Relational Algebra

S.PRABHAVATHI
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SEIENCE & IT
JAMAL MOHAMED COLLEGE
TRICHY – 620 020

Relational Algebra

- ✓ Procedural language
- ✓ The fundamental operations in the relational algebra
- ❖ **SIX BASIC OPERATORS**
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- ✓ The operators take one or two relations as inputs and produce a new relation as a result.
- ❖ **SEVERAL OTHER OPERATIONS**
  - Set intersection,
  - Natural join,
  - Division
  - Assignment

➢ The select, project, and rename operations are called **unary operations,** because they operate on one relation.

➢ Other 3 operations operate on pairs of relations.
  - union: $\cup$
  - set difference: –
  - Cartesian product: x - This can be called **binary operation.**

1.Select Operation

⇒ The select operation sects tuples that satisfy a given predicate.

⇒ We use the lowercase Greek letter sigma(σ) to denote selection.
⇒ Predicate appears as a subscript to σ.

Notation: $\boldsymbol{\sigma p(r)}$
p is called the **selection predicate**

Defined as:

$$\sigma P(\mathbf{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

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

Example of selection:

σ *branch_name="Perryridge"(account)*

loan-number	branch-name	amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Figure 3.6 The *loan* relation.

account-number	branch-name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Figure 3.1 The *account* relation.

Result of selection

Accoumt – number Select Operation	Branch name	Balance
A102	*Perryridge*	400

Example

1. We can find all tuples in more than $1200 by writing
   σ amount > 1200(loan)

   σ branch – name =" Perryridge" $\wedge$ amount > 1200(loan)
   Comparisons between **two attributes.**

Three attributes
customer-name, banker – name and loan –officer.
Banker is the loan officer – loan that belong to some customer.
To find all customers who have the same name as their loan officer

σ cus-name = banker – name (loan-officer)

Special value null indicates.
Any comparisons involving a null value evaluate to false.

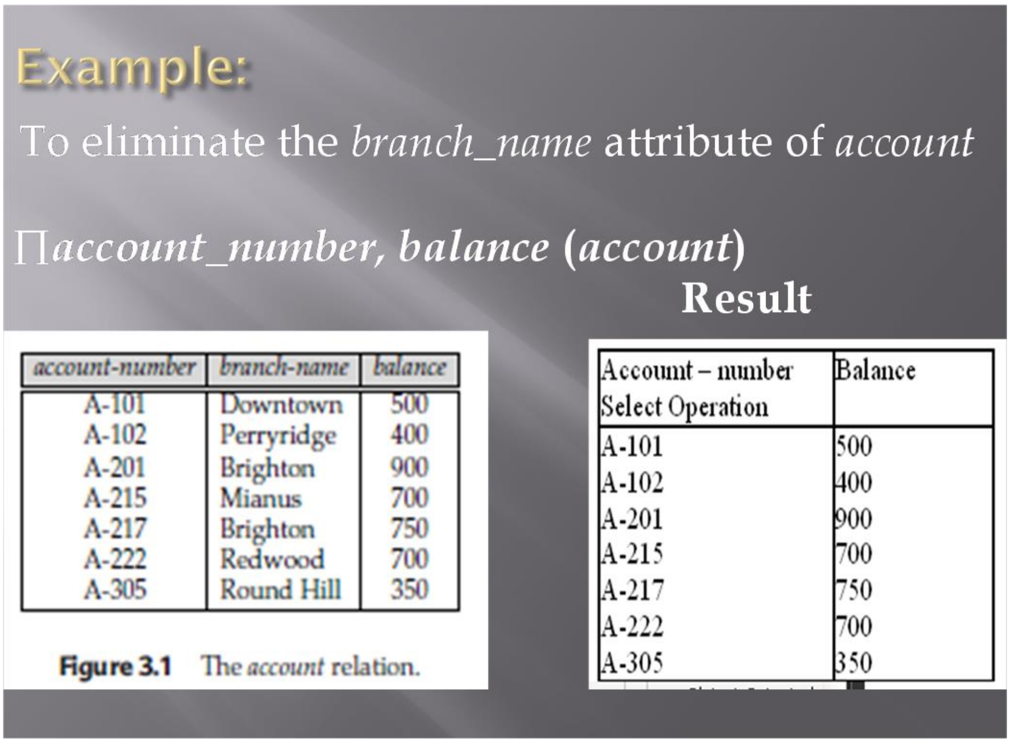
2.Project Operation

Notation: $\prod_{A_1, A_2, \ldots, A_k}(r)$

where *A1*, *A2* are attribute names and *r* is a relation name.

