Relational Algebra

- The basic set of operations for the relational model is the relational algebra.
  - enable the specification of basic retrievals
- The result of a retrieval is a new relation, which may have been formed from one or more relations.
  - algebra operations thus produce new relations, which can be further manipulated the same algebra.
- A sequence of relational algebra operations forms a relational algebra expression,
  - the result will also be a relation that represents the result of a database query (or retrieval request).

What is an Algebra?

- A language based on *operators* and a *domain of values*
- Operators map values taken from the domain into other domain values
- Hence, an expression involving operators and arguments produces a value in the domain
- When the domain is a set of all relations we get the *relational algebra*

Relational Algebra Definitions

- *Domain*: set of relations
- *Basic operators*: select, project, union, set difference, Cartesian (cross) product
- *Derived operators*: set intersection, division, join
- *Procedural*: Relational expression specifies query by describing an algorithm (the sequence in which operators are applied) for determining the result of an expression

Unary Relational Operations

- **SELECT Operation:** used to select a *subset* of the tuples from a relation that satisfy a **selection condition**. It is a filter that keeps only those tuples that satisfy a qualifying condition.

  Examples:

  $$\boldsymbol{\sigma_{\mathrm{DNO}\,=\,4}\,(\mathrm{EMPLOYEE})}$$

  $$\boldsymbol{\sigma_{\mathrm{SALARY}\,>\,30{,}000}\,(\mathrm{EMPLOYEE})}$$

  – denoted by $\sigma_{\text{<selection condition>}}(R)$ where the symbol $\boldsymbol{\sigma}$ (sigma) is used to denote the select operator, and the selection condition is a *Boolean expression* specified on the attributes of relation R

SELECT Operation Properties

The SELECT operation $\sigma_{\text{<selection condition>}}(R)$ produces a relation S that has the same schema as R

The SELECT operation σ is **commutative;** i.e.,

$$\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(R)) = \sigma_{\text{<condition2>}}(\sigma_{\text{< condition1>}}(R))$$

A cascaded SELECT operation **may be applied in any order;** i.e.,

$$\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(\sigma_{\text{<condition3>}}(R))$$
$$= \sigma_{\text{<condition2>}}(\sigma_{\text{< condition3>}}(\sigma_{\text{< condition1>}}(R)))$$

A cascaded SELECT operation may be replaced by a single selection with a conjunction of all the conditions**;** i.e.,

$$\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(\sigma_{\text{<condition3>}}(R))$$
$$= \sigma_{\text{<condition1> AND < condition2> AND < condition3>}}(R)))$$

Selection Condition

- Operators: $<, \leq, \geq, >, =, \neq$
- Simple selection condition:
  - <*attribute*> *operator* <*constant*>
  - <*attribute*> *operator* <*attribute*>
  - <*condition*> AND <*condition*>
  - <*condition*> OR <*condition*>
  - NOT <*condition*>

Select Examples

Person

Id	*Name*	*Address*	*Hobby*
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

$$\sigma_{Id>3000 \text{ OR } Hobby=\text{'hiking'}}(\text{Person})$$

$$\sigma_{Id>3000 \text{ AND } Id<3999}(\text{Person})$$

$$\sigma_{\text{NOT}(Hobby=\text{'hiking'})}(\text{Person})$$

$$\sigma_{Hobby\neq\text{'hiking'}}(\text{Person})$$

Unary Relational Operations (cont.)

- **PROJECT Operation:** selects certain *columns* from the table and discards the others.

  Example:

  $$\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$$

  The general form of the project operation is:

  $\pi_{\text{<attribute list>}}(R)$ where π is the symbol used to represent the project operation and <attribute list> is the desired list of attributes.

  PROJECT *removes duplicate tuples*, so the result is a set of tuples and hence a valid relation.

PROJECT Operation Properties

The number of tuples in the result of $\pi_{\text{<list>}}(R)$ is always less or equal to the number of tuples in R.

