Workshop No.1 - Database Design

Daniel Alonso Chavarro Chipatecua
20241020066
September 15, 2024

This report presents the design from the ground up of the database of a complex apartment application, which includes the basic functionalities required by the main client, the user interview on the possible functionalities that could be added to the application and how it would be done in the database, and finally the process of designing the complete database for said application

1 User Stories

The first phase to be able to design the database is to know the functionalities required by the client and additional ones by users. For this, first an interview was conducted with 15 students asking what they would like to see in an apartment application, where they mentioned their point of view based on personal experiences with their neighbors or inconveniences with other people. These interviews were transformed into user stories where all the interviews are summarized, these user stories are in the format

As a (role), I want (action), so what (impact)

1. As a user, I would like to know what common spaces available and what days they are available, so I can schedule my appointment without problems.

2. As an administrator, I would like to know which tenants have not paid for services, so I know who to remind and keep the administration accounts in mind.

3. As a user, I would like to receive a notification that I have not yet paid for a service, because I usually forget things and need reminders.

4. As a tenant, I would like to receive a notification of the order when it arrives, because I want to know when it arrives and when I am busy, I can ensure that it arrived.

5. As a user, I would like to be notified of possible service cuts or repairs to something within the complex that involves me, so I can prepare and be informed.

6. As a tenant, I would like there to be the option to report situations that affect me in my apartment / block, because there may be conflicts that cannot be resolved by dialogue between those involved and a mediator is needed.

2 Technical considerations

The first thing to consider is the basic functionalities required by the client to design the application database, which are:

- Shows a list of all blocks and apartments.
- Receive payments for administration services and everything related.
- Make reservations for common spaces.

In addition, there are more functionalities according to the history of users based on their needs such as:

- Add the query of common spaces available by day and time to the make reservations for common spaces functionality.
- Functionality for the administrator to know which tenants have not paid for administration and related services.
- Functionality for users to receive notifications about possible service cuts (water, electricity, etc.), as well as repairs that may affect users.
- On the other hand, notification to the person of the arrival of their order.
- And finally, notifications to tenants who have not paid for common services.
- Functionality for tenants to report problematic situations.

3 Database design process

3.1 Define components

The components are the application modules; in this case they are:

- Blocks
- Common Spaces
- Order
- Notifications

- Outages and repairs

Apartments are not considered as a main module because any apartment is within an apartment block. On the other hand, common spaces must be another module because there are general common spaces and common spaces per block, and in that module the functionality of scheduling spaces and their respective schedule must be added, notifications and outages / repairs have other functionalities.

3.2 Define entities

With the components defined, we can identify main entities found from the components:

- Apartment
- User
- Block
- Common space
- Common service payment
- Incident report
- Notifications
- Outages / Repairs
- Delivery
- Space Reservation
- Entity.type (this will be a dictionary-like entity for options of any entity type)
- Entity.status (also like entity.type for options status to any entity)

3.3 Define attributes

Now that we have the main entities we can define their attributes

1. Apartment
    - apartment_id
    - apartment_number
2. User
    - User_id
    - Name
    - Phone

- Email
- User type (Administrator / Resident)
- Apartment

3. Block
    - Block_id
    - Name
    - Location
    - Number of floors (usually a tower, therefore it has n floors)
4. Common space
    - Common_space_id
    - Type of space (check if another entity can be added)
    - Description
    - Capacity
5. Common Service Payment:
    - Payment_id
    - Due_date
    - Amount
    - Status (paid / pending)
6. Incident report
    - Id
    - Description
    - Date and time of issue
    - Status (solved / pending)
7. Notifications
    - Notification_id
    - Type (outage / order / repair / service_payment)
    - Message
    - Date and time
    - Status (Received / not received)
8. Service Outages / Repair
    - id

- Type (outage/repair)
- Description
- Start date and time
- End date and time
- status

