PR 1

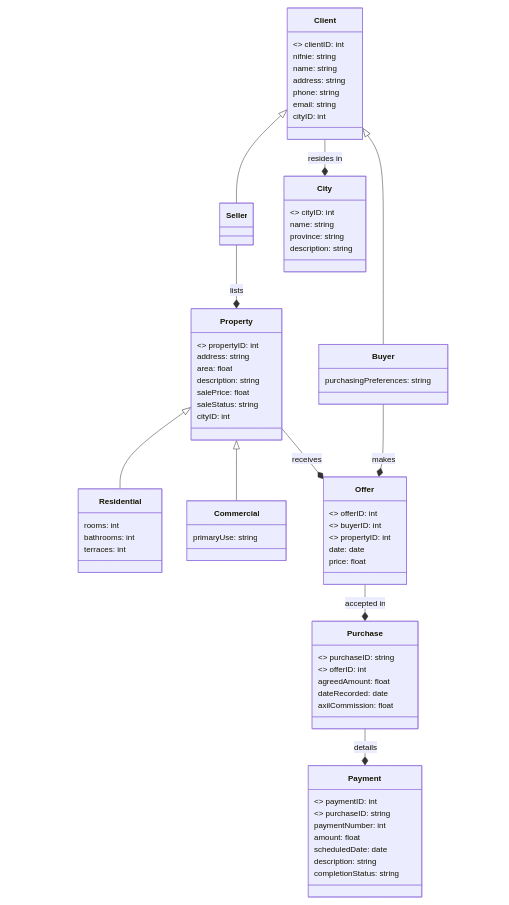
Database Design

Alejandro Pérez Bueno

Nov 25, 2024

Table of Contents

## Exercise 1

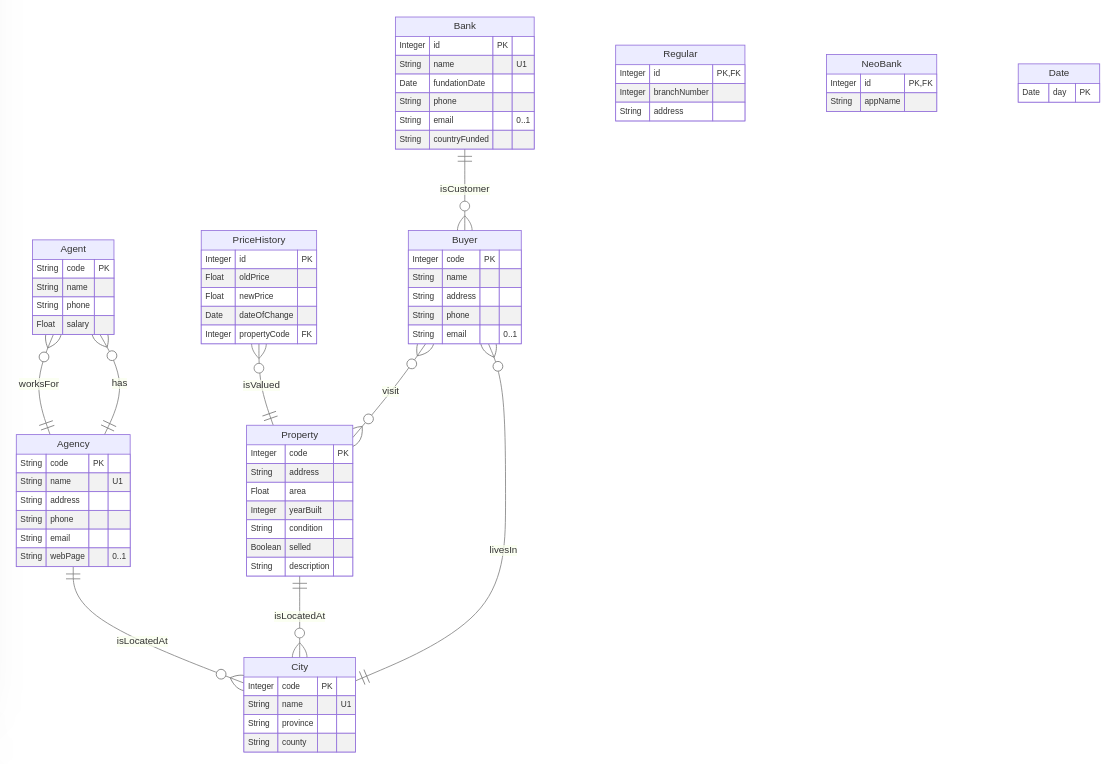


UML Class Diagram

### Semantic Assumptions and Constraints:

* **Client:**
  + clientID is the primary key (PK), ensuring each client has a unique ID.
  + nifnie could serve as an alternate key (AK) if it’s guaranteed to be unique for every client. This information isn’t explicitly stated.
  + cityID is a foreign key (FK) referencing the City entity, establishing the client’s city of residence.
* **Buyer/Seller:**
  + These are subtypes of Client, inheriting all client attributes.
  + Buyer has an additional attribute purchasingPreferences to record specific buying criteria.
  + Seller has no additional attributes based on the provided information.
* **City:**
  + cityID is the primary key, ensuring each city has a unique ID.
  + A constraint should be added to enforce uniqueness of city names within the same province. This is a requirement stated in your description, but not directly reflected in the schema as an attribute.
* **Property:**
  + propertyID is the primary key, ensuring each property has a unique ID.
  + cityID is a foreign key referencing the City entity, specifying the property’s location.
  + A constraint is added to ensure each property is listed by a seller. This relationship enforces that sellers must list at least one property.
* **Residential/Commercial:**
  + These are subtypes of Property, inheriting all property attributes.
  + Residential has additional attributes for rooms, bathrooms, and terraces.
  + Commercial has an additional attribute primaryUse to describe the property’s main commercial activity.
* **Offer:**
  + offerID is the primary key, ensuring each offer has a unique ID.
  + buyerID and propertyID are foreign keys referencing Buyer and Property respectively, linking an offer to a specific buyer and property.
  + A constraint needs to be added to prevent multiple offers on the same property by the same buyer on the same day. This constraint is beyond the basic structure of the schema and would need additional implementation logic.
* **Purchase:**
  + purchaseID is the primary key, ensuring each purchase has a unique ID.
  + offerID is a foreign key referencing Offer, connecting the purchase to the accepted offer.
* **Payment:**
  + paymentID is the primary key, ensuring each payment has a unique ID.
  + purchaseID is a foreign key referencing Purchase, linking the payment to a specific purchase.
  + paymentNumber tracks the sequence of payments within a purchase.

## Exercise 2



UML Diagram

## Exercise 3

### 1. Functional Dependencies in ClientInterest

Based on the provided description of the ClientInterest table, the following functional dependencies can be identified:

* **Client\_ID → {Client\_NIF, Client\_Name, Client\_Phone}**: A client’s ID uniquely determines their tax identification number, name, and phone number.
* **Property\_ID → {Address, City, Province, Sale\_Price, Agency}**: A property’s ID uniquely determines its address, city, province, sale price, and the managing agency.
* **{Client\_ID, Property\_ID} → {Proposed\_Price}**: The combination of client ID and property ID determines the proposed price offered by that client for that specific property.