- List all account number and the balance of the account , but not care about the branch name.
- Projection operation is a unary operation that returns its argument relation with certain attributes left out.
- Any duplicate rows are eliminated.
- Projection is denoted b Greek letter pi (π)
- We list those attributes that we wish to appear in the result as a subscript to π. The argument relation follows in parentheses.

Example:



account-number	branch-name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Figure 3.1 The *account* relation.

Account – number Select Operation	Balance
A-101	500
A-102	400
A-201	900
A-215	700
A-217	750
A-222	700
A-305	350

COMPOSITION OF RELATIONAL OPERATIONS

- The fact that the result of a relational operation is itself a relation is important.
- More complicated query "find those customers who live in " Trichy " we write:
  *π customer – name (σ customer – city=" *Perryridge*" (customer))*
- relational algebra is just like composing arithmetic operations (such +, -, * and ÷) into arithmetic expressions.

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**
   (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

⇒ A query to find the names of all bank customers who have either on account or a loan (or) both.
⇒ We need the information in depositor relation and in the borrower relation.

Find the names of all customers with a loan in the bank:

π customer – name (borrower)

Find the names of all customers with an account in the bank:

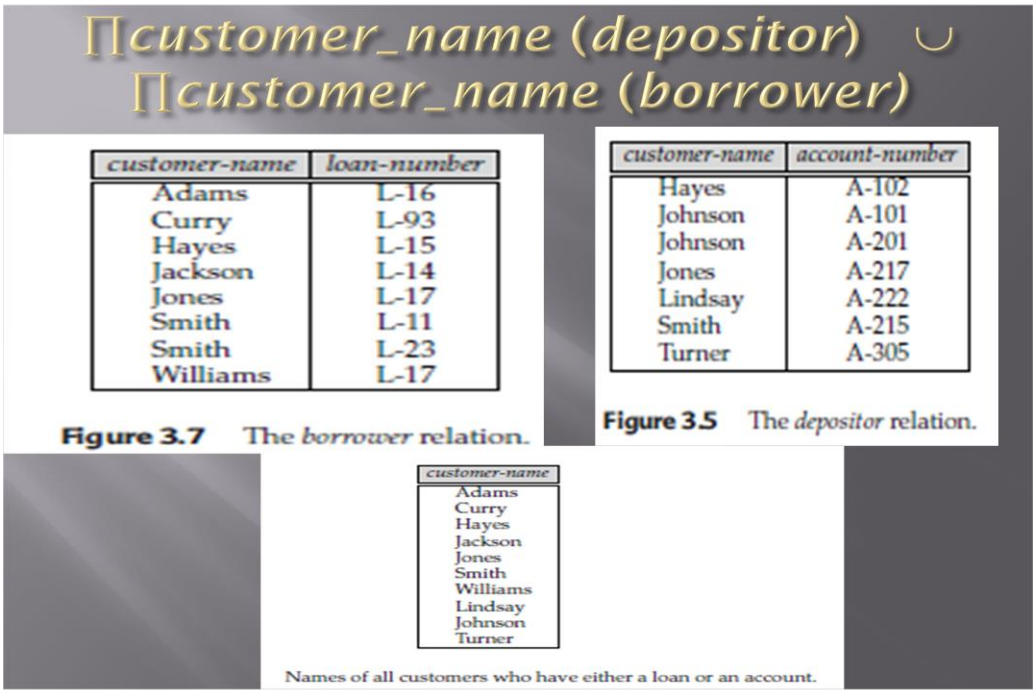
π customer – name (depositor)

⇒ We need the union of these two sets;
⇒ All customer name that appear in either or both of two relations.
⇒ Binary operation union, denoted, as in set theory, by U.
⇒ Names of all customer who have either a loan or an account

Example: to find all customers with either an account or a loan

$$\Pi_{customer\_name}(depositor) \cup \Pi_{customer\_name}(borrower)$$

- relations are sets, duplicate values are eliminated.
- Unions are taken between compatible relations.



customer-name	loan-number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Figure 3.7 The *borrower* relation.

customer-name	account-number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Figure 3.5 The *depositor* relation.

customer-name
Adams
Curry
Hayes
Jackson
Jones
Smith
Williams
Lindsay
Johnson
Turner

Names of all customers who have either a loan or an account.

Example: Intersection

S1

sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

S2

sid	name	gpa
53666	Jones	3.4
53688	Smith	3.2
53700	Tom	3.5
53777	Jerry	2.8
53832	Guldu	2.0

$S1 \cap S2 = S1 - (S1 - S2)$

$S1 \cap S2 =$

sid	name	gpa
53666	Jones	3.4
53688	Smith	3.2
53832	Guldu	2.0

4.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

- Set difference operation, denoted by -, allows us to find tuples that are in one relation but are not in another.
- The expression r – s results in a relation containing those tuples in r but not in s.
- *Customer with an account but not loan*

  Customer – name
  Johnson
  Turner
  Lindsay
- Find all customers in bank, who have an account but not a loan.
  π *customer – name (depositor)* – π *customer – name(borrower)*
- set difference are taken between compatible relations.

