University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: _____2__

Date: _____18th July 2024_____

Group Number: ___23_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Syed Araf Imam	64100027	h9y4j@ugrad.cs.ubc.ca	syedarafimam27@gmail.com
Sandy Chu	71623268	m3s8d@ugrad.cs.ubc.ca	tianrui3368@gmail.com
Eric Fan	23904865	q3e6e@ugrad.cs.ubc.ca	ericfan1110@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

2. Brief summary of project:

Our project is a comprehensive database system for Public Facility Management and Maintenance. It encompasses user management, maintenance management, budget management, and event scheduling. The system categorizes users into different roles, such as Managers, Event Organizers, and Staff Members, each with specific permissions. Managers can add budgets, add electronic equipment to rooms, and perform budget allocation for different time periods. Event Organizers can book rooms based on attendee numbers and required devices, ensuring efficient resource utilization. Staff Members can manage maintenance requests, assign servicing contracts (also managers), and keep track of device maintenance schedules. This database system aims to streamline facility management processes, optimize resource allocation, and improve overall operational efficiency.

3. ER diagram

One change was made since milestone 1 where we removed total participation constraint between electronic equipment entity and “installed in” relation as feedback from mentor, where it considers possibility of some equipment not currently installed.

We also made the change from Many to 1 & total participation between user create maintenance request to many to 1, without total participation, because some times, a user is temporary, but should still be allowed to create maintenance request.

We made the change that only staff member can create service contract, because we feel like staff member need its unique job to justify itself.

We also added some attributes in many of the entities to have some non-PK/CK Functional dependencies.

4. Schemas from ER diagram

User(
userID: INTEGER [Candidate Key],
firstName: CHAR(20) NOT NULL,
lastName: CHAR(20) NOT NULL,
SIN: INTEGER NOT NULL UNIQUE, [Candidate Key],
address: CHAR(20) NOT NULL,
province: CHAR(20) NOT NULL,
city: CHAR(20) NOT NULL,
postalCode: CHAR(20) NOT NULL,
PRIMARY KEY (userID),
)

StaffMember(
role: CHAR(20) NOT NULL,
userID: INTEGER [Candidate Key],
PRIMARY KEY (userID),
FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE CASCADE
)

Manager(
workExperience: INTEGER NOT NULL,
userID: INTEGER [Candidate Key],
PRIMARY KEY (userID),
FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE CASCADE
)

EventOrganizer(
organizerLevel: INTEGER NOT NULL,
userID: INTEGER [Candidate Key],
PRIMARY KEY (userID),
FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE CASCADE
)

ServiceContracts(
serviceID: INTEGER [Candidate Key],
staffMemberID: CHAR(20),
companyName: CHAR(50) NOT NULL,
companyRating: INTEGER NOT NULL,

---

cost: DECIMAL(10, 2),
PRIMARY KEY (serviceID)
FOREIGN KEY (staffMemberID) REFERENCES StaffMember(userID)
ON DELETE SET NULL
)

MaintenanceRequests(
requestID: INTEGER [Candidate Key],
userID: CHAR(20),
serviceID: INTEGER,
description: CHAR(20),
severity: INTEGER Default 1,
status: CHAR(20),
date: DATE,
PRIMARY KEY (requestID)
FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE SET NULL
FOREIGN KEY (serviceID) REFERENCES ServiceContracts(serviceID)
ON DELETE SET NULL
)

Budget(
budgetID: INTEGER [Candidate Key],
amount: DECIMAL(10,2),
amountLimit: DECIMAL(10,2),
startDate: Date,
endDate: Date,
managerID: INTEGER,
projectType: CHAR(20),
PRIMARY KEY (budgetID),
FOREIGN KEY (managerID) REFERENCES Manager(userID)
ON DELETE SET NULL
)

Buildings(
buildingName: CHAR(20) [Candidate Key],
Address: CHAR(20) NOT NULL,
City: CHAR(20) NOT NULL,
Province: CHAR(20) NOT NULL,
PostalCode: Char(20) NOT NULL,
PRIMARY KEY (buildingName)
)

RoomsInBuildings(

---

roomNumber: INTEGER,
capacity: INTEGER NOT NULL,
floorNumber: INTEGER NOT NULL,
roomType: CHAR(20) NOT NULL,
buildingName: CHAR(20),
PRIMARY KEY(roomNumber, buildingName),
FOREIGN KEY (buildingName) REFERENCES Buildings(buildingName)
ON DELETE CASCADE
) // [Candidate Key] -> roomNumber + buildinName

MaintainenceRequestsForRoomsInBuildings(
maintenanceRequestID: INTEGER,
roomNumber: INTEGER,
buildingName: CHAR(20),
PRIMARY KEY (maintenanceRequestID, roomNumber, buildingName)
FOREIGN KEY (maintainenenceRequestID) REFERENECES (requestID) MaintenanceRequest(requestID)
ON DELETE CASCADE
FOREIGN KEY (roomNumber, buildingName) REFERENCES RoomsInBuildings (roomNumber, buildingName)
ON DELETE CASCADE
) // [Candidate Key] -> roomNumber + buildingName + maintenanceRequestID

ElectronicEquipment(
deviceID: INTEGER [Candidate Key],
expiryDate: Date NOT NULL,
equipmentType: CHAR(200 NOT NULL,
energyRate: DECIMAL(10,2) NOT NULL,
installationDate: Date NOT NULL,
lastDateOfMaintainenece: Date,
cost: DECIMAL(10,2) NOT NULL,
managerID: INTEGER,
roomNumber: INTEGER,
buildingName: CHAR(20),
manufacturer: CHAR(20) NOT NULL,
warranty: INTEGER NOT NULL,
PRIMARY KEY (deviceID),
FOREIGN KEY (managerID) REFERENCES Manager(userID)
ON DELETE SET NULL,
FOREIGN KEY (roomNumber, buildingName) References RoomsInBuildings(roomNumber, buildingName)
ON DELETE SET NULL;
)

Events(
eventID: INTEGER [Candidate Key],
numberOfPeople: INTEGER NOT NULL Default 0,
startTime: Date NOT NULL,
endTime: Date NOT NULL,
eventType: CHAR(20),
eventCost: DECIMAL(10,2) NOT NULL,
PRIMARY KEY(eventID),
)

