Database Management System (DBMS)

Lecture-20

Dharmendra Kumar

September 10, 2020

Updating

To update value of a particular row into a relation, we write the following type of query:-

$$r \leftarrow \Pi_{F_1, F_2, \ldots, F_n}(r)$$

where each F_i is either the i^{th} attribute of r, if the i^{th} attribute is not updated, or, if the attribute is to be updated, F_i is an expression, involving only constants and the attributes of r, that gives the new value for the attribute.

If we want to select some tuples from r and to update only them, we can use the following expression; here, P denotes the selection condition that chooses which tuples to update:

$$r \leftarrow \Pi_{F_1, F_2, \ldots, F_n}(\sigma_P(r)) \cup (r - \sigma_P(r))$$

Relational Algebra

Example: Suppose that interest payments are being made, and that all balances are to be increased by 5 percent.
Solution: $account \leftarrow \Pi_{account-number, branch-name, balance*1.05}(account)$

Example: Suppose that accounts with balances over \$10,000 receive 6 percent interest, whereas all others receive 5 percent.
Solution: $account \leftarrow \Pi_{account-number, branch-name, balance*1.06}(\sigma_{balance>10000}$ $(\text{account})) \cup \Pi_{account-number, branch-name, balance*1.05}(\sigma_{balance \leq 10000}(account))$

Views

Any relation that is not part of the logical model, but is made visible to a user as a virtual relation, is called a view.

We define a view by using the create view statement. To define a view, we must give the view a name, and must state the query that computes the view. The form of the create view statement is

create view v as $<$ *query expression* $>$ where $<$ *query expression* $>$is any legal relational-algebra query expression. The view name is represented by v.

Example: Consider the view consisting of branches and their customers. We wish this view to be called all-customer. We define this view as follows:

create view all-customer as $\Pi_{branch-name,customer-name}$ (depositor $\bowtie$ *account*) $\cup$ $\Pi_{branch-name,customer-name}$ (borrower $\bowtie$ *loan*)

Once we have defined a view, we can use the view name to refer to the virtual relation that the view generates. Using the view all-customer, we can find all customers of the Perryridge branch by writing

$$\Pi_{customer-name}(\sigma_{branch-name="Perryridge"}(all-customer))$$

Tuple Relational Calculus

A query in the tuple relational calculus is expressed as

t — P(t)

that is, it is the set of all tuples t such that predicate P is true for t. We use t[A] to denote the value of tuple t on attribute A, and we use t $\in$ r to denote that tuple t is in relation r.

Example Queries

- Find the branch-name, loan-number, and amount for loans of over $1200.
  Solution: $\{\ t \mid t \in loan \wedge t[amount] > 1200\ \}$
- Find the loan number for each loan of an amount greater than $1200.
  Solution: $\{\ t \mid \exists\ s \in loan(t[loan - number] = s[loan - number] \wedge s[amount] > 1200)\ \}$
- Find the names of all customers who have a loan from the Perryridge branch.
  Solution: $\{\ t \mid \exists\ s \in borrower(t[customer - name] = s[customer - name] \wedge \exists\ s \in loan(u[loan - number] = s[loan - number] \wedge u[branch - name] = "Perryridge"))\}$

Tuple Relational Calculus

- Find all customers who have a loan, an account, or both at the bank.

  Solution:

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

- Find those customers who have both an account and a loan at the bank.

  Solution:

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

- Find all customers who have an account at the bank but do not have a loan from the bank.

  Solution:

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

Tuple Relational Calculus

- Find all customers who have an account at all branches located in Brooklyn.

  Solution:

  $\{\ t \mid \forall u \in branch(u[branch - city] = ''Brooklyn'' \Rightarrow \exists s \in depositor(t[customer - name] = s[customer - name] \land \exists w \in account(w[account - number] = s[account - number] \land w[branch - name] = u[branch - name])))\}$

Note:

1. The formula P $\Rightarrow$ Q means "P implies Q"; that is, "if P is true, then Q must be true."
2. Note that P $\Rightarrow$ Q is logically equivalent to P $\lor$ Q.