Security and Data Integrity Analysis

By Fred Zhang and Mercury Ding

1. Privacy Analysis

In our system, different data can only be accessed by certain users. As for the course and section information, the time, location, instructors should be accessible to everyone. As for students' information, his or her name, major, class and the courses enrolled are public, but his or her plans and potential schedules should only be available to friends. However, everyone can see the aggregate data, indicating the number of students who intend to take a certain course or section. Those aggregate data can help professors schedule proper number of sections and inform students of what courses are popular. As for professors' information, his or her name, department and course taught are public. Students can view what courses a professor has taught before to get an idea of their specialization.

There are also many privacy controls in our system. First of all, password is private. Each student's plans and published potential schedules are only available to friends. For each plan, a student might see many search results, but not necessary want to publish all of them. A student can save all of those search results in their account but only publish a few of them. Each schedule would have a boolean attribute -- PublicOrPrivate. Private schedule can only be viewed and modified by the student alone, while the public schedule is accessible by friends. For professors, all information about courses and ratings should be available to every student as a reference and help them make decision. The messages between students and professors should be kept private. Anybody else should not have access to private conversations records.

2. Security Analysis

First of all our data are protected against sql injection and XML attacks. All communications between front end and database is through stored procedure, so there is no way for user to directly directly enter an sql. All HTML according to database content using helper methods of asp.net. Those methods handle scripts injection.

Data integrity implies that only the administrator and those who have permissions from him can make certain modifications. To prevent inconsistency and integrity of our database, students and professors cannot change their name, userID, department. Those changes must be confirmed by the administrator.

Both students and professors can only change data directly related to them. They can modify their password at any time. Students can add and delete plans, add, delete, publish and hide their schedules. Each plan or schedule belongs to one student and neither other students nor professors should be able to change it. Students have to login to view and make changes to its own data. However, they don’t have to login to get access to the course and section information. As for the course data, only the professors are allowed to modify and update them. For a specific course, only its instruction has the right to adjust the capacity and enroll students in the wait list.

To protect the system from malicious attack, our system only allows section deletion one at a time. The professor can only remove one section at a time. To avoid confusion, sections are not supposed to change a lot. Only the administrator can add or remove a course, because change to curricular needs to be discussed and confirmed by the committee.

3. Entity Integrity Analysis

1. For the People table, UserID must be 9 digit unique char that is the primary key. FName and LName must be a var 30 character nvarchar and not null. Type is the type of student or professor. Password is a var 20 char.
2. For the Professor table, PUserID is a unique 9 digit char that is the primary key. DepartID must be a var 5 character nvarchar that cannot be null. Office must be a var 6 character nvarchar that cannot be null.
3. For the Student table, SUserID is a unique 9 digit char that is the primary key. Major must be a var 12 character nvarchar and not null. Year is an integer.
4. For the Term table, TermID is an integer that is the primary key. Start_date and End_date are date that cannot be null.
5. For the Plan table, PID is an integer that is the primary key. SUserID is a unique 9 digit char. TermID is an integer.
6. For the Course table, CourseID is a smallint that is the primary key. CourseDP must be a var 5character nvarchar and not null. CourseNum is a smallint. Descript is a text with ‘’ as default. Credits must be an integer and not null.
7. For the Contain table, PID is an integer. CourseID is an integer. Both of them are the primary key.
8. For the Schedule table, ScheID is an integer that is the primary key. PID is an integer. Probability and Priority are integer. PublicOrPrivate is a 6-7 char.
9. For the Section table, SectID is an integer that is the primary key. TermID is an integer that is not null . CourseID is a smallint and is not null. SectNum is a tinyint and is not null. PUserID is a unique 9 digit char. Capacity is an integer. EnrollNum is a tinyint.

