ISSN: 2277-9655

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Document Storage And Retrieval System Protocols To Access Multimedia Data

R. N. Jugele[*1] and Dr. V. N. Chavan[2]

[*1]Department of Computer Science, Science College, Nagpur, Maharashtra, India

[2]Head, Department of Computer Science, S. K. Porwal College, Kamptee, Maharashtra, India

rn_jugele@yahoo.com

Abstract

Multimedia documents are complex accumulations of content portions that belong to different information types. Each information type needs dedicated tools that allow for generation, modification and presentation. The object oriented concept of modelling data seems to be a preferred one in the field of multimedia documents as can be seen within standards specification and proprietary approaches.

Although it is partly possible to extend relational database technology to satisfy the needs of multimedia document storage. There are systems more suitable as a base for a multimedia database system which support for the representation of the needed information types and the extensibility. RDA or DFR design concepts focuses on specifications and lower layer protocols supporting isochronous transfer of data.

Keywords: DBMS, Multimedia, Document, RDA, Coopeation, Mesa, DFR, Synchronization, Protocols.

Introduction

Document handling systems are the most frequently used computer applications. Their popularity starts from the implied rise in document quality and handling efficiency. The electronic document is the exact equivalent of a traditional paper document. It comprises information represented by text and images structured in a desired fashion.

The recent technological developments are expected to considerably advance the way in which users interact with their systems as well as the way they communicate among each other. Several new approaches to store and retrieve multimedia data their use in traditional DBMS for the storage of multimedia data, the necessary improvements to the existing DBMS and database access protocols is evaluated.

Access Protocols

There are two standards for remote access to document storage and retrieval systems.

i. Remote Database Access (RDA) : For remote DBMS access
ii. Document Filing and Retrieval (DFR) : For document storage and retrieval.

Both used in the interface between the application and the storage system. Following are the requirements of the quality of services :

- Stored data must be accessible as one data unit by a copying-like operation.
- Support for stream-oriented interface for extraction and storage of data units.
- Support for stream oriented interface and continuous media applications.

RDA :

Following are the RDA standard :

- ISO/IEC 9579-1: Generic Model Service and Protocol[23] defines generic RDA operations with generic parameters.
- ISO/IEC 9579-2: SQL Specialization: defines special SQL dependent types for the parameters of the RDA operations and makes some further restrictions.

Following are RDA operation services :

- Execution of remote operations
- Control services.
- Dialog management
- Resource management
- Transaction management

Following are the RDA server queries :

- R-ExecuteDBL (Immediate execution operation): It specify a DBL query which send to the server are executed immediately and send back the result to the client.
- R-DefineDBL/R-InvokeDBL (Stored execution): It specify a DBL query which send to the server and the client can invoke this query several times with different arguments.

There are some minor differences in the concrete specification of the parameters, both methods are common for the structure of the argument and result

parameters but suffer from the same problems with respect to multimedia data i.e. continuous media data.
The RDA standard defines the result of a remote query as a structured data type (ASN.1) which contains a list of result values. The exact meaning and structure of one of these result values is defined in a specialization where as some of these elements contain large pieces of data representing multimedia information.
The first requirement is fulfilled with RDA but there might be restrictions with respect to huge data units which depends on the implementation of RDA and the lower communication modules. Part of this problem can be solved by partitioning the data in smaller pieces which are stored as different units in the DBMS. This solution is not transparent for the application. To get all parts from the DBMS server the application must explicitly invoke several queries. The overall need for temporary buffer space can be lowered with this approach but the complexity of the application, the overhead for additional communication and the resource requirements at the server side will increase. RDA allows only one outstanding request so the response time is affected and batching of several requests to the server is not possible.
RDA is based on the Remote Operation Service Element[26] which provides no bulk data transfer service to the service users. Therefore it allow to support stream-oriented access method for transparent transfer of data in smaller units. The same holds for the real-time transfer of continuous media data too, which could be realized on top of the bulk data transfer service. As there is no isochronous transport protocol in the ISO/OSI hierarchy so the addition of a bulk data transfer service to the RDA and ROSE services is not enough. Both RDA and the underlying communication module protocols don't support the necessary Type of Service parameters i.e. guaranteed bandwidth and maximum delay to provide adequate service for the timely transmission of continuous media data.

