**Final Project:  
Easy\_Intern Database**

Department of Computer Science  
Georgia State University  
  
CSc4710 Database System  
Due Date: April/29/2021  
Instructor: Berkay Aydin  
Student: Chengpeng Wu

**Introduction**

The Easy\_Intern database I designed is about a unique environment where all kinds of internships can be easily found without browsing several websites. In Easy\_Intern, I collected all the best internship opportunities from different companies across the country, which are suitable for students of all majors. The motivation for me to build this database is that as students, it is always difficult to find the best internship opportunities, and students would also spend a lot of time looking for perfect internships in many places too. Such waste of time and energy! Therefore, in order to make students’ life easier to find the best internship opportunities, I decided to create Easy\_Intern to minimize the time and energy to find the best internship opportunities and perhaps some frustrations that come with looking for the best internship. With it, students can spend extra time on school assignments / projects. Some important features will be included, for example: company information, specific internship location, salary and requirements and so on. It will also include a brief description of the company's internship, as well as the preferred major for the company's specific internship.

Requirements Analysis

Part A: Data Requirements

* Easy\_Intern stores many different companies (for example: Google, Facebook, Microsoft, Wells Fargo Bank, Office Depot, etc). And for each company, it has multiple locations, names, and unique company\_id.
* Each company has multiple and different departments. Departments are identified by department name(for example: Information Technology, Human Resource, Research and Development, Business Development, etc), unique department id, and the total number of employees.
* Each company will post job\_posting. Job\_posting includes a unique post\_id, description, application\_link, and post\_title.
* Job\_posting has several job\_locations. Each job\_location is associated with its unique location\_id, and address (zip\_code, street, state, and city)
* An applicant, in this case which is student, can apply through the job\_posting of each company.
* A student can have an applied\_job that stores the student\_id, company\_id, and the post\_id.
* A student can also have an apply\_later job that stores the post\_id.
* A company can accept or reject a student for an interview after reviewing their applications.
* A student has a student\_id, name(Fname, Lname), GPA, college, major, and address.
* Each company employs employees and indicates the employment date. Employees are identified by an employee\_id, name(Fname, Lname), age, and employee\_id.
* Employees are categorized as an intern, manager, IT specialists, business developers.
* Each manager manages only one department.
* An Employee works on projects. The project is identified by its unique name, unique number, and location.
* Information Technology specialists have software skills.
* Business Developers have financial skills and payroll systems.

Part B: Functional Requirements

* Requirement 1: Students should be able to create an account with information about themselves.
* Requirement 2: Students should be taken to an external link when a student wants to apply to an internship.
* Requirement 3: Students will be able to mark what internships they have applied for.
* Requirement 4: Easy\_Intern should generate the number of applications that a particular company or all companies in the database have received.
* Requirement 5: Students should be able to add/remove an internship to their wishlist, favorite or apply\_later files.
* Requirement 6: Students should be able to access the number of interns that are currently interning for a particular company.
* Requirement 7: Should be able to access the type of projects that interns tend to work on in a particular company.
* Requirement 8: Should return all the internships specific to Computer Science
* Requirement 9: Admin can see the number of applications for each internship in descending order.
* Requirement 10: Admin can see the number of students in each major

Part C: Conceptual Design

Below are some diagrams for each entity:

1.Company

Diagram

Description automatically generated

2. Department

Diagram

Description automatically generated

3. Student

Diagram

Description automatically generated

4. job\_posting

Diagram

Description automatically generated

5. job\_location

Diagram

Description automatically generated

6. employee

Diagram

Description automatically generated

6. Intern

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Relational Design

Part A: Entities

1. **Company**

Description: It describes the company’s information

Attributes:

* company\_id: it is the identification and also the primary key for this entity.
* name: it is the name of the company.
* location: it is a multivalued attribute describing the company's location(s).

Relationships:

* \*accepts: Related to STUDENT, it indicates how many students a company accepts.
* \*has: Related to DEPARTMENT, it shows how many departments a company has.
* \*employs: Related to EMPLOYEE, it shows the employment of employees of a company.
* \*posts: Related to JOB\_POSTING, it indicates the jobs that a company post.

