MIGRATED LEGACY DATA ISSUES: RECOVERY, CONCURRENCY & SECURITY

Er. Muheet Ahmed Butt
Scientist Directorate of Information Technology & Support Systems, University of Kashmir, Srinagar, J&K, India
ermuheet@gmail.com

Er. Majid Zaman
Scientist Directorate of Information Technology & Support Systems, University of Kashmir, Srinagar, J&K, India
zamanmajid@rediffmail.com

Dr. S. M. K. Quadri
Head & Director, PG Department of Computer Science, University of Kashmir, Srinagar, Srinagar, J&K, India

Abstract: This paper presents the management and rectification of the data pertaining to an Enterprise Information System after being migrated from Legacy Information sources. The experience from using this research shows that the basic parameters like Recovery, Concurrency and Security could be attained in a better way after migration to new Systems from Legacy Systems. A major anxiety at the first glance is about data retaining its integrity at a large extent as was being attained in Legacy Systems also. At present, the Data Warehouse used by the Enterprise under research is the centralized database that it uses for many tasks pertaining to Finance, Management, Marketing, Human Resource Development Personnel Management, etc. Evidently, there is not much data protection control available. Hence, if something adversely happened, the Central Data Warehouse will be directly affected and may lead to a system crash especially by those users who are not aware of the mistake when using the new migrated system. Unexpected and unexplored problems might also occur and may need a tremendous amount of work to rectify such data inconsistencies after migration.

INTRODUCTION

The enterprise under study uses Microsoft SQL Server 2005 as a database management program for its Data Warehouse Solution. Due to the growth of the number of employees and customers associated with an enterprise, many new business rules have to be implemented which are well served for the use of the computerized database. However flaws in the database have shown up from time to time caused by users who are inexperienced in managing Information Technology Infrastructure in a proper way or are doing it intentionally. Consider the case when two processes executing interfere with one another, thereby producing incorrect results. Sensitive data might be exposed or worse, changed by unauthorized users. The fact is there are indeed many risks that the data might be exposed to.

The other case concerns updates which might change data illegally. [1] Therefore the system has to provide an extensive support to protect the database against such threats which may arise while recovery, concurrency, security and integrity measures are being implemented on a database. For the stability of the database, using one central data warehouse for enterprise data source is perilous since some situations may adversely happen, such as system crash. Since only one central data warehouse is used, the situation may leave the database in an incorrect state in which recovery may not be possible. Consequently, one way to ensure a recoverable database is to certain that every piece of information it contains can be reconstructed from some other information stored in other place redundancy. The solution we provide is that we create another parallel Data Warehouse (TDW) which is used only during the transaction period of the enterprise. We will refer to this new Central Data Warehouse (CDW) as the central database as the main database throughout the paper.

PROBLEMS PERTAINING TO NEW SYSTEMS AFTER MIGRATING FROM LEGACY SOURCES

The various problems pertaining to new systems after migration from legacy sources are discussed below:

Problem Pertainning to Recovery:

Before we go into the details of why we need recovery controls, we would first like to clarify the meaning of recovery. In Date “swords” [1], recovery is depicted as “Recovery in database system means, primarily, recovering the database itself—that is, restoring the database to a state that is known to be correct after some failure has rendered the current state incorrect, or at least suspicious.” There are several possible reasons for a transaction to fail in the middle of the execution. For example, system crash may cause error in the computer system during transaction execution. Some transactions might violate the concurrency control enforcement and the control may decide to abort the transaction because it violates serializability or because several transactions are in a state of deadlock.

Physical problems (media failure) may happen such as head crash on the disk, fire or sabotage or physical theft. For a system crash, the content in the buffer memory is a critical point. The state of any transaction in a progress is not known; such a transaction did not successfully complete, and so must be undone (rolled back) when the system restarts. For example, while some users are updating the record a power failure occurs. Hence those unfinished transactions must be rolled back to its previous state. For data protection when media failure occurs, the backup copy of registration database is needed for restoration; there is no need for a roll back.

