SOEN 363 - Database Systems for Software Engineer
Assignment 1

Ali Hanni - 40157164

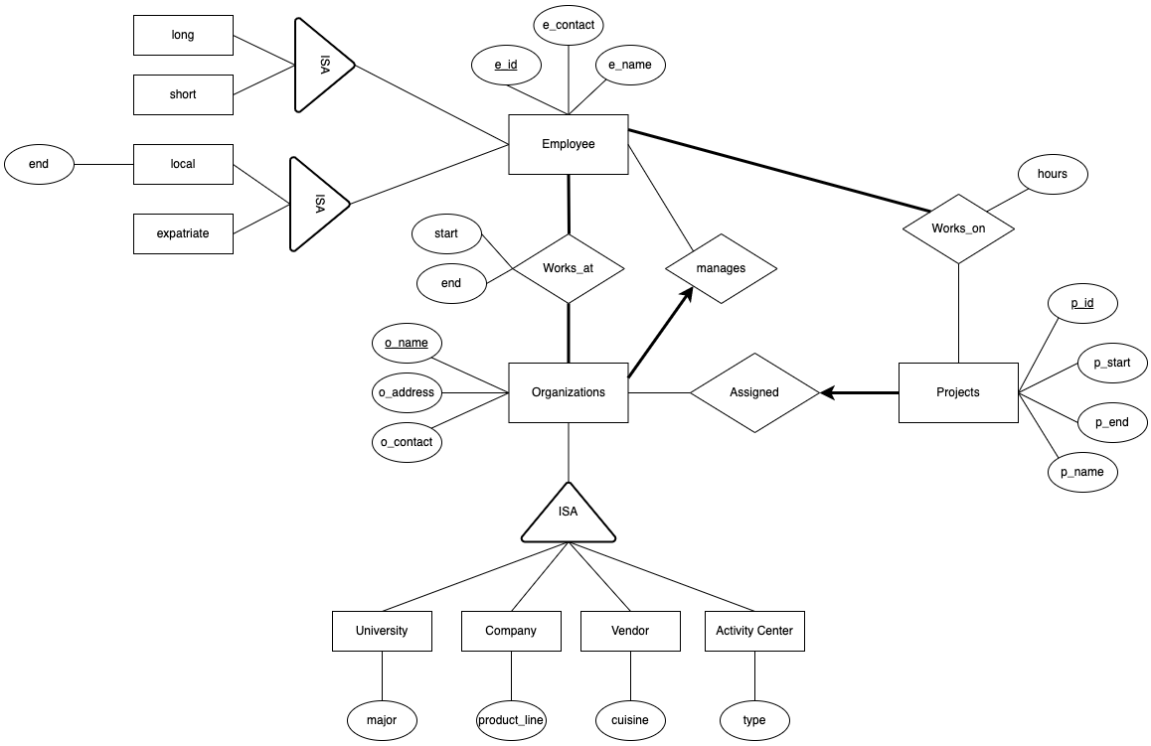
Gina Cody School of Computer Science and Software Engineering
Concordia University, Montreal, QC, Canada

Winter 2022

Part 1 - Building a Database for Concordia Foundation

(a) In the *Assigned* relationship, not all organizations are necessarily assigned a project, and organizations can be assigned multiple projects, hence the line from *Organizations* to *Assigned* is thin and do not have an arrow. However, for the line between *Assigned* and *Projects*, we are told that each project is assigned **to one and only one** organization. That means that it is a one-to-many total relationship. The line from *Projects* to *Assigned* is thick (total) ans has an arrow (one-to-many). Note that we are assuming that a project has to be assigned to a Organization, that why we use a total relationship. Here is the corrected ER diagram representing the *Assigned*.

(b) Bellow is the full ER diagram for the C.F. database.



We are assuming that all organizations have at least one employee.

(c) Since the organizations are classified as either a university, a company, a vendor, or a activity center, we can say that these 'categories' cover all the instances

of the Organizations entity set. An organization cannot be classified as two different type. The four possible types of organization are disjoint subsets of the Organization entity set. Hence, the Organizations ISA hierarchy has covering constraints.

(d) First, we know that the **long** and *short* subsets cover *Employees*. We also know that The **local** and *expatriate* subsets cover *Employees*. However, **long**, **short**, **local**, and **expatriate** subsets overlap employee since **long** and **short** are not disjoint from **local** and **expatriate**. For example, a long term employee can be expatriate while another long term employee is local.