Primary key:

* It is identified by the company\_id attribute because it shows the uniqueness of a company.

1. **Department**

Description:It describes the departments in a company

Attributes:

* dept\_name: it is the name of a department.
* dept\_id: it is the identification of a department and also the primary key for the entity.
* Number of employees: the number of employees in a department.

Relationships:

* \*is managed by: Related to MANAGER, it determines the managers of departments.
* \*has: Related to COMPANY, it shows how many departments that company has.

Primary key:

* It is identified by dept\_id because each department has a unique id.

1. **Student**

Description: It describes the students that apply to an internship in a company.

Attributes:

* Student\_id: it is the identification of a student and also the primary key of the entity.
* Name: is it the name of the student and it is also a composite attribute with first and last name.
* Gpa: it is the grade point average of the student.
* College: it is the college that the student attends.
* Major: it is the field of study of the student.
* Address: it is the address of a student.

Relationships:

* \*applies: Related to JOB\_POSTING, it shows how many students apply for the job of an internship.
* \*accepted: Related to COMPANY, it reports the number of student acceptance of a company.
* \*has: Related to APPLIED\_JOB, it reports the jobs that a student has applied for.

Primary key:

* It is identified by student\_id because each student has a unique identification.

1. **Job\_posting**

Description: It describes the job/internship that a company has posted.

Attributes:

* Post\_id: it is the identification of the post and also the primary key of the entity.
* Description: it describes the job that has been posted.
* post \_title: it is the title of the post.
* application \_link: it is the link from which application form can be found.

Relationships:

* \*has applied: Related to STUDENT, it indicates how many students have applied for a job/internship.
* \*has: Related to JOB\_LOCATION, it indicates how many locations that a job has.
* Primary key:
* It is identified by post\_id because it separates the jobs posted by each company.

1. **Job\_location**

Description: It describes the location of each job posted.

Attributes:

* Location\_id: it is the identification of a job location and also the primary key of the entity.
* Address: it is the address of the location and it is also a composite attribute with the city, state, zip code and street.

Relationships:

* \*has: Related to JOB\_POSTING, it shows how many jobs a location has.

Primary key:

* It is identified location\_id because it separates the locations of jobs.

1. **Employee**

Description: It describes the employees who work for a company.

Attributes:

* Employee\_id: it is the identification of an employee.
* Name: it is the name of an employee and it is also a composite attribute with first and last name.
* Salary: it is the amount of money that an employee makes.

Relationships:

* \*employed: Related to COMPANY, it shows the number of employees employed in a company.
* \*works on: Related to PROJECT, it indicates the projects that employees work on.

Primary key:

* It is identified by employee\_ssn because each employee has a unique social security number.

1. **Intern**

Description: It describes interns in a company

Attributes:

* Employee\_id: it is the identification of an intern.
* Name: it is the name of an intern and it is also a composite attribute with first and last name.
* Salary: it is the amount of money that an intern makes (if paid intern).

Relationships:

* \*is-a: Related to EMPLOYEE, it is the subclass of a superclass employee.
* \*employed: Related to COMPANY, it shows the number of interns employed in a company.
* \*works on: Related to PROJECT, it indicates the projects that interns work on.

Primary key:

* It is identified by employee\_ssn, because each employee has a unique social security number.

1. **Manager**

Description: It describes the managers of a company.

Attributes:

* Employee\_id: it is the identification of a manager
* Name: it is the name of a manager and it is also a composite attribute with first and last name.
* Salary: it is the amount of money that a manager makes.

Relationships:

* \*is-a: Related to EMPLOYEE, it is the subclass of a superclass employee.
* \*manages: Related to DEPARTMENT, it shows how many managers manage a department.
* \*employed: Related to COMPANY, it shows the number of managers employed in a company.
* \*works on: Related to PROJECT, it indicates the projects that managers work on.

Primary key:

* It is identified by employee\_ssn, because each employee has a unique social security number.