DFR

The DFR standard (ISO/IEC 10166)[4] defines a service for the remote storage and retrieval of documents in a client-server-environment. Each document has associated attributes which can be used to support search for documents in the document store or used for version and access control. Beside the information stored in the attributes the server has no knowledge of the contents or internal structure of the stored documents.
One of the defined operations READ allows a client to access the document itself and the values of the attributes. The document is sent as one piece of data from the DFR server to the DFR client and then forwarded to the application.
The DFR access protocol is defined through the specification of data structures and the usage of supporting services like ROSE and ACSE. Although DFR is different from RDA in functionality but same problems with respect to multimedia exist for DFR.

- Stream-oriented bulk data transfer service supports the transfer of large data units in smaller pieces
- Support for isochronous transport service and interfaces for the specification of quality of service parameters needed for the real-time transfer of continuous media data.

Cooperation

In conventional applications measures are taken to allow concurrent access to shared data by multiple users. In the database context, the notion of a transaction was introduced[5],[8]. A transaction of an atomic operation on shared data shows the behavior of a single-user system to all users. As a consequence, cooperation of users cannot be achieved in these systems. In conventional database applications transactions typically run for a short time and access a small amount of data. In this context the transaction abort due to a system crash, deadlock or other failure.
In applications working with complex data objects, long-lasting operations on these objects should not be modeled as conventional transactions, as the abort of a transaction which ran for some time, may result in the loss of work. Even worse, system crashes and deadlocks probably affect long-lived transactions more often than short ones.
To support this sort of applications, new transaction concepts have been developed where transactions are composed of a hierarchy of sub transactions[20]. In these nested transactions the failure parts of the transaction does not have to cause the whole transaction to abort, alternatives to the failed operation can be tried. An extension of nested transactions are multilevel transactions and as a further extension open nested transactions is introduced by Weikum and others[29],[30]. A sub transaction of an open nested transaction may commit and therefore make its results visible to other top-level transactions before its own top-level transaction commits. In open nested transactions the abort of a top-level transaction requires compensation for committed sub transactions.

Requirements

The applications that are investigated for this report under the aspect of cooperation are in the field of joint editing. Examples are Computer Supported Cooperative Work (CSCW) group editors, CAD-CASE systems and publication systems. With data management and cooperation concerned these applications have the following common properties :

- **Data Objects-Large and complex**: It handle large and complex data objects like CAD-objects, publications, multimedia documents and software projects. All these objects consist of components of different media though continuous media and it plays secondary role in most existing systems.
- **Long lasting operations**: All these applications have interactive user interfaces, manipulations of objects last for a considerable amount of time. As a consequence, most applications manipulate data in a read-modify-write-cycle, it means they do not work directly on the data managed by the storage system.
- **Multiuser cooperation**: Multiple users cooperate to design a complex object. In contrast to concurrent use of shared data, cooperating users have a common goal and know about each other's tasks. Therefore they might be interested in viewing and discussing intermediate results of their peers.

Following requirements can be stated for the synchronization and recovery mechanisms that are needed in data management systems supporting cooperation on multimedia data.

- o **Availability of data:** Cooperating users should be able to access shared data as freely as possible. Blocking each other's work must be minimize.
- o **Fast propagation of changes:** In applications, where multiple users work on the same data simultaneously all users should be able to see the changes made by others as soon as possible. This requirement appears in an extreme form in applications, where each participant in a cooperative session can see the same screen image.
- o **Correctness criteria:** Serializability is the traditional correctness criterion of transaction based systems where operations are executed in a way which is equivalent to a serial execution. As this restricts cooperation in a considerable way, cooperative applications must be able to define own correctness criteria which take into account the semantics of the application.
- o **Minimal lost work:** As operations contain a considerable amount of work, synchronization and recovery have to be flexible to preserve as much of this work as possible when a failure occurs.
- o **Partial undo:** As cooperating users can commit single steps of operations on complex objects they also need the possibility to undo these steps one by one. Recovery has to take into account that operations of other users may depend on the steps which are undone. Therefore user interaction may be required to perform correct recovery actions.
- o **Multilevel synchronization:** In a typical cooperative application, three levels of data copies can be identified, which differ in persistence, up-to-date and form of usage:In a typical setting, the persistent data in the storage system are copied to the workspace of the application, where all operations on the data are performed, before they are written back to the storage system. Parts of the data in the application are converted to a human readable form and displayed to the user. The working-copy of the application is the most up-to-date version, but exists only in volatile storage. The information presented to the user is redundant and can always be created from the application workspace. A storage system that has to support cooperative applications using multimedia data allow synchronization on all three levels of data-copies: Persistent data, application workspace and user presentation.

Approaches

Following are the approaches to synchronization in systems.

a. Conservative:

CAD-databases has a concept of a checkout-modify-checkin cycle to manipulate complex data-objects.

- Checkout : a user can reserve a data-object for exclusive access.
- Checkin : After changing the object modifications are available to others.

Here some data may be unavailable to all but one user for a long time which limits cooperation[12],[15],[19],[18].

b. Floor control:

It allow editing of an object by several participants in a cooperative session. The floor holder is a single user who can execute operations of the application at each point of time and can be passed between the participants according to a special floor passing protocol for real cooperation and all other users can only watch his actions. In this way degree of cooperation increases by reducing the size of the objects but most systems use one floor per application[3],[17],[1],[9].

c. Advanced transaction:

New transaction concepts are developed to support requirements imposed on data management by long-lasting and cooperative transactions on complex data-objects, these are based on hierarchically structured transactions. It reflects the structure of the problem it has to solve or the organizational structure of the group of people, who cooperate in the transaction. Sub transaction can commit, abort a parent transaction, compensating transactions are needed for committed sub transactions to enhance data availability before its parent commits. Application-specific rules for synchronization and recovery field of cooperation schemes are defined[2],[14],[22],[13],[21].

d. Flexible locking:

Locking concepts with more flexible semantics have been developed because other users is somehow contradicting the idea of cooperation.

- A tickle lock : It automatically released if the lock-holder does not access the locked data for some time or if someone else wants to access the data[11],[9].
- A notify lock : It inform the holder of a read lock if someone wants to write the locked data[24],[10],[28].
- Probabilistic lock : It established if no conflicts occur[9]. Otherwise the user or application trying to get a lock will be informed and can decide what action to take.

Another method of flexible synchronization allows several users to create different versions of a data item and to resolve conflicts between different version users[9],[22].

e. Social protocols:

It allow total user freedom and have to synchronize themselves by means outside the application. The computer system may or may not provide a channel over which a social protocol can be run. These are described in[6],[16],[7],[27].

Existing Systems

Mesa file system

It enhances availability of shared data and allows fast propagation of changes, correctness, minimization of lost work, partial undo and multilevel synchronization. The interface of the file system contains functions for cooperative lock management and it is developed to allow cooperation between different processes[24]. A process acquires a lock by defining the lock mode and a callback function that will be called by the file system, when another process acquires a conflicting lock. A process which cannot acquire a lock instantly are supplied to the file system, a callback function is called as soon as the previously acquired lock can be granted. A request to release a lock may be accepted, rejected or deferred.

CoDraW

The main focus of it is to achieve as much freedom for a single user's actions as possible. It is a sketching tool which manipulate a common drawing workspace[16] where all participants run an instance of the application which has replicated all data objects that are manipulated in the session. To synchronize the instances of it in a session the operations executed by each participant are broadcast to all instances. For ensuring data integrity a participant must hold a lock on an object before manipulating it.

Here locks are established by communicating with all other participants. If another participant holds a conflicting lock, one or both users are notified of the conflict and may take appropriate actions. Due to full data replication and notify locks data is highly available to all users. Data synchronization is done by a specialized application which ensures correctness criteria, lost work is minimized by user intervention and changes to data are propagated very fast to all participants.

Collaborative Editing System :

Here the concept of an active transaction is useful because problems are encountered when screens of other users is updated while transaction abort and for this purpose user defined actions bound to begin or end of transactions are desirable. A document[11],[9] contain an outline and a set of sections containing text is replicated on all participant's machines and is stored on the machine of its original creator. Users of it display parts of the outline and one or more sections and editing one section at a time. It commit every few keystrokes of an editing user so that other users see a nearly up-to-date version of a section and versions of a section are put on a version stack, so that a user can undo changes step by step.

