Evolution of Groupware for Business Applications: A Database Perspective on Lotus Domino/Notes

C. Mohan, R. Barber, S. Watts, A. Somani, M. Zaharioudakis

IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA
{mohan, barber, somani, markos}@almaden.ibm.com, swatts@us.ibm.com
www.almaden.ibm.com/u/mohan/, www.almaden.ibm.com/u/barber/

Abstract

In this paper, we first introduce the database aspects of the groupware product Lotus Domino/Notes and then describe, in some more detail, many of the logging and recovery enhancements that were introduced in R5. We discuss briefly some of the changes that had to be made to the ARIES recovery method to accommodate the unique storage management characteristics of Notes. We also outline some of the on-going logging and locking work in the Dominotes project at the IBM Almaden Research Center.

1. Introduction

Over a decade ago, Iris Associates, now a subsidiary of IBM's Lotus, pioneered the concept of groupware and released the product Lotus Notes in 1989. It was based on a research prototype, called PLATO Notes, which was built by some of the Iris founders while they were students at the University of Illinois in Urbana Champaign (a lengthier description of the product's historical evolution can be found in http://www.notes.net/history.nsf/). Notes provides a feature-rich application development and deployment environment [Moore95]. Over the years, more and more of the functionality that used to be in other products complementary to Notes have been folded into Notes itself (e.g., calendaring, scheduling, high-level workflow process definition capabilities). Notes lets program scripts be defined by users and be stored in the Notes DB. Triggers, which are valuable for implementing workflow applications, are supported via the notion of agents. While Notes was initially designed as a workgroup product for use by a small number of users working collaboratively, subsequently it has been enhanced extensively with functionality and infrastructure improvements, allowing it to be successfully deployed as a platform for business applications in numerous large enterprises. Currently, it has an install base of over 55 million seats. Without relying on a DBMS, Notes does its own persistent storage management. Our aim here is to provide a database (DB) perspective on this product.

Since the time Notes was enabled for the internet a few years ago, the name *Domino* has been used to refer to the server and the name *Notes* to the client. Because the DB functionality supported in the client and the server is almost identical, we use the two names interchangeably.

Proceedings of the 26th VLDB Conference, Cairo, Egypt, September 2000.

2. Semi-Structured Data Management

Since its first release in 1989, long before the topic became fashionable in the DB and web research communities, Lotus Notes had been targeted at the management of semi-structured data. Notes supports the storage and manipulation of documents (*notes*) that contain structured as well as unstructured data (e.g., audio, video). *Views* can be used for the presentation of a subset of the data in the documents of a DB in a structured way. View columns can have collation options associated with them. From the GUI, *forms* can be used to create, view and update documents. The Notes API can be used by programs for performing these and other operations. Document sizes could vary widely. Every document in a Notes DB could potentially be structured differently (e.g., with respect to number and types of fields) compared to every other document in the same DB. Document structure could evolve easily over time. At anytime, existing fields in a document could be deleted or their types could be modified, and new fields could be added. A document can point to another document, in the same or different DB, via a *DocLink*. Parent-child relationships

could exist between documents. In addition, documents could be classified along category hierarchies. Querying of a DB's contents can be done using a fairly high-level query language, although the latter is not as sophisticated as the recently proposed query languages for semi-structured DBs and XML data. Since Notes does not have an RDBMS-style query optimizer, choice of an access path to process a query needs to be made by the user.

3. Storage Architecture

All user data and metadata belonging to a Notes DB is stored in a single file dedicated to that DB. A server or a client can manage any number of DBs. Data is stored on disk in a machine-independent format so that binary copying of a DB file across dissimilar machine architectures (e.g., PC and RISC) does not require any conversions to be performed before the DB becomes accessible on the target system. Because of the unstructured nature of the supported data model, DBs as well as individual documents within them are stored in a completely location independent and self-describing format. Storage management is done differently for structured fields versus multimedia or rich text fields (e.g., attachments). Within a DB (e.g., when an index entry points to a document), a document is identified using a short *NoteID* and across DBs (e.g., for replication purposes) it is identified using a longer *UNID* (Universal Note ID).

