Group No. 01

Group Members

1. Abdul Rahman (20-28)
2. Ali Riaz (20-01)
3. Ali Shair (20-02)
4. Muhammad Yousaf (20-35)
5. Muhammad Adnan (20-36)
6. Muahmmad Behzad (19-39)

Accommodation Management System

An Integrated Solution for Efficient Accommodation Management

**Accommodation Management System**

# 1. Introduction:

The Accommodation Management System is a comprehensive database solution designed to streamline the management of accommodations, agreements, proposals, requests, notifications, users, settings, and websites. It serves as a centralized platform for managing various aspects related to accommodations, including rental agreements, user interactions, notifications, and system configurations.

## **1.1 Key Features:**

1. **Accommodations Management:** The system allows for the management of various types of accommodations, including details such as location, facilities, descriptions, pictures, and status.
2. **Agreements and Proposals:** Users can create and manage rental agreements, proposals, and offers, specifying terms, prices, check-in/out dates, and additional information.
3. **User Interaction:** It facilitates interactions between landlords, tenants, and additional users, managing user types, contact details, signatures, and notifications.
4. **Notifications:** Users receive notifications regarding important events, messages, and updates within the system.
5. **Requests Management:** Users can create, track, and manage accommodation requests, specifying requirements, preferences, and statuses.
6. **Settings Configuration:** System administrators can configure system settings, preferences, and global options through the settings module.
7. **Websites Management:** The system allows for the management of multiple websites, providing descriptions, city-specific information, and global settings.

# 2. Gather Requirements:

The process of gathering requirements for the Accommodations Management System plays a crucial role in defining the scope, functionalities, and data structures of the project.

## **2.2 Data Requirements**

1. **Accommodations Data:** Capture structured data about accommodations, including location details (city, address, latitude, longitude), accommodation type, facilities, number of beds, bedrooms, rooms, toilets, showers, and termination period.
2. **Agreements Data:** Store information about rental agreements, including agreement ID, manager ID, location details (city, address, latitude, longitude), price type, price, check-in/out dates, facilities, and additional information.
3. **Users Data:** Maintain user data such as user ID, role, name, email, phone, address, password, language, and user settings.
4. **Requests Data:** Track accommodation requests with data such as request ID, customer ID, manager ID, location details (city, address, latitude, longitude), check-in/out dates, beds, sharing preferences, furnished status, and additional details.
5. **Notifications Data:** Manage system notifications with user ID, notification content, notification status, and timestamp information.
6. **Websites Data:** Store website information, including website ID, website name, global status, city-specific details, descriptions, and creation timestamps.
7. **Settings Data:** Maintain system-wide settings and configurations, including settings ID, setting values, and configuration details.

## **2.2 Functional Requirements:**

1. **Data Entry:** Provide interfaces for users to input and update accommodations, agreements, users, requests, notifications, websites, and system settings.
2. **Data Retrieval:** Enable users to retrieve data based on various criteria, such as location, dates, user ID, and status.
3. **Data Update:** Allow authorized users to update and modify existing data records, ensuring data accuracy and consistency.
4. **Data Query:** Support query functionalities for users to perform complex searches and retrieve specific information from the database.
5. **Integration:** Facilitate integration with other systems or APIs for data exchange, synchronization, and seamless functionality across platforms.
6. **User Management:** Provide functionalities for user authentication, authorization, roles management, and user profile customization.
7. **Notifications Management:** Handle system-generated notifications, user alerts, and notification preferences management.
8. **Settings Configuration:** Allow administrators to configure system-wide settings, preferences, and global options through an administrative interface.

# 3. Database Design:

The database design for the Accommodations Management System involves structuring the database tables, defining relationships, and optimizing data storage for efficient system functionality.

Here is structure of some tables.

-- Table structure for table accommodations

CREATE TABLE tbl\_accommodations (

  id *NUMBER* PRIMARY KEY,

  city *VARCHAR2*(100) NOT NULL,

  address *VARCHAR2*(255) NOT NULL,

  lat *VARCHAR2*(20) NOT NULL,

  lng *VARCHAR2*(20) NOT NULL,

  beds *NUMBER*,

  bedrooms *NUMBER*,

  rooms *NUMBER*,

  toilets *NUMBER*,

  showers *NUMBER*,

  termination\_period *NUMBER*,

  accommodation\_type *NUMBER*,

  facilities *VARCHAR2*(100),

  description CLOB,

  pictures CLOB,

  status *NUMBER* DEFAULT 0 NOT NULL,

  landlord\_id *NUMBER*,

  added\_by *NUMBER*,

  created\_at *TIMESTAMP* DEFAULT CURRENT\_TIMESTAMP NOT NULL

);

-- Table structure for table agreements

CREATE TABLE tbl\_agreements (

  id *VARCHAR2*(20) PRIMARY KEY,

  manager\_id *NUMBER*,

  city *VARCHAR2*(100),

  address *VARCHAR2*(255),

  lat *VARCHAR2*(20),

  lng *VARCHAR2*(20),

  price\_type *VARCHAR2*(15),

  price *NUMBER*,

  beds *NUMBER*,

  check\_in\_date *VARCHAR2*(20),

  check\_out\_date *VARCHAR2*(20),

  facilities *VARCHAR2*(100),

  more\_info CLOB,

  files CLOB,

  status *NUMBER* DEFAULT 0 NOT NULL,

  added\_by *NUMBER*,

  created\_at *TIMESTAMP* DEFAULT CURRENT\_TIMESTAMP NOT NULL

);

Inserting data into tables.

