Database for Student Registration System at SIIT

Nakarin Netcharussaeng, Nichalin Suakkaphong
Sirindhorn International Institute of Technology
Thammasat University
Pathum Thani 12121, Thailand

Apichat Tungthangthum, Pichet Chintrakulchai
Asian University of Science and Technology
Chonburi 20261, Thailand

Abstract

This paper presents the rectification of the student registration database systems of Sirindhorn International Institute of Technology (SIIT). The experience from using this registration system shows that many amendments can be attained in order to benefit the users and the institute. A major anxiety at the first glance is about data protection. At present, the database used for student registration is the central database that the institute uses for many tasks such as student registration, student academic record, student profile, etc. Evidently, there is not much data protection control. Hence, if something adversely happened, the central database will be directly affected and may collapse specially by those users who are not aware of the mistake when using computer-based registration. Unexpected problems might occur and may need a tremendous amount of work to rectify problems such as data inconsistency of the registration database.

1. Introduction

Sirindhorn International Institute of Technology (SIIT), Thammasat University, uses Microsoft Access version 2.0 as a database management program for its registration system. Due to the growth of the number of students, many registration works have been successfully implemented and are well served from the use of the computerized database. However flaws in the database have shown up from time to time caused by the experience of many users. The flaws signal that data protection measures should be improved to protect the database against a variety of possible threats (both deliberate and accidental). For example, the system might crash in the middle of some unfinished transactions, thereby leaving the central database in an unpredictable state. Consider the case when two processes executing concurrently 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 the data illegally. Therefore the system has to provide an extensive set of controls to protect the database against such threats specifically, *recovery, concurrency, security, and integrity* controls. For the stability of the database, using one central database for registration is perilous since some situations may adversely happen, such as a system crash. Since only one central database 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 be certain that every piece of information it contains can be reconstructed from some other information stored in other places-redundancy. The solution we provide is that we create another database which is used only during the registration period. We will refer to this new database as the *registration database* and the central database as the *main database* throughout this paper.

2. Problems

2.1 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's words [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 (computer failure) 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. 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 students are updating the record a power failure occurs. Hence, those unfinished transactions must be rolled back. 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.

2.2 Concurrency

Registration database is a shared resource. We must expect and plan for the likelihood that several users will attempt to access and manipulate data at the same time. With concurrent processing involving updates, 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. This technique is essential to our registration database, such as when more than one student are registering the same course with a limited number of students in class. When the student decides to register on the course, the database integrity will check first whether or not the class is full by having one variable used to store the number of students who have registered that course in the current semester. This situation can lead the database to the problem of lost updates. Assume that the maximum students in class to be 50 and there are already 49 students registered. Student (A) wants to register the course so he checks to make sure that the course is not yet full and 49 is the number that appears to him. But *before* he can update the number to be 50, another student (B) also wants to register for this course too. So B does the same operation as A; checks the number of students and, if possible, then updates it. At this moment, *before* A updates the record, 49 is also read by B which indicates that he can register for this course since it is not full yet. Hence the update by A is lost. An invalid result is obtained since after these two students register the course, the actual number of students is 51, not 50 as it appears in database. It is not only lost update problems 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.

2.3 Security

After a workable application for the registration database had been made, it is time to concentrate on how to manage who can use which features and establish application security. Security concerns ensuring that users can do only what they are allowed to do. In the SIIT central database, all information concerned with every student is kept in it. If no security system is implemented into the system, tremendous problems will arise since a student might alter his academic record by changing his grades, his colleagues grades, or putting in a course and grade in which he did not actually register. Hence a security system is needed for maintaining a usable database for student registration.

security. Some constraints must be enforced to ensure that authorized users are doing correct operations, satisfying all constraints. For example, the institute enforces the constraint that each student must register at least 9 credits and must not exceed 22 credits per semester, or that a student cannot register some courses as regrade if he has gotten a better than D+ grade for that course. Another example is that some courses have prerequisite courses in which students must pass their prerequisite before he can register the courses, etc.