Sophisticated (hierarchical and ranked) B^+-trees are used for managing views. The latter are like the indexes or materialized views of RDBMSs. With each view, an expression can be associated to determine which documents in the DB qualify to be included in the view. Unlike RDBMS indexes, Notes views are not maintained synchronously as the underlying documents are updated. Timestamps contained in documents and in tombstones of deleted documents are exploited to efficiently update the views. Not using a log for this purpose poses an interesting problem since the old values of a modified document's fields are not available to compute and remove the old key. This has been resolved by maintaining for each view an inverse NoteID to key mapping.

Full text indexes are managed differently from view indexes and are maintained in files external to a Notes DB file. A single so-called Domain Index can be used to index multiple Notes DBs to allow uniform searching across those DBs.

4. Replication

From its first release, support for replication and disconnected operation has been one of the most significant and innovative features of Notes. The replication mechanism is very flexible with respect to with which server(s) and when to synchronize. With each replica of a DB, an expression can be specified to determine which documents should be included in that replica, thereby supporting selective replication. One can also restrict only a subset of the qualifying documents' fields to be replicated. Initially, concurrent updates to the same document were checked for conflicts at document granularity [KBHOG92]. Subsequent enhancements have made it possible to do conflict checking at field granularity. As in the case of views, document timestamps are relied upon to identify changed documents. Sequence numbers associated with individual fields are used to support the optional field-level conflict checking functionality. Notes DB can also be replicated with PDAs like the Palm Pilot.

5. High Availability

In order to provide high availability in the event of server failures, Domino allows the clustering of a collection of servers for supporting automatic failover. The clustered servers manage replicated DBs that are synchronized more often and differently than in the case of normal replication. The switchover of a client from one server in the cluster to another can be made to happen even if the first server is not responsive enough, thereby providing load balancing functionality.

6. Security

Sophisticated access control features and very early support of RSA public key technology for authentication have been the hallmarks of the product. Field level encryption of documents is also supported. These security features are exploited in business applications, especially when role-based workflows are involved. In the web context, Domino's features can be exploited to dynamically create highly personalized web pages.

7. Heterogeneous Data Access

Through companion products like NotesPump and DECS (Domino Enterprise Connection Services), it is possible to integrate data from Notes and other sources (e.g., RDBMSs, SAP R/3). Notes applications can be written as if all the data comes from a Notes DB itself when in fact some of the data may be dynamically or periodically materialized from other sources. This is one way to integrate backend enterprise data using Notes on the desktop. Domino can be accessed from not only Notes but also web browsers and CORBA clients. Similarly, Notes can be used to access not only Domino but also CORBA, SMTP and POP3 servers.

8. ARIES for Semi-Structured Data

Through the joint efforts of Iris Associates and IBM Almaden's Dominotes project, one of the major features that was introduced in the latest release (R5) of Lotus Domino is a traditional DBMS-style, write-ahead logging-based recovery scheme [Mohan99]. This optional feature can be enabled at the granularity of a DB. At anytime, logging can be turned on or off by an administrator. When logging is on, each Notes API call is implicitly treated as an ACID transaction. Even with this restriction, a single transaction could run for a long time by manipulating multiple documents and/or multiple DBs in a single API call. Since Notes had not been originally designed with log-based recovery in mind, adding this sophisticated technology required significant design work. This is because enhancements had to be made to our ARIES recovery method [MHLPS92] to deal with the fact that storage management in Notes is done in very unconventional ways. We call this version of our recovery method **ARIES/SSD** (ARIES for Semi-Structured Data).

Notes stores persistently in a DB file numerous kinds of data structures - different kinds of hash-based search structures, lists of NoteIDs, B-trees, bit vectors, objects, tables, ... Some of these structures are paginated while others are not. Different page sizes are used by different structures. Over time, these data structures might also be moved around within the file in arbitrary ways. Since some of the data structures might contain attachments like audio, video, etc., internally logging had to be made optional at the data structure level also.

