# Databases and Workflow Management: What is it All About?

## Position Statement

Workflow management has become a fairly hot topic during the past two or three years. There is an increasing number of new products, and in addition, numerous vendors that have been around for a while have discovered that what their products actually do is workflow management. Terminology is just as undefined as in any other field of our discipline, and so it is not trivial to define precisely what workflow actually is, which components it includes, how the minimal functionality can be characterized, and so on. A consortium of vendors, users and researchers called "workflow coalition" has been established to sort out these issues, to find a a reference model, and come up with standards wherever this seems possible. Although this group has making good progress, it has not had much influence on existing products yet.

The kinds of services which are currently subsumed under the heading of "workflow management" come from very different origins. Some of them are functionally enhanced mail systems; the most notable example of this group is Lotus Notes. Another group is defined by TP-managers, which typically also have some kind of high-level controlflow language, such as STDL. A third group comprises those systems, which have been developed as workflow management systems "from first principles", generalizing the ideas one cannot find in production planning systems, job control systems, etc. A typical example of this approach is IBM's FlowMark system.

Now, is there anything database-specific in workflow management? Consider that the key purpose of workflow management is to orchestrate, control, supervise and schedule the execution and dependencies of a large number of related activities, the completion of which may take a fairly long time. Typically, some of the activities of a complex workflow script will execute transactions against some shared database, but this is not the topic of this panel. The key aspect is the long duration of the computation defined by a workflow specification. Since such computations can go on for days, weeks, or months, they will accumulate a large amount of "state", which is not shared with anybody else, but as private that particular workflow execution. But because it is a long-lived computation, it must be guaranteed that no hardware or software failure can stop it or cause it to roll back; it rather has to be rolled forward according to the specification, which means appropriate recovery mechanisms and preservation of the state information in the presence of failures. Maintaining state is something database-systems are good at. On the other hand, under the transaction paradigm they do roll back recovery in case of a failure rather than rolling things forward to the most recent state. But the obvious question is, where the database systems should be functionally enhanced in order to support workflow management systems, or whether those systems should roll their own.

The second interesting issue has to do with synchronization of accesses by workflow activities on shared databases. We assume that the database accesses issued by activities being part of the workflow are covered by conventional transactions. This means all locks are released at the end of the transaction, but of course the workflow still goes on, and most likely there are consistency constraints expressed in terms of data intershare database which are relevant for the workflow in the long run. So the question is how to maintain "long-lived" consistency constraints on shared data for individual workflows without making conventional locks long-lived - because this would be an obviously bad idea. This looks like a database problem, or at least it has a lot to do with databases. The third group of problems has to do with events. Events play an important role in coordinating and synchronizing the activities within a workflow, and due to the recoverability requirements, events have to be recoverable, too. The question here is, what databases can do in support of "stable" events, and whether this is similar to what is discussed in the field of active databases.

To sum it up: Even though the functional requirements for workflow management systems are just beginning to emerge, I think there is clear evidence for the need of additional database functionality in order to support such systems.


**Proceedings of the 21st VLDB Conference**
**Zurich, Switzerland, 1995**