University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 21/10/2022

Group Number: 93

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Timothy Yap	43665868	n8g3b	timothyyap1106@gmail.com
Ezekiel Eteve	65462202	a9d7m	zeke.eteveuni@gmail.com
Vihangi Perera	28944866	y2y8m	vihangiperera9@gmail.com

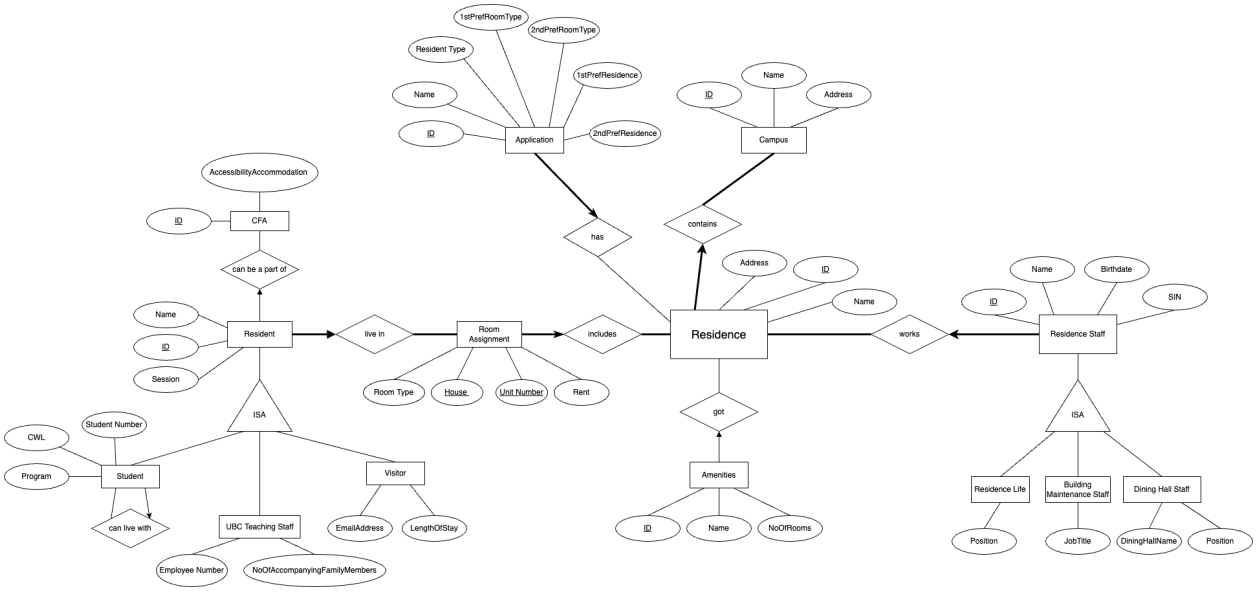
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

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

Milestone 2

2. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why.

We changed the total participation relationship between Application and Residence to a partial participation relationship.
Relationship diamonds were unbolded because our ER diagram does not have weak entities.
Changed redundant ‘has’ relationship labels to other relevant labels.



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.

Primary Keys -> Underlined
Foreign Keys -> Bold
Candidate Keys -> Italics

-RoomAssignment(Room Type, House, Unit Number, Rent)
-Campus(ID, Name, Address)

-ResidenceStaff(ID, Name, Birthdate, *SIN,*
ResidenceLifePosition,BuildingMaintenanceStaffJobTitle, DiningHallName, DiningHallPosition)

-Residence(ID, Address, Name)
-Amenities(ID, Name, NoOfRooms)
-Application(ID, Name, Resident Type, 1stPrefRoomType, 2ndPrefRoomType,
1stPrefResidence, 2ndPrefResidence)
-CFA(ID, Accessibility Accommodation)

-Resident(ID, Name, Session, CWL, Student Number, Program, Email Address, Length of Stay,
Employee Number, NoOfAccompanysFamilyMembers)

- Contains(**CampusID, ResidenceID**)
- Has(**ResidenceID, ApplicationID**)
- Includes(**ResidenceID, House, Unit Number)**
- LiveIn(**House, Unit Number, ResidentID**)
- CanbeAPartOf(**ResidentID, CFAID**)
- CanLiveWith(**ResidentID,OtherResidentID**)
- Works(**ResidenceStaffID, ResidenceID**)
- Got(**AmenityID, ResidenceID)**

4.Functional Dependencies (FDs)

