8-15-1997

Supporting End-Users' Non-Consistent Views for Decision Support Applications

David Chao
San Francisco State University, dchao@sfsu.edu

Robert C. Nickerson
San Francisco State University, rnick@sfsu.edu

Supporting End-Users' Non-Consistent Views for Decision Support Applications

David Chao
College of Business
San Francisco State University
San Francisco, CA 94132
E-mail: dchao@sfsu.edu
Robert C. Nickerson
College of Business
San Francisco State University
San Francisco, CA 94132
E-mail: rnick@sfsu.edu

Abstract

Database views typically are maintained to be consistent with the database at a specific point in time. There are applications, however, where users may prefer or require views which are not consistent with the database. Supporting non-consistent views provides better information for end-users in a decision support environment such as data warehousing. This paper examines the characteristics of non-consistent views and investigates their management.

1. Introduction

Supporting an end-user's information needs for decision making is a major objective of data warehouses. In a conventional database environment, database views have been used to satisfy these needs. A database view is the part of a database in which a particular user is interested. It can be defined as any legal query operation over single or multiple base tables to meet a user's needs. Typically, views are maintained to be consistent with the database, reflecting the state of the database either at the current time or at a specific point in time. Because views are derived from the database, inconsistency is generally considered to be an error.

Conventional databases model an organization as it changes dynamically with a *snapshot* at a particular time (Snodgrass and Ahn, 1986). The state of a database changes when an update is committed; its past state is not remembered. A user's view of an organization is much broader than what the database can represent. Rather than looking at a specific point in time, a user's view of an organization may include the transitions from the past to the present. Trend analysis in many decision support applications is an example. The data needed in such analysis cannot be provided by a consistent view because it requires historical data. For this and other reasons discussed below, views requested by a user often are not consistent with the database.

Figure 1 gives an example with two base tables and a join view used to produce a report showing the sales order count by salesperson. When a salesperson changes title, such as when E2 is promoted to manager, the salesperson becomes disqualified for the view. Then, if the view is maintained as a consistent view, sales records related to that salesperson are no longer included in the view. Sometimes, however, it is necessary to keep those records in the view so as to have a complete and accurate report. Such a view is not consistent with the database state because it includes records that are not in the consistent view.

Non-consistent views usually require support from customized programs. By supporting only consistent views, database management systems do not serve users adequately. Since a data warehouse is designed to support a user's decision making, it is appropriate to support a user's non-consistent views in a data

warehouse. This paper examines the various kinds of non-consistent views and investigates their management.

2. Analysis of Non-Consistent Views

A consistent view, CV, is the realization of the function CV(A, C, T, D(T)) where A is the sequence of relational algebra operations needed to select the view, C is the set of logical conditions used with A, T is a point in time called the consistent point, and D(T) is the database state at T. Base table records satisfying the condition C are said to be qualified to the view. A consistent view reflects all the database updates qualified to the view up to the consistent point. Treating a modification as the deletion of an old record followed by the insertion of a new record (Hanson, 1987), there are basically two kinds of database updates: insertion and deletion. In maintaining a consistent view, a view deletion occurs when its matching record in the base table is either deleted or becomes unqualified for the view due to modification, and a view insertion occurs when a new record is inserted into the base table or an old base table record becomes qualified for the view due to modification (Chao, Diehr & Saharia, 1996).

A non-consistent view, NCV, is the realization of the function NCV(A, C, T, D(T), U) where A, C, T, and D(T) are as before, and U is the user's requirement for including and excluding records in the view. Typically, a view may become non-consistent if a user purposely excludes from the NCV some consistent view records or ignores in the NCV some updates applied to the consistent view. Since there are only two kinds of updates, those ignored records could be records that otherwise would be deleted or inserted. Hence a non-consistent view is a consistent view with user-identified records included or excluded; it is not a *random* view because its inconsistency is controlled and managed.

Algebraically, NCV = CV + UI - UE where UI (user include) is a set of records that are supposed to be deleted from the view but that the user wishes to include, and UE (user exclude) is a set of records that are supposed to be included in the view but that the user wishes to exclude. Since UE is a subset of CV and UI is a subset of NCV, UI = NCV - CV and UE = CV - NCV.

We call the time T associated with a non-consistent view the control point. UI will be non-null if a user keeps certain view deletions. UE will be non-null if a user rejects certain view insertions. Note that if both UI and UE are null, the view is consistent. Therefore, there are three possible cases for a non-consistent view: (1) UI is non-null and UE is null, (2) UI is null and UE is non-null, (3) UI and UE are both non-null. There are many reasons for users to request a non-consistent view. We give an example of a reason for each case.

