

DAMG 6210

Data Management and Database Design

Team No: 1

Project Name: Rental Property Management system

Course name: DAMG 6210 Data Management and Database Design

Year: Fall 2021



Team Members:

| Name | Contact Information |
|-------------------|--|
| Aishwarya Balyaya | balyaya.a@northeastern.edu |
| Divyesh Darji | darji.di@northeastern.edu |
| Srushti Desai | desai.sr@northeastern.edu |
| Shrutika Salian | salian.s@northeastern.edu |



Objectives:

- To develop an efficient database system which eases the process of rental management
- To build a data source for performing analytics
- Maintain records for apartment details to keep track of availability
- Analyze overdues, early payment and accordingly inform the tenants
- To track the profit margin using revenue generated and expenditure



Problem Statement:

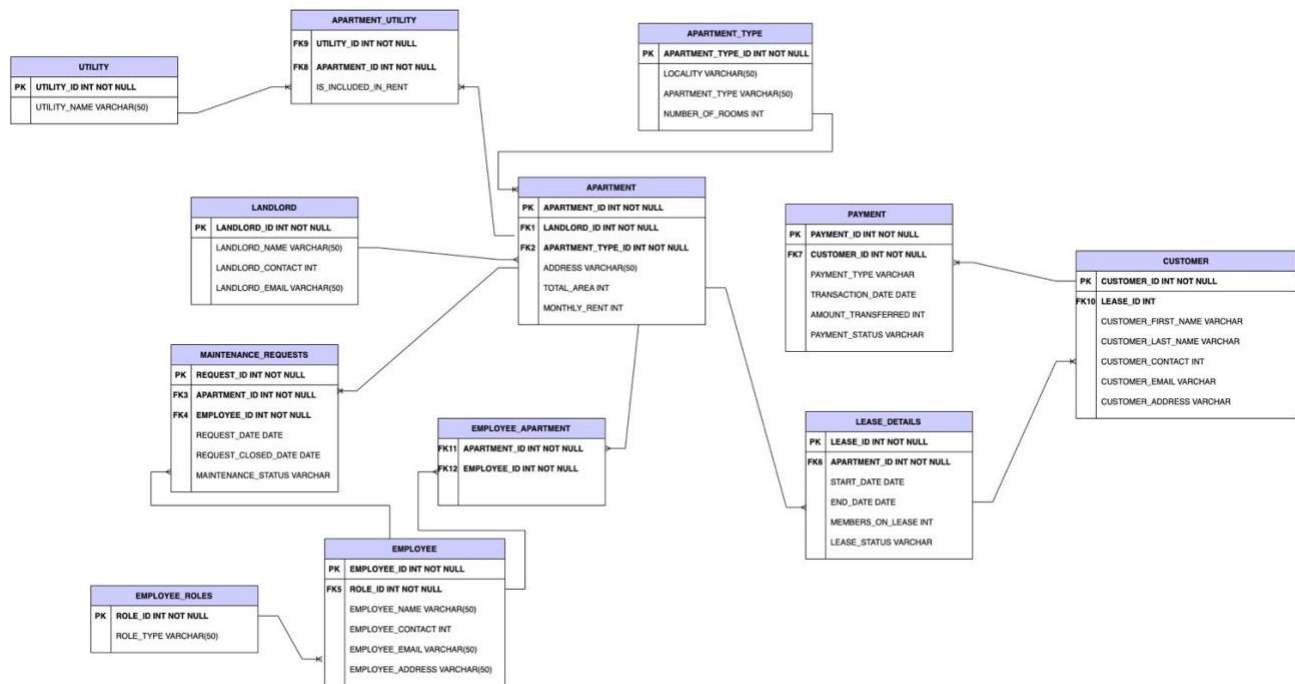
In this fast-paced world, with so many ways to generate data, it is very essential the data being generated is collected and managed in a structured manner so that it can be converted into useful information. Storing such an enormous amount of data is a cause of concern for many stakeholders. Such problems are faced even by real estate firms where they have multiple customers looking for rental housing. Thus, it requires a robust database management system that facilitates connecting the customers with the appropriate property that suits their needs. The firms have multiple customers approaching to search for a rental property and storing their data and analyzing it accurately can help them gain revenues. This can only be possible if they have an organized process where the data is stored appropriately. The aim of our system is to design such a model that can help the organization to manage all the customers who are on lease or searching for a property along with keeping the track of organization's profit and has a hassle-free experience.



PROPOSED SOLUTION:

- Using **MIN_AMOUNT** and **MAX_AMOUNT**, availability of the apartment can be shown to customers as per their budget preferences.
- To know about utilities included in rent for each apartment, we are using association entity **APARTMENT_UTILITY** having **PRIMARY KEY (APARTMENT_ID, UTILITY_ID)** which connects **APARTMENT** and **UTILITY** entity
- For each apartment in **APARTMENT** entity, there is a **LEASE_ID** associated through which we can find the current tenants from the **CUSTOMER** entity using **FOREIGN KEY(LEASE_ID)**. This can further help to track the payments done by each customer using **FOREIGN KEY (CUSTOMER_ID)** in **PAYMENT** entity
- Each landlord can view the current tenants living in associated apartments by creating **VIEWS** using **APARTMENT-> LEASE_DETAILS-> CUSTOMER** relationship
- Tracking the maintenance required for each apartment we are using **FOREIGN KEY APARTMENT_ID** in **MAINTENANCE_REQUESTS** entity

ENTITY-RELATIONSHIP DIAGRAM



APARTMENT ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|-------------------|--------------|--|
| APARTMENT_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| ADDRESS | VARCHAR (50) | NOT NULL |
| TOTAL_AREA | INTEGER (10) | NOT NULL |
| MONTHLY_RENT | INTEGER (10) | NOT NULL |
| EMPLOYEE_ID | INTEGER (10) | FOREIGN KEY which references EMPLOYEE_ID from EMPLOYEE entity. This is used to fetch the details of assigned employee. |
| LANDLORD_ID | INTEGER (10) | FOREIGN KEY which references LANDLORD_ID from LANDLORD entity. This is used to fetch the details of landlord. |
| APARTMENT_TYPE_ID | INTEGER (10) | FOREIGN KEY which references APARTMENT_ID from APARTMENT entity. This is used to fetch the details of apartments. |

LANDLORD ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|------------------|--------------|-----------------------|
| LANDLORD_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| LANDLORD_NAME | VARCHAR (50) | NOT NULL |
| LANDLORD_CONTACT | INTEGER (10) | NOT NULL, UNIQUE |
| LANDLORD_EMAIL | VARCHAR (50) | NOT NULL, UNIQUE |

EMPLOYEE ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|------------------|--------------|---|
| EMPLOYEE_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| EMPLOYEE_NAME | VARCHAR (50) | NOT NULL |
| EMPLOYEE_CONTACT | INTEGER (10) | NOT NULL, UNIQUE |
| EMPLOYEE_EMAIL | VARCHAR (50) | NOT NULL, UNIQUE |
| EMPLOYEE_ADDRESS | VARCHAR (50) | NOT NULL |
| ROLE_ID | INTEGER (10) | FOREIGN KEY which references ROLE_ID from EMPLOYEE_ROLES entity. This is used to fetch the details of employee roles. |

EMPLOYEE_ROLES ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|------------|--------------|-----------------------|
| ROLE_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| ROLE_TYPE | VARCHAR (50) | NOT NULL |

APARTMENT_UTILITY ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|---------------------|--------------|---|
| IS_INCLUDED_IN_RENT | VARCHAR (50) | NOT NULL |
| APARTMENT_ID | INTEGER (10) | FOREIGN KEY which references APARTMENT_ID from APARTMENT entity. This is used to fetch the details of apartments. |

| | | |
|------------|--------------|--|
| UTILITY_ID | INTEGER (10) | FOREIGN KEY which references UTILITY_ID from UTILITY entity. This is used to fetch the details of apartment utilities. |
|------------|--------------|--|

