Synopsis of the project
CSci 4707 Fall 2018
Team 18

Overview

There are eighteen testing requirements overall, and we write at least one test case for each requirement. There are two main situations in order to figure out if we create our database correctly, like below:

1. We try to INSERT some legal data, and it should succeed based on our constraints. If it fails, then there are some problems in our original design, so that we do not need to write any more queries to test it.
2. We also try to INSERT some illegal data, and it may fail because of our constraints. There are two situations:
    a. It fails. In that case it means the constraint we set when creating tables works well. On the other hand, if it does not fail, it can mean we did not set enough constraints when creating tables. This way our design is bad.
    b. And another situation, however, is that it can be inserted. Afterwards, we will write some tests queries, and will get some outputs after applying corresponding queries. For example, getting the count of some rows which meet or do not meet the requirements. For different situations, we will check the returned value manually and check our design quality.

Here are the specific tests and explanation, including the reasons why our design is bad or good:

1. Does your design capture the requirement that all patients under the age of 18 must have a parent or guardian in the system?

Our design does not check if a patient actually has a guardian when he/she is under 18. So there is a flaw. Additionally, we have five testing queries to catch those illegal situations, such as checking if a guardian has a child who is under 18.

2. Does the system document when a person does not have insurance? This could be a new situation after having insurance.

We try to get the number of insurance that a patient has. If we get a count greater than 0, it means the patient has insurance. Otherwise a patient does not have any insurance, which is what our design allows. But our design does not often check if a patient has valid insurance, so there is a flaw. We should fix it by writing some triggers to keep checking if a patient has more valid insurance recently.

3. Does your design capture the system's need to store old insurance information along with current insurance information? A clerk should be able to determine what dates the old insurance was active and what is currently active.

We try to insert multiple insurance information for the same patient. However, it fails as our original design made PID unique, so there is a flaw.

4. Can a patient be seen by more than one doctor or physician assistants for the same visit? Does the first doctor become the primary provider?

We try to get the number of service providers that saw a patient. If the output is more than 1, then patient visited more than 1 doctor or physician assistants. However, our design completely missed primary service provider, so there is a flaw.

5. Can a patient have more than one diagnosis per visit? Assume all patients must be given a Diagnosis.

Our design does not have any information about when the visit happens, so we cannot allow a patient has more than one diagnosis. There is a flaw.

We should add a new attribute like "Date" to indicate a specific visit, to solve this problem.

6. Does the system properly document followup tests/procedures with proper coding?

No. Our design only allows a patient to have one followup procedure. So there is a flaw. We try to insert multiple followup tests for one patient, and it fails.

7. Does the system document what clerk collected the patient's personal information?

Our design successfully get all the information about patients and their medical records. So the system does documents what clerk collected the patient's personal information.

8. Does the system allow more than one initial assessment to be in the system at a time?

We try to get the number of initial assessments that a patient has. If it is more than 1, it means the system allows more than one initial assessment at a time. However, our design does not have information about time or date when the assessment is created, so we do not allow more than one initial assessment. So we cannot see if two happen at the same time.

In order to fix it, we can add a Date attribute in the table Initial_Assessment.

9. Do assessments have the nurse's information associated with it? Can we retrieve the nurses information?

Yes, we can retrieve the nurses information from the assessments, through joining the two tables Initial_Assessment and Nurse. In Initial_Assessment, we have Nurse_ID as an attribute, so we can find the nurse information according to that, as Nurse_ID is the same thing as EmployeeID in Nurse table**.**

10. Are we able to review the vitals as defined on the writeup of the patient through queries for a given visit?

No. Here our system only allows to record information for one visit. So we cannot review the vitals as defined on the writeup of the patient through queries for a given visit.

In order to improve our system, we decide to add an attribute called "Date_time" to table Initial_Assessment. What is more, we want to make the combination of assessment id and Date_time as primary key of assessment. After that, additionally, we try to get information about the patient, initial assessment and medical record, when given a date time. That should work at that time.

