TEAM 48

CSC2042

Rowan Adair 40226787
radair10@qub.ac.uk

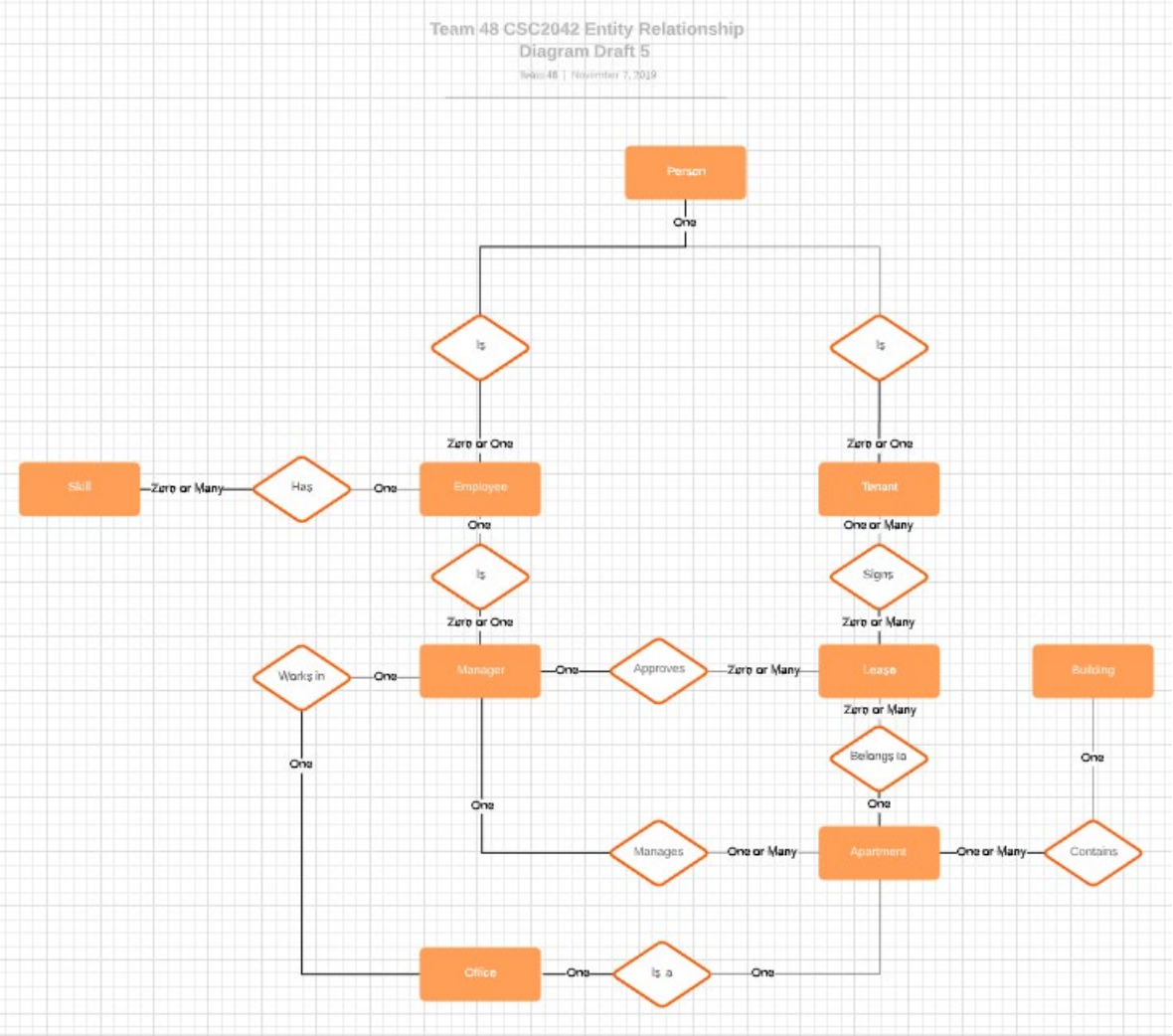
Barney Young 40231585
byoung03@qub.ac.uk

Scott Lam McGonnell 40227918
slammcgonnell01@qub.ac.uk

Youssef Emam 40247459
yemanmohammedsaber01@qub.ac.uk

Connall Carlin 40204680
ccarlin17@qub.ac.uk

Entity Relationship Diagram



Constraints and About the Entity Relationship Diagram

A building owned by Queens University Accommodation will contain one or many apartments, but an apartment must be contained within an apartment building.

It has been assumed that in reference to the document, a person is an entity that can be zero or one tenant or an employee. A person will have a separate set of bank details as a tenant needs to pay rent from a bank account and an employee needs a salary payed into their account. A person will have one and only one set of bank details.

