CSCE 608
Database Systems

Course Project #1

"Career-Fair" Report

Contents:

Submitted by:

Mayank Sharma
426007370
(mayank.sharma@tamu.edu)

1. Project Description

Career Fair Database project aims to create a centralized generic repository for any job fair hosted by universities and external organizations. It covers most of the critical aspects of organizing a Career Fair. We expect various Companies to register for a Career Fair by paying a nominal fee which is reflected as Sponsorship for the event. Depending on the amount paid by the Company, we categorize each one of them into Silver, Gold or Platinum sponsors. The Booths allocated to a Company in terms of location and size also depends on the Sponsorship amount. After successful one time registration, the company is redirected for creating job postings for the available positions. The company can log back in any time it wants to manage its profile. For the current application, security is not considered as one of the prime aspects. The Manage profile section of a company list out all the information about the company like Sponsorship category, amount, Booth assigned, its location and size, along with all of its job postings. The Company can add, modify and delete its job postings on this page. To create or edit a job posting, the company has to fill a quick online form specifying Position, Job type (as in Full Time, Part Time or Internship), and other student specific information like which majors can apply and what are the degree levels of student being considered and whether or not International Students can apply? After successful submission, it redirects back to the Manage Job profile section of the Company.

The other prime aspect of the Career Fair is Students. The students can register for the Fair to see what all Companies are coming for the Fair and what are the job positions they are hiring for? Student Registration is also a onetime thing but it is not mandatory. After registration, the Students are provided with job recommendation depending on their department, degree level and other information. Although there is an option to apply for a job, the user should understand, this project is not meant to be an online job portal. Thus even after applying to a job, the student needs to attend the Career Fair to meet the recruiters and get to know more about the position. Applying online just intimates the company about that students interest in a particular position and for statistical purposes.

Even if a Student does not want to register he/she can have a look on the generic Job Search tool, which filters out Jobs from all the Companies based on the Department, Degree Level and International candidature of the user.

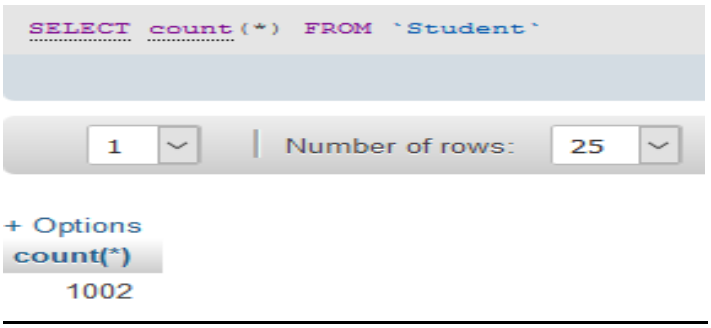
2. Data Collection

Random Data for the web application was generated with the help of an online mock data generator tool (link: http://mockaroo.com). Data was downloaded in the csv format. Using this data, an sql script was created, which was then imported into the Database using the "phpMyAdmin" Database query management tool. This process was repeated for all the tables in the Database.

Bulk Insert sql file for Students table:

```
INSERT into Student (name, department, degree_level, isInternational) VALUES ("Mead Sandiso",9,5,0), ("Whittaker Alexandro",6,1,1), ("Kerr Moxha",8,3,0), ("Florance Belfiel",9,3,1), ("Teddie Crumpto",4,4,0), ("Bili Benois",8,5,0), ("Jemimah Zanicchell",5,2,0), ("Heidie Murrthu",4,4,0), ("Prudy Winn",8,6,1), ("Ragnar Melma",5,5,1), ("Ashla Senchenk",3,3,1), ("Bailie Bernarde",5,3,0), ("Kristian Sprin",1,1,0), ("Scottie O'Feene",4,6,0), ("Newton Dimitro",4,4,0), ("Ritchie Deene",3,3,0), ("Maegan Wallbridg",9,5,0), ("Ashil Blizar",4,2,1), ("Gianina Guyneme",2,6,0), ("Kippie Lindhol",4,4,0), ("Silvanus Adamide",1,4,0), ("Tessie Ortig",7,6,0), ("Ximenez Fishle",6,6,0), ("Ophelie Polloc",5,6,1), ("Hartley Sloya",5,2,1), ("Benetta Smitha",1,1,0), ("Mirabelle O'Her",4,5,1), ("Ralina Stickel",5,5,1), ("Kris Fishpon",6,6,0), ("Andres Dawidsoh",1,2,1)
```