(e)

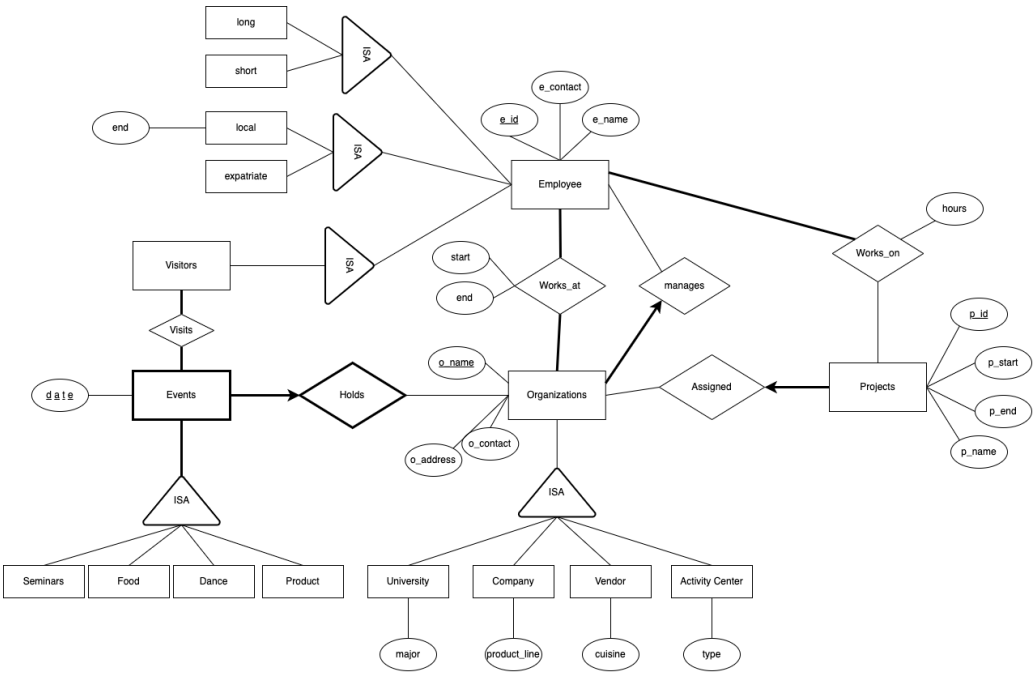
```
CREATE TABLE Organizations(
    o_name: CHAR(20) NOT NULL,
    o_address: CHAR(20),
    o_contact: CHAR(20),
    PRIMARY KEY (o_name),
    UNIQUE (o_name)
    );

CREATE TABLE Employees(
    e_id: CHAR(20) NOT NULL,
    e_contact: CHAR(20),
    e_name: CHAR(20),
    PRIMARY KEY (e_id),
    UNIQUE (e_id)
    );

CREATE TABLE Projects(
    p_id: CHAR(20) NOT NULL,
    p_start: DATE,
    p_end: DATE
    p_name: CHAR(20),
    PRIMARY KEY (p_id),
    UNIQUE (p_id)
    );

CREATE TABLE University(
    o_name: CHAR(20) NOT NULL,
    major: CHAR(20)
    PRIMARY KEY (o_name),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Company(
```

```
    o_name: CHAR(20) NOT NULL,
    product_line: CHAR(20)
    PRIMARY KEY (o_name),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Vendor(
    o_name: CHAR(20) NOT NULL,
    cuisine: CHAR(20)
    PRIMARY KEY (o_name),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Activity Center(
    o_name: CHAR(20) NOT NULL,
    type: CHAR(20)
    PRIMARY KEY (o_name),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Works_at (
    o_name: CHAR(20) NOT NULL,
    e_id: CHAR(20) NOT NULL,
    end: DATE,
    start: DATE,
    PRIMARY KEY (o_name, e_id),
    FOREIGN KEY (o_name) REFERENCES Organizations,
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE Works_on (
    p_id: CHAR(20) NOT NULL,
    e_id: CHAR(20) NOT NULL,
    hours: TIME,
    PRIMARY KEY (p_id, e_id),
    FOREIGN KEY (p_id) REFERENCES Projects,
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE Assigned (
    o_name: CHAR(20) NOT NULL,
    p_id: CHAR(20) NOT NULL,
    PRIMARY KEY (p_id, o_name),
```

```
    FOREIGN KEY (p_id) REFERENCES Projects,
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Manages (
    o_name: CHAR(20) NOT NULL,
    e_id: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id, o_name),
    FOREIGN KEY (e_id) REFERENCES Employees,
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE long (
    e_id: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE short (
    e_id: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE local (
    e_id: CHAR(20) NOT NULL,
    end: DATE,
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE expatriate (
    e_id: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
    );
```

Part 2 - Extending CF's database

(a) Bellow is the full ER diagram for the C.F. database.



