Design Document CS301

Project Phase A

Team:
Het Fadia - 2019CSB1084
Shikhar Soni - 2019CSB1119

The project required us to build certain functionalities of an academic portal. Individual components of the project are as follows with their brief description.

Tables:
The tables used in the project are:

- ➔ This table course_catalog contains the LTPSC structure of all the courses that are taught at the institute
  - ◆ course_catalog(courseid varchar(7) primary key, L real, T real, P real, S real, C real);
- ➔ The below table contains all the prerequisite corresponding to each courses
  - ◆ pre_requisite(courseid varchar(7), pre_req varchar(7));
- ➔ The below table contains the time table slots corresponding to each course
  - ◆ time_table(courseid varchar(7), slot integer);
- ➔ The below table course_batches contains the batches to which the course is available
  - ◆ course_batches(courseid varchar(7),secid integer,sem integer,year integer,batch varchar(10));
- ➔ This table stores the courses offered corresponding to the teacher, section id , semester and year
  - ◆ course_offerings(courseid varchar(7), teacherid integer, secid integer,sem integer, year integer, cg real, primary key(courseid, secid));
- ➔ There is a table student_info which stores the studentid, name and department name of the students
  - ◆ student_info(studentid varchar(12) primary key, _name varchar(50), dept_name varchar(20));
- ➔ There is a table batch advisor which stores the information about batch advisors
  - ◆ ba_info(batchadvisorid varchar(8), teacherid integer, primary key(batchadvisorid));
- ➔ There is a table instructor info which stores the information about the instructors

- instructor_info(teacherid integer, _name varchar(50) not null, dept_name varchar(20) not null, courseid varchar(7), secid integer, primary key(courseid, secid));

➔ Corresponding to each course_offering table there will be course grade table and course enrollment table
- NEW.courseid || NEW.secid(studentid varchar(12));
- NEW.courseid || NEW.secid || '_g'(studentid varchar(12) primary key, grade integer);

➔ For each student there are 3 tables namely transcript table, enrollment table and history table shown below
- studentid_t(courseid varchar(7) primary key, credits real , sem integer , year integer , grade integer);
- studentid_e(courseid varchar(7), sem integer, year integer, secid integer, primary key(courseid, sem, year)
- studentid_h(courseid varchar(7) primary key, sem integer, secid integer, year integer, status varchar(50));

➔ Corresponding to every instructor there is history table
- instructor || '_h' (courseid varchar(7), sem integer, year integer, status varchar(50), secid integer, entry_no varchar(10), primary key(courseid, entry_no));

➔ Corresponding to every batch advisor there is history table
- batch_advisor || '_h'(courseid varchar(7),sem integer,year integer,status varchar(50), ins_status varchar(50), secid integer, entry_no varchar(10) primary key(courseid,secid,entry_no));

➔ Corresponding to the dean there is request/history table
- 'dean' || '_h' (courseid varchar(7), sem integer, year integer, status varchar(50), ins_status varchar(50), ba_status varchar(50), secid integer, entry_no varchar(10) primary key(courseid,secid,entry_no));

➔ We have student information table
- create table student_info(studentid varchar(12) primary key,_name varchar(50), dept_name varchar(20));

➔ We have instructor information table
- create table instructor_info(teacherid integer, _name varchar(50) not null, dept_name varchar(20) not null, courseid varchar(7), secid integer, primary key(courseid, secid));

➔ We have batchadvisor information table
- create table ba_info(batchadvisorid varchar(8), teacherid integer, primary key(batchadvisorid));

Triggers:

The Triggers implemented in the Project are listed below

- ➔ check_enrollment
  - ◆ This trigger checks for 1.25 rule, clashes of time table, 7 cgpa criteria, course offerings before a student tries to enroll in a course.
  - ◆ After insert we use another trigger to insert into course_e table as it would be a successful enrollment.
- ➔ Insert_course_offering
  - ◆ Creates two tables corresponding to each course_offering table namely course grade table and course enrollment table
- ➔ insert _students
  - ◆ create student tables such as student_t, student_e, student_h on inserting new students
- ➔ Insert_instructor
  - ◆ creates tables such as instructor_h for inserting new instructors
- ➔ Insert_ba
  - ◆ creates tables such as ba_h for inserting new batch advisors
- ➔ Generate_ticket
  - ◆ before insert into student_h in case of student_h check everything and ensure 1.25 rule is exceeded
  - ◆ After insert into student_h insert into instructor_h or from instructor_h to batchadvisor_h or so on.