-- Inserting data into table tbl\_accommodations

INSERT INTO tbl\_accommodations VALUES(1, 'Domsjö', 'Syskonhemsvägen 10, Domsjö, Sweden', '63.27325440000001', '18.6654404', 5, 4, 5, 1, 1, 1, 0, 'p,i,f,w,d,k', NULL, NULL, 1, 23, 17, TO\_TIMESTAMP('2024-02-14 08:03:34', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO tbl\_accommodations VALUES(2, 'Husum', 'KASAMARK 119, Husum, Sweden', '63.3730105', '19.0644589', 7, 5, 6, 1, 1, 1, 0, 'p,i,f,w,d,k', '.', NULL, 1, 32, 25, TO\_TIMESTAMP('2024-02-14 08:14:29', 'YYYY-MM-DD HH24:MI:SS'));

-- Inserting data into table tbl\_agreements

INSERT INTO tbl\_agreements VALUES('A8832', 1, 'Strömstad', 'Daftö Resort AB, Strömstad, Sweden', '58.90395720000001', '11.199898', 'm', 424, NULL, '17-02-2024', '09-02-2024', 'p,i,w', NULL, NULL, 0, 4, TO\_TIMESTAMP('2024-02-05 17:56:15', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO tbl\_agreements VALUES('D6368', 1, 'Byxelkrok', 'Tokenäs Camping, Enerumsvägen, Byxelkrok, Sweden', '57.3203988', '16.9992403', 'b', 346, 12, '20-09-2023', '19-10-2023', 'f,w,d', NULL, NULL, 1, 23, TO\_TIMESTAMP('2024-02-14 13:20:59', 'YYYY-MM-DD HH24:MI:SS'));

Creating indexes on tables

-- Index on tbl\_accommodations

CREATE INDEX idx\_accommodations\_city ON tbl\_accommodations(city);

CREATE INDEX idx\_accommodations\_beds ON tbl\_accommodations(beds);

CREATE INDEX idx\_accommodations\_status ON tbl\_accommodations(status);

-- Index on tbl\_agreements

CREATE INDEX idx\_agreements\_city ON tbl\_agreements(city);

CREATE INDEX idx\_agreements\_price ON tbl\_agreements(price);

CREATE INDEX idx\_agreements\_status ON tbl\_agreements(status);

Adding foreign key constraints on some tables

ALTER TABLE tbl\_accommodations

ADD FOREIGN KEY (landlord\_id) REFERENCES tbl\_users (id) ON DELETE SET NULL;

ALTER TABLE tbl\_accommodations

ADD FOREIGN KEY (added\_by) REFERENCES tbl\_users (id) ON DELETE SET NULL;

ALTER TABLE tbl\_agreements

ADD FOREIGN KEY (manager\_id) REFERENCES tbl\_users (id) ON DELETE SET NULL;

ALTER TABLE tbl\_agreements

ADD FOREIGN KEY (added\_by) REFERENCES tbl\_users (id) ON DELETE SET NULL;

# 4. Queries and Reports

In the Accommodations Management System, SQL queries play a pivotal role in retrieving and manipulating data stored in the database tables. These queries can be used to generate reports that provide valuable insights into various aspects of the system. Here are some examples of SQL queries and reports relevant to the project:

1. Retrieve Accommodation Information:

SELECT \* FROM tbl\_accommodations WHERE city = 'Strömstad';

1. Generate Agreements Report:

SELECT id, city, address, price, check\_in\_date, check\_out\_date

FROM tbl\_agreements

WHERE status = 1;

1. Count Accommodations by Type:

SELECT accommodation\_type, COUNT(\*) AS total\_accommodations

FROM tbl\_accommodations

GROUP BY accommodation\_type;

1. Accommodation Details View

CREATE OR REPLACE VIEW vw\_accommodation\_details AS

SELECT a.\*, u.name AS landlord\_name, u.email AS landlord\_email

FROM tbl\_accommodations a

JOIN tbl\_users u ON a.landlord\_id = u.id;

1. Request Log Details View

CREATE OR REPLACE VIEW vw\_request\_log\_details AS

SELECT rl.id, rl.request\_id, rl.user\_id, rl.log\_type, rl.log\_data, rl.created\_at, u.name AS user\_name, u.email AS user\_email

FROM tbl\_request\_logs rl

JOIN tbl\_users u ON rl.user\_id = u.id;

# 5. Scalability and Performance

To ensure scalability and optimal performance of the database, consider the following best practices:

* Implement efficient indexing on frequently queried columns (e.g., id, city) to speed up search operations.
* Utilize materialized views for complex queries to precompute and store results, reducing query execution time.
* Regularly monitor database performance metrics (e.g., query response times, resource utilization) and optimize queries or database configurations as needed.

# 6. Backup and Recovery

To safeguard data integrity and recover from potential data loss scenarios, adopt robust backup and recovery strategies:

* Perform regular full backups of the database to capture all data.
* Implement incremental backups to capture changes since the last full backup, minimizing backup duration and storage requirements.
* Store backup copies securely, preferably in off-site locations, to protect against disasters or system failures.

**Suggestions for Database Management:**

* Conduct regular database maintenance tasks such as index reorganization, statistics updates, and database reorganizations to ensure optimal performance.
* Implement data validation checks and constraints to maintain data accuracy and integrity.
* Consider implementing a data archival strategy for historical data to optimize database performance and storage utilization.

# 7. Conclusion

Key aspects of effective database management in the Accommodations Management System include efficient querying and reporting, scalability and performance optimization, robust backup and recovery procedures, and proactive database maintenance practices. By adhering to best practices and leveraging SQL capabilities effectively, the database can support the system's functionalities and data management requirements seamlessly.

Good luck with out project!