Chapter 10

Views

Topics in this Chapter

- What are Views For?
- View Retrievals
- View Updates
- Snapshots
- SQL Facilities

Views

- A view is created by applying an expression of relational algebra to a base relvar
- A view is a derived or virtual relvar
- Updates applied to the base relvar are immediately visible to the view
- Updates to values in the view “really” only update the base relvar by reiterating the original algebraic expression

What are Views For?

- A view can make subsequent queries much simpler (analogous to a macro)
- Views allow the same data to be seen by different users in different ways at the same time
- Views provide automatic security for hidden data
- Views can promote logical data independence

Logical Data Indendence

- Logical data independence means the immunity of users and user programs to changes in the logical structure of the database
- Growth aspect: New attributes or relvars can be added without affecting existing user programs
- Restructuring aspect: If a base relvar is split, the “old” structure can be emulated by a view; the “old” programs can now access the view

Base Relvars and Views

- The Principle of Interchangeability (of Base and Derived Relvars):

  There are no arbitrary and unnecessary distinctions between base and derived relvars
- The Principle of Database Relativity:

  From a user's point of view, base relvars and views are logically equivalent

View Retrievals

- A relational expression is a relation-valued function
- It takes in relation(s) as arguments, and returns a relation
- A view is created using a relational expression, in which the relation returned is the view

View Updates

- A view is a relvar, and so is updateable by definition
- Earlier discussions of view updates tended to limit them
- It is in principle not necessary to limit them, apart from constraint enforcement
- View updates must adhere to the Golden Rule: no relvar must ever be allowed to violate its own predicate

A View Updating Mechanism

- View updates must apply to appropriate underlying persistent relvars
- INSERT and DELETE should be inverses of each other
- UPDATE can be treated as a DELETE and an INSERT, so long as no intermediate actions or checks permitted
- View updates should be compatible with all relational operators

A View Updating Mechanism – UNION

- Let view UV be defined as follows:
- VAR UV VIEW
  (S WHERE STATUS > 25) UNION
  (S WHERE CITY = ‘Paris’) ;
- Insert the following tuple into the view:
- (S6, Smith, 50, Rome)
- This will insert the tuple into the suppliers base relvar, and thus appear in the view
- Similar effect for DELETE and UPDATE

A View Updating Mechanism – INTERSECT

- If both sides of the intersect evaluate to true, the operation applies to the base relvars successfully
- Analogous to the effect in the case of a Union, except that both conditions must be satisfied
- For example, the status must be > 25, and the city must be Paris

A View Updating Mechanism – Difference

- A MINUS B
- The predicate is (PA) AND NOT (PB);
- The new tuple must satisfy PA, and not PB, and if so, it is inserted into A, or deleted from it, depending on which is invoked
- On an update the result must satisfy PA and not PB, and then the old version is deleted from A, and the updated version is inserted into A

A View Updating Mechanism – Restrict

- A where p, with a predicate of PA
- The predicate for V is (PA) AND p
- If the new tuple is presented to the view based on the RESTRICT, and it satisfies PA and p, it is inserted, deleted, or updated, as appropriate

A View Updating Mechanism – Project

- Let the attributes of A, with predicate PA, be partitioned into two disjoint groups, X and Y
- Consider X and Y as single composite attributes
- For INSERT, Y must have an acceptable default value to be used when X is satisfied (or *vice versa*)
- For DELETE, all tuples satisfied by X are deleted, regardless of the value of Y
- For UPDATE, prior Y values are retained

A View Updating Mechanism – Join

- To update a tuple of a view based on a join, it may be necessary to affect other tuples to preserve the view's predicate
- Earlier writers concluded that it was impermissible to update such views
- Mr. Date believes a better approach is to follow through the ancillary implications, rather than to avoid such updates
- Joins can be one to one, one to many, or many to many

A View Updating Mechanism – Join

- Let J = A JOIN B, where A, B, and J have headings {X,Y}, {Y,Z} AND {X,Y,Z}
- The predicates are PA, PB, PJ, where PJ for J is PA (a) AND PB (b), where a is the A portion and b is the B portion
- For INSERT, the new tuple j must satisfy PJ. If the A portion of j does not appear in A, it is inserted, and likewise for the B portion in B
- Similar for DELETE and UPDATE

A View Updating Mechanism – Join – One to One

- A one to one join is more accurately called (zero-or-one) to (zero-or-one)
- For example, a separate supplier relvar whose only other attribute is favorite restaurant, and not all suppliers have one
- The view that joins supplier to suprest can permit inserts, deletes and updates that affect the underlying relvars in appropriate ways

A View Updating Mechanism – Join – One to Many

- A one to many join is more accurately called (zero-or-one) to (zero-or-more)
- Operations on the view will affect the underlying relvars appropriately
- A tuple inserted into such a view will insert a tuple into the underlying relvar on the one side, as well as a tuple into the underlying relvar on the many side
- DELETE will occur based on constraints on underlying relvars

A View Updating Mechanism – Join – Many to Many

- A many to many join is more accurately called (zero-or-more) to (zero-or-more)
- Operations on the view will affect the underlying relvars appropriately
- A tuple inserted into such a view will insert a tuple into the underlying relvars
- DELETE will occur based on constraints on underlying relvars
- Base relvars and views retain their predicates

Snapshots

- Snapshots are derived relvars, but they are not views
- Snapshots are the result of a relational operation that generates a read only relvar that is refreshed periodically
- An important case of controlled redundancy, used often for point in time reporting
- Snapshots have come to be misnamed “materialized views,” an oxymoron

SQL Facilities

- CREATE VIEW <view name> AS
  <table exp>
  [WITH [<qualifier>] CHECK OPTION] ;
- WITH CHECK OPTION is used for INSERTs and SELECTs: will make them fail if integrity constraints are violated; the default is that they will succeed regardless
- Can only be used with updateable views, as defined in SQL, which is narrower

SQL Facilities – Updateable Views in SQL

- The standard is more complex than any implementation of it; simplified, we say:
- Views defined as a restriction and/or projection of a single base table
- Views defined as a one to one join
- The many side of a one to many join
- UNION ALL or INTERSECT
- Other restrictions may apply