As is stated in the assignment a person can be a tenant, hence a zero or one relation and the same is to be said for an employee. Both an employee and tenant must be one person. Hypothetically a person can be a tenant and employee. Tenants and employees can do many different things.

Employees may have zero or many skills, implying them to be an engineer of some degree. A list of skills will be available to be assigned to one employee. The role of a manager must exist within the structure of the database as inferred from the assignment documentation. It is then assumed that a manager is derived from employee meaning that an employee is or is not one manager, but a manager must be one and only one employee.

The role of a manager is twofold according to the assignment documentation. Firstly, they are responsible for the signing and approval of leases; a lease must be approved by one manager, but a manger can approve many leases in their tenure; hence the approval relationship in the ERD. Secondly, a manager manages at least one apartment in a building or range of buildings as it can be assumed that an apartment requires one and only one manager. Hence a manger may manage one or many apartments while employed. A perk of being a manger means you have an office in one of the buildings according to the assignment. So, a manager will have one office and an office will belong to one manager at any one time. Since a office is one of the apartments then an office is one apartment and one apartment can be an office.

In relation to the role of tenants in the database, when a tenant moves into an apartment, they must sign a lease. A tenant can assign zero or many leases if they wish to rent multiple facilities. A lease must be assigned by a minimum of one tenant. Leases will belong to one and only one apartment, but it can be assumed over time multiple leases will be taken out for an apartment therefore an apartment can have no leases, or many leases linked to it.

Such is the relational structure of this database system.

Database Design

Building(**BuildingID**, Address, Postcode)

Apartment(Apt. ID, Apt. No, Bedrooms, Bathrooms, Total Area, **ManagerID**)

ApartmentBuilding(**AptID, BuildingID**)

Office(OfficeID, **AptID**)

ManagerOffice(**ManagerID, OfficeID)**

ApartmentManager(**AptID, ManagerID**)

Person(PersonID, FirstName, Surname, Emergency Contact Name, Emergency Conatct Numer, Emergency Contact Relation, BankAccNo)

Tenant(TenantID, **PersonID**)

Employee(EmployeeID, **PersonID**, Pay)

Manager(ManagerID, **EmployeeID**)

TechnicianSkill(**EmployeeID**, **SkillID)**

Skill(**SkillID**, SkillName)

Lease(LeaseID, **AptID**, StartDate, Duration, Rent)

LeaseTenant(LeaseTenantID**, LeaseID**, **TenantID**)

LeaseManager(**LeaseID, ManagerID)**

- Primary Key
- **Foreign Key**

SQL Queries

Query 1: Find the number of apartments with/without ensuites and the total apartments in the building.

```
SELECT apartmentbuilding.BuildingID AS "Building",
building.Address AS "Address",
 COUNT(CASE
    WHEN apartment.Bedrooms = apartment.Bathrooms
    THEN apartment.aptid
    END) AS "Number of Ensuite Apartments",
 COUNT(CASE
    WHEN  apartment.Bedrooms != apartment.Bathrooms
    THEN apartment.aptid
    END) AS "Number of Non-ensuite Apartments",
 COUNT(apartment.aptid) AS "Total Apartments"
FROM apartment
INNER JOIN apartmentbuilding ON apartment.AptID = apartmentbuilding.AptID
INNER JOIN building ON apartmentbuilding.BuildingID = building.BuildingID
GROUP BY apartmentbuilding.buildingid;
```

This query will return five separate columns. Firstly, the ID of the building being queried as “Building”. The address of the building as to confirm the location. Along with this identifying information, the next three columns return separate conditional counts based on assumptions about the system. Numbered 1 – 3 below explains each count case in order :

1. In the case where the number of bedrooms is equal to the number of bathrooms it can be assumed that the apartment is an ensuite apartment, as in reality apartments may have x number of bedrooms and possibly two/three shared bathrooms. Along with this in real life if x number of bedrooms is the same as the bathrooms then each bedroom has a bathroom.
2. The second count case is the reverse of the above count case assuming where the number of bedrooms does not equal the number of bathrooms; it can be inferred that the apartment is without ensuites.
3. The final count simply returns an unconditional value returning the total apartments in the queried building.

The query does make certain assumptions that based on the Elms Village layout of apartments that there is either a matching number of ensuites to bedrooms, and if there are not then there is a shared bathroom between tenants. A query like this may benefit from changes to the system going forward, for example perhaps bedrooms themselves should have their own tables determining if there is or is not an ensuite in this room. The query however functionally works based on the previous assumptions.

