University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

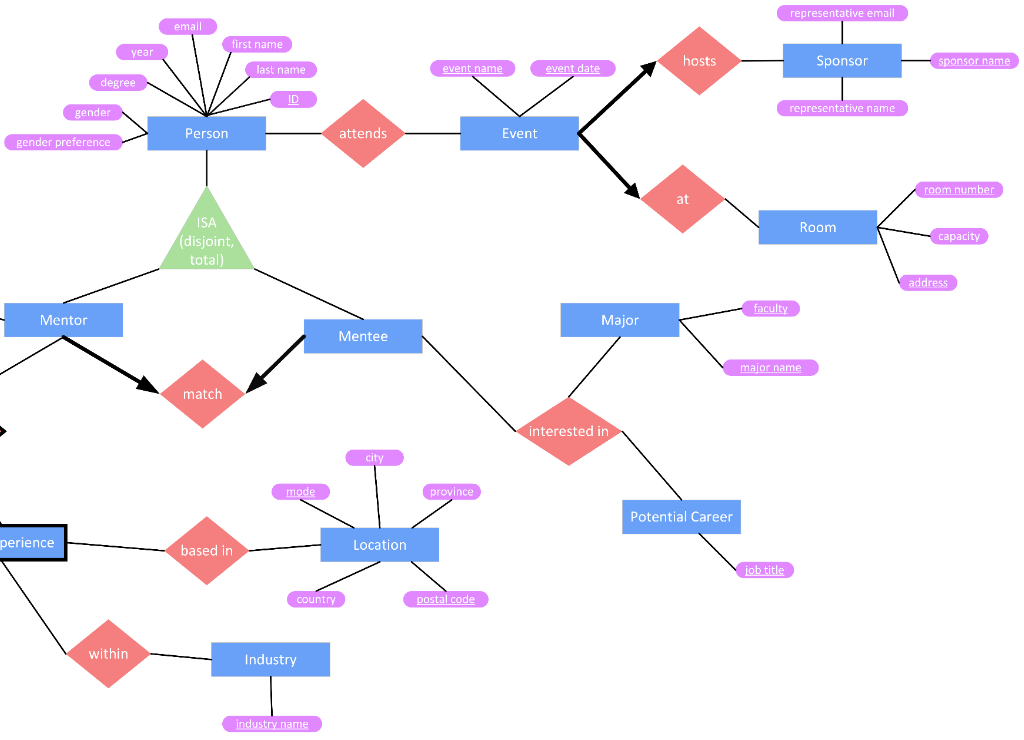
Milestone #: 2

Date: October 21, 2022

Group Number: 28

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Sandra Radic	23018088	w1f3b	sandra_radic@outlook.com
Sarah Wong	43440171	q9k0c	sarah.wong@shaw.ca
Carina Tze	77998987	s6y2b	carinatze0@gmail.com

2. ER Diagram



Changes made to ER Diagram:

- Outlined "holds" in weak entity with Work Experience
- Turned Matched into relationship "match", removed two "paired in" relationships that was originally between “Mentor and match” and “Mentee and match”
    - Mentor and Mentee are in a one-to-one relationship now
- Added attributes: mode, city, province and postal code to Location entity
- Added attribute: salary to Work Experience entity
- Removed Location from "interested in" relationship
- Removed "under" relationship
- Renamed venue name attribute to room number in Room

3. The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:

a. **List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...))**
b. **Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints that the table must maintain.**

Legend:

- <u>Primary key (PK)</u>
- **Foreign key (FK)**
- *Candidate key (CK)*

Person(<u>pid</u>: integer, email: string, *firstName*: string, *lastName*: string, *year*: integer, genderPreference: string, gender: string, degree: string)

WorkExperience (company: string, jobTitle: string, duration: integer, **pid**: integer, salary: integer)

- There are no CKs.

Mentor(major: string, **pid**: integer)

- There are no CKs.

Mentee(**pid**: integer)

- There are no CKs.

Match(**menteeID**: integer, **mentorID**: integer)

- There are no CKs.

Major(faculty: string, majorName: string)

- There are no CKs or FKs.

InterestedIn(**pid**: integer, **faculty**: string, **majorName**: string, **jobTitle**: string)

- There are no CKs.

Attends(**pid**: integer, **eventName**: string, **eventDate**: string)

- There are no CKs.

PotentialCareer (jobTitle: string)

- There are no CKs or FKs.