1. **Business Developers**

Description: It describes the employees who deal with the business side of a company.

Attributes:

* Employee\_id: it is the identification of a business developer.
* Name: it is the name of a business developer and it is also a composite attribute with first and last name.
* Salary: it is the amount of money that a business developer makes.
* Financial skills: it is a skill that a business developer has.
* Payroll systems: business developer distributes(takes charge) of the payroll.

Relationships:

* \*is-a: Related to EMPLOYEE, it is the subclass of a superclass employee.
* \*employed: Related to COMPANY, it shows the number of business developers employed in a company.
* \*works on: Related to PROJECT, it indicates the projects that business developers work on.

Primary key:

* It is identified by employee\_ssn, because each employee has a unique social security number.

1. **IT Specialist**

Description: It describes the employees who deal with information technology in a company.

Attributes:

* Employee\_id: it is the identification of an IT specialist
* Name: it is the name of an IT specialist and it is also a composite attribute with first and last name.
* Salary: it is the amount of money that an IT specialist makes.
* Software skills: it is a skill that an IT specialist has.

Relationships:

* \*is-a: Related to EMPLOYEE, it is the subclass of a superclass employee.
* \*employed: Related to COMPANY, it shows the number of IT specialists employed in a company.
* \*works on: Related to PROJECT, it indicates the projects that IT specialists work on.

Primary key:

* It is identified by employee\_ssn because each employee has a unique social security number.

1. **Project**

Description: It describes the projects that a company handles.

Attributes:

* Project\_number: it is the number of a project and also the primary key of the entity.
* Name: it is the name of the project.
* Location: it is the location of the project.

Relationships:

* \*worked on by: Related to EMPLOYEE, it shows the projects and the employees who worked on them.

Primary key:

* It is identified as PNo because it indicates the uniqueness of a project.

1. **Applied\_job**

Description: It describes the job that a student has applied for and it is also a weak entity.

Attributes:

* Student\_id: it is the identification of a student.
* Company\_id: it is the identification of a company.
* Post\_id: it is the identification of the posted job.

Relationships:

* \*has: Related to STUDENT, it shows the job has been applied by a student.

Primary key:

* It is identified by student\_id and company\_id because they are both partial keys, which together form a primary key.

1. **Applied\_later**

Description: It describes the job that a student has marked down to apply later and it is also a weak entity.

Attributes:

* Post\_id: it is the identification of the posted job.

Relationships:

* \*has: Related to STUDENT, it shows the job that will be applied later by a student.

Primary key:

* It is identified by post\_id and it is also a partial key because it is a weak entity with only one key.

Part B: Relationships

1. **Company - Department**

A picture containing text, clock, watch, clipart

Description automatically generated

Relation: Shows the relation between company and its department

Attributes:

* Dept\_id (Foreign key from company): it is the unique identification of a department.
* Company\_id (Foreign key from department): it is the unique identification of company.

Cardinalities: the reason for the cardinalities is that I assume that at least one or at most many company(s) has at least one or at most many departments in order to be called a company.

1. **Company - job\_posting**

A picture containing text, clipart

Description automatically generated

Relation: Shows the relation between a company and its job posting

Attributes:

* Post\_id (Foreign key from job\_posting): it is the unique identification of a job posting.
* Company\_id (Foreign key from Company): it is the unique identification of company.

Cardinalities: the reason for the cardinalities is that a company may post multiple jobs, but a specific job will be posted by one company only.

1. **Company - Student**

Diagram

Description automatically generated

Relation: Shows the relation between a company and a student that can accept an offer

Attributes:

* Student\_id (Foreign key from Student): it is the unique identification of a student.
* Company\_id (Foreign key from Company): it is the unique identification of company.

Cardinalities: the reason for the cardinalities is that a company might hire one or multiple interns, but a student can accept one company only.

1. **job\_posting - job\_location**

A picture containing text, clock

Description automatically generated

Relation: Shows the relation between a job posting and its location

Attributes:

* post\_id (Foreign key from job\_posting): it is the unique identification of job posted by a company.
* Location\_id (Foreign key from job\_location): it is the unique identification of a job location.

