

Dr. Ole J. Anfindsen, founder & CEO  
Xymphonic Systems AS  
Grensesvingen 7  
P. O. Box 6339 Etterstad  
0604 Oslo, Norway  
[ole@xymphonic.com](mailto:ole@xymphonic.com)

Revision 2: 2 October 2003<sup>1</sup>  
Revision 1: 11 March 2002<sup>2</sup>  
Revision 0: 13 February 2002<sup>3</sup>

# Collaborative Work and Xymphonic™ Transactions

## Executive summary

The purpose of this memo is to discuss the commercial, and to some degree also technical, aspects of a collaborative transaction model called *Xymphonic<sup>4</sup> Transactions*. This transaction model has the potential to profoundly change the way we use databases. This is so because the Xymphonic transaction model offers a combination of simplicity and generality never seen before. This opens up some *very* interesting business opportunities.

## 1 Introduction

The technology discussed in this memo has its roots firmly planted in the database world. To put this into perspective from the very outset, we make the following observation. The foundation of database management systems (DBMSs) has two main components:

1. A data model, describing the *structure* of the stored data.
2. A transaction model, describing the *behavior* of the database as a whole.

---

<sup>1</sup> Revision 2: updated information about company and products.

<sup>2</sup> Revision 1: only minor changes to the Introduction section have been made from Revision 0.

<sup>3</sup> Revision 0 is a significantly modified version of a memo written in 2000 under the names *Collaborative transactions*, and *Xymphonic transactions*. Material has been removed, rewritten, and added.

<sup>4</sup> Xymphonic™ and Xymphony™ are trademarks of Xymphonic Systems AS.

For the last twenty years the preferred data model has been the relational one, and the dominant database products are now relational. Even so, vendors of object-oriented systems have captured small but important niches of the database market (and legacy systems, in particular hierarchical and network-based, are still used by many). The international standards community and the relational vendors are currently putting significant efforts into generalizing the relational model of data with object-oriented extensions.

For an even longer period of time, the database industry has been dominated by the classical transaction model, characterized by serializability and the so-called ACID properties (atomicity, consistency, isolation, durability). Variations on this transaction model have been implemented by pretty much all the database vendors, e.g. by implementing the SQL standard's concept of isolation levels, or by multiversion concurrency control methods that eliminate conflicts between readers and writers. But in spite of a strong need in the market for a much more general model of database behavior, no one has yet been able to deliver this. This is where we claim to make a breakthrough contribution.

The xymphonic transaction model has the classical transaction model as a special case, but is capable of supporting a much broader class of interaction patterns among database users. This makes it possible for databases to play a much more active role than before in a number of areas. While the classical transaction model is perfect for some application domains, it will be limiting or even counterproductive in others. This is particularly true for all sorts of *collaborative work*, where the Xymphonic model is ideal.

Thus, the Xymphonic model supersedes the classical one. Xymphonic transactions provide the good, old ACID-properties to all applications for which this is the optimal mode of execution, while at the same time allowing other applications to explore the new world of *Xymphonic collaboration*.

## 2 Two simple extensions to the classical model

Xymphonic transactions have the classical model as a default, but add two new and powerful features that enable users to work *xymphonically* whenever they need to.

The most central concept in the Xymphonic model is that of a *xymphony*, which is a dynamically created *subdatabase* or *nested database*. By adding recursion and a few other things to standard concurrency control algorithms, a whole new set of possibilities opens up, and arbitrarily complex patterns of interaction between two or more users can be established. Using xymphonies just to deal with simple sharing of data between readers and writers would however be quite awkward. The Xymphonic model therefore offers a much more convenient mechanism for this, based on user-defined *access parameters*. This enables users to selectively share data with one another, and also serves as a vehicle for providing meta-information about the data objects in question (such as e.g. reliability, maturity, or degree of completeness).

Taken together, xymphonies and access parameters enable *any* potential conflict between transactions to be handled in a simple yet controlled fashion. The formal foundation for this consists of two simple generalizations of classical *serializability* theory.

