TOWARDS THE INFORMATION RETRIEVAL WITH MOBILE AGENT

Daniela Barreiro Claro   João Bosco Mangueira Sobral
Department of Computer Science and Statistics
Computer Science Pos-Graduate Course
Federal University of Santa Catarina
Florianópolis - Santa Catarina - Brazil
E-mail: {danclaro,bosco}@inf.ufsc.br

KEYWORDS
Mobile Agents, Agents, Information Retrieval, Database Integration, Databases.

ABSTRACT

Currently, the information retrieval has been reflected directly in the final products of an organization. The databases integration reduces in the costs of an organization, where the legacy systems and the new data storage technologies can be inter-related, supplying to the final users more consistent database information which proceed from a broader range of databases. So, the information retrieval between distinct databases will increase the data recuperation and permits facilities to users.

Besides bringing these benefits, the use of the mobile agents must allow a greater consistency in the data, since these agents move to the machines, and execute their tasks locally. This way, the data will not keep moving through the network, but will be stored in the mobile agents compartment, thus supplying one more level of security.

1. INTRODUCTION

Since the arrival of the first computer systems, the necessity to store and to recover data in a reliable and safe manner has become one of the main challenges recently.

Initially, the great peak of data storage occurred with the appearance of the relational databases (RDB). This caused the emergence of the data abstraction and the development of applications which were not directed to pointers or registers, but to a dataset.

Approximately one decade later, the object-oriented databases appeared, due to the complexity of the applications during that period of time, which were making the related databases inadequate for storage. The OODB (Object Oriented Database) adopted many of the concepts implemented in the programming languages which had this same paradigm. The main difference between the same was that, in the databases, the objects have persistent characteristics, while in the languages, the same are transient. Persistent objects have the basic characteristic of existing permanently in a database, so that they may be accessed by other programs and applications.

The basic concept of object oriented databases is the object itself. An object is an abstract representation of a real world entity.

Despite the advantages cited in relation to the OODB, these represent a very drastic change in relation to the technology currently adopted by the relational database (RDB). In addition to the fact that they are incompatible with previous technologies (RDB), the same also require new sets of abilities of the professionals in the area, as well as requiring specialized training.

With this context in mind the Relational- Object databases appears (RODB), to allow an integration of the current technologies with the concepts object oriented. In this sense, the ORDB represent a natural progress based on the benefits of the combination of a relational database with an object-oriented database (Fortier 1999).

2. DATA MANIPULATION

The manipulation of the data in these diverse databases occurs in different ways. With relation to a RDB, the access to the database is accomplished through the SQL-92 language that allows the insertion, deletion and data query through the structures called tables or relations.

With intention of extending the SQL-92 (SQL-2) standard, the SQL-3 language was developed. This means that the databases and the applications that use this standard will continue functioning, without any alteration, with the SQL-3 model. The extension occurs at the moment of the inclusion of the abstract data types, that allow the manipulation of the objects. This language is used in the accesses of object-relational databases.

The data storage, as well as its manipulation in an OODB occurs by means of the persistence of the objects, where the state of the objects can be stored in a non-volatile environment.

Due to the growth of the object-oriented paradigm and new forms of access to the database, not only object-relational but those oriented to objects as well, the integration of these databases with current relational technology is necessary.

The databases are being designed with similar structures, differing merely in their form of organization. In relation to the relational ones, tables are being analyzed and structured, containing the primary keys and their foreign keys when there is a need, as well as the code fields, which are responsible for the identification of the rows. In relation to the object-relational database, its form of storage is similar to the relational databases, differing in the form of access to the data, via SQL-3. However in regards to the Objects Oriented databases, the classes and objects are being defined according to the pre-established necessities.

3. MOBILE AGENT ENVIRONMENT

Due to the constant growth of computer technology in the world, many environments that long ago were active, now take on a new connotation, and begin to be called legacies. With this, new applications, new databases, new development environments have been growing commercially and changing the nature of developments and the manner of access to databases.

