RELATIONAL ALGEBRA

Objectives

- Understand the meaning of symbols used in relational algebra.
- How to form queries in relational algebra.

Introduction

- **Relational algebra :** is a theoretical language with operations that work on one or more relations to define another relation without changing the original relation(s).

- *Relational Algebra is a basis for other Data Manipulation Languages for relational databases-illustrates the basic operations required for any DML.*

Relational Algebra

5 basic operations in relational algebra:

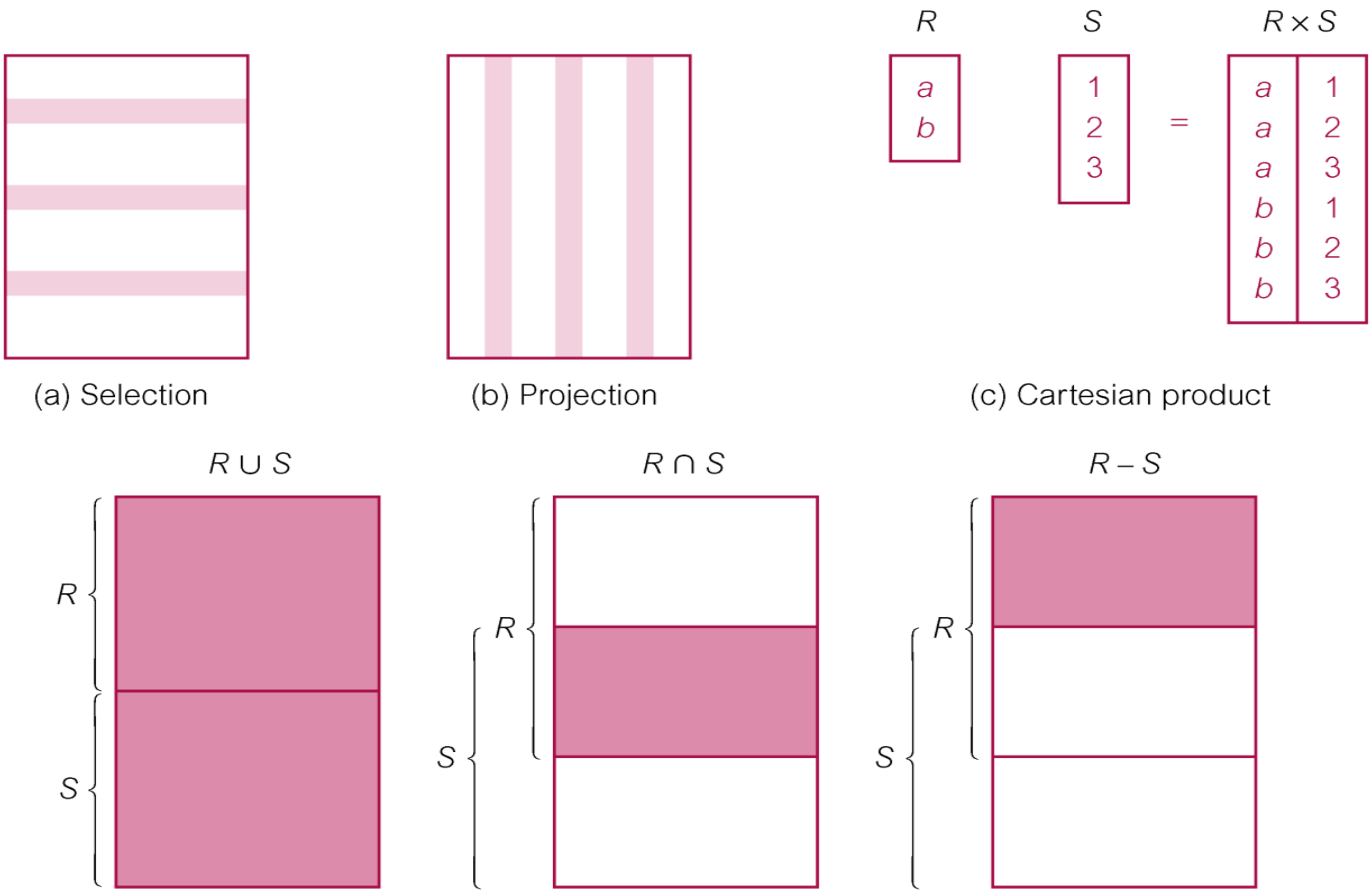
- Selection, Projection, Cartesian product, Union, and Set Difference.
- These perform most of the data retrieval operations needed.
- Also have Join, Intersection, and Division operations, which can be expressed in terms of 5 basic operations.

