Chapter 6: Formal Relational Query Languages
Outline

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus
Relational Algebra

- Procedural language
- Six basic operators
  - select: \( \sigma \)
  - project: \( \Pi \)
  - union: \( \cup \)
  - set difference: \( - \)
  - Cartesian product: \( \times \)
  - rename: \( \rho \)
- The operators take one or two relations as inputs and produce a new relation as a result.
Select Operation

- Notation: $\sigma_p(r)$
- $p$ is called the selection predicate
- Defined as:

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

Where $p$ is a formula in propositional calculus consisting of terms
connected by: $\land$ (and), $\lor$ (or), $\neg$ (not)
Each term is one of:

- <attribute> op <attribute> or <constant>

where op is one of: $=$, $\neq$, $>$, $\geq$, $<$, $\leq$

- Example of selection:

$$\sigma_{\text{dept\_name}="Physics"}(\text{instructor})$$
Project Operation

- Notation:
  \[ \Pi_{A_1, A_2, \ldots, A_k}(r) \]
  where \( A_1, A_2 \) are attribute names and \( r \) is a relation name.

- The result is defined as the relation of \( k \) columns obtained by erasing the columns that are not listed.

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

- Example: To eliminate the \textit{dept\_name} attribute of \textit{instructor}
  \[ \Pi_{ID, name, salary}(\textit{instructor}) \]
Union Operation

- Notation: \( r \cup s \)
- Defined as:
  \[
  r \cup s = \{ t \mid t \in r \text{ or } t \in s \}
  \]
- 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** (example: 2\(^{nd}\) column of \( r \) deals with the same type of values as does the 2\(^{nd}\) column of \( s \))
- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both
  \[
  \Pi_{course \_id} (\sigma \text{ semester="Fall" } \Lambda \text{ year=2009}(section)) \cup \\
  \Pi_{course \_id} (\sigma \text{ semester="Spring" } \Lambda \text{ year=2010}(section))
  \]
Set Difference Operation

- Notation \( r - s \)
- Defined as:
  \[
  r - s = \{ t | t \in r \text{ and } t \notin s \}
  \]

- Set differences must be taken between compatible relations.
  - \( r \) and \( s \) must have the same arity
  - attribute domains of \( r \) and \( s \) must be compatible

- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

\[
\Pi_{course_id} (\sigma_{semester=\text{“Fall”} \land year=2009} (section)) - \\
\Pi_{course_id} (\sigma_{semester=\text{“Spring”} \land year=2010} (section))
\]
Set-Intersection Operation

- Notation: \( r \cap s \)
- Defined as:
  \[
  r \cap s = \{ t \mid t \in r \text{ and } t \in s \}
  \]
- Assume:
  - \( r, s \) have the same arity
  - attributes of \( r \) and \( s \) are compatible
- Note: \( r \cap s = r - (r - s) \)
Cartesian-Product Operation

- Notation $r \times s$
- Defined as:
  
  $$r \times s = \{ t q \mid t \in r \text{ and } q \in s \}$$

- Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.
Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

  \[ \rho_x(E) \]

  returns the expression \( E \) under the name \( X \)

- If a relational-algebra expression \( E \) has arity \( n \), then

  \[ \rho_x(A_1, A_2, \ldots, A_n)(E) \]