File Caching Some of the recovery complications also come from the fact that Notes relies on file caching being done by the file system of the operating system. In other words, Notes does not provide raw device support. Under certain conditions (e.g., when some metadata is changed), Notes issues a *file sync* call to the operating system to force the file cache contents to be written to disk immediately. This is an expensive operation and with our logging enhancement in R5 we have been able to improve performance by reducing the number of times such a call needs to be issued.

Recover_LSN Tracking Until R5, Notes did not have a full-blown buffer manager (BM). Unlike an RDBMS BM, the Notes BM has to manage variable sized pages in a single buffer pool (BP) since the Notes DB contains structures with many different page sizes. Even with the new BM in place, the non-paginated data structures are managed outside of the buffer pool. This fact, coupled with the existence of the file cache in the operating system that might contain some recently written data means that the *Recover_LSN* information tracked by BM in ARIES for checkpointing and restart redo recovery purposes needs to be supplemented with additional such information relating to the data not in BP. We now have a table in virtual storage which tracks Recover_LSNs for non-paginated structures of a DB which are manipulated outside BP. We also track a global Recover_LSN for the file cache on a per DB basis. This value is computed based on the Recover_LSNs of the recently written pages and other non-paginated structures. File sync calls cause this value to be reset. This resetting has to be done carefully since writes to the file cache may occur while a sync is in progress.

Analysis and Redo Passes Accommodating the storage management characteristics of Notes has required changes to the analysis and redo passes of ARIES. We could not rely on an LSN (Log Sequence Number) field that was created at a certain offset in the DB file continuing to be at that same offset after a while. This is because pages might be migrated (within a DB) or deallocated and later some user data might be stored at that LSN location. For such reasons, in ARIES/SSD, the analysis pass gathers information about space allocations. The latter is used during the redo pass to skip processing some log records whose LSNs, in ARIES, might have been compared with LSNs on corresponding DB pages. Whenever possible, ARIES/SSD does logical logging and LSN-based recovery. Otherwise, it does physical logging and non-LSN-based recovery.

DB Migrations Notes users frequently move or replicate DBs by doing file copying via the operating system. This can cause a logged version of a DB to be overlaid with an older or newer replica of that DB from another system. Attempting to apply the log records to the wrong version of a DB can cause major problems. We track extra information in the DB header to deal with such situations. When we detect that a DB had been migrated from one system to another, we reset the LSN fields in that DB since the logs at the 2 systems may be growing at different rates. In particular, the LSNs being assigned in the new system may be lower than the LSNs already assigned by the old system. For a number of reasons, we did not adopt the solution of [MoNa94] where a similar problem arose because DB pages could migrate from client to client and the logger was at the server but an LSN had to be assigned locally in a client machine, while producing a log record, without communicating with the server.

Backup and Restore Prior to R5, backup and restore of a Notes DB were not supported directly in the product itself. Notes administrators had to rely on file system utilities for accomplishing those functions. Starting with R5, APIs are provided for backup vendors to use to get a transaction-consistent copy of a DB, while still permitting concurrent updates to the DB as the copying is done. This approach is very different from the one implemented in DB2 [MoNa93].

Methods to deal with partial writes to disk have been added [Mohan95a]. Due to lack of time, in R5, we did not enhance the view index manager with support for logging. Just as we exploited the *Nested Top Actions* feature of ARIES extensively in ARIES/IM [Mohan95b, MoLe92], ARIES/LHS [Mohan93] and ARIES for MQSeries [MoDi94], in ARIES/SSD also we have benefited tremendously from it. It has permitted us to improve performance and increase concurrency. By using a single log for logging the changes made to all the DBs managed by a server, we have been able to gain performance advantages even if no single DB encounters significant update activity.

