BITS Pilani
Pilani Campus

Database Systems (CSF212)
Lecture – 10

Query Language

Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form

  $\{ t \mid P(t) \}$, where t is the set of all tuples for which predicate P is true
- 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 Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., <, $\leq$, $=$, $\neq$, >, $\geq$)
3. Set of connectives: and (^), or (v), not ($\neg$)
4. Implication ($\Rightarrow$): x $\Rightarrow$ y, if x if true, then y is true

$$x \Rightarrow y \equiv \neg x \vee y$$

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

Banking Example

- ***branch*** ***(branch-name, branch-city, assets)***
- ***customer*** ***(customer-name, customer-street, customer-city)***
- ***account*** ***(account-number, branch-name, balance)***
- ***loan*** ***(loan-number, branch-name, amount)***
- ***depositor*** ***(customer-name, account-number)***
- ***borrower*** ***(customer-name, loan-number)***

Example Queries

loan (loan-number, branch-name, amount)

Q: Find the *loan-number, branch-name*, and *amount* for loans of over $1200

$$R = \{t \mid t \in loan \wedge t[amount] > 1200\}$$

Q. Find the loan number for each loan of an amount greater than $1200

$$R = \{t \mid \exists\, s \in \text{loan}\ (t[loan\text{-}number] = s[loan\text{-}number] \wedge s[amount] > 1200)\}$$

Q. Find the loan number and branch-name for each loan of an amount greater than $1200

$$R = \{t \mid \exists\, s \in \text{loan}\ (t[loan\text{-}number] = s[loan\text{-}number] \wedge t[branch\text{-}name] = s[\text{branch}\text{-}name] \wedge s[amount] > 1200)\}$$

Example Queries

depositor *(customer-name, account-number)*

borrower *(customer-name, loan-number)*

Q. Find the names of all customers having a loan, an account, or both at the bank

$\{t \mid \exists s \in borrower(t[\textit{\textbf{customer-name}}] = s[customer\text{-}n$

$\exists u \in depositor(t[\textit{\textbf{customer-name}}] = u[customer\text{-}nam$

Q. Find the names of all customers who have a loan and an account at the bank

$R = \{t \mid \exists s \in borrower(t[\textit{\textbf{customer-name}}] = s[customer\text{-}name]) \wedge \exists u \in depositor(t[\textit{\textbf{customer-name}}] = u[customer\text{-}name])$

Example Queries

***loan* (loan-number, branch-name, amount)**

***borrower* (customer-name, loan-number)**

Q. "Find the names of all customers having a loan at the "Perryridge" branch

R = {t / $\exists s \in$ *borrower*(t[***customer-name***] = s[*customer-name*] $\wedge$ $\exists u \in$ *loan*(u[*branch-name*] = "Perryridge" $\wedge$ u[*loan-number*] = s[*loan-number*]))}

Example Queries

***loan* (loan-number, branch-name, amount)**

***depositor* (customer-name, account-number)**

***borrower* (customer-name, loan-number)**

Q. Find the names of all customers who have a loan at the Perryridge branch, but no account at any branch of the bank

R = {*t* / ∃*s* ∈ *borrower*(*t*[***customer-name***] = *s*[customer-name] ^ ∃*u* ∈ *loan* (*u*[*branch-name*] = "Perryridge"^ *u*[*loan-number*] = *s*[loan-*number*])) ^ **not** ∃*v* ∈ *depositor* (*v*[*customer-name*] = *t*[**customer-name**]) }

Example Queries

customer (customer-name, customer-street, customer-city)

loan (loan-number, branch-name, amount)

borrower (customer-name, loan-number)

Q. Find the names of all customers having a loan from the Perryridge branch, and the cities they live in

$$= \{t \mid \exists s \in loan(s[branch\text{-}name] = \text{“Perryri}$$
$$\wedge\ \exists u \in borrower\ (u[loan\text{-}number] = s[loan\text{-}num$$
$$\wedge\ t\ [\boldsymbol{customer\text{-}name}] = u[customer\text{-}na$$
$$\wedge\ \exists\ v \in customer\ (u[customer\text{-}name] = v[customer\text{-}na$$
$$\wedge\ t[\boldsymbol{customer\text{-}city}] = v[customer\text{-}city])$$

Example Queries

***branch** (branch-name, branch-city, assets)*

***customer** (customer-name, customer-street, customer-city)*

***account** (account-number, branch-name, balance)*

***depositor** (customer-name, account-number)*

Q. Find the names of all customers who have an account at all branches located in Brooklyn:

R ={*t* / ∃ c ∈ customer (*t*[**customer.name**] = c[customer-name]) ˄ ∀ *s* ∈ *branch* (*s*[*branch-city*] = “Brooklyn” ⇒

∃ *u* ∈ *account* (*s*[*branch-name*] = *u*[branch-name]

˄ ∃ *s* ∈ *depositor* (*t*[***customer-name***] = c[customer-name] ˄ c[*account-number*] =

Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations
- dom(P) : Cross product of the domains of all the relations used in formula P
- For example, {t | $\neg$ $t \in$ *loan*} results in an infinite relation. dom(P) = {integer X string X float}, which is infinite
- **Safe expression:** An expression {t | $P(t)$} 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*

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

***loan* (loan-number, branch-name, amount)**

***borrower* (customer-name, loan-number)**

Q. Find the *loan-number*, *branch-name*, and *amount* for loans of over $1200

$\{< l, b, a > \mid < l, b, a > \in loan \wedge a > 1200\}$

Q. Find the names of all customers who have a loan of over $1200

$\{< c > \mid \exists\, l, b, a\, (< c, l > \in borrower \wedge < l, b, a > \in loan \wedge a > 1200)\}$

Example Queries

***loan* (loan-number, branch-name, amount)**

***borrower* (customer-name, loan-number)**

Q. Find the names of all customers who have a loan from the Perryridge branch and the loan amount:

$\{< c, a > \mid \exists\, l\, (< c, l > \in borrower \wedge \exists b (< l, b, a > \in loan \wedge$

$b = \text{“Perryridge”}))\}$

or

$\{< c, a > \mid \exists\, l\, (< c, l > \in borrower \wedge$
$< l, \text{“Perryridge”}, a > \in loan)\}$

Example Queries

***loan* (loan-number, branch-name, amount)**

***borrower* (customer-name, loan-number)**

***depositor* (customer-name, account-number)**

***account* (account-number, branch-name, balance)**

Q: Find the names of all customers having a loan, an account, or both at the Perryridge branch:

$$\{< c > \mid \exists\, l\, (\{< c, l > \in borrower \wedge \exists\, b,a(< l, b, a > \in loan \wedge b = \text{“Perryridge”})) \vee \exists\, a(< c, a > \in depositor \wedge \exists\, b,n(< a, b, n > \in account \wedge b = \text{“Perryridge”}))\}$$

Safety of Expressions

IDK THIS

A DRC formula $\{ < x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n)\}$ is safe if all of the following hold:

- All values that appear in tuples of the expression are values from *dom*(P) (that is, the values appear either in P or in a tuple of a relation mentioned in P).
- 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 *dom*(P_1) such that $P_1(x)$ is true.
- 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 *dom* (P_1).