The inner joins join the tables :

1. apartment to apartmentbuilding through the “aptID” foreign key in apartmentbuilding and primary key in apartment.
2. apartmentbuilding to t building through “buildingID” foreign key in apartmentbuilding and primary key in building.

Finally, the query will group each record by “buildingID”.

Query 2 : Find managers managing multiple apartments over multiple buildings.

Queen's Accommodation wish to find out which managers are managing apartments that are in different buildings to each other. With this information they would be able to keep track of how difficult it is for the manager to manage their designated apartments and redistribute the management of apartments accordingly to ease the workload of their staff.

```
SELECT Person.FirstName AS "First name", Person.LastName AS "Last name", Manager.ManagerID AS "ManagerID"
FROM Person
JOIN Employee ON Person.PersonID = Employee.PersonID
JOIN Manager ON Employee.EmployeeID = Manager.EmployeeID
JOIN (
    SELECT ApartmentManager.ManagerID, COUNT(DISTINCT ApartmentBuilding.BuildingID) AS "Number of buildings"
    From ApartmentManager
    JOIN ApartmentBuilding ON ApartmentManager.AptID = ApartmentBuilding.AptID
    GROUP BY ManagerID
    HAVING `Number of buildings` > 1
    ) dataTbl ON Manager.ManagerID = dataTbl.ManagerID;
```

The first SQL query will return a table of all managers' first names, last names, and manager IDs. The nested query returns a table with managers' IDs, and a numerical value, which is the number of buildings, across which, they manage apartments.

The two tables are then joined to provide an output of only the managers' first names, last names, and manager IDs who manage apartments across more than one building.

Query 3 : Find employees with two or more skills and increase their pay appropriately.

```
UPDATE employee
SET employee.pay=employee.pay * 1.2
WHERE employeeID in(
SELECT * FROM(SELECT technicianskill.EmployeeID
FROM technicianskill
LEFT JOIN employee ON technicianskill.EmployeeID=employee.EmployeeID
GROUP BY EmployeeID
HAVING  COUNT(DISTINCT technicianskill.SkillID)>1)tblTmp
);
```

Coping with Changes

Expansion Plan:

Queens Accommodation has a requirement for better management of their employees , in particular, the managers. Seeking for ways to distinguish different roles within the company more clearly than the current system provides. Therefore, they can place people in the correct roles, more suited to their ability and experience. QA are also in search of more experienced staff with broader skill sets and in conjunction; a system to track such experience and manage it appropriately. A system to better determine who is fit management positions based on relevant skills, tenure and various other quantifiers that will be explained later. The may also be required to keep track of staff conduct, rewarding or disciplining accordingly.

System Changes:

Individual Contributions Record

Group Member St. Number	Task(i)	Task(ii)	Task(iii)	Task(iv)	Group Member Totals
40226787	60	5	17.5	15	97.5
40231585	10	37.5	17.5	15	80
40247459	10	30	17.5	15	72.5
40204680	10	0	30	40	80
40227918	10	27.5	17.5	15	70
Total	100	100	100	100	400

The team has agreed on the proportional weighting of this ICR and believe it is reflective of work done.

Individual Contributions Narrative

Rowan Adair: My primary role within the project was as team lead. Organizing the work done by team members using a Kanban through Trello to track stories. I also set up a private Git repository with extensive information on the syntax we would use, software used and anything else that needs to be known. Making sure everyone had the same scripts, report version and relevant resources. Along with this I organized and coordinated team meetings weekly, with stand-up meetings and more extensive 2-hour group work meetings. I designed the entity relationship diagram and established the different constraints within the system documenting all relevant assumptions. I was also responsible for completing query 1 in our group's querydb48.sql file. Along with writing my own script I tested the rest of the team's scripts and corrected any bugs as they arose. In summary my primary role was team lead and organization, my secondary role was system design and query script writing. My tertiary role was testing and debugging any scripts pushed to the master branch.
Connall Carlin: My roles within this project range from assisting group members in tasks listed for this project where appropriate. My specific roles in which I invested the largest amount of time was in creating the insert script for the database which I quickly learned this was a long task if done insufficiently. As a result, I designed small java programs to generate random values based off a defined array or random numbers in order to speed up the creation process and ensure accurate yet plentiful test data. Once finalised we as a group discussed coping with changes within the database as well as allowing for/looking in to possible areas for expansion and it was my role to bring these ideas together into a coherent analysis for future proofing.
Barney Young: The most significant part of my contribution was the createdb48.sql file, which I wrote, based on the ERD and schema. I was also responsible for completing query 2 in our group's querydb48.sql file. Like the other group members, I helped in other ways by attending and contributing to the group meetings and correcting and assisting with some errors in group files.
Youssef Emam:
Scott Lam: