Project Details

Project Name: Student Exams database

Team Members: Hon Ching Li (honching), Haohua Feng(haohufe)

Abstract

During the exam period, there are many students who attend and take the exams. It is very difficult to organize all the data. With the data model created by our team, it is going to be easy to maintain the information in a good way. Each relation contains unique attributes and keys that can be used as a primary key or foreign key to reference or get reference to another relation(s). It is going to be helpful and friendly to instructors to modify students' exam information and access them.

It is better to use a database instead of an excel file. Even though all datas can fit in an excel file, it is much more convenient to access and modify data with the function provided by the database management system.

1. Introduction

We design our database in several tasks, identify user cases, our target users, BCNF relations, clear illustration ER diagram, dataset generated by our scripts and query execution analysis. These tasks ensure database quality and provide better access to our student exams database which contains all exam records for all students. Also we create to setup privilege to limit access of students, and assign accesses for instructor.

2. Tasks

2.1 Use case

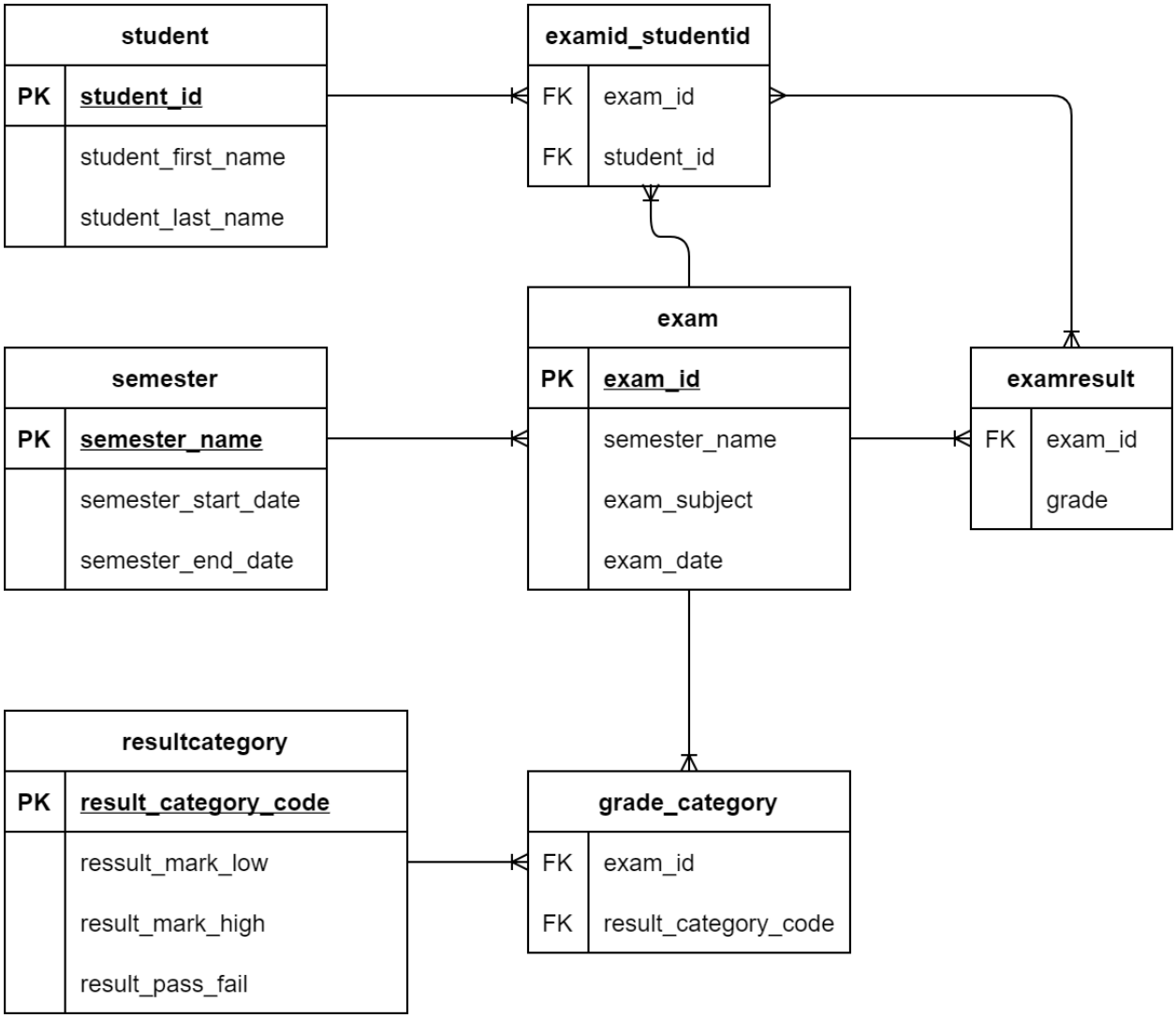
Instructors are able to quickly filter out the range of students based on their grade. After exam week and the grade is ready, the instructors and TAs can modify the database to insert and update students' grades. And also, they can easily access the grade to calculate students' GPA. While student has limit access on our database.