Cardinalities: the reason for the cardinalities is a specific job posting can be in one location only, but a location may have multiple job openings.

1. **Student - job\_posting**

A picture containing text, clock, clipart

Description automatically generated

Relation: Shows the relation between a student and a job posted by a company

Attributes:

* Student\_Id(Foreign key from Student): it is the unique identification of a student.
* post\_id(Foreign key from job\_posting): it is the unique identification of a job posting

Cardinalities: the reason for the cardinalities is a student can apply to zero or more companies. Similarly, a job posts applied to by zero or more students.

1. **Student - applied\_job**

A picture containing text, clock, clipart

Description automatically generated

Relation: Shows the relation between a student and jobs the student has applied to.

Attributes:

* Student\_id (Foreign key from Student): it is the unique identification of a student.
* Post\_id (Foreign key from job\_posting): it is the unique identification of a job posting.
* company\_id (Foreign key from company): it is the unique identification of company.

Cardinalities: the reason for the cardinalities is that a student can apply to zero or more companies and similarly, a job can be applied by zero or more students.

1. **Student - apply\_later**

A picture containing text, clock, clipart

Description automatically generated

Relation: Shows the relation between student and job posts that the student plans to apply in future

Attributes:

* Student\_id (Foreign key from Student): it is the unique identification of a student.
* Post\_id (Foreign key from job\_posting): it is the unique identification of a job posting.

Cardinalities: the reason for the cardinalities is a student can decide to add zero or more job posts to apply\_later.

1. **Company-Employee**

Diagram

Description automatically generated

Relation: Shows the relation between company and the employees that a company employs.

Attributes:

* Company\_id (Foreign key from company): it is the unique identification of a company.
* Employee\_id (Foreign key from employee): it is the unique identification of an employee.

Cardinalities: the reason for the cardinalities is that a company can employ one or many employees but exactly one employee can work for one company.

1. **Employee-Project**

Diagram

Description automatically generated

Relation: Shows the relation between employees and the projects they work on.

Attributes:

* Project\_number (Foreign key from company): it is the unique number of a project.
* Employee\_id (Foreign key from employee): it is the unique identification of an employee.

Cardinalities: the reason for the cardinalities is that one or more employees can work on one or many projects.

1. **Employee-(IT\_specialist, Business\_developer, manager,intern)**

Diagram

Description automatically generated

Relation: Shows the relation between employee (superclass) and it’s subclasses (IT\_specialist, Business\_developer, manager, intern)

Attributes:

* Employee\_id (Foreign key from employee): it is the unique identification of an employee.

Cardinalities: this is a specialization/generalization case, it is partial and overlapping because there might be more other categories of employees and each employee might be involved in two or more categories.

1. **Manager-Department**

A picture containing text, clock

Description automatically generated

Relation: Shows the relation between a department and the manager who manages it.

Attributes:

* Manager\_id (Foreign key from employee): it is the unique identification of a manager.
* Department\_id(Foreign key from department): it is the unique identification of a department.

Cardinalities: the reason for the cardinalities is that exactly one manager manages exactly one department.