## 3 Simplicity and generality: some technical background

The Xymphonic model is aimed at bringing the familiar benefits of transaction management to application domains that need long-lasting or collaborative transactions. It is generally acknowledged that so-called classical transactions are unsuitable for this purpose (see e.g. Gray & Reuter 1993, page 180). The problem of collaborative work is therefore well known

in computer science, and lots of research papers have been published on this topic. A few examples, covering most of the application domains mentioned in this memo, include (Alonso et al 1995; Kaiser 1990, 1995; Korth & Speegle 1990, 1994; Kumar & Wong 1993; Nodine & Zdonic 1992; Rauft et al 1990), and many additional references are given in (Anfindsen 1997).

While a considerable number of new transaction models have been proposed in the research literature over the years in order to deal with these problems, and although these models have many theoretically interesting features, they tend to be too complicated to implement and use. Thus, they have had very little commercial impact. Xymphonic transactions is radically different from these other transaction models, and to the best of our knowledge, no other model is even close to having the Xymphonic model's combination of *simplicity and generality*. This unique combination of simplicity and generality, combined with an obvious need for collaborative transactions in several application domains (see below), is what convinces us that Xymphonic transactions have an important role to play in the years ahead.

The simplicity of the Xymphonic model is evident at four levels:

1. It has a simple formal foundation based on a straightforward mathematical generalization of classical transaction theory.
2. It can be implemented by relatively modest extensions to the data structures and algorithms used in lock managers and similar transaction-related components.
3. Its functionality can easily be exploited by application programmers.
4. It enables end-users to navigate the *xymphonic space* simply by clicking on menu options, buttons, check boxes, and other familiar GUI-elements found in all state-of-the-art desktop applications.

Claim 1 is obvious from the definition of Xymphonic transactions (Anfindsen 1997), and has repeatedly been pointed out by reviewers of my journal and conference papers (Anfindsen 1995, 1996abc, 1998ab). Claims 2, 3, 4 have been demonstrated by the implementations of the model.

## **4 The need for collaborative transactions**

The problem of collaboration shows up whenever two or more users need to work with the same data objects *during an extended period of time* (anything from minutes to months). The classical textbook example is that of designers or engineers using a CAD/CAM tool. On the one hand, they need *access*, both read and write access in the general case, to each other's portions of the data. On the other hand, they need *control* over their data. And there is an inherent conflict between concurrent access and control.

The need for long-lasting, interactive, collaborative transactions is common to several application domains, and the most well-known examples include collaborative editing, workflow management, software development, CASE tools, product data management (PDM), and all sorts of software used by engineers, architects, and the like who design and build complicated structures like e.g. houses, bridges, ships, engines, cars, and airplanes. It has also been suggested that e-commerce could benefit from transactional behavior, and we believe the kind of tools offered by some for interactive specification of orders (by a customer who wants to buy a new, "tailor made" car, say), is an interesting example in this context.

Of course, thousands of people all over the world are already using a myriad of software products in the just mentioned areas. The problem is that the mechanisms offered (if any) to help users carry out the collaborative aspects of their work, are often far less than satisfactory. Apart from techniques like check-out/check-in and versioning, which may work well in some special cases but do not provide a general solution to the underlying general problem, users are by and large stuck with ad-hoc mechanisms.

The just mentioned application domains are large and important. CAD/CAM alone generates annual revenue of approximately 20 billion dollars. And there are millions of people who use word processors on a daily basis, many of which would benefit greatly from being able to do collaborative editing of large reports, bids, specifications, user manuals, or the next issue of a newspaper or magazine, just to mention a few examples.

Clearly, a general solution to the problem of collaborative work opens up some *very* interesting business opportunities. This is probably best explained by the tremendous cost saving potential businesses can utilize when their staff is equipped with xymphonic software.

## **5 The premises of this memo**

The validity of the arguments advanced in this memo, critically depends on three premises, all of which will now be highlighted.