  returns the result of expression \( E \) under the name \( X \), and with the attributes renamed to \( A_1, A_2, \ldots, A_n \).
Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let $E_1$ and $E_2$ be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p(E_1)$, $P$ is a predicate on attributes in $E_1$
  - $\Pi_S(E_1)$, $S$ is a list consisting of some of the attributes in $E_1$
  - $\rho_x(E_1)$, $x$ is the new name for the result of $E_1$
Tuple Relational Calculus
Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form
  \[ \{ t \mid P(t) \} \]
- It is the set of all tuples \( t \) such that predicate \( P \) is true for \( t \)
- \( t \) is a tuple variable, \( t[A] \) denotes the value of tuple \( t \) on attribute \( A \)
- \( t \in r \) denotes that tuple \( t \) is in relation \( r \)
- \( P \) is a formula similar to that of the predicate calculus
Predicate Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., <, ≤, =, ≠, >, ≥)
3. Set of connectives: and (∧), or (v), not (¬)
4. Implication (⇒): $x \Rightarrow y$, if $x$ if true, then $y$ is true
   \[ x \Rightarrow y \equiv \neg x \lor y \]
5. Set of quantifiers:
   - $\exists t \in r (Q(t)) \equiv "there \ exists" \ a \ tuple \ in \ t \ in \ relation \ r$
     such that predicate $Q(t)$ is true
   - $\forall t \in r (Q(t)) \equiv Q \ is \ true \ "for \ all" \ tuples \ t \ in \ relation \ r$
Example Queries

- Find the ID, name, dept_name, salary for instructors whose salary is greater than $80,000

\[ \{ t \mid t \in instructor \land t[salary] > 80000 \} \]

Notice that a relation on schema (ID, name, dept_name, salary) is implicitly defined by the query.

- As in the previous query, but output only the ID attribute value

\[ \{ t \mid \exists s \in instructor (t[ID] = s[ID] \land s[salary] > 80000) \} \]

Notice that a relation on schema (ID) is implicitly defined by the query.
Example Queries

- Find the names of all instructors whose department is in the Watson building

\[
\{ t \mid \exists s \in instructor ( t [\text{name}] = s [\text{name}] \\
\quad \wedge \exists u \in department ( u [\text{dept\_name}] = s[\text{dept\_name}] \\
\quad \quad \wedge u [\text{building}] = \text{"Watson"}) )\}
\]

- Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

\[
\{ t \mid \exists s \in section ( t [\text{course\_id}] = s [\text{course\_id}] \\
\quad s [\text{semester}] = \text{"Fall"} \wedge s [\text{year}] = 2009 \\
\quad \lor \exists u \in section ( t [\text{course\_id}] = u [\text{course\_id}] \\
\quad \quad u [\text{semester}] = \text{"Spring"} \wedge u [\text{year}] = 2010 )\}
\]
Example Queries

- Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

\[ \{ t | \exists s \in \text{section} ( t [\text{course_id}] = s [\text{course_id}] \land \\
    s [\text{semester}] = \text{"Fall"} \land s [\text{year}] = 2009 \\
    \land \exists u \in \text{section} ( t [\text{course_id}] = u [\text{course_id}] \land \\
    u [\text{semester}] = \text{"Spring"} \land u [\text{year}] = 2010 ) \} \]

- Find the set of all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

\[ \{ t | \exists s \in \text{section} ( t [\text{course_id}] = s [\text{course_id}] \land \\
    s [\text{semester}] = \text{"Fall"} \land s [\text{year}] = 2009 \\
    \land \neg \exists u \in \text{section} ( t [\text{course_id}] = u [\text{course_id}] \land \\
    u [\text{semester}] = \text{"Spring"} \land u [\text{year}] = 2010 ) \} \]
Universal Quantification

- Find all students who have taken all courses offered in the Biology department

\[
\{ t \mid \exists r \in \text{student} \ (t \ [ID] = r \ [ID]) \land \\
(\forall u \in \text{course} \ (u \ [\text{dept}_\text{name}] = \text{"Biology"}) \Rightarrow \\
\exists s \in \text{takes} \ (t \ [ID] = s \ [ID] \land \\
\ s \ [\text{course}_\text{id}] = u \ [\text{course}_\text{id}])\}
\]
Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations.
- For example, \{ t \mid \neg t \in r \} results in an infinite relation if the domain of any attribute of relation \( r \) is infinite.
- To guard against the problem, we restrict the set of allowable expressions to safe expressions.
- An expression \{t \mid P(t)\} in the tuple relational calculus is safe if every component of \( t \) appears in one of the relations, tuples, or constants that appear in \( P \).
  
