TOWARDS THE SUPPORT OF INTEGRATED VIEWS OF MULTPLE DATABASES AN AGGREGATE SCHEMA FACILITY

SWARTWOUT DONALD

TOWARDS THE SUPPORT OF INTEGRATED VIEWS OF MULTIPLE DATABASES: AN AGGREGATE SCHEMA FACILITY

Donald Swartwout
James P. Fry
Database Systems Research Group
The University of Michigan, Ann Arbor, Michigan

ABSTRACT: Supporting multiple user views of databases is currently an important problem area in database management system development. An interesting facet of this problem arises whenever a user needs an integrated view of several distinct databases. Using traditional database concepts, an aggregate schema facility has been developed to address this problem. The basic functions of an aggregate schema facility are discussed, as well as their implementation in a CODASYL/DBTG-like environment. Interest in an aggregate schema facility grew out of a problem in restructuring large databases. The application of this facility to restructuring is discussed, as well as potential applications to dynamic translation and distributed databases.

KEYWORDS AND PHRASES: Database integration, aggregate schema, database management systems, data definition languages, database restructuring, data translation, dynamic translation, distributed databases.

1.0 INTRODUCTION

Today we find numerous databases implemented under various accessing schemes being utilized by many diverse users. With the installation of more and more database management systems, the trend has been toward databases which are larger (in terms of volume), more complex (in terms of interrelationships among records), and used by a broader spectrum of users. To reduce this complexity, and to some degree improve security, the database administrator often provides a subset of the database for the user to process. In CODASYL terms, this would be a subschema, whereas in IMS terminology it is called a PSB (Program Specification Block) or in ANSI/SPARC vocabulary, an External Schema. Independent from the logical subsetting capabilities DBMS also provide features to enhance database access--CODASYL provides AREA/REALM and indexing mechanisms and IMS provides several access methods--physical databases, secondary dataset groups, and indexing. The problem is that in either case, there is no facility for the direct connection of a user view with an "optimized" access method.

While this general problem of supporting multiple user "windows" is encountered whenever data is shared by a diverse user community, it has appeared in restricted form in the restructuring of large databases. During the development of the Michigan Data Translator (MDT) we found that while some restructuring transformations substantially alter a database, many affect only a few record and set types. In such cases, only a small portion of the total data actually requires processing, while the remainder need only be copied from the original to the translated database.

Our efforts to exploit this fact soon led to a situation which sometimes required one of the MDT's major modules to process two distinct databases as if they were a single database. To facilitate this unusual processing, the Aggregate Schema Facility was developed. It allows the user (in this case a translator module) to view and to access two distinct, but possibly interrelated network databases at the same time, as if they were a single database.

More generally, an aggregate schema facility is any set of data definition and manipulation capabilities which permit the processing of several physically and schematically distinct, possibly interrelated databases as if they were a single database. This single database is referred to as an aggregate schema database; its schema is an aggregate schema. The physically existing databases are referred to as underlying or component databases; their schemas as underlying or component schemas. An aggregate schema facility can represent a considerable relief for a user whose processing requires substantial interfacing of several databases. The user is no longer responsible for keeping track of the current database, selecting the database in which relevant data resides, accounting for naming discrepancies among the databases, etc. In fact, in the MDT implementation, the module which uses the aggregate schema facility is

unaware of the number of underlying databases it is processing.

While similar at the the conceptual level to the IMS logical database concept where several physical hierarchical databases can be integrated into a complex logical structure, the aggregate schema facility differs in the following way. An IMS user may only process/access (in his PSB) a hierarchical subset of the logical database, although the logical database may in fact be a network. The ASF permits full access to the underlying databases. The aggregate schema approach also bears similarity to the CODASYL-DBTG Schema/Subschema facility. While this facility is a true subsetting capability, perhaps the best way to view the aggregate schema facility is a "supersetting capability". An aggregate schema database is an aggregation of subsets of databases. Objects (records, items, sets) in an aggregate network database correspond more or less directly with objects in the underlying databases. An aggregate schema facility is by no means as general as an ANSI/SPARC External/Internal Schema interface, but is a step in this direction.

This paper describes the functions of aggregate schema facilities, and some of the implementation problems they present. The ASF is used as an example throughout. Section 2 discusses the basic tasks of an aggregate schema facility, and Section 3 describes the implementation of the ASF. The application of ASF to the restructuring of large databases is described in Section 4, and Section 5 concludes by discussing some additional applications.

2.0 BASIC TASKS OF AN AGGREGATE SCHEMA FACILITY

In order to achieve functional equivalence between an aggregate schema database and its set of underlying databases, four basic tasks must be performed: i) mapping of aggregate schema names to underlying databases, ii) maintenance of inter-database connections, iii) maintenance of currency for the aggregate database, and iv) protecting the consistency of the aggregate databases. DBTG terminology will be used in the discussion, but analogous tasks exist for other classes of databases. Also, we discuss exclusively aggregate schema facilities built "on top of" an existing DBMS; that is, those which do not alter the existing DBMS functions or implementation. At run time, they act approximately as dispatchers translating the aggregate schema operations into DML calls for the appropriate underlying database. The more complicated problem of incorporating aggregate schema capabilities into existing DBMS functions is not considered here. Finally, implementation strategies used in the Aggregate Schema Facility (ASF) will be identified.

2.1 Name Mapping

Many aggregate schema names differ from the names of the corresponding underlying schema constructs. In fact, if a particular name is used for different objects in two or more of the underlying databases, only one of them can use the common name as its aggregate schema name. Therefore, an aggregate schema facility must provide a mechanism for translating references to aggregate schema names into references to the appropriate underlying schema names.

The implementation details for such a facility are not very complex, and are determined largely by the time at which the binding of aggregate schema names to underlying schema names is performed. Choices for this binding time are at aggregate schema DDL compile time, aggregate database application time, or at each DML call. The latter is unlikely to be advantageous unless applications are expected to issue relatively few DML calls which accomplish large amounts of data transfer. The first two options differ in one important respect: binding at DDL compile requires recompiling the aggregate schema DDL whenever a recompile of one of the underlying database DDLs occurs, but binding at run time does not. In the ASF, binding occurs at DDL compile, since the underlying schemas were expected to be stable enough that the increased flexibility of later binding would not outweigh the increased cost.

Finally, a somewhat less obvious form of name-mapping is necessary. Each DBTG record instance has a name, known as its database key. Database keys generally do not contain information identifying the database in which the record instance resides. As a result, if the aggregate schema system permits the user access to database keys, some sort of database identification must be appended to underlying database keys, when they are passed to the user. Conversely, a database key received from the user must be decoded into an identifier for an underlying database and a database key for a record residing in that database.

2.2 Inter-database Connections

Name mapping is the only major task required of an aggregate schema facility which does not recognize connections among underlying databases. However, among databases which are reasonable candidates for aggregation, there are likely to exist implicit and/or explicit inter-database relationships. In the environment which led to the development of the ASF, such relationships were a necessary feature of the underlying databases (see Section 4). In DBTG environments, inter-database relationships take two natural forms. In the first form, the owner of a set resides in one of the underlying databases, and its member record types in another. In the second form, data items which the aggregate schema user views as the items of a single record type are divided among record types in different underlying databases. Such a record type in an aggregate schema is referred to as a "split" record type and the underlying record types are its "components". In the same spirit, we will refer to a divided set as a "split set".

At the heart of both forms is the necessity for information contained in a record instance in one of the underlying databases to identify a record instance in one of the other underlying databases. Record instances can be identified