### **5.1 The importance of collaboration**

Premise number one underlying this memo is that collaboration is important. It is our firm belief that once the necessary technology becomes generally available, hardly anyone will do any of the following without using xymphonic software tools:

- Design and build airplanes, cars, trucks, ships, rockets, engines, roads, houses, bridges, factories, oil drilling rigs, or anything else of sufficient complexity
- Develop complicated software
- Publish newspapers or magazines
- Write large documents, such as complex bids or tenders
- Manage complex workflow processes
- And more.

It follows that there is an enormous market potential for xymphonic software.

### **5.2 The relevance of transactions**

Premise number two is that *database transactions* represent the best way to deal with the challenges of data-centric collaboration. Every research paper in this memo's reference list argues in favor of this position, or simply takes it for granted. Although there are researchers who think differently, many more references could be cited in support of using transactions in this context, see (Anfindsen 1997) for examples.

And there are some very good reasons for this. The essence of the argument is that transactions are at just the right level of abstraction to be easy to use and at the same time

powerful enough to actually do the job. Users of transactional systems basically have to say "begin transaction" in order to enter a *sphere of control* where concurrency control is automatically provided for them by the system. When it's time to leave that sphere of control, it is done by issuing a commit or abort command. This means that all the low-level stuff that is necessary to make this work, is hidden from the users. And on top of this transactions provide recovery control as well. It could hardly be simpler.

This is in stark contrast to non-transactional mechanism for concurrency control. If they are powerful, then they are not easy to use, require low-level manipulations, require a deep understanding of the system in question, and may require total knowledge of all the concurrent users at any given point in time. Conversely, any non-transactional tools for concurrency control that are easy to use, are bound not to be powerful and general. And recovery has to be treated as a separate issue, as opposed to the integrated approach taken by transactions.

The reasons some people reject transactions as the basis for data-centric collaboration basically fall in one of the following two categories: Either (1) transactions are not powerful enough to allow the desired interaction patterns, or (2) the only transaction models that *are* powerful enough are also too complicated to use, too hard to implement, has a high run-time overhead, or some other disqualification like that. None of these objections apply to Xymphonic transactions.

### **5.3 The superiority of Xymphonic transactions**

Premise number three is that the Xymphonic model is the best collaborative transaction model available. This is based on positive as well as negative evidence. Beginning with the former, the Xymphonic model:

- Has a simple yet strong formal foundation
- Enables dynamic sharing of data (by means of access parameters)
- Enables dynamic collaboration (by means of xymphonies)
- Is customizable
- Is easy to implement
- Is easy to use
- Has low run time overhead
- Is general, i.e., not tailored to any specific application domain
- Is flexible
- Is orthogonal to and/or compatible with all other relevant technologies that we are aware of, including recovery, savepoints, versioning, data models, programming languages, constraint management, splitting and joining of transactions, the use of pre- and postconditions, auditing, access control, and authorization schemes
- In sum, has a unique combination of simplicity and generality.

First, these claims can be backed up by patents, published papers, and my PhD thesis. Second, they can be backed up by our implementations of Xymphonic transactions, the most important of which uses the Oracle 9iFS platform.

The negative evidence in favor of Xymphonic transactions, is that there seems to be no other models nearly as well suited for collaborative transactions. Frankly, we are not aware of a single *commercially viable* competitor, but would more than welcome a careful comparison of Xymphonic transactions to any other transaction model that could potentially be of interest in this context.

## 6 A few words about Xymphonic Systems AS

Xymphonic Systems AS is a small technology start-up established in 1998 (originally under the name of Apotram AS) by Telenor Research and Development. Telenor Corporation ([www.telenor.com](http://www.telenor.com)) is the largest telco in Norway.

We have six US patents related to Xymphonic transactions, two granted and four pending.

## 7 Concluding remarks

Collaboration between human beings is important in any civilized society. The lack of proper support for collaboration in today's computing infrastructure is undoubtedly one of its most profound shortcomings. Thus, we venture the following predictions:

- The IT industry is headed for a long overdue revolution or paradigm shift in this area
- Transactions are going to play a pivotal role in this transition
- This will have a major impact on the way we use computers.