Problems Pertainning Concurrency:
Enterprise Information System’s Data Warehouse is a shared resource. It is assumed at a particular instance of time many concurrent users will be accessing the database for doing read and write operations on data. With concurrent processing involving updating the data in parallel, a database without concurrency control will be compromised due to interference between users. Concurrency control allows many users to access and update the database simultaneously while preventing partially completed updates from happening[2][3]. This technique is essential to our Enterprise Information System, such as when more than one user is registering a customer in the same groups or service with a limited number of available spaces for customers. When the user decides to register a customer for the service, the database integrity parameters will check first whether or not the Service Group is full by having one variable used to store the number of Customers per group who have registered for that service. While in a multiuser structure it can lead to a violation of data Integrity. If the problem of violation takes place then it is not only last updates that concurrency control mechanism has to address, uncommitted dependency and inconsistent analysis problems are also possible. These problems can cause the database to be in inconsistency state [4].

Problems Pertaining Security:
The unauthorized access can modify the sensitive data pertaining to both employees and customers so emphasis has to be given to how proper roles and privileges are given to the users so that the security of the data is not violated [7]. It can be concluded that Security concerns are implemented so as to ensure that users can do only what they are allowed to do and nothing more. In the EIS data, all information concerned with every customer and employee is kept in it. If no security system is implemented into the system, tremendous problems will arise since a user might alter the sensitive data of the enterprise. Hence a security system is needed for maintaining a usable database for EIS[6]. Some constraints must be enforced to ensure that authorized users are doing correct operations, satisfying all constraints.

PROPOSED SOLUTIONS

By using a central database for Enterprise Information Systems, many problems are likely to occur due to unaware or deliberate action. All the errors will be adversely directed to the central database Log. With the great importance that the central database/warehouse is used for most enterprises for performing various tasks, a small Parallel Warehouse (TDW) is implemented in order to be used as a substitute during the transaction period of the Enterprise. So any adverse effect will only be confined to the small enterprise CDW [8][9]. Consequently, advantages concerning security are also provided. Moreover, by separating the this warehouse out, the enterprise can modify or change the structure of the database easily, without too much concern about its side effect on the CDW.

The TDW can be generated easily from the CDW by transferring only the necessary information and Structures. Some information is transformed into more appropriate form just like EMP, dept, finance, log schemas and relationships, etc. This information will be transformed into the form of which can be temporarily used in the system. Any unauthorized users who try to change these details will fail since no as the changes are not permanent for the CDW. It is also more convenient for our enterprise database since the transformed transaction is more relevant to the objective of performing transaction. The transactions are consolidated in this TDW and when the transaction operations are completed the data is reflected in the CDW. During every export a log database is maintained keeping track of new records added and the tables which are being modified. So every time the export takes place only those data tables are modified which have an update associated in the transaction and the other data table are kept un-altered.

Proposed Solution for Recovery:
Since we are using separate databases any failure that might cause the database to corrupt is now limited only to the TDW, leaving the CDW untouched. For the case that TDW is collapsed, it can easily be reconstructed within three minutes, since it contains only information Transaction details, it is then ready to be used again to provide uninterrupted service during the transaction period. Furthermore, for any terminated transaction. The transaction manager is used to provide the atomicity of important transactions. In other words, it guarantees that if the transaction executes some updates and a failure occurs (whatever the cause) before the transaction finishes (reach its plan), then those updates will be undone. Thus, the transaction either executes in its entirety or is totally cancelled. In this way a sequence of operations that is fundamentally non-atomic can be viewed as if it were atomic from this point of view. The commit transaction and rollback transaction are the key to the way recovery works. For commit transaction, it tells the Database Management System (DBMS) that the atomicity of the process has been thoroughly finished. The database is in a consistent state and the updates made by the process can now be made that is fundamentally non-atomic can be viewed as if it were atomic from this point of view.

The commit transaction and rollback transaction are the key to the way recovery works [10][11]. For commit transaction, it tells the Database Management System (DBMS) that the atomicity of the process has been thoroughly finished. The database is in a consistent state and the updates made by the process can now be made permanent or committed. In contrast, the signal of failure to end the transaction is indicated by the rollback transaction. The database might be in an inconsistent state and the updates by that transaction must be undone or rollback. A log will be maintained by the system about the details of all update operations. So, if it is necessary to undo any specific update, a log file will be used to update value to its previous value. For EIS, the technique of commit and rollback transaction is implemented in Microsoft Visual Basic .NET for SQL Server 2005 by using the reserved words Begin Trans, Commit Trans and Rollback. These three functions are used for important transactions that might be able to compromise the consistency of the database. For example, by using these functions, when user decides to register a customer, the program adds the customer to the EIS record and updates the number of Customers in Service Group with a fallback recovery against any failure. A clearer idea of the
importance of recovery might be depicted when the transaction is concerned with payment. Therefore transaction should be made as atomicity in order to avoid the inconsistency of the database