### 2. Possible Primary Key

A suitable primary key for the ClientInterest relation is **{Client\_ID, Property\_ID}**. This composite key ensures that each tuple in the table represents a unique combination of a client and a property they are interested in.

* **Uniqueness**: The combination of Client\_ID and Property\_ID guarantees uniqueness because a client can be interested in multiple properties and a property can have multiple interested clients.
* **Minimality**: Both attributes are necessary for uniqueness, as using only one wouldn’t be sufficient to identify a specific client’s interest in a particular property.

### 3. Anomalies

The current design of the ClientInterest table can lead to several anomalies:

**a) Insertion Anomaly**

* **Problem**: Adding a new client who hasn’t yet shown interest in any property is impossible.
* **Reason**: The primary key requires both Client\_ID and Property\_ID. Without an existing property to link to, a new client entry cannot be created.

**b) Update Anomaly**

* **Problem**: Changing the sale price of a property requires updating multiple tuples.
* **Reason**: The Sale\_Price attribute is dependent on Property\_ID. If a property’s price changes, all tuples related to that property need to be updated. The number of updates would equal the number of clients interested in that property.

**c) Deletion Anomaly**

* **Problem**: Deleting a client’s interest in a property could lead to loss of information about the property itself.
* **Reason**: If a client’s interest is the only tuple associated with a particular property, deleting that tuple removes all data about the property from the table.

### 4. Normal Form and BCNF Compliance

**Current Normal Form**

The ClientInterest relation is in **Second Normal Form (2NF)**.

* **1NF Compliance**: All attributes are atomic (no repeating groups or multi-valued attributes).
* **2NF Compliance**: No non-prime attribute (attributes not part of any candidate key) is dependent on a part of a candidate key.

**BCNF Violation**

The relation is **not in Boyce-Codd Normal Form (BCNF)** because it violates the rule that every determinant must be a candidate key.

* **Violation**: The dependencies Client\_ID → {Client\_NIF, Client\_Name, Client\_Phone} and Property\_ID → {Address, City, Province, Sale\_Price, Agency} exist, but neither Client\_ID nor Property\_ID is a candidate key on its own. The composite key {Client\_ID, Property\_ID} is the only candidate key.

**Solution to Achieve BCNF**

To make the relation BCNF-compliant, the table needs to be decomposed into three separate relations:

**1. Client Table (1NF)**

* **Primary Key**: Client\_ID

**2. Property Table (1NF)**

* **Primary Key**: Property\_ID

**3. ClientInterest Table (1NF)**

* **Primary Key**: {Client\_ID, Property\_ID}
* **Foreign Keys**: Client\_ID referencing the Client table, Property\_ID referencing the Property table.

**Justification**

This decomposition achieves BCNF:

* Each relation is in 1NF (and consequently 2NF) as all attributes are atomic.
* All determinants in each relation are candidate keys: Client\_ID in the Client table, Property\_ID in the Property table, and {Client\_ID, Property\_ID} in the ClientInterest table.

**Benefits of Decomposition**

* Eliminates redundancy: Client and property information is stored only once.
* Solves anomalies:
  + New clients can be added without needing a property link.
  + Property price updates only affect one tuple in the Property table.
  + Deleting client interest doesn’t affect property data.