Technical report Workshop 1 Database foundations

Angel Andres Diaz Vergara - *aadiazv@udistrital.edu.co*

1. INTRODUCTION

This report outlines the database design for an apartment complex management system, detailing user stories, technical decisions, and the entire process to form the database

2. user stories

As a tenant, I want to pay administration online so what I can manage payments quickly.

As a tenant, I want to report maintenance issues so what they can be addressed in a timely manner.

As a tenant, I want to book common spaces so what I can plan and organize my personal activities.

As a tenant, I want to view available apartments and blocks so that I can make informed decisions about leasing.

As a tenant, I want to reserve and manage my parking spot so that I can ensure I have a designated space for my vehicle.

3. technical and design consideratios/decisions

The database is designed using normalization principles to ensure data integrity and avoid redundancy, key relationships, such as between tenants, apartments, and payments, are modeled using one-to-many and many-to-many relationships with linking tables.

4. DataBase Design

4.1. define components

the components are:
- list of apartments
- payment administration services
- maintenance request
- reservation common spaces
- manage of parking lots

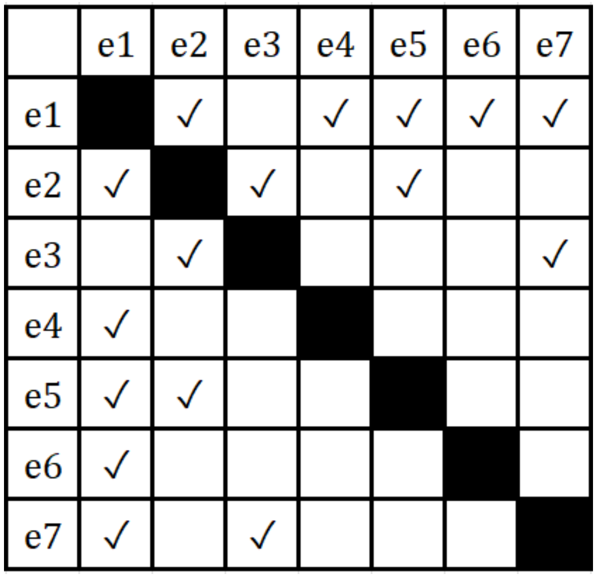
4.2. define entities

the entities are
1. Tenant
2. Apartment
3. block
4. payment
5. maintenance_request
6. reservation
7. parking spot

4.3. define attributes per entity

attributes per entity are:
1. Tenant = tenant_id, name, email, phone_number
2. Apartment = apartment_id, tenant_id, block_id, apartment_number, floor, num_bedrooms, num_bathrooms, administration_price, available
3.Block = block_id, block_name, address
4.Payment = payment_id, tenant_id, amount, payment_date, payment_method, payment_description, status
5.Maintenance Request = request_id, tenant_id, apartment_id, issue_description, request_date, status, priority
6.Reservation = reservation_id, tenant_id, common_zone, reservation_date, start_time, end_time
7.Parking Spot = spot_id, tenant_id, block_id, spot_number, reservation_status

4.4. define relationships



	e1	e2	e3	e4	e5	e6	e7
e1	■	✓		✓	✓	✓	✓
e2	✓	■	✓		✓		
e3		✓	■				✓
e4	✓			■			
e5	✓	✓			■		
e6	✓					■	
e7	✓		✓				■

Figure 1. relationship table.

4.5. define relationships types

the relation types are:

1. Tenant - Apartment (One-to-Many)
One tenant can rent one or more apartments, but an apartment can be rented by one tenant at time.

2. Tenant - Payment (One-to-Many)
One tenant can make many payments.

3. Tenant - Maintenance Request (One-to-Many)
One tenant can submit many maintenance requests.

4. Tenant - Reservation (Many-to-Many)
One tenant can make many reservations.

5. Tenant - Parking Spot (One-to-One)
One tenant can reserve one parking spot.

6. Apartment - block (Many-to-one)
One block can have many apartments.

7. Apartment - Maintenance Request (One-to-Many)
One apartment can have many maintenance requests.

8. Block - Parking Spot (One-to-Many)
One block can have many parking spots.

4.6. first entity-relationships draw

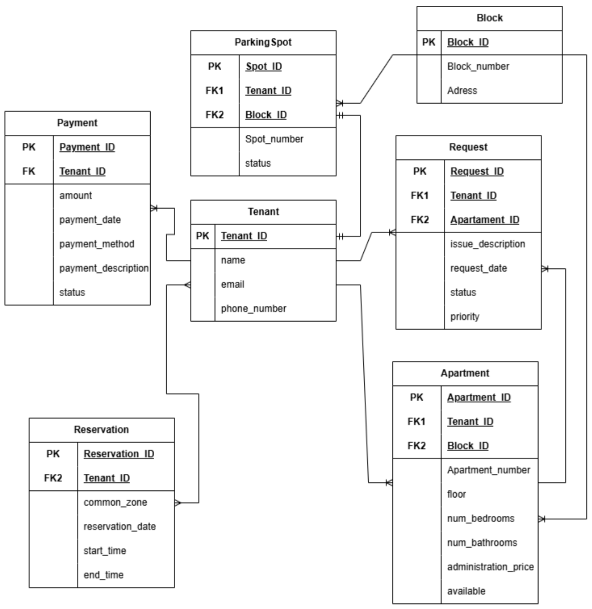


Figure 2. First E.R draw.

4.7. first split many-to-many relationships

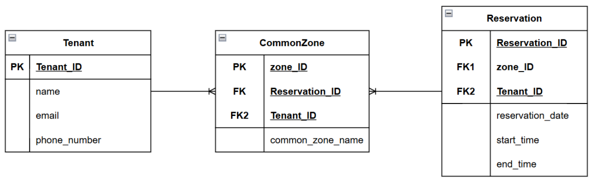


Figure 3. split many to many draw.

4.8. second entity-relationship draw

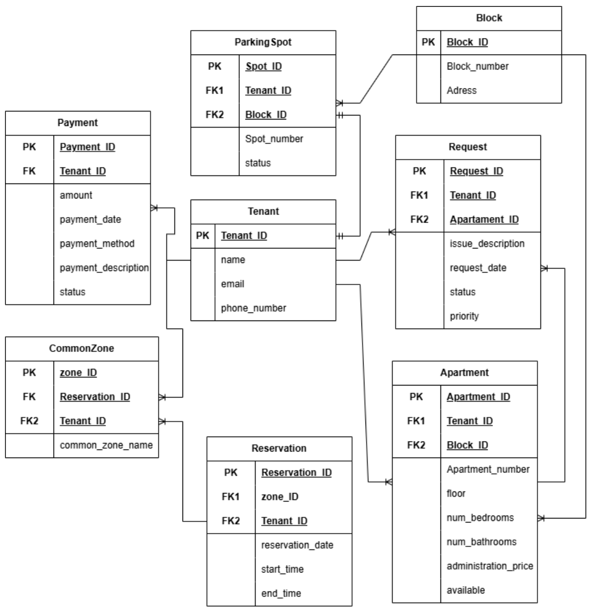


Figure 4. Second E.R draw.

4.9. get data-structure E-R M

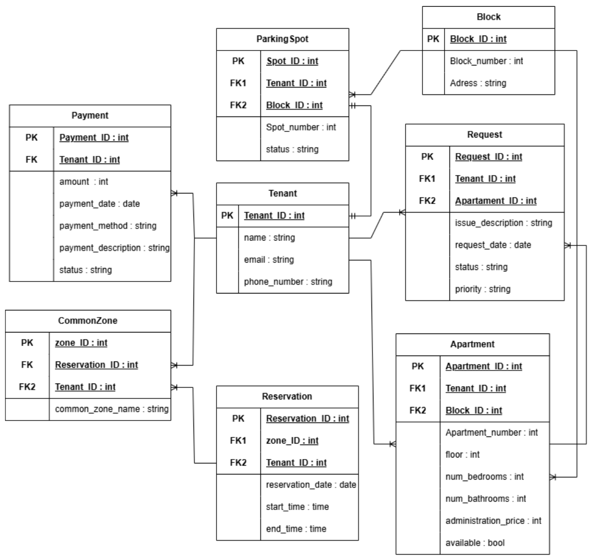
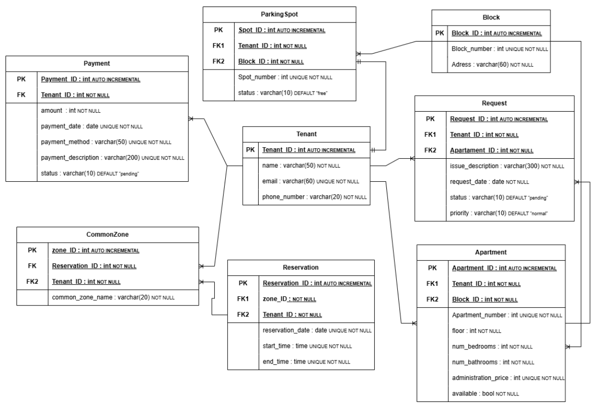


Figure 5. Data structure E.R draw.

4.10. define constraints and properties of data



ParkingSpot
PK Spot_ID : int AUTO INCREMENTAL
FK1 Tenant_ID : int NOT NULL
FK2 Block_ID : int NOT NULL
Spot_number : int UNIQUE NOT NULL
status : varchar(10) DEFAULT "free"
Block
PK Block_ID : int AUTO INCREMENTAL
Block_number : int UNIQUE NOT NULL
Adress : varchar(60) NOT NULL
Payment
PK Payment_ID : int AUTO INCREMENTAL
FK Tenant_ID : int NOT NULL
amount : int NOT NULL
payment_date : date UNIQUE NOT NULL
payment_method : varchar(50) UNIQUE NOT NULL
payment_description : varchar(200) UNIQUE NOT NULL
status : varchar(10) DEFAULT "pending"
Tenant
PK Tenant_ID : int AUTO INCREMENTAL
name : varchar(50) NOT NULL
email : varchar(60) UNIQUE NOT NULL
phone_number : varchar(20) NOT NULL
Request
PK Request_ID : int AUTO INCREMENTAL
FK1 Tenant_ID : int NOT NULL
FK2 Apartament_ID : int NOT NULL
issue_description : varchar(300) NOT NULL
request_date : date NOT NULL
status : varchar(10) DEFAULT "pending"
priority : varchar(10) DEFAULT "normal"
CommonZone
PK zone_ID : int AUTO INCREMENTAL
FK Reservation_ID : int NOT NULL
FK2 Tenant_ID : int NOT NULL
common_zone_name : varchar(20) NOT NULL
Reservation
PK Reservation_ID : int AUTO INCREMENTAL
FK1 zone_ID : NOT NULL
FK2 Tenant_ID : NOT NULL
reservation_date : date UNIQUE NOT NULL
start_time : time UNIQUE NOT NULL
end_time : time UNIQUE NOT NULL
Apartment
PK Apartment_ID : int AUTO INCREMENTAL
FK1 Tenant_ID : int NOT NULL
FK2 Block_ID : int NOT NULL
Apartment_number : int UNIQUE NOT NULL
floor : int NOT NULL
num_bedrooms : int NOT NULL
num_bathrooms : int NOT NULL
administration_price : int UNIQUE NOT NULL
available : bool NOT NULL

Figure 6. Define data propierties.