CSC 742
Database Management Systems

Topic #8:
Relational Calculus

Relational Calculus: 1

- Can define the information to be retrieved
  - not any specific series of operations
  - In contrast, relation algebra defines the sequence of operations.
- Closer than relational algebra to how users would formulate queries
  - in terms of their information needs, rather than in terms of operations.

Relational Calculus: 2

- Involves variables that range over tuples or domain attributes
  - Tuple Relational Calculus: Variables range over tuples
  - Domain Relational Calculus: Variables range over domain attributes
- Gives a way of referring to attributes of relations (and the specific tuples in them)
- Is based on predicate logic—gives the usual quantifiers to construct complex queries.

Tuple Relational Calculus (TRC)

- TRC is based on tuple variables.
- Each tuple variable ranges over a relation.
  - The relation is called the range relation of the variable.
- Examples:
  - {t | Employee(t) and t.salary > 50000}
    - Retrieve all attribute values.
  - {t.Fname, t.Lname | Employee(t) and t.salary > 50000}
    - Retrieve selected attributes.

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	45000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	56000
555-44-5555	English	Joyce	53000

{t | Employee(t) and t.salary > 50000}

{t.Fname, t.Lname | Employee(t) and t.salary > 50000}

TRC (Cont'd)

- To specify a TRC expression:
  - Specify the range relation R of each tuple variable t.
    - In the form of R(t).
  - Specify a condition to select particular combinations of tuples.
  - Specify a set of attributes to be retrieved.

{t.Fname, t.Lname | Employee(t) and t.salary > 50000}

Employee

SSN	Lname	Fname	Salary
111-22-3333	Smith	John	30000
121-23-3333	Wong	Frank	45000
153-32-1342	Wallace	Jennifer	43000
154-33-3333	Borg	James	56000
555-44-5555	English	Joyce	53000

1. Retrieve the salary of 'John Smith'.
2. Retrieve the names of the employees whose salary is less than 40000.

General TRC expression

- {t1.A, ..., tn.An| COND(t1, ..., t(n+m))}
  - COND → COND & COND
    | COND \/ COND
    | ~COND
    | ATOM
    | (∀t: COND)
    | (∃t: COND)
  - Forms of ATOMs
    - R(t)
    - Ti.A *op* Tj.B, where *op* ∈ {=, >, ≥, <, ≤, ≠}.
    - Ti.A *op* c, where *op* ∈ {=, >, ≥, <, ≤, ≠} and c is a constant.

Tuple Variables

- Tuple variables in a COND may be
  - *bound*: within the scope of a quantifier
  - *free*: not within the scope of a quantifier
- Intuitively, the COND describes a property of its free variables
- All and only the free variables of COND can show up in the answer tuple as specified in TRC

{t.Fname, t.Lname | Employee(t) and t.salary > 50000}

{t.Fname, t.Lname | Employee(t) and t.salary > 50000 and Department(d) and d.MgrSSN = t.SSN}

{t.Fname, t.Lname | Employee(t) and t.salary > 50000 and (∃d)(Department(d) and d.MgrSSN = t.SSN)}

Conventions

- Nested forall expressions can be simplified: (∀ t1: (∀ t2: C)) = (∀ t1, t2: C)
- Nested exists expressions can be simplified: (∃ t1: (∃ t2: C)) = (∃ t1, t2: C)
- For safety (described below), all tuple variables are declared to belong to some relation, e.g., project(p1) or employee(e2)

TRC Examples: 1

- Find names of employees in the research dept

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Department(Dname, Dnumber, MgrSSN, MgrStartDate)

TRC Examples: 2

- For every project in 'Stafford' list the controlling dept number and the dept manager's last name

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Department(Dname, Dnumber, MgrSSN, MgrStartDate)

Project(Pname, Pnumber, Plocation, Dnum)

TRC Examples: 3

- Find SSNs of employees who work on all the projects controlled by dept 5.

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Works_on(ESSN, Pno, Hours)

Project(Pname, Pnumber, Plocation, Dnum)

Using ER Diagrams

ER diagrams can be used to understand and formulate queries

- Paths
  - correspond to constraints on what is being retrieved
  - can be mapped to calculus expressions
  - encapsulate some common English grammatical constructions

- Find SSNs of employees who work on all the projects controlled by dept 5.

Employee(Fname, Lname, SSN, Bdate, Address, Sex, Salary, SuperSSN, Dno)

Works_on(ESSN, Pno, Hours)

Project(Pname, Pnumber, Plocation, Dnum)

Safe TRC Expressions

- A Safe TRC expression:
  - One that is guaranteed to yield a finite number of tuples as its result.
- An unsafe TRC expression
  - {t | not (Employee (t))}

Safe TRC Expressions (Cont'd)

- Domain of a TRC expression:
  - set of all values that appear in the expression or in any tuple of any relation mentioned in the expression
  - Example: {t | not (Employee (t))}
- Safe expressions:
  - those whose result includes values only from their own domain
  - {t| Employee(t) & ...} is *safe* because the answer is a subset of the Employee relation
  - {t | not (Employee (t))} is *not safe* because …

Safe TRC Expressions (Cont'd)

- Outer-level negations are unsafe
- Interestingly, even disjunctions can lead to trouble:
  - {<x, y>| R(x) \/ S(y)} returns tuples whose first member is in relation R or whose second member is in relation S. Suppose x1 is in R. Then the result contains <x1, y> for every y, irrespective of whether y is in S. Hence this query is not safe.

Safe TRC Expressions (Cont'd)

Unsafe expressions are undesirable because they can involve

- infinite answer sets
- answers using values that the user (who produced the query) may have no way of predicting.

Limitations of TRC

- It cannot express queries involving
  - aggregations
  - closure
- It cannot express non-query operations
  - insert
  - delete
  - update

because those involve modeling the changes in the state of a database.