Chapter 6

Relational Calculus

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Content

- Introduction
- Tuple Relational Calculus (TRC)
- Domain Relational Calculus (DRC)

Introduction

- Is the formal query language
- Introduced by Codd in 1972, “Data Base Systems”, Prentice Hall, p33-98

- Properties
  - Nonprocedural language – declarative language
    - Calculus expression specifies *what is to be retrieved* rather than *how to retrieve*
  - ***One declarative expression to specify a retrieval request***
    - ***There is no description of how to evaluate query***
  - ***A calculus expression may be written in different way***
    - ***The way it is written has no bearing on how a query should be evaluated***

Introduction

- Categories
  - **Tuple relational calculus – TRC**
    - SQL
  - Domain relational calculus – DRC
    - QBE (Query By Example)
    - DataLog (Database Logic)

Content

- Introduction
- **Tuple relational calculus**
- Domain relational calculus

Tuple relational calculus – TRC

- A simple tuple calculus query is of the form

$$\{ t.A \mid P(t) \}$$

  - *t* is a tuple variable
    - Its value is any individual tuple from a relation
    - *t.A* is a value of a tuple *t* at an attribute *A*

  - | *(vertical bar) is used to divide the query into two parts:*
    - *P* is a conditional expression involving *t*
      - P(t) has the TRUE or FALSE value depending on *t*
    - The result is the set of all tuples *t* that satisfy *P(t)*

Example 1

- Find employees whose salary is larger than 30000

$$\{ t \mid \underbrace{t \in EMPLOYEE}_{P(t)} \wedge \underbrace{t.SALARY > 30000}_{P(t)} \}$$

  - $t \in EMPLOYEE$: TRUE
    - If *t* is an instance of relation EMPLOYEE
  - t.SALARY > 30000 : TRUE
    - If the attribute SALARY of tuple *t* has a value being larger than 30000
- The result is all tuples t which satisfy:
  - $t \in EMPLOYEE$ and t.SALARY > 30000

Example 2

- Retrieve the SSN and first name of employees whose salary is larger than 30000

$$\{ t.SSN, t.FNAME \mid t \in EMPLOYEE \wedge t.SALARY > 30000 \}$$

  - The set of SSNs and first names of employees of tuples *t* such that *t* are instances of EMPLOYEE and their values are larger than 30000 at the attribute SALARY

Example 3

- Find employees (SSN) who work for the department ‘Nghien cuu’

$$\mathbf{t}.\text{SSN} \mid \mathbf{t} \in \text{EMPLOYEE}$$

$$\mathbf{s} \in \text{DEPARTMENT} \wedge \mathbf{s}.\text{DNAME} = \text{‘Nghien cuu’}$$

  - Select tuples ***t*** that belong to relation *EMPLOYEE*
  - Compare ***t*** to a certain tuple ***s*** to find employees working for the department *‘Nghien cuu’*
  - Use the existential quantifier

$$\exists t \in R\ (Q(t))$$

Existing a tuple t of the relation R such that the expression Q(t) is TRUE → the result of the existential quantifier is TRUE

Example 3

- Find employees (SSN) who work for the department ‘Nghien cuu’

$$\{\ t.SSN \mid t \in EMPLOYEE \wedge$$
$$\exists s \in DEPARTMENT\ (\ s.DNAME = \text{‘Nghien cuu’} \wedge s.DNUMBER = t.DNO\ )\ \}$$

Q(s)

Example 4

- Find employees (FNAME) who work on projects or who have dependents

$$\{ t.FNAME \mid t \in EMPLOYEE \wedge (\exists s \in WORKS\_ON\ (t.SSN = s.ESSN) \vee \exists u \in DEPENDENT\ (t.SSN = u.ESSN)) \}$$

Example 5

- Retrieve the FNAME of employees who participate in projects and have dependents

$$\{ t.FNAME \mid t \in EMPLOYEE \wedge (\exists s \in WORKS\_ON\ (t.SSN = s.ESSN) \wedge \exists u \in DEPENDENT\ (t.SSN = u.ESSN)) \}$$