BasedIn(**postalCode**: string, **mode**: string, **company**: string, **jobTitle**: string, **pid**: integer)

- There are no CKs.

Event(eventName: string, eventDate: string, **sponsorName**: string, **address**: string, **roomNumber**: integer)

- There are no CKs.

Location(postalCode: string, mode: string, country: string, city: string, province: string)

- There are no CKs or FKs.

Industry(industryName: string)

- There are no CKs or FKs.

Room(roomNumber: integer, capacity: integer, address: string, floorNumber: integer)

- There are no CKs or FKs.

Sponsor(sponsorName: string, *repName*: string, *repEmail*: string)

- There are no FKs.

Within(**industryName**: string, **company**: string, **jobTitle**: string, **pid**: integer)

- There are no CKs.

4. Functional Dependencies (FDs)

a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key).

Note: In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalisation exercise.

Legend:
- Primary key
- **Foreign key**
- *Candidate key*

Person(pid, email, *firstName*, *lastName*, year, genderPreference, gender, degree)
Pid → email + firstName + lastName + year + genderPreference + gender + degree

WorkExperience (company, jobTitle, duration, **pid,** salary)
Pid + company → duration
jobTitle + company + pid → salary

Mentor(*major*, pid)
Pid → major

Mentee(pid)
There are no non-trivial functional dependencies for Mentee

Match(menteeID, mentorID)
There are no non-trivial functional dependencies for Match

Major(faculty, majorName)
There are no non-trivial functional dependencies for Major

InterestedIn(**pid**, **faculty**, **majorName**, **jobTitle**)
There are no non-trivial functional dependencies for InterestedIn

Attends(**pid**, **eventName**, **eventDate**)
There are no non-trivial functional dependencies for Attends

PotentialCareer (jobTitle)
There are no non-trivial functional dependencies for PotentialCareer

BasedIn(**postalCode**, **mode**, **company**, **jobTitle**, **pid**)
There are no non-trivial functional dependencies for BasedIn

Event(eventName, eventDate, **sponsorName**, **address**, **roomNumber**) eventName + eventDate → sponsorName + address + roomNumber

Location(<u>postalCode</u>, <u>mode</u>, country, city, province)
Postal code → city + province
City + province → country

Industry(<u>industryName</u>)
There are no non-trivial functional dependencies for Industry.

Room(<u>roomNumber</u>, capacity, <u>address</u>, floorNumber)
roomNumber + address → capacity + floorNumber

Sponsor(<u>sponsorName</u>: string, *repName*: string, *repEmail*: string)
SponsorName → RepEmail + RepName
SponsorName + RepEmail → RepName

Within(**<u>industryName</u>**, **<u>company</u>**, **<u>jobTitle</u>**, **<u>pid</u>**)
There are no non-trivial functional dependencies for Within.

5. Normalization

a. **Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization.**
You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown. The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only the ones post normalisation.

WorkExperience(<u>company</u>, <u>jobTitle</u>, duration, **<u>pid</u>,** salary)
FD:
- jobTitle + company + pid → salary
- Company + pid → duration

Closures:
- {jobTitle, company, pid}+ = {company, jobTitle, salary, pid, duration}
- {company, pid}+ = {company, pid, duration}

Minimal key: {company, jobTitle, pid}+ = {company, jobTitle, pid, duration, salary}

WorkExperience violates 3NF because in the FD: pid + company → duration, pid + company is not a super key, and duration is not part of a key

1. Decompose Company + pid → duration
   R1(<u>pid</u>, <u>company</u>, duration} R2(<u>company</u>, **<u>pid</u>**, <u>jobTitle</u>, salary)

Both of the relations are in BCNF because R1 FD is a superkey for duration and R2"s FD is a superkey for salary and no further decomposition is required for both relations. Therefore, our final decomposition for WorkExperience is:

R1(pid, company, duration) R2(company, **pid**, jobTitle, salary)

Location(Postal Code, province, city, country, mode)
FD:

1. City + province → country
2. Postal code → city + province

Closures:

1. {city,province}+ = {city, province, country}
2. {postalCode}+ = {city, province, postalCode, country}

Minimal key: {postalCode, mode}+ = {city, province, postalCode, country, mode}

Location violates 3NF because in the FD: City + province → country city and province is not a superkey and country is not part of a minimal key. In addition, in FD: Postal Code → city + province, postal code is not a superkey for Location and city + province is not part of a minimal key

1. Decompose Location on "City + province → country "
   R1(city, province, country), R2(city, province, postalCode, mode)

   R1 holds because "city + province → country", but R2 still violates BCNF because postalCode is not a super key for R2.