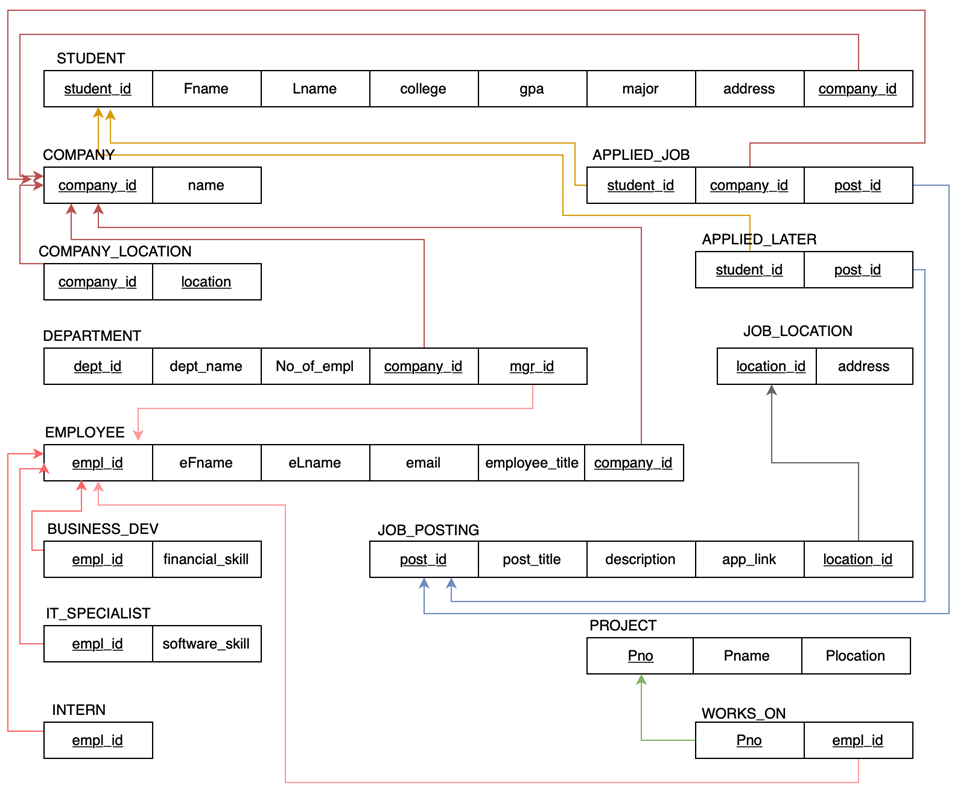
Part C: EER Diagram

Diagram

Description automatically generated

Relational Model

Part A: Relational Diagram



Part B: Process of Mapping

I started with a step one which was to map all the regular entities,STUDENT, DEPARTMENT, COMPANY, JOB\_LOCATION, JOB\_POSTING AND PROJECT.

In step two, I mapped the weak entities, APPLIED\_JOBS AND APPLIED\_LATER.

In step three, I mapped the binary 1:1 relation types; with the cardinalities of MANAGER(1,1) and DEPARTMENT(1,1) being total participation, I merged the two entities into one with all their attributes. In step four, I mapped 1:N relation types; with the cardinalities of DEPARTMENT(1,1) and COMPANY(1,N), the primary key in company becomes the foreign key in department. The same 1:N concept applies to relationships STUDENT-COMPANY, COMPANY-EMPLOYEE, JOB\_POSTING-JOB\_LOCATION, STUDENT-APPLIED\_LATER and STUDENT-APPLY\_LATER. In step five, I mapped M:N relation types; with EMPLOYEE(1,N) and PROJECT(1,N), I created a new entity as WORKS\_ON with the primary keys of both employee and project entities. The same concept goes to STUDENT-JOB\_POSTING. In step six, I mapped the multivalued attributes; in our case I have location as a multivalued attribute in the company entity, therefore, I created a new entity as COMPANY\_LOC with the primary key of company as a foreign key in the entity company\_loc and the attribute(location) itself as a partial key. In step seven, I mapped the specialization/generalization; with EMPLOYEE being a super class and BUSINESS\_DEV, IT\_SPECIALIST, INTERN and MANAGER being subclasses with overlapping and partial participation, I created a relation for all the entities with their foreign keys being the primary key of the superclass(employee).

