

# Mobile Database Agents for Building Data Warehouses

Edgar Weippl, Josef Altmann, Wolfgang Essmayr

Software Competence Center Hagenberg

Hauptstr. 99, A-4232 Hagenberg, Austria

E-Mail: {edgar.weippl, josef.altmann, wolfgang.essmayr}@scch.at

## Abstract

*Agent technology has a number of characteristics that make it well suited for automating information processing tasks and information retrieval. In this paper<sup>1</sup>, we propose a pragmatic approach for applying mobile agent technology within distributed database management systems (DBMSs). The developed agent environment is based on the Oracle DBMS. It differs from other agent platforms (e.g. Aglets, Voyager) in its support of error recovery, transactions, efficient data processing, and security mechanisms. Building mobile database agents upon these services, the run-time environment is faster, more secure and much easier to handle.*

## 1. Introduction

Today, *operational database systems* are widely applied in enterprises for *on-line transaction processing* (OLTP) which is determined by a large number of transactions that are in general structured, repetitive, require accurate data, and access relatively few data items at once. The main goals of OLTP are high data throughput and many transactions-per-seconds, even in case of distributed databases (DBs).

Recently, enterprises increasingly tend to analyze consolidated data, referred to as *on-line analytical processing* (OLAP), in order to support management and strategic decision makers. OLAP applications are determined by complex evaluations of aggregated data with the main focus on a high throughput of complex queries. Since extensive analytical processing apparently would impact the proper service of the enterprise's operational DBs the use of *data warehouses* (DWH) is common in order to meet the demands of OLAP.

For building a DWH, data has to be *extracted* from operational DBMS, *transformed* in order to overcome formal and semantic inconsistencies between the operational and the warehouse data models, *loaded* into the DWH, and periodically *refreshed* in order to reflect the changes of the operational systems within the DWH. However, we claim that the application of *mobile agents* for extracting and transforming operational data as well as for loading, and refreshing the warehouse could significantly enhance the current practice.

While there are many definitions of mobile agents, they can be usefully viewed as autonomous objects that can move between various hosts in a system according to appropriate commands within their own code. This mobility can mean a substantial performance gain and a reduction in network traffic if large volumes of data stored at remote hosts do not have to be transferred over the network for constructing DWHs. Mobile agents offer a possibility both to analyze and filter the data locally and report only preprocessed data.

The majority of the enterprises' data sources for building warehouses are operated by DBMSs, which already offer a range of services, like recovery, transaction handling, efficient data processing, concurrency, security, etc., not naturally available in conventional agent systems. Therefore, this work proposes to apply the mobile agent technology within the context of DBMSs, resulting in *mobile DB agents*. This approach offers several important advantages compared to regular agent systems since building upon DBMS services makes the run-time environment faster, more reliable, more secure, and much easier to handle.

DWHs are an emerging research area especially with the increased interest in business intelligence and analytical applications. An exhaustive introduction from the perspective of the underlying data management systems provides, for instance, Elmasri [4]. Brackett [2] analyzes the development of DWH systems whereas Kimball et al. [6] provides practical considerations for

designing, developing, and deploying DWHs. Wu [8] identifies research issues ranging from modeling issues, architectures, the maintenance of DWHs, to the operation and optimization of DWHs. Katic et al. [5] develops a prototype security model for DWHs based on meta-data.

Mobile agents are an active research topic, especially as tools such as Java make mobile agent systems relatively simple to implement. Milojicic et al. [7] gives an overview on systems like Aglets, Concordia, etc. Cabri et al. [3] summarize the main benefits why the mobile agent approach is useful.

As Aridor [1] states, agent-based application design is still a pioneering discipline. One possible approach for capturing good solutions to common problems lies in the use of design patterns. Some mechanisms we identified during our work, are very similar to the *Forwarding* and the *Locker* pattern. The *Forwarding* pattern provides a way for a host to forward newly arrived agents automatically to another host. The *Locker* pattern can be used to temporarily store data. Agents can avoid taking along data that is not needed. Later on, agents can return and retrieve the private data stored in the *Locker*. In our system design, the finishing line (section 3.3) offers this functionality.

## 2. Application Domain

The work is part of a research cooperation with AMS Engineering Austria. The major goal of this cooperation is to efficiently manage various tool machines distributed across a worldwide production environment. AMS Engineering designs, manufactures, and installs machine tools, which are *computerized numerical control* machines connected to a computer that controls them. These computers have access to mostly local DBs; they read outstanding orders to be processed from the DB and write back production data, like errors, supply shortages, etc., into the DB.