Relational Algebra Operations cont’d

- These operations can be categorised into two groups; unary and binary operations.
- **Unary Operations**: These operate on one relation. Examples of unary operations include; selection and projection
- **Binary Operations**; These work on pairs of relations. Examples include the; Cartesian product, union, set difference, join, intersections and division operations

Relational Algebra Operations

(illustrations showing the functions of the relational algebra operations)



(a) Selection (b) Projection (c) Cartesian product

(d) Union (e) Intersection (f) Set difference

Relational Algebra Operations

T

A	B
a	1
b	2

U

B	C
1	x
1	y
3	z

$T \bowtie U$

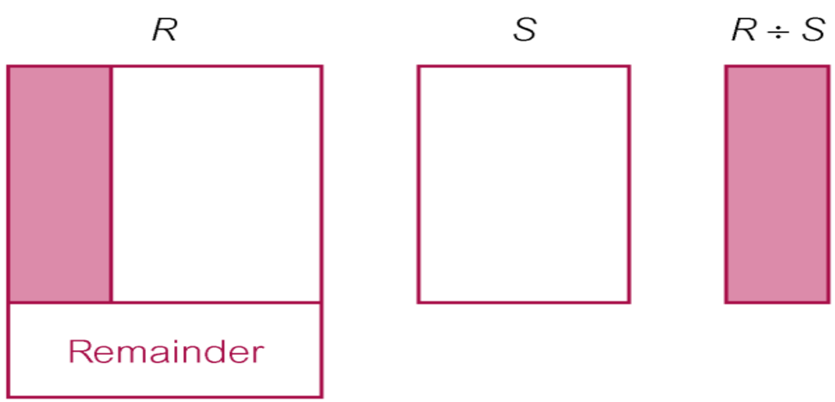
A	B	C
a	1	x
a	1	y

(g) Natural join

$T ⟕_C U$

A	B	C
a	1	x
a	1	y
b	2	

(i) Left Outer join



(j) Division (shaded area)

V

A	B
a	1
a	2
b	1
b	2
c	1

W

B
1
2

$V \div W$

A
a
b

Example of division

Selection (or Restriction)

- $\sigma_{predicate}$ (R)
- Purpose : Picks rows according to some criteria.
  - Works on a single relation R and defines a relation that contains only those tuples (rows) of R that satisfy the specified condition (*predicate)*.
  - More complex predicates can be generated using the logical operators; ∧ (AND), ∨ (OR) and ~ (NOT)
  - Eg;
  - σ salary>10000 **AND** σ SALARY <2000 (Staff)

Table 5.1 Result table for Example 5.1.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

STAFF TABLE

Example - Selection (or Restriction)

- List all staff with a salary greater than £10,000.

$$\sigma_{salary > 10000} \textbf{(Staff)}$$

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24- Mar-58	18000	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003

Projection

Purpose : Picks some of the attributes of the relation.

- $\Pi_{col1, \ldots, coln}(R)$
  - Works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.
  - For example : Picking Reg No's and all names of students.

Example - Projection

- Produce the names and respective marks from student table

$$\Pi_{\text{fName, marks}}(\text{Student})$$

Name	Age	Regno	Marks
Peter	19	S09b13/311	80
Sam	20	S09b13/230	75
Mary	21	S09b13/567	65

Name	Marks
Peter	80
Sam	75
Mary	65

11

Union

- $R \cup S$
  - Union of two relations R and S defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated.
  - (Union) is the relation containing all tuples that appear in R1, R2, or both
- If R and S have *I* and *J* tuples, respectively, union is obtained by concatenating them into one relation with a maximum of (*I* + *J*) tuples.

Example – Union

The union of two relations R and S defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated.

- List all cities where there is either a branch office or a property for rent.

$$\Pi_{city}(Branch) \cup \Pi_{city}(PropertyForRent)$$

city
London
Aberdeen
Glasgow
Bristol

Make use of Connolly & BEGG Page 80. PropertyForRent & Branch Table.

Set Difference

- R – S
  - Defines a relation consisting of the tuples that are in relation R, but not in S.

Example - Set Difference

- List all cities that are in the branch office but not in property for rent.

$$\Pi_{city}(\text{Branch}) - \Pi_{city}(\text{PropertyForRent})$$

city
Bristol

15

Intersection (And)

- $R \cap S$
  - Defines a relation consisting of the set of all tuples that are in both R and S.
- List all cities where there is both a branch office and at least a property for rent.

$$\Pi_{city}(Branch) \cap \Pi_{city}(PropertyForRent)$$

city
Aberdeen
London
Glasgow

16

Cartesian product
Purpose: Pairs rows from 2 tables.

