Khulna University of Engineering and Technology

PropertiFy

Real Estate Management System

Oracle SQL Database Project

May 8, 2024

Name : Ahmed Nur E Safa
Roll : 2007114
Year : 3rd
Semester : 1st
Course : CSE-3110

Project Introduction

Description:

PropertiFy is a real estate management project which's goal is to use Oracle SQL to create a comprehensive real estate management system. Efficient real estate property management, including listing, purchasing, selling, and renting, will be made easier by this technology. Real estate brokers, property owners, and prospective tenants or buyers will be able to interact with the database through an easy-to-use interface.

Objectives:

This system will facilitate efficient management of real estate properties, including listing, buying, selling, and details information of the owner of the properties. It will provide a user-friendly interface for real estate agents, property owners, and potential buyers/sellers to interact with the database.

Database Design and overview:

Overview of the Database Schema:

The PropertiFy database is structured around several key tables designed to encompass all necessary real estate related data. The main tables include:

I. Admin: Stores detailed information about Admin panel, such as username, password, email, phone number.

II. Client: Contains data about clients, including full name, email, phone number, and budget.

III. FlatDetails: Includes information of Flats, such as floor number,number of bedrooms, number of bathrooms.

IV. PropertyDetails: Records details about properties, such as address, type (Lands or Flats), size and price.

V. TransactionDetails: Captures information about transactions made by clients.

Table Relations:

The relationships between the tables are primarily defined by foreign keys that ensure data integrity and relational connections. For example:

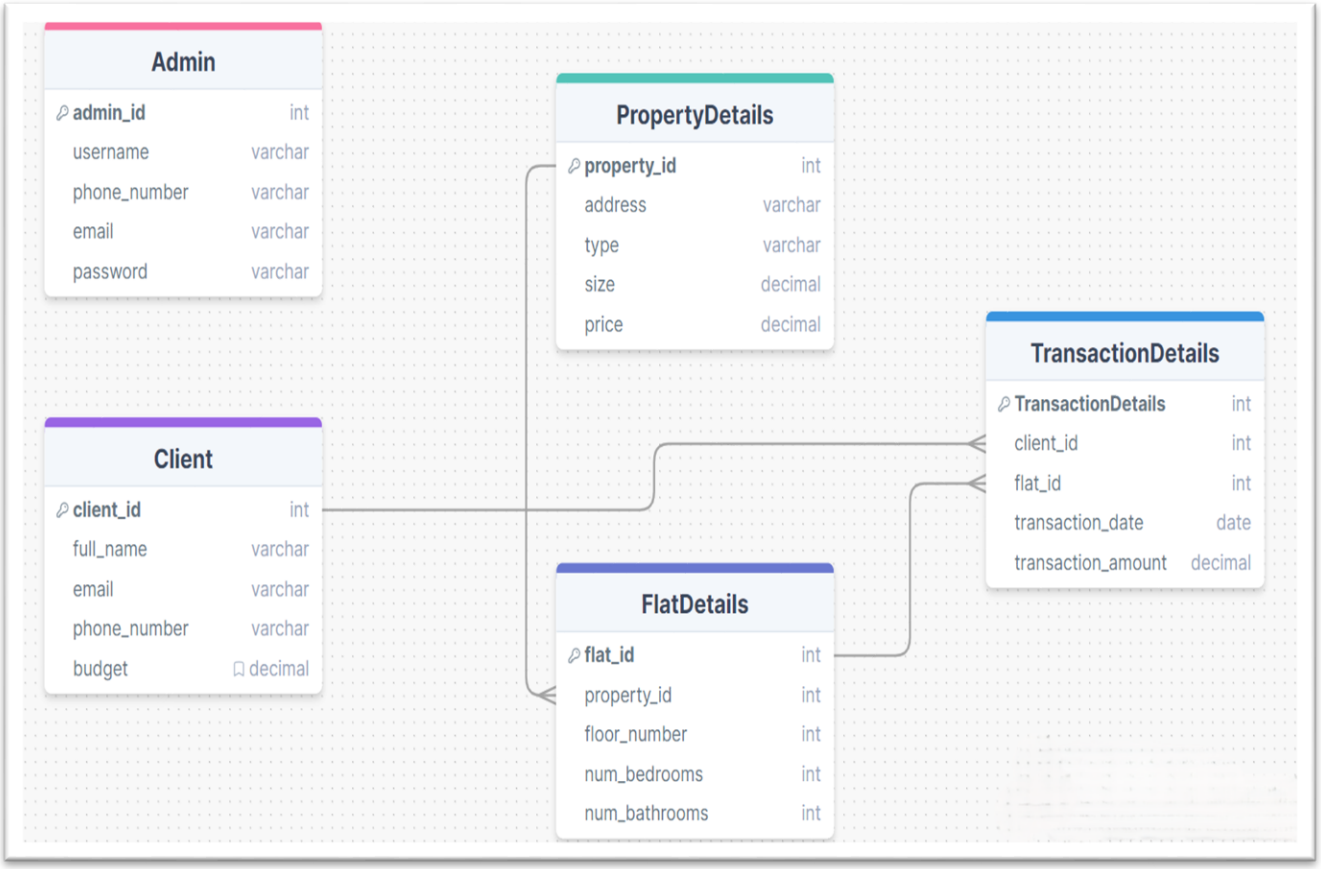
```
-- Transaction Table
create table TransactionDetails (
    transaction_id int primary key,
    client_id int,
    flat_id int,
    transaction_date DATE,
    transaction_amount decimal(15, 2),
    foreign  key (client_id) references Client(client_id),
    foreign  key (flat_id) references FlatDetails(flat_id)
);

-- Client Table
```

```sql
create table Client (
    client_id int primary key,
    full_name varchar(100),
    email varchar(100),
    phone_number varchar(20),
    budget decimal(15, 2)
);

-- Flat details Table
create table FlatDetails (
    flat_id int primary key,
    property_id int,
    floor_number int,
    num_bedrooms int,
    num_bathrooms int,
    foreign  key (property_id) references PropertyDetails(property_id)
);

-- Admin Table
create table Admin (
    admin_id int primary key,
    username varchar(50),
    password varchar(50),
    email varchar(100),
    phone_number varchar(20)
);

-- Property details Table
create table PropertyDetails (
    property_id int primary key,
    address varchar(255),
    type varchar(50),
    size decimal(15, 2),
    price decimal(15, 2)
);
```

