University of British Columbia, Department of Computer Science

# CPSC 304

## Cover Page for Project Part 1

## Date: September 26, 2018

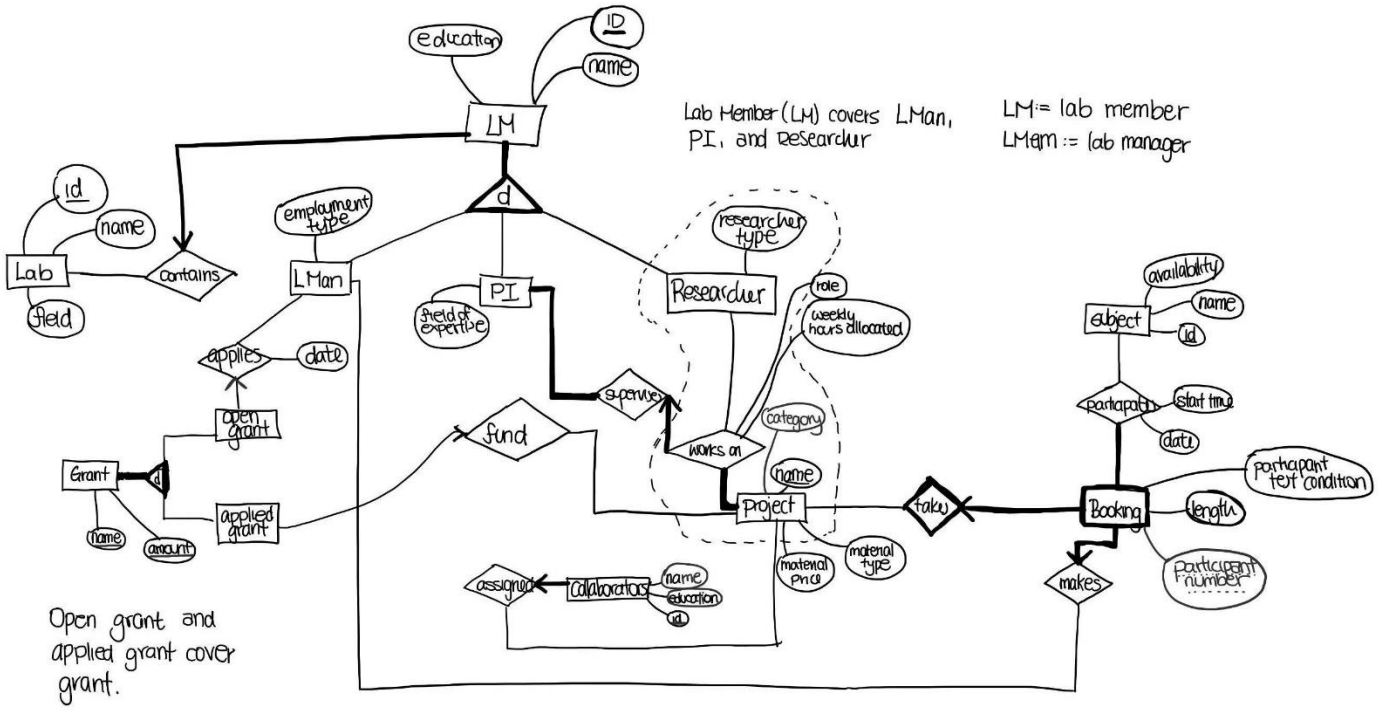
## Group Members:

| Name | Student Number | CS Userid | Tutorial Section | Email Address |
|---|---|---|---|---|
| Damasia Maria Correch | 28649151 | o2e1b | T1C | damacorrech@gmail.com |
| Naitong Chen | 21539151 | q7l0b | T1G | chennaitongubc@hotmail.com |
| Chloe You | 11654150 | i0e1b | T1D | chloeyxy@outlook.com |
| Christina Cheung | 37405157 | d8f1b | T1D | ccheung256@gmail.com |

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above.

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

**E/R Diagram**


education
ID
name
LM
Lab Member (LM) covers LMan, PI, and Researcher
LM:= lab member
LMgm := lab manager
id
name
Lab
contains
field
employment type
LMan
d
PI
field of expertise
Researcher
researcher type
role
weekly hours allocated
applies
date
open grant
Grant
d
name
amount
applied grant
fund
supervises
works on
category
name
Project
material price
material type
assigned
Collaborators
name
education
id
take
availability
name
id
subject
participates
start time
date
Booking
length
participant text condition
participant number
makes
Open grant and applied grant cover grant.