Fig. 1 shows two local sites of the production environment. Each machine is connected to a local DB and stores data in this DB while operating. Whenever an error occurs, it saves the current parameters that describe the error conditions. When repaired, the steps that have been taken to correct the malfunction are recorded in the local DB. Our goal is to provide means that if a machine fails one can submit a query to check what have been the circumstances leading to the breakdown, whether similar situations have happened before, and – if yes - what has been done to restart the

machine. In order to answer such kind of questions the use of a DWH that maintains consolidated data retrieved from the various operational DBs is evident. The emerged production DWH then allows applying OLAP and data mining techniques for knowledge-based information retrieval. Data warehousing technologies have already been deployed successfully in many business domains, for instance, manufacturing or retail. DWH technology is used within a manufacturing environment in order to analyze error conditions and to support the recovery process of the tool machines.

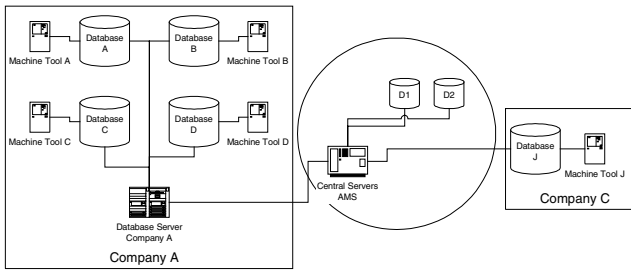
The worldwide production environment builds upon distributed DBs that are explored by agents in order to build up a DWH. The agents we use are executed and migrated within a DBMS unlike many other agent-based systems, giving them the name *mobile DB agents*. Users enter a simple query or a small stored procedure and start an *agent dispatcher*. The dispatcher is being executed within the starting DBMS, transfers the query or stored procedure encapsulated within an agent to other DBMSs. The *agent runtime environment* in these DBs executes the queries respectively stored procedure codes, returns the results and propagates the agent to other DBMSs that have not yet been visited. Agents that collect and process information are nothing new. However, agents that execute and migrate within DBMSs have not yet been widely used. There are major advantages in comparison with general-purpose agent environments:

First, the DBMS itself offers recovery, transaction handling, efficient data processing, concurrency and security mechanisms. Therefore, the whole run-time environment is much easier to implement because it can build upon services that are stable and have been tested by thousands of users.

Second, as the agents are executed within the DBMS, data access is much faster than via external interfaces (e.g. JDBC), even if the Java program and the DB reside at the same computer.

Third, simple SQL queries and schema modification can be executed much easier. If, for example, a new table has to be created in all DBs, in our approach the user only has to enter the relevant SQL code and can start this SQL command encapsulated within an agent.

As all agents execute within the DB, they cannot be used for tasks other than data gathering, data manipulation and schema modifications. Nevertheless, these operations are still crucial within the context of DWHs and their importance will certainly increase in the near future.

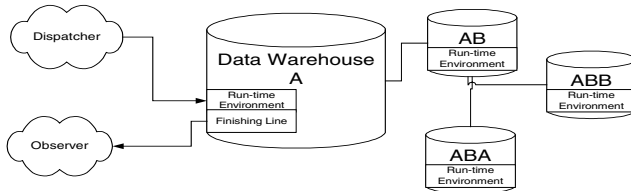


**Fig. 1: An example of a worldwide manufacturing environment.**

### 3. Concepts for Mobile DB Agents

Within this section we describe a set of concepts needed for realizing *mobile DB agents*. Note, that we concentrate on how to realize agent functionality within DBMSs due to the limited scope of this paper. Within future work we will elaborate how to realize extraction, transformation, loading and refreshment functionality for DWHs by the use of mobile DB agents.

A worldwide production environment as described in section 2 may look like illustrated in Fig. 2 when using a DWH for analytical processing. The DWH *A* directly retrieves data from two operational DBs, namely, *AA* and *AB* and is indirectly linked with two further DBs, *ABA* and *ABB*. In order to realize mobile DB agents, every DB - including the DWH - has to have an *agent run-time environment*, which is capable of executing and migrating mobile DB agents. Furthermore, an *agent dispatcher* is needed in order to initially create an agent within a particular run-time environment. And finally, an *agent's finishing line* collects the agent's results and servers the user for observing them.



**Fig. 2: A DWH that builds upon data from operational DBs.**

Within the example of Fig. 2, a dispatcher inserts a particular query into the run-time environment of *A*. The run-time environment now executes the task on *A* and

then creates and migrates the agents to *AA* and *AB*. The run-time environment at *AA* notices the new agent, executes it and finds it impossible to send it to another DB. Therefore, *AA* only sends the results back to *A*. DB *AB* also accepts and executes the agents but additionally propagates them to *ABA* and *ABB*. The following subsections describe the particular components of a DB agent system in more detail.

