Relational Calculus

Calculus and Algebra

- Algebra: specifying how to obtain results
  - Procedural
  - SQL: specifying how to derive the results using the tables in the database
- What if a user does not know how to obtain the results?
  - Specifying what the results are can be easier than specifying how to get the results
  - Relational calculus specifies what instead of how

Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form {t | P (t) }
  – Results: the set of all tuples t such that predicate P is true for t
- 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 Calculus Formula

- A set of attributes and constants
- A set of comparison operators: (e.g., $<$, $\leq$, $=$, $\neq$, $>$, $\geq$)
- A set of connectives: and ($\wedge$), or (v), not ($\neg$)
- Implication ($\Rightarrow$): $x \Rightarrow y$, if x if true, then y is true $x \Rightarrow y \equiv \neg x \vee y$
- A 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
  - $\forall t \in r\, (Q\,(t\,)) \equiv$ Q is true "for all" tuples t in relation r

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

- Find the loan_number, branch_name, and amount for loans of over $1200
  - $\{t \mid t \in loan \land t[amount] > 1200\}$
- Find the loan number for each loan of an amount greater than $1200
  - $\{t \mid \exists s \in \text{loan} (t[loan\_number] = s[loan\_number] \land s[amount] > 1200)\}$
  - A relation on schema [*loan_number*] is implicitly defined by the query

Example Queries

- Find the names of all customers having a loan, an account, or both at the bank
  - $\{t \mid \exists s \in borrower\ (t[customer\_name] = s[customer\_name]) \vee \exists u \in depositor\ (t[customer\_name] = u[customer\_name])\}$
- Find the names of all customers who have a loan and an account at the bank
  - $\{t \mid \exists s \in borrower\ (t[customer\_name] = s[customer\_name]) \wedge \exists u \in depositor\ (t[customer\_name] = u[customer\_name])\}$

Example Queries

- Find the names of all customers having a loan at the Perryridge branch
  - $\{t \mid \exists s \in borrower\ (t[customer\_name] = s[customer\_name] \land \exists u \in loan\ (u[branch\_name] = \text{“Perryridge”} \land u[loan\_number] = s[loan\_number]))\}$

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch, but no account at any branch of the bank
  - $\{t \mid \exists s \in borrower\ (t[customer\_name] = s[customer\_name] \wedge \exists u \in loan\ (u[branch\_name] = \text{"Perryridge"} \wedge u[loan\_number] = s[loan\_number])) \wedge \text{not } \exists v \in depositor\ (v[customer\_name] = t[customer\_name])\}$

Example Queries

- Find the names of all customers having a loan from the Perryridge branch, and the cities in which they live
  - $\{t \mid \exists s \in loan\ (s\ [branch\_name\ ] = \text{“Perryridge”} \wedge \exists u \in borrower\ (u\ [loan\_number\ ] = s\ [loan\_number\ ] \wedge t\ [customer\_name\ ] = u\ [customer\_name\ ]) \wedge \exists\ v \in customer\ (u\ [customer\_name\ ] = v\ [customer\_name\ ] \wedge\ t\ [customer\_city\ ] = v\ [customer\_city\ ])))\}$

Example Queries

- Find the names of all customers who have an account at all branches located in Brooklyn
  - $\{t \mid \exists\ r \in customer\ (t\ [customer\_name\ ] = r\ [customer\_name\ ]) \land (\forall\ u \in branch\ (u\ [branch\_city\ ] = \text{"Brooklyn"} \Rightarrow \exists\ s \in depositor\ (t\ [customer\_name\ ] = s\ [customer\_name\ ] \land \exists\ w \in account\ (\ w[account\_number\ ] = s\ [account\_number\ ] \land (\ w\ [branch\_name\ ] = u\ [branch\_name\ ])))) \}$

Safety of Expressions

- $\{ t \mid \neg\, t \in r \}$ results in an infinite relation if the domain of any attribute of relation r is infinite
  - It is possible to write tuple calculus expressions that generate infinite relations
- To guard against the problem, we restrict the set of allowable expressions to safe expressions

Safe Expressions

- 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
  - More than just a syntax condition
  - { t | t [A] = 5 $\vee$ true } is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants 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 $\{ < x1, x2, \ldots, xn > \mid P(x1, x2, \ldots, xn)\}$
  - x1, x2, …, xn 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
  - $\{< c > \mid \exists\, l, b, a\ (< c, l > \in borrower \wedge < l, b, a > \in loan \wedge a > 1200)\}$

Example Queries

- Find the names of all customers who have a loan from the Perryridge branch and the loan amount
  - $\{< c, a > \mid \exists\, l\ (< c, l > \in borrower \wedge \exists b\ (< l, b, a > \in loan \wedge b = \text{“Perryridge”}))\}$
  - $\{< c, a > \mid \exists\, l\ (< c, l > \in borrower \wedge < l, \text{“Perryridge”}, a > \in loan)\}$

Example Queries

- Find the names of all customers having a loan, an account, or both at the Perryridge branch
  - $\{< c > \mid \exists\, l\, (< c, l > \in borrower \wedge \exists\, b,a\, (< l, b, a > \in loan \wedge b = \text{“Perryridge”})) \vee \exists\, a\, (< c, a > \in depositor \wedge \exists\, b,n\, (< a, b, n > \in account \wedge b = \text{“Perryridge”}))\}$
- Find the names of all customers who have an account at all branches located in Brooklyn
  - $\{< c > \mid \exists\, \text{s},n\, (< c, s, n > \in \text{customer}) \wedge \forall\, x,y,z\, (< x, y, z > \in branch \wedge y = \text{“Brooklyn”}) \Rightarrow \exists\, a,b\, (< x, y, z > \in account \wedge < c,a > \in depositor)\}$

Safety of Expressions

- The expression {< x1, x2, …, xn > | P(x1, x2, …, xn)} 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 ∃ x (P1(x)), the subformula is true if and only if there is a value of x in dom (P1) such that P1(x) is true
  – For every "for all" subformula of the form ∀x (P1 (x)), the subformula is true if and only if P1(x) is true for all values x from dom (P1)

Summary

- Relational calculus
  - An alternative query language
  - Specifying what instead of how
- Tuple relational calculus
- Domain relational calculus

To-Do-List

- Read Chapters 5.1 and 5.2 in the textbook
- Rewrite the queries in relational algebra using relational calculus