**Proposed Solution for Concurrency:**
As mentioned before, without concurrency control the problems of lost updates, uncommitted dependency, and inconsistent analysis are expected to occur. Since the CDW is a shared and classified resource, careful database management must be incorporated. Microsoft SQL Server 2005 provides three levels of locking. These are record locking, table and record set locking, and opening with exclusive access. For record locking, only the record currently being edited is locked. Fordable and record set locking, an entire table or all tables underlying a form are locked while any user is editing any record in the form. Finally, for opening with exclusive access, the entire database is locked by a single user-change into single-user environment. Microsoft SQL Server 2005 automatically locks the record currently being edited even though the programmer did not predefine the lock mechanism [12][13].

In TDW, since we always operate under multiuser environment, the opening with exclusive access is irrelevant and will not be mentioned twice. The most used mechanism in our CDW is table and recordsetlocking. Since this database is a relational database and has a quite complicated relation and query, record set will be used for most of the time. As mentioned before about concurrency, the inconsistency about counting the number of customers class can be solved by using tableland record set locking which will be referenced as lock. As the Customers decide to register any Group, the lock is made active so that counting the number of Customers in each Group is correct. The reason why the original system locks the entire table is that it uses one table to store all records of who had registered for which courses. Consequently, the counting method also uses this table to count the number of customers for each class. In other words, from the CDW uses the table "REGEMP" to store and count the number of customers in each course by using Customer ID and Group ID as criteria for counting the number of Customers registered in each group. In our new system, the TDW stores records in table "TCusTRegradable" which is then used to count the number of customers in each course, DCount is used in Visual Basic.NET code. Hence, if these tables are locked while any user is currently updating then, the lost updates problem can be solved.

**Proposed Solution for Security:**
In the past, the original system used table TUSERS to store login name and password for the super users. As the user log on to the database, the application only verifies login name and password using ordinary Visual Basic.NET code to check the information in TUSERS. User Lookup is used to check the data in the table. Hence, if anyone is able to look into this table, he can find the login names and passwords easily. If the Transaction database is not encrypted, anyone with a disk editor can view the contents of the file. Although the data within the file will not appear in an easy-to-read format, the data is there and available for unauthorized individuals to see. Therefore, the encryption is used for the Transaction database even though the performance of the application will drop but it is necessary to keep it encrypted. In other words, another level of security is made from encrypting the database. Typically, the DBMS supports either or both of two broad approaches to data security [14][15]. The approaches are known as discretionary and mandatory control. In the case of discretionary control, a given user will get different authority or privilege.

Discretionary schemes are very flexible. In contrast to discretionary, mandatory control defines each data object to be labelled with a certain classification level, and each user is given a specific level of clearance. A given data object can then be accessed only by users with the appropriate clearance. Consequently, mandatory control is rigid but appropriate to use with an EIS database [16]. It is easier and clearer to maintain a class of users than to concentrate on individual users, since every customer must have the same right. Login name and password table will not be used; TUSER and new approach should be used here. Microsoft SQL Server 2005 provides a very powerful and comprehensive feature to maintain user account. The information on each clearance level of user and password, and access right for each object will be stored in the system file. This file is distinct from the database file which Microsoft SQL Server uses to store information database security. The privileges of each level will be discussed in the next section.

**Solution for Integrity:**
As mentioned before, integrity concerns protection against authorized users. For customers they must be assigned the rights to read all that data and update only the table TREGISTER since this table is used to store who is registering which services. Customer must not be able to view database application asin the design view to avoid any adverse alteration by them. The other thing of concerns that customer should not be allowed to use the toolbar since it provides features beyond the necessity of customer registration. For the super class of users, it depends on the policy of the Enterprise to what the registrar is allowed to do as does the higher Admin of users [17]. The other policies such as how many credits customers mustenroll in each group, the maximum number of Services a Customer can register each Group, prerequisite, co-requisite, etc, are deliberately ignored from user privileges levels since it varies from organization to organization.

**CONCLUSION**
So far, all fundamental problems have been cleared up. Recovery, Concurrency and Security of the Enterprise Information System Data Warehouse have been solved. However, a lot of delicate improvements can be implemented because the further use of this CDW in the future will tell what is appropriate and what should be improved. User-interface is another area for improvement since it can reduce the error that users might make but requires further work to produce an appropriate and elegant design that suits the user’s needs.
REFERENCE


