CSEN 501 – CSEN501 - Databases I

Lecture 6:
The Relational Calculus

Prof. Dr. Slim Abdennadher
Dr. Nada Sharaf

German University Cairo, Faculty of Media Engineering and Technology

Relational Data Manipulation Languages

- Variety languages used by relational database management systems
  - Procedural languages: The user tells the system how to manipulate the data, e.g. Relational Algebra
  - Declarative languages: the user states what data is needed but not exactly how it is to be located, e.g. Relational Calculus and SQL
  - Graphical languages: allowing the user to give an example or an illustration of what data should be found, e.g. QBE

What is the Relational Calculus?

- **Relational calculus** is a formal query language where we write one declarative expression to specify a retrieval request.
- A calculus expression specifies what is to be retrieved rather than how to retrieve it. Therefore, relational calculus is considered to be a nonprocedural language.
- There are two types of relational calculus:
  - **Tuple relational calculus**
  - Domain relational calculus.

Tuple Relational Calculus

- The tuple relational calculus is based on specifying a number of tuple variables.
- Each tuple variable usually ranges over a particular database relation.
- A tuple expression is written as

$$\{t | C(t)\}$$

  Where t is a tuple variable and C(t)is a conditional expression involving t.
- Example: Find all employees whose salary is more than 50,000.

$$\{t | employee(t) \wedge t.salary > 50000\}$$

  Note: The condition employee(t) specifies that the range relation of tuple variable t is employee.

Tuple Relational Calculus Expressions

- A general expression of a tuple relational calculus is of the form:

$$\{t_1.A_j, t_2.A_k, \ldots, t_n.A_m | C(t_1, t_2, \ldots, t_n, t_{n+1}, \ldots t_{n+m})\}$$

  Where:

  - $t_1, t_2, \ldots, t_{n+m}$ are tuple variables
  - A_i is an attribute of the corresponding relation on which ti ranges.
  - C is a condition or a formula of the tuple relational calculus.
- In Relational calculus a safe expression is the one guaranteed to yield a finite number of tuples otherwise the expression is unsafe.
- Example:

$$\{t | \neg(employee(t))\}$$

  is unsafe expression.

Tuple Relational Calculus - Atoms

- An atom is a building block of a relational calculus expression.
- An atom can have one of the following forms:
  - $R(t_i)$:where R is a relation name. This atom specifies the range of tuple variable ti.
  - $t_i.A \ op \ t_j.B$: where op is one of the comparison operators.
  - $t_i.A \ op \ c$ or $c \ op \ t_i.B$: where op is one of the comparison operators and c is a constant value.
- Each atom evaluates to either true or false for a specific value of tuples, This is called the truth value of an atom.

Relational Schema Example

employee(fname, lname, $\underline{\text{ssn}}$, bdate, address, sex,
salary, dnr, superssn)

department(dname, $\underline{\text{dnr}}$, dMangssn)

departmentLocation($\underline{\text{dnr,dlocation}}$)

project(pname,$\underline{\text{pnr}}$,plocation,dnr)

workson($\underline{\text{ssn,pnr}}$,hours)

dependent($\underline{\text{ssn,dependentname}}$,sex,bdate,relationship)

Examples (I)

- Retrieve all employees.

$$\{e | employee(e)\}$$

- Retrieve the names of all employees.

$$\{e.fname, e.lname | employee(e)\}$$

- Retrieve employees with salary greater than 5000.

$$\{e | employee(e) \wedge e.salary > 5000\}$$

- Retrieve the names and salary of all employees who work in department 1 and whose salary exceeds 5000

$$\{e.fname, e.lname, e.salary | employee(e) \wedge e.dnr = 1 \wedge e.salary > 5000\}$$

Examples (II)

- For each employee, retrieve the employee's first and last name and the first and last name of his/her immediate supervisor.

$$\{e.fname, e.lname, s.fname, s.lname | employee(e) \wedge employee(s)$$

$$\wedge e.superssn = s.ssn\}$$

Tuple Relational Calculus - Formulas

- A formula (condition) is made up of one or more atoms connected via the logical operators: $\wedge$, $\vee$, and $\neg$.
- A **formula** can be recursively defined as:
  - Every atom is a formula
  - If F and G are formulas, then so are the following:
    - $F \wedge G$
    - $F \vee G$
    - $\neg F$

Universal and Existential Quantifiers

- Two quantifiers symbols may appear in a formula:
  - The existential quantifier: $\exists$
  - The universal quantifier: $\forall$
- The truth values of formula with quantifiers is based on the concept of free and bound tuple variables in the formula.

Free and Bound Tuple variables

- An occurrence of a tuple variable t in a formula F that is an atom is free in F.
- An occurrence of a tuple variable t is free or bound in a formula made up of logical connectives - $(F \wedge G), (F \vee G) and ()$, - depending whether it is free or bound in F or G.
- In the formula of the form $F = (G \wedge H)$ or $F = (G \vee H)$,a tuple variable may be free in G and bound in H, or vise versa. In this case, one occurrence of the tuple variable is bound and the other is free in F
- All free occurrences of a tuple variable t in F are bound in a formula $F = (\forall t)(G)$ or $F = (\exists t)(G)$. The tuple variable is bound to the quantifier specified in F.

Truth Value of a Formula With Quantifier

- If F is a formula then so is $(\exists t)(F)$, where t is a tuple variable.
- The formula $(\exists t)(F)$ is true if the formula F evaluates to true for some (at least one) tuple assigned to free occurrence of t in F, otherwise $(\exists t)(F)$ is false.
- If F is a formula then so is $(\forall t)(F)$,where t is a tuple variable.
- The formula $(\forall t)(F)$ is true if the formula F evaluates to true for every tuple (in the universe) assigned to free occurrence of t in F, otherwise $(\forall t)(F)$ is *false*.

Free Variables in Expressions

- The only free tuple in a relational calculus expression should be those that appear to the left of the bar (|).
- A free variable is bound successively to each tuple.

Examples with Existential & Universal Quantifiers

- Retrieve the name and address of all employees who work for the research department.
  $\{t.fname, t.lname, t.address | employee(t) \wedge (\exists d)(department(d) \wedge d.dname = \text{`}Research\text{'} \wedge d.dnr = t.dnr)\}$
- Find the names of employees who have no dependents.
  $\{e.fname, e.lname | employee(e) \wedge \neg(\exists d)(dependent(d) \wedge e.ssn = d.ssn)\}$
  - Push negation to the atoms:
    $\{e.fname, e.lname | employee(e) \wedge (\forall d)(\neg dependent(d) \vee \neg(e.ssn = d.ssn))\}$
  - Use Implication:
    $\{e.fname, e.lname | employee(e) \wedge (\forall d)(dependent(d) \Rightarrow \neg(e.ssn = d.ssn))\}$
- List the names of managers who have at least one dependent.
  $\{e.fname, e.lname | employee(e) \wedge (\exists d)(\exists p)(department(d) \wedge dependent(p) \wedge e.ssn = d.ssn \wedge p.ssn = e.ssn)\}$