Using Referential Integrity To Easily Define Consistent Subset Replicas

Brad Hammond
Microsoft

Introduction

Microsoft Access 7.0 has a powerful "update anywhere" replication feature. It allows replicas to be refreshed on demand and at scheduled times, and replicas can be disconnected for long periods of time. One of its limitations is that all replicas must contain the complete set of replicated data. Users want to create "partial replicas" that contain only a subset of the data, so that they can have smaller replicas for smaller machines, and so that different customers can have replicas which exclude the data for other customers. The Office 97 version of Access will have a "partial replica" feature that creates and maintains replicas which contain only a subset of the rows. In designing this feature, several alternatives were considered for defining the replicated view:

1. Allow only single table queries to define a subset of a table
2. Allow a combination of single table queries, and "relationship filters" which allow joins on foreign keys to define which rows belong in the subset
3. Allow any view definition, including joins and subqueries, to define the replicated view

Key criteria in determining our choice were efficiencies of incremental replication and how easily users could create useful partial replicas.

I. Allow Only Single Variable Queries

This alternative allows efficient incremental replication, since only rows that have been recently updated/inserted need to be sent to the partial replica, and it can be determined whether a row belongs at a partial replica simply by evaluating the filter predicate. However, since the filter predicate can only refer to columns in the table being filtered, this restriction often requires denormalizing of databases. For example, consider a simple database that has Customer, Order, OrderDetail, and Product tables. In order to create a partial replica containing the Washington customers, it would be necessary to add Customer_state as a column to both the Order and OrderDetail tables. This alternative was rejected as being overly restrictive.

**Proceedings of the 22nd VLDB Conference
Mumbai(Bombay), India, 1996**

II. Single Variable Queries And Relationship Filters

This alternative allows single-variable queries on any table. For tables which contain foreign keys, it also allows the automatic selection of all rows containing foreign keys which correspond to primary keys that belong at the partial replica. The partial replicas enforce the same referential integrity constraints as full replicas, so if there is a relationship for Table A (fkey) that references Table B (unique key), and rows from Table A are in the partial replica, then the rows in Table B with the corresponding unique keys are *automatically* included in the partial replica. However, the converse is not true - the partial replica can contain rows that have a unique key, yet exclude the rows with the foreign keys that reference them. If the rows with foreign keys are to be included at the partial replica, a relationship filter is added for the referencing table.

For example: To create a partial replica consisting of the Washington customers and their related information, the partial replica definition would have a simple predicate " State = 'WA' " on the Customer table, and relationship filters on the Order and OrderDetail tables. There is an Access Wizard that automatically generates these filters, which is easy to use but restricts one's choices. The relationship filters and boolean filters can also be added programmatically using Visual Basic.

This alternative eliminates the need to denormalize data, yet there is still enough restriction on the partial replica definition to allow fairly efficient incremental replication.

III. Allow Any Predicate

Allowing any predicate, including joins and nested subqueries, eliminates the need to denormalize the database. However, it makes incremental replication difficult or impossible. The reason is that any data change may affect a subquery's result, and thus cause rows which have *not changed* recently to now meet the filter criteria. For example, suppose a partial replica is supposed to contain the customers (and related orders, etc.) whose average order is more than $1000. A customer could become a member of the replicated view as the result of an insert or update to an Order Detail row, or the deletion of an Order row. It is easy to think of predicates where the database system would have to examine virtually the entire database to see what belongs at the partial replica after a data change. While this would not be the case for all sets of filter predicates, it would be a very difficult problem to always propagate changes to the partial replica in the most efficient way.