2. Decompose R2 on "postalCode → city + province"
   R3(postalCode, province, city), R4(postalCode, mode)

   Both of the above relations are good because R3 holds and R4 is a 2-attribute entity. Therefore, our final decomposition of Location is:

R1(city, province, country), R3(postalCode, province, city), R4(postalCode, mode)

Person(pid: integer, email: string, *firstName*: string, *lastName*: string, *year*: integer, genderPreference: string, gender: string, degree: string)

- There are no FKs

WorkExperienceDuration (company: string, **pid**: integer, duration: integer)

- There are no CKs.

WorkPay(company: string, jobTitle: string, **pid**: integer, salary: integer)

- There are no CKs.

Mentor(major: string, **pid**: integer)
- There are no CKs.

Mentee(**pid**: integer)
- There are no CKs.

Match(**menteeID**: integer, **mentorID**: integer)
- There are no CKs.

Major(**faculty**: string, **majorName**: string)
- There are no CKs.

InterestedIn(**pid**: integer, **faculty**: string, **majorName**: string, **jobTitle**: string)
- There are no CKs.

Attends(**pid**: integer, **eventName**: string, **eventDate**: string)
- There are no CKs.

PotentialCareer (jobTitle: string)
- There are no CKs or FKs.

BasedIn(**postalCode**: string, **mode**: string, **company**: string, **jobTitle**: string, **pid**: integer)
- There are no CKs.

Event(eventName: string, eventDate: string, **sponsorName**: string, **address**: string, **roomNumber**: integer)
- There are no CKs.

Country(country: string, city: string, province: string)
- There are no CKs or FKs.

PostalCode(postalCode: string, city: string, province: string)
- There are no CKs or FKs.

Workplace(postalCode: string, mode: string)
- There are no CKs or FKs.

Industry(industryName: string)
- There are no CKs or FKs.

Room(roomNumber: integer, capacity: integer, address: string, floorNumber: integer)
- There are no CKs or FKs.

Sponsor(sponsorName: string, *repName*: string, *repEmail*: string)
- There are no FKs.

Within(<u>**industryName**</u>: string, <u>**company**</u>: string, <u>**jobTitle**</u>: string, <u>**pid**</u>: integer)

- There are no CKs.

6. The SQL DDL statements required to create all the tables from item #5. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc.

```
CREATE TABLE Person(
        pid INTEGER PRIMARY KEY,
        email CHAR(40),
        firstName CHAR(20),
        lastName CHAR(20),
        year INTEGER,
        genderPreference CHAR(10),
        gender CHAR(10),
        degree CHAR(10)
);

CREATE TABLE Mentor(
        pid INTEGER PRIMARY KEY,
        major CHAR(20)
);

CREATE TABLE Mentee(
        pid INTEGER PRIMARY KEY
);

CREATE TABLE WorkExperienceDuration (
        pid INTEGER,
        company CHAR(40),
        duration INTEGER,
        PRIMARY KEY(pid, company),
        FOREIGN KEY (pid) REFERENCES Mentor
                ON UPDATE CASCADE
                ON DELETE CASCADE
);

CREATE TABLE WorkPay (
        pid INTEGER,
        company CHAR(40),
        jobTitle CHAR(40),
        salary INTEGER,
        PRIMARY KEY(pid, company, jobTitle),
        FOREIGN KEY (pid) REFERENCES Mentor
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

```
CREATE TABLE Match(
	menteeID INTEGER NOT NULL,
	mentorID INTEGER NOT NULL,
	PRIMARY KEY(menteeID) REFERENCES Person(pid)
		ON UPDATE CASCADE
		ON DELETE CASCADE,
	PRIMARY KEY(mentorID) REFERENCES Person(pid)
		ON UPDATE CASCADE
		ON DELETE CASCADE,
);

CREATE TABLE InterestedIn(
	pid INTEGER,
	faculty CHAR(40),
	majorName CHAR(40),
	jobTitle  CHAR(40),
	PRIMARY KEY (pid, faculty, majorName),
	FOREIGN KEY (pid) REFERENCES Person,
		ON UPDATE CASCADE
		ON DELETE CASCADE,
	FOREIGN KEY(faculty, majorName) REFERENCES Major(faculty, majorName),
		ON UPDATE CASCADE
		ON DELETE NO ACTION,
	FOREIGN KEY(jobTitle) REFERENCES PotentialCareer(jobTitle)
		ON UPDATE CASCADE
		ON DELETE NO ACTION,
);