Example 6

- Find the FNAME of employees who work on projects and have no dependents

$$\{ t.FNAME \mid t \in EMPLOYEE \wedge$$

$$\exists s \in WORKS\_ON\ (t.SSN = s.ESSN) \wedge$$

$$\neg\exists u \in DEPENDENT\ (t.SSN = u.ESSN) \}$$

Example 7

- For each project in ‘TP HCM’, find the project number, the department number that controls the project and the FNAME of the manager

$$\{ s.PNUMBER, s.DNUM, t.FNAME \mid s \in PROJECT \wedge t \in EMPLOYEE \wedge$$
$$s.PLOCATION = \text{‘TP HCM’} \wedge \exists u \in DEPARTMENT$$
$$(u.DNUMBER = s.DNUM \wedge$$
$$u.MGRSSN = t.SSN) \}$$

Example 8

- Find employees (SSN) who work on <u>all</u> projects
  - Use the universal quantifier

$$\forall t \in R\ (Q(t))$$

If Q is TRUE with all tuples t of relation R, the universal quantifier is TRUE; otherwise FALSE.

Example 8a

- Find employees whose salary is highest.

$$\{ t.SSN, t.LNAME, t.FNAME \mid t \in EMPLOYEE \wedge$$

$$\forall e \in EMPLOYEE\ (t.Salary >= e.Salary) \}$$

Example 8

- Find employees (SSN, FNAME, LNAME) who work on all projects

$$\{ t.SSN, t.LNAME, t.FNAME \mid t \in EMPLOYEE \wedge$$
$$\forall s \in PROJECT (\exists u \in WORKS\_ON ($$
$$u.PNO = s.PNUMBER \wedge$$
$$u.ESSN = t.SSN)) \}$$

Example 9

- Find employees (SSN, LNAME, FNAME) who work on all projects controlled by the department 4

$$\{ t.SSN, t.LNAME, t.FNAME \mid t \in EMPLOYEE \wedge$$
$$\forall s \in PROJECT ($$
$$s.DNUM = 4 \wedge (\exists u \in WORKS\_ON ($$
$$u.PNO = s.PNUMBER \wedge$$
$$u.ESSN = t.SSN))) \}$$

X

Example 9

- Find employees (SSN, LNAME, FNAME) who work on all projects controlled by the department 4
  - Use the “implies” operator

$$P \Rightarrow Q$$

If P then Q

Example 9

- Find employees (SSN, LNAME, FNAME) who work on all projects controlled by the department 4

$$\{ t.SSN, t.LNAME, t.FNAME \mid t \in EMPLOYEE \wedge$$
$$\forall s \in PROJECT ($$
$$s.DNUM = 4 \Rightarrow (\exists u \in WORKS\_ON ($$
$$u.PNO = s.PNUMBER \wedge$$
$$u.ESSN = t.SSN))) \}$$

Example 9 – Solution 2

- Find employees (SSN, LNAME, FNAME) who work on all projects controlled by the department 4

$$\{ t.SSN, t.LNAME, t.FNAME \mid t \in EMPLOYEE \wedge$$
$$\forall s \in PROJECT ($$
$$s.DNUM \neq 4 \vee (\exists u \in WORKS\_ON ($$
$$u.PNO = s.PNUMBER \wedge$$
$$u.ESSN = t.SSN))) \}$$

Example 10

- a. Find employees whose salary is larger than at least one employee of department 4.
- b. Find employees whose salary is larger than all employees of department 4.