2.2 Target User

The target users will be instructors, teaching assistants, and students.

The instructors and teaching assistants will administer the database.

2.3 ER diagram



2.4 Dataset

Our dataset is not from the real world. Instead, we wrote the script “generate.py” to generate the large dataset. The large dataset that we generate is tested and it's a suitable form to load into our databases.

2.5 BCNF

In our project2, all relations in the table are in BCNF forms, which means all functional dependencies’s LHS is SuperKey. We did this by using the BCNF decomposition algorithm.
There are a total of 7 relations.

Semester relation
FD set: semester_name -> semester_start_date, semester_end_date
Semester_name here is the PK for this Semester relation.

Exam relation:
FD set: exam_id -> semester_name, exam_subject, exam_date
Exam_id is the PK for Exam relation.

Student relation:
FD set: student_id -> student_first_name, student_last_name
Student_id is the Primary Key for Student relation

examID_studentID relation
FD set: exam_id -> studentID
Exam_id and student_id are both Foreign Key for examID_student_ID relation

ExamResult relation

FD set: exam_id -> grade
Exam_id is the Foreign Key

ResultCategory relation
FD set: result_category_code ->result_mark_low, result_mark_high, result_pass_fail
Result_category_code is the primary key.

Grade_category relation
FD set: exam_id -> result_category_code
Both exam_id and result_category_code are Foreign key

3. Query Execution Analysis (Bonus)

Demonstrates some queries to access our database

we want to find all exam id and the student id where the score is between 85 to 100

Case 1: Runtime analysis = O((n+r)*n) ~ O(n^2) where n is the total number of exams taken by students, and r is the total number of grade code (1,2,3,4,...). For each exam_id in grade_category, find the corresponding exam_id in examid_studentid, this takes O(n) time. Then also search the result_category_code of this exam_id if this is fall in the grade of 85-100. This takes O(r). Overall, the runtime is O((n+r)n).

```
Select ES.exam_id, ES.student_id
From Grade_category as G, ExamID_StudentID as ES
Where G.exam_id = ES.exam_id
AND G.result_category_code = (
        Select R.result_category_code From ResultCategory as R
        Where R.result_mark_low = 85 And R.result_mark_high = 100
) ;
```

Case 2: Runtime analysis = O(n) where n is the total number of exams taken by students. For each tuple in grade, find the corresponding student_id in examid_student_id (O(n)). Then compare the grade to check if it is between 85 to 100(O(1)). Overall, the runtime will be O(n).

```
Select ES.exam_id, ES.student_id
From examresult as E, ExamID_StudentID as ES
Where E.exam_id = ES.exam_id AND E.grade >= 85 AND  E.grade <= 100;
```

4. SQL files

1. To create our relations, copy all the codes beside Drop table commands from create.sql file, if want to reset all relations, simply copy all the codes from the create.sql into query tool
2. Copy all the codes from load.sql to insert all the tuples into our relations. The instruction is also within the readme.txt file as well.
3. Our data isn't from real world dataset, we generate all the tuples using the script that we wrote, which is generate.py.

4. We also setup privilege for users:
   For example, an English instructor wants to make an update to a student's grade based on the exam result. This instructor cannot view students' grades that are not in this course.
   To grant access for instructors, copy the code from privilege.sql into query tool
   First create a role English instructor.
   Then create a view that only can view the English class students' grade.
   Grant all privileges to eng_instructor on this view. So that, this instructor can make modifications on this view and lead to the modification of the original table.

   This will be similar to a user as a student, by creating a view that only contains the information of this student. So this student can only view his/her exam time, exam result, and grade.

References

"Citation Style and Reference Formats." Accessed April 29, 2021. https://www.acm.org/publications/authors/reference-formatting.

"Exams." Accessed April 29, 2021. http://www.databaseanswers.org/data_models/student_exams/index.htm.