Building Secure Knowledge Bases: Combining Java Agents and DBagents

Edgar Weippl
Software Competence Center Hagenberg
Hagenberg, A-4232 Austria
+43 7236 3343 837
edgar.weippl@scch.at

ABSTRACT
Today, databases are ubiquitous. In factories machine tools are connected to databases that process outstanding orders and log occurring errors. Our industrial partner AMS engineering builds and maintains machine tools for various client companies that operate worldwide. AMS' goal is to create a data warehouse that helps them analyzing machine failures and quickly launching response by reusing knowledge applied at one of their client's sites. Agents that roam through these databases are an efficient way of gathering the required information. As all operations heavily rely on database transactions, we propose to implement agents that are executed within a database management system. A Java-based agent environment allows encapsulating the database agents in Java agents to securely transmit them. Having arrived at the client's computer, the Java agent unpacks the database agent that performs the required tasks.

Categories and Subject Descriptors
C.2.4 [**Distributed Systems**]: Distributed databases

General Terms
Security

Keywords
Mobile Agents, Database Management Systems, Data Warehouse

1. INTRODUCTION
This paper presents a fundamentally revised architecture of DBagents [4]. The first paper on DBagents elaborated on how an agent runtime environment can be built *within* a database management system (DBMS). These agents perform the tasks necessary for building a consolidated data warehouse within a distributed database environment. This means that there are agents that migrate from database to database to extract and transform the data locally. The aggregated data is then transferred back to the agents' starting point, i.e. the data warehouse.

However, this first version of DBagents had inherent security problems. The database could not be effectively protected against malicious agents and malicious runtime environments were even more dangerous for the agents they host. For our industrial partner these security flaws did not matter as the database agents were used in a closed environment within one company. Organizational means were sufficient to provide a reasonable level of security.

As already mentioned we completely redesigned the system architecture for the current version of DBagents to address various security issues. This was necessary because our cooperation partner, AMS Engineering, later decided to use DBagents to build data warehouses including consolidated data of their clients. Obviously, the clients would not let agents originating from AMS collect and possibly modify data without security restrictions. We therefore decided to integrate already existing Java-based agent technology into our concept. This hybrid approach is elaborated in this paper. Compared to the first version of DBagents the migration and deployment process completely changed. Basically, the only thing that remained was the idea to execute agents directly inside the database management system and not to have an agent connect via JDBC.

AGENTS'01, May 28-June 1, 2001, Montréal, Quebec, Canada.

2. DBagents
In this paper we always mean *mobile agents* when referring to agents. There are other forms of agents (see [3], [1] for an exhaustive overview) like intelligent agents that we will not consider. Mobile agents can autonomously move their code base to other computers that offer host services, also referred to as runtime environments or as (Java) agent platforms.

2.1 Strengths
The agent's ability to migrate to other computers offers various advantages compared to static code.

First, data can be processed at the remote host and does not have to be transferred. AMS, our cooperation partner, for instance wants to know the average value of an item in all production databases.