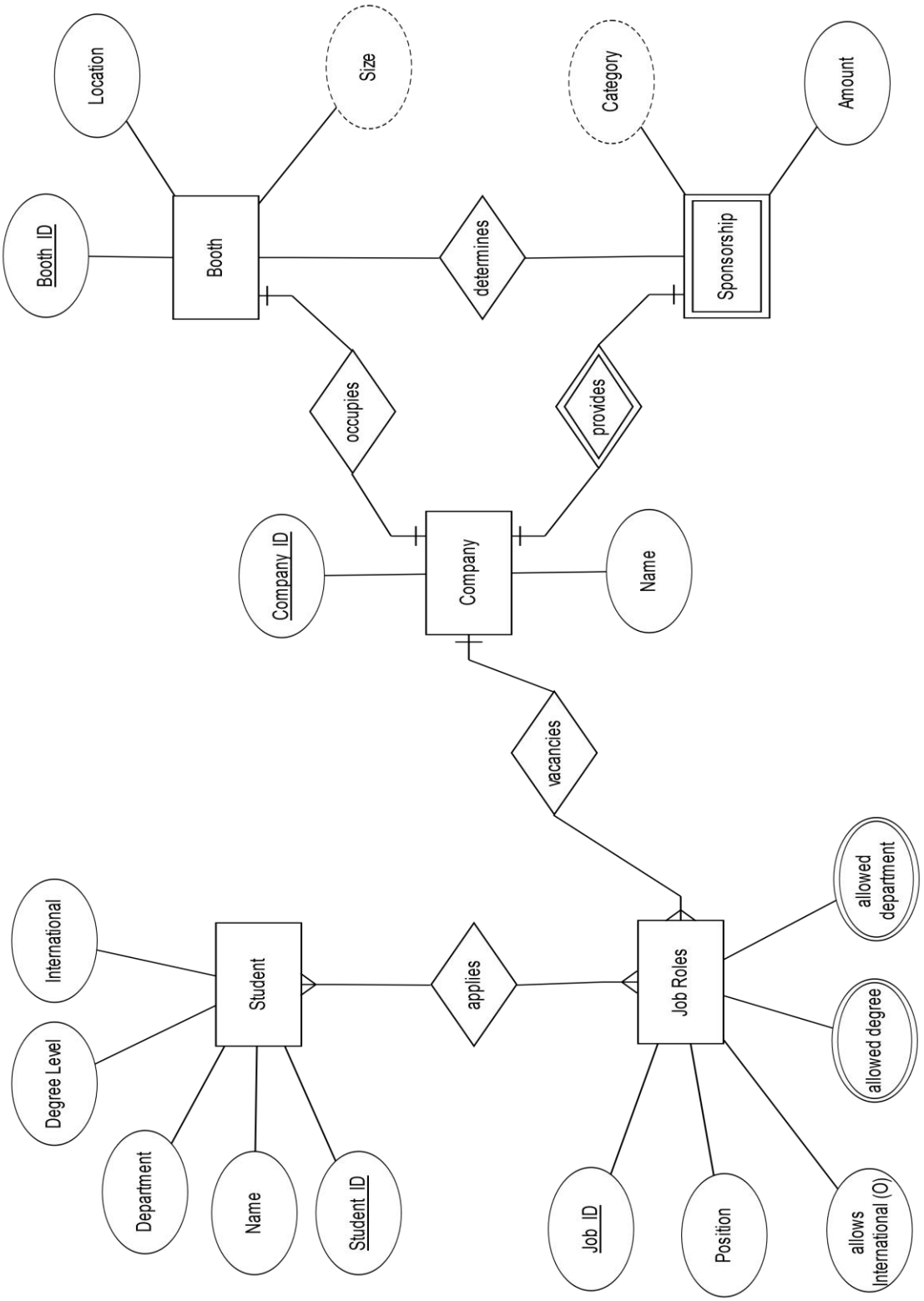
The Student table contains more than *1000 entries*:



Bulk Insert sql file for Company table:

```
INSERT INTO Company (id , Booth_id, name) VALUES (3400,2400,"Voolia"), (3401,2401,"Avaveo"), (3402,2402,"Kayveo"), (3403,2403,"Zoomlounge"), (3404,2404,"Mybuzz"), (3405,2405,"Flashdog"), (3406,2406,"Myworkso"), (3407,2407,"Edgeclub"), (3408,2408,"Buzzshare"), (3409,2409,"Centizu"), (3410,2410,"Pixoboo"), (3411,2411,"Quire"), (3412,2412,"Gabtune")
```

3. E-R Diagram


Location
Size
Category
Amount
Booth ID
Booth
determines
Sponsorship
occupies
provides
Company ID
Company
Name
vacancies
International
Degree Level
Department
Name
Student ID
Student
applies
Job Roles
allowed department
allowed degree
Job ID
Position
allows International (O)

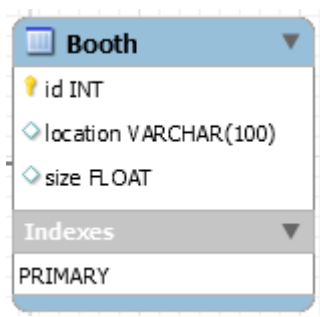
The E-R Diagram shown represents all the entities of the Database which are Student, Company, Sponsorship, Booth, and Job Roles. The attributes of Student entity are Student Id, Student Name, Department, Degree Level, and International classification. The entity company has only Company Id and name as its attributes. Sponsorship is a weak entity of parent entity Company and is determined completely using the attribute Company Id. It has its own attributes like Sponsorship amount and category where category is a derived attribute and is calculated using sponsorship amount. The booth entity is defined using the attributes Booth Id, location and Booth size. Here also Booth size is a derived attribute which is fixed for a particular Booth location, which is again set for a particular Company, based on the Sponsorship amount paid. Finally the Job Roles entity is composed of attributes like Job Id, position name, allowed departments, allowed Degree levels, and allows International Students. The allows International Student attribute is optional to specify and has a default value set in case not specified. The allowed Departments and allowed Degree level can have multiple values for a particular Job position creating Multi Valued Dependencies (MVDs) in the Job entity, which needs proper normalization to solve.

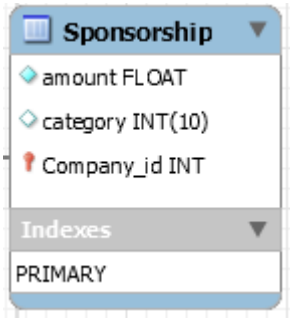
Relationships are described in the E-R Diagram among the declared entities. To begin with, the entity Company depends on Booth, having a relation *'Company occupies Booth*'. The Company and Sponsorship entities are connected by a weak relation '*Company provides Sponsorship*' and the Booth for a particular Company is determined using the Sponsorship amount, thus Booth is related to the Sponsorship entity as '*Sponsorship determines Booth*'. The entity Job is the prime connection between the Student and Company entities. A company has Job openings for which Students apply. Thus the entity Company is related to Job entity using the relationship '*Company* has *vacancies* for *Job Roles'* and the Student entity is connected with the Job entity via the relationship '*Student applies* for *Job Roles*'.

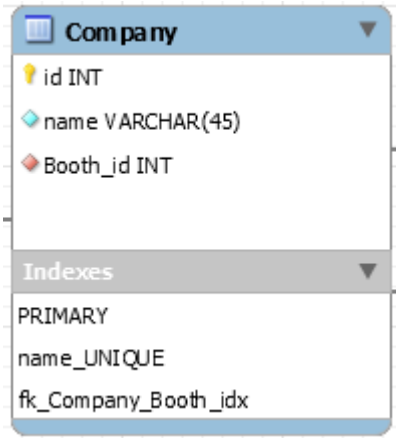
The E-R Diagram was generated using an online E-R diagram drawing tool (link: http://erdplus.com). It represents all the essential entities and relations required to create the Career Fair Database. But the entity Job Roles needs to be normalized before creating the database schema to avoid redundancies caused by having multiple values for two of its attributes.