UI is non-null and UE is null. In this case the user wants to include some view deletions while accepting all view insertions. Hence, CV is a subset of NCV. As discussed earlier, a user's view of an organization may include transitions from the past to the present and, consequently, may include historical information. There are many kinds of historical information, such as a record's update history, outdated earlier views kept for analysis, etc. Historical information cannot be found in a consistent view. Views that contain historical information are non-consistent because they include information which no longer exists in the database. UI in this case is a set of historical information.

UI is null and UE is non-null. In this case the user does not want to include any view deletions and wants to exclude some, but not all, view insertions. Hence, NCV is a subset of CV. Some applications do not include all qualified records in the view. Statistical sampling provides an example: Typically, only a small fraction of a large qualified population is chosen. If a view contains the selected sample, then UE contains those qualified but not selected records.

UI is non-null and UE is non-null. In this case the user wants to include some view deletions and exclude some view insertions. Very often, not-up-to-date but "good enough" information is sufficient for some users' applications. Hence, for records already in the view, a user may choose not to update them even

though they may be continuously updated in the database. Thus, when a view record is updated in the database and not updated in the view, the old record is part of UI and the new record is part of UE.

3. Continuing Research in Managing Non-Consistent Views

Defining and maintaining non-consistent views are the two major tasks in managing non-consistent views. To define a non-consistent view, a database management system must provide a means for users to specify requirements for including and excluding records in the view. Typically, a non-consistent view must be materialized because of its inconsistency. Hence it can be refreshed (updated) automatically by the database management system or, alternatively, upon a user's request. We propose using an SQL-like syntax to define a non-consistent view. A tentative scheme is given below:

CREATE NON-CONSISTENT VIEW viewname

AS any legal query

AS OF control point

Non-consistent clause

REFRESH [AUTOMATIC|ON DEMAND]

The AS OF clause lets a user specify the control point. Without the AS OF clause, the control point defaults to the current time. The non-consistent clause lets a user specify requirements for including and excluding records in the view. The REFRESH clause specifies whether the view maintenance is automatic or upon a user's request. To maintain a non-consistent view, the maintenance scheme must be able to identify view updates, and then generate UI and UE according to the non-consistent clause. We are continuing our research in identifying typical non-consistent views and ways of expressing inconsistency, and in developing a maintenance scheme.

4. Conclusion

Non-consistent views incorporate a user's requirements to define views and hence to fit a user's information needs. In a decision support environment such as a data warehouse, non-consistent views can be a useful tool to support end-users. We continue our research in managing non-consistent views.

References

Chao, D., Diehr, G., and Saharia, A. (1996). "Maintaining Join-based Remote Snapshots Using Relevant Logging," *Proceedings of the Workshop on Materialized Views*, ACM-SIGMOD Conference, June 1996.

Hanson , E. N. (1987). "A Performance Analysis of View Materialization Strategies," *Proceedings of the 1987*

ACM-SIGMOD Conference, pp. 440-453.

Snodgrass, R., and Ahn, I. (1986). "Temporal Databases," *IEEE Computer*, Sept. 1986, pp. 35-42.

Base table 1: EMPLOYEE(EMPID, NAME, TITLE, SEX, SALARY, DEPT).

Base table 2: ORDER(ORD#, DATE, EMPID, CUSTID).

Join view:

```
CREATE VIEW order_count AS
 SELECT  NAME, COUNT(ORD#)
 FROM EMPLOYEE, ORDER
 WHERE   EMPLOYEE.EMPID=ORDER.EMPID  AND TITLE='SALES'
 GROUP BY  EMPID;
```

Base tables:

EMPLOYEE

EMPID	NAME	TITLE	...
E1	SMITH	SALES	...
E2	CHEN	SALES	...
E3	BOXER	MGR	...
E4	HART	SALES	...
E5	DUNN	SCTRY	...

ORDER

ORD#	DATE	EMPID	CUSTID
O1	9/1/96	E2	C6
O2	9/1/96	E2	C2
O3	9/1/96	E4	C1
O4	9/1/96	E1	C3
O5	9/2/96	E2	C4
O6	9/2/96	E1	C1
O7	9/2/96	E4	C7
O8	9/2/96	E1	C6
O9	9/3/96	E2	C8
O10	9/3/96	E4	C3

Consistent view given the state of the database:

NAME	COUNT(ORD#)
SMITH	3
CHEN	4
HART	3

New update occurs on the database: Change E2's title to MGR.

Views after the update:

CONSISTENT VIEW:

NAME	COUNT(ORD#)
SMITH	3
HART	3

NON-CONSISTENT VIEW if user chooses to keep the disqualified record:

NAME	COUNT(ORD#)
SMITH	3
CHEN	4
HART	3

Figure 1: Comparison of a consistent and a non-consistent view.