CREATE TABLE Major(
	faculty string,
	majorName string,
	PRIMARY KEY(faculty, majorName)
);

CREATE TABLE Attends(
	pid INTEGER,
	eventName CHAR(40),
	eventDate CHAR(30),
	PRIMARY KEY (eventName, eventDate, pid),
	FOREIGN KEY (eventName, eventDate) REFERENCES event
		ON UPDATE CASCADE
		ON DELETE CASCADE,
	FOREIGN KEY (pid) REFERENCES person
		ON UPDATE CASCADE
		ON DELETE CASCADE,
)
```

```
CREATE TABLE Event (
        eventName CHAR(40),
        eventDate CHAR(40),
        sponsorName CHAR(30),
        Address CHAR(40),
        roomNumber INTEGER,
        PRIMARY KEY(eventName, eventDate),
        FOREIGN KEY (sponsorName) REFERENCES Sponsor
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (address, roomNumber) REFERENCES Room
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE PotentialCareer (
        jobTitle CHAR(20) PRIMARY KEY
);

CREATE TABLE BasedIn(
        postalCode CHAR(10),
        mode CHAR(10),
        company CHAR(10),
        jobTitle CHAR(10),
        pid INTEGER,
        PRIMARY KEY (postalCode, mode, company, jobTitle, pid),
        FOREIGN KEY (mode, postalCode) REFERENCES Workplace
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (company, jobTitle, pid) REFERENCES WorkPay
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Country(
        country CHAR(30),
        city CHAR(30),
        province CHAR(30),
        PRIMARY KEY(city, province)
);

CREATE TABLE PostalCode (
        postalCode CHAR(6) PRIMARY KEY,
        city CHAR(30),
        province CHAR(30)
);
```

```
CREATE TABLE Workplace (
        postalCode CHAR(6) PRIMARY KEY,
        mode CHAR(30)
);

CREATE TABLE Industry(
        industryName CHAR(40) PRIMARY KEY
);

CREATE TABLE Room(
       roomNumber INTEGER,
       capacity INTEGER,
       address CHAR(40),
        floorNumber: INTEGER,
        PRIMARY KEY(roomNumber, address)
);

CREATE TABLE Sponsor(
        sponsorName CHAR(30) PRIMARY KEY,
        repName CHAR(30),
        repEmail CHAR(30)
);

CREATE TABLE Within(
        industryName CHAR(40),
        company CHAR(40),
        jobTitle CHAR(40),
        pid INTEGER,
        PRIMARY KEY(industryName, company, jobTitle, pid)
        FOREIGN KEY (industryName) REFERENCES Industry
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (company, jobTitle, pid) REFERENCES WorkPay
                ON DELETE CASCADE
                ON UPDATE CASCADE,
);
```

7. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later on.

```
INSERT INTO Person(pid, email, firstName, lastName, year, genderPreference, gender,
degree)
VALUES (1, "billsmith@student.com","bill", "smith", 1, "male", "male", "BComm"),
        (2, "carlysmith@student.com","carly", "smith", 2, "female", "male", "BA"),
        (3, "sandrabullock@student.com","sandra", "bullock", 3, "female", "female", "BSc"),
        (4, "sarahpaulson@student.com","sarah", "paulson;, 3, "female", "female", "BSc"),
        (5, "carinacostanza@student.com","carina", "costanza", 4, "female", "female", "BSc");

INSERT INTO Mentor(pid, major)
VALUES(1, "BUCS"),
        (2, "Cognitive Sciences"),
        (3, "Computer Science"),
        (4, "Computer Science"),
        (5, "Computer Science");

INSERT INTO Mentee(pid)
VALUES(6),
        (7),
        (8),
        (9),
        (10);

INSERT INTO WorkPay(company, jobTitle, pid, salary)
VALUES(SAP, Business Analyst, 1, 100000),
        (Stripe, Recruiter, 2, 98000),
        (Teck, Data Analyst, 3, 270000),
        (Amazon, Project Manager, 4, 345000),
        (Google, Software Engineer, 5, 400000);

INSERT INTO WorkExperienceDuration(company, pid, duration)
VALUES(SAP, 8, 1),
        (Stripe,10, 2),
        (Teck, 4, 3),
        (Amazon, 16, 4),
        (Google, 8, 5);

INSERT INTO Match(menteeID, mentorID)
VALUES(1,6),
        (2,7),
        (3,8),
        (4,9),
        (5,10);