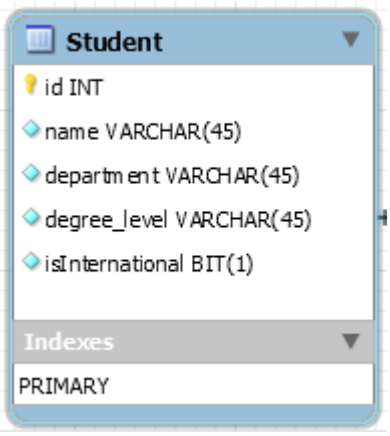
4. Table Normalization

Using the E-R Diagram drawn earlier we can create relations/tables describing all the entities and their relationships. The Student, Booth, Company and Sponsorship relation is already BCNF normalized and we can model them as:

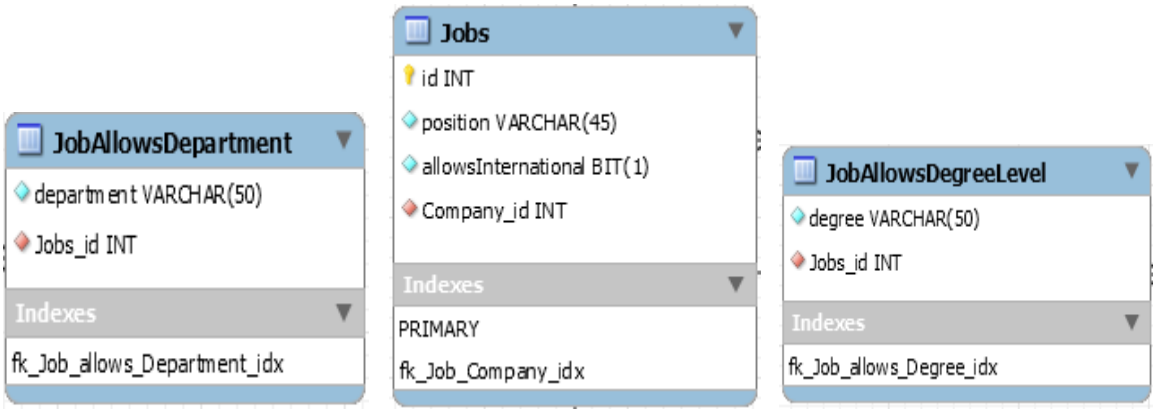




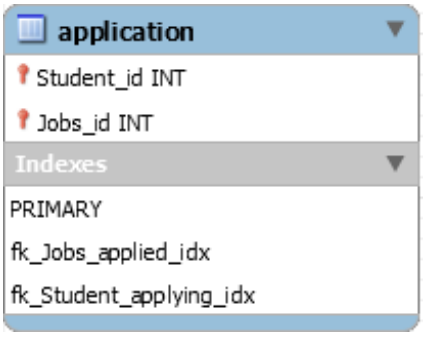




The relation Jobs describing the entity Job Roles contains **Multi Valued Dependencies (MVDs)** in the attributes *'allowed Departments'* and *'allowed Degree Levels'* as a particular job opening can be open for multiple Degree Levels spanning across various majors. To avoid the redundancies generated by inculcating the above two attributes in the Job relation can be resolved by applying **Fourth Normal Form (4NF).** Applying 4NF to the Job relation bifurcates the original relation into separate relation where each of those attribute would be kept in a separate relation identified by the primary key of parent relation. Thus we would have 3 different relations namely Jobs, JobAllowsDepartment, and JobAllowsDegreeLevels, which can be modeled as:

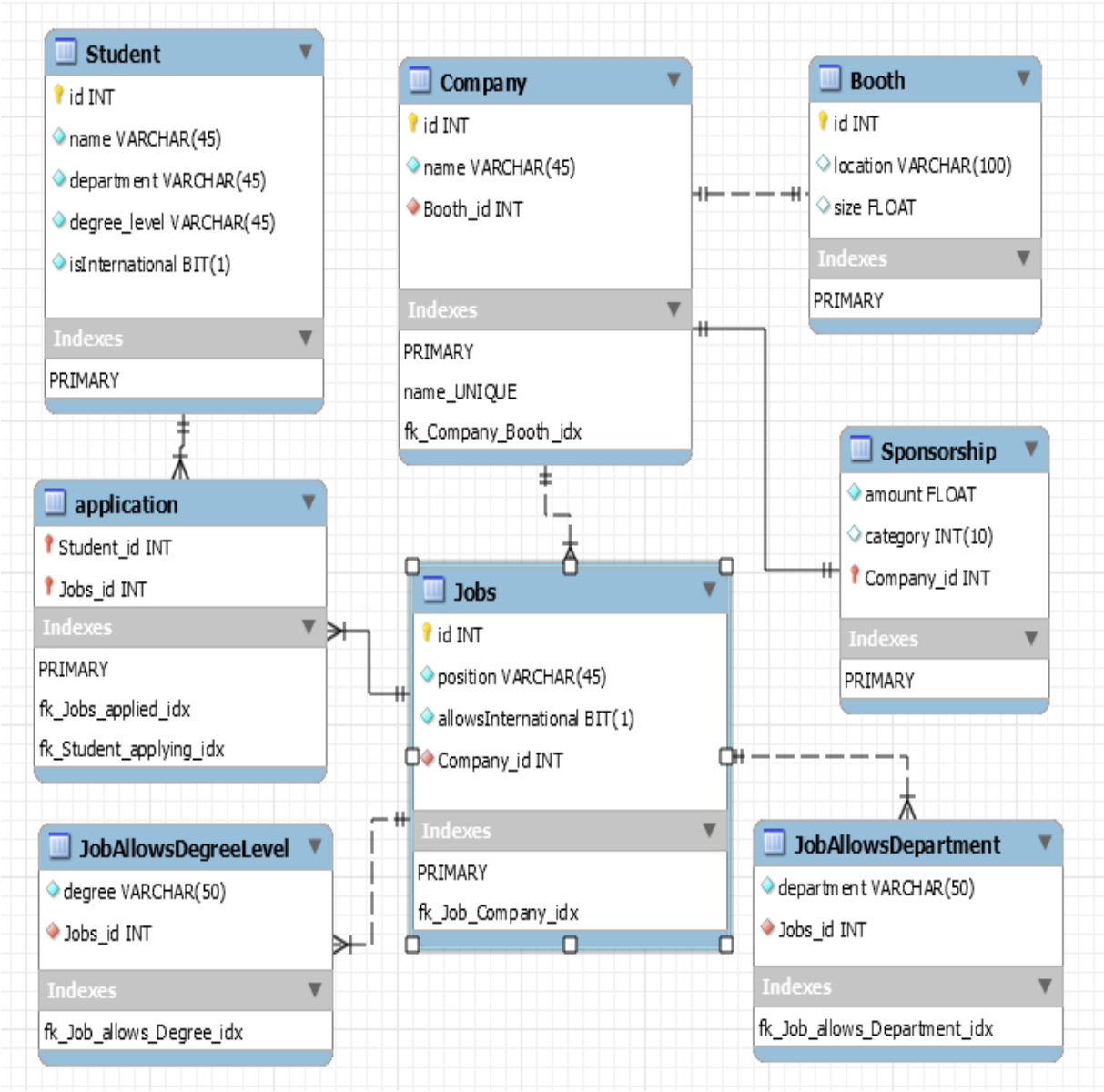


All the relationships except one are modeled using Foreign Key constraints in the relations itself. The relationship '*Student applies* for *Job Roles*' needs to be modeled separately as it is a "Many to Many" relationship where one student can apply for multiple Jobs and also multiple students can also apply for a single Job. We model this relation using a separate relation called Application.