Formal definition

- A general expression is of the form

$$\{ t_1.A_i, t_2.A_j, \ldots, t_n.A_m \mid P(t_1, t_2, \ldots, t_n, \ldots, t_{n+m}) \}$$

  - $t_1, t_2, \ldots, t_n$ are tuple variables
  - $A_i, A_j, \ldots, A_m$ are attributes of tuples *t*
  - P is a condition or well-formed formula
    - P is made up of predicate calculus atoms

Tuple variable

- Free variable

$$\{ t \mid t \in EMPLOYEE \wedge t.SALARY > 30000 \}$$

t is a free variable

- Bound variable

$$\{ t \mid t \in EMPLOYEE \wedge \exists s \in DEPARTMENT\ (s.DNUMBER = t.PNO) \}$$

Free variable

Bound variable

Atoms

- (i) $t \in R$
  - t is a tuple variable
  - R is a relation

$t \in EMPLOYEE$

- (ii) $t.A\ \theta\ s.B$
  - A is an attribute of the tuple variable t
  - B is an attribute of the tuple variable s
  - θ is comparison operators, eg. $<$, $>$, $\leq$, $\geq$, $\neq$, $=$

$t.SSN = s.ESSN$

- (iii) $t.A\ \theta\ c$
  - C is a constant
  - A is an attribute of the tuple variable t
  - θ is comparison operators, eg. $<$, $>$, $\leq$, $\geq$, $\neq$, $=$

$t.SALARY > 30000$

Atoms

- Each of atoms evaluates to either TRUE or FALSE for a specific combination of tuples

- Formula (i) $t \in R$
  - TRUE value if *t* is a tuple of the specified relation *R*
  - FALSE value if *t* does not belong to *R*

R	A	B	C
	α	10	1
	α	20	1

t1 = <α, 10, 1> t1 ∈ R has the TRUE value

t2 = <α, 20, 2> t2 ∈ R has the FALSE value

Atoms

- Formula (ii) $t.A\ \theta\ s.B$ and (iii) $t.A\ \theta\ c$
  - If the tuple variables are assigned to tuples such that they satisfy the condition, then the atom is TRUE

R	A	B	C
	α	10	1
	α	20	1

If *t* is the tuple <α, 10, 1>

Then *t.B > 5* has the TRUE value (10 > 5)

Rules

- (1) Every atom is formula
- (2) If P is a formula then
  - $\neg$P is a formula
  - (P) is a formula
- (3) If P1 and P2 are formulas then
  - P1 $\vee$ P2 is a formula
  - P1 $\wedge$ P2 is a formula
  - P1 $\Rightarrow$ P2 is a formula

Rules

- (4) If P(t) is a formula then
  - $\forall t \in R\ (P(t))$ is a formula
    - TRUE when *P(t)* is TRUE for all tuples in *R*
    - FALSE when there is one tuple that makes *P(t)* FALSE
  - $\exists t \in R\ (P(t))$ is a formula
    - TRUE when there exists some tuple that makes *P(t)* TRUE
    - FALSE when *P(t)* is FALSE for all tuples *t* in *R*

Rules

- (5) If P is an atom then
  - Tuple variables *t* in *P* are free variables

- (6) Formulas $P=P1 \wedge P2$, $P=P1 \vee P2$, $P=P1 \Rightarrow P2$
  - A variable *t* in *P* is free or bound variable will depends on its role in P1 and P2

Transform

- (i) $P1 \wedge P2 = \neg (\neg P1 \vee \neg P2)$
- (ii) $\forall t \in R\ (P(t)) = \neg \exists t \in R\ (\neg P(t))$
- (iii) $\exists t \in R\ (P(t)) = \neg \forall t \in R\ (\neg P(t))$
- (iv) $P \Rightarrow Q = \neg P \vee Q$

Safe expression

- Examine

$$\{ t \mid \neg(t \in \text{EMPLOYEE}) \}$$

- Unsafe
  - Many tuples in the universe that are not EMPLOYEE tuples
  - Even though they do not exist in the database
  - The result is infinitely numerous

Safe expression

- Safe expression
  - Guarantee to yield *a finite number of tuples*

- A formula P is called safe expression
  - If its resulting values are from the domain of P
    - The domain of a tuple relational calculus expression: DOM(P)
    - The set of all values
      - Either appear as constant values in P
      - Or exist in any tuple in the relation referenced in P

