



BITS, PILANI – K. K. BIRLA GOA CAMPUS

Database Systems and Applications (IS F243)

by

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:

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

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

Each **term** is one of:

$\langle \text{attribute} \rangle \text{ op } \langle \text{attribute} \rangle$ or $\langle \text{attribute} \rangle$ or $\langle \text{constant} \rangle$

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

- Example of selection:

$\sigma_{\text{branch-name}=\text{"Perryridge"}}(\text{account})$

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:

$$\Pi_{A_1, A_2, \dots, A_k}(r)$$

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*

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

Union Operation – Example

- Relations r, s :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cup s$:

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

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

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

$r \times s$:

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

Cartesian-Product Operation

- Notation $r \times s$
- 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:

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

- $r \times s$

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
α	1	α	10	<i>a</i>
α	1	β	10	<i>a</i>
α	1	β	20	<i>b</i>
α	1	γ	10	<i>b</i>
β	2	α	10	<i>a</i>
β	2	β	10	<i>a</i>
β	2	β	20	<i>b</i>
β	2	γ	10	<i>b</i>

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

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
α	1	α	10	<i>a</i>
β	2	β	20	<i>a</i>
β	2	β	20	<i>b</i>

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

$$\rho_x(A1, A2, \dots, An)(E)$$

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

attributes renamed to $A1, A2, \dots, An$.

Additional Operations

- Set intersection
- Natural join
- Division
- Assignment

Set-Intersection Operation

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

Natural Join – Example

- Relation *loan*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation *borrower*

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

loan X borrower

<i>loan.loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>borrower.loan-number</i>	<i>customer-name</i>
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

<i>loan.loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>borrower.loan-number</i>	<i>customer-name</i>
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

<i>loan.loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>borrower.loan-number</i>	<i>customer-name</i>
L-170	Downtown	3000	L-170	Jones
L-230	Redwood	4000	L-230	Smith

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>
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

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

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

$$result = temp1 - temp2$$

- 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.

$$\Pi_{F_1, F_2, \dots, F_n}(E)$$

- E is any relational-algebra expression
- Each of F_1, F_2, \dots, F_n 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:

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

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

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

- E is any relational-algebra expression
- G_1, G_2, \dots, 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

$g_{\text{sum}(c)}(r)$

$\text{sum-}C$
27

Aggregate Operation – Example

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

<i>branch-name</i>	<i>account-number</i>	<i>balance</i>
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

branch-name $g_{sum(balance)}$ (*account*)

<i>branch-name</i>	<i>balance</i>
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*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation *borrower*

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

Outer Join – Example

- **Inner Join**

loan ⋈ *Borrower*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- **Left Outer Join**

loan ⋈_L *Borrower*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	<i>null</i>

Outer Join – Example

- **Right Outer Join**

loan ⋈_r *borrower*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	<i>null</i>	<i>null</i>	Hayes

- **Full Outer Join**

loan ⋈_f *borrower*

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	<i>null</i>
L-155	<i>null</i>	<i>null</i>	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.

$account \leftarrow account - \sigma_{branch-name = "Perryridge"}(account)$

- 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.

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

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

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

$account \leftarrow account - r_2$

$depositor \leftarrow depositor - r_3$

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.

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

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

- 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.

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

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

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

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

$$r \leftarrow \Pi_{F_1, F_2, \dots, F_l}(r)$$

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

Q1. List the name of all Employees

- 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.

- 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.

- 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;

- 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.

- 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 Manager field contains eno.

- 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.

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

Q8. Find the Employee who are living in the same city as 'XYZ'.

- 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.

- 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.

- 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.