This relation has Primary key as composite of both the attributes Student Id and Jobs Id. The entire UML Diagram can be seen on the next page.

UML Diagram:


Student
id INT
name VARCHAR(45)
department VARCHAR(45)
degree_level VARCHAR(45)
isInternational BIT(1)
Indexes
PRIMARY
Company
id INT
name VARCHAR(45)
Booth_id INT
Indexes
PRIMARY
name_UNIQUE
fk_Company_Booth_idx
Booth
id INT
location VARCHAR(100)
size FLOAT
Indexes
PRIMARY
application
Student_id INT
Jobs_id INT
Indexes
PRIMARY
fk_Jobs_applied_idx
fk_Student_applying_idx
Jobs
id INT
position VARCHAR(45)
allowsInternational BIT(1)
Company_id INT
Indexes
PRIMARY
fk_Job_Company_idx
Sponsorship
amount FLOAT
category INT(10)
Company_id INT
Indexes
PRIMARY
JobAllowsDegreeLevel
degree VARCHAR(50)
Jobs_id INT
Indexes
fk_Job_allows_Degree_idx
JobAllowsDepartment
department VARCHAR(50)
Jobs_id INT
Indexes
fk_Job_allows_Department_idx

5. User Interface

The UI for the Career Fair web application was made using HTML and Bootstrap and the backend code is written in PHP. It is very easy to use the application as it is having a smooth User Experience with proper redirection and Navigation panels for impromptu switching among pages. To start working, the user first need to open the project URL (https://may1sharma.000webhostapp.com/) in the browser. We have classified the user either as Student or Company. We will explain the user experience for both of them one by one.

For the Company as a User, it needs to be Registered first. User has to provide the Company name, Website and Sponsorship Amount for registering. It will create an account for the Company and it can then login anytime by just specifying the credentials. Successful Registration will redirect the Company user to Add New Job Position, which is basically a simple form describing the nature and essentialities of the required Job opening. The users will also have a Manage Jobs view where they can add, edit or delete Job openings. The page lists out all the necessary information about the Company and its posted Job openings. When user logs back in again at a later time, it would be redirected to this page.

For the Student as a User, there is a registration page which takes in the necessary inputs from the user to create an account. Upon successful registration, the user is provided with a recommendation list of some Job Positions depending on his registration information. He can apply to as many positions as he wants but this doesn't mean the job application has been submitted to the Company, rather it just for intimation and statistics on the Company side as how many students are interested in the position.

For the Guest users (Student orientated), there is a custom generic Job Search that has been implemented so they can have a look on the available Job positions. Job Search is conducted based on the specified Department and Degree Level of the Guest Student user. It also has an additional filter for selecting the jobs which are open for International Students.

6. Source Code

The complete source code is provided along with the report and I will provide proper snippets to explain the functioning and the steps followed for creating this web based Career Fair Database application. All the sql related code is present in the file *handler.php* inside "utils" folder.

- **Database Creation**

```
CREATE SCHEMA IF NOT EXISTS `mayank.sharma-CareerFair` DEFAULT CHARACTER SET utf8 ;
USE `mayank.sharma-CareerFair` ;
```