Set Difference Operation – Example

- Relations *r*, *s*:

r

A	B
α	1
α	2
β	1

s

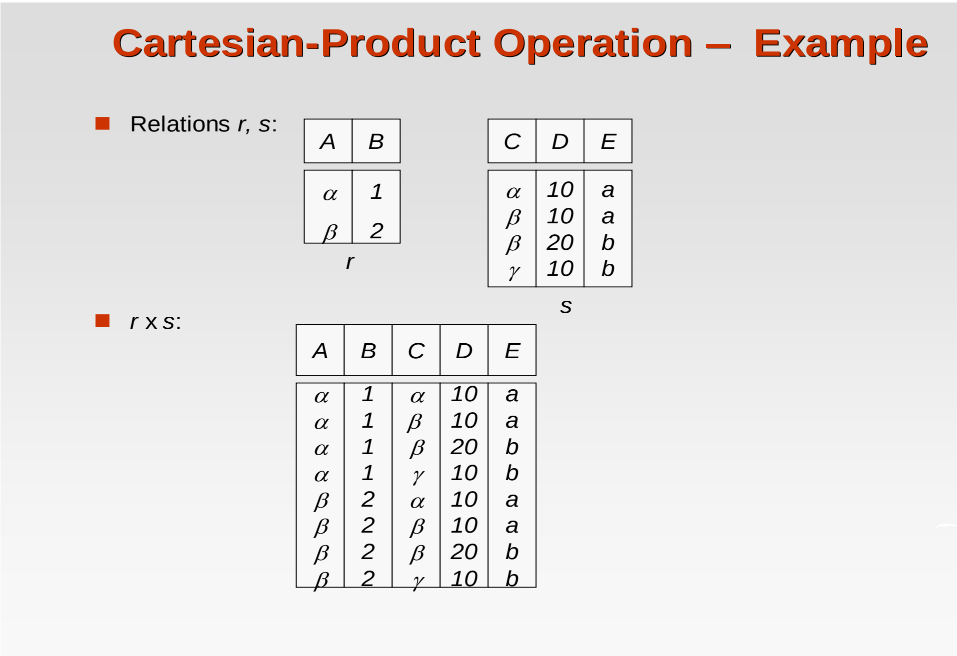
A	B
α	2
β	3

- *r – s:*

A	B
α	1
β	1

5. Cartesian-Product Operation

- Denoted by a cross (x),
- Combine information from any two relations.
- Cartesian product of relations r1 & r2 and r1 * r2.
- Cartesian product of a set of domains.
- Relation schema for r = borrower * loan is
- (borrower. customer – name, borrower. Loan – number, loan. Branch – name, loan. Loan – number, loan . amount



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

6.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

  returns the result of expression *E* under the name *X*, and with the attributes renamed to *A1 , A2 , …., An* .
- RENAME operation – which can rename either the relation name or the attribute names, or both
- The general RENAME operation ρ can be expressed by any of the following forms:
  - $\rho_S(R)$ changes:
    - the *relation name* only to S
  - $\rho_{(B1, B2, \ldots, Bn)}(R)$ changes:
    - the *column (attribute) names* only to B1, B1, …..Bn
  - $\rho_{S\,(B1, B2, \ldots, Bn)}(R)$ changes both:
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn

7.DIVISION:

(a)