10. For the Has table, ScheID is an integer. SectID is an integer. Both of them are the primary key.
11. For the Time table, SectID is an integer. Classroom is a var 7 char. Period is an integer from. Classroom, TermID and Period compose the primary key.
12. For the Enroll table, SUserID is a unique 9 digit char. SectID is an integer. Both of them are the primary key. Rating is an tiny integer 0-5. Status is a var 8 char. Time is a datetime type.
13. For the Prerequisite table, Prerequisite is an integer. Requisite is an integer.
14. For the Friend table, Requester is a unique 9 digit char and is the primary key. Accepter is a unique 9 digit char and is the primary key. Both of them are the primary key. Status is a var 10 char.
15. For the Message table, MessID is a var 5 char that is the primary key. Sender is a unique 9 digit char and is the primary key. Receiver is a unique 9 digit char and is the primary key. All of these three are the primary key. Content is a 30 var char. Time is a date.
16. For the WaitList table, SectID is a not null integer, SUserID is a 9 digit char, T is a datetime and Rating is a tinyint.
17. For the FriendRequest table, Requester and Accepter is a 9 digit char.
18. For the Department table, DepartID is a 5 digit char and is the primary key. DepartName is a 50 digit char.

4. Referential Integrity Analysis

1. UserID in People table is referenced by Student and Professor. This design ensures that student and professor are a subset of people. UserID is also referenced by Friend, FriendResquest, Message tables, since those functionalities apply to any users. All of those foreign keys are all cascade on delete and update, since all people in our system are either student or professors.
2. SUserID in Student table is referenced by SPlan, Enroll and WaitList tables. Those two foreign keys both cascade on delete and update, since once a student drops out of school, we no longer have to keep track of his or her information.
3. PUserID in Professor table is referenced by Section table. It should cascade on update but reject on delete. If a professor has taught any classes, removing him from the system can cause loss of information for those sections he taught.

4. TermID in Term table is referenced by SPlan and Section tables, since they are only meaningful under the context of a certain term. Other entities like Time, schedule and enrollment also needs term information. However, we can get the termID for them through sectionID. TermID probably should not be changed at all. Our database would reject any changes made to TermID.
5. CourseID in Course table is referenced by Contain(a plan contains a course), Section and Prerequisite tables. Once a course is removed from the curricula, its previously taught section should still exists. If this course was taught in the past, the system should reject the deletion but remove its sections that have not begin yet. Plan containing removed coursed will be recognized as incomplete. The student should receive some notifications. A trigger would be necessary to handle those situations.
6. SectID in Section table is referenced by Has (a schedule has a section), Time, Enroll and WaitList tables. All of those foreign keep cascade on update and delete.
7. PID in Plan table is referenced by Schedule and Contain (a plan contains a course). Both foreign key should cascade on update and delete.
8. ScheID in Schedule table is referenced in has (a schedule has a section) table. The foreign key should cascade on both update and delete.
9. DepartID in Department table is referenced by course and professor. It should reject its changes, since Department is not something we can change arbitrarily.

5. Business Rule Integrity Analysis

There are a lot of constraints posed in our system.

1. A schedule should only contain sections in the same term. This constraint should be checked whenever a schedule is updated.
2. All users in our system are either students or professors. We have a type attribute in People, which is constrained to be either ‘S’ or ‘P’.
3. When a student is drafting a potential schedule (consists of a list of sections), it should match one of its plans (consists of a list of courses), where each section in the schedule corresponds to a course in the plan. For most of the time, a student should not have the time conflicts in the schedule. The database should remind the user of conflicts, but it will be allowed if the user insists.
4. We need to ensure that there are no room conflicts for different sections. No two sections should meet at the same time in the same room.

5. As for the enrollment process, whenever a student enrolls or drop a section, a trigger should update the numEnrolls in section table accordingly. Students and professors should be granted with proper rights. If this section is under the capacity, a student should be able to enroll right away, but if this section is already full, he can only be put into the wait list.
6. The professor can decide whether to schedule a new section, set higher capacity, accept or reject students in the wait list. With proper rights assigned to students and professors, registration process can be made easier.