9. Delivery
    - v - Id_delivery
    - Description
    - Estimated arrival time
    - Status (Delivered/Not delivered)
10. Space Reservation
    - Reservation_id
    - Reservation date + time
    - Status (confirmed/pending/cancelled)
11. Entity.type
    - Id_type_entity
    - Description
12. Entity.status
    - Id_status_entity
    - Description

3.4 Relationship Table

Now I set the relationship of every entity

Entities	Apartment	User	Block	Common space	Common service payment	Incident report	Notifications	Outages / Repairs	Delivery	Space Reservation	Entity.type	status_Entity
Apartment		x	x					x				
User	x		x		x	x	x	x	x	x	x	
Block	x	x						x				
Common space										x	x	
Common service payment		x					x					
Incident report		x										
Notifications		x			x			x			x	x
Outages / Repairs	x	x	x	x			x				x	x
Delivery		x										x
Space Reservation		x		x								
Entity.type		x		x			x	x				
status_Entity							x	x	x			

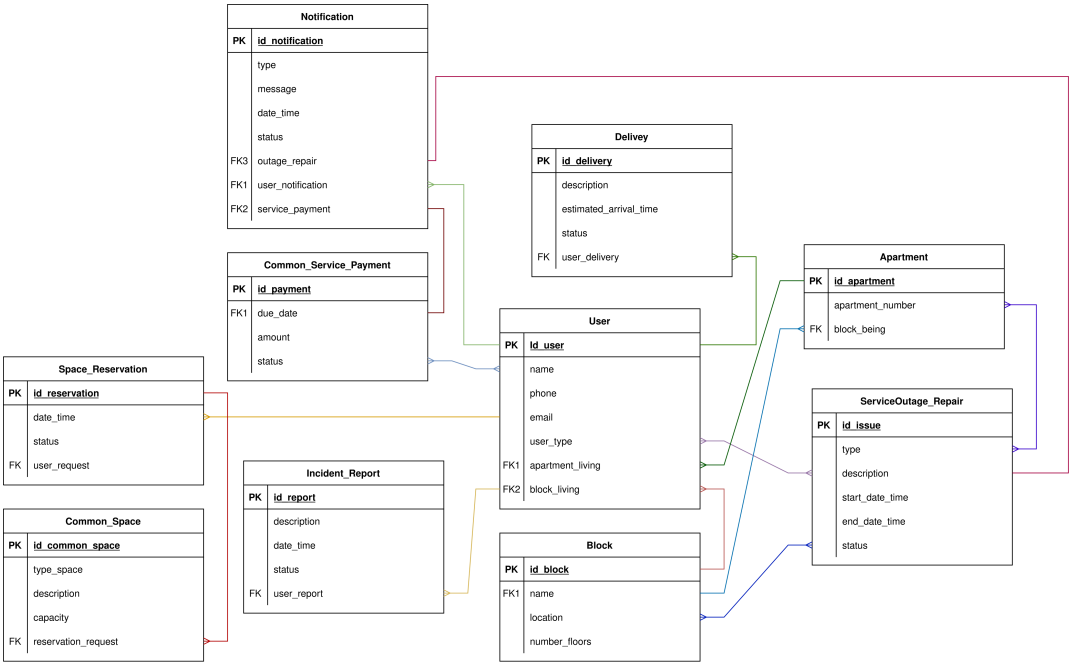
3.5 Define Relationship Types

Now I have all the relationships then assign the type of it, the method will be just coherent phrases which help me to understand the cardinality of the relationships

- A user belongs to an apartment and an apartment can have many users
- An apartment belongs to a block and a block can have many apartments
- An apartment can be affected by different services outages or repairs, and a service outage or repair can affect many apartments
- A user belongs to a block and a block can have several users
- A user can make many service payments, and a service must be paid by different users
- A user can have many incident reports, and an incident is created by one user
- A user can have many notifications and notifications are sent to one user (the orders are individual)
- A user can be affected by different services outages or repairs and a service outage or repair can affect many users
- A user can order many deliveries, and a delivery is only for one user
- A user can reserve several common spaces, and a common space is reserved by one user
- A user have one type of user and one type of user have many users
- A block can be affected by several services outages or repairs and a service outage or repair can affect many blocks
- A common space is reserved in a Space Reservation request and in a Space Reservation request many common spaces can be reserved
- A service payment is sent by a notification and a notification can only send one payment message
- A notification has as its only message the outage or repair of something and an outage or repair has a single notification to be sent to users
- A entity has one possible type (if it has the attribute type) and a type contains many entities.
- A entity has one possible status (if it has the attribute status) and a status contains many entities.

