A Web-based Workflow Management System *

Xinfeng Ye
Department of Computer Science
Auckland University
New Zealand
email: xinfeng@cs.auckland.ac.nz

Abstract

This paper describes a web-based workflow management system for a rental car company. Web browsers are the interface between the system and the users. The system consists of two types of tasks: transactional and non-transactional. The transactional tasks are atomic transactions. They can access more than one database. Thus, the system provides atomic operations on multiple databases.

1 Introduction

Workflow management systems provide an automated framework for managing intra- and inter-enterprise business processes [5]. This paper describes a web-based workflow management system for a rental car company. Web browsers are the interface between the system and the users. The system is Web-based due to two features of Web: the ubiquitous nature and the solid communication infrastructure. These features allow the users with any computing platform to be able to use the system without any additional hardware or special-purpose training.

There are two types of tasks in the system: transactional and non-transactional. The transactional tasks are atomic transactions. That is, in the presence of failure, the updates made by the transactional tasks are either all been carried out successfully or none of the updates been carried out. Since a task might access resources stored on different machines, to achieved atomicity, the two phase commit protocol is used for the transactional tasks. In order to make the system portable, all the tasks are written in Java.

The system provides easy access to database management systems (DBMSs). Since an organisation might have different types of DBMSs, in order to achieve maximum flexibility, access to databases is provided through JDBC API calls. As a result, a variety of DBMSs, e.g. Oracle, MS SQL Server, etc., can be incorporated into the system. Many existing web-based workflow management systems, e.g. [4, 6, 7], do not support atomic updates on multiple databases. The system in this paper allows transactional tasks access more than one database. Thus, the system provides atomic operations on multiple databases.

*This work is supported by Auckland University under grant A18/XXXXX/62090/F3414079.

2 The Structure of the Company

The company has several regional offices which are located at major cities across Australasia. The regional offices are responsible for dispatching and collecting cars to and from the customers. Each regional office keeps a database which records (a) the information on the cars (e.g. whether a car has been booked or dispatched) and (b) the reservations made by the customers. The head office oversees the operations of the regional offices. The head office has a marketing department which is responsible for analysing sales and reservation information. According to the analysis, the marketing department introduces packages targeting various customer groups.

Customers' queries which cannot be automatically processed are forwarded to the customer service center at the head office. These queries are submitted through Email or HTML forms. The center is responsible for handling the queries submitted by the customers.

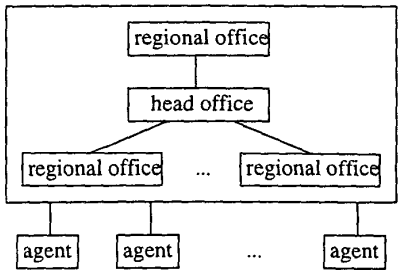

Figure 1 Company Structure

The company has contracts with many travel agents around the world. The travel agents are labeled as *agents* in Figure 1. The agents are responsible for handling reservations made by the customers and for collecting payments made by the customers.

3 The System

3.1 An Overview of the System

The structure of the system is shown in Figure 2. The system is web-based. The users use web browsers to make requests (e.g. make reservation and initiate tasks that handle the business within the company). There are two kinds of users in the system. One is the people from outside the company, e.g. the travel agents and the customers. The other is the company

0-7803-5731-0/99$10.00 ©1999 IEEE

employees. The outside users can only access the company's public web site, while the company employees can also access the company's private web site.

The head office maintains the company's public web site. The web site keeps all the information on the products of the company. Therefore, agents and customers can browse the product information through the Internet connection. While the head office keeps the personal information (e.g. the age group of the customers, the countries of origins of the customers, etc.), the regional offices maintain the reservation information (e.g. the type of car being reserved, the duration of the reservation, etc.). Reservations are sent by the users to the head office through the Internet. The head office processes the request and forwards the reservation to the relevant regional office[1]. On receipt of the reservation, the regional office updates its DB.