EventsOrganizedBy(
eventID: INTEGER,
eventOrganizerID: INTEGER,
PRIMARY KEY(eventID, eventOrganizerID)
FOREIGN KEY(eventID) REFERENCES Events(eventID)
ON DELETE CASCADE
FOREIGN KEY(eventOragnizerID) references EventOrganizer(userID)
ON DELETE CASCADE
) // [Candidate Key] -> eventID + eventOrganizerID

EventsHeldAt(
eventID: INTEGER,
roomNumber: INTEGER,
buildingName: char(20),
purpose: CHAR(20),
PRIMARY KEY(eventID, roomNumber, buildingName)
FOREIGN KEY(eventID) references Events(eventID)
ON DELETE CASCADE
FOREIGN KEY(roomNumber, buildingName) references Rooms(buildingName, roomNumber)
ON DELETE CASCADE
) // [Candidate Key] -> eventID + roomNumber + buildingName

5. Functional dependencies

User Entity:
userID → { firstName, lastName, phoneNumber, SIN, address, province, city, postalCode }
SIN → { firstName, lastName, userID, phoneNumber, address, province, city, postalCode }
postalCode → {City, Province}

Staff Member Entity:

userID → { role }

Manager Entity:

userID → { workExperience }

EventOrganizer Entity:

userID → { organizerLevel }

ServiceContracts Entity:

serviceID → {staffMemberID, companyName, cost }
companyName → { companyRating }

MaintenanceRequests Entity:

requestID → {userID, serviceID, description, severity, status, date }

Budget Entity:

budgetID → {amount, amountLimit, startDate, endDate, managerID }
projectType → { amountLimit }

Buildings Entity:

buildingName → {Address, City, Province, PostalCode}
postalCode → {City, Province}

RoomsInBuildings Entity:

{roomNumber, buildingName} → {capacity, roomType}
roomType → {capacity}
roomNumber → {floorNumber}

MaintainenceRequestsForRoomsInBuildings Entity:
No non-trivial functional dependencies.

ElectronicEquipment Entity:

deviceID → { expiryDate, energyRate, installationDate, lastDateOfMaintenance, cost, managerID, roomNumber, buildingName }

equipmentType → { energyRate, cost }

manufacturer → { warranty }

Events Entity:

eventID → {numberOfPeople, startTime, endTime, eventCost}
eventType → {eventCost, numberOfPeople}

EventsOrganizedBy Entity:

No non-trivial functional dependencies.

EventsHeldAt Entity:

{eventID, roomNumber, buildingName} → {purpose}

6. Normalization

We will be doing normalization via BCNF for all of our entities.

User Entity:

1) userID → { firstName, lastName, phoneNumber, SIN, address, province, city, postalCode}
2) SIN → { firstName, lastName, userID, phoneNumber, address, province, city, postalCode}
3) postalCode → {City, Province}

User(userID: INTEGER, firstName: CHAR(20), lastName: CHAR(20), SIN: INTEGER, address: CHAR(20), province: CHAR(20), city: CHAR(20), postalCode: CHAR(20))

Functional dependency 1 and 2 does not violate BCNF. This is because userID and SIN are both superkeys (infact they are also minimal keys – candidate keys).

Functional dependencies postalCode → {City, Province} violates BCNF. So we can decompose the User Entity table into such:

User(
userID: INTEGER [Candidate Key],
firstName: CHAR(20) NOT NULL,
lastName: CHAR(20) NOT NULL,
SIN: INTEGER NOT NULL UNIQUE, [Candidate Key],
address: CHAR(20) NOT NULL,

---

postalCode: CHAR(20),
PRIMARY KEY (userID),
FOREIGN KEY (postalCode) REFERENCES AddressInformation (postalCode)
ON DELETE SET NULL
)

AddressInformation(
postalCode: CHAR(20),
City: CHAR(20) NOT NULL,
Province: CHAR(20) NOT NULL,
PRIMARY KEY (postalCode)
)

Staff Member Entity:

Functional dependency: userID → { role }

StaffMember(
role: CHAR(20) NOT NULL,
userID: INTEGER [Candidate Key],
PRIMARY KEY (userID),
FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE CASCADE
)

The functional dependency does not violate BCNF hence no decomposition required.

Manager Entity:

Functional dependency: userID → { workExperience }

Manager(
workExperience: INTEGER NOT NULL,
userID: INTEGER [Candidate Key],
PRIMARY KEY (userID),
FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE CASCADE
)

The functional dependency does not violate BCNF hence no decomposition required.

EventOrganizer Entity:

userID → { organizerLevel }

EventOrganizer(
organizerLevel: INTEGER NOT NULL,
userID: INTEGER [Candidate Key],
PRIMARY KEY (userID),
FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE CASCADE
)

The functional dependency does not violate BCNF hence no decomposition required.

ServiceContracts Entity:

1) serviceID → {staffMemberID, companyName, cost }
2) companyName → { companyRating }

ServiceContracts(
serviceID: INTEGER [Candidate Key],
staffMemberID: CHAR(20) NOT NULL,
companyName: CHAR(50) NOT NULL,
companyRating: INTEGER,
cost: DECIMAL(10, 2),
PRIMARY KEY (serviceID)
FOREIGN KEY (staffMemberID) REFERENCES StaffMember(userID)
ON DELETE SET NULL
)

Functional dependency 1 does not violate BCNF. This is because serviceID is a superkey (in fact it is also a minimal key – candidate key).