The difficulty of this process is, at first, the costs that the organization will have to expend with staff training, allowing the old employees to use the new technologies. It is also related to unproductive time, in the sense of migrating from the old systems to the new ones. In addition, there is the reorganization and modeling that systems analysts will have to conduct to guarantee a satisfactory migration in the end of the process.

With this in mind, the implantation of an appropriated environment must have the objective of integrating distinct databases, using, for instance, the mobile agents, where the same will search the local information, moving the code to the specific host to guarantee a better performance and integrity at the end of the process..

4. THE STRUCTURE OF THE ENVIRONMENT

The environment is made up of a local network, being able to reach the Internet environment, and is composed of three machines that contain the databases, and a machine responsible for the initialization of the agent. In this manner, only one user will be responsible starting the execution of the mobile agent. The environment contains the following structure in Figure 1.

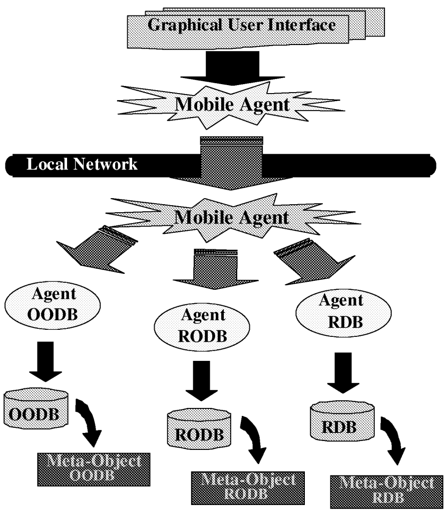


Figure 1. Structure of the Environment

The user interacts with the environment through a graphical interface, which shows the fields where the user can choose anyone and give this information to the mobility agent. So, this agent can perform the desired research on this information. After determining the selection fields, the mobile agent is gone off.

Once initialized, the mobile agent moves to the host present in its itinerary, where it communicates with the respective static agent to access the corresponding database. With this structure, the user's access to the databases is restricted, since the host which has the databases do not have direct access to the database, making it difficult to damage or poor manipulation of relevant information, and thus consequently, avoiding the inconsistency of the environment. Moreover, the agents had also been partitioned into mobile and static agents to increase the performance of the environment. If this were not done, the mobile agent would have to carry alone all the code necessary to access to the databases all by himself, increasing the information bottleneck in the network, thus overloading the mobile agent. Thus, in this case, the mobile agent is responsible for sending the requests and receiving the responses while the static agent is responsible for carrying through the access to the respective databases.

5. THE EXECUTION ENVIRONMENT

The execution environment is primarily responsible for the mobility of the agent inside the network. This migration occurs so the agent can carry the information and completely execute the task that was assigned to it. The first measure that the agent must take is to select the host that provides the service it needs. Although the agents carry with them a fixed itinerary, in the majority of the cases they must decide for themselves where to go. Mechanisms such as research for key words, or for semantic paths, that try to find the computer destination of an established agent based on the task it desires to execute, can assist the agents to find their service providers.

Beyond the synthesis of the agent's route choice, the execution environment is responsible for guaranteeing the mobility of the agent in the network. The mobility process involves two steps, the first one would be to send the agent from the origin host to the destination host and the other would be to receive the agent in the destination host. To carry the agent from the origin host to the destination host, it is necessary to suspend the execution of the agent, in the sense of interrupting the action that the same it is carrying out, and to codify the agent so that it can be transmitted and finally to liberate the resources that the agent was using in the origin host, making these available for other users, agents or environments. When the mobile agent arrives in the destination host, it is first necessary for the agent to be decoded, and immediately verifying if the feature which the agent needs is available in that machine. If it is, the agent is effectively excluded from its original execution environment. This transitory process occurs because, if for some reason, the agent does not reach the host desired, or suffers some external modification, it will return to its original environment, restructuring itself. In this manner, security aspects are involved to insure the integrity of the environment and of the agent as well (Chess et al. 1995).