UTILITY ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|--------------|--------------|-----------------------|
| UTILITY_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| UTILITY_NAME | VARCHAR (50) | NOT NULL |

MAINTENANCE_REQUESTS ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|---------------------|--------------|---|
| REQUEST_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| REQUEST_DATE | DATE | NOT NULL |
| REQUEST_CLOSED_DATE | DATE | NOT NULL |
| APARTMENT_ID | INTEGER (10) | FOREIGN KEY which references APARTMENT_ID from APARTMENT entity. This is used to fetch the details of apartments. |

LEASE_DETAILS ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|------------------|--------------|---|
| LEASE_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| START_DATE | DATE | NOT NULL |
| END_DATE | DATE | NOT NULL |
| MEMBERS_ON_LEASE | INTEGER (10) | NOT NULL |
| APARTMENT_ID | INTEGER (10) | FOREIGN KEY which references APARTMENT_ID from APARTMENT entity. This is used to fetch the details of apartments. |

CUSTOMER ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|------------------|--------------|---|
| CUSTOMER_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| CUSTOMER_NAME | VARCHAR (50) | NOT NULL |
| CUSTOMER_CONTACT | INTEGER (10) | NOT NULL, UNIQUE |
| LEASE_ID | INTEGER (10) | FOREIGN KEY which references LEASE_ID from LEASE_DETAILS. This is used to fetch the details of lease. |

PAYMENT ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|--------------------|--------------|---|
| PAYMENT_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| PAYMENT_TYPE | VARCHAR (50) | NOT NULL |
| TRANSACTION_DATE | DATE | NOT NULL |
| AMOUNT_TRANSFERRED | INTEGER (10) | NOT NULL |
| CUSTOMER_ID | INTEGER (10) | FOREIGN KEY which references CUSTOMER_ID from CUSTOMER. This is used to fetch the details of customers. |

APARTMENT_TYPE ENTITY

| ATTRIBUTES | DOMAIN | COMMENTS |
|-------------------|--------------|-----------------------|
| APARTMENT_TYPE_ID | INTEGER (10) | NOT NULL, PRIMARY KEY |
| LOCALITY | VARCHAR (50) | NOT NULL |
| APARTMENT_TYPE | VARCHAR (50) | NOT NULL |
| NO_OF_ROOMS | INTEGER (10) | NOT NULL |

BUSINESS RULES:

- **Employee** can check the **availability** of **one or more rooms** according to **customer preferences**
- **Each landlord** can check the number of **apartments** he/she is leasing
- **Each landlord** can check the details of **one or more tenants** associated with **one or more apartments**
- **Each landlord** can keep a track on **one or more apartment's rent** under him/her
- **Employee** can track **history of lease details** for **every apartment** for **each year** using **Lease_Details** entity
- **Employee** can keep a check on whether the **payment** for a **particular month** for an **apartment** is done or not with the help of **Payment entity** based on the **transaction date**
- **Employee** can manage **multiple apartments**
- Once the **apartment** has been rented, the **customer** is assigned **lease_id**
- **Customer** can keep a track on their **payment dues**
- **Each payment** can belong to **one or more tenants** but **only one lease_id**
- **One booking** can only be processed by **one payment_id**
- **Each customer** can choose the **apartment** according to their **preferences** based on **locality and budget**
- **Each apartment** can have **multiple utilities** which **may or may not be included in rent**
- **Employee** can view and manage **maintenance requests**
- Once the **lease date** is assigned to a **customer**, the **status** of the **apartment is changed accordingly**
- According to **maintenance request closed date**, maintenance **status** can be tracked

SECURITY CONSTRAINTS: (USER LEVEL/PERMISSIONS)

ADMIN:

1. Has full access i.e., READ/WRITE/UPDATE all the entities in the database

MANAGER:

1. Has read access for all the entities in the database.
2. Has UPDATE access to customer, lease, payment, and maintenance requests.