A Cooperative Database System based on Workspace Hierarchy

Didier Donsez, Pascal Faudemay, Philippe Homond

Laboratoire MASI / UPMC
4 Place Jussieu, 75252 Paris cedex 05, France
{donsez, faudemay, homond}@masi.ibp.fr

Abstract.

Cooperative work enables a users group to collaborate in the same project. The use of a cooperative database system enables distributed users to cooperate on large data volumes. We modelize the cooperative work by alternative versions which belong to each participant, and a merge mechanism which is controled by a group supervisor. This model is a variant of the distributed architecture model, called the Workspace Model. This model proposes a distributed virtual memory of objects, in a hierarchy of servers and clients. We have developed the WEA system, which implements this model.

Keywords : Cooperative Work, Object Oriented Databases, Distributed Systems, Workspace Model, Distributed Virtual Memory, Versions.

Résumé.

Le travail coopératif permet à un groupe d'utilisateurs de collaborer dans la réalisation d'un projet. L'utilisation d'une base de données coopérative rend possible la coopération d'utilisateurs distants et répartis sur un réseau, sur des volumes de données importants. Nous modélisons ce travail coopératif par des versions alternatives appartenant à chaque utilisateur, et un mécanisme de fusion de versions contrôlé par le superviseur du groupe. Ce modèle est une variante du modèle d'architecture distribuée appelé Modèle des Espaces de Travail, qui permet de gérer une mémoire virtuelle distribuée d'objets dans une hiérarchie de serveurs et de clients. Nous avons développé une implémentation de ce modèle, le système WEA.

Mots-Clés : Travail Coopératif, Bases de Données Orientées Objet, Systèmes Distribués, Modèle des Espaces de Travail, Mémoire Virtuelle Distribuée, Versions.

1. Introduction.

Cooperative work is an important activity in organizations (companies, laboratories, etc...). This type of activity may be favoured by the use of workstations networks. In such a case, group members may communicate between distant workstations, while using computer-based cooperative tools[1,2,3]. Examples of such tools are group editors, group debuggers, etc...E.g. in a group editor several participants update together a same document. Cooperative work may be asynchronous, as when using electronic mail, or synchronous, as for computer-assisted meetings. It may support a small group, or a whole organization. Application may execute in a single site, with interface replication, or in several sites on a common database.

In numerous cases, cooperative work uses a large number of archived data, such as in a CAD or software engineering workshop. Group work must then be supported by the underlying database system [4,5].

In this context, group work may be seen as a specific behaviour of embedded and global transactions. It combines a global transaction, called the supervisor, and embedded ones, called the cooperative transactions. More than isolated transactions, we shall consider transactions sequences, called activities. These transactions update database versions. These notions will be explained further.

à paraître dans les actes de*CODATA '94, Committee on Data for Science and Technology*, Chambery, France, Septembre 1994 .

A cooperative session is then composed of the following steps:

- An user creates a supervisor activity, which defines entry conditions for other participants, and decision procedures. Participants join the group.
- A participant of the cooperative group proposes database updates to the supervisor, as an alternative version.
- The supervisor informs the other participants of a new proposal. In a first step, this proposal is not known outside the group
- The supervisor concludes the cooperative work by deciding to set a final version of the database, composed of items from one or several alternative versions. The decision procedure executed by the supervisor is specific to each application and usually implements a consensus within the group.

This model of cooperative work uses known mechanisms from recent database systems. Objects or tuples versions are implemented in several database systems, either relational or object-oriented ones. Embedded transactions mechanisms have also been proposed.

In our approach, cooperative work is supported by a new entity which is both client and server, and which is called the Workspace [6]. This entity supports a new shared distributed architecture for distributed data, which extends the client-server architecture (see also [7]).We first present the Workspace properties which are used for cooperative work. We then describe the main utilized mechanisms. We conclude on the support of cooperative work by the Workspace model, and future prospects.