3. Solutions to the Problems

By using a central database for student registration, many problems are likely to occur due to unaware or deliberate action. All the errors will be adversely directed to the central database. With the great importance that the central database is used for most institute administration work, a small database is implemented in order to be used as a substitute during the registration period. So any adverse affect will only be confined to the registration database. Consequently, advantages concerning security are also provided. Moreover, by separating the registration database out, the institute can modify or change the structure of the database easily, without too much concern about its side effect on the central database. The registration database can be generated easily from the central database by transferring only the necessary information for registration. Some information is transformed into more appropriate form such as grades. Grades will be transformed into the form of *regradable* and *canreregis*, as shown in Figure 1 and 2. Hence, as shown in Figure 3, there will no longer be actual grades of each student in the registration database, but only flags to indicate whether he can regrade or reregister or not. Any unauthorized users who try to change the grades will fail since no grade is kept in the registration database. It is also more convenient for our registration database since the transformed grade is more relevant to the objective of the registration's query than the actual grade. The initializing procedure of registration database is simply depicted in Figure 4. From Figure 4, the table "TREGISTER" and the table "TregisterW Regradable" are doing an unmatch query to find any student that might register later, after the registration period. The result is appended back to table "TregisterW Regradable" which is then exported to the registration database. All of the adbove processes are done by the registrar.The table "TregisterW Regradable" is used as an intermediate between the central and the registration database. In addition, the grade information in the table "TREGISTER" is transformed into *regradable* and *canreregis*, and appended to the table "TregisterW Regradable". All of the student academic information in the previous semester is stored in this table and will be used as reference only. During student registration, new records are added in table "TableX" and this table will be exported back to the central database after registration period. There is no need to transfer the table "TregisterW Regradable" back to the central database since no change was made to this table during registration period.

3.1 Solution for Recovery

Since we are using separate databases any failure that might cause the database to corrupt is now limited only to the registration database, leaving the central database untouched. For the case that registration database is collapsed, it can easily be reconstructed within three minutes, since it contains only information for registration, it is then ready to be used again to provide uninterrupted service during the registration 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 canceled. 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. 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 SIIT, the technique of commit and rollback transaction is implemented in Microsoft Visual Basic for Access by using the reserved words BeginTrans, CommitTrans, 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 students decide to register, the program adds the students to the institute record and updates the number of students in class 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. For example, total tuition fee that student A must pay is 40,000 Baht which he will pay in two installments of 20,000 Baht each. While he is in the middle of the payment process, at time t_1 the transaction is terminated for some reason just after 20,000 Baht is deducted from his balance. Assume that the transaction did not complete yet since it needs to update the total amount that he had paid, update that he pays by cash or check, and update the date that he had paid. As a result to the database, the transaction did not complete and must be redo. The amount that he still owes the institute now turns to be 20,000 Baht instead of 40,000 Baht. So he needs to process his payment again, and finds that he only had to pay 20,000 Baht once instead of a total of 40,000 Baht (the first 20,000 Baht installment was not paid to the cashier yet, only processed by the computer that he was *going to* pay 20,000 Baht). If an atomicity is used for the payment transaction, whenever a termination is forced on the transaction the whole transaction must be undo- instead of redo only the incomplete transaction. Therefore this kind of payment transaction should be made as atomicity in order to avoid the inconsistency of the database

3.2 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 registration database is a shared and classified resource, careful database management must be incorporated. Microsoft Access, provides three levels of locking. These are *record locking, table and recordset locking, and opening with exclusive access*. For record locking, only the record currently being edited is locked. For table and recordset 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 Access automatically locks the record currently being edited even though the programmer did not predefine the lock mechanism. In registration database, since we always operate under multi-user environment, the opening with exclusive access is irrelevant and will not be mentioned twice. The most used mechanism in our registration database is table and recordset locking. Since this database is a relational database and has a quite complicated relation and query, recordset will be used for most of the time. As mentioned before about concurrency, the inconsistency about counting the number of students in class can be solved by using table and recordset locking which will be referenced as *lock*. As the students decide to register any course, the lock is made active so that counting the number of students in each class 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 students for each class. In other words, from Figure 2, the central database uses the table "TREGISTER" to store and count the number of students in each course by using student ID and section ID as criteria for counting the number of students registered in each course. In our new system, the registration database stores records in table "TRegiterW Regradable" which is then used to count the number of students in each course. *DCount* is used in Visual Basic code. Hence, if these tables are locked while any student is currently updating then, the lost updates problem can be solved.

3.3 Solution for Security

In the past, the original system used table TUSERS to store login name and password for the registrar. As the registrar log on to the database, the application only verifies login name and password using ordinary Visual Basic code to check the information in TUSERS. *DlookUp* 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 registration 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 registration 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. 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 labeled 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 a registration database. It is easier and clearer to maintain a class of users than to concentrate on individual users, since every student must have the same right. Login name and password table will not be used, TUSER and new approach should be used here. Microsoft Access 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 file SYSTEM.MDA. This file is distinct from the database file which Microsoft Access uses to store information database security. The privileges of each level will be discussed in the next section.