Functional dependency 2 violates BCNF hence needs to be decomposed into the following tables:

ServiceContracts(
serviceID: INTEGER [Candidate Key],
staffMemberID: CHAR(20),
companyName: CHAR(50) NOT NULL,
cost: DECIMAL(10, 2),
PRIMARY KEY (serviceID)
FOREIGN KEY (staffMemberID) REFERENCES StaffMember(userID)
ON DELETE SET NULL
FOREIGN KEY (companyName) REFERENCES CompanyInformations (companyName)

```
        ON DELETE CASCADE
)
```

CompanyInformations(
        companyName: CHAR(20),
        companyRating: INTEGER NOT NULL,
        PRIMARY KEY (companyName)
)

MaintenanceRequests Entity:

1) requestID → {userID, serviceID, description, severity, status, date }

MaintenanceRequests(
   requestID: INTEGER [Candidate Key],
   userID: CHAR(20),
   serviceID: INTEGER,
   description: CHAR(20),
   severity: INTEGER Default 1,
   status: CHAR(20),
   date: DATE,
   PRIMARY KEY (requestID)
   FOREIGN KEY (userID) REFERENCES User(userID)
        ON DELETE SET NULL
   FOREIGN KEY (serviceID) REFERENCES ServiceContracts(serviceID)
        ON DELETE SET NULL
)

The functional dependency does not violate BCNF hence no decomposition required.

Budget Entity:

1) budgetID → {amount, amountLimit, startDate, endDate, managerID }
2) projectType → {amountLimit}

Budget(
        budgetID: INTEGER [Candidate Key],
        amount: DECIMAL(10,2),
        amountLimit: DECIMAL(10,2),
        startDate: Date,
        endDate: Date,
        managerID: INTEGER NOT NULL,
        projectType: CHAR(20),
        PRIMARY KEY (budgetID),
        FOREIGN KEY (managerID) REFERENCES Manager(userID)

ON DELETE SET NULL
)

Functional dependency 1 does not violate BCNF. This is because budgetID is a superkey (in fact it is also a minimal key – candidate key).

Functional dependency 2 violates BCNF hence needs to be decomposed into the following tables:

Budget(
budgetID: INTEGER [Candidate Key],
startDate: Date,
endDate: Date,
amount: DECIMAL(10,2)
managerID: INTEGER NOT NULL,
projectType: CHAR(20),
PRIMARY KEY (budgetID),
FOREIGN KEY (managerID) REFERENCES Manager(userID)
ON DELETE CASCADE
FOREIGN KEY (ProjectType) REFERENCES ProjectCost(ProjectType)
ON DELETE SET NULL
)

ProjectCost(
ProjectType: CHAR(20),
AmountLimit: Decimal(10,2),
PRIMARY KEY(ProjectType),
)

Buildings Entity:

1) buildingName → {Address, City, Province, PostalCode}
2) PostalCode → {City, Province}

Buildings(
buildingName: CHAR(20) [Candidate Key],
Address: CHAR(20) NOT NULL,
City: CHAR(20) NOT NULL,
Province: CHAR(20) NOT NULL,
PostalCode: Char(20) NOT NULL,
PRIMARY KEY (buildingName)
)

Functional dependency 1 does not violate BCNF. This is because buildingName is a superkey (in fact it is also a minimal key – candidate key).

Functional dependency 2 violates BCNF. Here we can just use the AddressInformation table.

Buildings(
buildingName: CHAR(20) [Candidate Key],
Address: CHAR(20) NOT NULL,
PostalCode: Char(20) NOT NULL,
PRIMARY KEY (buildingName)
FOREIGN KEY (PostalCode) REFERENCES AddressInformation(PostalCode)
ON DELETE SET CASCADE
)

AddressInformation(
postalCode: CHAR(20),
City: CHAR(20),
Province: CHAR(20),
PRIMARY KEY (postalCode)
)

RoomsInBuildings Entity:

1) {roomNumber, buildingName} → {capacity, roomType}
2) roomType → {capacity}
3) roomNumber → {floorNumber}

RoomsInBuildings(
roomNumber: INTEGER,
capacity: INTEGER NOT NULL,
floorNumber: INTEGER NOT NULL,
roomType: CHAR(20) NOT NULL,
buildingName: CHAR(20),
PRIMARY KEY(roomNumber, buildingName),
FOREIGN KEY (buildingName) REFERENCES Buildings(buildingName)
ON DELETE CASCADE
) // [Candidate Key] -> roomNumber + buildinName

Functional dependency 1 does not violate BCNF. This is because roomNumber+ buildingName is a superkey (in fact it is also a minimal key – candidate key).

Functional dependency 2 violates BCNF, this is because roomType is not a super key. hence we need to decompose into the following tables:

RoomsInBuildings(
roomNumber: INTEGER,
floorNumber: INTEGER NOT NULL,
roomType: CHAR(20) NOT NULL,
buildingName: CHAR(20),
PRIMARY KEY(roomNumber, buildingName),
FOREIGN KEY (buildingName) REFERENCES Buildings(buildingName)
ON DELETE CASCADE
FOREIGN KEY (roomType) REFERENCES RoomCapacityInformation (roomType)
ON DELETE SET CASCADE
) // [Candidate Key] -> roomNumber + buildinName

RoomCapacityInformation(
roomType: CHAR(20),
capacity: INTEGER NOT NULL,
PRIMARY KEY(roomType)
)

Functional dependency 3 violates BCNF, this is because roomNumber is not a super key. hence we need to decompose RoomsInBuildings. RoomCapacityInformation does not need to be decomposed further.

RoomsInBuildings(
roomNumber: INTEGER,
roomType: CHAR(20) NOT NULL,
buildingName: CHAR(20),
PRIMARY KEY(roomNumber, buildingName),
FOREIGN KEY (buildingName) REFERENCES Buildings(buildingName)
ON DELETE CASCADE
FOREIGN KEY (roomType) REFERENCES RoomCapacityInformation (roomType)
ON DELETE SET CASCADE
FOREIGN KEY (roomNumber) REFERENCES RoomFloorInformation(roomNumber)
ON DELETE CASCADE
) // [Candidate Key] -> roomNumber + buildinName

RoomFloorInformation(
roomNumber: INTEGER,
floorNumber: INTEGER NOT NULL,
PRIMARY KEY(roomNumber)

)

MaintainenceRequestsForRoomsInBuildings Entity:

No non-trivial functional dependencies. No BCNF decomposition required. Hence:

MaintainenceRequestsForRoomsInBuildings(
maintenanceRequestID: INTEGER,
roomNumber: INTEGER,
buildingName: CHAR(20),
PRIMARY KEY (maintenanceRequestID, roomNumber, buildingName)
FOREIGN KEY (maintainenenceRequestID) REFERENECES (requestID) MaintenanceRequest(requestID)
ON DELETE CASCADE
FOREIGN KEY (roomNumber, buildingName) REFERENCES RoomsInBuildings (roomNumber, buildingName)
ON DELETE CASCADE
) // [Candidate Key] -> roomNumber + buildingName + maintenanceRequestID

ElectronicEquipment Entity:

1) deviceID → { expiryDate, energyRate, installationDate, lastDateOfMaintenance, cost, managerID, roomNumber, buildingName }

2) equipmentType → { energyRate, cost }

3) manufacturer → { warranty }

ElectronicEquipment(
deviceID: INTEGER [Candidate Key],
expiryDate: Date NOT NULL,
equipmentType: CHAR(200 NOT NULL,
energyRate: DECIMAL(10,2) NOT NULL,
installationDate: Date NOT NULL,
lastDateOfMaintainenece: Date,
cost: DECIMAL(10,2) NOT NULL,
managerID: INTEGER DEFAULT 1,
roomNumber: INTEGER,
buildingName: CHAR(20),
manufacturer: CHAR(20) NOT NULL,
warranty: INTEGER NOT NULL,
PRIMARY KEY (deviceID),
FOREIGN KEY (managerID) REFERENCES Manager(userID)
ON DELETE SET NULL,
FOREIGN KEY (roomNumber, buildingName) References RoomsInBuildings(roomNumber, buildingName)
ON DELETE SET NULL;)

Functional dependency 1 does not violate BCNF. This is because deviceID is a superkey (in fact it is also a minimal key – candidate key).

Functional dependency 2 violates BCNF, this is because equipmentType is not a super key. hence we need to decompose into the following tables:

ElectronicEquipment(
deviceID: INTEGER [Candidate Key],
expiryDate: Date NOT NULL,
equipmentType: CHAR(200 NOT NULL,
installationDate: Date NOT NULL,
lastDateOfMaintainenece: Date,
managerID: INTEGER,
roomNumber: INTEGER,
buildingName: CHAR(20),
manufacturer: CHAR(20) NOT NULL,
warranty: INTEGER NOT NULL,
PRIMARY KEY (deviceID),
FOREIGN KEY (managerID) REFERENCES Manager(userID)
ON DELETE SET NULL,
FOREIGN KEY (roomNumber, buildingName) References RoomsInBuildings(roomNumber, buildingName)
ON DELETE SET NULL
FOREIGN KEY equipmentType REFERENCES (equipmentType) EquipmentEnergyAndCostInformation (equipmentType)
ON DELETE CASCADE
)

EquipmentEnergyAndCostInformation(
equipmentType: CHAR(20)
energyRate: Decimal(10,2) NOT NULL,
cost: Decimal(10,2) NOT NULL,
PRIMARY KEY (equipmentType)
)

Functional dependency 3 violates BCNF, this is because manufacturer is not a super key. hence we need to decompose into the following tables:

ElectronicEquipment(
deviceID: INTEGER [Candidate Key],
expiryDate: Date NOT NULL,
equipmentType: CHAR(200 NOT NULL,
installationDate: Date NOT NULL,
lastDateOfMaintainenece: Date,
managerID: INTEGER,

roomNumber: INTEGER,
buildingName: CHAR(20),
manufacturer: CHAR(20) NOT NULL,
PRIMARY KEY (deviceID),
FOREIGN KEY (managerID) REFERENCES Manager(userID)
ON DELETE SET NULL,
FOREIGN KEY (roomNumber, buildingName) References RoomsInBuildings(roomNumber, buildingName)
ON DELETE SET NULL
FOREIGN KEY equipmentType REFERENCES (equipmentType) EquipmentEnergyAndCostInformation (equipmentType)
ON DELETE CASCADE
FOREIGN KEY manufacturer EquipmentManufacturerWarrantyInformation (manufacturer) ON DELETE CASCADE
)

EquipmentManufacturerWarrantyInformation(
manufacturer: CHAR(20),
warranty: INTEGER NOT NULL,
PRIMARY KEY (manufacturer)
)

Events Entity:

1) eventID → {numberOfPeople, startTime, endTime, eventCost}
2) eventType → {eventCost, numberOfPeople}

Events(
eventID: INTEGER [Candidate Key],
numberOfPeople: INTEGER NOT NULL Default 0,
startTime: Date NOT NULL,
endTime: Date NOT NULL,
eventType: CHAR(20),
eventCost: DECIMAL(10,2) NOT NULL,
PRIMARY KEY(eventID),)

Functional dependency 1 does not violate BCNF. This is because eventID is a superkey (in fact it is also a minimal key – candidate key).

Functional dependency 2 violates BCNF, this is because eventType is not a super key. hence we need to decompose into the following tables:

Events(

---

eventID: INTEGER [Candidate Key],
startTime: Date NOT NULL,
endTime: Date NOT NULL,
eventType: CHAR(20),
PRIMARY KEY(eventID),
FOREIGN KEY eventType References EventInformation(eventType)
ON DELETE CASCADE)

EventInformation(
eventType: INTEGER,
eventCost: DECIMAL(10,2) NOT NULL,
numberOfPeople: INTEGER NOT NULL DEFAULT 0,
PRIMARY KEY (eventType)
)

EventsOrganizedBy Entity:

No non-trivial functional dependencies. No BCNF decomposition required. Hence:

EventsOrganizedBy(
eventID: INTEGER,
eventOrganizerID: INTEGER,
PRIMARY KEY(eventID, eventOrganizerID)
FOREIGN KEY(eventID) REFERENCES Events(eventID)
ON DELETE CASCADE
FOREIGN KEY(eventOragnizerID) references EventOrganizer(userID)
ON DELETE SET NULL
) // [Candidate Key] -> eventID + eventOrganizerID

EventsHeldAt Entity:

{eventID, roomNumber, buildingName} → {purpose}. No BCNF decomposition required:

EventsHeldAt(
eventID: INTEGER,
roomNumber: INTEGER,
buildingName: char(20),
purpose: CHAR(20),
PRIMARY KEY(eventID, roomNumber, buildingName)
FOREIGN KEY(eventID) references Events(eventID)
ON DELETE CASCADE
FOREIGN KEY(roomNumber, buildingName) references Rooms(buildingName, roomNumber)
ON DELETE CASCADE

```
) // [Candidate Key] -> eventID + roomNumber + buildingName
```

7. SQL DDL statements

```
CREATE TABLE User(
   userID INTEGER,
   firstName VARCHAR(20) NOT NULL,
   lastName VARCHAR(20) NOT NULL,
   SIN INTEGER NOT NULL UNIQUE,
   address VARCHAR(20) NOT NULL,
   postalCode VARCHAR(20),
   PRIMARY KEY (userID),
   FOREIGN KEY (postalCode) REFERENCES AddressInformation (postalCode)
ON DELETE SET NULL
);

CREATE TABLE AddressInformation(
        postalCode VARCHAR(20),
        City VARCHAR(20) NOT NULL,
        Province VARCHAR(20) NOT NULL,
        PRIMARY KEY (postalCode)
);

CREATE TABLE StaffMember(
   role VARCHAR(20) NOT NULL,
   userID INTEGER,
   PRIMARY KEY (userID),
   FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE CASCADE
);

CREATE TABLE Manager(
   workExperience INTEGER NOT NULL,
   userID INTEGER,
   PRIMARY KEY (userID),
   FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE CASCADE
);

CREATE TABLE EventOrganizer(
   organizerLevel INTEGER NOT NULL,
   userID INTEGER,
   PRIMARY KEY (userID),
   FOREIGN KEY (userID) REFERENCES User(userID)
```

```
ON DELETE CASCADE
);

CREATE TABLE ServiceContracts(
   serviceID INTEGER,
   staffMemberID VARCHAR(20),
   companyName VARCHAR(50) NOT NULL,
   cost DECIMAL(10, 2),
   PRIMARY KEY (serviceID)
   FOREIGN KEY (staffMemberID) REFERENCES StaffMember(userID)
ON DELETE SET NULL
  FOREIGN KEY (companyName) REFERENCES CompanyInformations
(companyName)
        ON DELETE CASCADE
);

CREATE TABLE CompanyInformations(
        companyName VARCHAR(20),
        companyRating INTEGER NOT NULL,
        PRIMARY KEY (companyName)
);

CREATE TABLE MaintenanceRequests(
   requestID INTEGER,
   userID VARCHAR(20),
   serviceID INTEGER,
   description VARCHAR(20),
   severity INTEGER Default 1,
   status VARCHAR(20),
   date DATE,
   PRIMARY KEY (requestID)
   FOREIGN KEY (userID) REFERENCES User(userID)
ON DELETE SET NULL
   FOREIGN KEY (serviceID) REFERENCES ServiceContracts(serviceID)
ON DELETE SET NULL
);

CREATE TABLE Budget(
        budgetID INTEGER,
        startDate Date,
        endDate Date,
        amount DECIMAL(10,2),
        managerID INTEGER NOT NULL,
        projectType VARCHAR(20),
        PRIMARY KEY (budgetID),
        FOREIGN KEY (managerID) REFERENCES Manager(userID)
```

```
            ON DELETE CASCADE
        FOREIGN KEY (ProjectType) REFERENCES ProjectCost(ProjectType)
            ON DELETE SET NULL
);

CREATE TABLE ProjectCost(
        ProjectType VARCHAR(20),
        AmountLimit Decimal(10,2),
        PRIMARY KEY(ProjectType)
);

CREATE TABLE Buildings(
        buildingName VARCHAR(20),
        Address VARCHAR(20) NOT NULL,
        PostalCode VARCHAR(20) NOT NULL,
        PRIMARY KEY (buildingName),
        FOREIGN KEY (PostalCode) REFERENCES AddressInformation(PostalCode)
            ON DELETE CASCADE
);

CREATE TABLE RoomCapacityInformation(
        roomType VARCHAR(20),
        capacity INTEGER NOT NULL,
        PRIMARY KEY(roomType)
);

CREATE TABLE RoomsInBuildings(
        roomNumber INTEGER,
        roomType VARCHAR(20) NOT NULL,
        buildingName VARCHAR(20),
        PRIMARY KEY(roomNumber, buildingName),
        FOREIGN KEY (buildingName) REFERENCES Buildings(buildingName)
            ON DELETE CASCADE
        FOREIGN KEY (roomType) REFERENCES RoomCapacityInformation
(roomType)
            ON DELETE CASCADE
        FOREIGN KEY (roomNumber) REFERENCES
RoomFloorInformation(roomNumber)
            ON DELETE CASCADE
);

CREATE TABLE RoomFloorInformation(
        roomNumber INTEGER,
        floorNumber INTEGER NOT NULL,
        PRIMARY KEY(roomNumber)
);
```

```
CREATE TABLE MaintenanceRequestsForRoomsInBuildings(
        maintenanceRequestID INTEGER,
        roomNumber INTEGER,
        buildingName VARCHAR(20),
        PRIMARY KEY (maintenanceRequestID, roomNumber, buildingName)
        FOREIGN KEY (maintenanceRequestID) REFERENCES
MaintenanceRequest(requestID)
                ON DELETE CASCADE
        FOREIGN KEY (roomNumber, buildingName) REFERENCES
RoomsInBuildings(roomNumber, buildingName)
                ON DELETE CASCADE
);

CREATE TABLE EquipmentEnergyAndCostInformation(
        equipmentType VARCHAR(20),
        energyRate Decimal(10,2) NOT NULL,
        cost Decimal(10,2) NOT NULL,
        PRIMARY KEY (equipmentType)
);

CREATE TABLE ElectronicEquipment(
        deviceID INTEGER,
        expiryDate Date NOT NULL,
        equipmentType VARCHAR(200) NOT NULL,
        installationDate Date NOT NULL,
        lastDateOfMaintenance Date,
        managerID INTEGER,
        roomNumber INTEGER,
        buildingName VARCHAR(20),
        manufacturer VARCHAR(20) NOT NULL,
        PRIMARY KEY (deviceID),
        FOREIGN KEY (managerID) REFERENCES Manager(userID)
                ON DELETE SET NULL,
        FOREIGN KEY (roomNumber, buildingName) References
RoomsInBuildings(roomNumber, buildingName)
                ON DELETE SET NULL
        FOREIGN KEY (equipmentType) REFERENCES
EquipmentEnergyAndCostInformation(equipmentType)
                ON DELETE CASCADE
        FOREIGN KEY (manufacturer) REFERENCES
EquipmentManufacturerWarrantyInformation(manufacturer)
                ON DELETE CASCADE
);

CREATE TABLE EquipmentManufacturerWarrantyInformation(
```

```
        manufacturer VARCHAR(20),
        warranty INTEGER NOT NULL,
        PRIMARY KEY (manufacturer)
);

CREATE TABLE Events(
        eventID INTEGER,
        startTime Date NOT NULL,
        endTime Date NOT NULL,
        eventType VARCHAR(20),
        PRIMARY KEY(eventID)
        FOREIGN KEY (eventType) References EventInformation(eventType)
ON DELETE CASCADE);

CREATE TABLE EventInformation(
        eventType INTEGER,
        eventCost DECIMAL(10,2) NOT NULL,
        numberOfPeople INTEGER NOT NULL DEFAULT 0,
        PRIMARY KEY (eventType)
);

CREATE TABLE EventsOrganizedBy(
        eventID INTEGER,
        eventOrganizerID INTEGER,
        PRIMARY KEY(eventID, eventOrganizerID)
        FOREIGN KEY(eventID) REFERENCES Events(eventID)
                ON DELETE CASCADE
        FOREIGN KEY(eventOrganizerID) references EventOrganizer(userID)
                ON DELETE SET NULL
);

CREATE TABLE EventsHeldAt(
        eventID INTEGER,
        roomNumber INTEGER,
        buildingName VARCHAR(20),
        purpose VARCHAR(20),
        PRIMARY KEY(eventID, roomNumber, buildingName)
        FOREIGN KEY(eventID) references Events(eventID)
                ON DELETE CASCADE
        FOREIGN KEY(roomNumber, buildingName) references Rooms(buildingName,
roomNumber)
                ON DELETE CASCADE
);
```