For the transmission of the agents, protocols such as TCP/IP, HTTP, X.25 and SMTP can be used as inter-hosts agents carrier mechanisms, and the same are defined based on the execution environment in question.

The execution environments of the mobile agents must provide available resources control mechanisms. This type of control allows an efficient use of the resources, and prevents them from being inadequately manipulated by the agents. The agents can contain information relative to the necessities of their resources and the usage capacity of the same. When the agent is dispatched to another environment, the resources it had been using become then free.

In relation to the environment that is being developed, it was necessary to execute one of these execution environments on each machine present in the mobile agent's itinerary. Thus, an execution environment pertaining to this environment must be initialized on the station where the agent will be servant, as well as the respective execution environments on each machine that contains the databases. In this specific case, the agent will be directed according to the IP address of the machine involved and with the corresponding gate, since on one machine their may be two execution environments listening to distinct gates.

5.1 Mobile Agents

The use of an agent with the mobility features allows better use of the user available time to conduct the research, since it will be able to set-off the agent and then immediately disconnect from the network to carry out other tasks, thus getting the results at another moment when the connection is re-established.

Once the location is instanced and it's tasks have been carried out on the specific machine, the mobile agent prepares for the beginning of its itinerary carrying out its functions. With this purpose, the mobile agent carries by itself two objects: One which is responsible for the necessary requirements to execute the research, and the other one responsible for storing the results of the request to the database.

The object that will contain the requirements is supplied by the user, after the creation of the agent, and has as content the data of the fields present in the graphical interface, by which the client desires to use to conduct the research. With this object on hand, the agent invokes another method that is responsible for its movement to the next station where it searches the other results. After communicating with the static agent, which has the objective of sending through the object containing the solicitations obtained at the user interface, this database agent supplies as rollback a new object responsible for the storage of the result of the research in question. Getting these results, the mobile agent invokes the method responsible for its movement proceeds to carry out its task on the next assigned machine, where the same process, previously explained, will occur.

Each result obtained through the communication with the static agents to accesses to the relational databases, relational-object and oriented to objects will be added to the object at the mobile agent responsible for the storage of the results. Thus, at the end of the itinerary, the mobile agent will have all the resultant data of the requested research, when the same returns to the machine of origin. Arriving at the machine of origin, the same is responsible for sending the obtained information to the client, thus completing its execution goal.

5.2 Static Agents

The agents characterized as static are distinct according to the amount of databases of this environment. For each specific database a new static agent is instanced, that contains the functions necessary for the data access.

At the moment when the static agent is instantiated, a fact that occurs when the mobile agent arrives at the destination machine, the static receives the object containing the user research solicitations. With the solicitations at hand, the static agents consult the databases, using to conduct this, the functions pertaining to the database involved. Since these functions are multiple and divergent, it becomes necessary the implementation of a static agent for each type of database.

Another important feature of the static agents is the ability that the same have in working with managers of drivers and direct connections to the database. Since this task is unique to each database, and at the same time, not corresponding to the main functions of the static agents, one can perceive the need for the creation of a meta-object. This meta-object, situated in a meta-level of this application, is responsible for the definition of driver corresponding to the databases, and for the usage and accomplishment of the connection through the same. Since the drivers as well are specific to each database, as for example the driver used by Oracle is different from the driver used by Microsoft, which in turn is different from the driver used by ObjectStore, then the creation of meta-objects is realized for each static agent involved.

6. PARTIAL IMPLEMENTATION

The partial implementation has a main objective to serve as an example of the mobility characteristics of the code, thus clarifying the stages essential to its development.

6.1 The Platform

The platform is responsible to start the execution environment, thus allowing the mobility of the agent. After the evaluation of various platforms, among them the Aglets, Grasshopper, Concordia and Voyager, this last one was chosen mainly because of the availability of the features, since the other do not make the complete freeware available, thus inhibiting the developer from access to all the modules. Another important feature related to the Voyager, is that the same is completely developed in Java, with this, the use of this environment with the agents that may transit freely around the world, becomes evident, due to independence of the platform inherent to the language.

