma_{date>DOJ \wedge Assigned.pno=123} (Assigned \times (\sigma_{eno=managerid}(Employee \times (\sigma_{pno = 123} (Project))))))$$

- $A \leftarrow (\sigma_{pno=123}(Project))$

- $B \leftarrow (\sigma_{eno=managerid}\ (Employee \times A))$

- $C \leftarrow (\sigma_{date>DOJ\ \boldsymbol{\wedge}\ \mathbf{Assigned.pno=123}}$
  $(Assigned \times B))$

- $R \leftarrow \prod_{eno} (C)$

- Employee(eno,Name,Telno,post,DOJ, sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q7. Find the eno who are assigned the pno 123 after the manager of the project joined the organisation.

$\Pi_{eno}(\sigma_{date>DOJ \land Assigned.pno=123 \land Project.pno = 123 \land eno=Managerid}$ (Assigned X Employee X Project))

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q8. Find the Employees who are living in the same city as that of employee ‘XYZ’.

$temp1 \leftarrow \Pi_{city} (\sigma_{Name= \text{'XYZ'}} \text{ Employee})$

$\Pi_{eno} (\sigma_{Name<> \text{'XYZ'} \wedge \text{ temp1.city = Employee.city}} (\text{temp1 X Employee}))$

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q8. Find the Employees who are living in the same city as that of employee ‘XYZ’.

$$\Pi_{Name}(\sigma_{Name= \text{'XYZ'} \wedge E.city = Employee.city \wedge E.Name <> \text{'XYZ'}}(Employee \times \rho_E Employee))$$

- Employee(eno,Name,Telno,post,DOJ,sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q9. Find the complete details of all the employees who are assigned to project no 123.

$$\text{Employee} \mid X \mid \Pi_{Eno} (\sigma_{pno=123} (\text{Assigned})$$

- Employee(eno,Name,Telno,post,DOJ, sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q10. Find the employee who are working at the same location where they reside.

$\Pi_{name} (\sigma_{city = location}$ (Project |X| Assigned |X| Employee)

- **Employee(eno,Name,Telno,post,DOJ, sal,city)**
- **Project (pno,Managerid,location)**
- **Assigned(eno,pno,date,task)**

Q11.Find the employees who are working with employee xyz.

Step1: Find employee whose name is xyz

$\sigma_{\textbf{name = 'xyz'}}(\text{Employee})$

Step2: Projects on which xyz is working

$A \longleftarrow \Pi_{pno}(\text{Assigned } |X| (\sigma_{\textbf{name = 'xyz'}}(\text{Employee})$

Step3: Employees working on that project

$B \longleftarrow \Pi_{eno}(\text{Assigned } |X| \ A)$

Step 4: Name of Employees working on the project

$\Pi_{eno,name}(\sigma_{\textbf{name <> 'xyz'}}(\text{ Employee } |X| \ B)$

- Employee(eno,Name,Telno,post,DOJ, sal,city)
- Project (pno,Managerid,location)
- Assigned(eno,pno,date,task)

Q12. Find the projects whose location is same as their manager's city.

$\Pi_{pno}(\sigma_{manager=eno\ \boldsymbol{\wedge}\ \mathbf{city=\ location}}$ (Project X Employee))

Q13. Find Maximum salary

Employee

Name	Salary
A	10k
B	20k
C	30k

E

E.Name	E.Salary
A	10k
B	20k
C	30k

Employee X E

Emp. Name	Emp. Sal	E. Name	E. Sal
A	10k	A	10k
A	10k	B	20k
A	10k	C	30k
B	20k	A	10k
B	20k	B	20k
B	20k	C	30k
C	30k	A	10k
C	30k	B	20k
C	30k	C	30k

Emp. Name	Emp. Sal
A	10K
B	20K
C	30K

—

E. Name	E. Sal
A	10K
A	10K
B	20K

Emp.Name	Emp. Sal
C	30k

Solution

$\Pi_{\text{emp.sal}}(\text{Emp}) - \Pi_{\text{E.sal}}(\sigma_{\text{emp.sal>E.sal}}(\text{Employee X } \rho_E \text{Employee}))$

OR

$\Pi_{\text{emp.sal}}(\text{Emp}) - \Pi_{\text{Emp.sal}}(\sigma_{\text{emp.sal<E.sal}}(\text{Employee X } \rho_E \text{Employee}))$

For Finding Minimum salary

$\Pi_{\text{emp.sal}}(\text{Emp}) - \Pi_{\text{E.sal}}(\sigma_{\text{emp.sal<E.sal}}(\text{Employee X } \rho_E \text{Employee}))$

Exercise 2

Database schema

- Professor(<u>ssn</u>, profname, status, salary)
- Course(<u>crscode</u>, crsname, credits)
- Taught(<u>crscode, semester</u>, ssn)

Assumptions: (1) Each course has only one instructor in each semester; (2) all professors have different salaries; (3) all professors have different names; (4) all courses have different names; (5) status can take values from “Full”, “Associate”, and “Assistant”.

- Find those professors who have taught ‘C1’ but have never ‘C2’.
- Find those professors who have taught both ‘C2’ and ‘C3’.
- Find those professors who have never taught ‘C2’.
- Find those professors who taught ‘C2’ and ‘C3’ in the same semester.

- Find those professors who taught ‘C1’ or ‘C2’ but not both.
- Find those courses that have never been taught.
- Find those courses that have been taught at least in two semesters.
- Find the names of professors who ever taught ‘C3’.

- Find the names of Assistant Professors who ever taught ‘C2’.
- Find the names of professors who ever taught at least two courses in one semester.
- List all the course names that professor ‘Dr. Bhoomi Desai’ taught in Even sem of 2009.
- List those courses that have been taught ONLY by Assistant professors.