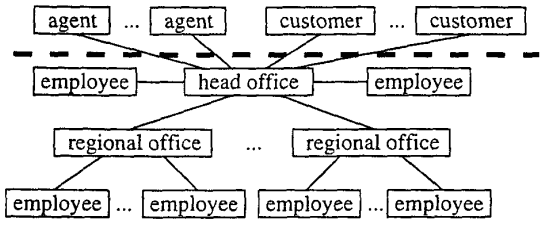

Figure 2 System Structure

This model might be more efficient if each regional office has its own web site. In this case, reservations made by agents or customers can be sent directly to the relevant regional office instead of being re-directed through the head office. However, this approach requires each regional office maintains its own web site. This is an extra cost to the company. When analysing the access pattern to the head office's web site, it is observed that the access spread reasonably well through the day. This is because the agents and the customers are all over the world. Due to the time difference between different parts of the world, the accesses to the head office's web site do not concentrate on certain period of time. Thus, the head office's web site does not appear to be an access bottleneck. As a result, redirecting the messages by the head office does not seem to degrade the performance of the system noticeably.

The head office and each of the regional offices has a private web site. The private web site stores various HTML forms for specifying and initiating tasks. When a company employee wants to initiate a task, the employee downloads the form from the local web site. After filling in the form, the employee submits the form. The form triggers the CGI programs on the local web site. The CGI programs carry out the task requested by the employee. The HTML forms and the CGI programs for processing these forms are the core of the workflow management system. The forms and the programs are replicated on each of the regional offices and the head office. This allows the users to download the forms locally. As a result, the access efficiency is improved.

A task might update the values of more than one resource in the system. These resources might be stored on different machines in the system. Since the machines and the communication links amongst the machines might fail, to maintain the consistency of the information stored in the system, some of the tasks must be made atomic. That is, either the updates are all carried out or none of the updates are carried out. The tasks which have atomicity property are called as transactional tasks. The other tasks are called non-transactional tasks.

The system handles the requests sent by parties outside the company (e.g. travel agents and customers) as well as the activities in the day to day running of the company. The tasks initiated by the users outside the company might trigger the tasks which require the attention of the company's employees (e.g. the tasks which are triggered when the customers submit queries) and vice versa. Therefore, the tasks initiated by different types of users interact with each other. However, since the procedures in processing the tasks initiated by different users are different, in the following, the tasks initiated by different types of users are discussed separately.

3.2 Handling the Tasks Initiated by Outside Users

First, the tasks initiated by the travel agents and the customers are discussed. Figure 3 shows the flow of control. The action starts when an user downloads an HTML form from the company's public web site (labeled 1 in Figure 3). The user submits the form to the company's web server (labeled 2 in Figure 3). The CGI programs on the web server process the form submitted by the user. The programs extract information from the form. Depending on the types of forms being submitted, the CGI programs invoke the following tasks:

- If the form is a query form[2] submitted by a potential customer, the information of the prospective customer are stored in the head office's database (labeled 3 in Figure 3). Some queries need access to the regional offices' databases (e.g. queries about the availability of cars). In this case, the queries are forwarded to the regional offices by the head office (labeled 4 in Figure 3).

- If the form is a reservation form, the information of the customer is stored in the head office's database (labeled 3 in Figure 3) and the reservation is forwarded to the relevant regional office and added to the regional office's database (labeled 4 and 5 in Figure 3).

- If the form is a cancellation form, the information of the customer recorded in the head office's database must be removed (labeled 3 in Figure

[1]The request is forwarded to the regional office where the customer wants to pick up the car.

[2]An query form allows the users to specify (a) their personal details, (b) the expected travel time, (c) the type of cars required and (d) their specific requirements.

3), and the reservation must be deleted from the relevant regional office's database (labeled 4 and 5 in Figure 3).

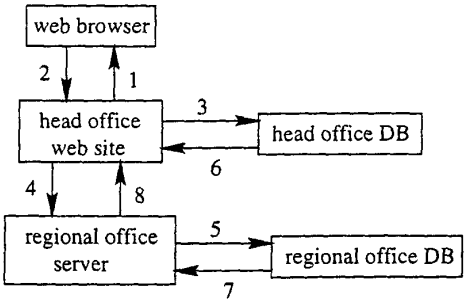

Figure 3 Tasks Initiated by Outside Users

The CGI programs for processing the forms are written in Java. This allows the system to be easily ported to various platforms. ODBC is a standard low level API and is available with many of today's popular DBMSs (e.g. Oracle and MS SQL Server). Java includes a package which converts API calls in JDBC to API calls in ODBC. Therefore, applications with JDBC API calls can be used to access a wide range of DBMSs. Due to this reason, the CGI programs in the system use APIs provided in JDBC to query and update the databases in the system.