Data availability is raised by the use of tickle locks and changes are propagated by commit of partial actions. Using tickle locks and version stacks specific correctness criteria, minimal lost work and partial undo are achieved. Data are primarily synchronized[25] on the level of persistent stored data, which induces problems when trying to synchronize screen images.

Cooperative Transaction Model

Cooperative transaction model for design databases was developed at Brown University[22]. In figure 1 top-level transaction containing all design activities for some object, consists of a hierarchy of transaction groups (TG), cooperative transactions (CT) and leaves of the transaction tree. Every transaction represents some part of the design team. Members of a transaction group may be other transaction groups or cooperative transactions. A cooperative transaction is a collection of atomic operations executed by a single designer. Each TG defines correctness criteria for the operations issued by its members. Patterns define sequences of operations for correct execution and conflicts define illegal sequences of operations which reflect the style of work of a design team, legal constraints to the design process and differ from one TG to another.

A TG has a private copy of each object and accessed by one of its members. TG may merge its version with the version of its parent TG by performing a checkpoint and TG supply an equivalent summary of the operations to its parent. The parent TG will check those checkpointed operations against its own patterns and conflicts. According to patterns and conflicts TG kept a log that contains information about all committed operations, patterns and conflicts they are involved in for recovery. For each recovery action, patterns and conflicts have to be rechecked and humans may be involved to decide which operations really must be undone. If it undone, operations of the same or other CTs become invalid.

TGs keep local versions and operations are atomic due to this data availability is enhanced. Data propagated fast between CTs of a TG, but no automatic notification of changes is done. The main goals of the model is application specific correctness criteria, minimal lost work and partial undo where synchronization is only done on the database level. A transaction model proposed at Brown University adds the notion of active transactions, triggering user defined actions on begin, commit or abort of transaction[13]. This model support for notification of data changes.

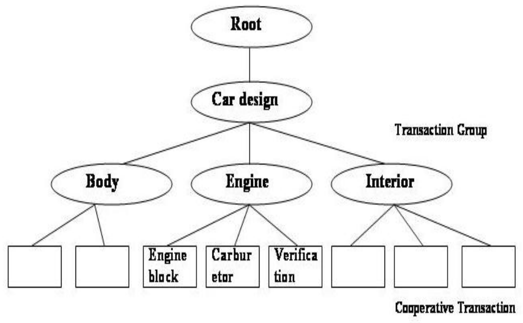

Fig. 1: Hierarchical design transaction (Source[22])

GMD-IPSI-Publication environment

Open publication environment proposed for multimedia documents by GMD-IPSI[21]. Multimedia data are modeled as objects it has fixed set of operations components called tools. To add new object classes to the environment the appropriate tools have to be provided. The principal components of the environment are shown in figure 2. User accesses the environment through assistant, it differ according to the role plays in the publication process, it contains one or more task experts to utilize a set of tools. If a user wants some task to perform, it requires this task from its assistant and if assistant has an appropriate task expert, this will access the right tool otherwise the assistant may ask an assistant of some other user. This assistant may know how to do the task it may ask another assistant or its own user. In this way users of different knowledge and in different roles can cooperate via their assistants. Assistant generate a software component called agent, which has the basic functionality of one of the assistant's task experts. An agent can be exported to another user, whose assistant then can access the corresponding tool directly.

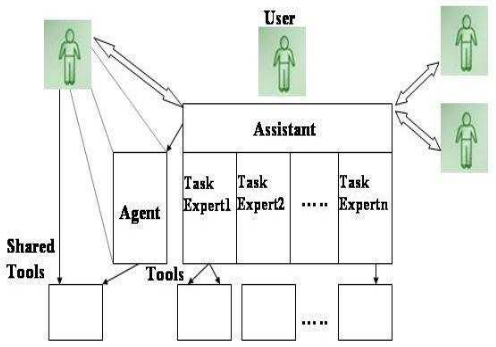

Fig. 2: Application structure of the publication environment

Interaction between assistants, task experts, agents and tools is structured according to the transaction concept of open nested transactions[30], which is a generalization of multilevel transactions introduced by Weikum and others in[29]. In the publication environment, each operation of one software component on another is modeled as a subtransaction. On any level of the transaction tree, application specific correctness criteria can be specified by defining conditions under which two operations are commutative. This can be done on the object level, as each object class has a fixed set of operations. High data availability and application specific correctness criteria are achieved by defining commutative between operation. Lost work can be minimized and partial undo is possible because compensating operations exist for all operations. Synchronization done on the persistent data by the transaction mechanism to convey changes of data to other users.

