Cameron Ellis, Sneha Reddy, Micah Remus

Group 16

**School Database User Manual**

Database Functions:

**insert Function:**

* **Description:** This function is a general insert function used to insert a new tuple into any of the relations in the DB (Student, Course, School, etc.)
* **Generalized Query Code:******
* **Input:** In this function the user will put in the name of the table “TABLE\_NAME”, attributes they would like to insert values for in a new tuple (Attr\_1, Attr\_2, Attr\_3, ... , Attr\_n), and the values they want to put in for the attributes of the entry they wish to insert
* **Steps:**
  + The user selects which relation they want to enter a new tuple into.
  + The function will check to make sure that the primary key of the new tuple does not match that of any other existing tuple within the relation.
  + If the new tuple passes the primary key check then it is inserted into the desired relation.

**delete Function:**

* **Description:** This function is a general delete function used to delete a tuple from any of the relations in the DB (Student, Course, School, etc).
* **Input:** The primary key of the tuple in the relation the user is wanting to delete from.
* **Steps:**
  + The user selects which relation they want to delete a tuple from.
  + The user will input the primary key of the tuple they wish to delete.
  + The function will check to see if this primary key exists in the relation the user is wanting to delete from.
  + If the primary key exists then the tuple is deleted, and any tuples with this primary key in other relations with dependencies are deleted as well.

**update Function:**

* **Description:** This function is a general update function used to update a tuple in any of the relations in the DB (Student, Course, School, etc.)
* **Input:** The primary key of the tuple that the user wants to update, the attributes in the tuple they want to update, as well as what they want to update them to.
* **Steps:**
  + The user selects which relation they want to update a tuple in.
  + The user will then input the primary key of the tuple they wish to update.
  + The function will check to make sure that the primary key of the tuple matches that of a tuple within the relation.
  + If the primary key check is passed, then the user can select which attributes they wish to update and input the updated information for the attributes in that tuple.
  + The tuple will then be updated within the relation once submitted.

**getTuple Function:**

* **Description:** This function is to get all the information of a single tuple from any of the entity sets (Student, Faculty, School, etc.).
* **Input:** Relation to get from, and the tuple’s primary key.
* **Steps:**
  + The user is asked to input the relation they wish to get information from, and the primary key of the tuple they are looking for.
  + The user inputs the primary key, and the function checks to see if the relation contains said primary key.
  + If the primary key check is passed then the function will find and return the tuple from that relation and all its attributes.

**getSchoolCourses Function:**

* **Description:** This function will get all the courses offered at a particular school.
* **Input:** School name.
* **Steps:**
  + The user will input the school name to find courses from.
  + The function will then access the “Courses” relation and return all the unique courses offered at the selected school.

**getMaxMinCredits Function:**

* **Description:** This function will get the minimum or maximum credit hours out of all the courses offered at school and return the max or min credit hours along with the name of the course with the max or min credit hours.
* **Input:** Whether max or min, and the name of the school.
* **Steps:**
  + The user will input the school name to find courses from. The user will also input whether they want the max or min.
  + The function will then use the aggregate function to group by the school and find the maximum or minimum credit hours for a course from the “Courses” relation and give the name of the course.

**countStudentsSchool Function:**

* **Description:** This function will get the count of the number of students who are enrolled at a single school.
* **Input:** The name of the school to find the student count for.
* **Steps:**
  + The user will input the school name to find count students from.
  + The function will then access the “Attends” relation.
  + It will then use the aggregate function to group by and count the total number of students per school
  + The function will then return the count of the total number of students who attend that particular school.

**countStudentsCourse Function:**

* **Description:** This function will get the count of the number of students in a particular course at a school.
* **Input:** School name and course name to count the number of students from.
* **Steps:**
  + The user will input the school and course name to count.
  + The function will then access the “Takes” relation and group by the course and school
  + It will then use an aggregate count function to return how many students are in the particular course.

**avgSchoolGPA Function:**

* **Description:** This function will get the average GPA of all the students at a particular school.
* **Input:** School name to get the average from.
* **Steps:**
  + The user will input the school name to get average from.
  + The function will then access and get the StudentID’s for the student’s at a particular school.
  + Then the function will take those StudentID’s and cross reference with the “Academic\_Record” relation.
  + Using the cross reference of StudentID’s from the School and Academic Records, the function will use an aggregate function to get the average gpa of the students from the school.