8. Insert statements

```
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(1, "Jessica", "Bator", 123321123, "666 Hello Street", "V11 0X0");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(2, "Bryan", "Chang", 890098890, "123 Bye Street", "V11 0X0");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(3, "William", "Cheng", 456654456, "456 AHH Street", "V66 Z7X");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(4, "Praveen", "Gupta", 789987789, "678 No.4 Road", "V66 Z7X");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(5, "Kate", "Kiczales", 234432234, "890 No.5 Road", "V12 Z34");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(6, "Julie", "Lee", 432623466, "346 No.1 Road", "V12 Z56");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(7, "Ivy", "Li", 245722347, "363 No.4 Road", "V12 Z78");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(8, "Melissa", "Lin", 125352362, "363 No.11 Road", "V11 0X0");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(9, "Vito", "Lin", 236543534, "363 No.12 Road", "V12 Z78");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(10, "Mohammed", "Shaikh", 457324762, "363 No.13 Road", "V66 Z7X");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(11, "Kyle", "Tam", 5785463262, "363 No.14 Road", "V12 Z78");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(12, "Sean", "Xu", 234653643, "363 No.15 Road", "V11 0X0");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(13, "Kevin", "Zhang", 578367946, "363 No.16 Road", "V66 Z7X");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(14, "Eric", "Fan", 234234444, "363 No.16 Road", "V66 Z7X");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(15, "Syed", "Imam", 456456444, "363 No.17 Road", "V66 Z7X");
INSERT INTO User(userID, firstName, lastName, SIN, address, postalCode)
VALUES(16, "Sandy", "Chu", 789789999, "363 No.16 Road", "V66 Z7X");

INSERT INTO AddressInformation(postalCode, City, Province) VALUES ("V11 0X0",
"Vancouver", "BC");
INSERT INTO AddressInformation(postalCode, City, Province) VALUES ("V66 Z7X",
"Richmond", "BC");
INSERT INTO AddressInformation(postalCode, City, Province) VALUES ("V12 Z34",
"Calgary", "AB");
INSERT INTO AddressInformation(postalCode, City, Province) VALUES ("V12 Z56",
"Toronto", "ON");
INSERT INTO AddressInformation(postalCode, City, Province) VALUES ("V12 Z78",
"Burnaby", "BC");
```

```
INSERT INTO StaffMember(role, userID) VALUES('Janitor', 1);
INSERT INTO StaffMember(role, userID) VALUES('Assistance', 2);
INSERT INTO StaffMember(role, userID) VALUES('Security', 3);
INSERT INTO StaffMember(role, userID) VALUES('Assistance', 4);
INSERT INTO StaffMember(role, userID) VALUES('Assistance', 5);

INSERT INTO Manager(workExperience, userID) VALUES(12, 6);
INSERT INTO Manager(workExperience, userID) VALUES(2, 7);
INSERT INTO Manager(workExperience, userID) VALUES(1, 8);
INSERT INTO Manager(workExperience, userID) VALUES(5, 9);
INSERT INTO Manager(workExperience, userID) VALUES(2, 10);

INSERT INTO EventOrganizer(organizerLevel, userID) VALUES(1, 11);
INSERT INTO EventOrganizer(organizerLevel, userID) VALUES(4, 12);
INSERT INTO EventOrganizer(organizerLevel, userID) VALUES(4, 13);
INSERT INTO EventOrganizer(organizerLevel, userID) VALUES(4, 14);
INSERT INTO EventOrganizer(organizerLevel, userID) VALUES(5, 15);

INSERT INTO ServiceContracts(serviceID, staffMemberID, companyName, cost)
VALUES (1, 1, "Facebook", 1000.00);
INSERT INTO ServiceContracts(serviceID, staffMemberID, companyName, cost)
VALUES (2, 2, "Tesla", 999.99);
INSERT INTO ServiceContracts(serviceID, staffMemberID, companyName, cost)
VALUES (3, 3, "Google", 99.99);
INSERT INTO ServiceContracts(serviceID, staffMemberID, companyName, cost)
VALUES (4, 4, "Sierra", 4444.44);
INSERT INTO ServiceContracts(serviceID, staffMemberID, companyName, cost)
VALUES (5, 5, "BC Hydro", 3333.33);

INSERT INTO CompanyInformations(companyName, companyRating) VALUES
("Facebook", 1);
INSERT INTO CompanyInformations(companyName, companyRating) VALUES
("Tesla", 4);
INSERT INTO CompanyInformations(companyName, companyRating) VALUES
("Google", 1);
INSERT INTO CompanyInformations(companyName, companyRating) VALUES
("Sierra", 3);
INSERT INTO CompanyInformations(companyName, companyRating) VALUES ("BC
Hydro", 3);

INSERT INTO MaintenanceRequests(requestID, userID, serviceID, description,