#### 3.1. Agent Dispatcher and Observer

An *agent dispatcher* is required to initially insert an agent's *code base* into a particular DB and to provide the required data, i.e. the *migration option* and the *finishing line*, so that the run-time environment can execute the agent.

The *code base* consists of the program code that the agent will execute in order to perform his task. The code base can be as simple as a single SQL statement or as complex as a function or procedure that is to be executed within a DBMS. The *migration option* specifies whether the agent will migrate to one DB after another or to all DBs simultaneously (see also section 3.2). Almost all agents will have to return data. The *finishing line* is a single location where an agent places its results before migrating to the next DB. In many cases this finishing line will be located where the agent first started.

Once the dispatcher has shipped a new agent to a particular DB, the run-time environment will notice the arrival of a new agent and take all necessary steps to execute and migrate the agent.

The *observer* can be any program that processes the results returned by the agent. It watches the agents' finishing line (see section 3.3) and responds to modifications of this table.

#### 3.2. Agent Run-Time Environment

Every DB that agents may visit needs to have a run-time environment to accommodate agents. The run-time environment within a DBMS will be based on tables and stored procedures. The tables hold the code base of the mobile agents, the table names of the results and the particular states of each individual agent.

The agent itself has to be realized as stored procedure located in the DB. A stored procedure on the local DB is triggered if a run-time environment located on another DB or an agent dispatcher transfers a new agent to the run-time environment. This procedure

initializes the mobile agent, i.e. it creates the result tables and starts the agent.

After the agent has finished, another stored procedure will be started. The agent may ask this procedure to transfer the entries of its result tables to a specified DB where its finishing line is located. The result tables are then deleted and the agent is ready for migration. The run-time environment has to send the mobile agent to the run-time environment of the target DB. Optionally, the agent may contain a list of DBs it wishes to visit or a list of DBs, it does not wish to go.

Depending on the agent's setting, two *migration modes* are available: the agent can either be transferred to only one other DB (*serialized*) or to all connected DBs *simultaneously*. The simultaneous dispatching of agents normally reduces the overall execution time of a query (not the processing time in terms of overall systems load). Still, there are some cases in which it is not feasible to deploy many agents at the same time. E.g., for finding only one of possibly many rows for which a particular condition (e.g. attribute1='A' AND attribute2='B') holds, it is better to deploy the agents in a serialized way since if the first row meeting the condition is found the work is done. The serialized mode is comparable to a depth-first search whereas the simultaneous mode to a breadth-first search.

### 3.3. Agents' Finishing Line

The *agents' finishing line* consists of tables in one DB where an agent deposits all results. It collects the final results that mobile DB agents generate while exploring other DBs.

The mobile agent basically has two ways of shipping results to the finishing line. The first way is to send the results just before migrating to the next DB.

Fig. Fig. 2 shows a network topology for which this approach is feasible. The major advantage is that users can access parts of the results long before the agents have visited all DBs. The second option for mobile DB agents is to take the results along with their code when migrating and only return them after having performed all of the work. However, the feasibility of the second approach has to be proved outside of test labs.

In this section we explained the basic concepts needed to build a system for mobile DB agents. The next section demonstrates these concepts using a small prototype implementation in Oracle.

### 3.4. Prototype Implementation

We realized the *agent dispatcher* as a Java program which offers a simple user interface to enter an agents *code base* (a query or stored procedure), an agents *finishing line* (e.g. a DB link), information about the mode of migration (simultaneous or serialized), means to load a code base from a file, and finally a method for deploying the agent. The primary task of a dispatcher is to ship a new agent to a particular DB.

We implemented the basic run-time environment with two tables and two stored procedures. As mentioned above, this run-time environment has to be located in every DBMS. The *agent residence table* contains a list of all agents that have ever been at the local DB. Agent identifiers are not deleted after the agents leave the DB to avoid re-accepting the very same agent several times. The *agent environment table* stores the agents' code bases, the names and definitions of the result tables and the dispatching mode. The table below shows an example with one agent residing at the DB.

When a new agent arrives one or more *result tables* are created. Then, the mobile agent - being a stored procedure itself - is created by the stored procedure `A_init`. Once the agent has finished, the stored procedure `A_terminate` dispatches it simultaneously to all other DBs linked to the local DB. Before the result table `A_A1_MYRESULT` is deleted, the results are sent directly to the finishing line at `MYFINRESULT@DBLINK`.