Data Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table** | **Attribute** | **Data Type** | **Primary Key** | **Foreign Key** | **Constraints** |
| STUDENT\_PERSONAL\_INFO | Student\_id | INT | YES | N/A | 7 digits, positive |
| STUDENT\_PERSONAL\_INFO | Fname | VARCHAR(20) | N/A | N/A | N/A |
| STUDENT\_PERSONAL\_INFO | Lname | VARCHAR(20) | N/A | N/A | N/A |
| STUDENT\_PERSONAL\_INFO | College | VARCHAR(40) | N/A | N/A | N/A |
| STUDENT\_PERSONAL\_INFO | Gpa | FLOAT | N/A | N/A | N/A |
| STUDENT\_PERSONAL\_INFO | Major | VARCHAR(20) | N/A | N/A | N/A |
| STUDENT\_LOCATION\_INFO | Student\_id | INT | N/A | STUDENT\_PERSONAL\_INFO(Student\_id) | 7 digits, positive |
| STUDENT\_LOCATION\_INFO | location\_id | VARCHAR(25) | N/A | N/A | N/A |
| COMPANY | Company\_id | INT | YES | N/A | 4 digit, positive |
| COMPANY | Name | VARCHAR(20) | N/A | N/A | N/A |
| COMPANY\_LOCATION | Company\_id | INT | N/A | COMPANY(Company\_id) | 4 digit, positive |
| COMPANY\_LOCATION | Location | VARCHAR(25) | N/A | N/A | N/A |
| STUDENT\_COMPANY | Student\_id | INT | N/A | STUDENT\_PERSONAL\_INFO(Student\_id) | 7 digits, positive |
| STUDENT\_COMPANY | Company\_id | INT | N/A | COMPANY(Company\_id) | 4 digit, positive |
| DEPARTMENT\_DETAIL\_INFO | Dept\_id | INT | YES | N/A | 5 digit, positive |
| DEPARTMENT\_DETAIL\_INFO | Dept\_name | VARCHAR(25) | N/A | N/A | N/A |
| DEPARTMENT\_DETAIL\_INFO | No\_of\_empl | INT | N/A | N/A | positive |
| DEPARTMENT\_EMP | Dept\_id | INT | N/A | DEPARTMENT\_DETAIL\_INFO(dept\_id) | 5 digits, positive |
| DEPARTMENT\_EMP | Mgr\_id | INT | N/A | EMPLOYEE\_INFO(empl\_id) | 7 digits, positive |
| DEPARTMENT\_COMPANY\_INFO | Dept\_id | INT | N/A | DEPARTMENT\_DETAIL\_INFO(dept\_id) | 5 digits, positive |
| DEPARTMENT\_COMPANY\_INFO | Company\_id | INT | N/A | COMPANY(Company\_id) | 4 digit, positive |
| EMPLOYEE\_INFO | Empl\_id | INT | YES | EMPLOYEE\_INFO(empl\_id) | 7 digit, positive |
| EMPLOYEE\_INFO | eFname | VARCHAR(20) | N/A | N/A | N/A |
| EMPLOYEE\_INFO | Lname | VARCHAR(20) | N/A | N/A | N/A |
| EMPLOYEE\_INFO | Email | VARCHAR(45) | N/A | N/A | N/A |
| EMPLOYEE\_INFO | Empl\_title | VARCHAR(20) | N/A | N/A | N/A |
| EMPLOYEE\_COMPANY | Empl\_id | INT | N/A | EMPLOYEE\_INFO(empl\_id) | 7 digit, positive |
| EMPLOYEE\_COMPANY | Company\_id | INT | N/A | COMPANY(Company\_id) | 4 digit, positive |
| INTERN | Empl\_id | INT | N/A | EMPLOYEE\_INFO(empl\_id) | 7 digit, positive |
| BUSINESS\_DEV | Empl\_id | INT | N/A | EMPLOYEE\_INFO(empl\_id) | 7 digit, positive |
| BUSINESS\_DEV | Financial\_skill | VARCHAR(25) | N/A | N/A | N/A |
| IT SPECIALIST | Empl\_id | INT | N/A | EMPLOYEE\_INFO(empl\_id) | 7 digit, positive |
| IT SPECIALIST | Software\_skill | VARCHAR(25) | N/A | N/A | N/A |
| PROJECT | Pno | INT | YES | N/A | 3 digits, positive |
| PROJECT | Pname | VARCHAR(20) | N/A | N/A | N/A |
| PROJECT | Plocation | VARCHAR(25) | N/A | N/A | N/A |
| JOB\_POSTING\_INFO | Post\_id | INT | YES | N/A | 2 digits, positive |
| JOB\_POSTING\_INFO | Post\_title | VARCHAR(40) | N/A | N/A | N/A |
| JOB\_POSTING\_INFO | Description | VARCHAR(40) | N/A | N/A | N/A |
| JOB\_POSTING\_INFO | App\_link | VARCHAR(40) | N/A | N/A | N/A |
| JOB\_POSTING\_INFO | Location\_id | INT | N/A | JOB\_LOCATION(Location\_id) | N/A |
| JOB\_POSTING\_LOCATION | Post\_id | INT | N/A | JOB\_POSTING\_INFO(post\_id) | 2 digits, positive |
| JOB\_POSTING \_LOCATION | Location\_id | INT | YES | N/A | 4 digits, positive |
| JOB\_LOCATION | Location\_id | INT | YES | N/A | 4 digits, positive |
| JOB\_LOCATION | address | VARCHAR(50) | N/A | N/A | N/A |
| APPLIED\_JOB | student\_id | INT | N/A | STUDENT\_PERSONAL\_INFO(student\_id) | 7 digits, positive |
| APPLIED\_JOB | post\_id | INT | N/A | JOB\_POSTING\_INFO(post\_id) | 2 digits, positive |
| APPLIED\_LATER | student\_id | INT | N/A | STUDENT\_PERSONAL\_INFO(student\_id) | 7 digits, positive |
| APPLIED\_LATER | post\_id | INT | N/A | JOB\_POSTING\_INFO(post\_id) | N/A |
| WORKS\_ON | Pno | INT | YES | PROJECT(Pno) | 3 digits, positive |
| WORKS\_ON | empl\_id | INT | N/A | EMPLOYEE\_INFO(empl\_id) | 7 digits, positive |