**Database Schema Description**

Lab (id: string, name: string, field: string)
Primary key: id
Candidate key: id
Foreign key: none
Functional dependencies: id -> name, field

Contains_LabMember (id: string, **labId**: string, name: string, education: string)
Primary key: id
Candidate key: id
Foreign key: labId references Lab (id)
Constraints: labId is not null
Functional dependencies: id -> labId, name, education

LabManager (**id**: string, empolymentType: string)
Primary key: id
Candidate key: id
Foreign key: id references LabMember (id)
Constraints: none
Functional dependencies: id -> empolymentType

PI (**id**: string, fieldOfExpertise: string)
Primary key: id
Candidate key: id
Foreign key: id references LabMember (id)
Constraints: none
Functional dependencies: id -> fieldOfExpertise

Researcher (**id**: string, researcherType: string)
Primary key: id
Candidate key: id
Foreign key: id references LabMember (id)
Constraints: none
Functional dependencies: id -> researcherType

Project (name: string, category: string, materialType: string, materialPrice: float)
Primary key: name
Candidate key: name
Foreign key: none
Constraints: none
Functional dependencies: name -> category, materialType
materialType -> materialPrice (non-obvious)

We impose that each project in the lab uses a unique material, so the type of material (materialType) used can be determined by the projectName - materialType is something like “imaging”, “DNA”, “video”, “survey”.
The price of the materials is then uniquely determined by the type of materials. Every materialType has a different price.

Applies_OpenGrant (name: string, amount: float, **lmid**: string**,** date: date)

Primary key: name, amount
Candidate key: name, amount
Foreign key: lmid references LabManager (id)
Constraints: none
Functional dependencies: name, amount -> date
date, name -> lmid (non-obvious)

Given the date applied to the grant and the name of the grant, we can uniquely identify the lab manager who applied to the grant, as we impose/assume that a grant with a given name can only be applied to once on a given day. For example, a LabManager can apply to one NSERC grant a day - even though there are two grants: NSERC 20k and NSERC 40k. The LabManager can only apply to one of these each day. They can apply to NSERC 20k and AllerGen 40k, however, on the same day because they have different names. Therefore, date and name datamine lmid, which is the primary key for LabManager.

Fund_AppliedGrant (name: string, amount: float, **projectName**: string)

Primary key: name, amount
Candidate key: name, amount
Foreign key: ProjectName references Project (name)
Constraints: none
Functional dependencies: name, amount -> projectName

Supervises_WorksOn (**projectName**: string, **rid**: string, **piid**: string, role: string, weeklyHoursAllocated: integer)

Primary key: projectName, rid
Candidate key: projectName, rid
Foreign key: projectName references Project (name)
rid references Researcher (id)
piid references PI (id)
Constraints: piid is not null
Functional dependencies: projectName, rid -> piid, role
role -> weeklyHoursAllocated

Researchers may work on various projects at once. As such, they need to allocate a certain number of hours per week to work on each of their project - this hourly allocation is determined by the role the researcher is taking on for that project. A researcher with role “lead” will allocate more hours to that project than a researcher with the role “assistant”.

Assigned_Collaborators (id: string, **projectName**: string, name: string, education: string)

Primary key: id
Candidate key: id
Foreign key: projectName references Project (name)
Constraint: projectName is not null
Functional dependencies: id -> name, education
name, education -> projectName (non-obvious)

We impose that there are no two collaborators with the same name and level of education, and that each collaborator can only work on one project. Therefore, having a name and level of education uniquely identifies the project name.