severity, status, date) VALUES (1, 1, 1, "Heater broken", 2, "unhandled", '2021-12-01');
INSERT INTO MaintenanceRequests(requestID, userID, serviceID, description,
severity, status, date) VALUES (2, 2, 1, "Light bulb broken", 1, "Fixing", '2021-12-02');
```

```
INSERT INTO MaintenanceRequests(requestID, userID, serviceID, description,
severity, status, date) VALUES (3, 2, 1, "Something broken", 1, "unhandled",
'2021-12-03');
INSERT INTO MaintenanceRequests(requestID, userID, serviceID, description,
severity, status, date) VALUES (4, 4, 1, "Something else broken", 4, "unhandled",
'2021-12-01');
INSERT INTO MaintenanceRequests(requestID, userID, serviceID, description,
severity, status, date) VALUES (5, 4, 1, "Even more things broken", 5, "unhandled",
'2021-12-05');

INSERT INTO Budget(budgetID, startDate, endDate, amount, managerID, projectType)
VALUES (1, '2021-12-01', '2022-12-01', 1000.00, 'Cafeteria maintenance');
INSERT INTO Budget(budgetID, startDate, endDate, amount, managerID, projectType)
VALUES (1, '2021-12-01', '2022-01-01', 100.00, 'Office cleaning');
INSERT INTO Budget(budgetID, startDate, endDate, amount, managerID, projectType)
VALUES (1, '2022-01-01', '2022-09-01', 300.00, 'Window cleaning');
INSERT INTO Budget(budgetID, startDate, endDate, amount, managerID, projectType)
VALUES (1, '2021-01-01', '2022-12-01', 500.00, 'Parkinglot cleaning');
INSERT INTO Budget(budgetID, startDate, endDate, amount, managerID, projectType)
VALUES (1, '2021-12-01', '2021-12-01', 5000.00, 'Ventilation system maintenance');

INSERT INTO ProjectCost(ProjectType, AmountLimit) VALUES('Cafeteria
maintenance', 5000.00);
INSERT INTO ProjectCost(ProjectType, AmountLimit) VALUES('Office cleaning',
500.00);
INSERT INTO ProjectCost(ProjectType, AmountLimit) VALUES('Window cleaning',
1000.00);
INSERT INTO ProjectCost(ProjectType, AmountLimit) VALUES('Parkinglot cleaning',
1000.00);
INSERT INTO ProjectCost(ProjectType, AmountLimit) VALUES('Ventilation system
maintenance', 50000.00);

INSERT INTO Buildings(buildingName, Address, PostalCode) VALUES("Hennings",
"363 No.11 Road", "V11 0X0");
INSERT INTO Buildings(buildingName, Address, PostalCode) VALUES("Hebb", "890
No.5 Road", "V12 Z34");
INSERT INTO Buildings(buildingName, Address, PostalCode) VALUES("Forestry", "363
No.13 Road", "V66 Z7X");
INSERT INTO Buildings(buildingName, Address, PostalCode) VALUES("Buchanon",
"666 Hello Street", "V11 0X0");
INSERT INTO Buildings(buildingName, Address, PostalCode) VALUES("Swing", "363
No.16 Road", "V66 Z7X");

INSERT INTO RoomCapacityInformation(roomType, capacity) VALUES ("Washroom",
5);
INSERT INTO RoomCapacityInformation(roomType, capacity) VALUES ("Office", 50);
```

```
INSERT INTO RoomCapacityInformation(roomType, capacity) VALUES ("Study rooms", 4);
INSERT INTO RoomCapacityInformation(roomType, capacity) VALUES ("Classroom", 50);
INSERT INTO RoomCapacityInformation(roomType, capacity) VALUES ("Lobby", 30);

INSERT INTO RoomsInBuildings(roomNumber, roomType, buildingName) VALUES (100, "Lobby", "Buchanon");
INSERT INTO RoomsInBuildings(roomNumber, roomType, buildingName) VALUES (202, "Washroom", "Forestry");
INSERT INTO RoomsInBuildings(roomNumber, roomType, buildingName) VALUES (323, "Classroom", "Hebb");
INSERT INTO RoomsInBuildings(roomNumber, roomType, buildingName) VALUES (121, "Classroom", "Hebb");
INSERT INTO RoomsInBuildings(roomNumber, roomType, buildingName) VALUES (666, "Office", "Swing");

INSERT INTO RoomFloorInformation(roomNumber, floorNumber) VALUES (100, 1);
INSERT INTO RoomFloorInformation(roomNumber, floorNumber) VALUES (202, 2);
INSERT INTO RoomFloorInformation(roomNumber, floorNumber) VALUES (323, 3);
INSERT INTO RoomFloorInformation(roomNumber, floorNumber) VALUES (121, 1);
INSERT INTO RoomFloorInformation(roomNumber, floorNumber) VALUES (666, 6);

INSERT INTO MaintenanceRequestsForRoomsInBuildings(maintenanceRequestID, roomNumber, buildingName) VALUES (1, 100, "Buchanon");
INSERT INTO MaintenanceRequestsForRoomsInBuildings(maintenanceRequestID, roomNumber, buildingName) VALUES (2, 100, "Buchanon");
INSERT INTO MaintenanceRequestsForRoomsInBuildings(maintenanceRequestID, roomNumber, buildingName) VALUES (3, 202, "Forestry");
INSERT INTO MaintenanceRequestsForRoomsInBuildings(maintenanceRequestID, roomNumber, buildingName) VALUES (4, 666, "Swing");
INSERT INTO MaintenanceRequestsForRoomsInBuildings(maintenanceRequestID, roomNumber, buildingName) VALUES (5, 100, "Buchanon");