Safe expression

- Example

$$\{ t \mid t \in EMPLOYEE \wedge t.SALARY > 30000 \}$$

  - DOM(t ∈ EMPLOYEE∧ t.SALARY > 30000)
  - The set of values
    - Lager than 30000 at the attribute SALARY
    - Other values at the remaining attributes that appear in EMPLOYEE
  - Safe expression

Content

- Introduction
- Tuple relational calculus
- **Domain relational calculus**

Domain relational calculus

- An expression of the domain calculus is of the form

$$\{ x_1, x_2, \ldots, x_n \mid P(x_1, x_2, \ldots, x_n) \}$$

  - $x_1, x_2, \ldots, x_n$ are domain variables
    - Accepting single values from the domain of attributes
  - P is a formula of variables $x_1, x_2, \ldots, x_n$
    - P is formed from atoms
  - The result
    - The set of values such that when assigned to variables x_i, they make *P* TRUE

Example 1

- Find employees whose salary is larger than 30000

$$\{ r, s \mid \exists x (\langle p, q, r, s, t, u, v, x, y, z \rangle \in \text{EMPLOYEE} \wedge x > 30000) \}$$

Example 3

- Find employees (SSN) who work for the department ‘Nghien cuu’

$$\{ s \mid \exists z (<p, q, r, s, t, u, v, x, y, z> \in EMPLOYEE \wedge \exists a, b (<a, b, c, d> \in DEPARTMENT \wedge a = \text{‘Nghien cuu’} \wedge b = z)) \}$$

Example 10

- Find employees (SSN, LNAME, FNAME) who have no dependents

$$\{ p, r, s \mid \exists s (<p, q, r, s, t, u, v, x, y, z> \in EMPLOYEE \wedge \neg \exists a (<a, b, c, d, e> \in DEPENDENT \wedge a = s)) \}$$

Atoms

- (i) $\langle x_1, x_2, \ldots, x_n \rangle \in R$
  - x_i is a domain variable
  - R is a relation with *n* attributes
- (ii) $x \; \theta \; y$
  - x, y are domain variables
  - Domains of *x* and *y* are identical
  - θ is comparison operators, eg. $<$, $>$, $\leq$, $\geq$, $\neq$, $=$
- (iii) $x \; \theta \; c$
  - c is a constant
  - x is a domain variable
  - θ is comparison operators, eg. $<$, $>$, $\leq$, $\geq$, $\neq$, $=$

Discussion

- Atoms evaluate to either TRUE or FALSE for a set of values
  - Called the truth values of the atoms
- Rules and transforms are in the similar way to the tuple calculus

Safe expression

- Examine

$$\{ p, r, s \mid \neg (<p, q, r, s, t, u, v, x, y, z> \in \text{EMPLOYEE}) \}$$

  - Values in the result do not belong to the domain of the expression
  - Unsafe

Safe expression

- Examine

$$\{ x \mid \exists y (\langle x, y \rangle \in R) \wedge \exists z (\neg \langle x, z \rangle \in R \wedge P(x, z)) \}$$

Formula 1 Formula 2

- *R* is a relation with a finite number of values
- We also have a finite number of values that does not belong to *R*
- Formula 1: examine values in *R* only
- Formula 2: could not validate cause we do not know the finite number of values of variable *z*

Safe expresion

- Expression

$$\{ x_1, x_2, \ldots, x_n \mid P(x_1, x_2, \ldots, x_n) \}$$

is safe if :

- Values that appear in tuples of the expression must belong to the domain of *P*
- ∃ quantifiers: expression ∃x (Q(x)) is TRUE iff
  - Values of *x* belong to DOM(Q) and make *Q(x)* TRUE
- ∀ quantifiers: expression ∀x (Q(x)) is TRUE iff
  - *Q(x)* is TRUE for all values of *x* belonging to DOM(Q)