THE MANAGEMENT OF DYNAMICALLY DISTRIBUTED DATABASE WINDOWS
(Extended Abstract)

Qiming Chen
National Land Information System
Research Institute of Surveying and Mapping, Beijing, China

Data processing in an engineering environment, such as a CAD system or a Geographic Information System (GIS), requires a significantly different database architecture in buffering and user-interfacing from that developed for conventional business applications, since

- A CAD or GIS transaction typically involves a number of steps and intermediate results. This feature requires a manageable data check-out environment.
- The CAD and GIS systems are characterized by high degree of function distribution as they are facilitated with versatile specially designed workstations, which may be more suitable than the main system for performing certain types of transactions. This feature requires a distributed management over the check-out environment.
- The database objects check-out are just temporary, swappable copies. This feature requires the handling of dynamic data distribution, rather than static distribution.

In order to provide such an engineering application environment in this paper a new approach for managing multiple Database Windows (DBW) is proposed, which may be considered as an issue in between DDB and multicache management, and augments two major types of previous approaches, the database program interface [Ston 84] [Chen 85ab, 86] [Melk 83], which did not support distributed management over the check-out environment; and the statically distributed database (DDB), which did not cover the notion of dynamic data swapping.

A DBW, residing on a workstation connected to the Main database (MDB), is handled as, firstly, a check-out environment, containing objects copied from the MDB, together with the universally quantified constrains on these objects, which provides an extended programming environment for the MDB; secondly, a semi-independent system supported by a local data manager, where data can be manipulated by multiple users; and finally, a buffer of data swapping, not for keeping fixed set of data, but for buffering the required data for the current applications.

A DBW is defined by specifying a query to the selected database objects. An OPEN/ADD request can be made at any workstation, but must be sent by the system to MDB for execution. Managed as a temporary database, a DBW can be queried and updated, while committed updates made to the objects are propagated to the MDB and those DBW's who contain the same objects. The application program interface also includes certain particular language constructs to EXECUTE, SUSPEND, and RESUME a transaction, to FORK a transaction into a hierarchy of sub-transactions for a cooperative task, and to GRANT and REMOVE R and W privileges to specific sub-transactions to use certain objects.

For example, we use the following statement on our GIS to define a DBW named "map", containing certain feature data of a digital map "J-47", as

DEFINE map ON station_a INCLUDE J-47 WHERE feature = "elevation" AND feature = "land_use"

The local management of a DBW is quite different from the traditional database approaches in concurrency and recovery control, as

- Allowing non-serialized sub-transactions for cooperative work at a DBW, through issuing certain commands mentioned above.
- Global state update is made only after the transaction, including its all sub-transactions, has been completed.
- At a DBW, upon failure of satisfying pre-defined constraints or post-conditions, the update effects are not completely erased as traditionally.

These features provide a tolerant environment for long duration, conscious decisions involved engineering tasks.

The global management of a multi-DDW system is characterized by dynamic distribution. Issues must be taken into account include data coherence problem, caused by the existence of multiple access paths to each logic database object, and Missing operation problem, caused by improper ordering of a DROP operation and a transaction at a DBW.

To describe the operational behavior of a multi-DBW system mentioned above we refine the concept of a transaction T, as a 5-tuple(A, PA, IC, U, PU) where

- A is a set of actions, which may cause a temporary inconsistent state, but is undesirable to be followed by an immediate synchronization effort,
- PA is a partial order on A,
- IC is a set of integrity constraints,
- U is a set of post-actions for enforcing the system legality, such as update synchronization, acceptance test, recovery,
- PU is a set of protocols on U.

Considering an extended database stat D~ as the combination of the states of MDB and all its DBW's, a transaction T is viewed as a mapping from one stable extended database state to another, as T: $S(D\sim)\rightarrow S(D\sim)$. Under this sense, a DBW OPEN/ADD or CLOSE/DROP operation must be consider as a transaction, Although it neither changes the primary state of MDB, nor has confliction with another OPEN/ADD or CLOSE/DROP operation, it does cause a transition of the extended state, and may have conflict data set with another (usual) transaction. In fact, the improper interaction of OPEN/ADD and CLOSE/DROP DBW operations to transactions makes the data coherence problem more complex than that in a static multicopy DDB system. For example, suppose the copy of object X residing in DBW_i is identified as X_i, in a multi-DBW environment, such operation sequences asMODIFY(X_i), OPEN/ADD(X_j),... or MODIFY(X_i), CLOSED/DROP(X_i),... are illegal.

We propose a DBW semi-centralized (while also taking into account the dominant role of MDB) update synchronization approach, differing from usual DDB approaches by handling dynamic distribution, in which timestamps are employed and put on transactions (rather than data items), however, viewing DBW OPEN/ADD, CLOSE/DROP operations as transactions, and as interruptions to the system, the timestamps may not be the only factor for ordering.

The policy of handling an OPEN/ADD request A and a transaction T with conflict data set, depends on

(a) if T is beyond its computation (new values of data) phase but at its update synchronization phase, it has higher priority.

(b) if T is within the computation phase, timestamps ordering is the general principle, however, if a user desires to sacrifice consistency for speed, transfering inconsistent data (such as a unfinished design) in read-only mode is allowed.

The policy of handling a CLOSE/DROP request D and a transaction T with conflict data set, depends on

(a) both are issued at a same site, then if D is issued earlier, T is to be canceled, otherwise D must be postponed until T is completed, to ensure the update effects transfer to other sites of the system.

(b) T is propagated from a foreign site, then D always can be committed and T be canceled (only at this site), since the update at this DBW becomes insignificant.

For a usual transaction T initiated at DBWi, the update synchronization is generally based on timestamps ordering principle. The algorithm includes the following steps:

1. Checking environment, and computing new values if the checking is passed.
2. Submitting update request to MDB and each DBW where copies of the objects to be modified reside, receiving echo, and committing/aborting based on conflict checking with other foreign requests.
3. Global updating.

At phase 1, the possible conflict between T and ADD requests is checked at MDB, to determine if T should be postponed; The possible conflict between T and DROP requests submitted to DBWi is also checked, if there is, T is to be canceled. At phase 2 and 3, the possible conflicts between T and DROP requests submitted to MDB and other involved DBW's are checked, to see if the update effects are still necessary over there.

This algorithm, beside handling the dynamic data ADD/DROP to/from DBW's, has as sufficient condition as two phase locking (corresponding to phase 2 and 3) for consistency [Esw 76].

REFERENCES

[Chen 86] Q. Chen, "A Rule-based Object/Task Modelling Approach", Proc. ACM-SIGMOD, 1986.

[Chen 85a] Q. Chen, "Extending the Implementation Scheme of Functional Programming System FP for Supporting the Formal Software Development Methodology", Proc. 8th Software Engineering, 1985.

[Chen 85b] Q. Chen, "Toward A Generalized Data/Action Management: An Approach for Specifying and Implementing Operational Schemes", Proc. 1st Pan Pacific Computer Conference, 1985.

[Esw 76] K. Eswaran, J. Gray, R. Lorie and I. Traiger, "The Notions of Consistency and Predicate Locks in A Database System", CACM Vol.19, No.11, pp.624-633, 1976.

[Melk 83] M. Melkanoff and Q. Chen, "Integrating Action Capabilities into Information Databases", Proc. 2nd Int. Conf. On Databases(ICOD-2), 1983.

[Ston 84] M. Stonbraker and L. Rowe, "Database Portals: A New Application Program Interface", Proc. VLDB, 1984.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.