11. Does each employee only work in one and only one department?

Yes. Our design meets that requirement. We try to get the number of departments that an

employee works in. If the output Num_of_department is more than 1, the employee is working on more than one department. If each one gets 1 as results, then each employee only works on one department, which means our design is reasonable.

12. Does every department have at least one employee assigned to it? Is there a way to ensure this?

(1). It is like the opposite way of the last test case. Here we try to get the number of employees that a department has, by grouping by department id. If the outputs are all larger than 1, then the system works well.

(2). There is a way to ensure this, which we think should be making Department own a foreign key referencing Employee, instead of making Employee own a foreign key to Department. So that employee id is the primary key which shouldn't be null, then there'll be at least on employee.

13. Are a patient's medication listed for each visit?

No. Our system only allow one treats relationship for each patient since PID is unique. So we cannot get information for each visit, but only the first visit. So when we try to insert a second visit information for the same patient, it will fail.

In order to improve that, we want to add a new attribute called "date" to table Treat, so that we can record and get multiple times of information for different visits.

14. Does a patient have to have a referral for a given visit?

We try to get PID of patients who does not have a referral. If it return records, it means a patient doesn't have to have a referral. Otherwise, it means they have to have a referral. However, again, we don't have information about date(time) of the referral, so we cannot determine a "given visit". So the quality of our design is not good.

We want to add an attribute called "Date" to table Follow_up_order. So that we can retrieve some information when given a visit.

15. Is there an accounting mechanism for the medical items used. We should be able to determine what doctor annotated the billable supplies and items. The nurse is responsible for the accounting to the doctor.

In our design, we have no access to what doctor annotated the billable supplies and items. The nurse is responsible for the accounting to the doctor.

The approach to solve it is like below:

Add an attribute Medical_Record_ID in Doctor table, as a foreign key referencing Medical_record, so that we can access the information from doctor. And also, in the Medical_record table, we can add two attributes, which are Used_Item_ID and Quality, and they can record information for the billable supplies and items.

16. Does each patient have at least one doctor or PA assigned to them? The system has a mechanism to record the physical visit with the doctor/PA?

INSERT INTO Consult (Employee_ID, Service_provider_ID, PID) VALUES (2, 23456, 2);
INSERT INTO Consult (Employee_ID, Service_provider_ID, PID) VALUES (2, 23456, 3);

The last insert will fail. Our system couldn't add a duplicate service provider id in the consult table so the patient with pid 3 didn't get recorded.

It is obvious that our design has a flaw that we made Service_provider_ID unique. So that we cannot have one service provider match multiple patients, which is not practical in real life. Additionally, as you can see in the template, we write a query to see if there is a case where a patient does not have any doctor or PA. We can get a number of 3 as a patient who does not have any doctor or PA.

17. Can the hospital access the medical record of patients?

Based on our tables and design, we can successfully get all the medical records according to patient ID(pid). Because the PID is the primary key in Patient table, and it is also the foreign key referencing Medical_record table, we can join the two tables on PID, so that we can access the medical records of all the patients. In this case, our design makes sense.

18. When a patient came to the hospital and finished his diagnose, can we find which doctor or physician assistant updated which medical record?

Based on our tables and design, we can successfully get the Service_provider_ID and the Record_ID corresponding to a specific patient, which means we can find it. The query above is an example to check the availability, and here we choose the case with PID=2 and PID=1.

The Doctor table has a composite foreign key, which is Employee_ID and Service_provider_ID, referencing Service_Provider table, the Diagnose table has a foreign key called Record_ID referencing Medical_record, so we can find corresponding Service_provider_ID and the Record_ID when given a PID. Overall our design makes sense.

The Physician_Assisstant table has a composite foreign key, which is Employee_ID and Service_provider_ID, referencing Service_Provider table, the Diagnose table has a foreign key called Record_ID referencing Medical_record, so we can find corresponding Service_provider_ID and the Record_ID when given a PID.

Overall our design makes sense.