If our predictions are not too far off, it goes without saying that vendors who choose to support xymphonic collaboration in their products will benefit significantly.

## 8 References

Alonso, G, Agrawal, D, El Abbadi, A, Kamath, M, Günthör, R, Mohan, C. 1995. *Advanced transaction models in workflow contexts*. IBM Almaden Research Center, San Jose, California, IBM research report RJ9970.

Anfindsen, O J. 1995. Dynamic cooperation between database transactions by means of generalized isolation levels. In: *Proceedings of 2nd International Conference on Concurrent Engineering: Research and Applications*. McLean, Virginia, 249-260. (this paper was published shortly before the formal foundation of Xymphonic transactions had been developed, but is included here in the interest of completeness)

Anfindsen, O J. 1996a. Cooperative work support in engineering environments by means of nested databases. In: *Proceedings of 3rd International Conference on Concurrent Engineering & Electronic Design Automation*, Poole, UK, 325-330.

Anfindsen, O J. 1996b. Cooperative work support in multimedia conferences by means of nested databases. In: *Information Network and Data Communication*. (Proceedings of 6th IFIP/ICCC INDC International Conference, Trondheim, Norway) F A Aagesen, H Botnevik, D Khakhar (eds). London, Chapman & Hall, 317-326.

Xymphonic transactions white paper, 2 October 2003, Revision 2

- Anfindsen, O J. 1996c. Supporting Cooperative Work in Multimedia Conferences by means of Nested Databases. In: *Proceedings of Norwegian Informatics Conference - NIK'96*, Alta, Norway, 311-322.
- Anfindsen, O J. 1997. *Apotram - an application-oriented transaction model*. PhD Thesis, Department of Informatics, University of Oslo, Norway. Research Report 215. (this paper defines Xymphonic transactions, the academic name of which is Apotram)
- Anfindsen, O J. 1998a. Conditional Conflict Serializability - an application-oriented correctness criterion. In: *Proceedings of International Workshop on Issues and Applications of Database Technology - IADT'98*, Berlin, Germany. 47 - 54.
- Anfindsen, O J. 1998b. Conditional Conflict Serializability - an application-oriented correctness criterion (extended version). *Journal of Database Management*, Vol 9, No 4, 22 - 30.
- Gray, J, Reuter, A. 1993. *Transaction processing: concepts and techniques*. SanMateo, Calif., Morgan Kaufmann Publishers.
- Kaiser, G E. 1990. A flexible transaction model for software engineering. In: *Proceedings of the International Conference on Data Engineering*, 560-567.
- Kaiser, G E. 1995. Cooperative transactions for multiuser environments. In: *Modern database systems*, Kim, W (ed). Reading, Mass, Addison-Wesley, 409-433.
- Korth, H F, Speegle, G. 1990. Long-duration transactions in software design projects. In: *Proceedings of the International Conference on Data Engineering*, 568-574.
- Korth, H F, Speegle, G. 1994. Formal aspects of concurrency control in long-duration transaction systems using the NT/PV model. *ACM Transactions on Database Systems*, 19, (3), 492-535.
- Kumar, M, Wong, J. 1993. Transaction management in design databases. *Journal of Systems Software*, 22, 3-15.
- Magnusson, B, Asklund, U. 1995. Collaborative Editing - distribution and replication of shared versioned objects. In: *Proceedings of ECOOP'95 Workshop on Mobility and Replication*. Aarhus, Denmark.
- Nodine, M, Zdonik, S. 1992. Cooperative transaction hierarchies: Transaction support for design applications. *VLDB Journal*, 1, (1), 41-80.
- Rauft, M A, Rehm, S, Dittrich, K R. 1990. How to share work on shared objects in design databases. In: *Proceedings of the International Conference on Data Engineering*, 575-583.
- Vogel, A, Rangarao, M. 1999. *Programming with Enterprise JavaBeans, JTS and OTS - building distributed transactions with Java and C++*. Wiley Computer Publishing, New York.