If attribute list includes a key of R, then the number of tuples is equal to the number of tuples in R.

$\pi_{\text{<list1>}}(\pi_{\text{<list2>}}(R)) = \pi_{\text{<list1>}}(R)$ as long as <list2> contains the attributes in <list1>

SELECT and PROJECT Operations

(a) $\sigma_{\text{(DNO=4 AND SALARY>25000) OR (DNO=5 AND SALARY>30000)}}\text{(EMPLOYEE)}$

(b) $\pi_{\text{LNAME, FNAME, SALARY}}\text{(EMPLOYEE)}$

(c) $\pi_{\text{SEX, SALARY}}\text{(EMPLOYEE)}$

(a)

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
Franklin	T	Wong	333445555	1955-12-08	638 Voss,Houston,TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 FireOak,Humble,TX	M	38000	333445555	5

(b)

LNAME	FNAME	SALARY
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

(c)

SEX	SALARY
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

Relational Algebra Operations from Set Theory

- The UNION, INTERSECTION, and MINUS Operations
- The CARTESIAN PRODUCT (or CROSS PRODUCT) Operation

Set Operators

- A relation is a *set* of tuples, so set operations apply:

  $\cap$, $\cup$, – (set difference)

- Result of combining two relations with a set operator is a relation => all elements are tuples with the same structure

UNION Operation

Denoted by $R \cup S$

Result is a relation that includes all tuples that are either in R or in S or in both. Duplicate tuples are eliminated.

Example: Retrieve the SSNs of all employees who either work in department 5 or directly supervise an employee who works in department 5:

DEP5_EMPS $\leftarrow \sigma_{\text{DNO=5}}$ **(EMPLOYEE)**

RESULT1 $\leftarrow \pi_{\text{SSN}}$**(DEP5_EMPS)**

RESULT2(SSN) $\leftarrow \pi_{\text{SUPERSSN}}$**(DEP5_EMPS)**

RESULT $\leftarrow$ **RESULT1** $\cup$ **RESULT2**

The union operation produces the tuples that are in either RESULT1 or RESULT2 or both. The two operands must be “type compatible”.

UNION Operation

Type (Union) Compatibility

The operand relations $R_1(A_1, A_2, ..., A_n)$ and $R_2(B_1, B_2, ..., B_n)$ must have the same number of attributes, and the domains of corresponding attributes must be compatible, i.e.

- $dom(A_i) = dom(B_i)$ for $i=1, 2, ..., n$.

Example

Tables:

Person (*SSN*, *Name*, *Address*, *Hobby*)

Professor (*Id*, *Name*, *Office*, *Phone*)

are <u>not</u> union compatible.

But

π_{Name} (Person) and π_{Name} (Professor)

<u>are</u> union compatible so

π_{Name} (Person) - π_{Name} (Professor)

makes sense.

UNION Example

STUDENT ∪ INSTRUCTOR:

STUDENT	FN	LN
	Susan	Yao
	Ramesh	Shah
	Johnny	Kohler
	Barbara	Jones
	Amy	Ford
	Jimmy	Wang
	Ernest	Gilbert

INSTRUCTOR	FNAME	LNAME
	John	Smith
	Ricardo	Browne
	Susan	Yao
	Francis	Johnson
	Ramesh	Shah

(b)

FN	LN
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

What would STUDENT ∩ INSTRUCTOR be?

Set Difference Operation

Set Difference (or MINUS) Operation

The result of this operation, denoted by R - S, is a relation that includes all tuples that are in R but not in S. The two operands must be "type compatible”.

Set Difference Example

S1

SID	SName	Age
473	Popeye	22
192	Jose	22
715	Alicia	28
914	Hal	24

S2

SID	SName	Age
202	Rusty	21
403	Marcia	20
914	Hal	24
192	Jose	22
881	Stimpy	19

Relational Algebra Operations From Set Theory (cont.)

- Union and intersection are *commutative operations:*

  $\mathbf{R \cup S = S \cup R}$**, and** $\mathbf{R \cap S = S \cap R}$

- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative operations;* that is

  $\mathbf{R \cup (S \cup T) = (R \cup S) \cup T}$**, and**

  $\mathbf{(R \cap S) \cap T = R \cap (S \cap T)}$

- The minus operation is *not commutative;* that is, in general

  $\mathbf{R - S \neq S - R}$

Cartesian (Cross) Product

- If *R* and *S* are two relations, $R \times S$ is the set of all concatenated tuples <*x*,*y*>, where *x* is a tuple in *R* and *y* is a tuple in *S*
  - *R* and *S* need not be union compatible
- $R \times S$ is expensive to compute:
  - Factor of two in the size of each row; Quadratic in the number of rows

R

A	B
x1	x2
x3	x4

S

C	D
y1	y2
y3	y4

$R \times S$

A	B	C	D
x1	x2	y1	y2
x1	x2	y3	y4
x3	x4	y1	y2
x3	x4	y3	y4

Cartesian Product Example

- We want a list of COMPANY’s female employees dependents.

FEMALE_EMPS	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle,Spring,TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
	Joyce	A	English	453453453	1972-07-31	5631 Rice,Houston,TX	F	25000	333445555	5

EMPNAMES	FNAME	LNAME	SSN
	Alicia	Zelaya	999887777
	Jennifer	Wallace	987654321
	Joyce	English	453453453

EMP_DEPENDENTS	FNAME	LNAME	SSN	ESSN	DEPENDENT_NAME	SEX	BDATE	• • •
	Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	• • •
	Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	• • •
	Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	• • •
	Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	• • •
	Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	• • •
	Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	• • •
	Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	• • •
	Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	• • •
	Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	• • •
	Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	• • •
	Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	• • •
	Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	• • •
	Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	• • •
	Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	• • •
	Joyce	English	453453453	333445555	Alice	F	1986-04-05	• • •
	Joyce	English	453453453	333445555	Theodore	M	1983-10-25	• • •
	Joyce	English	453453453	333445555	Joy	F	1958-05-03	• • •
	Joyce	English	453453453	987654321	Abner	M	1942-02-28	• • •
	Joyce	English	453453453	123456789	Michael	M	1988-01-04	• • •
	Joyce	English	453453453	123456789	Alice	F	1988-12-30	• • •
	Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	• • •

ACTUAL_DEPENDENTS	FNAME	LNAME	SSN	ESSN	DEPENDENT_NAME	SEX	BDATE
	Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28

RESULT	FNAME	LNAME	DEPENDENT_NAME
	Jennifer	Wallace	Abner

Binary Relational Operations: JOIN and DIVISION

- The JOIN Operation
- The EQUIJOIN and NATURAL JOIN variations of JOIN
- The DIVISION Operation

JOIN Operation

- *Cartesian product* followed by *select* is commonly used to identify and select related tuples from two relations => called **JOIN**. It is denoted by a $\bowtie$
  - This operation is important for any relational database with more than a single relation, because it allows us to process *relationships* among relations.
  - The general form of a join operation on two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ is:

    $R \bowtie_{<\text{join condition}>} S$

    where R and S can be any relations that result from general *relational algebra expressions*.

The Binary Join Operation

DEPT_MGR ←
DEPARTMENT $_{\textbf{MGRSSN=SSN}}$ EMPLOYEE

DEPT_MGR	DNAME	DNUMBER	MGRSSN	• • •	FNAME	MINIT	LNAME	SSN	• • •
	Research	5	333445555	• • •	Franklin	T	Wong	333445555	• • •
	Administration	4	987654321	• • •	Jennifer	S	Wallace	987654321	• • •
	Headquarters	1	888665555	• • •	James	E	Borg	888665555	• • •

EQUIJOIN & NATURAL JOIN

- **EQUIJOIN**
  - most common join: join conditions with equality comparisons only.
  - in the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have *identical values* in every tuple.
  - The JOIN in the previous example was EQUIJOIN.
- **NATURAL JOIN**
  - Because one of each pair of attributes with identical values is superfluous, a new operation called natural join—denoted by *—was created to get rid of the second (superfluous) attribute.
  - The standard definition of natural join requires that each pair of corresponding join attributes, have the **same name** in both relations. If this is not the case, a renaming operation is applied.

