Week2Lecture02 Relational Algebra

What is relational algebra

English <-> Relational Algebra <->SQL queries

Relations: mathematical definition

A relation R of degree n, where values come from domains A1, ..., An:

$$R \subseteq A1 \times A2 \times \ldots \times An$$

subset of the **Cartesian product of domains**

Unary Operators

example

University		
uName	County	Enrollment
NOTT	Nott/shire	18000
CAM	Cam/shire	22000
UCL	Great/Lon	20000

Student			
SID	sName	GPA	HS
0135	John	18.5	100
0025	Mary	19.3	1000
0423	Mary	17.5	300

Apply			
SID	uName	Subj	Dec
0135	CAM	CS	'A'
0135	NOTT	CS	'A'
0423	NOTT	ENG	'R'

Selection

- σ: selection operator
- α: properties
- R: target relation

σ_$\alpha(R) = \{(a1, \ldots, an) \mid (a1, \ldots, an) \in R, \alpha(a1, \ldots, an)\}$

example

- 1. Find out all students with GPA more than 19.

  $$\sigma_{GPT\ >\ 19}(Student)$$

- 2. Find out all students with GPA more than 19 and high school size less than 1000.

  $$\sigma_{GPT\ >\ 19\ and\ HS\ <\ 1000}(Student)$$

- 3. Find out all applications to University of Nottingham with subject CS

  $$\sigma_{uName\ =\ 'Notts'\ and\ Subj\ =\ 'CS'}(Apply)$$
  $$\sigma_{Subj\ =\ 'CS'}(\sigma_{uName\ =\ 'Notts'}(Apply))$$

```
select * from Student where GPA>19
```

```sql
select * from Student where GPA>19 and HS<1000
```

```sql
select * from Apply where uName='Notts' and Subj='CS'
```

```sql
select * from (select * from Apply where uName='Notts') where Subj='CS'
```

Projection

Projection works as slicing

• Projection of R on X is represented as:

$$\pi_X(R)$$

• π: projection operator
• X: a set of attributes
• R: target relation
• $\pi X\ R$ is a new relation only contain attributes from X

example

- 1. Get IDs and decisions from all applications.

$$\pi_{SID,\ Dec}(Apply)$$

- 2. Get IDs and names of students with GPA greater than 19.

$$\pi_{SID,\ sName}(\sigma_{GPA\ >\ 19}(Student))$$

```sql
select SID,Dec from Apply
```

```sql
select SID,sName from (select * from Student where GPA>19)
```

Set Operators

Union

- Standard set-theoretic definition:

$$A \cup B = \{x \mid x \in A\ or\ x \in B\}$$

- E.g., $\{a, b, c\} \cup \{a, d, e\} = \{a, b, c, d, e\}$

Set difference

- Standard set-theoretic definition:

$$A - B = \{x \mid x \in A \; or \; x \notin B\}$$

- E.g., $\{a, b, c\} - \{a, d, e\} = \{b, c\}$
- Partial Operation: requires union-compatible

Intersection

- Standard set-theoretic definition:

$$A \cap B = \{x \mid x \in A \; and \; x \in B\}$$

- E.g., $\{a, b, c\} \cap \{a, d, e\} = \{a\}$
- Partial Operation: requires union-compatible

Cartesian product

- Standard set-theoretic definition:

$$A \times B = \{(x, y) \mid x \in A, y \in B\}$$

- E.g., $\{a, b\} \times \{d, e\} = \{(a, d), (a, e), (b, d), (b, e)\}$
- Total Operation
- Extended Cartesian product:

$$A \times B = \{(c_1, \dots, cn, d_1, \dots, dm) \mid (c_1, \dots, cn) \in A, (d_1, \dots, dm) \in B\}$$

Joint Operators

Natural Join Operator

Same as Cartesian Product but enforces equality on all attributes with the same name (S.SID and A.SID in our case)

- **Student ⋈ Apply** (bowtie)
  - Same as Cartesian Product but enforces equality on all attributes with the same name (S.SID and A.SID in our case)
  - Automatically sets values equal when attribute names are the same
  - Gets rid of multiple copies of the attributes with the same name (there will be only one common SID attribute in the result)

Student ⋈ Apply

SID	sName	GPA	HS	uName	Subj	Dec
0135	John	18.5	100	CAM	CS	'A'
0135	John	18.5	100	NOTT	CS	'A'
0423	Mary	17.5	300	NOTT	ENG	'R'

example

- E.g. 1 "Names and GPAs of students with HS>1000 who applied to CS and were rejected"
  - $\pi_{GPA,sName}(\sigma_{HS>1000 \text{ and subj='CS' and dec='Rej'}}$ **(Student ⋈ Apply)**)

- E.g. 2 "Names and GPAs of students with HS>1000 who applied to CS at Universities with enrolment > 20000 and were rejected"
  - $\pi_{GPA,sName}(\sigma_{HS>1000 \text{ and subj='CS' and Enrollment>20000 and dec='Rej'}}$ **(Student ⋈ (Apply ⋈ Uni)**))

if two graph do not have the common attribute, Natural joint will have the same as the Cartesian joint

Theta Join Operator

- Cartesian Product satisfy certain properties
- Can be implemented via Cartesian Product and Select.
- The Theta Join operator is defined as
  - $R \bowtie_\theta S = \sigma_\theta (R \times S)$
- The result of this operation consists of all combinations of tuples in R and S that satisfy property θ

Rename Operation

- The rename operator has 3 forms. The first one is the most general
  - ρR(A1, A2, …An)(E).
    - This should be read as: "Evaluate E, and get a relation as a result. Then call the result relation R with attributes A1,…,An." From now on, we can use this schema to describe the result of E.
  - ρR(E).
    - "Use the same attribute names but change the relation name to
  - ρ(A1, A2, …An)(E).
    - "Use the same relation name but change the attribute names to A1,…,An."