2. The Workspace model.

The Workspace is mainly composed of a database image, and of activities. The database image is a set of local objects, plus objects included in other, connected workspaces. Activities include service activities which display a view of the database image, and application activities, which are composed of transactions sequences. A Workspace can publish its services, it can also suscribe to services from other workspaces. In that case, its database image is the union of local data and of data available through suscribed services.

The Workspace may operate in passing mode or in retaining mode. In passing mode, committed updates are immediately copied in the Workspaces where the data come from (the passing Workspace operates as a write-invalidate cache). In the retaining mode, there is a retaining transaction, also called supervisor, which must commit the Workspace updates in order to make them visible to the outside world.

The Workspace implements versions mechanisms, which are based on database versions and not on objects versions. A database version is derived from a root version, which is itself derived from the initial database state. An object state is the state in the initial database, if it has not been updated since the creation of the new version, and the state in the new version, otherwise. In the case of the components of a composite object, the use of database versions enable to know the version of components, which would be more difficult with objects versions. If a component has been modified since the creation of the new version, their state is that of the new version. Otherwise it is the state of the initial database state or of the root version.

A Workspace can recursively access other Workspaces through a direct acyclic graph of subscriptions. The Workspace hierarchy is well adapted to heterogeneous hardware and networking environments, with various levels of servers. In such environments, various local area networks (LAN) can be connected by one or more wide area networks (WAN). Connection of a LAN to a WAN can be implemented by a connection Workspace, which plays the role of a front-end for this connection. A simple example of a Workspace hierarchy is presented at figure 1.

Workspace hierarchies are also well adapted to the modeling of cooperative activities in organizations as cooperative subgroups. Each subgroup corresponds to a Workspace. Several

subgroups cooperate through a connecting Workspace, which publishes global cooperative services suscribed by each subgroup. Detailed presentation of cooperating subgroups is outside the scope of this paper.

