Tuple and Domain Calculus

- Tuple Relational Calculus
- Domain Relational Calculus
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
The predicate $P(t)$ will contain several types of syntactic elements:

- **Tuple and relation variables:**

  - $t \in \text{customer}$  -- $t$ has all of the attributes of customer
  - $u \in \text{depositor}$  -- $u$ has all the attributes of depositor
  - $v \in \text{loan}$  -- $v$ has all the attributes of load
Predicate Calculus Formula

- Attribute names:

  - $t[customer-name]$
  - $t[customer-street]$
  - $t[customer-city]$
  - $u[account-number]$
  - $u[customer-name]$
  - $v[amount]$
  - $v[loan-number]$
  - $v[branch-name]$
Comparisons: $<, \leq, =, \neq, >, \geq$

- $t[\text{customer-name}] = \text{“Smith”}$
- $u[\text{account-number}] = A-175$
- $v[\text{amount}] \geq 1000$
- $t[\text{customer-city}] = \text{“Orlando”}$
- “Orlando” = $t[\text{customer-city}]$
- $u[\text{loan-number}] = v[\text{loan-number}]$
Predicate Calculus Formula

Connectives: $\land$, $\lor$, $\lnot$

$u[account-number] = A-175 \land u[balance] > 1000$

$(t[customer-name] = \text{"Smith"} \land t[city] = \text{"Orlando"}) \lor \lnot (u[account-number] = A-175)$
Predicate Calculus Formula

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

$\left( t[\text{customer-name}] = \text{"Smith"} \right) \implies \left( t[\text{city}] = \text{"Orlando"} \lor u[\text{amount}] < 500 \right)$

- By the way, $x \implies y \equiv \neg x \lor y$
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

\[ \exists t \in \text{customer} (t[\text{customer-name}] = \text{"Smith"}) \]
\[ \forall u \in \text{account} (u[\text{balance}] > 1000 \land u[\text{branch-name}] = \text{"Perryridge"}) \]
Example Queries

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

\[ \{ t \mid t \in \text{loan} \land t[\text{amount}] > 1200 \} \]

- How about the following?

\[ \{ t \mid t[\text{amount}] > 1200 \} \]

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

\[ \{ t \mid \exists s \in \text{loan} \ (t[\text{loan-number}] = s[\text{loan-number}] \land t[\text{branch-name}] = s[\text{branch-name}] \land t[\text{amount}] = s[\text{amount}] \land t[\text{amount}] > 1200) \} \]
Example Queries

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

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

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

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

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

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

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

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

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.

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

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

\[ \{ t \mid \exists s \in borrower( t[customer-name] = s[customer-name] \\
\& \exists u \in loan( u[branch-name] = “Perryridge” \\
\& u[loan-number] = s[loan-number])) \\
\& \neg \exists v \in depositor( v[customer-name] = t[customer-name]) \} \]
Example Queries

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

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

- Note the above contains a mistake…and a couple of other issues too…
Some tuple calculus expressions result in infinite relations.

\[ \{ t \mid \neg t \in r \} \]

\[ \{ t \mid t[A]=5 \lor \text{true} \} \]

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

Such expressions don’t make sense in the context of databases.
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:

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

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

- Consider a \textit{branch} relation that consists of no Brooklyn branches.
  - Every customer is in the result.
  - Even “garbage” values are in the result.
Find the names of all customers who have an account at all branches located in Brooklyn (safe version):

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

Note how this solution eliminates the “garbage” values.
Example Queries

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

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

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

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

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:

\[
\{ < x_1, x_2, \ldots, x_n > | P(x_1, x_2, \ldots, x_n) \}
\]

- \(x_1, x_2, \ldots, x_n\) 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 > | < l, b, a > \in loan \land a > 1200\}

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

  \{< c > | \exists l, b, a (< c, l > \in borrower \land < l, b, a > \in loan \land a > 1200)\}

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

  \{< c, a > | \exists l (< c, l > \in borrower \land \exists b(< l, b, a > \in loan \land b = "Perryridge")\})
  
or \{< c, a > | \exists l (< c, l > \in borrower \land < l, "Perryridge", a > \in loan)\}
Example Queries

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

\[
\{ \langle c \rangle \mid \exists l (\langle c, l \rangle \in borrower \\
\quad \land \exists b,a(\langle l, b, a \rangle \in loan \land b = \text{“Perryridge”}) \\
\quad \lor \exists a(\langle c, a \rangle \in depositor \\
\quad \land \exists b,n(\langle a, b, n \rangle \in account \land b = \text{“Perryridge”}))\}
\]

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

\[
\{ \langle c \rangle \mid \exists s, n (\langle c, s, n \rangle \in customer) \land \\
\forall x,y,z((\langle x, y, z \rangle \in branch \land y = \text{“Brooklyn”}) \Rightarrow \\
\exists a,b(\langle a, x, b \rangle \in account \land \langle c, a \rangle \in depositor))\}
\]
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.