  - NOTE: this is more than just a syntax condition.
    
    - E.g. \{ t \mid t[A] = 5 \lor \text{true} \} is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in \( P \).
Consider again that query to find all students who have taken all courses offered in the Biology department

\[
\{ t \mid \exists r \in \text{student} \ (t [ID] = r [ID]) \land \\
(\forall u \in \text{course} \ (u [\text{dept}_\text{name}] = \text{"Biology"}) \Rightarrow \\
\exists s \in \text{takes} \ (t [ID] = s [ID] \land \\
s [\text{course}_\text{id}] = u [\text{course}_\text{id}])\}
\]

Without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.
Domain Relational Calculus
Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus
- Each query is an expression of the form:

  \[ \{ < x_1, x_2, ..., x_n > \mid P (x_1, x_2, ..., x_n) \} \]

  - \( x_1, x_2, ..., x_n \) represent domain variables
  - \( P \) represents a formula similar to that of the predicate calculus
Example Queries

- Find the ID, name, dept_name, salary for instructors whose salary is greater than $80,000
  \[
  \{< i, n, d, s> | < i, n, d, s> \in instructor \land s > 80000\}\]

- As in the previous query, but output only the ID attribute value
  \[
  \{< i> | < i, n, d, s> \in instructor \land s > 80000\}\]

- Find the names of all instructors whose department is in the Watson building
  \[
  \{< n > | \exists i, d, s (< i, n, d, s > \in instructor \\
  \land \exists b, a (< d, b, a> \in department \land b = “Watson” ))\}\}
Example Queries

Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

\[ \{<c> | \exists a, s, y, b, r, t ( <c, a, s, y, b, r, t > \in section \land
\quad s = \text{“Fall”} \land y = 2009 ) \lor
\exists a, s, y, b, r, t ( <c, a, s, y, b, r, t > \in section ] \land
\quad s = \text{“Spring”} \land y = 2010)\} \]

This case can also be written as

\[ \{<c> | \exists a, s, y, b, r, t ( <c, a, s, y, b, r, t > \in section \land
\quad ( s = \text{“Fall”} \land y = 2009 ) \lor ( s = \text{“Spring”} \land y = 2010) )\} \]

Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

\[ \{<c> | \exists a, s, y, b, r, t ( <c, a, s, y, b, r, t > \in section \land
\quad s = \text{“Fall”} \land y = 2009 ) \land
\exists a, s, y, b, r, t ( <c, a, s, y, b, r, t > \in section ] \land
\quad s = \text{“Spring”} \land y = 2010)\} \]
Safety of Expressions

The expression:

\[ \{ < x_1, x_2, \ldots, x_n > \mid P (x_1, x_2, \ldots, x_n) \} \]

is safe if all of the following hold:

1. All values that appear in tuples of the expression are values from \( \text{dom} (P) \) (that is, the values appear either in \( P \) or in a tuple of a relation mentioned in \( P \)).

2. For every “there exists” subformula of the form \( \exists x \ (P_1(x)) \), the subformula is true if and only if there is a value of \( x \) in \( \text{dom} (P_1) \) such that \( P_1(x) \) is true.

3. For every “for all” subformula of the form \( \forall x \ (P_1(x)) \), the subformula is true if and only if \( P_1(x) \) is true for all values \( x \) from \( \text{dom} (P_1) \).
Find all students who have taken all courses offered in the Biology department

\[
\{ <i> \mid \exists \ n, \ d, \ tc \ ( <i, \ n, \ d, \ tc> \in student \ \wedge \\
(\forall \ ci, \ ti, \ dn, \ cr \ ( <ci, \ ti, \ dn, \ cr> \in course \ \wedge \ dn = \text{“Biology”} \\
\Rightarrow \exists \ si, \ se, \ y, \ g \ ( <i, \ ci, \ si, \ se, \ y, \ g> \in takes )))} 
\]

Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.
End of Chapter 6