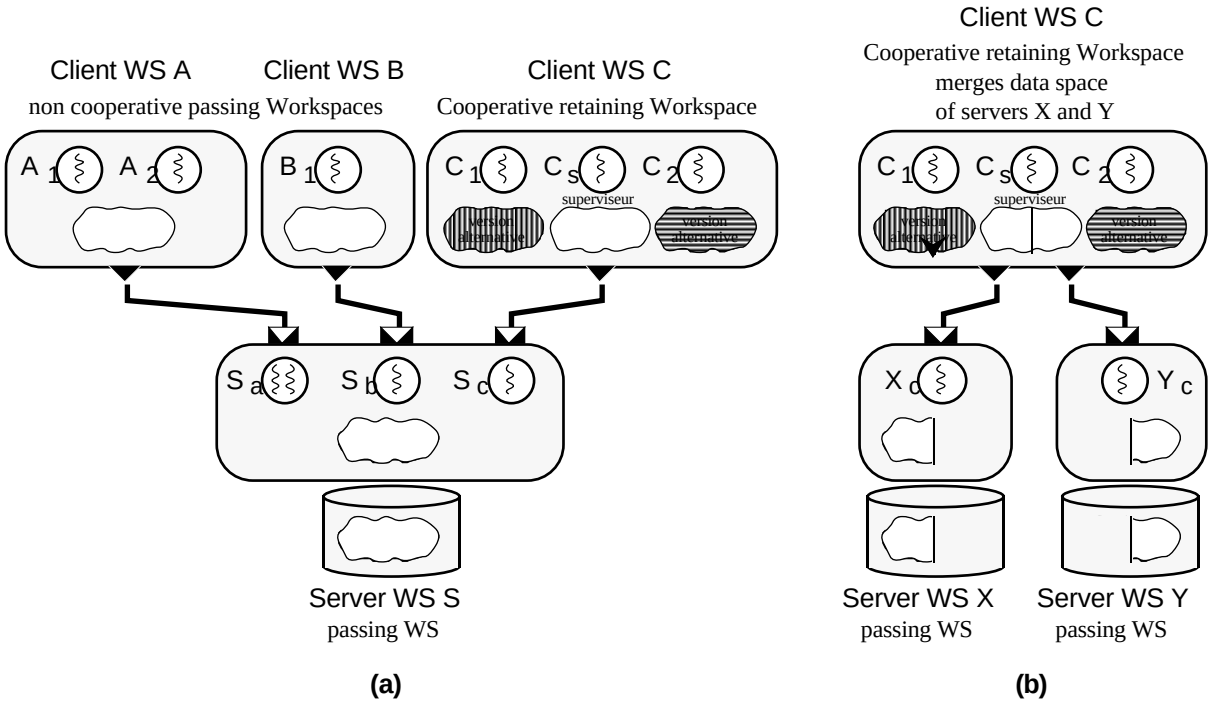

figure 1. examples of Workspace connections

3. Cooperation within the Workspace.

Cooperation mechanisms, which will be described further, do not interpret the semantics of the updates proposed by participants to the cooperative group. Updates analysis is left to the designer of the cooperative application, who defines the cooperative transactions and the supervisor procedures. Our basic mechanisms define generic operations for generation and update of alternative versions, and communication mechanisms between participants and supervisor.

Our approach of the cooperative work follows a centralized model, in which a participant (or member) of the cooperative group only communicates with the supervisor, or by shared access to a version. The cooperative work is modelized by a retaining Workspace, in which each group member is represented by a cooperative activity, and the supervisor by a supervisor activity, which is unique in the cooperative group. A shipping service links each cooperative activity with the supervisor activity.

The cooperative work deals with the whole database which is accessed by the Workspace. An update proposal of the cooperative database is presented as an alternative version. A version includes the updates of one or more objects in the database.

The cooperative activity instanciates one or more successive cooperative transactions, in order to read the alternative database versions, or to update them or propose new ones. The supervisor activity also instanciates cooperative activities to read or update versions which have been proposed by cooperative activities. This activity also instanciates a "normal" transaction, to update the database with the final version.

Cooperative transactions differentiate from other transactions (such as application transactions or services), as they never directly update the database. Proposed updates are made in the database version.

In the cooperative Workspace, a version is the property of one of the group members. There is a final version, which copied back into the database, and which is the supervisor property. Committed updates of a version are visible by the supervisor, and usually by cooperative transactions.

A version is usually updatable by the following transactions belonging to the same owner. The version owner and the supervisor may possibly decide that a version is stabilized, in which case it can no longer be updated. The following version of its owner is then derived from this stable version. The root version is considered as stable and visible by all. If a group member wants to keep a version at a given date, it must ask the supervisor for the stabilisation of this version.

The cooperative transaction ends either by the validation or by the abort of the proposed updates. The validation stores the updates in the current alternative version of the group member, or creates a new derived version. After the validation of the cooperative version, the corresponding cooperative activity informs the supervisor activity of the proposed modifications. The supervisor checks the modifications and notifies them to the other cooperative activities.

Supervisor activity

In the cooperative group, the supervisor activity is in charge of :

- the moderation of modified or derived versions, and their notification to the participants. The supervisor activity may ask a group member to modify its version before making it visible to the other group members.
- the compilation of the final version according to some decision procedure. In general this procedure includes a notification to other participants of the modifications proposed to a version by each group member, a vote (choice of a note) on each modification, together with vote explanations which are annotations visible by all group members, a decision, which is the choice of a version for each cooperation granule (e.g. by majority vote), and the publication of the result to group members and to the outside.

Notification transmits to the other participants, the name and the author of a modified version (possibly together with modification date), and references of all modified objects inside this update (and possibly the localisation of the update inside each modified object). The notification precision depends on the grain of the objects which compose the version.

Choice of a cooperation grain is important for the compilation of the final version based on the participants versions. E.g. if the cooperative work is the writing of a document, the final document may be composed of sections from the different versions, or of paragraphs from the different versions, etc...

The compilation of the final version by the supervisor enables some flexibility in the use of the cooperation grain. E.g. it is possible to notify a modification from character i to character j in object O, or to insert in the final version a modification characterised by an object name and by a characters range of indices.