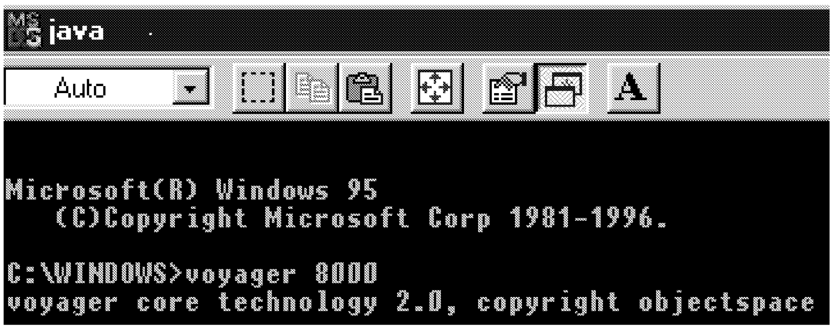
6.2 Configuration of the machines

Once the agent's execution environment is chosen, the configuration of the machine is conducted, where the same can be executed. Initially, it is necessary that the JDK (Java Developers Kit) be installed in a version newer than 1.1. Also, Voyager must be installed on all the machines that the agent will cover, since it will be responsible for initiating the execution environment.

To initiate the execution environment, one must only type in the DOS screen the command:

```
voyager 8000
```

With 8000 being the number of the agents communication gate. After this task is conducted, the execution environment shall be functional, as shown in the figure 2:



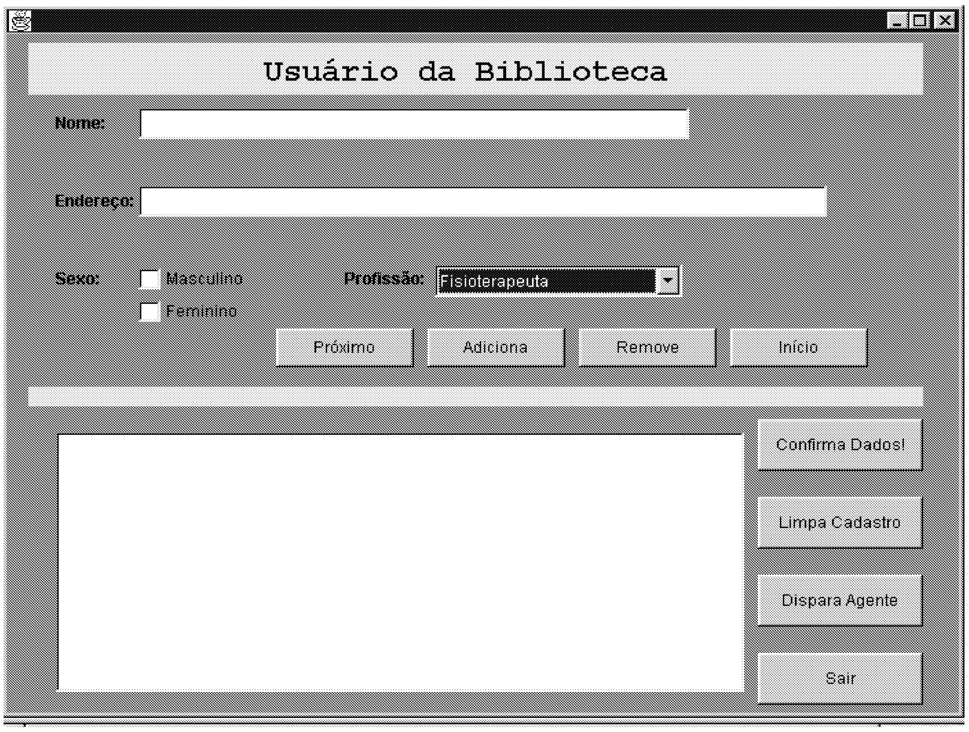
```
Microsoft(R) Windows 95
   (C)Copyright Microsoft Corp 1981-1996.

C:\WINDOWS>voyager 8000
voyager core technology 2.0, copyright objectspace
```

Figure 2. Execute Voyager Environment

6.3 The User

The user of this environment will be the responsible for setting-off the mobile agent to carry out the research in accordance with its necessities. For such, a graphical interface was developed, where the same selects the information that it desires to search, immediately to the setting-off the agent for its itinerary.