Redundant Data and Anomalies:

In a database, redundant data is information that is kept in several records or tables. Data discrepancies and inefficient storage might result from this redundancy. Redundant data can lead to anomalies, including insertion, update, and deletion anomalies, which can cause inconsistencies in data management activities. By reducing the likelihood of anomalies and removing redundant data, normalization procedures help to maintain consistency and integrity of the data.

Entity-Relationship Diagram (E-R Diagram)



SQL Queries and Functionality

The PropertiFy project relies heavily on SQL queries and capabilities to allow users to efficiently interact with the database. These are a few crucial SQL queries and features for the system:

Create (INSERT) operations:

Insert a new client:

Insert into Client (client_id, full_name, email, phone_number, budget) values (1, 'sf ahm', 'sf.ahm@gmail.com', '12345678', 50000);

Insert a new property:

Insert into PropertyDetails (property_id, address, type, size, price) values (1, '123 Main St', 'House', 2000, 250000);

Read operations:

select full_name, email from Client where client_id in (select client_id from TransactionDetails where flat_id in (select flat_id from FlatDetails where property_id in (select property_id from PropertyDetails where size>1500)));

Above query selects full name and email addresses of clients who have made transactions for properties with a size larger than 1500 square feet.

Update (UPDATE) operations:

update Client set budget = 550000 where client_id = 1;

Delete (DELETE) operations:

```
delete from Client where client_id = 1;
```

Implementation of Advanced SQL Features

Using subqueries:

select full_name, email, budget from Client where budget > (select avg(budget) from Client);

Above query finds clients who have a budget greater than the average budget.

Using joins:

Select t.transaction_id, c.full_name as client_name, p.address as property_address, t.transaction_date, t.transaction_amount from TransactionDetails t join Client c on t.client_id = c.client_id join FlatDetails f on t.flat_id = f.flat_id join PropertyDetails p on f.property_id = p.property_id;

Above query retrieves the details of transactions along with the corresponding client and property details.

Views:

Create view FlatPropertyDetails as select f.flat_id, f.property_id, p.address, p.type, f.floor_number, f.num_bedrooms, f.num_bathrooms from FlatDetails f jopin PropertyDetails p on f.property_id = p.property_id;

Above query create a view to simplify retrieving flat details along with the corresponding property details.

Targets

Real Estate Agents:

Real estate agents will use the system to manage property listings, schedule viewings, communicate with clients, track leads, and facilitate transactions.

Property Owners:

Property owners will utilize the system to list their properties for sale or rent, monitor listing performance, receive inquiries from potential buyers/renters, and track transaction progress.

Conclusion

With its structured design and support for advanced SQL capabilities, the PropertiFy database project provides a solid foundation for a real estate management system; nonetheless, implementation and operation require careful assessment of the associated advantages and downsides. The system may efficiently support real estate transactions and enable the management of properties and clients with the right planning, optimization, and upkeep. Creative methods were used to address the problems, such as the application of strict data validation procedures and database speed improvement.

Pros:

- Structured Data Management: By keeping the tables standardized and well-organized, the tables help to reduce redundancy and promote data integrity.

- Scalability: An increasing number of real estate transactions and property listings can be supported by the schema.

- Flexibility: It makes it simple to expand and change to accommodate changing business needs.

- Performance: Effective data retrieval can be ensured by proper indexing and optimization, which can improve query performance.

- Security: To prevent orphaned records and guarantee data consistency, foreign key constraints enforce referential integrity.

Cons:

- Complexity: Advanced SQL features have the potential to add complexity, necessitating thorough preparation and specialized knowledge for setup and upkeep.

- Overhead: During data modification activities, indexing and stored procedures may create extra overhead that affects database performance.

- Data Redundancy: Even with a normalized schema, data redundancy can still occur, particularly if it is not well maintained.

Thank you