Chapter 6. Formal Relational Query Languages

6.1 The Relational Algebra

<u>Remark</u>. The relational algebra is a procedural query language.

It consists of a set of operations that take one or two relations as input and produce a new relation as output.

<u>Def</u>. A basic expression in the relational algebra is either

i) A relation in the database or ii) A constant relation.

"constant values" ex) {(3, 'a'), (4, 'b')}.

Let E_1, E_2 is a relational-algebra expressions.

Then, the following are all relational-algebra expressions.

i) $E_1 \cup E_2$: Union (E_1 UNION E_2)

ii) $E_1 - E_2$: Set difference (E_1 EXCEPT E_2)

iii) $E_1 \times E_2$: Cartesian product (FROM E_1, E_2)

iv) $\sigma_p(E_1)$: Select (WHERE p)
  ↳ predicate on attributes of E_1.

v) $\Pi_s(E_1)$: Projection (SELECT s)
  ↳ list consisting of some of the attributes of E_1

vi) $\rho_x(E_1)$: Rename (AS x)
  ↳ new name for the result of E_1

* Additional relational-algebra operations

i) Set intersection : $E_1 \cap E_2 = E_1 - (E_1 - E_2)$.

ii) Natural join : $E_1 \bowtie E_2 = \Pi_{a_1, a_2, \cdots, a_n} (\sigma_{E_1.a_1 = E_2.a_1 \wedge \cdots \wedge E_1.a_n = E_2.a_n} (E_1 \times E_2))$

iii) Outer joins

- Left outer join : $E_1 ⟕ E_2 = E_1 \bowtie E_2 \cup (E_1 - \Pi(E_1 \bowtie E_2)) \times \{null, null, \ldots, null\}$.
  ↳ Schema $E_2 - E_1$
- Right outer join : $E_1 ⟖ E_2 = E_1 \bowtie E_2 \cup (E_2 - \Pi(E_1 \bowtie E_2)) \times \{null, null, \cdots, null\}$
  ↳ Schema $E_1 - E_2$.
- Full outer join : $E_1 ⟗ E_2 = (E_1 ⟕ E_2) \cup (E_1 ⟖ E_2)$.

* Extended relational-algebra operations

i) Generalized projection : $\Pi_{F_1, F_2, \cdots, F_n}(E)$ ex) $\Pi_{a_1 \times 3 \;\; a_2 + 2 \; \cdots \; a_n \div 2}(E)$.

where $F_1, F_2, \cdots, F_n$: arithmetic expression involving constants.

ii) Aggregation : $_{G_1, G_2, \cdots, G_n} \mathcal{G}_{F_1(a_1) \, F_2(a_2) \cdots F_m(a_m)} (E)$ ex) $_{a_1, a_3} \mathcal{G}_{avg(a_2) \;\; count(a_5)} (E)$.

where $G_1, G_2, \cdots, G_n$: list of attributes on which to group

$F_1, F_2, \cdots, F_m$: aggregation function

<u>Remark</u> Aggregation functions can operate on relations which allows multiple occurences of a same tuple.

Those relations are called multisets.

<u>Def.</u> Multiset relational algebra.

Let's say there is C_1 copies of tuple t_1 in r_1, C_2 of t_2 in r_2.

i) If t_1 satisfies selection σ_θ, there are C_1 copies of t_1 in $\sigma_\theta(t_1)$

ii) For each copy of t_1, there is $\Pi_A(t_1)$ in $\Pi_A(r_1)$.

iii) There are $C_1 C_2$ copies of the tuple $t_1 t_2$ in $r_1 \times r_2$.

6.2 The Tuple Relational Calculus.

Remark. The tuple relational calculus is nonprocedural,

i.e. it describes the desired info without procedures.

ex) $\{t \mid t \in r_1 \wedge t[a] > 10\}$.

Def. A tuple-relational-calculus formula is built up out of *atoms*. An atom has one of the following forms:

- $s \in r$, where s is a tuple variable and r is a relation (we do not allow use of the $\notin$ operator).
- $s[x]\ \Theta\ u[y]$, where s and u are tuple variables, x is an attribute on which s is defined, y is an attribute on which u is defined, and Θ is a comparison operator ($<, \leq, =, \neq, >, \geq$); we require that attributes x and y have domains whose members can be compared by Θ.
- $s[x]\ \Theta\ c$, where s is a tuple variable, x is an attribute on which s is defined, Θ is a comparison operator, and c is a constant in the domain of attribute x.

We build up formulae from atoms by using the following rules:

- An atom is a formula.
- If P_1 is a formula, then so are $\neg P_1$ and (P_1).
- If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$, and $P_1 \Rightarrow P_2$.
- If $P_1(s)$ is a formula containing a free tuple variable s, and r is a relation, then

$$\exists\, s \in r\ (P_1(s)) \text{ and } \forall\, s \in r\ (P_1(s))$$

are also formulae.

Remark. Redundant rules.

i) $P_1 \wedge P_2 = \neg((\neg P_1) \vee (\neg P_2))$

ii) $\forall t \in r(P_1(t)) = \neg \exists t \in r(\neg P_1(t))$

iii) $P_1 \Rightarrow P_2 = \neg P_1 \vee P_2$.

Def. A tuple variable is free until it is quantified by $\exists$ or $\forall$.
If it is quantified, it is bound.

Def. Domain of formula P, $dom(P)$, is the set of all the values referenced by P.

Def. We say expression $\{t \mid P(t)\}$ is safe if $\{t \mid P(t)\} \subset dom(P)$.

Example. $\{t \mid \neg(t \in r)\}$ contains infinite tuples $t \notin dom(P) = r$.
So the expression is unsafe.

6.3 The Domain Relational Calculus.

<u>Example</u>. Find all the tuple in r such that $a_4 > 100$.

Tuple relational calculus : $\{t \mid t \in r \wedge t[a_4] > 100\}$

Domain relational calculus : $\{\underbrace{\langle x_1, x_2, \cdots x_n \rangle}_{\text{domain variables.}} \mid \langle x_1, x_2, \cdots, x_n \rangle \in r \wedge x_4 > 100\}$.

<u>Remark</u>. All three of the following are equivalent.

- The basic relational algebra
  (without the extended operations on multisets)
- The tuple relational calculus restricted to safe expressions
- The domain relational calculus restricted to safe expressions