Views and Indexes

Controlling Concurrent Behavior Virtual and Materialized Views Speeding Accesses to Data

Lecture notes by Prof Jeffrey Ullman of Stanford Revised by Xiannong Meng for use at Bucknell

Views

- A view is a relation defined in terms of stored tables (called base tables) and other views.
- Two kinds:
 - 1. Virtual = not stored in the database; just a query for constructing the relation.
 - Materialized = actually constructed and stored.

2

Declaring Views

- ◆ Declare by: CREATE [MATERIALIZED] VIEW <name> AS <query>;
- ◆Default is virtual.

Example: View Definition

 CanBuy(store, sneaker) is a view "containing" the store-sneaker pairs such that the buyers will be able to purchase a sneaker brand from a store:

CREATE VIEW CanBuy AS

SELECT store, sneaker

FROM Sneakers, Sells

WHERE Sneakers.name = Sells.sneaker;

4

Example: Accessing a View

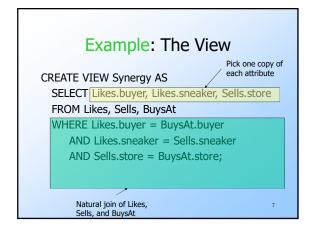
- Query a view as if it were a base table.
- Also: a limited ability to modify views if it makes sense as a modification of one underlying base table.
- ◆Example query:

SELECT sneaker FROM CanBuy
WHERE store = 'Danville';

5

Triggers on Views

- Generally, it is impossible to modify a virtual view, because it doesn't exist.
- But an INSTEAD OF trigger lets us interpret view modifications in a way that makes sense.
- Example: View Synergy has (buyer, sneaker, store) triples such that the store sells the sneaker, the buyer goes to the store and likes the sneaker.



Interpreting a View Insertion

- We cannot insert into Synergy --- it is a virtual view.
- But we can use an INSTEAD OF trigger to turn a (buyer, sneaker, store) triple into three insertions of projected pairs, one for each of Likes, Sells, and BuysAt.
 - Sells.price will have to be NULL.

8

The Trigger

CREATE TRIGGER ViewTrig
INSTEAD OF INSERT ON Synergy
FOR EACH ROW
BEGIN

INSERT INTO LIKES VALUES(new.buyer, new.sneaker); INSERT INTO SELLS(store, sneaker) VALUES(new.buyer, new.sneaker);

INSERT INTO BUYSAT VALUES(new.buyer, new.store); FND:

9

Materialized Views

- Problem: each time a base table changes, the materialized view may change.
 - Cannot afford to recompute the view with each change.
- Solution: Periodic reconstruction of the materialized view, which is otherwise "out of date."

10

Example: Axess/Class Mailing List

- ◆The class mailing list cs145-aut0708students is in effect a materialized view of the class enrollment in Axess.
- Actually updated four times/day.
 - You can enroll and miss an email sent out after you enroll.

1

Example: A Data Warehouse

- Wal-Mart stores every sale at every store in a database.
- Overnight, the sales for the day are used to update a data warehouse = materialized views of the sales.
- The warehouse is used by analysts to predict trends and move goods to where they are selling best.

Indexes

- ◆ Index = data structure used to speed access to tuples of a relation, given values of one or more attributes.
- Could be a hash table, but in a DBMS it is always a balanced search tree with giant nodes (a full disk page) called a B-tree.

13

Declaring Indexes

- ♦ No standard!
- ◆Typical syntax:

CREATE INDEX SneakerInd ON
 Sneakers(manf);
CREATE INDEX SellInd ON
 Sells(store, sneaker);

14

Using Indexes

- Given a value ν, the index takes us to only those tuples that have ν in the attribute(s) of the index.
- ◆ Example: use SneakerInd and SellInd to find the prices of sneakers manufactured by M One's and sold by Joe's. (next slide)

15

Using Indexes --- (2)

SELECT price FROM Sneakers, Sells
WHERE manf = 'M One' AND
 Sneakers.name = Sells.sneaker
 AND
 Sells.store = 'Joe''s Store';

- 1. Use SneakerInd to get all the sneakers made by M One.
- 2. Then use SellInd to get prices of those sneakers, with store = 'Joe''s Store'

16

Database Tuning

- A major problem in making a database run fast is deciding which indexes to create.
- Pro: An index speeds up queries that can use it.
- Con: An index slows down all modifications on its relation because the index must be modified too.

17

Example: Tuning

- Suppose the only things we did with our sneakers database was:
 - 1. Insert new facts into a relation (10%).
 - 2. Find the price of a given sneaker at a given store (90%).
- Then SellInd on Sells(store, sneaker) would be wonderful, but SneakerInd on Sneaker(manf) would be harmful.

Tuning Advisors

- A major research thrust.
 - Because hand tuning is so hard.
- An advisor gets a query load, e.g.,:
 - 1. Choose random queries from the history of queries run on the database, or
 - 2. Designer provides a sample workload.

19

Tuning Advisors --- (2)

- The advisor generates candidate indexes and evaluates each on the workload.
 - ▶ Feed each sample query to the query optimizer, which assumes only this one index is available.
 - Measure the improvement/degradation in the average running time of the queries.