We assume that all events have at least one visitor.

(b)

```
CREATE TABLE Organizations(
    o_name: CHAR(20) NOT NULL,
    o_address: CHAR(20),
    o_contact: CHAR(20),
    PRIMARY KEY (o_name),
    UNIQUE (o_name)
    );

CREATE TABLE Employees(
    e_id: CHAR(20) NOT NULL,
    e_contact: CHAR(20),
    e_name: CHAR(20),
    PRIMARY KEY (e_id),
    UNIQUE (e_id)
    );

CREATE TABLE Projects(
    p_id: CHAR(20) NOT NULL,
    p_start: DATE,
    p_end: DATE
    p_name: CHAR(20),
    PRIMARY KEY (p_id),
    UNIQUE (p_id)
```

```
    );

CREATE TABLE University(
    o_name: CHAR(20) NOT NULL,
    major: CHAR(20)
    PRIMARY KEY (o_name),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Company(
    o_name: CHAR(20) NOT NULL,
    product_line: CHAR(20)
    PRIMARY KEY (o_name),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Vendor(
    o_name: CHAR(20) NOT NULL,
    cuisine: CHAR(20)
    PRIMARY KEY (o_name),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Activity Center(
    o_name: CHAR(20) NOT NULL,
    type: CHAR(20)
    PRIMARY KEY (o_name),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Works_at (
    o_name: CHAR(20) NOT NULL,
    e_id: CHAR(20) NOT NULL,
    end: DATE,
    start: DATE,
    PRIMARY KEY (o_name, e_id),
    FOREIGN KEY (o_name) REFERENCES Organizations,
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE Works_on (
    p_id: CHAR(20) NOT NULL,
    e_id: CHAR(20) NOT NULL,
```