3.4 Solution for Integrity

As mentioned before, integrity concerns protection against *authorized users*. For students 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 courses. Students must not be able to view database application as in the design view to avoid any adverse alteration by them. The other thing of concern is that students should not be allowed to use the toolbar since it provides features beyond the necessity of student registration. For the registrar class of users, it depends on the policy of the institute to what the registrar is allowed to do as does the faculty class of users. The other policies such as how many credits students must enroll in each semester, the maximum number of credits a student can register each semester, prerequisite, corequisite, etc, are deliberately ignored from user privileges levels since it varies from organization to organization.

4. Conclusion

So far, all fundamental problems have been cleared up. Security and stability of the registration database have been solved. However, a lot of delicate improvements can be implemented because the further use of this registration database 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 users needs.

5. Reference

[1]Date, C. J. (1995), An Introduction to Database Systems, Addison-Wesley Publishing Company, Inc.

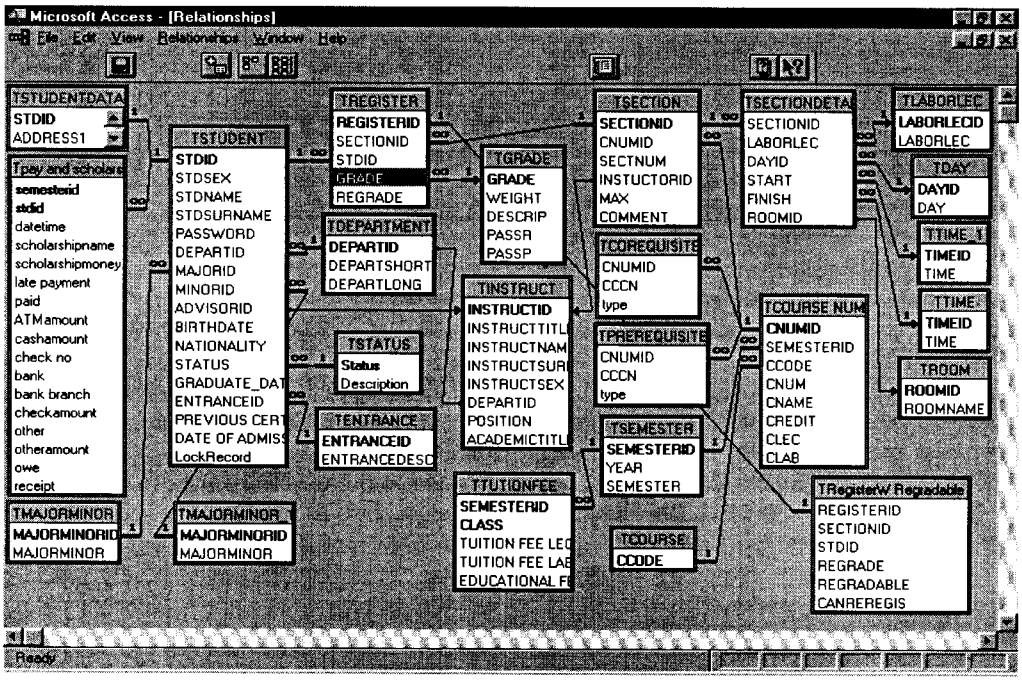

Figure 1. Relationship of the Central Database

GRADE	Regradable		Can Re-register	
	Basic Course	Engineering Course	Basic Course	Engineering Course
A	No	No	No	No
B+	No	No	No	No
B	No	No	No	No
C+	No	No	No	No
C	No	No	No	No
D+	No	Yes	No	No
D	No	Yes	No	No
F	No	No	Yes	Yes
W	No	No	Yes	Yes
I	-	-	-	-
ACC	No	No	No	No
EXE	No	No	No	No
S	No	No	No	No
U	No	No	Yes	Yes

Figure 2. Grades Transformation Table

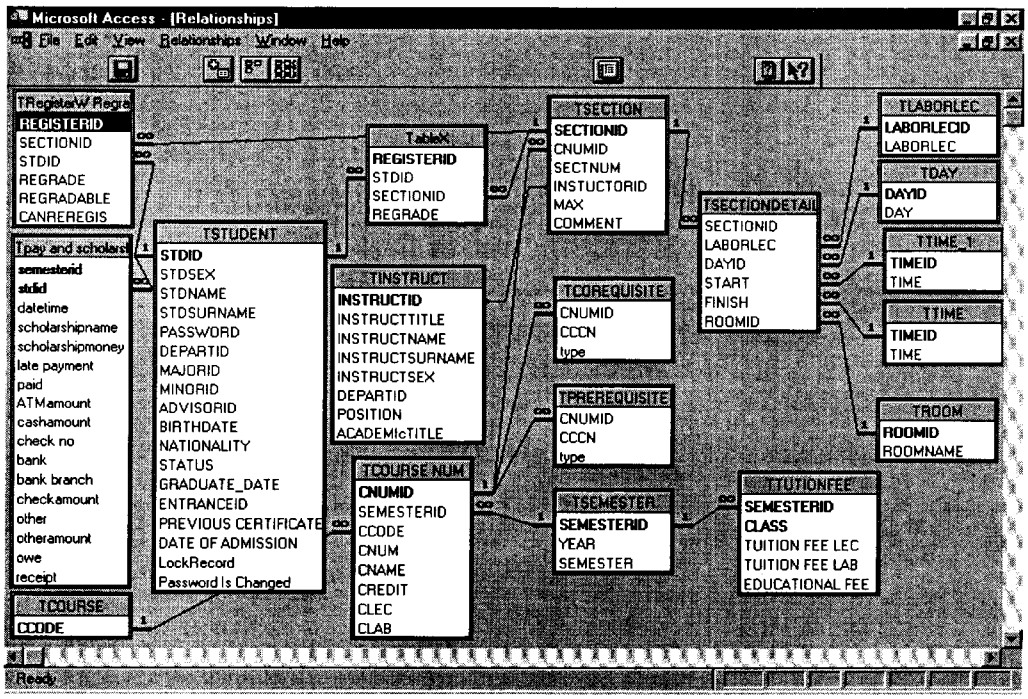

Figure 3. Relationship of the Registration Database

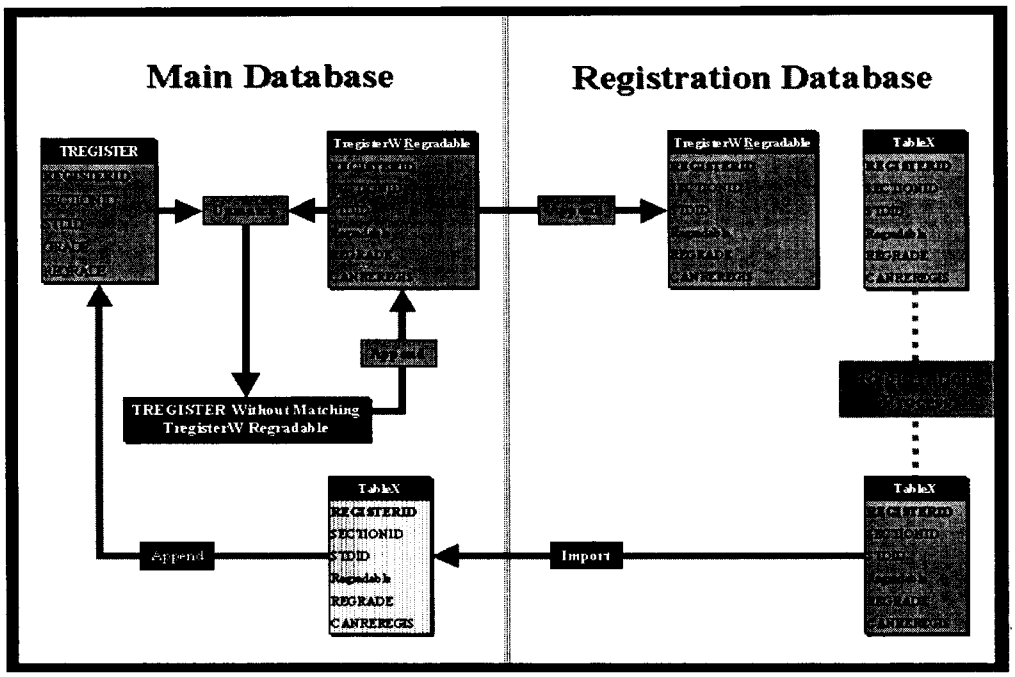

Figure 4. Pictograph of the Initialization of Registration database