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

CampusID -> {Name, Address}

ResidenceStaffID -> {Name,Birthdate, Sin, ResidenceLifePosition}
ResidenceStaffID -> {Name,Birthdate, Sin, JobTitle}
ResidenceStaffID -> {Name,Birthdate, Sin, DiningHallName, DiningHallPosition}

ApplicationID -> {Name, Resident Type, 1stPrefRoomType, 2ndPrefRoomType,
1stPrefResidence, 2ndPrefResidence}
ResidenceID -> {Address, Name}

{House, Unit Number} -> Room Type, Rent
{Room Type, House} -> Rent

House -> Residence ID

AmenityID -> {Name, NoOfRooms}
CFAID -> {Accessibility Accommodation}

- ResidentID -> {Name, Session, StudentNumber, CWL, Program}
- ResidentID -> {Name, Session, EmployeeNumber, NoOFAccompanyingFamilyMembers}
- ResidentID -> {Name, Session, EmailAdress, LengthOfStay}

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 normalization.

- **R** : RoomAssignment(Room Type, House, Unit Number, Rent)
  FD's :
  {House, Unit Number} -> Room Type, Rent
  {Room Type, House} -> Rent
  Closures:
  {House, Unit Number}+ = {House, Unit Number, Room Type, Rent}
  {Room Type, House}+ = {Room Type, House, Rent}

  Decomposition: R(House, UnitNumber, RoomType, Rent)
  Decompose on {Room Type, House} -> Rent
  R1(House, Room Type, Rent), R2(House, Unit Number, Room type)

  Final Answer: **Rent**(House, Rent, Room Type), **RoomType**(House, Unit Number, Room Type)

- **R** : LiveIn(**House, Unit Number, ResidentID**)
  FD's :
  {House, Unit Number} -> Room Type
  {House, Unit Number} -> Rent
  {Room Type, House} -> Rent

ResidentID -> {Name, Session, StudentNumber, CWL, Program}
ResidentID -> {Name, Session, EmployeeNumber, NoOFAccompanyingFamilyMembers}
ResidentID -> {Name, Session, EmailAddress, LengthOfStay}

Closures:

{House, Unit Number}+ = {House, Unit Number, Room Type, Rent}
{Room Type, House}+ = {Room Type, House, Rent}
{ResidentID}+ = {ResidentID, Name, Session, StudentNumber, CWL, Program, EmployeeNumber, NoOFAccompanyingFamilyMembers, EmailAddress, LengthOfStay}

Decomposition: R(House, UnitNumber, RoomType, Rent, ResidentID, Name, Session, StudentNumber, CWL, Program, EmployeeNumber, NoOFAccompanyingFamilyMembers, EmailAddress, LengthOfStay)

Decompose on {House, Unit Number} -> Room Type, Rent
R1(House, Unit Number, Room Type, Rent), R2(House, Unit Number,ResidentID, Name, Session, StudentNumber, CWL, Program, EmployeeNumber, NoOFAccompanyingFamilyMembers, EmailAddress, LengthOfStay)

Final Answer: **RoomInfo**(House, Unit Number, Room Type, Rent), **RoomAllocationInfo**(House, Unit Number,ResidentID, Name, Session, StudentNumber, CWL, Program, EmployeeNumber, NoOFAccompanyingFamilyMembers, EmailAddress, LengthOfStay)

Relationship: Includes(**ResidenceID, House, Unit Number)**

FD's: ResidenceID -> {Address, Name}
{House, Unit Number} -> {Room Type, Rent}
{Room Type, House} -> Rent
House -> Residence ID

FD Closures:

{ResidenceID}+ = {ResidenceID, Address, Name}
{House, Unit Number}+ = {House, Unit Number, Room Type, Rent}
{Room Type, House}+ = {Room Type, House, Rent}

R(ResidenceID, RAddress, RName, House, Unit Number, Room Type, Rent**)**

Decompose R

R1(ResidenceID**,** RAddress**,** RName**), R2(**ResidenceID**,** House**,** Unit Number**,** Room Type**,** Rent**)**

Decompose R2 on {House, Unit Number} -> {Room Type, Rent}

R3(House**,** Unit Number**,** Room Type**,** Rent**), R4(**House**,** Unit Number**,** ResidenceID**)**

FINAL ANSWER:
ResidenceInfo(ResidenceID, Residence Address, Residence Name**), HouseRentInfo(**House, Unit Number, Room Type, Rent**), HouseResidenceInfo(**House, Unit Number, ResidenceID**)**

Everything else is in BCNF.