**Implementation**

* **Requirement 1: Students should be able to create an account with information about themselves.**

This was implemented by inserting tuples into the student account and then I retrieved a student’s info. PS: only accessible to companies (whoever is in control of reviewing application)

|  |
| --- |
| # Create table statements for table “student\_personal\_info”  create table student\_personal\_info(  student\_id int primary key,  Fname varchar(20),  Lname varchar(20),  college varchar(40),  gpa float,  major varchar(20),  CONSTRAINT CHECK (length(student\_id) = 7 AND student\_id > 1)  );  #Query that represent this function:  select \*  from student\_personal\_info; |

* **Requirement 2: Students should be taken to an external link when a student wants to apply to an internship.**

I implemented this by retrieving a job’s external link.

|  |
| --- |
| # Create tuple statements for “app\_link”  create table job\_posting\_info(  post\_id int primary key,  post\_title varchar(30),  description varchar(45),  app\_link varchar(250),  CONSTRAINT CHECK (length(post\_id) = 2 AND post\_id > 1 )  );  #Query that represent this function:  select app\_link  from job\_posting\_info; |

* **Requirement 3: Students will be able to mark what internships they have applied for.**

This was implemented retrieving jobs/internships that a student has applied for.

|  |
| --- |
| # Create table statements for table “applied\_job”  create table applied\_job(  student\_id int,  foreign key(student\_id) references student\_personal\_info(student\_id),  post\_id int,  foreign key(post\_id) references job\_posting\_info(post\_id),  CONSTRAINT CHECK (length(student\_id) = 7 AND student\_id > 1 AND length(post\_id) = 2 AND post\_id > 1)  );  #Query that represent this function:  select post\_id  from applied\_job; |

* **Requirement 4: Easy\_Intern should generate the number of applications that a particular company or all companies in the database have received.**

I implemented this by retrieving the number of applicants(students) that all the companies have received to determine the most applied company.

|  |
| --- |
| # Create table statements for table “applied\_job”  create table applied\_job(  student\_id int,  foreign key(student\_id) references student\_personal\_info(student\_id),  post\_id int,  foreign key(post\_id) references job\_posting\_info(post\_id),  CONSTRAINT CHECK (length(student\_id) = 7 AND student\_id > 1 AND length(post\_id) = 2 AND post\_id > 1)  );  #Query that represent this function:  select COUNT(student\_id)  from applied\_job; |