Therefore the role of the cooperation grain is to facilitate the phase of negociation and vote. E.g. if characters i to k of an update belong to a granule, and characters k to j belong to the next granule, there will be two independent votes on the two granules (which may be e.g. to consecutive paragraphs).

Example

Let us illustrate these mechanisms with the example of figure 1. The cooperative database is first composed of white objects. The cooperative activity C1 has already proposed an alternative version VA1. In our figure, objects Spades, Clubs and Diamonds represent composite objects which serve as the cooperation granule of the application. A composite object is modified if one of its components is updated.

(a) cooperative activity C2 creates a new version called VA2 and submits it to the supervisor activity S. S acts as a moderator and refuses to broadcast VA2 to the group.

(b) activity C2 modifies version VA2 (object Clubs is modified), and submits it again to the supervisor activity, which accepts it, and broadcasts VA2 to the group.

(c) (1) activity S starts the consultation of cooperative activities in order to compose the final version. (2) Cooperative activities answer by voting. (3) Activity S receives the vote results, and decides to compose the final version with updates which have received unanimous votes such as grey Spades object and hachured Diamonds object. The Clubs object is therefore not selected for the final version. The supervisor activity ends the job of groups C1 and C2 by validating this version of the database. Then alternative versions can be suppressed.

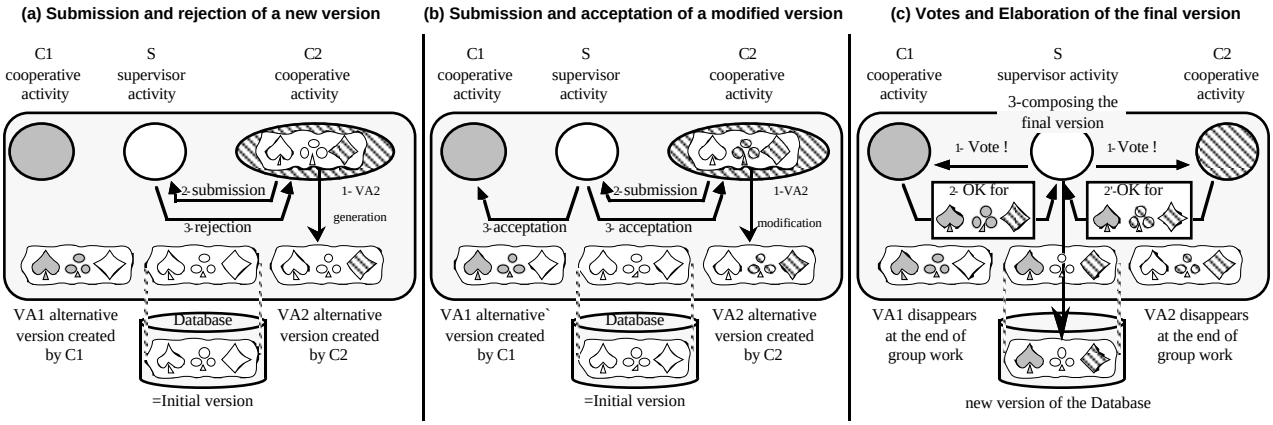

figure 2. Example of cooperative session

Cooperative work and cooperative steps

The cooperative work is a succession of cooperative steps. A cooperative step includes proposals of alternative versions, and composition of the final version by the supervisor activity. This final version is then used as the initial version for the next step. The cooperative step can end by the dropping of alternative versions produced during this step. It can also be desirable to keep the previous alternative versions if consensus in the previous step has not been reached on all objects. However, if alternative versions are implemented as differences vs the current root version, it may be difficult to keep the alternative versions of the previous step.

