Florida Institute of Technology

Tuple and Domain Calculus

- Tuple Relational Calculus
- Domain Relational Calculus

Banking Example

- Recall the banking database:

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)

Tuple Relational Calculus

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

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

  - Read as “the set of all tuples t such that predicate P is true for t”
  - P is a formula similar to that of the predicate calculus

Predicate Calculus Formula

- The predicate P(t) will contain several types of syntactic elements:

- Tuple and relation variables:

$t \in customer$	*-- t has all of the attributes of customer*
$u \in depositor$	*-- u has all the attributes of depositor*
$v \in loan$	*-- v has all the attributes of load*

Predicate Calculus Formula

- Attribute names:

t[customer-name]	*t[customer-street]*	*t[customer-city]*
u[account-number]	*u[customer-name]*	
v[amount]	*v[loan-number]*	*v[branch-name]*

Predicate Calculus Formula

- Comparisons: $<, \leq, =, \neq, >, \geq$

t[customer-name] = “Smith”

u[account-number] = A-175

v[amount] ≥ 1000

t[customer-city] = “Orlando”

“Orlando” = t[customer-city]

u[loan-number] = v[loan-number]

Predicate Calculus Formula

- Connectives: $\wedge$, v, $\neg$

$u[\text{account-number}] = \text{A-175} \wedge u[\text{balance}] > 1000$

$(t[\text{customer-name}] = \text{“Smith”} \wedge t[\text{city}] = \text{“Orlando”}) \vee \neg (u[\text{account-number}] = \text{A-175})$

Predicate Calculus Formula

- Implication: x $\Rightarrow$ y, if x is true, then y is true

  (t[customer-name] = "Smith") $\Rightarrow$ (t[city] = "Orlando" v u[amount] < 500)

- By the way, x $\Rightarrow$ y $\equiv \neg$x v y

Predicate Calculus Formula

- Quantifiers:
  - $\exists$ t $\in$ r (Q(t)) $\equiv$ ”there exists” a tuple t in relation r such that Q(t) is true
  - $\forall$t $\in$ r (Q(t)) $\equiv$ Q(t) is true “for all” tuples t in relation r

$\exists t \in customer\ (t[customer\text{-}name] = \text{“Smith”})$

$\forall u \in account\ (u[balance] > 1000 \wedge u[branch\text{-}name] = \text{“Perryridge”})$

Example Queries

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

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

- How about the following?

$$\{t \mid t\,[amount] > 1200\}$$

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

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

Example Queries

- Find the loan number for each loan having an amount greater than $1200.

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

Note a relation on [*loan-number*] is implicitly defined by the expression.

Florida Institute of Technology

Example Queries

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

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

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

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

Example Queries

- If someone has an account or a loan at the bank, shouldn't their name appear in the *customer* relation?
- If it is the case that a name will appear in *customer* if and only if it appears in *borrower* or *depositor*, then:

$$\{t \mid \exists s \in customer(\ t[customer\text{-}name] = s[customer\text{-}name])\}$$

However, there is nothing in the text or schema description to indicate this is the case, so the depositor and borrower relations must be examined.

Example Queries

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

$$\{t \mid \exists s \in \textit{borrower}(t[\textit{customer-name}] = s[\textit{customer-name}] \wedge \exists u \in \textit{loan}(u[\textit{branch-name}] = \text{“Perryridge”} \wedge u[\textit{loan-number}] = s[\textit{loan-number}]))\}$$

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

$$\{t \mid \exists s \in \textit{borrower}(t[\textit{customer-name}] = s[\text{customer-name}] \wedge \exists u \in \textit{loan}(u[\textit{branch-name}] = \text{“Perryridge”} \wedge u[\textit{loan-number}] = s[\text{loan-}\textit{number}])) \wedge \neg \exists v \in \textit{depositor}\ (v[\textit{customer-name}] = t[\text{customer-name}])\}$$

Example Queries

- Find the names of customers and their cities of residence for those customers having a loan from the Perryridge branch.