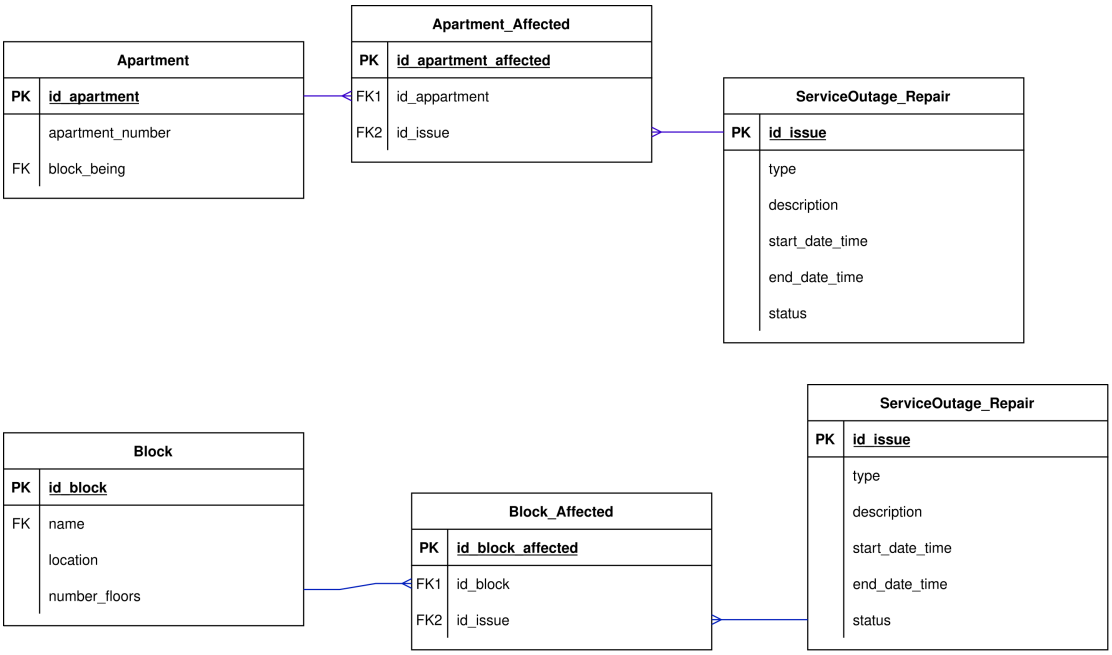
3.6 First Entity-Relationship Draw

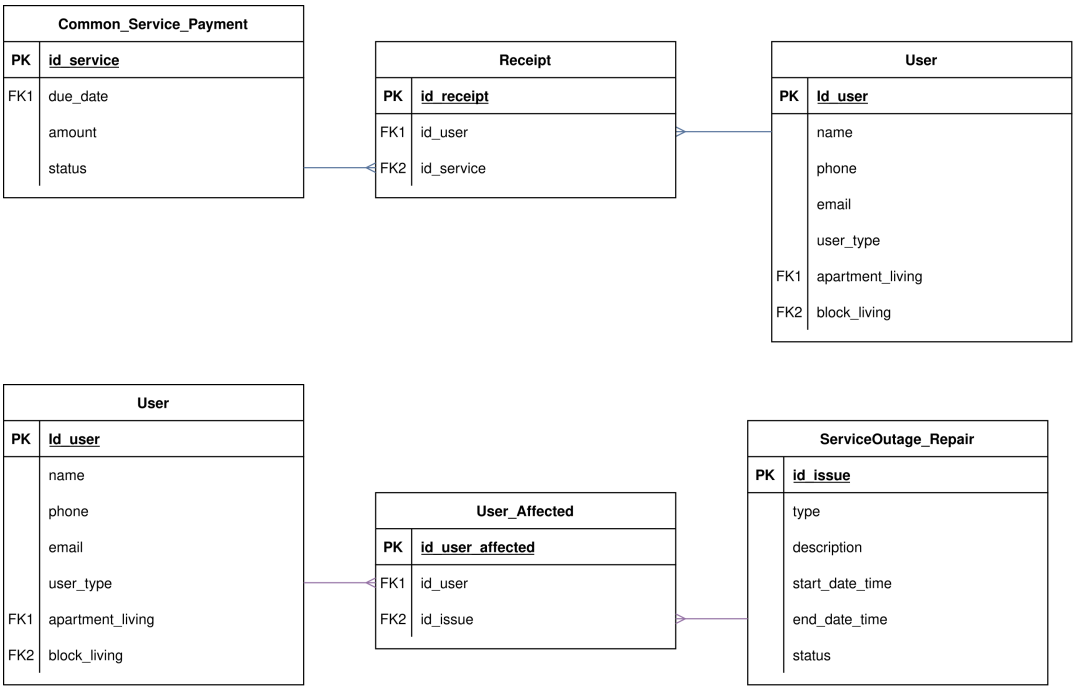
now I have all the entities, it's attributes and it's relationships I can do the first diagram



Here I didn't include the type and status because is not visible until the analysis of data structure