Subjects (id: string, name: string, availability: boolean)

Primary key: id
Candidate key: id
Foreign key: none
Constraints: none
Functional dependencies: id -> name, availability

Takes_Booking (**projectName**: string, participantNumber: integer, length: integer, participantTestCondition: string)

Primary key: projectName, participantNumber
Candidate key: projectName, participantNumber
Foreign key: projectName references Project (name)
Constraints: projectName is not null
Functional dependencies: projectName, participantNumber -> length, participantTestCondition

Each project has their own enumeration of participants, and participants can participate in many projects with different participantNumbers - for example, a subject with SID 10A can participate in Project A with participantNumber 4, because they were the 4th participant in that project, while also participating in Project B with participant number 2, because they were the 2nd participant in that project. Therefore, the participantNumber is unique within a project, but different subjects could have the same participantNumber in different projects. Therefore, both the participantNumber and projectName are needed to identify the weak entity Booking.
Length refers to how much time the booking is for. Since we can uniquely identify a participant using projectName and participantNumber, we can use these two pieces of information to determine how long the participant is booked for this particular project.
Finally, participants need to be randomly assigned the test condition they will participate in. We assume each project determines the conditions according to participantNumber before any participants are booked, so a participantNumber will always yield the condition that the participant in that booking will get.

Participates (**sid**: string, **projectName**: string, **participantNumber**: integer, startTime: integer, date: date)

Primary key: sid, projectName, participantNumber
Candidate key: sid, projectName, participantNumber
Foreign key: sid references Subject (id)
projectName, participantNumber reference takes_Booking (projectName, partipantNumber)
Constraints: none
Functional dependencies: participantNumber, projectName, sid-> startTime, date

Makes_Booking (**projectName**: string, **participantNumber**: integer, **lmid**: string)

Primary key: projectName, participantNumber
Candidate key: projectName, participantNumber
Foreign key: projectName, participantNumber reference takes_Booking (projectName, partipantNumber)
lmid references LabManager (id)
Constraints: lmid is not null
Functional dependencies: projectName, participantNumber -> lmid

**Normalized Relations**

Lab (id: string, name: string, field: string)
Primary key: id
Foreign key: none

Contains_LabMember (id: string, **labId**: string, name: string, education: string)
Primary key: id
Foreign key: labId references Lab (id)

LabManager (**id**: string, empolymentType: string)
Primary key: id
Foreign key: id references LabMember (id)

PI (**id**: string, fieldOfExpertise: string)
Primary key: id
Foreign key: id references LabMember (id)

Researcher (**id**: string, researcherType: string)
Primary key: id
Foreign key: id references LabMember (id)

Project_MaterialType (name: string, category: string, materialType: string)
Primary key: name
Foreign key: none

MaterialType_MaterialPrice (materialType: string, materialPrice: float)
Primary key: materialType
Foreign key: none

Applies_OpenGrant_Date (name: string, amount: float**,** date: date)
Primary key: name, amount
Foreign key: none

Date_Name_Lmid (name: string, date: date, **lmid**: string)
Primary key: name, date
Foreign key: lmid references LabManager (id)

Fund_AppliedGrant (name: string, amount: float, **projectName**: string)
Primary key: name, amount
Foreign key: ProjectName references Project (name)

Supervises_WorksOn (**projectName**: string, **rid**: string, **piid**: string, role: string)
Primary key: projectName, rid
Foreign key: projectName references Project (name)
rid references Researcher (id)
piid references PI (id)

Role_WeeklyHoursAllocated (role: string, weeklyHoursAllocated: integer)
Primary key: role
Foreign key: none

Assigned_Collaborators_Id (id: string, name: string, education: string)
Primary key: id
Foreign key: none

Name_Education_ProjectName (name: string, education: string, **projectName**: string)
Primary key: name, education
Foreign key: projectName references Project (name)

Subjects (id: string, name: string, availability: boolean)
Primary key: id
Foreign key: none

Takes_Booking (**projectName**: string, participantNumber: integer, length: integer, participantTestCondition: string)
Primary key: projectName, participantNumber
Foreign key: projectName references Project (name)

Participates (**sid**: string, **projectName**: string, **participantNumber**: integer, startTime: integer, date: date)
Primary key: sid, projectName, participantNumber
Foreign key: sid references Subject (id)
projectName, participantNumber reference takes_Booking (projectName, partipantNumber)

Makes_Booking (**projectName**: string, **participantNumber**: integer, **lmid**: string)
Primary key: projectName, participantNumber
Foreign key: projectName, participantNumber reference takes_Booking (projectName, partipantNumber)
lmid references LabManager (id)