The Residence table and ResidenceInfo tables are the same, so we will only keep Residence.
The RoomInfo table and HouseRentInfo tables are the same, so we will only keep RoomInfo.

After normalization,our final tables will be:

1. Campus(ID, Name, Address)
2. ResidenceStaff(ID, Name, Birthdate, *SIN,* ResidenceLifePosition,BuildingMaintenanceStaffJobTitle, DiningHallName, DiningHallPosition)
3. Residence(ID, Address, Name)
4. Amenities(ID, Name, NoOfRooms)
5. Application(ID, Name, Resident Type, 1stPrefRoomType, 2ndPrefRoomType, 1stPrefResidence, 2ndPrefResidence)
6. CFA(ID, Accessibility Accommodation)
7. Resident(ID, Name, Session, CWL, Student Number, Program, Email Address, Length of Stay, Employee Number, NoOfAccompanysFamilyMembers)
8. Contains(**CampusID, ResidenceID**)
9. Has(**ResidenceID, ApplicationID**)
10. CanBeAPartOf(**ResidentID, CFAID**)
11. CanLiveWith(**ResidentID,OtherResidentID**)
12. Works(**ResidenceStaffID, ResidenceID**)
13. Got(**AmenityID, ResidenceID)**
14. RoomInfo(**House, Unit Number**, **Room Type, Rent**)
15. RoomType(**House, Unit Number, Room Type**)
16. Rent(**House**, **Rent, Room Type**)
17. RoomAllocationInfo(**House, Unit Number,ResidentID**, **Name, Session, StudentNumber, CWL, Program, EmployeeNumber, NoOFAccompanyingFamilyMembers, EmailAddress, LengthOfStay**)
18. HouseResidenceInfo**(House, Unit Number, ResidenceID)**

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. If there are any constraints that you cannot enforce because we have not yet learned about assertions, state that you need assertions.

1. CREATE TABLE Campus (
    CampusID int PRIMARY KEY,
    Name varchar(100) NOT NULL,
    Address varchar(150) UNIQUE,
);

2. CREATE TABLE ResidenceStaff (
    ID int PRIMARY KEY,
    Name varchar(100) NOT NULL,
    Birthdate date,
    SIN int UNIQUE,
    Residence Life Position varchar(50),
    Building Maintenance Title varchar(50),
    Dining Hall Name varchar(50),
    Dining Hall Position varchar(50),
);

3. CREATE TABLE Residence (
    ID int PRIMARY KEY,
    Address varchar(150) UNIQUE,
    Name varchar(75) NOT NULL
);

4. CREATE TABLE Amenities (
    ID int PRIMARY KEY,
    Name varchar(75) NOT NULL,
    NoOfRooms int NOT NULL
);

5. CREATE TABLE Application (
    ID int PRIMARY KEY,
    Name varchar(75) NOT NULL,
    Resident Type varchar(30) NOT NULL,
    1stPrefRoom varchar(50) NOT NULL,

```
        2ndPrefRoom          varchar(50)    NOT NULL,
        1stPrefResidence     varchar(75)    NOT NULL,
        2ndPrefResidence     varchar(75)    NOT NULL
);
```

6.
```
CREATE TABLE CFA (
        ID                                  int             PRIMARY KEY,
        Accessibility Accommodation         varchar(100)    NOT NULL
);
```

7.
```
CREATE TABLE Resident (
        ID                                  int             PRIMARY KEY,
        Name                                varchar(100)    NOT NULL,
        Session                             varchar(75)     NOT NULL,
        CWL                                 varchar(10)     UNIQUE,
        Student Number                      int             UNIQUE,
        Program                             varchar(75)
        Email Address                       varchar(50)     UNIQUE,
        Length of Stay                      varchar(50)
        Employee Number                     int,
        NoOfAccompanysFamilyMembers         int

);
```

8.
```
CREATE TABLE Contains (
        CampusID                            int,
        ResidenceID                         int,
        PRIMARY KEY(CampusID, ResidenceID),
        FOREIGN KEY(CampusID) REFERENCES Campus ON DELETE CASCADE,
        FOREIGN KEY(ResidenceID) REFERENCES Residence (ID) ON DELETE
    CASCADE,
    );
```