- R X S
  - Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S; *relation with all possible pairs of tuples from the two relations.*
  - In case the relations have attributes with the same names, these are prefixed with the relation name to maintain the *uniqueness* of attribute names in the resulting relation.
  - Output: For each row r in R and each row s in S, output a row rs; the output table has the columns of R and the columns of S
  - Note we can express the intersection operation in terms of set difference. R n S=R-(R-S)

Example - Cartesian Product

- List the names and comments of all clients who have viewed a property for rent.

$(\Pi_{\text{clientNo, fName, lName}}(\text{Client}))\ \text{X}\ (\Pi_{\text{clientNo, propertyNo,comment}}(\text{Viewing}))$

Example - Cartesian Product cont'd

client.clientNo	fName	lName	Viewing.clientNo	propertyNo	comment
CR76	John	Kay	CR56	PA14	too small
CR76	John	Kay	CR76	PG4	too remote
CR76	John	Kay	CR56	PG4	
CR76	John	Kay	CR62	PA14	no dining room
CR76	John	Kay	CR56	PG36	
CR56	Aline	Stewart	CR56	PA14	too small
CR56	Aline	Stewart	CR76	PG4	too remote
CR56	Aline	Stewart	CR56	PG4	
CR56	Aline	Stewart	CR62	PA14	no dining room
CR56	Aline	Stewart	CR56	PG36	
CR74	Mike	Ritchie	CR56	PA14	too small
CR74	Mike	Ritchie	CR76	PG4	too remote
CR74	Mike	Ritchie	CR56	PG4	
CR74	Mike	Ritchie	CR62	PA14	no dining room
CR74	Mike	Ritchie	CR56	PG36	
CR62	Mary	Tregear	CR56	PA14	too small
CR62	Mary	Tregear	CR76	PG4	too remote
CR62	Mary	Tregear	CR56	PG4	
CR62	Mary	Tregear	CR62	PA14	no dining room
CR62	Mary	Tregear	CR56	PG36	

Join Operations

- This is one of the essential operations in relational algebra; *a join operation combines two relations to form a new one.*

- A Join is a derivative of Cartesian product; Equivalent to performing a Selection, using join predicate as selection formula, over Cartesian product of the two operand relations *(Cartesian product that satisfies a given condition(s))*.

- One of the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMSs have intrinsic performance problems.

Join Operations

- Various forms of join operation
  - Theta join
  - Equi -join
  - Natural join
  - **Outer join**
  - Left Outer Join
  - Right Outer Join.

Theta join (θ-join)

- $R \bowtie S$
  - Defines a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S.

  - The predicate F is of the form $R.a_i \; \theta \; S.b_i$ where θ may be one of the comparison operators ($<, \leq, >, \geq, =, \neq$).

Theta join (θ-join)

- Can rewrite Theta join using basic Selection and Cartesian product operations.

$$R \bowtie_F S = \sigma_F(R \text{ X } S)$$

- Degree of a Theta join is sum of degrees of the operand relations R and S; *as the Cartesian product*.

- If predicate F contains only equality (=), the term Equijoin is used.

The Natural join

- Determine the common attributes by looking for attributes with identical names.
- Select only the rows with common values in the common attributes.

Example Theta join

Suppose a customer wants to buy a SAM and HTC, but doesn't want 2 spend more money for the HTC than for the SAM Samsung price > HTC Price

SAMSUNG	SAMSUNG Price
SAM A	20000
SAM B	30000
SAM C	50000

HTC model	HTC Price
HTC 1	10000
HTC 2	40000
HTC3	60000

SAM Model	SAM Price	HTC Model	HTC Price
SAM A	20000	HTC 1	10000
SAM B	30000	HTC 1	10000
SAM C	50000	HTC 1	10000
SAM C	50000	HTC 2	40000

Theta Join

SAM Model	SAM Price	HTC Model	HTC Price
SAM A	20000	HTC 1	10000
SAM B	30000	HTC 1	10000
SAM C	50000	HTC 1	10000
SAM C	50000	HTC 2	40000

Example – Equijoin

Qn . Generate the resultant equi join from the Employee and Dept Table below

Employee TABLE		
NAME	**EMP ID**	**DEPT NAME**
HARRY	3415	FINANCE
SALLY	2241	SALES
GEORGE	3401	FINANCE
HARRIET	2202	SALES

DEPT TABLE	
DEPT NAME	**MANAGER**
FINANCE	GEORGE
SALES	HARRIET
PRODN	CHARLES

NAME	EMPID	DEPTNAME	MANAGER
HARRY	3415	FINANCE	GEORGE
SALLY	2241	SALES	HARRIET
GEORGE	3401	FINANCE	GEORGE
HARRIET	2202	SALES	HARRIET

Natural Join

- $R \bowtie S$
  - An Equijoin of the two relations R and S over all common attributes *x*. One occurrence of each common attribute is eliminated from the result.
  - The degree of a natural join is the sum of the degrees of the relations R and S less the number of attributes in x.