9. Current Work and Conclusions

We are currently enhancing Notes to expose the transaction API calls (Begin, Commit, Rollback) to users, thereby allowing a transaction to span multiple Notes API calls. We are also improving the granularity of locking. This is requiring significant work to be done since the earlier coarse granularity of locking had been taken advantage of in many unobvious ways. The new enhancements are now forcing us to address space reservation problems to handle rollbacks correctly [MoHa94]. We are exploiting the Commit_LSN technique [Mohan90] in a number of places to improve pathlengths. By exploiting some of the logical logging techniques of [MoLe92, Mohan95b], we are in the process of adding logging to the view index manager.

In R5, with the changes made to some of its core storage structures, scalability of the product has been enhanced significantly. Without logging support, recovering from a failure took time that was proportional to the size of an affected DB. Implementing logging-based recovery has enabled restart from a failure to be much faster. Apart from the introduction of such industrial-strength features, Notes, which has been much more than merely a messaging system from its very beginning, is now evolving more and more with knowledge management capabilities also.

Acknowledgements: We would like to thank our past colleagues in the Dominotes project at IBM Almaden Research Center and our partners in Iris Associates. Our joint work gave us deep insights into the internals of the product and led to significant enhancements to its DB infrastructure/functionality.

10. References

[KBHOG92] Kawell, L., Beckhardt, S., Halvorsen, T., Ozzie, R., Greif, I. *Replicated Document Management in a Group Communication System*, In **Groupware: Software for Computer-Supported Cooperative Work**, IEEE Computer Press, 1992.

[MHLPS92] Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., Schwarz, P. *ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging*, **ACM Transactions on Database Systems**, Vol. 17, No. 1, March 1992.

[MoDi94] Mohan, C., Dievendorff, R. *Recent Work on Distributed Commit Protocols, and Recoverable Messaging and Queuing*, **Data Engineering**, Vol. 17, No. 1, March 1994.

[MoHa94] Mohan, C., Haderle, D. *Algorithms for Flexible Space Management in Transaction Systems Supporting Fine-Granularity Locking*, **Proc. 4th International Conference on Extending Database Technology**, Cambridge, March 1994.

[Mohan90] Mohan, C. *Commit_LSN: A Novel and Simple Method for Reducing Locking and Latching in Transaction Processing Systems*, **Proc. 16th International Conference on Very Large Data Bases**, Brisbane, August 1990.

[Mohan93] Mohan, C. *ARIES/LHS: A Concurrency Control and Recovery Method Using Write-Ahead Logging for Linear Hashing with Separators*, **Proc. 9th International Conference on Data Engineering**, Vienna, April 1993.

[Mohan95a] Mohan, C. *Disk Read-Write Optimizations and Data Integrity in Transaction Systems Using Write-Ahead Logging*, **Proc. 11th International Conference on Data Engineering, Taipei, March 1995.**

[Mohan95b] Mohan, C. *Concurrency Control and Recovery Methods for B^+-Tree Indexes: ARIES/KVL and ARIES/IM*, In **Performance of Concurrency Control Mechanisms in Centralized Database Systems**, V. Kumar (Ed.), Prentice Hall, 1995.

[Mohan99] Mohan, C. *Repeating History Beyond ARIES*, **Proc. 25th International Conference on Very Large Data Bases**, Edinburgh, September 1999.

[MoLe92] Mohan, C., Levine, F. *ARIES/IM: An Efficient and High Concurrency Index Management Method Using Write-Ahead Logging*, **Proc. ACM SIGMOD International Conference on Management of Data**, San Diego, June 1992.

[MoNa93] Mohan, C., Narang, I. *An Efficient and Flexible Method for Archiving a Data Base*, **Proc. ACM SIGMOD International Conference on Management of Data**, Washington, D.C., May 1993. A *corrected* version of this paper is available as IBM Research Report RJ9733, IBM Almaden Research Center, March 1993.

[MoNa94] Mohan, C., Narang, I. *ARIES/CSA: A Method for Database Recovery in Client-Server Architectures*, **Proc. ACM SIGMOD International Conference on Management of Data**, Minneapolis, May 1994.

[Moore95] Moore, K. *The Lotus Notes Storage System*, **Proc. ACM SIGMOD International Conference on Management of Data**, San Jose, May 1995.