Week 2

Saturday, 25 April 2020 3:17 PM

- Relational Algebra
- SQL

Relational Algebra

•Relational Algebra is a procedural DML.
•It specifies operations on relations to define new relations:
Select, Project, Union, Intersection, Difference, Cartesian Product, Join, Divide.

SELECT

•Selects a subset of the tuples of a relation r, satisfying some condition.

$$\sigma_B(r) = \{t \in r : B(t)\}$$

•B is the selection condition, composed of selection clauses combined using AND, OR and NOT.
Example: Select the enrolment records for person 1's non-Ph.D. students:

$$\sigma_{(Supervisor=1)\text{AND NOT}(Name\neq\text{"}PH.D\text{"})}(ENROLMENT)$$

PROJECT

•Projects onto a subset X of the attributes of a relation.

$$\pi_X(r) = \{t[X] : t \in r\}$$

•Remember that a tuple, t is a mapping from attributes to elements of their domains. t[X] is the restriction of that mapping to the set of attributes X.

Example:

R = (Animal, Cat), S = (Animal, Dog)

π: project on the first column

$\pi\ (R \cap S)) = \{\}$

$\pi\ (R) \cap \pi\ (S) = \{\text{Animal}\}$

UNION

•
Is the set theoretic union of the tuples of two relations.

$$r \cup s = \{t : t \in r \text{ or } t \in s\}$$

INTERSECTION

•
Is the set theoretic intersection of the tuples of two relations.

$$r \cap s = \{t : t \in r \text{ and } t \in s\}.$$

DIFFERENCE

•
Is the set difference of the tuples of two relations.

$$r - s = \{t : t \in r \text{ and } t \notin s\}$$

CARTESIAN PRODUCT

$$r \times s = \{t_1 || t_2 : t_1 \in r \text{ and } t_2 \in s\}$$

Where $t_1 || t_2$ indicates the concatenation of tuples.

Example:

ENROLMENT × *RESEARCHER*

E'ment#	S'ee	S'or	D'ment	E'ment. Name	Person#	R'cher. Name
1	1	2	Psych.	Ph.D.	1	Dr C.C. Chen
1	1	2	Psych.	Ph.D.	2	Dr R.G.Wilkinson
2	3	1	Comp.Sci	Ph.D.	1	Dr C.C. Chen
2	3	1	Comp.Sci	Ph.D.	2	Dr R.G.Wilkinson
3	4	1	Comp.Sci	M.Sc.	1	Dr C.C. Chen
3	4	1	Comp.Sci	M.Sc.	2	Dr R.G.Wilkinson
4	5	1	Comp.Sci	M.Sc.	1	Dr C.C. Chen
4	5	1	Comp.Sci	M.Sc.	2	Dr R.G.Wilkinson

More useful is:

R1 ← *ENROLMENT* × *RESEARCHER*

$\sigma_{(Supervisor=Person\#)}(R1) =$

E'ment#	S'ee	S'or	D'ment	E'ment. Name	Person#	R'cher. Name
1	1	2	Psych.	Ph.D.	2	Dr R.G.Wilkinson
2	3	1	Comp.Sci.	Ph.D.	1	Dr C.C. Chen
3	4	1	Comp.Sci.	M.Sc.	1	Dr C.C. Chen
4	5	1	Comp.Sci.	M.Sc.	1	Dr C.C. Chen

- or even better:

  R1 ← *ENROLMENT* × *RESEARCHER*

  R2 ← *σ(Supervisor=Person#)(R1)* =

  π{E'ment#,S'ee,S'or,R'cher.Name,D'ment,E'ment.Name}(R2) =

E'ment#	S'ee	S'or	R'cher. Name	D'ment	E'ment. Name
1	1	2	Dr R.G.Wilkinson	Psych.	Ph.D.
2	3	1	Dr C.C. Chen	Comp.Sci.	Ph.D.
3	4	1	Dr C.C. Chen	Comp.Sci.	M.Sc.
4	5	1	Dr C.C. Chen	Comp.Sci.	M.Sc.

- The last of these is also known as natural join, the next to last is equi-join.

Example:

$$ENROLMENT \underset{RESEARCHER}{\bowtie} (Supervisor=Person\#)$$

- 3.7.1 Theta-join

  $r \bowtie_B s = \{t_1 || t_2 : t_1 \in r \text{ and } t_2 \in s \text{ and } B\}$

- 3.7.3 Natural join

Is an equi-join where only one attribute from each comparison is retained.

Example: $ENROLMENT \underset{RESEARCHER}{\bowtie} (Supervisor),(Person\#)$

DIVIDE

Example:

P	
A	B
a_1	b_1
a_1	b_2
a_2	b_1
a_3	b_2
a_4	b_1
a_5	b_1
a_5	b_2

Q
B
b_1
b_2

$P \div Q =$

A
a_1
a_5

For a2, we don't have a2 b2,so its not the result. Same for a3 and a4.

SQL = Structured Query Language (pronounced “sequel”).
An ANSI/ISO standard language for querying and manipulating relational DBMSs.
Developed at IBM (San Jose Lab) during the 1970’s, and standardised during the 1980’s.
Appears that SQL will survive the rise of object
relational database systems.
Designed to be a “human readable” language supporting:
◦relational algebra operations
◦aggregation operations

Query syntax is:
SELECT attributes
FROM relations
WHERE condition

All attributes in SQL relations have domain specified.
SQL supports a small set of useful built
in data types: strings, numbers,
dates, bit strings.
Self defined data type is allowed in PostgreSQL.

Two kinds of string are available:
◦CHAR(n) ... uses n bytes, left justified, blank padded
◦VARCHAR(n) ... uses 0..n bytes, no padding

Two kinds of pattern-matching:

- % matches anything (like *)
- _ matches any single char (like .)

Examples:

- Name LIKE ‘Ja%’ — Name begins with ‘Ja’
- Name LIKE ‘_i%’ — Name has ‘i’ as 2nd letter
- Name LIKE ‘%o%o%’ — Name contains two ‘o’s

(start1, end1) OVERLAPS (start2, end2)

- This expression yields true when two time periods (defined by their endpoints) overlap, false when they do not overlap.
- SELECT (DATE '2001-02-16', DATE '2001-12-21') OVERLAPS (DATE '2001-10-30', DATE '2002-10-30'); -> *Result:* true

- AVG(*attr*) ... mean of values for *attr*
- COUNT(*attr*) ... number of rows in *attr* column
- MIN/MAX(*attr*) ... min/max of values for *attr*
- SUM(*attr*) ... sum of values for *attr*

Note: NULL value produces NULL result for arithmetic operation, but NULL is ignored in column operations.

Querying a Single Relation
Formal semantics (relational algebra):
◦start with relation R in FROM clause
◦apply σ using Condition in WHERE clause
◦apply π using Attributes in SELECT clause
SELECT Attributes
FROM R
WHERE Conditions

Names of drinkers: $\pi_{Name}(Drinkers)$

- SELECT Name FROM Drinkers;

$\sigma_{Cond}(Rel)$ is implemented in SQL as:

SELECT * FROM Rel WHERE Cond

```
SELECT a1, a2, a3
FROM Rel
WHERE Cond
```