Example

VIEWING

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-01	Too small
CR62	PA14	14-May-01	No dinning room
CR76	PG4	20-Apr-01	Too remote
CR56	PG4	26-May-01	
CR56	PG36	28-Apr-01	

CLIENT

clientNo	fName	lName	Sex
CR56	Aline	Stewart	F
CR62	Mary	Tregar	F
CR74	Mike	Ritchie	M
CR76	John	Kay	M

Example - Natural Join

- List the names and comments of all clients who have viewed a property for rent.

$(\Pi_{clientNo,fName,lName}(Client)) \bowtie (\Pi_{clientNo,propertyNo,comment}(Viewing))$

clientNo	fName	lName	propertyNo	comment
CR76	John	Kay	PG4	too remote
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR56	Aline	Stewart	PG36	
CR62	Mary	Tregear	PA14	no dining room

Outer join

- To display rows in the result that do not have matching values in the join column, use Outer join.

- Advantage: preservation of information (that would otherwise be lost).

- There are basically three types of outer join operations
  - Left Outer join
  - Right Outer join
  - Full Outer join

Outer Join

To display rows in the result that do not have matching values in the join column, use Outer join.

R S

(Left) Outer join is the join in which tuples from R that do not have matching values in common columns with S are also included in the result relation.

A	B
a	1
b	2

B	C
1	x
1	y
3	z

A	B	C
a	1	x
a	1	x
b	2	Null

Left Outer join ( )

- R ⟕ S
  - (Left) outer join is join in which tuples from R that do not have matching values in common columns of S are also included in result relation.
  - The missing values in the second relation are set to null.
  - Returns all values from the left table , plus matched values from the left table (or Null in case of No matching join predicate)

Example

PropertyForRent

propertyNo	street	city	rooms
PA14	16 Holhead	Aberdeen	4
PL94	6 Argyll St	London	2
PG4	6 Lawrence St	Glasgow	3
PG16	5 Novar Dr	Glasgow	2
PG21	18 Dale Rd	Glasgow	6
PG36	2 Manor Rd	Glasgow	3

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-01	Too small
CR62	PA14	14-May-01	No dinning room
CR76	PG4	20-Apr-01	Too remote
CR56	PG4	26-May-01	
CR56	PG36	28-Apr-01	

Example - Left Outer join

- Produce a status report on property viewings.

$\Pi_{\text{propertyNo,street,city}}$(PropertyForRent) ⟕ Viewing

propertyNo	street	city	clientNo	viewDate	comment
PA14	16 Holhead	Aberdeen	CR56	24-May-01	too small
PA14	16 Holhead	Aberdeen	CR62	14-May-01	no dining room
PL94	6 Argyll St	London	null	null	null
PG4	6 Lawrence St	Glasgow	CR76	20-Apr-01	too remote
PG4	6 Lawrence St	Glasgow	CR56	26-May-01	
PG36	2 Manor Rd	Glasgow	CR56	28-Apr-01	
PG21	18 Dale Rd	Glasgow	null	null	null
PG16	5 Novar Dr	Glasgow	null	null	null

Outer joins cont'd

- **Right Outer Join**:

  This join keeps all the tuples in the right-hand relation in the result.

  Returns all the values from the right table and matched values from the right table. (Null in case of no matching join predicate)

- **Full Outer Join**:

  Combines the result of both left and right outer join . The joined table will contain records from both tables and fill in Nulls for missing matches on either tables.

Example

Attributes

Branch

Relation

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Cardinality

Degree

Primary key

Foreign key

Staff

Relation

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Division (R ÷ S)

- Defines a relation over the attributes C that consists of set of tuples from R *(with attribute set A)* that match combination of *every* tuple in S *(with attribute set B)* .
  Note: C is the set of attributes of R that are not attributes of S (A-B)

Example - Division

- Identify all clients who have viewed all properties with three rooms.

$$(\Pi_{clientNo,propertyNo}(Viewing)) \div (\Pi_{propertyNo}(\sigma_{rooms\ =\ 3}(PropertyForRent)))$$

$\Pi_{clientNo,propertyNo}(Viewing)$

clientNo	propertyNo
CR56	PA14
CR76	PG4
CR56	PG4
CR62	PA14
CR56	PG36

$\Pi_{propertyNo}(\sigma_{rooms=3}(PropertyForRent))$

propertyNo
PG4
PG36

RESULT

clientNo
CR56