MSCI 346

**“Task Trader”**

**Database Project**

Deliverable to Professor B. Ghaddar

June 22nd, 2017

Aaron Yim 20555691

Ayser Choudhury 20603805

Victoria Li 20622970

# Description of the Project

Our web application is an internal task board meant to be a System as a Service (SaSS) for companies. Through the use of the job board, employees of the same company (i.e. Maple Leaf Foods) have the ability to apply for short task from various departments within their own company in order to increase versatility, encourage and motivate employees to learn skillsets that are beyond their expectations of their current job description.

The goal of the job board is to help redistribute existing skills and talents within a given company, improve productivity, worker engagement and transparency while also giving employees an insight into how other roles and departments operate, and to give them opportunities they otherwise would not have, without creating major change, thereby enriching their overall experience at the company.

# Requirements Analysis

## Company

Each company will be able to utilize the application and is completely separate from every other company. Our Web application will identify each company uniquely by its company name.

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attributes** | **Relationships** |
| Company | * company\_name (primary key) | * Located at a **location** * Has multiple **departments** * Has multiple **employees** |

**Relationship Constraints**

* Company has departments
  + Min: A company must have at least one department
  + Max: A company has multiple departments (i.e. Engineering team, Design team, Finance Team, Business Development Team, etc).
* Company has locations
  + Min: a company must have at least one location
  + Max: departments can be located in various locations.
* Company has employees
  + Min: a company must at least have one employee
  + Max: a company must

**Constraints**

company\_name is set as NOT NULL as it is the Primary Key

:

Assumptions:

Note that the relationship of department works under employees holds the relationship of the department where the employee actually works and not the department at which the task they’re applying for is.

## Department

A company can have many departments each can be uniquely identified by their dept\_num. The entity also holds the department name and holds relationships to company, employees, locations and tasks.

* Department is under a company
  + Min: Department must be part of a company to exist
  + Max: Department can only be part of one company
* Department holds tasks
  + Min: Department may not have any tasks under the specific department
  + Max: Department can have as many tasks as they need
* Department has employees that works under it
  + Min: Departments must have at least one employee
  + Max: Departments can have as many employees as they desire to hire
* Department has a location
  + Min: departments must have one location
  + Max: departments can have various locations (i.e. There can be a UX Department in the Seattle location and one in Los Angeles)

**Constraints:**

* dept\_number is set to NOT NULL because it is the primary key
* Dept\_name is set to NOT NULL since a department must have a name
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted, as it is no longer relevant

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attributes** | **Relationships** |
| Department | * dept\_num (primary key) * dept\_name * location | * Is under a **company** * **Task** is under a department * **Employees** work under department |

## Location

Companies will have various locations and departments reside in these various locations. We can uniquely identify each location by it’s campus\_name and hold the composite address of the location. We also want to identify where the location of the task so that when employees apply or post a job, they can identify where the job may be located (assume that jobs can also be remote (i.e. programming task)

* Location is located for a Company
  + Min: A company can have a minimum of 1 location
  + Max: A company can have as many locations as they need (i.e. Google has an office in Silicon Valley, Berlin, etc.)
* Location is located for a task
  + Min: location doesn’t have to have an available task
  + Max: locations have many available tasks
* Location is located for department
  + Min: each department must have at least one location
  + Max: there can be many locations where each department exists

**Constraints:**

* Campus\_name is set to NOT NUll as it is the Primary key
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attributes** | **Relationships** |
| Location | * campus\_name (primary key) * Address (composite)   + city   + country   + postal\_Code   + street\_address | * **Company** has several locations * **Tasks** have several locations * Each location has departments |

## Employee

Employees are one of the most important entity. It holds all the information about each employee as well as a weak entity and relationship for the account information. Each employee will have an empoyee\_id which we can uniquely identify. The database will also hold first name, last name, job title, supervisor’s name as well as a multi-variable attribute of skills which can hold multiple skills of the employee which will help task search and help task posters a way of filtering out candidates. Employees can apply to tasks and well as post tasks to hire employees. There is also a recursive relationship with employees where each employee can hire another employee.

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attributes** | **Relationships** |
| Employee | * E\_ssn (primary key) * Job\_title * Skills (multi-variable) * First\_name * Last\_name * supervisor\_id | * Works for a **department** * Works for a **company** * Posts **tasks** * Applies to **tasks** * Is a **stakeholder** * Is authenticated by an **account** |

Each employee will have an account that they can log in to the job portal. This is why account is a weak entity since it would not exist with employees. We also have a weak entity for all attachments (i.e resume, cover letters) since file system storage holds a lot of data, information and expensive to query, we want to minimize cost and efficiency by having all file storage as a separate entity.

**Constraints**

* E\_ssn is set to not null because it is a primary key

**Relationship Constraints**

* Employees work for a department
  + Min: Employees can work for a minimum of one department
  + Max: Employees can ony work for a maximum of one department
  + Assumptions: As mentioned above, the department that Employees work under applies to the department that the employee’s full time job is under.
* Employees work for a company
  + Min: Employees can only work for a minimum of one company
  + Max: Employees can work for a maximum of one company
* Employees can post to tasks
  + Min: Employees do not need to post any tasks (minimum 0 tasks)
  + Max: Employees can post as many tasks as they need to
* Employees can apply to tasks
  + Min: Employees have the ability to apply to 0 tasks
  + Max: Employees can apply to as many tasks as they want
* Employees can be a stakeholder
  + Min: Employees must have a minimum of one stakeholder for a task (stakeholder refers to the employee that posts the task as well as any person who is specifically interested following along the task but not an employee completing the task (i.e. supervisor, manager, team-lead, advisor)
  + Max: There can be numerous amounts of employees that can be a stakeholder for a task
* Employees are authenticated to our web applications system by an account
  + Min: Employees must have one account to access the web application
  + Max: Employees can only have one account.

|  |  |  |
| --- | --- | --- |
| **Entity (Weak)** | **Attributes** | **Relationships** |
| Account | * Username (partial key) * Password * Profile\_pic\_url | * Authenticated by **employee** |

**Constraints:**

* Username is set to NOT NULL because it is a unique partial key
* ESSN is set as a foreign key to refer to the employee who the account belongs to in the Employee table
* Foreign key E\_ssn ON DELETE attribute is set to CASCADE because if the employee (parent table) is deleted, the employee’s account is no longer relevant

The Task entity will hold information about the job tasks. It has task\_id which uniquely identifies each task, task\_title, task\_description. We also want to hold attributes that can help employees filter out the specific time commitment of a task. The attribute time\_commitment is an attribute that holds the number of days the job requires an employee to commit per time frame and the start\_date and end\_date show the length of the task.

* Task has a location
  + Min: Task must have a location pinned on it (i.e. specific campus or remote)
  + Max: Tasks can only have one location pinned on it
* Tasks are posted by an employee
  + Min: There can be no employees posting a task. (i.e. no tasks available)
  + Max: There can be many employees posting tasks
* Tasks are applied by employees
  + Min: There can be no employees applying to the task
  + Max: There can be various employees applying to the task
* Task is under a specific department
  + Min: Task must be under one department
  + Max: Task can only be under one department
* Tasks can be viewed by stakeholders
  + Min: There must be one stakeholder for a task (i.e. poster)
  + Max: There can be many stakeholders (i.e. employees that follow along the task such as a supervisor, advisor, department lead, client)

**Constraints**

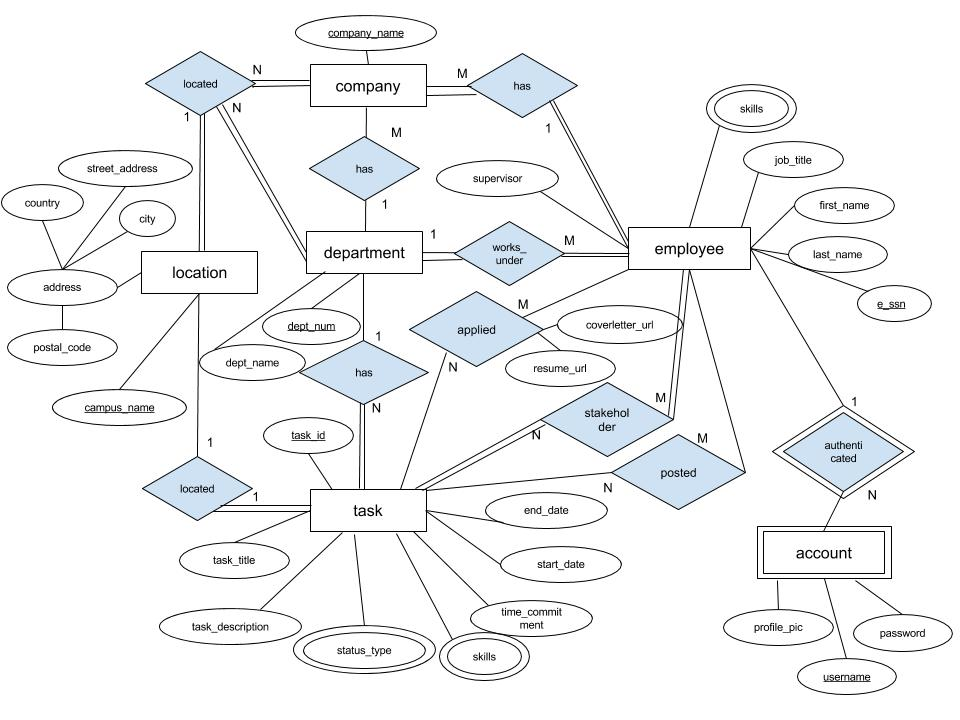
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted
* The task description is set to the data type of clob, as the length of the description could potentially be long

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attributes** | **Relationships** |
| Task | * Task\_id (primary key) * Task Title * Task\_description * End\_date * Start\_date * Time\_comitment * Location * Status (multivalue) | * Located at a **location** * Task is posted by **employee** * Task is applied by **employee** * Task belongs a department * Task has stakeholders * Task is posted by employees |

* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted
* The task description is set to the data type of clob, as the length of the description could potentially be long

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attributes** | **Relationships** |
| Applied | * E\_ssn * task\_id * Resume * Cover Letter | * Each application belongs to a singular task and employee |

**ER Diagram**

****

**Create Tables**

CREATE TABLE Company (

company\_name VARCHAR(30) NOT NULL,

PRIMARY KEY (company\_name),

);

* company\_name is set as NOT NULL as it is the Primary Key

CREATE TABLE Employee (

Job\_title VARCHAR(30),

First\_name VARCHAR(30),

Last\_name VARCHAR(30),

E\_ssn INT NOT NULL,

Supervisor\_id INT,

PRIMARY KEY (E\_ssn),

);

* E\_ssn is set to not null because it is a primary key

CREATE TABLE Account (

Profile\_pic\_url VARCHAR(30),

Username VARCHAR(30) NOT NULL,

Password VARCHAR(30),

E\_ssn INT

PRIMARY KEY (E\_ssn),

FOREIGN KEY (E\_ssn) REFERS TO Employee (E\_ssn),

ON DELETE CASCADE

PARTIAL KEY (username)

);

* Username is set to NOT NULL because it is a unique partial key
* ESSN is set as a foreign key to refer to the employee who the account belongs to in the Employee table
* Foreign key E\_ssn ON DELETE attribute is set to CASCADE because if the employee (parent table) is deleted, the employee’s account is no longer relevant

CREATE TABLE Employee\_skills (

E\_ssn INT UNIQUE NOT NULL,

Skills VARCHAR(50) NOT NULL,

PRIMARY KEY (E\_ssn),

FOREIGN KEY (E\_ssn) REFERS TO Employee(E\_ssn),

ON DELETE CASCADE

);

* E\_ssn is set to NOT NULL because it is the primary key
* Skill is set to NOT NULL, because if there is a skill entry, there must be a skill
* Foreign key E\_ssn ON DELETE attribute is set to CASCADE because if the employee (parent table) is deleted, the employee’s skills are no longer relevant

CREATE TABLE Department (

dept\_num INT NOT NULL,

dept\_name VARCHAR(30) NOT NULL,

location VARCHAR(30),

PRIMARY KEY (dept\_num),

FOREIGN KEY (company\_name) REFERS TO Company(company\_name),

ON DELETE CASCADE

);

* dept\_number is set to NOT NULL because it is the primary key
* Dept\_name is set to NOT NULL since a department must have a name
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted, as it is no longer relevant

CREATE TABLE Location (

campus\_name VARCHAR(30) NOT NULL,

city VARCHAR(30),

country VARCHAR(30),

postal\_code VARCHAR(30),

street\_address VARCHAR(30),

PRIMARY KEY (Campus\_name),  
 FOREIGN KEY (company) REFERS TO company(company\_name)

ON DELETE CASCADE

);

* Campus\_name is set to NOT NUll as it is the Primary key
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted

CREATE TABLE Task (

Task\_id INT NOT NULL,

Task\_title VARCHAR(100),

Task\_description CLOB,

End\_date DATE,

Start\_date DATE,

Time\_commitment INT,

Location VARCHAR(30)

PRIMARY KEY (Task\_id)

FOREIGN KEY (Location) REFERS TO Location(campus\_name),

ON DELETE CASCADE

FOREIGN KEY (Campus\_name) REFERS TO Company(Location)

ON DELETE CASCADE

);

* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted
* The task description is set to the data type of clob, as the length of the description could potentially be long

CREATE TABLE Status (

Status\_type VARCHAR(50) NOT NULL,

PRIMARY KEY (Status\_type),

FOREIGN KEY (Status\_type) REFERS TO Task(Status\_type)

ON DELETE CASCADE

);

* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted

CREATE TABLE Task\_skills (

Skills VARCHAR(50) NOT NULL,

PRIMARY KEY (skills),

FOREIGN KEY (skills) REFERS TO Task(skills)

);

* Skulls is set as NOT NULL as it is a Primary KEy
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted

CREATE TABLE Applied\_to\_task (

Task\_id INT NOT NULL,

E\_ssn INT NOT NULL,

Resume\_url VARCHAR(100),

Cover\_letter VARCHAR(100),

PRIMARY KEY (task\_id),

PRIMARY KEY (e\_ssn),

FOREIGN KEY (task\_id) REFERS TO TASK(task\_id)

ON DELETE CASCADE,

FOREIGN KEY (e\_ssn) REFERS TO EMPLOYEE(e\_ssn)

ON DELETE CASCADE

);

* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted
* Task\_id and E\_ssn are set to NOT NULL as they refer to the specific instance of an interaction between the an employee and task.

CREATE TABLE Posted\_to\_task (

Task\_id INT NOT NULL,

E\_ssn INT NOT NULL,

PRIMARY KEY (Task\_id),

PRIMARY KEY (E\_ssn),

FOREIGN KEY (Task\_id) REFERS TO TASK(Task\_id)

ON DELETE CASCADE,

FOREIGN KEY (e\_ssn) REFERS TO EMPLOYEE(e\_ssn)

ON DELETE CASCADE

);

* Task\_id and E\_ssn are set to NOT NULL as they refer to the specific instance of an interaction between the an employee and task.
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted

CREATE TABLE department\_has\_task (

Task\_id INT NOT NULL,

Dept\_num INT NOT NULL,

PRIMARY KEY (task\_id),

PRIMARY KEY (dept\_num),

FOREIGN KEY (task\_id) REFERS TO task(task\_id),

ON DELETE CASCADE

FOREIGN KEY (dept\_num) REFERS TO department(dept\_num)

ON DELETE CASCADE

);

* Task\_id and Dep\_num are set to NOT NULL as they refer to the specific instance of an interaction between the an employee and task.
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted

CREATE TABLE task\_stakeholders (

task\_id INT NOT NULL,

e\_ssn INT NOT NULL,

resume\_url VARCHAR(75),

coverletter\_url VARCHAR(75),

PRIMARY KEY (task\_id, e\_ssn),

FOREIGN KEY (task\_id) REFERS TO task(task\_id)

ON DELETE CASCADE,

FOREIGN KEY (e\_ssn) REFERS TO Employee(E\_ssn)

ON DELETE CASCADE

);

* Task\_id is set to NOT NULL as the each the task the stakeholder relationship should be based needs to be identified
* E\_ssn is set to NOT NUll as the stakeholders working on each task needs to be identified
* ON DELETE the the Foreign Keys CASCADE as when a record in the parent table is deleted, the related information should also be deleted