9.
```
CREATE TABLE Has (
        ResidenceID                         int,
        ApplicationID                       int,
        PRIMARY KEY(ResidenceID, ApplicationID),
        FOREIGN KEY(ResidenceID) REFERENCES Residence(ID) ON DELETE
    CASCADE,
        FOREIGN KEY(ApplicationID) REFERENCES Application(ID) ON DELETE
    CASCADE,
    );
```

10.
```
CREATE TABLE CanBeAPartOf (
```

```
        ResidenceID                 int,
        CFAID                       int,
        PRIMARY KEY(ResidenceID, CFAID),
        FOREIGN KEY(ResidenceID) REFERENCES Residence(ID) ON DELETE
    CASCADE,
        FOREIGN KEY(CFAID) REFERENCES CFA(ID) ON DELETE CASCADE,
    );
```

11.
```
    CREATE TABLE CanLiveWith  (
        ResidenceID                 int,
        OtherResidenceID            int,
        PRIMARY KEY(ResidenceID, OtherResidenceID),
        FOREIGN KEY(ResidenceID) REFERENCES Residence(ID) ON DELETE
    CASCADE,
        FOREIGN KEY(OtherResidenceID) REFERENCES Residence(ID) ON DELETE
    CASCADE,
    );
```

12.
```
    CREATE TABLE Works (
        ResidenceStaffID            int,
        ResidenceID                 int,
        PRIMARY KEY(ResidenceStaffID, ResidenceID),
        FOREIGN KEY(ResidenceStaffID) REFERENCES ResidenceStaff(ID) ON
    DELETE CASCADE,
        FOREIGN KEY(ResidenceID) REFERENCES Residence(ID) ON DELETE
    CASCADE,
    );
```

13.
```
    CREATE TABLE Got (
        AmenityID                   int,
        ResidenceID                 int,
        PRIMARY KEY(AmenityID, ResidenceID),
        FOREIGN KEY(AmenityID) REFERENCES Amenity(ID) ON DELETE
    CASCADE,
        FOREIGN KEY(ResidenceID) REFERENCES Residence(ID) ON DELETE
    CASCADE,
    );
```

14. CREATE TABLE RoomInfo (
    House varchar(20) PRIMARY KEY,
    Unit Number int PRIMARY KEY,
    Room Type varchar(20) NOT NULL,
    Rent int NOT NULL,
    );
15. CREATE TABLE RoomType (
    House varchar(20) PRIMARY KEY,
    Unit Number int PRIMARY KEY,
    Room Type varchar(20) NOT NULL,
    FOREIGN KEY(House, Unit Number, Room Type) REFERENCES RoomInfo ON DELETE CASCADE,
    );
16. CREATE TABLE Rent (
    House varchar(20) PRIMARY KEY,
    Room Type varchar(20) NOT NULL,
    Rent int NOT NULL,
    FOREIGN KEY(House, Room Type, Rent) REFERENCES RoomInfo ON DELETE CASCADE,
    );

17. CREATE TABLE RoomAllocationInfo (
    House varchar(20) PRIMARY KEY,
    Unit Number int PRIMARY KEY,
    ResidentID int PRIMARY KEY,
    Name varchar(100) NOT NULL,
    Session varchar(75) NOT NULL,
    CWL varchar(10) UNIQUE,
    Student Number int UNIQUE,
    Program varchar(75),
    Email Address varchar(50) UNIQUE,
    Length of Stay varchar(50),
    Employee Number int, UNIQUE
    NoOfAccompanysFamilyMembers int,
    FOREIGN KEY(House, Unit Number) REFERENCES RoomInfo ON DELETE CASCADE,
    FOREIGN KEY (ResidentID, Name, Session, StudentNumber, CWL, Program, EmployeeNumber, NoOFAccompanyingFamilyMembers, EmailAddress, LengthOfStay) REFERENCES Resident ON DELETE CASCADE
    );

18. CREATE TABLE HouseResidenceInfo (
        House varchar(20) PRIMARY KEY,
        Unit Number int PRIMARY KEY,
        ResidentID int PRIMARY KEY,
        FOREIGN KEY(House, Unit Number) REFERENCES RoomInfo ON DELETE CASCADE,
        FOREIGN KEY (ResidentID) REFERENCES Resident ON DELETE 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.

Here is a link to an excel sheet with our tables and data.

https://docs.google.com/spreadsheets/d/11THn7SlCvVR1n7VNMObMJb4OKJibpmnUYCVHTSLdsMY/edit?usp=sharing