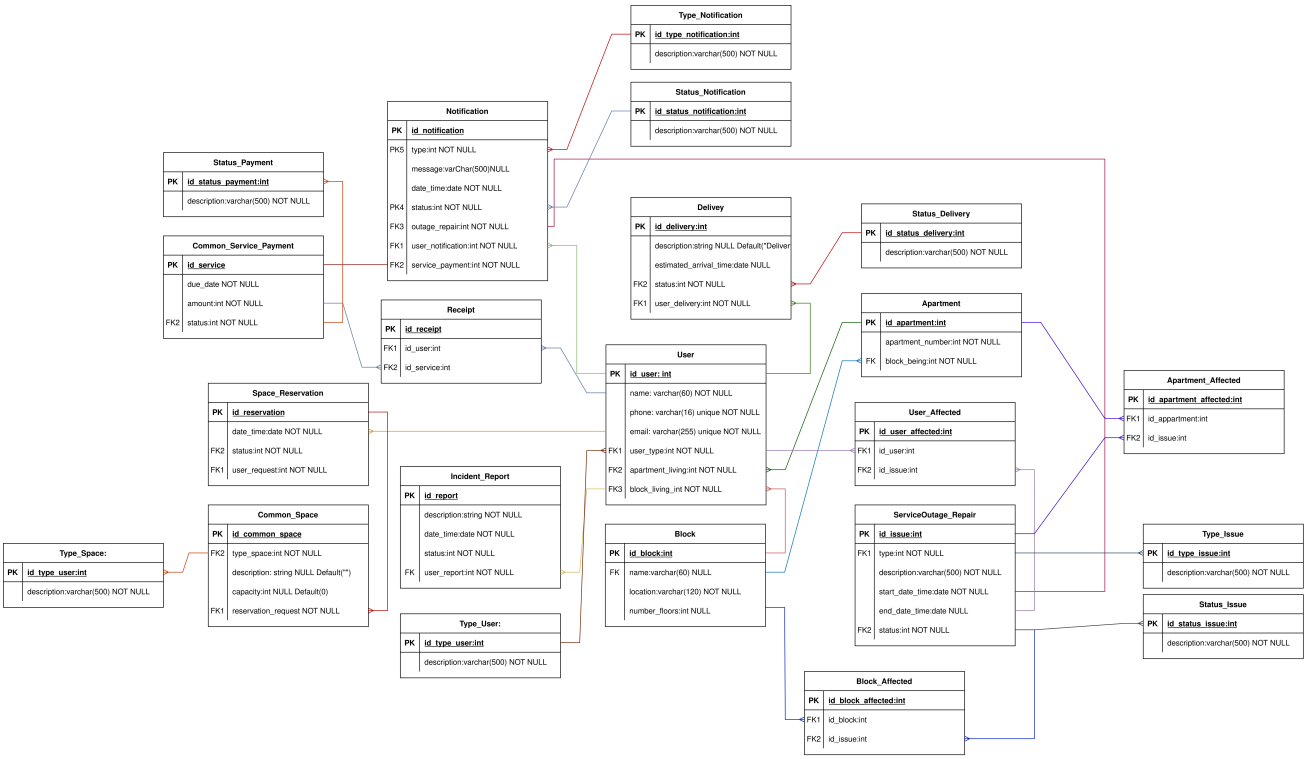
3.7 Split Many-to-Many Relationships





3.8 Defining Data Structure and Properties for Attributes

Finally we define the type of variable to the attributes and it's properties and the final result was this:


Type_Notification
PK id_type_notification:int
description:varchar(500) NOT NULL
Status_Notification
PK id_status_notification:int
description:varchar(500) NOT NULL
Notification
PK id_notification
PK5 type:int NOT NULL
message:varChar(500)NULL
date_time:date NOT NULL
PK4 status:int NOT NULL
FK3 outage_repair:int NOT NULL
FK1 user_notification:int NOT NULL
FK2 service_payment:int NOT NULL
Status_Payment
PK id_status_payment:int
description:varchar(500) NOT NULL
Common_Service_Payment
PK id_service
due_date NOT NULL
amount:int NOT NULL
FK2 status:int NOT NULL
Delivey
PK id_delivery:int
description:string NULL Default("Deliver
estimated_arrival_time:date NULL
FK2 status:int NOT NULL
FK1 user_delivery:int NOT NULL
Status_Delivery
PK id_status_delivery:int
description:varchar(500) NOT NULL
Receipt
PK id_receipt
FK1 id_user:int
FK2 id_service:int
Apartment
PK id_apartment:int
apartment_number:int NOT NULL
FK block_being:int NOT NULL
User
PK id_user: int
name: varchar(60) NOT NULL
phone: varchar(16) unique NOT NULL