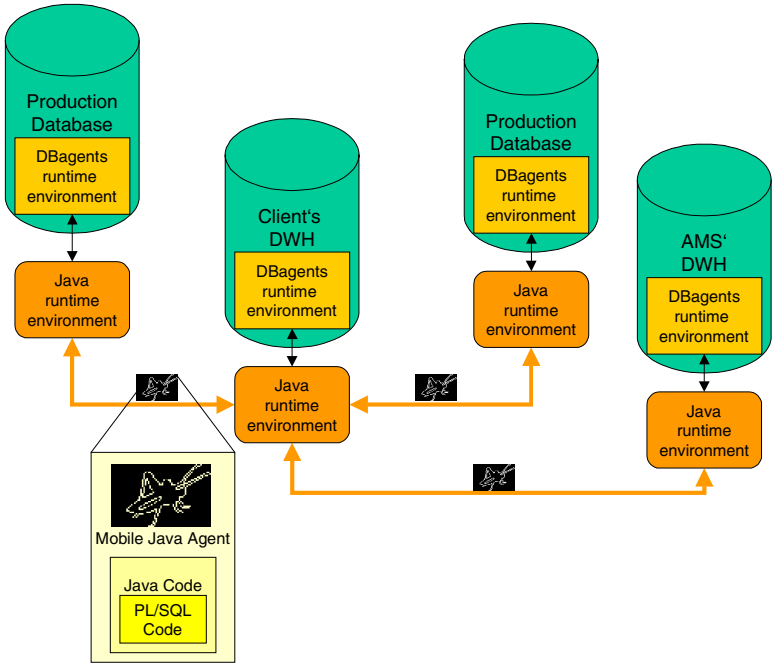

Figure 1: Java-based Agents migrate between database servers.

Figure 1 shows what the system's setup would look like. Without agents, AMS would have to connect to various databases and retrieve consolidated data. To be able to this, all values would have to be transferred via the network despite the fact that AMS will only store one single value, e.g. the average in its data warehouse. Agents, however, can migrate to the databases and process the data locally; they only have to return the average and the number of items over which the average was calculated. At AMS' data warehouse these averages can then be combined.

Second, the network connection does not have to be so reliable. Whenever the network is down, the agent has to wait for its return. As soon as the connection is re-established it will migrate home.

Third, DBagents can use the full functionality of the underlying database management system. Today, database management systems offer a wide range of services including transaction handling, recovery mechanisms, concurrency control and role-based security. These functions do not have to be included into the Java-based run-time environment.

Forth, DBagents is significantly easier to use than Java programming environments like Aglets or Grasshopper. DBagents provides means to issue database commands, which will be transparently wrapped into agents.

Finally, the Java-based version of DBagents significantly increases the security compared to the first version [4].

2.2 System Architecture

The DBagents system consists of five parts (see also Figure 1): the database, the DB runtime environment, the Java runtime environment (or the Java agent platform; in Grasshopper's jargon agency), Java agents and DBagents.

As the whole DBagents system is designed to optimize data processing within databases, the *database* is used as a basis to perform all operations. For our implementation we use Oracle but any other DBMS should also work.

Within every database, a *DB runtime environment* exists. This runtime environment receives DBagents that will be executed within the DBMS. For Oracle these DBagents are stored procedures, i.e. PL/SQL or Java programs that run within Oracle8*i*. The DB runtime environment takes care of creating, starting and removing the DBagents. It also makes sure that the results are communicated back to the Java Agent.

The *Java runtime environment* or *Java agent platform* hosts the incoming Java agents. The agencies (i.e. runtime environment) are part of IKV's Grasshopper Agent Development Platform.

We implement the *Java agents* using Grasshopper's mobile agent class. These agents are Java objects that can move from one host on the Internet to another. When the Java agent migrates to another host, it takes along its program code as well as its data.

DBagents are small pieces of software that can be executed directly by a database management system. For Oracle we use small PL/SQL procedures. These DBagents are encapsulated within Java agents while migrating. When a Java agent arrives, it unpacks the DBagent and connects to the DB runtime environment using JDBC. The DBagent code base is then transferred to the DB runtime environment. Finally, the Java agent then waits for the DB runtime environment to return the DBagent's results.

3. REFERENCES

[1] Bradshaw J.M, Software Agents, MIT Press, 1997.

[2] Milojicic D., Douglis F., Wheeler R. (1999) Mobility: Processes, Computers and Agents. Addison-Wesley, Bonn, Paris, Reading Massachusetts, 1999

[3] Nwana H.S., Software agents: an overview. Knowledge Engineering Review, 11(3), Oct/Nov 1996. 205-244, Cambridge University Press, URL: http://www.labs.bt.com /projects/agents/publish/papers/agentreview.htm, 1996.

[4] Weippl E., Altmann J., Essmayr W., Mobile Database Agents for Building Data Warehouses, International Workshop on Agent-Based Information Systems (ABIS2000), Greenwich, London, UK, A.M. Tjoa, R.R. Wagner, A. Al-Zobaidie (Eds.) IEEE Computer Society, ISBN 0-7695-0680-1. Sep 2000.