Discussion

Some approaches allow user cooperation but data integrity is a major claim and data synchronization is done on persistent data, it is costly and time-consuming operations. Following are ad-hoc character of cooperative applications :

- Synchronization is done on applications workspaces by a broad spectrum of mechanisms from floor control to chaotic interaction controlled by social protocols and the replicas are brought up-to-date by broadcasting the user input.

- Inputs are synchronized by a floor passing mechanism and it is done on the displayed data. The application and data are centralized, broadcasting output to all participants.

Limitations

The access protocols are suffers from the following problems:

- The request-response technique allows only the access to data units as a whole, which is not suitable for continuous media data or large discrete data.
- It does not support real-time transfer of continuous media data.

Conclusion

Based on RDA/DFR access to remote MMDBMS provides only limited functionality and are partially capable of supporting the copying-like access method sufficient for some multimedia applications. Stream-oriented and real-time access provided through massive extensions to these protocols and requires extensions for isochronous transfer of data.

Document architectures are designed with different applications in mind and the integration of time dependent information requires extensions with respect to structuring and specification of temporal relationship. Database support for multimedia documents requires additional facilities these are :

- Query language and the description of records should support for multimedia data types
- Structural and relationship between parts of a document should be representable
- Piecewise access to large data and real-time access of continuous media data should be stream-oriented.

User cooperation on multimedia documents and requirements on data management have been identified. Existing approaches to cooperative applications are evaluated according to requirements.

References

[1] Hussein M. Abdel-Wahab, Sheng-Uei Guan, and Jay Nievergelt. Shared Workspaces for Group Collaboration: An Experiment Using Internet and UNIX Interprocess Communications. *IEEE Communications Magazine*, pages 10–16, 11 1988.

[2] F. Bancilhon, W. Kim, and H. Korth. A Model of CAD Transactions. In *60 6 Bibliography Proceedings of the 11th International Conference on Very Large Databases*, pages 25–33, Stockholm, Sweden, 1985.

[3] T. Crowly, P. Milazzo, E. Baker, H. Forsdick, and R. Tomlinson. MMConf: An Infrastructure for Building Shared Multimedia Applications. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 329–342, New York, 10 1990. ACM SIGCHI & SIGOIS, ACM Press.

[4] ISO International Standard 10166: Information Technology - Document Filing and Retrieval (DFR), 1991.

[5] K.P. Eswaran, J.N. Gray, R.A. Lorie, and I.L. Traiger. The Notions of Consistency and Predicate Locks in a Database System. *Communications of the ACM*, 19(11):624–633, 11 1976.

[6] C.A. Ellis, S.J. Gibbs, and G.L. Rein. Groupware: Some Issues and Experiences. *Communications of the ACM*, 34(1):38–58, 1 1991.

[7] G. Foster and M. Stefik. Cognoter, theory and practice of a Colab-orative tool. In *Proceedings of the Conference on Computer Supported Cooperative Work '86*, pages 7–15, 1986.

[8] J.N. Gray. Notes on Data Base Operating Systems. In *Operating Systems*, volume 60 of *Lecture Notes in Computer Science*, pages 394–481. Springer Verlag, 1978.

[9] Irene Greif and Sunil Sarin. Data Sharing in Group Work. *ACM Transactions on Office Information Systems*, 5(2):187–211, 4 1987.

[10] L.N. Garret, K.E. Smith, and N. Meyrowitz. Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System. In *Proceedings of the Conference on Computer Supported Cooperative Work '86*, pages 163–174. ACM, 1986.