* **Requirement 5: Students should be able to add/remove an internship to their wishlist, favorite or apply\_later files.**

I implemented this by retrieving the job(s) that a student wants to apply later.

|  |
| --- |
| # Create table statements for table “apply\_later”  create table apply\_later(  student\_id int,  foreign key(student\_id) references student\_personal\_info(student\_id),  post\_id int,  foreign key(post\_id) references job\_posting\_info(post\_id),  CONSTRAINT CHECK (length(student\_id) = 7 AND student\_id > 1 AND length(post\_id) = 2 AND post\_id > 1)  );  #Query that represent this function:  select post\_id  from apply\_later; |

* **Requirement 6: Students should be able to access the number of interns that are currently interning for a particular company to determine their chances of being called for an interview.**

I implemented this by retrieving the number of interns that are interning in a particular company.

|  |
| --- |
| # Create table statements for table “intern”  create table intern(  empl\_id int, foreign key(empl\_id) references employee\_info(empl\_id),  CONSTRAINT CHECK (length(empl\_id) = 7 AND empl\_id > 1 )  );  #Query that represent this function:  select COUNT(empl\_id)  from intern; |

* **Requirement 7: Should be able to access the type of projects that interns tend to work on in a particular company to determine the type of projects that they might work on if hired.**

This was implemented by retrieving projects that interns work on.

|  |
| --- |
| # Create table statements for table “project”  create table project(  Pno int primary key,  Pname varchar(30),  Plocation varchar(30),  CONSTRAINT CHECK (length(Pno) = 3 AND Pno > 1 )  );  #Query that represent this function:  select Pname  from project; |

* **Requirement 8: Should return all the internships specific to Computer Science.**

I implemented this by retrieving all the internships/jobs that are specifically for computer science major.

|  |
| --- |
| # Create table statements for table “iob\_posting\_info”  create table job\_posting\_info(  post\_id int primary key,  post\_title varchar(30),  description varchar(45),  app\_link varchar(250),  CONSTRAINT CHECK (length(post\_id) = 2 AND post\_id > 1 )  );  #Query that represent this function:  select post\_id  from job\_posting\_info  where post\_title = 'Computer Science'; |

* **Requirement 9: Admin can see the number of applications for each internship along with the job post title in descending order.**

I implemented this by retrieving the number of applications in descending order and joining the applied\_job table and the job\_posting table.

|  |
| --- |
| # Create table statements for table “iob\_posting\_info”  create table job\_posting\_info(  post\_id int primary key,  post\_title varchar(30),  description varchar(45),  app\_link varchar(250),  CONSTRAINT CHECK (length(post\_id) = 2 AND post\_id > 1 )  );  #Query that represent this function:  select job\_posting\_info.post\_title, COUNT(applied\_job.post\_id)  from applied\_job  JOIN job\_posting\_info ON job\_posting\_info.post\_id = applied\_job.post\_id  group by applied\_job.post\_id  order by COUNT(applied\_job.post\_id) DESC; |

* **Requirement 10: Admin can see the number of students in each major to determine the most dominant major in the database.**

This was implemented by retrieving the total number of students in each major.

|  |
| --- |
| # Create table statements for table “iob\_posting\_info”  create table student\_personal\_info(  student\_id int primary key,  Fname varchar(20),  Lname varchar(20),  college varchar(40),  gpa float,  major varchar(20),  CONSTRAINT CHECK (length(student\_id) = 7 AND student\_id > 1)  );  #Query that represent this function:  select COUNT(student\_id)  from student\_personal\_info  group by major; |

Summary

All in all, Easy\_ Intern is an ideal place for students to find the best internship. Unlike other places looking for internships, Easy\_ Intern gives students the opportunity to explore the company of their choice, because it gives students the opportunity to see the types of workers working for a company, including interns, the number of interns and their work projects. These types of information may help students prepare for internships at the company, if hired. It also provides students with email of current interns, so they can ask for help on how to apply for an internship at a company and how to prepare for an interview to improve their chances of getting the internship they need.