Agent	Attribute Type	Attribute Value
A1	CODE	INSERT INTO MYRESULT (SELECT sum (c) FROM T)
A1	RESULT_TABLE	MYRESULT
A1	RESULT_T_CREATE	Sumcost Number
A1	FIN_LINE	MYFINRESULT@DBLINK
A1	DISPATCH_MODE	SIMUL

**Table 1: An example of the agent environment**

A simple Java program, the observer, accesses the agent's result tables in this DB and acts as the user interface for the finishing line. In most cases result tables of the finishing line are located in the same DB that the agent was initially deployed to.

### 3.5. Heterogeneity Considerations

An important feature in agent-based systems is the ability to cope with heterogeneous environments. The prototype system as described in section 3.4 works only with Oracle DBMS. As there are also other major DBMSs, we currently work on extending the agent environment towards non-Oracle systems. The term "non-Oracle system" implies both, non-Oracle DBs that are accessed using SQL, and systems that are accessed procedurally.

Oracle provides means to access data stored on non-Oracle systems. *Heterogeneous Services* is an integrated component within the Oracle8i server, and provides the generic technology for accessing non-Oracle systems from the Oracle server. *Heterogeneous Services* can be used (1) to transparently access data stored in non-Oracle systems as if the data resides within an Oracle server, and (2) to use Oracle procedure calls to transparently access non-Oracle systems, services, or *application programming interfaces* (APIs).

When writing applications, you need not have to take into account that a non-Oracle system is accessed. *Heterogeneous Services* makes the non-Oracle system appear as if it were another Oracle8i server.

In a first step we avoid migrating the agent itself to a non-Oracle system. Instead we think of 'virtually' migrating it, which means that the runtime environment executes the agent code using a *Heterogeneous Services* agent. The trade-off for this quick and easy implementation is that the mobile agent cannot migrate to DBs that are connected to the non-Oracle systems.

### 4. Conclusions and Future Work

We have shown how the concept of integrating agent functionality into distributed Oracle DBs can be used to establish mobile DB agents for building DWHs. Our main contribution is the implementation and evaluation of a first prototypical agent-based system appropriate for processing large volumes of production data stored in DBs connected via TCP/IP.

The main issue in this ongoing project is to elaborate the application of mobile DB agents for realizing typical DWH functionality, like, extraction, transformation, loading, and refreshment of operational data. As mentioned above, heterogeneity is also an important issue with respect to mobile agents. We described a way for adding support of non-Oracle DB systems by using Oracle's *Heterogeneous Services*.

However, additional work has to be done in order to allow agents to physically migrate to non-Oracle DB systems. Security will be another focus of our future research. We try to use built-in DB functionality to protect an agent's run-time environment against malicious agents. In addition, it can not be guaranteed, that an agent's run-time environment does not try to manipulate arriving DB agents respectively learn about the agent's mission and thus potentially leaks confidential information carried by the agent. The use of cryptographic methods seems to be feasible in order to protect agents against malicious systems.

### 5. References

- [1] Aridor Y., Lange. D.B Agent Design Patterns: Elements of Agent Application Design. *Proc. of the 2nd ACM International Conference on Autonomous Agents*, Minneapolis, pp. 108-115, 1998.
- [2] Brackett M.H. *The DWH Challenge - Taming the Data Chaos*. John Wiley & Sons, 1996. ISBN 0-471-12744-2.
- [3] Cabri G., Leonardi L., Zambonelli F. Mobile-Agent Coordination Models for Internet Applications. *IEEE Computer*, Vol. 41, pp. 82-89, 2000.
- [5] Elmasri R.A., Navathe S.B. *Fundamentals of Database Systems* 3rd ed., Addison-Wesley Pub Co., 1999.
- [6] Katic N., Quirmayr G., Schiefer J., Tjoa A M. A Prototype Model for DWH Security based on Metadata. *Proc. 9th Int. Conf. on Database and Expert Systems Applications (DEXA'98)*, Aug. 26-28, 1998, Vienna, Austria. *IEEE Computer Society*, Vol. 8, pp. 300-308, 1998.
- [7] Kimball R., Reeves L., Ross M., Thornthwaite W. *The DWH Lifecycle Toolkit; Expert Methods for Designing, Developing, and Deploying DWHs*. John Wiley & Sons, Inc.
- [9] Milojicic D., Douglass F., Wheeler R. *Mobility: Processes, Computers and Agents*. Addison-Wesley, Bonn, Paris, Reading Massachusetts, 1999
- [11] Wu M., Buchmann A. P. Research Issues in Data Warehousing. *Proc. BTW'97*, Ulm, March 1997.

---

<sup>i</sup> This work has been funded by the Kplus program of the Austrian government, the province of Upper-Austria, and the Chamber of Commerce of Upper-Austria