**SQL DDL- Create Tables**

```
CREATE TABLE Lab
        (id CHAR(4) PRIMARY KEY,
        name CHAR(20),
        Field CHAR(4));

CREATE TABLE Contains_LabMember
        (id CHAR(4) PRIMARY KEY,
        labId CHAR(4) NOT NULL,
        name CHAR(20),
        Education CHAR(4),
        FOREIGN KEY (labId) REFERENCES Lab(id)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);

CREATE TABLE LabManager
        (id CHAR(4) PRIMARY KEY,
        employmentType CHAR(20),
        FOREIGN KEY (id) REFERENCES Contains_LabMember(id)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);

CREATE TABLE PI
        (id CHAR(4) PRIMARY KEY,
        fieldOfExpertise CHAR(20),
        FOREIGN KEY (id) REFERENCES Contains_LabMember(id)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);

CREATE TABLE Researcher
        (id CHAR(4) PRIMARY KEY,
        researcherType CHAR(20),
        FOREIGN KEY (id) REFERENCES Contains_LabMember(id)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);

CREATE TABLE Project_MaterialType
        (name CHAR(20) PRIMARY KEY,
        category CHAR(20),
        materialType CHAR(20));

CREATE TABLE MaterialType_MaterialPrice
        (materialType CHAR(20) PRIMARY KEY,
        materialPrice FLOAT);

CREATE TABLE Applies_OpenGrant_Date
        (name CHAR(20),
        amount FLOAT,
        date DATE,
        CHECK (amount > 0),
        PRIMARY KEY (name, amount));
```

```
CREATE TABLE Date_Name_Lmid
        (name CHAR(20),
        date DATE,
        lmid CHAR(4),
        PRIMARY KEY (name, date),
        FOREIGN KEY (lmid) REFERENCES LabManager(id)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);

CREATE TABLE Fund_AppliedGrant
        (name CHAR(20),
        amount FLOAT,
        projectName CHAR(20),
        CHECK (amount > 0),
        PRIMARY KEY (name, amount),
        FOREIGN KEY (projectName) REFERENCES Project(name)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);

CREATE TABLE Supervises_WorksOn
        (projectName CHAR(20),
        rid CHAR(4),
        piid CHAR(4) NOT NULL,
        role CHAR(20),
        PRIMARY KEY (projectName, rid),
        FOREIGN KEY (projectName) REFERENCES Project(name)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY (rid) REFERENCES Researcher(id)
                ON DELETE  NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY (piid) REFERENCES PI(id)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);

CREATE TABLE Role_WeeklyHoursAllocated
        (role CHAR(20) PRIMARY KEY,
        weeklyHoursAllocated INTEGER);

CREATE TABLE Assigned_Collaborators_Id
        (id CHAR(4) PRIMARY KEY,
        name CHAR(20),
        education CHAR(4));

CREATE TABLE Name_Education_ProjectName
        (name CHAR(20),
        education CHAR(4),
        projectName CHAR(20) NOT NULL,
        PRIMARY KEY (name, education),
        FOREIGN KEY (projectName) REFERENCES Project(name)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);
```

```
CREATE TABLE Subject(
        id CHAR(4) PRIMARY KEY,
        name CHAR(20),
        availability BOOLEAN);

CREATE TABLE Takes_Booking(
        projectName CHAR(20) NOT NULL,
        participantNumber INTEGER,
        length INTEGER,
        participantTestCondition CHAR(10),
        PRIMARY KEY (projectName, participantNumber),
        FOREIGN KEY (projectName) REFERENCES Project(name)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);

CREATE TABLE Participates
        (sid CHAR(4),
        projectName CHAR(20),
        participantNumber INTEGER,
        date DATE,
        PRIMARY KEY (sid, projectName, participantNumber),
        FOREIGN KEY (sid) REFERENCES Subject(id)
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY (projectName, participantNumber) REFERENCES Takes_Booking(projectName,
        participantNumber)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE);

CREATE TABLE Makes_Booking(
        projectName CHAR(20),
        participantNumber INTEGER,
        lmid CHAR(4) NOT NULL,
        PRIMARY KEY (projectName, participantNumber),
        FOREIGN KEY (projectName, participantNumber) REFERENCES Takes_Booking(projectName,
        participantNumber)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE,
        FOREIGN KEY (lmid) REFERENCES LabManager(id)
                ON DELETE NO ACTION
                ON UPDATE CASCADE);
```