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

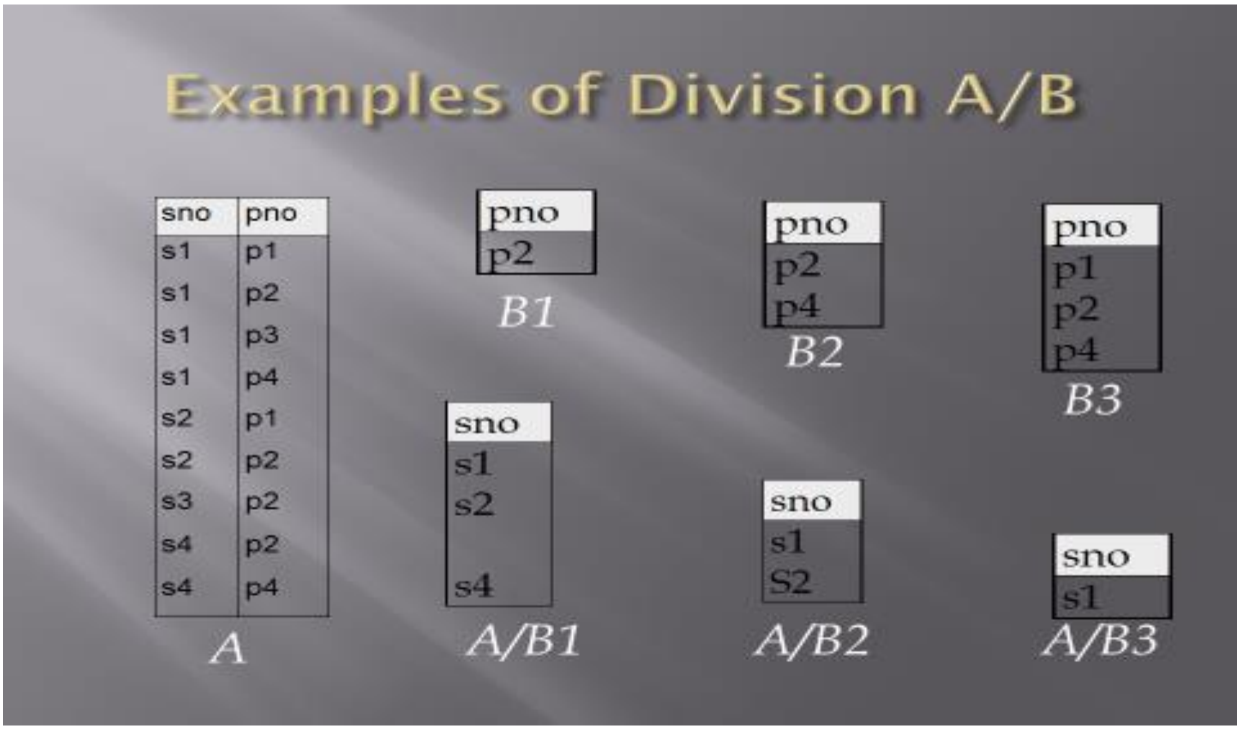
S

A
a1
a2
a3

T

B
b1
b4

Figure 6.8
The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.



8.JOIN

Natural -Join

employee-name	street	city
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

employee-name	branch-name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

gure 3.31 The *employee* and *ft-works* relations.

employee-name	street	city	branch-name	salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500

Figure 3.32 The result of *employee* ⋈ *ft-works*.

employee ⋈ *ft-works*

The result of this expression appears in Figure 3.32. Notice that we have lost the street and city information about Smith, since the tuple describing Smith is absent from the *ft-works* relation; similarly, we have lost the branch name and salary information about Gates, since the tuple describing Gates is absent from the *employee* relation.

Types of Outer – Join

We can use the *outer-join* operation to avoid this loss of information. There are actually three forms of the operation: *left outer join*, denoted ⟕; *right outer join*, denoted ⟖; and *full outer join*, denoted ⟗. All three forms of outer join compute the join, and add extra tuples to the result of the join. The results of the expressions

employee-name	street	city
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

Left outer join

employee-name	branch-name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

The **left outer join** (⟕) takes all tuples in the left relation that did not match with any tuple in the right relation, pads the tuples with null values for all other attributes from the right relation, and adds them to the result of the natural join. In Figure 3.33, tuple (Smith, Revolver, Death Valley, *null*, *null*) is such a tuple. All information from the left relation is present in the result of the left outer join.

employee-name	street	city	branch-name	salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Smith	Revolver	Death Valley	*null*	*null*

Figure 3.33 Result of *employee* ⟕ *ft-works*.

employee-name	street	city
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

Right outer join

employee-name	branch-name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

The **right outer join** (⟖) is symmetric with the left outer join: It pads tuples from the right relation that did not match any from the left relation with nulls and adds them to the result of the natural join. In Figure 3.34, tuple (Gates, *null*, *null*, Redmond, 5300) is such a tuple. Thus, all information from the right relation is present in the result of the right outer join.

employee-name	street	city	branch-name	salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Gates	*null*	*null*	Redmond	5300

Figure 3.34 Result of *employee* ⟖ *ft-works*.

Full outer join

The **full outer join**(⟗) does both of those operations, padding tuples from the left relation that did not match any from the right relation, as well as tuples from the right relation that did not match any from the left relation, and adding them to the result of the join. Figure 3.35 shows the result of a full outer join.

employee-name	*street*	*city*	*branch-name*	*salary*
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Smith	Revolver	Death Valley	*null*	*null*
Gates	*null*	*null*	Redmond	5300

Figure 3.35 Result of *employee* ⟗ *ft-works*.