**maxMinSchoolGPA Function:**

* **Description:** This function will get the max or min GPA from a particular school.
* **Input:** School name to get max or min from, and whether the user wants the max or min.
* **Steps:**
  + The user will input the school name to get max/min from as well as if they want the max or min GPA.
  + The function will then access and get the StudentID’s for the student’s at a particular school.
  + Then the function will take those StudentID’s and cross reference with the “Academic\_Record” relation.
  + Using the cross reference of StudentID’s from the School and Academic Records, the function will use an aggregate function to get the max/min gpa from the school along with the StudentID.

**getStudentCourses Function:**

* **Description:** This function will return all the courses that a student is taking.
* **Input:** The StudentID of the student to get courses for.
* **Steps:**
  + The user will input the StudentID for the desired student.
  + The function will then access the “Takes” relationship, and retrieve all the courses that the student is taking by using the StudentID entered.

**getStudentClubs Function:**

* **Description:** This function will return all the clubs that a student is a part of.
* **Input:** The StudentID of the student to get clubs for.
* **Steps:**
  + The user will input the StudentID for the desired student.
  + The function will then access the “Joins” relationship, and retrieve all the club names that the student is a part of using the entered StudentID.
  + The function will then return the club names for all the clubs retrieved in the prior step.

**getStudentSchool Function:**

* **Description:** This function will return the school that a particular student of interest attends.
* **Input:** The StudentID of the student to get school for.
* **Steps:**
  + The user will input the StudentID for the desired student.
  + The function will then access the “Attends” relationship, and retrieve the name and address of the school they attend.
  + The function will then give the school name and address of the student from the prior step.

**calcGradCreds Function:**

* **Description:** This function will calculate and return the number of credits a student has left before they graduate.
* **Input:** StudentID and number of credits required for graduation.
* **Steps:**
  + The user will input the StudentID for the desired student, and the number of credits needed to graduate at the given school.
  + The function will then access the “Academic\_Records” relation, get the particular student from the ID, and retrieve their total number of credits.
  + The function will then subtract the number of credits the student has from the credits required to graduate and then return the result.

**getFacultyDept Function:**

* **Description:** This function will return the department that a particular faculty member is a part of.
* **Input:** The FacultyID of the faculty member to get a department for.
* **Steps:**
  + The user will input the FacultyID for the desired faculty member.
  + The function will then access the “Department” relation, and retrieve the particular department for the FacultyID and return it.

**getSchoolFaculty Function:**

* **Description:** This function will get and return all the faculty members that work at a particular school.
* **Input:** The School name to get faculty list for.
* **Steps:**
  + The user will input the school name to get a list for.
  + The function will then access the “Works” relationship, and retrieve all the names and IDs of the faculty members for that particular school.
  + The function will then display all the faculty members names and FacultyID’s for the members.

**getFacultyCourses Function:**

* **Description:** This function will return all the courses that a faculty member is teaching.
* **Input:** The FacultyID of the faculty member to get courses for.
* **Steps:**
  + The user will input the FacultyID for the desired member.
  + The function will then access the “Teaches” relationship, and retrieve all the courses that the faculty member is teaching.
  + The function will then display the unique course names it retrieved from the above step.

**cumulativeGPA Function:**

* **Description:** This function is used to get the cumulative gpa of a student by averaging their past semester GPAs.
* **Input:** The StudentID the user wishes to get the cumulative GPA for.
* **Steps:**
  + The user will input the StudentID for the desired student.
  + The function will then use the aggregate function on the “Academic\_Record” relation to get the average of the semester GPAs of that student to represent the cumulative GPA.
  + The function will then return the calculated cumulative GPA from the prior step.

**clubMemberCount Function:**

* **Description:** This function is used to get the member count of a given club at a school.
* **Input:** The club name and the school name for that club.
* **Steps:**
  + The user will input the club name and school name of the club they want the member count of.
  + The function will then use the aggregate function on the “Joins” relation to group by the school and club name entered and use the count function to count the number of members.
  + The function will then return the counted members in the club.