INSERT INTO EquipmentEnergyAndCostInformation(equipmentType, energyRate, cost) VALUES ("Heater", 300.00, 400.00);
INSERT INTO EquipmentEnergyAndCostInformation(equipmentType, energyRate, cost) VALUES ("Projector", 78.00, 500.00);
INSERT INTO EquipmentEnergyAndCostInformation(equipmentType, energyRate, cost) VALUES ("Thermostat", 0.50, 10.00);
INSERT INTO EquipmentEnergyAndCostInformation(equipmentType, energyRate, cost) VALUES ("AC", 200.00, 300.00);
INSERT INTO EquipmentEnergyAndCostInformation(equipmentType, energyRate, cost) VALUES ("Speaker", 20.00, 350.00);
```

```
INSERT INTO ElectronicEquipment(deviceID, expiryDate, equipmentType,
installationDate, lastDateOfMaintenance, managerID, roomNumber, buildingName,
manufacturer) VALUES (1, '2021-12-05', "Heater", '2015-12-05', '2021-12-05', 6, 100,
"Buchanon", 'Zaber');
INSERT INTO ElectronicEquipment(deviceID, expiryDate, equipmentType,
installationDate, lastDateOfMaintenance, managerID, roomNumber, buildingName,
manufacturer) VALUES (2, '2021-09-01', "Projector", '2015-12-05', '2022-12-05', 6, 100,
"Buchanon", 'BlueSky');
INSERT INTO ElectronicEquipment(deviceID, expiryDate, equipmentType,
installationDate, lastDateOfMaintenance, managerID, roomNumber, buildingName,
manufacturer) VALUES (3, '2021-10-01', "Heater", '2015-12-05', '2022-10-01', 6, 202,
"Forestry", 'Canbat');
INSERT INTO ElectronicEquipment(deviceID, expiryDate, equipmentType,
installationDate, lastDateOfMaintenance, managerID, roomNumber, buildingName,
manufacturer) VALUES (4, '2021-11-15', "AC", '2015-12-05', '2022-11-05', 4, 666,
"Swing", 'Arius');
INSERT INTO ElectronicEquipment(deviceID, expiryDate, equipmentType,
installationDate, lastDateOfMaintenance, managerID, roomNumber, buildingName,
manufacturer) VALUES (5, '2021-07-11', "Heater", '2015-12-05', '2021-07-11', NULL,
NULL, NULL, 'Viker');

INSERT INTO EquipmentManufacturerWarrantyInformation(manufacturer, warranty)
VALUES ("Zaber", 5);
INSERT INTO EquipmentManufacturerWarrantyInformation(manufacturer, warranty)
VALUES ("BlueSky", 1);
INSERT INTO EquipmentManufacturerWarrantyInformation(manufacturer, warranty)
VALUES ("Canbat", 2);
INSERT INTO EquipmentManufacturerWarrantyInformation(manufacturer, warranty)
VALUES ("Arius", 5);
INSERT INTO EquipmentManufacturerWarrantyInformation(manufacturer, warranty)
VALUES ("Viker", 5);

INSERT INTO Events(eventID, startTime, endTime, eventType) VALUES (1,
'2021-11-22', '2021-11-15', 'Lunch event');
INSERT INTO Events(eventID, startTime, endTime, eventType) VALUES (2,
'2022-11-15', '2022-11-15', 'Dinner event');
INSERT INTO Events(eventID, startTime, endTime, eventType) VALUES (3,
'2021-12-01', '2021-12-01', 'Concert');
INSERT INTO Events(eventID, startTime, endTime, eventType) VALUES (4,
'2021-09-01', '2021-09-01', 'Lunch event');
INSERT INTO Events(eventID, startTime, endTime, eventType) VALUES (5,
'2021-07-12', '2021-07-12', 'Lunch event');

INSERT INTO EventInformation(eventType, eventCost, numberOfPeople) VALUES
('Concert', 5000, 200);
```

```
INSERT INTO EventInformation(eventType, eventCost, numberOfPeople) VALUES
('Lunch event', 1000, 500);
INSERT INTO EventInformation(eventType, eventCost, numberOfPeople) VALUES
('Dinner event', 2200, 500);
INSERT INTO EventInformation(eventType, eventCost, numberOfPeople) VALUES
('Orientation', 600, 100);
INSERT INTO EventInformation(eventType, eventCost, numberOfPeople) VALUES
('Games night', 300, 80);

INSERT INTO EventsOrganizedBy(eventID, eventOrganizerID) VALUES (1, 11);
INSERT INTO EventsOrganizedBy(eventID, eventOrganizerID) VALUES (1, 12);
INSERT INTO EventsOrganizedBy(eventID, eventOrganizerID) VALUES (2, 11);
INSERT INTO EventsOrganizedBy(eventID, eventOrganizerID) VALUES (2, 12);
INSERT INTO EventsOrganizedBy(eventID, eventOrganizerID) VALUES (3, 13);

INSERT INTO EventsHeldAt(eventID, roomNumber, buildingName, purpose) VALUES
(1, 323, 'Hebb', 'ENPH beef and pizza');
INSERT INTO EventsHeldAt(eventID, roomNumber, buildingName, purpose) VALUES
(2, 100, 'Buchanon', 'Career night');
INSERT INTO EventsHeldAt(eventID, roomNumber, buildingName, purpose) VALUES
(3, 323, 'Hebb', 'Guest speaker');
INSERT INTO EventsHeldAt(eventID, roomNumber, buildingName, purpose) VALUES
(4, 121, 'Hebb', 'Student party');
INSERT INTO EventsHeldAt(eventID, roomNumber, buildingName, purpose) VALUES
(5, 666, 'Swing', 'Cake day');
```