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

- Note the above contains a mistake…and a couple of other issues too…

Safety of Expressions

- Some tuple calculus expressions result in infinite relations.

$$\{t \mid \neg\, t \in r\}$$

$$\{\, t \mid t[A]=5 \vee \textbf{true}\,\}$$

$$\{\, t \mid \neg\, \exists\, u \in \textit{customer}\ (t[\textit{customer-name}] = u[\textit{customer-name}])\}$$

- Such expressions don't make sense in the context of databases.

Florida Institute of Technology

Safety of Expressions

- Hence, we restrict our use to what are called “safe” expressions.
- An expression {*t* | *P*(*t*)} in tuple calculus is said to be safe if every value in the result of the expression is a function of some value in the database, i.e., appears in, or is a modified version of a value in one of the relations, or is a tuple or constant that appears in *P.*
- In other words, the results have to come directly or indirectly out of the database.

Example Queries

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

$$\{t \mid \forall\, s \in branch(s[branch\text{-}city] = \text{“Brooklyn”} \Rightarrow \exists\, u \in account(u[branch\text{-}name] = s[\text{branch-name}] \wedge \exists\, v \in depositor(v[account\text{-}number] = u[\text{account-number}] \wedge t[customer\text{-}name] = v[\text{customer-name}]))))\}$$

- Note that the above query is unsafe, but why?

- Consider a *branch* relation that consists of no Brooklyn branches.
  - Every customer is in the result.
  - Even “garbage” values are in the result.

Another, Safe Version

- Find the names of all customers who have an account at all branches located in Brooklyn (safe version):

$$\{t \mid \exists\, c \in customer\ (t[\text{customer.name}] = c[\text{customer-name}]) \wedge$$
$$\forall\, s \in branch(s[branch\text{-}city] = \text{“Brooklyn”} \Rightarrow$$
$$\exists\, u \in account(u[branch\text{-}name] = s[\text{branch-name}]$$
$$\wedge \exists\, v \in depositor(v[account\text{-}number] = u[\text{account-number}]$$
$$\wedge\ t[customer\text{-}name] = v[\text{customer-name}]))))\}$$

- Note how this solution eliminates the “garbage” values.

Example Queries

- What would happen if we changed the logical implication to a conjunction?

$$\{t \mid \forall\, s \in branch(s[branch\text{-}city] = \text{“Brooklyn”} \wedge \exists\, u \in account(u[branch\text{-}name] = s[\text{branch-name}] \wedge \exists\, v \in depositor(v[account\text{-}number] = u[\text{account-number}] \wedge t[customer\text{-}name] = v[\text{customer-name}]))))\}$$

- More specifically, what would be the meaning of the tuple calculus expression?

- This is a more restrictive query (somewhat arbitrary) than the original, however, unlike the first expression, it is safe (why?).

Example Queries

- Similarly one could ask what would happen if we changed the logical implication to a disjunction?

$$\{t \mid \forall\ s \in branch(s[branch\text{-}city] = \text{“Brooklyn”} \vee \exists\ u \in account(u[branch\text{-}name] = s[\text{branch-name}] \wedge \exists\ v \in depositor(v[account\text{-}number] = u[\text{account-number}] \wedge t[customer\text{-}name] = v[\text{customer-name}]))))\}$$

- Exercise:
  - What would be the meaning of the tuple calculus expression? More specifically, what would have to be true for a tuple to appear in the result?
  - Is the expression safe?

Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus
- A 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 *loan-number, branch-name,* and *amount* for loans of over $1200

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

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

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

- Find the names of all customers who have a loan at the Perryridge branch; also include 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

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

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

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

$$\{< c > \mid \exists\, s, n\, (< c, s, n > \in \text{customer}) \wedge \forall\, x,y,z((< x, y, z > \in \textit{branch} \wedge y = \text{“Brooklyn”}) \Rightarrow \exists\, a,b(< a, x, b > \in \textit{account} \wedge < c,a > \in \textit{depositor}))\}$$

Safety of Expressions

- As with tuple calculus, we restrict ourselves to those domain relational calculus expressions that are “safe,” i.e., whose resulting values come directly or indirectly from the database.