INSERT INTO InterestedIn(pid, faculty, majorName, jobTitle)
VALUES(5, "Science", "Computer Science", "Software Developer"),
        (6, "Science", "Statistics, "Business Analyst"),
        (6, "Arts", "Computer Science",  "Product Manager"),
        (6, "Science", ""Math", "Professor"),
        (6, "Business", "Computer Science", "Product Manager");

INSERT INTO Major(faculty, majorName)
VALUES("Science", "Computer Science"),
        ("Science", "Statistics"),
        ("Arts", "Computer Science"),
        ("Science", "Math"),
        ("Business", "Computer Science");

INSERT INTO Attends(pid, eventName, eventDate)
VALUES(1, "Coffee Chats", "January 5"),
        (3, "Info Session", "November 15"),
        (4, "Coffee Chats", January 5"),
        (2, "Ice Breakers", "June 3"),
        (5, "Banquet", "May 5");

INSERT INTO Event(eventName, eventDate, sponsorName, address, roomNumber)
VALUES("Coffee Chats", "January 5", "Stripe", "1111 Main Mall", 3),
        ("Ice Breakers", November 15", "Amazon", "75 Agronomy Road", 404),
        ("Banquet, "May 5", "Stripe", "88 Main Mall", 102),
        ("Info Session", "June 30", "Stripe", "73 Main Mall", 101),
        ("Resume Workshop", "July  5", "Stripe", "90 Agronomy Road", 110);

INSERT INTO PotentialCareer(jobTitle)
VALUES("Software Engineer"),
        ("Web Developer"),
        ("Data Scientist),
        ("Data Engineer"),
        ("Data Analyst);

INSERT INTO BasedIn(postalCode, mode, company, jobTitle, pid)
VALUES("V5H6U3", "Virtual", "SAP", "Business Analyst", 1),
        ("V4K8K9", "Virtual", "Teck", "Data Analyst", 2),
        ("V5H6U3", "In-Person", "Google", "Software Engineer", 3),
        ("V8H9I0", "In-Person", "Google", "Project Manager", 4),
        ("B7H8J9", "Virtual", "SAP", "Project Analyst", 5);

INSERT INTO Country(country, city, province)
VALUES("Canada", "British Columbia", "Vancouver"),
        ("Canada", "British Columbia", "Victoria"),
        ("Canada", "Alberta", "Edmonton"),
        ("USA", "Washington", "Seattle"),
        ("USA", "California", "Los Angeles");

INSERT INTO PostalCode(postalCode, city, province)
VALUES("V5H6U3", "Vancouver", "British Columbia"),
        ("V4K8K9", "Vancouver", "British Columbia"),
        ("V8H9I0", "Vancouver", "British Columbia"),
        ("C95H63", "Edmonton", "Alberta"),
        ("B7H8J9", "Victoria", "British Columbia");

INSERT INTO Workplace(postalCode, mode)
VALUES("V5H6U3", "online"),
        ("V4K8K9", "in person"),
        ("V8H9I0", "in person"),
        ("V5H6U3", "in person"),
        ("C95H63", "online");

INSERT INTO Industry(industryName)
VALUES("IT/Technology"),
        ("Data Science"),
        ("HR"),
        ("Engineering"),
        ("UX Design");

INSERT INTO Room(roomNumber, capacity, address, floorNumber)
VALUES(236, 30, "1111 Main Mall", 2),
        (304, 50, "1111 Main Mall", 3),
        (144, 25, "75 Agronomy Road",  1),
        (100, 30, "81 West Mall", 1)
        (423, "1111 Main Mall", 4);

INSERT INTO Sponsor(sponsorName, repName, repEmail)
VALUES("Stripe", "Mary Lamb", "marylamb@stripe.com"),
        ("Amazon", "Jeff Bezos", "jeff@amazon.ca"),
        ("Apple", "Violet James", "violet@apple.com"),
        ("Splunk", "Boris Kovacevic", "boris@splunk.ca"),
        ("Unity", "Lola Evans", "lola@unity.com");

INSERT INTO Within(industryName, company, jobTitle, pid)
VALUES("IT/Technology", "SAP", "Business Analyst", 1),
        ("Data Science", "Teck", "Data Analyst", 2),
        ("IT/Technology", "Amazon", "Project Manager", 3),
        ("Engineering", "Google", "Software Engineer", 4),
        ("HR", "Stripe", "Recruiter", 5);
```