Permissions/Privileges:

The permissions granted to individual stakeholders are listed below.

Common privileges given to all users:

Select on course_e i.e., all enrollments of a course as is the case in AIMS
select on course_catalog
select on course_offerings
select on pre_requisite, batch_table, time_table
select on instructor_info, student_info, batchadvisor_info

Students:
select on student_t i.e., student transcript table.
select, insert on student_e i.e., student enrollment table.
select, insert on student_h i.e., student ticket table, contains status of various tickets raised by the students.

Instructors:
select on course_e i.e., course enrollment table for each course
select, insert, update on course_g i.e., course grade tables, exist for each course
select, update on instructor_h i.e., all the tickets and their status for an instructor

Batch Advisors:
select, update on batchadvisor_h i.e., all the tickets and their status for a batch advisor

Dean:
select, update, insert on student_t
select, update on dean_h i.e., all the tickets and their status for the dean
select, update, insert on student_e, course_e
select, update on student_h
select on course_g
select, insert, update, delete on time_table, batch_table, pre_requisite table, student_info, teacher_info, batchadvisor_info, course_catalog, course_offerings
grant create table and delete table

Functions and stored procedures:

enroll process:

- ➔ enroll() --> called by student to insert into student_e to enroll trigger_function everything including 1.25 rule checked. After insert a trigger inserts into course_e.

ticket generation process begins:

- ➔ make_ticket(); -> insert into student_h to generate ticket generate_ticket trigger everything including exceeding 1.25 rule checked. After insert a trigger inserts the ticket into teacherid_h
- ➔ approve_teacher('Y' or 'N') update teacherid_h and after update insert into batchadvisor_h through an after update trigger
- ➔ approve_ba('Y' or 'N') update batchadvisor_h and after update insert into dean_h through an after update trigger

- ➔ approve dean('Y' or 'N') insert into course_e, student_e if yes and update student_h and dean_h

Other functions used:

- ➔ semsg
  - ◆ Takes the semester and year parameter and calculates the sg of the student corresponding to semester
- ➔ yearsem
  - ◆ Returns the year and the semester in which the student is studying
- ➔ semcredits
  - ◆ Takes semester and year arguments as input and returns the total credits earned in that semester.
- ➔ Lasttwosemcredits
  - ◆ It returns the average of the total credits earned in the last two semesters
- ➔ check_enrollment
  - ◆ This procedure checks for 1.25 rule, clashes of time table, 7 cgpa criteria, course offerings before a student tries to enroll in a course.
- ➔ create_student
  - ◆ It takes studentid as input and creates the user student. It also creates the transcript table of the student, enrollment table of the student and history table of the student.
- ➔ enrollment_clashes
  - ◆ Checks if the time table of the input course clashes with the time table of the courses currently enrolled by the student.
- ➔ create_course_sec_table
  - ◆ we will create course grade as well as course enrollment table (anyone can view the enrollment table as in aims)
- ➔ Load_grade
  - ◆ Reads the csv file and loads the grades into the course_g table.
  - ◆ Allows the user to give name of csv file as input
- ➔ Load_grade_to_transcripts
  - ◆ Checks if all grades are uploaded, if not raises notice
  - ◆ Uploads them to student transcripts i.e., student_t, used by dean
- ➔ Generate_transcripts
  - ◆ Prints the content of student_t to the screen
- ➔ Upload_time_table
  - ◆ Reads csv and upload time table
- ➔ calculate_CG
  - ◆ Calculates CG using student_t