Chapter 6. Formal Relational Query Languages

6.1 The Relational Algebra

Procedural query language.

The operators take as input one or two relations and produce a new relation as a result.

6.1.1 Fundamental Operations

There are 6 primitive operators

- **Select (σ)**

$$\sigma_p(r) = \{ t \mid t \in r \text{ and } p(t) \}$$

Select rows satisfying the given condition(selection predicate p) -> **Tuple Selection**

$$\sigma_{\text{dept\_name}=\text{"}Comp.\ Sci.\text{"}}(instructor)$$

= Find tuples of *instructor* relation where the instructor is in the "Comp. Sci." department

- **Project (∏)**

$$\prod_{A1,A2,\ldots,Ak}(r)$$

Show k columns obtained by erasing the columns that are not listed. -> **Attribute Selection**

Duplicate rows are removed from the result, since relations are sets.

cf) SQL is based on multiset theory, so the result includes duplicate rows, so we need to delete those duplications explicitly.

$$\prod_{ID,\ name,\ salary}(instructor)$$

= Show *ID*, *name*, and *salary* column in the *instructor* relation

- **Union (∪)**

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

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**

$$\Pi_{course\_id}(\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009}(selection)) \cup$$

$$\Pi_{course\_id}(\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010}(selection))$$

= find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

- **Set difference (-)**

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

For $r - s$ to be valid:

1. r and s must have the **same arity**
2. The attribute domains must be **compatible**

$$\Pi_{course\_id}(\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009}(selection)) -$$

$$\Pi_{course\_id}(\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010}(selection))$$

- **Cartesian product (x)**

$$r \times s = \{ t\,q \mid t \in r \text{ and } q \in s \}$$

Assume that attributes of r(R) and s(S) are disjoint($R \cap S = \emptyset$).

If attributes of r(R) and s(S) are not disjoint, then rename must be used(usually *relation.attribute*).

As Cartesian Product operator makes all possible pairs, redundant tuples can be made. To get rid of those meaningless tuples, select more giving a specific condition.

$$\sigma_{instructor.ID=teaches.ID}(\sigma_{dept\_name=\text{"Physics"}}(instructor \times teaches))$$

- **Rename (ρ)**

$$\rho_{x(A1,A2,\ldots,An)}(E)$$

returns the result of expression E under the name X, and with the attributes renamed to A_1, A_2, …, A_n

Instead renaming, we can use positional natation for attributes, \$1, \$2, …

6.1.3 Additional Relational-Algebra Operations

- **Set-Intersection (∩)**

$$- \quad r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$$

same as ***r* - (*r* - *s*)**

r, s have the same arity, attributes of r and s are compatible

- **Natural-Join (⋈)**

$$r \bowtie s = \prod_{R \cup S} (\sigma_{r.A1=s.A1 \wedge r.A2=s.A2 \wedge \ldots \wedge r.An=s.An} (r \times s))$$

$$\text{where } R \cap S = \{A_1, A_2, \ldots, A_n\}$$

Among all possible tuples of Cartesian product of the two relations, If there exist the same attributes, pick those only have the same attribute values.

Therefore, if r ∩ s = Ø, then the result should be r x s

cf) **theta(θ) Join: $r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$**

- **Assignment (←)**

It is like assignment in a programming language.

Assignment must always be made to a temporary relation variable

- **Outer-Join**

There may be lost tuples in the result of the Join operation. Outer Join preserves those tuples that would be lost in an join by creating tuples in the result containing null values.

Left Outer Join: Show all tuples in the left side of the operator

Right Outer Join: Show all tuples in the right side of the operator

Full outer Join: Show every tuples

Outer-Join operations can be expressed using the basic operations

ex) Left Outer Join: $(r \bowtie s) \cup (r - \Pi_R(r \bowtie s)) \times \{\{null, \ldots, null\}\}$,
where {(null, ...,null)} is on the schema S-R

6.1.4 Extended Relational-Algebra Operations

- **Generalized Projection**

$$\Pi_{F1,F2,\ldots,Fn}(E)$$

allows arithmetic and string functions to be used in the projection list.

- **Aggregation**

$${}_{G1,G2,\ldots,Gn}\mathcal{G}_{F1(A1),F2(A2),\ldots,Fn(An)}(E)$$

E is any relational-algebra expressions

$G_1, G_2, \ldots, G_n$ is a list of attributes on which to group (can be empty)

Each F_i is an aggregate function

Each A_i is an attribute name

permits the use of aggregate functions, which take a collection of values(multisets) and returns a single value as a result.

If we want to eliminate duplicates in the aggregate function, add “-distinct” after the function name.

ex) Find the total number of instructors who teach a course in the Spring 2010 semester

$$\mathcal{G}_{\text{count-distinct}(ID)}(\sigma_{semester=\text{"Spring"} \wedge year=2010}(teaches))$$