Usuário da Biblioteca

Nome:

Endereço:

Sexo: Masculino / Feminino

Profissão: Fisioterapeuta

Figure 3. Graphical interface to carry through the research as Agent.

In the example shown above, it can be observed that the mobile agent is being initialized and has as a requirement to search all the users who have a physiotherapy profession.

Internally, for the mobile agent to be created on a remote machine or a local machine, it is first necessary that its communication interface can be defined, that is, the methods that can be executed by remote objects.

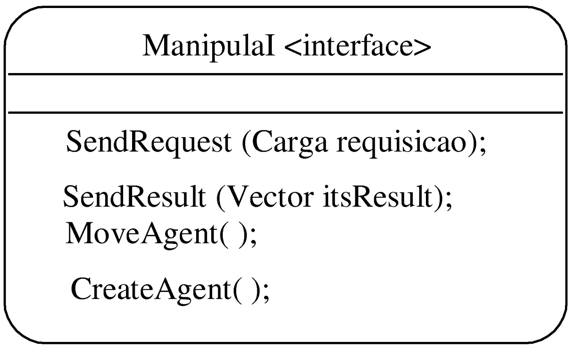

ManipulaI <interface>
SendRequest (Carga requisicao);
SendResult (Vector itsResult);
MoveAgent( );
CreateAgent( );

Figure 4. The ManipulaI Interface

The parameter passed through the method SendRequest, is an object of the Charge class, that contains the information supplied by the user through the graphical interface.

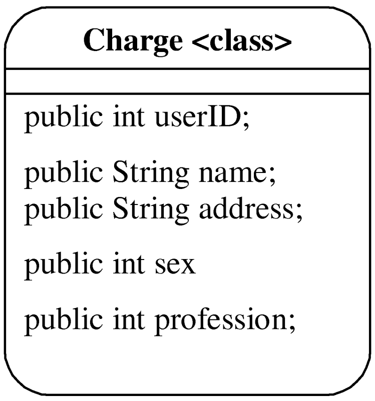


Figure 5. Charge class

6.4 The Mobile and Static Agents

The class called *MobileAgent* implements the agent that passes through the network in search of the information that the user requested. For this, it inherits the features of a Voyager class called *com.objectspace.voyager.agent.Agent*, thus supplying inherent functions to an agent, such as its mobility.

Beyond this, this *MobileAgent* class implements two other interfaces, the *ManipulaI*, previously characterized, and the *Serializable*, that supplies the necessary subsidies for the movement of the agents between machines in the network. So that for the agent may have the mobility between the machines, it is necessary that the developer clearly give a relation of his itinerary, describing his research path. For such, the use of the following string becomes necessary:

Agent.of(this).moveTo("//iris:8000","criaAgente");

In this case, there is an order for the agent to move to the machine called *iris*, and to communicate with the execution environment by means of the gate 8000, and after the same is created on the destination machine, it will have as a task to execute the "*createAgent*" method. This method shall be responsible for creating the *RBDAgent*, the static agent on the end or destination machine, that conducts the connection with the database, as well as the research, the rollback vector values.

The rollback generated by the *RBDAgent*, a denomination granted due to its relation to the static agent that conducts the research on the relational database to store it in an object called *Result*, instanced from the *Charge* class. The object *Result*, in turn, is organized in a vector and moved to the *MobileAgent*, as being the rollback carrier of this *MobileAgent*.