[11] I. Greif, R. Seliger, and W. Weihl. Atomic Data Abstractions in a Distributed Collaborative Editing System. In *Proceedings of the 13th Annual Symposium on Principles of Programming Languages*, pages 160–172, New York, 1 1986. ACM.

[12] T. Haerder, Ch. Huebel, K. Meyer-Wegener, and B. Mitschang. Processing and transaction concepts for cooperation of engineering workstations and a database server. *Data and Knowledge Engineering*, 3:87–107, 1988.

[13] Sandra Heiler, Sara Haradhvala, Stanley Zdonik, Barbara Blaustein, and Arnon Rosenthal. *A Flexible Framework for Transaction Management in Engineering Environments*, chapter 4, pages 87–121. In Elmagarmid [Elma92], 1992.

[14] W.H. Harrison, H. Ossher, and P.F. Sweeney. Coordinating Concurrent Development. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 157–169. ACM, 10 1990.

[15] W. Kim, R. Lorie, D. McNabb, and W. Plouffe. A transaction mechanism for engineering design databases. In U. Dayal, G. Schlageter, and L.H. Seng, editors, *Proceedings of the International Conference on Very Large Databases*, pages 355–362, 1984.

[16] Thomas Kirsche, Horst Lührsen, Klaus Meyer-Wegener, and Hartmut Wedekind. A distributed sketching tool featuring freedom of action and equality of all users. 1991.

[17] M.J. Knister and A. Prakash. DistEdit: A distributed toolkit for supporting multiple group editors. In *Proceedings of the 3rd Conference on Computer Supported Cooperative Work*, pages 343–355. ACM, 1990.

[18] B.T. Lewis and J.D. Hodges. Shared Books: Collaborative publication management for an office information system. In *Proceedings of the Conference on Office Information Systems*, ACM SIGOIS Bulletin, pages 197–204, 1988.

[19] R. Lorie and W. Plouffe. Complex objects and their use in design transactions. In *Proceedings Data Base Week: Engineering Design Applications*, pages 115–121, 1983.

[20] J. Eliot B. Moss. Nested Transactions: An Approach to Reliable Distributed Computing. Tech. Report 260, M.I.T. Laboratory for Computer Science, 4 1981.

[21] Peter Muth, Thomas C. Rakow, Wolfgang Klas, and Erich J. Neuhold. *A Transaction Model for an Open Publication Environment*, chapter 6, pages 159– 218. In Elmagarmid [Elma92], 1992.

[22] Marian H. Nodine, Sridhar Ramaswamy, and Stanley B. Zdonik. *A Cooperative Transaction Model for Design Databases*, chapter 3, pages 53–85. In Elmagarmid [Elma92], 1992.

[23] ISO Draft Proposal 9579-1: Remote Database Access, Part 1: Generic Model, Service and Protocol, 1990.

[24] L. G. Reid and P.L. Karlton. A File System Supporting Cooperation Between Programs. In *Proceedings of the 9th Symposium on Operating Systems Principles*, pages 20–29, New York, 1983. ACM. Communication - Remote Operations, 1989.

[25] R.N. Jugele and V.N. Chavan,"Synchronization Mechanism In Multimedia Documents", International Journal of Engineering Science & Advanced Technology, ISSN:2250-3676, Vol. 2, issue 6, Nov-Dec. 2012, pp. 1690-1695.

[26] ISO International Standard 9072: Information Processing Systems – Text.

[27] M. Stefik, G. Forster, D. Bobrow, K. Kahn, S. Lanning, and L. Suchman. Beyondthe Chalkboard - Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM*, 30(1):32–47, 1 1987.

[28] R.H. Trigg, L.A. Suchman, and F.G. Halasz. Suporting collaboration in NoteCards. In *Proceedings of the Conference on Computer Supported Cooperative Work '86*, pages 153–162. ACM, 1986.

[29] Gerhard Weikum and Christof Hasse. Multi-Level Transaction Management for Complex Objects: Implementation, Performance, Parallelism. Technical Report 162, Department of Computer Science, ETH Zürich, 1991.

[30] Gerhard Weikum and Hans-J. Schek. *Concepts and Applications of Multilevel Transactions and Open Nested Transactions*, chapter 13, pages 515–553. In Elmagarmid [Elma92], 1992.