Query form and reservation form both include the personal details of the customers. These information are collected and stored in the head office's database. The marketing department will use these data to analyse the trend of the sales.

Some of the queries can be processed automatically (e.g. a query which checks the availability of a type of car). In this case, the results of the queries are sent back to the travel agents or the customers immediately. The query results are sent to the users in HTML forms. Thus, the users can read the results using web browsers. If a query requires the participation of the regional offices, the regional offices make JDBC API calls to query the databases (labeled 5 and 7 in Figure 3). The results of the queries are sent back to the head office (labeled 8 in Figure 3). The head office constructs an HTML form holding the result of the query and sends the form to the customers.

Queries which cannot be processed automatically (e.g. a customer requests for items which are not included in any standard package) trigger the tasks which notify the service center staff. The staff process these queries manually. The need specified by the customers in the queries will be stored in a database. When new products are introduced by the company, the database is checked to see whether the new products suit the requirements of the prospective customers. The customers whose needs are matched by the new products are made aware of the introduction of the new products either by email or by post.

3.2.1 Enforcing Atomicity

Reservation and cancellation tasks need the participation of the head office and the regional offices. In this case, the head office forwards the information to the relevant regional office. A client/server model shown in Figure 4 is used to implement these operations.

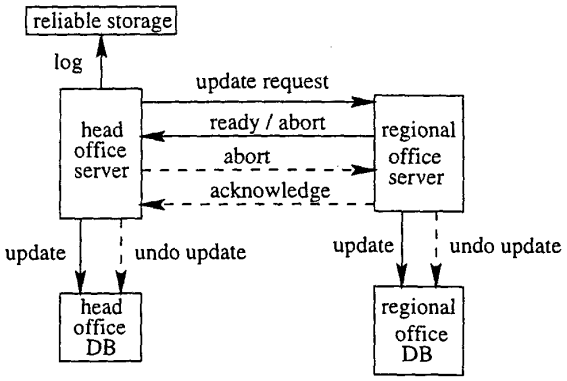

Figure 4 Enforcing Atomicity

A server process (written in Java) runs on each of the regional offices' machines. The process waits for messages from the head office. When a reservation or cancellation is made, the databases at the head office and the relevant regional office are updated simultaneously. However, the updates to the databases might not be carried out successfully due to machine crash or communication failure. In order to ensure the coherency of the information stored in the databases of the head office and the regional offices, the update operations must be made as transactional tasks.

The two phase commit protocol [2] is normally used to achieve update atomicity. However, the protocol causes long delays. To reduce the delays in using the two phase commit protocol, the semantic of the tasks can be explored. It can be seen that the reservation and cancellation tasks only involve inserting records into and deleting records from the databases. These tasks do not influence or depend on other tasks. Therefore, there is no strict ordering constrain in carrying out the insertion and the deletion operations when they are mixed with the other tasks. Thus, an augmented two phase commit protocol is used to guarantee the update atomicity for reservation and cancellation tasks. The protocol is described below and shown in Figure 4. In Figure 4, the solid arrows represent the normal operations and the dashed arrows represent the operations which are issued in the presence of a failure.

When a reservation or a cancellation task is initiated, the head office sets up a link with the server process running in the relevant regional office's machine. Before update requests are sent to the databases, the head office machine sets up a log to record the updates to be carried out. The log is stored on a reliable storage device. Thus, the log can always survive a failure. Then, the update requests are sent to the re-

gional office. The processes at the head office and the regional office will make JDBC API calls to update the databases.

The updates at the regional office might not be carried out successfully due to three reasons:

1. The communication link between the head office and the regional office is broken.
2. The machine at the regional office crashes.
3. The booking cannot be carried out due to the un-availability of the cars.

To discover the occurrence of problem 1 and 2, the process running at the head office polls the process running at the regional office periodically. If a reply is not received within the timeout period, it is assumed that the update cannot be carried out by the regional office's machine. In this case, the updates which have already been carried out by the head office's database must be undone. The process at the head office issues JDBC API calls to undo the update.

If a reservation cannot be carried out successfully due to system failure, the person who submitted the reservation will be notified and be asked whether he/she wants to keep the reservation. If the reservation is kept, when the failure is repaired, the process at the head office's machine will use the information kept in the log to make another attempt to update the databases. If the update is successful, the customer will be notified. A customer might decide to cancel a reservation which cannot be made successfully due to system failure. In this case, when the system returns to normal operation, the process on the head office's machine will send an abort request to instruct the machine in the regional office to cancel the updates to the database which were carried out before the failure occurred. When the machine at the regional office has undone the updates to the database, it acknowledges the head office. At this point, the log for the task can be deleted.

If the updates are carried out successfully by the regional office without being interrupted by system failure, a ready message is sent to the head office to notify the completion of the updates. Since the updates have been carried out successfully on both the head office's and the regional office's databases, the log of the task is deleted.

Apart from system failure, the updates to the regional office's database might also be aborted if there are no cars available to meet the customers need. The process running on regional office's machine first checks the availability of the cars requested in the reservation. The updates to the database will not be made unless a car is available to satisfy the customer's request. If there are no cars available, the head office is notified by an abort message. As a result, the head office undoes the updates to the head office's database.

The machine at the head office might crash as well. If the crash occurred before the log of the task is set up, the databases at the head office and the regional office have not been updated. This is because updates to the databases only occur after the log has been set up. In this case, no action needs to be taken when the system recovers from the failure. If the failure of the head office's machine occurs after the log is set up, the databases at the head office and the regional office might have been (partially) updated depending on whether the update requests have been sent to the databases before the failure occurs. As a result, the updates to the databases must be undone first.

When the head office's machine recovers from failure, it checks the log of the task. According to the log, the head office machine issues instructions to its own database and the regional office to undo the updates which have been carried out before the failure occurs[3]. Then, the updates recorded in the log are sent to the appropriate databases; so that the updates can be re-executed.

3.3 Managing the Workflow within the Company

The structure of the system for managing the workflow within the company is shown in Figure 5. The system is web based. Users initiate tasks by submitting HTML forms to the system. The head office and each regional office is called a site. Each site has a web site which is only accessible to the machines at that site. The site stores various forms that are used to initiate the tasks. The users need to fill in some of the forms. The information contained in the forms are used as the parameters passed to the tasks to be initiated. For example, a form which initiates a task for arranging a meeting should specify with whom the meeting should be arranged. Each site has a task manager. The manager is responsible for (a) handling the workflow at that site and (b) communicating and co-operating with the task managers at other sites.

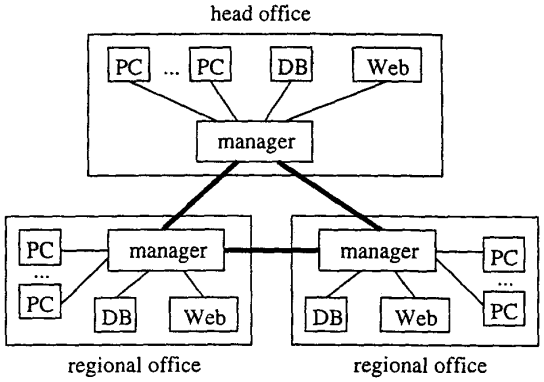

Figure 5 Managing Workflow within the Company

[3]The undo operations can be carried out by applying the reverse operations of the update operations. If the machines in the regional offices do not fail while the head office machine crashes, the databases at the regional offices are updated properly. As a result, it is unnecessary to undo these updates. However, the regional offices' machines might also fail. This means there is no guarantee that the updates to the databases at the regional offices have been carried out successfully. Therefore, to ensure information coherency, when the head office's machine recovers, the updates to the regional offices' databases are also undone.

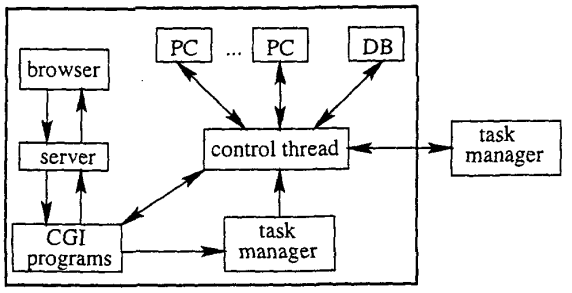

Figure 6 The Control flow of a Task

The activities involved in using the system is shown in Figure 6. First, the user uses a browser to download an HTML form which is used to specify and trigger a task. After filling in the form, the user submits the form to the local web site. The form is processed by the CGI programs. The CGI programs first register the task with the task manager. The manager creates a Java thread which is responsible for coordinating the task's access to the resources in the company. If the task need to access resources at other sites, the thread will be responsible for passing the requests of the task to the task manager of the relevant sites. If necessary, the results of the execution of the task are returned to the user.

Here is an example which illustrates how a task is handled in the system. A secretary at regional office A initiates a task which arranges a meeting for the directors of regional office A and B. The schedules of the two directors are stored in the databases at their respective site. In order to find a meeting time, the schedules of the two directors must be checked to find a time which suit both directors. Thus, the task requires access to the databases at site A and B. When the task manager at site A receives the task, it creates a thread TA to handle the task. Thread TA contacts the task manager at site B. The task manager at site B creates a thread TB to handle the interaction with TA. TA and TB access the database at their local site to find out the time which are available for the two directors to meet. Then, TA and TB updates the schedules of the directors and send messages to notify the directors and other parties that are concerned.

3.3.1 Handling Deadlock

Some of the tasks update databases at different sites. The execution of some of these tasks requires the tasks have exclusive access to the databases. Otherwise, errors might occur. For example, assume the secretaries of site A and C both initiate a task for arranging a meeting between their directors and site B's director. The task manager at site B creates two threads, say AB and CB, to handle the two tasks. Clearly, AB and CB should not access the schedule of director B simultaneously. This is because, if AB and CB access the schedule simultaneously, they might allocate the same time slot for director B to meet with director A and C. Thus, AB and CB must lock the schedule to prevent other threads from accessing the schedule. The task manager on each site is responsible for maintaining a table which records the locks on various local resources. A thread must register its lock on a resource in the table before it accesses the resource. If the resource has been locked, the thread must wait until the resource becomes available.

Although locking guarantees the exclusive access to shared resources, it might also cause deadlock. For example, assume that:

1. Task X tries to set up a meeting between director A and B has been initiated at site A.
2. Task Y for setting up a meeting between director A and C has been initiated at site C.
3. Task Z for setting up a meeting between director B and C has been initiated at site C.
4. Task X has locked director A's schedule at site A.
5. Task Y has locked director C's schedule at site C.
6. Task Z has locked director B's schedule at site B.

It can be seen that task X is waiting to access director B's schedule which is locked by task Z. Director B's schedule would not be available to task X until task Z completes its execution. However, task Z has been blocked waiting to access director C's schedule which is locked by task Y. Task Y cannot make progress since it is waiting for task X to release its lock on director A's schedule. It can be seen that none of the tasks can be completed.

In order to detect deadlock, the algorithm described in [3] is used. In the algorithm, a wait-for graph representing the dependency amongst the tasks is constructed during the execution. If there is a cycle in the wait-for graph, a deadlock exists. In the system, if a task has been blocked for a certain period of time, a Java thread is created to look for cycles in the wait-for graph. If a cycle is found, one of the tasks in the cycle is aborted. As a result, the resources locked by the task are released and the other tasks waiting for the resources are able to make progress. A wait-for graph might span across several sites. Thus, the task managers must cooperate with each other during the execution of the deadlock detection algorithm.