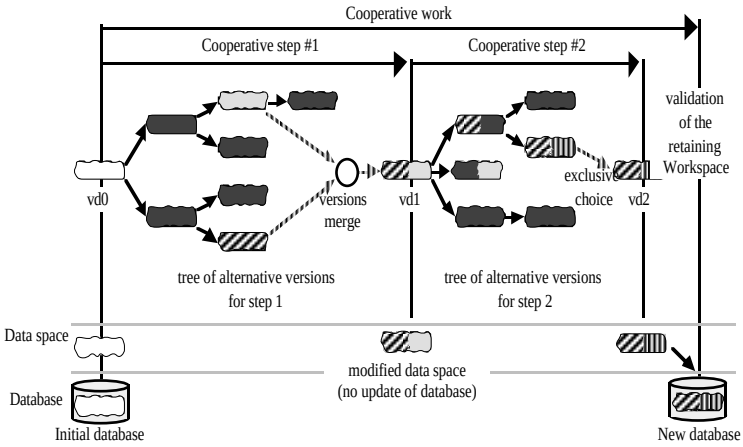

figure 3. Cooperative steps

Figure 3 illustrates a succession of cooperative steps. The group work begins on an initial database image (in white), which corresponds to version vd0. The end of the first step produces a final version vd1, by merging several versions of vd0. The second step begins with vd1 as an initial version. Alternative versions of vd0 are dropped. A second step ends by the composition

of final version vd2 based on a single alternative version. At last, the cooperative work ends by the validation of the retaining Workspace, which validates the database modifications resulting from vd2.

4. Conclusion

The Workspace model displays a natural way of managing cooperative activities which update a persistent distributed database. The model captures the basic activities mechanisms, while leaving the application semantics and the detailed negociation mechanisms to the cooperative activities and to the supervisor.

This approach appears to be well adapted to synchronous as well as asynchronous operation modes, in various dimensions groups. It also keeps the properties which directly result from the model, such as the ability to combine activities from several groups, and individual activities (figure 1), and the adaptation to many network configurations [Carey 94]. Therefore it is an alternative to approaches based on application sharing.

Its properties also include the access to cooperative services by members of distant groups, resistance to failures occurring during a group session, and the ability to structure the group into several sub-group levels. These properties will be the subject of future papers.

We have implemented the Workspace model on a Sparc workstation, under SunOS 4.& 5. This implementation is compatible with cooperative activities. Future tasks of the project include the implementation of libraries for the design of supervisors and cooperative activities, the experimentation on real working groups, and the evolution towards multimedia cooperative activities.

References

1 J.D. Palmer, N.A.. Fields, "Guest Editors' Introduction: Computer Supported Cooperative Work", IEEE Computer, vol. 27, No 5, Mai 1994

2 R.Reddy, K. Srinivas, V. Jagannathan, R. Karinthi, "Guest Editors'Introduction : Computer Support for Concurrent Engineering", Computer, vol. 26, No 1, Janvier 1993, 12-16

3 C.A. Ellis, S.J. Gibbs, G.L. Rein, "Groupware : Some Issues and Experiences", Communications of the ACM, vol. 34, No 1, Janv. 1991, 38-58

4 T.Kamita, S. Ichimura, K.I Okada, Y. Matsushita, "A Database Architecture and Version Control for Group Work", Proc. 27th Hawaii Int'l Conf. on System Sciences (HICSS-27), vol. III, Maui, Hawaii, Janvier 1994, 438-447

5 T. Kirsche, B. Reinwald, H. Wedekind, "Processing Dynamic Interactions in Cooperative Databases", Proc. 27th Hawaii Int'l Conf. on System Sciences (HICSS-27), vol. IV, Maui, Hawaii, Janvier 1994, 733-742

6 D. Donsez, P. Faudemay, P. Homond, "WEA, A Distributed Object Based on a Workspace Hierarchy", Proc. IFIP WG 10.3 Working Conference on Applications in Parallel and Distributed Computing, IFIP Transactions A-44, Caracas, Venezuela, Avril 1994, North-Holland, 247-256

7 M. Carey, et al., "Shoring Up Persistent Applications", ACM SIGMOD, Proc. Int'l Conf., Minneapolis, Minnesota, USA, Mai 1994, SIGMOD Record, vol. 23, No 2, Juin 1994, 383-394