email: varchar(255) unique NOT NULL
FK1 user_type:int NOT NULL
FK2 apartment_living:int NOT NULL
FK3 block_living_int NOT NULL
Apartment_Affected
PK id_apartment_affected:int
FK1 id_appartment:int
FK2 id_issue:int
Space_Reservation
PK id_reservation
date_time:date NOT NULL
FK2 status:int NOT NULL
FK1 user_request:int NOT NULL
User_Affected
PK id_user_affected:int
FK1 id_user:int
FK2 id_issue:int
Incident_Report
PK id_report
description:string NOT NULL
date_time:date NOT NULL
status:int NOT NULL
FK user_report:int NOT NULL
Common_Space
PK id_common_space
FK2 type_space:int NOT NULL
description: string NULL Default("")
capacity:int NULL Default(0)
FK1 reservation_request NOT NULL
Block
PK id_block:int
FK name:varchar(60) NULL
location:varchar(120) NOT NULL
number_floors:int NULL
ServiceOutage_Repair
PK id_issue:int
FK1 type:int NOT NULL
description:varchar(500) NOT NULL
start_date_time:date NOT NULL
end_date_time:date NULL
FK2 status:int NOT NULL
Type_Issue
PK id_type_issue:int
description:varchar(500) NOT NULL
Type_Space:
PK id_type_user:int
description:varchar(500) NOT NULL
Type_User:
PK id_type_user:int
description:varchar(500) NOT NULL
Status_Issue
PK id_status_issue:int
description:varchar(500) NOT NULL
Block_Affected
PK id_block_affected:int
FK1 id_block:int
FK2 id_issue:int