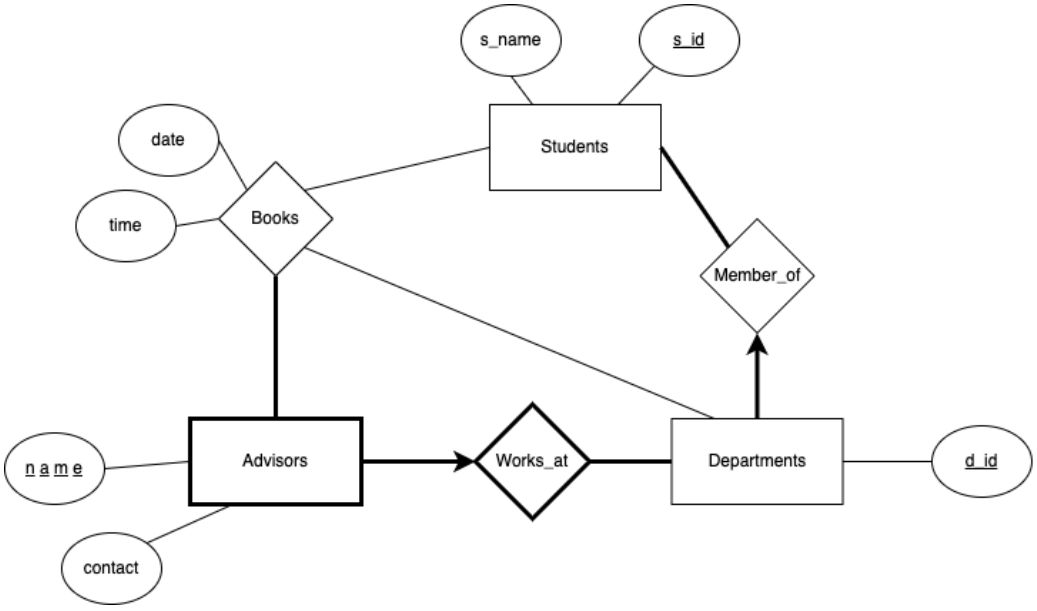
```
    hours: TIME,
    PRIMARY KEY (p_id, e_id),
    FOREIGN KEY (p_id) REFERENCES Projects,
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE Assigned (
    o_name: CHAR(20) NOT NULL,
    p_id: CHAR(20) NOT NULL,
    PRIMARY KEY (p_id, o_name),
    FOREIGN KEY (p_id) REFERENCES Projects,
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Manages (
    o_name: CHAR(20) NOT NULL,
    e_id: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id, o_name),
    FOREIGN KEY (e_id) REFERENCES Employees,
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE long (
    e_id: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE short (
    e_id: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE local (
    e_id: CHAR(20) NOT NULL,
    end: DATE,
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE expatriate (
    e_id: CHAR(20) NOT NULL,
```

```
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
    );

CREATE TABLE Events (
    o_name: CHAR(20) NOT NULL,
    date: DATE NOT NULL,
    PRIMARY KEY (o_name, date),
    FOREIGN KEY (o_name) REFERENCES Organizations
    );

CREATE TABLE Seminars (
    o_name: CHAR(20) NOT NULL,
    date: DATE NOT NULL,
    PRIMARY KEY (o_name, date),
    FOREIGN KEY (o_name, date) REFERENCES Events
    );

CREATE TABLE Food (
    o_name: CHAR(20) NOT NULL,
    date: DATE NOT NULL,
    PRIMARY KEY (o_name, date),
    FOREIGN KEY (o_name, date) REFERENCES Events
    );

CREATE TABLE Dance (
    o_name: CHAR(20) NOT NULL,
    date: DATE NOT NULL,
    PRIMARY KEY (o_name, date),
    FOREIGN KEY (o_name, date) REFERENCES Events
    );

CREATE TABLE Product (
    o_name: CHAR(20) NOT NULL,
    date: DATE NOT NULL,
    PRIMARY KEY (o_name, date),
    FOREIGN KEY (o_name, date) REFERENCES Events
    );

CREATE TABLE Visitors (
    e_id: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id),
    FOREIGN KEY (e_id) REFERENCES Employees
```

```
    );

CREATE TABLE Visits (
    e_id: CHAR(20) NOT NULL,
    date: DATE NOT NULL,
    o_name: CHAR(20) NOT NULL,
    PRIMARY KEY (e_id, date, o_name),
    FOREIGN KEY (e_id) REFERENCES Employees,
    FOREIGN KEY (o_name, date) REFERENCES Events
    );
```

Part 3 - Building a database for Concordia Academic Advising System

(a) Bellow is the full ER diagram for the database.



We assume that all advisors have been booked for an appointment at least once.

(b)
```
CREATE TABLE Students (
    s_id: CHAR(20) NOT NULL,
    s_name: CHAR(20),
    PRIMARY KEY (s_id)
    );

CREATE TABLE Departments (
```

```
    d_id: CHAR(20) NOT NULL,
    PRIMARY KEY (d_id)
    );

CREATE TABLE Advisors (
    name: CHAR(20) NOT NULL,
    d_id: CHAR(20) NOT NULL,
    contact: CHAR(20),
    PRIMARY KEY (d_id, name),
    FOREIGN KEY (d_id) REFERENCES Departments
    );

CREATE TABLE Books (
    s_id: CHAR(20) NOT NULL,
    name: CHAR(20) NOT NULL,
    d_id: CHAR(20) NOT NULL,
    date: DATE,
    time: TIME,
    PRIMARY KEY (s_id, name, d_id),
    FOREIGN KEY (s_id) REFERENCES Students,
    FOREIGN KEY (name, d_id) REFERENCES Advisors
    );

CREATE TABLE Works_on (
    name: CHAR(20) NOT NULL,
    d_id: CHAR(20) NOT NULL,
    PRIMARY KEY (d_id, name),
    FOREIGN KEY (d_id) REFERENCES Departments,
    FOREIGN KEY (name) REFERENCES Advisors
    );

CREATE TABLE Member_of (
    s_id: CHAR(20) NOT NULL,
    d_id: CHAR(20) NOT NULL,
    PRIMARY KEY (d_id, s_id),
    FOREIGN KEY (d_id) REFERENCES Departments,
    FOREIGN KEY (s_id) REFERENCES Students
    );
```

(c)

```
CREATE VIEW Advising_History (s_id, date, time)
    AS SELECT S.s_id, S.s_name, B.date, B.time, B.name
    FROM S Students, B Books,
```

```
WHERE S.s_id = B.S_id,
```