- **Table Creation**

```
CREATE TABLE IF NOT EXISTS `mayank.sharma-CareerFair`.`Student` (
        `id` INT NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(45) NOT NULL,
        `department` TINYINT(2) NOT NULL,
        `degree_level` TINYINT(2) NOT NULL,
        `isInternational` BIT(1) NOT NULL DEFAULT 0,
        PRIMARY KEY (`id`))

CREATE TABLE IF NOT EXISTS `mayank.sharma-CareerFair`.`Booth` (
        `id` INT NOT NULL AUTO_INCREMENT,
        `location` VARCHAR(100) NULL,
        `size` FLOAT UNSIGNED NULL DEFAULT 10.0 COMMENT 'in sqft',
        PRIMARY KEY (`id`))

CREATE TABLE IF NOT EXISTS `mayank.sharma-CareerFair`.`Company` (
        `id` INT NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(45) NOT NULL,
        `Booth_id` INT NOT NULL,
        PRIMARY KEY (`id`),
        CONSTRAINT `fk_Company_Booth1`
          FOREIGN KEY (`Booth_id`)
          REFERENCES `mayank.sharma-CareerFair`.`Booth` (`id`)
          ON DELETE NO ACTION
          ON UPDATE NO ACTION)
```

```
CREATE TABLE IF NOT EXISTS `mayank.sharma-CareerFair`.`Sponsorship` (
	`amount` FLOAT NOT NULL COMMENT 'in USD',
	`category` INT(1) GENERATED ALWAYS AS (amount / 5000) VIRTUAL,
	`Company_id` INT NOT NULL,
	PRIMARY KEY (`Company_id`),
	CONSTRAINT `fk_Sponsorship_Company1`
	  FOREIGN KEY (`Company_id`)
	  REFERENCES `mayank.sharma-CareerFair`.`Company` (`id`)
	  ON DELETE NO ACTION
	  ON UPDATE NO ACTION)

CREATE TABLE IF NOT EXISTS `mayank.sharma-CareerFair`.`Jobs` (
	`id` INT NOT NULL AUTO_INCREMENT,
	`position` VARCHAR(45) NOT NULL,
	`allowsInternational` BIT(1) NOT NULL DEFAULT 0,
	`Company_id` INT NOT NULL,
	PRIMARY KEY (`id`),
	CONSTRAINT `fk_Jobs_Company1`
	  FOREIGN KEY (`Company_id`)
	  REFERENCES `mayank.sharma-CareerFair`.`Company` (`id`)
	  ON DELETE NO ACTION
	  ON UPDATE NO ACTION)

CREATE TABLE IF NOT EXISTS `mayank.sharma-CareerFair`.`JobAllowsDepartment` (
	`department` TINYINT(2) NOT NULL,
	`Jobs_id` INT NOT NULL,
	PRIMARY KEY (`department`, `Jobs_id`),
	CONSTRAINT `fk_Jobs2`
	  FOREIGN KEY (`Jobs_id`)
	  REFERENCES `mayank.sharma-CareerFair`.`Jobs` (`id`)
	  ON DELETE NO ACTION
	  ON UPDATE NO ACTION)

CREATE TABLE IF NOT EXISTS `mayank.sharma-CareerFair`.`JobAllowsDegreeLevel` (
	`degree` TINYINT(2) NOT NULL,
	`Jobs_id` INT NOT NULL,
	PRIMARY KEY (`degree`, `Jobs_id`),
	CONSTRAINT `fk_Jobs1`
	  FOREIGN KEY (`Jobs_id`)
	  REFERENCES `mayank.sharma-CareerFair`.`Jobs` (`id`)
	  ON DELETE NO ACTION
	  ON UPDATE NO ACTION)
```

```
CREATE TABLE IF NOT EXISTS `mayank.sharma-CareerFair`.`Application` (
          `Student_id` INT NOT NULL,
          `Jobs_id` INT NOT NULL,
          PRIMARY KEY (`Student_id`, `Jobs_id`),
          CONSTRAINT `fk_Student_applying`
            FOREIGN KEY (`Student_id`)
            REFERENCES `mayank.sharma-CareerFair`.`Student` (`id`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION,
          CONSTRAINT `fk_Jobs_applied`
            FOREIGN KEY (`Jobs_id`)
            REFERENCES `mayank.sharma-CareerFair`.`Jobs` (`id`)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION)
```