3.3.2 Enforcing Atomicity

Some of the transactional tasks update more than one database, For example, the task for arranging a meeting between two directors needs to update the databases at two different sites. Generally speaking, unlike the tasks for handling reservation and cancellation, the execution of a task influences the outcome of the other tasks. Therefore, the order in carrying out the tasks is significant. To ensure atomicity, the two phase commit protocol is used. The site which initiates the task plays the role of the coordinator in the two phase commit protocol. First, the resources used

by the task are locked on all the sites. The locks will not be released until the two phase commit protocol is completed. Then, each site sets up a log to record the updates to be carried out. The log is stored in the stable storage on each site. After the log has been stored, the site notifies the coordinator. When the coordinator has been notified by all the sites, the coordinator records the identities of the participating sites in reliable storage. Then, the coordinator instructs all the sites to carry out the updates on the databases. The site releases its lock on the resource after the update is completed. Since the updates have been logged by all the participating sites, as shown in [2], it can be guaranteed that the updates can be carried out on all the sites. If a site fails before notifying the coordinator that the site has logged the updates operations in the stable storage, the coordinator instructs all the other sites to discard the update operations. In this case, the coordinator will re-start the task again when the failed site recovers.

3.3.3 *Utilising Computing Resources*

Apart from automating the workflow within the company, the system also tries to utilise the computing resources within the company. Some of the tasks are computation intensive. For example, the task for analysing (mining) the sales data to discover market trend is an I/O and CPU intensive task. If the task is run on a single machine, it takes many hours for the task to be completed. In fact, a majority of the machines in the company are idle most of the time. If these machines are utilised to carry out the computation intensive tasks, then (a) the time to complete the tasks can be reduce, and (b) the cost effectiveness of the computer systems can be improved.

To utilise the system resources, the task manager monitors the load of the machines at the local site. The manager assigns the idle machines to work on the compute intensive tasks. To allow a task to be executed by several machines, the task must have a parallel implementation. At the moment, the task which can be run in parallel is the one used by the marketing department for analysing market trend. The task is used to mining association rules [1], e.g. whether most German customers in age group 25-30 make their journey during summer. The job is implemented based on the algorithm in [8].

4 Conclusions

The workflow management system described in this paper combines the managing of workflow within the company with e-commence facilities. The system is designed for heterogeneous distributed environment. Thus, the system is web based and the programs for carrying out the operations are written in Java. Users use web browsers to interact with the system. The tasks are stored on the web servers as CGI programs. These features allow the system to be run on a variety of platforms.

The system uses JDBC API calls to access DBMSs. Since (a) JDBC calls can be converted to ODBC calls by a package included in Java, and (b) most DBMSs support ODBC calls, the system is able to access a wide range of commercial DBMSs, e.g. Oracle, MS SQL Server etc. Thus, it gives more flexibility to the company in choosing database products.

Tasks in the system can be transactional or non-transactional. Transactional tasks guarantee update atomicity. Two phase commit protocol is used to guarantee the update atomicity. As a result, during the execution of the transactional tasks, some of the information need to be saved to stable storage. Thus, transactional tasks are not as efficient as their non-transactional counterparts. The system improves the efficiency of some tasks by analysing the semantics of the tasks. Since the updates carried out by some of the tasks can be undone easily and the undone does not affect the other tasks, an augmented two phase commit protocol which requires less message exchange is used for some of the transactional tasks.

The system uses parallel processing techniques to speed up the processing of the compute intensive tasks. Instead of using expensive parallel computers, the system uses a pool of idle PCs to carry out parallel computation. As a result, the utilisation of the computer resources has been improved.

References

[1] R. Agrawal, T. Imielinski and A. Swami, Mining association rules between sets of items in large databases, *Proc. of ACM SIGMOD Conference*, pp. 207-216, 1993

[2] Bernstein P.A., Hadzilacos V. and Goodman N.: concurrency control and recovery in database systems, Addison-Wesley, 1987

[3] Chandy K.M., Misra J and Haas L.M.: Distributed deadlock detection, ACM Transactions on Computer Systems, Vol. 1, No. 2, pp144-156, 1983

[4] Information Management Consultants: WebFlo - delivery imaging and wokflow over the Web, Technical Report, Information Management Consultants, 1997

[5] Dogac A., Kalinichenko L., Ozsu M.T. and Sheth A.: Workflow management systems and interoperability, Springer, 1998

[6] Palaniswami D.V.: Development of WebWork, MSc Thesis, Georgia University, 1997

[7] Action Technologies: Metro 3.0 coordinates work across the Web, Technical Report, Action Technologies, 1997

[8] Ye X. and Keane J. A.: Mining Composite Items in Association Rules, Proc. of 1997 IEEE International Conference on Systems, Man and Cybernetics, pp. 1367-1372