Chapter 27: Formal-Relational Query Languages
Outline

- Tuple Relational Calculus
- Domain Relational Calculus
- Datalog
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 (\(\land\)), or (\(\lor\)), not (\(\neg\))
4. Implication (\(\Rightarrow\)): \(x \Rightarrow y\), if \(x\) if true, then \(y\) is true
   \[x \Rightarrow y \equiv \neg x \lor y\]
5. Set of quantifiers:
   - \(\exists t \in r (Q(t))\) ≡ ”there exists” a tuple in \(t\) in relation \(r\) such that predicate \(Q(t)\) is true
   - \(\forall t \in r (Q(t))\) ≡ \(Q\) 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

\[
\{ t \mid \exists s \in instructor \ (t [name ] = s [name ] \\
\quad \land \exists u \in department \ (u [dept_name ] = s[dept_name] “ \\
\quad \land u [building] = “Watson” )\}
\]

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

\[
\{ t \mid \exists s \in section \ (t [course_id ] = s [course_id ] \land \\
\quad s [semester] = “Fall” \land s [year] = 2009 \\
\quad \lor \exists u \in section \ (t [course_id ] = u [course_id ] \land \\
\quad u [semester] = “Spring” \land u [year] = 2010 )\}
\]
Example Queries

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

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

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

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

- Find all students who have taken all courses offered in the Biology department
  - $\{ t | \exists r \in \text{student} (t[\text{ID}] = r[\text{ID}]) \land (
    \forall u \in \text{course} (u[\text{dept\_name}] = \text{"Biology"}) \implies
    \exists s \in \text{takes} (t[\text{ID}] = s[\text{ID}] \land
    s[\text{course\_id}] = u[\text{course\_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 \( P \).
Safety of Expressions (Cont.)

- Consider again that query to find all students who have taken all courses offered in the Biology department

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

- 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:

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

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

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

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

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

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

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

This case can also be written as

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

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

\[
\{<c> \mid \exists a, s, y, b, r, t \ ( <c, a, s, y, b, r, t > \in section \land \\
\quad s = \text{"Fall"} \land y = 2009 ) \land \\
\quad \exists a, s, y, b, r, t \ ( <c, a, s, y, b, r, t > \in section ) \land \\
\quad s = \text{"Spring"} \land y = 2010)}
\]
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 \( \text{dom}(P_1) \).
Universal Quantification

- Find all students who have taken all courses offered in the Biology department
  
  \[
  \{ < i > | \exists n, d, tc ( < i, n, d, tc > \in \text{student} \land \\
  (\forall ci, ti, dn, cr ( < ci, ti, dn, cr > \in \text{course} \land dn = \text{“Biology”} \\
  \Rightarrow \exists si, se, y, g ( <i, ci, si, se, y, g> \in \text{takes} ))\})
  \]

- Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.
Datalog
End of Chapter 27