The RBDAgent is also responsible for the effective connection to the database, using JDBC. For such, it is necessary to create three classes which are inherent to the same: Connection, Statement and ResultSet. Through these classes the connection between the specific JDBC driver and the JDBC with the source of data obtained through the ODBC will allow access to the relational databases. After the *MobileAgent* passes through the relational database, the same will go to the other.

7. CONCLUSION

Many information storage technologies are being developed and used in organizations. However, the existing databases in these organizations cannot be rejected on a short term basis, thus creating a need to integrate various diverse databases using the code mobility.

The use of mobile agents allows the integration of these databases be transparent for the user, since the agent will be the main one responsible for the research of information stored in these databases. In addition, the mobile agents are also responsible for the communication with other agents, delegating the responsibility to access the databases to certain determined static agents.

In relational of Client/Server architecture, the mobile agent increase
It can be foreseen that the use of the mobility code for the integration of the databases will allow significant profits for the organization in terms of costs, performance and integrity of the information.

8. REFERENCES

Bushia, G.; Ferreira, J.E.1999. "Compartilhamento de Módulos de Bases de Dados Heterogêneas através de Objetos Integradores" *In proceedings of the 1999 Simpósio Brasileiro de Engenharia de Software* (Florianópolis,SC, Oct.13-15) Brazil, 129-145

Chess, D.M.; Harrison, C. G.;Lebine, D. 1995."Itinerant Agents for Mobile." Computing_IBM Reserach Report RC 20010, Research Division, IBM, 1995.

Fortier, P.J. 1999. *SQL-3 Implementing the Object-Relational Database*, McGraw-Hill.

Franklin, S. 1996.*Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents* Institute for Intelligent Systems, University of Memphis.

Horizon Systems Laboratory 1998 "Mobile Agent Computing: A White Paper". Technical Report, Mitsubishi Electric ITA, January (http://www.meitca.com/HSL/Projects/ Concordia)

Kotay, K; Kotz, D. 1994. *Transportable Agents.* Departament of Computer Science, Darmouth.

Melo, R.N.; Silva, S.D.; Tanaka, A.K.1998. *Banco de Dados em Aplicações Cliente-Servidor*, Infobook S.A.

Navathe, S.B.;Elmasri, R.1994. *Fundamentals of Database Systems*, Addison-Wesley, Second Edition.

Oshima, M.; Lange, D. B. 1998. *Programing and Deploying Java Mobile Agents with Aglets*, Addison-Wesley Publishing Company.

Reese, G.1997. *Database Programming with JDBC and Java*, O'Reilly.

Soler, O.O.;Prieto, F.M.;Gutiérrez, A.B.A. "A Reflective Persistence Middleware over na Object Oriented Database Engine." *In Proceedings of the 1999 Simpósio Brasileiro de Banco de Dados* (Florianópolis, SC,Oct11-13) Brazil, 137-151.

Voyager Core Technology Group. 1997."Agent-enhanced distributed computing for java". Technical report, - ObjectSpace, Inc.(Mar) http://www.objectspace.com/developers/voyager/white/index.html

9. BIOGRAPHY

Daniela Barreiro Claro is a student of Computer Science at UFSC - Federal University of Santa Catarina, in the area of Distributed Systems Engineering, She is bachelor in Computer Science for the UNIFACS - Salvador University. Currently, she works with computer systems and planning in private company and she has also worked at the Inter-Institutional Master Program of the Compute Science Graduate Course at Federal University of Santa Catarina. Her interest areas are Distributed Computing with Mobile Agents and Database Systems and its applications.

João Bosco M. Sobral is a professor at Federal University of Santa Catarina. He finished his doctorate at COPPE/Electrical Engineering Program of the Federal University of Rio de Janeiro (UFRJ) in 1996. He is MSc. in System Engineering and Computation by the COPPE/UFRJ since 1977 and he is undergraduate in mathematics by the Mathematic Institute/UFRJ (1973). Currently, he works at Department of Computer Science and Statistics and at the Graduate Course of the Federal University of Santa Catarina (UFSC). His interest areas are Distributed Computing, Mobile Agents and Computer Network Management.