Natural Join Operations

(a) PROJ_DEPT ← PROJECT * DEPT

(b) DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS

(a)

PROJ_DEPT	PNAME	PNUMBER	PLOCATION	DNUM	DNAME	MGRSSN	MGRSTARTDATE
	ProductX	1	Bellaire	5	Research	333445555	1988-05-22
	ProductY	2	Sugarland	5	Research	333445555	1988-05-22
	ProductZ	3	Houston	5	Research	333445555	1988-05-22
	Computerization	10	Stafford	4	Administration	987654321	1995-01-01
	Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
	Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

(b)

DEPT_LOCS	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE	LOCATION
	Headquarters	1	888665555	1981-06-19	Houston
	Administration	4	987654321	1995-01-01	Stafford
	Research	5	333445555	1988-05-22	Bellaire
	Research	5	333445555	1988-05-22	Sugarland
	Research	5	333445555	1988-05-22	Houston

The DIVISION Operation

(a) Dividing SSN_PNOS by SMITH_PNOS.
(b) $T \leftarrow R \div S$.

(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

Additional Relational Operations

- Aggregate Functions and Grouping
- Recursive Closure Operations
- The OUTER JOIN Operation

Aggregate Functions

E.g. SUM, AVERAGE, MAX, MIN, COUNT

(a)

R	DNO	NO_OF_EMPLOYEES	AVERAGE_SAL
	5	4	33250
	4	3	31000
	1	1	55000

(b)

DNO	COUNT_SSN	AVERAGE_SALARY
5	4	33250
4	3	31000
1	1	55000

(c)

COUNT_SSN	AVERAGE_SALARY
8	35125

Recursive Closure Example

(Borg's SSN is 888665555)

SUPERVISION	SSN1 (SSN)	SSN2 (SUPERSSN)
	123456789	333445555
	333445555	888665555
	999887777	987654321
	987654321	888665555
	666884444	333445555
	453453453	333445555
	987987987	987654321

RESULT 1	SSN
	333445555
	987654321

(Supervised by Borg)

RESULT 2	SSN
	123456789
	999887777
	666884444
	453453453
	987987987

(Supervised by Borg's subordinates)

RESULT	SSN
	123456789
	999887777
	666884444
	453453453
	987987987
	333445555
	987654321

(RESULT1 $\cup$ RESULT2)

OUTER JOINs

- In NATURAL JOIN tuples without a *matching* (or *related*) tuple are eliminated from the join result. Tuples with null in the join attributes are also eliminated. This loses information.

- *Outer joins*, can be used when we want to keep all the tuples in R, all those in S, or all those in both relations
  - regardless of whether they have matching tuples in the other relation.

- The left outer join operation keeps every tuple in the *first* or *left* relation R in $R ⟕ S$; if no matching tuple is found in S, then the attributes of S in the join result are "padded" with null values.

- A similar operation, right outer join, keeps every tuple in the *second* or *right* relation S in the result of $R ⟖ S$.

- A third operation, *full outer join*, denoted by $⟗$ keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with null values as needed.

Left Outer Join

E.g. List all employees and the department they manage, *if* they manage a department.

RESULT	FNAME	MINIT	LNAME	DNAME
	John	B	Smith	null
	Franklin	T	Wong	Research
	Alicia	J	Zelaya	null
	Jennifer	S	Wallace	Administration
	Ramesh	K	Narayan	null
	Joyce	A	English	null
	Ahmad	V	Jabbar	null
	James	E	Borg	Headquarters