Views and View Management

Olaf Hartig

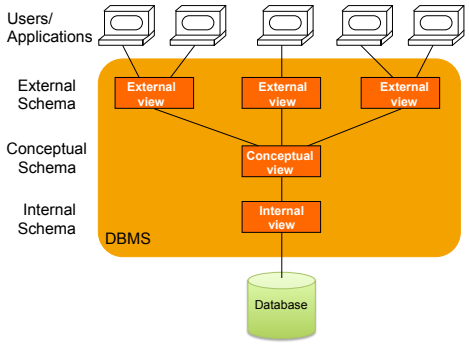
David R. Cheriton School of Computer Science
University of Waterloo

CS 640
Principles of Database Management and Use
Winter 2013

These slides are based on slide sets provided by M. T. Özsu and by R. Ramakrishnan and J. Gehrke

Notes

Recall the Three Levels of Abstraction



Notes

Views

Definition (View)

A *view* is a relation in the external schema whose instance is determined by instances of relations in the conceptual schema.

A view has many of the same properties as a base relation in the conceptual schema:

- its schema information appears in the database schema
- access controls can be applied to it
- other views can be defined in terms of it

Notes

Types of Views

Virtual view: Only the definition is stored

- evaluating statements concerning a virtual view means rewriting those statements based on the view definition

Materialized view: Instance is stored in the DBS

- i.e. corresponding objects exist on the physical level

Notes

Creating and Deleting Views using SQL

- General form:

```
create [materialized] view <name> as <query>

drop view <name>
```

- Example:

```
create view ResearchProjects as
  ( select projno, projname, lastname as empname
    from project, employee
    where respemp = empid and type = 'Research' )
```


Notes

Querying Views

Query a view as if it were a base relation.

```
select projname
from ResearchProjects
where empname = 'Smith'
```

What happens when you query a virtual view?

Notes

Recall: Query Processing Steps (View Resolution)

Parsing
↓
Query Validation
↓
View Resolution
↓
Optimization
↓
Plan Compilation
↓
Execution

- Substitute references to views by the view definitions

As given in:
Goetz Graefe: Query Evaluation Techniques for Large Databases. *ACM Comp. Surveys 25*(2): 73–170 (1993).

Notes

Querying Views (cont'd)

```
select projname
from ResearchProjects
where empname = 'Smith'
```

Substitution:

```
select projname
from ( select projno, projname, lastname as empname
       from project, employee
       where respemp = eid and type = 'Research' )
where empname = 'Smith'
```

Rewrite to:

```
select projname
from project, employee
where respemp = eid and type = 'Research'
      and lastname = 'Smith'
```


Notes

Updating a Database via Views

- Modifications to a view's instance must be propagated back to instances of relations in conceptual schema.
- Some views cannot be updated unambiguously.

Conceptual Schema

Persons

NAME	CITIZENSHIP
Ed	Canadian
Dave	Canadian
Wes	American

NationalPastimes

CITIZENSHIP	PASTIME
Canadian	Hockey
Canadian	Curling
American	Hockey
American	Baseball

⇔

External Schema

PersonalPastimes

NAME	PASTIME
Ed	Hockey
Ed	Curling
Dave	Hockey
Dave	Curling
Wes	Hockey
Wes	Baseball

1. What does it mean to insert (Darryl, Hockey)?
2. What does it mean to delete (Dave, Curling)?

Notes

View Updates in SQL

According to SQL-92, a view is updatable only if its definition satisfies a variety of conditions:

- The query references exactly one table
- The query only outputs simple attributes (no expressions)
- There is no grouping/aggregation/distinct
- There are no nested queries
- There are no set operations

These rules are more restrictive than necessary.

Notes

Motivation for Materializing Views

- Recall our example query:

```
select projname
from ResearchProjects
where empname = 'Smith'
```

- and the rewritten version:

```
select projname
from project, employee
where respemp = eid and type = 'Research'
      and lastname = 'Smith'
```

- Suppose we frequently ask this type of query, varying employees.
- Then, materializing the view might be a good idea.
- Additionally creating a clustered index on (the view's) attribute `empname` might be even better.

Notes

Motivation for Materializing Views (cont'd)

Advantage

Using materialized views may improve query execution performance significantly (in particular for views with a complex definition).

Downside

Need to *maintain* the view as the underlying base relations change.

Notes

Issues Related to Using Materialized Views

View design: What views should we materialize, and what indexes should we build on those views?

View exploitation: Given a query and a set of materialized views, can we use the materialized views to answer (parts of) the query more efficiently?

- Different uses may be possible (another dimension for the optimizer)

View maintenance: How and when to refresh the materialized views?

Notes

Maintenance Policies

Immediate: Do the refresh as part of the transaction that modifies the underlying base relations.

- Pro: Materialized view is always consistent
- Con: Updates are slowed

Deferred: Do the refresh some time later, in a separate transaction.

- Pro: Can scale to maintain many views without slowing updates
- Con: View may be out of date

Lazy Delay refresh until next query on view; then refresh before answering the query.

Periodic Refresh periodically. Widely used, especially for asynchronous replication in distributed databases, and for warehouse applications.

Event-based E.g., after a fixed number of updates to underlying base relations.

Notes

Summary and Outlook

- Views are relations in the external schema whose instances are determined by relations in the conceptual schema
- Types: virtual views, materialized views
- Updating a database via views may be ambiguous
  - First topic for our discussion next week
    A. M. Keller: The Role of Semantics in Translating View Updates. *IEEE Computer 19*(1): 63-73 (1986).
- Materialized views may speed up query execution significantly
- Issues related to using materialized views:
  - View design
  - View maintenance
  - View exploitation ← second topic next week
    J.Goldstein and P.Larson: Optimizing Queries Using Materialized Views: A Practical, Scalable Solution. *SIGMOD*, 2001.

Notes