- **Insert** (PHP code)

```
$insert_query = 'insert into Student(
          name,
          department,
          degree_level,
          isInternational
          )
          values  (
          "' . $formvars['name'] . '",
          "' . $formvars['department'] . '",
          "' . $formvars['degree'] . '",
          ' . $formvars['intl'] . '
          )';

$insert_query = 'INSERT INTO Application(Student_id, Jobs_id)
          VALUES (
          ' . $studentID . ',
          ' . $jobID . '
          )';
```

```
$insert_query = 'insert into Booth(
            location,
            size
            )    values    (
            "' . $formvars['location'] . '",
            ' . $formvars['size'] . '
            )';

$insert_query = 'insert into Company(
            name,
            Booth_id
            )    values    (
            "' . $formvars['name'] . '",
            ' . $boothID . '
            )';

$insert_query = 'insert into Sponsorship(
            amount,
            Company_id
            )    values    (
            ' . $formvars['amount'] . ',
            ' . $GLOBALS['companyID'] . '
            )';

$insert_query = 'insert into Jobs(
            position,
            allowsInternational,
            Company_id
            )    values    (
            "' . $formvars['position'] . '",
            ' . $formvars['intl'] . ',
            ' . $GLOBALS['companyID'] . '
            )';
```

```
$insert_query = 'insert into JobAllowsDepartment(
        department,
        Jobs_id
        )    values    (
        "' . $dept . '",
        ' . $formvars['jobID'] . '
        )';

$insert_query = 'insert into JobAllowsDegreeLevel(
        degree,
        Jobs_id
        )    values    (
        "' . $deg . '",
        ' . $formvars['jobID'] . '
        )';
```

- **Select**

```
$select_query = 'SELECT Jobs_id FROM Application WHERE
        Student_id = '.$studentID;

$check_query = 'SELECT id FROM Company WHERE name = "'.$name.'"';

$check_query = 'SELECT Booth.id as bID, Booth.location, Booth.size,
        Sponsorship.amount, Sponsorship.category FROM
        (Company INNER JOIN Booth ON
        Company.Booth_id = Booth.id) INNER JOIN Sponsorship
        ON Company.id = Sponsorship.Company_id
        WHERE Company.id = '.$id;

$select_query = 'SELECT id as jID, position, allowsInternational as intl
        FROM Jobs WHERE Jobs.Company_id = '.$companyID;
```

```
$select_query = 'SELECT department FROM JobAllowsDepartment
        WHERE Jobs_id = '.$jobID;

$select_query = 'SELECT degree FROM JobAllowsDegreeLevel
        WHERE Jobs_id = '.$jobID;

$search_query = 'SELECT Jobs.id as jID, Company.name as cName,
        Jobs.position, Booth.id as bID, Booth.location FROM
        ((Jobs INNER JOIN JobAllowsDepartment ON Jobs.id =
        JobAllowsDepartment.Jobs_id) INNER JOIN JobAllowsDegreeLevel
        ON Jobs.id = JobAllowsDegreeLevel.Jobs_id) INNER JOIN
        (Company INNER JOIN Booth ON Company.Booth_id = Booth.id)
        ON Jobs.Company_id = Company.id WHERE
        JobAllowsDepartment.department = '. $department .
        ' and JobAllowsDegreeLevel.degree = '. $degree.
        ' and Jobs.allowsInternational = '. $intl ;
```

- **Update**

```
$update_query = 'UPDATE Jobs SET position = "' . $formvars['position'] .
        '",  allowsInternational = '.$formvars['intl'].'
         WHERE id = ' . $formvars['jobID'] ;
```

- **Delete**

```
$delete_query = 'DELETE FROM JobAllowsDepartment WHERE
         Jobs_id = '.$jobID;

$delete_query = 'DELETE FROM JobAllowsDegreeLevel WHERE
        Jobs_id = '.$jobID;

$delete_query = 'DELETE FROM Jobs WHERE id = '.$jobID;
```

7. Discussion

It was great learning the niches of Database system design. I learned a lot of new things while working on this project. I learned a new coding language PHP and creating dynamic web pages it. I learned the end to end process of how to create a web application. The complete web application is composed of 3 parts: Database design, Front End UI and the backend connecting the UI to Database. I learnt how to create E-R diagram and the minute details it represent. A good database design is created only after normalizing the data. I didn't know how to normalize Multi Valued Dependency earlier, which after reading the text book and online tutorials was resolved and I applied Fourth Normalization Form on my Database in order to avoid the redundancy of data. I also learnt how to use a variety of new tools like MySQL Work Bench & phpMyAdmin, and website hosting.

The step by step approach of Database System designing is the value that got added to my skill set after completing this project.