CSE 3110: Database Systems Laboratory

Project name: Task Manager Oracle SQL Database Project

Submitted By

Ekramul Alam

Roll: **2007071**

Year: 3rd, Term: 1st

Introduction

Task Manager is a comprehensive system designed to enhance productivity and streamline task tracking in organizational and individual settings. This system utilizes a robust Oracle SQL database to manage tasks, activities, user interactions, and metadata associated with various tasks. A well-structured database is essential for efficient task management, offering capabilities for organizing, scheduling, and tracking tasks effectively. It supports real-time updates and integrations, allowing for enhanced collaboration, productivity, and decision-making within teams.

Project Objectives

1.To Design a Relational Database Schema for Task Management:

Develop a relational database schema that organizes and stores data related to tasks, users, activities, and tags effectively. This schema facilitates easy data integration, ensures consistency, and supports complex queries for task retrieval and management.

2.To Implement SQL Queries for Dynamic Data Handling:

Implement a range of SQL queries to manage and retrieve data, enabling functionalities such as task creation, updates, and dynamic searches based on user-defined criteria.

Database Design

Design Philosophy: The design is centered around ensuring data integrity, reducing redundancy, and facilitating easy data access and management. The use of foreign keys and constraints ensures the relational integrity of the data across various tables.

Overview of the Database Schema:

The Task Manager database is structured around several key tables designed to manage all necessary data related to task management:

- Users: Stores detailed information about system users, including authentication details and personal information.
- Tasks: Contains data about tasks assigned to users, along with metadata regarding the task's status, timeline, and associated content.
- Activities: Tracks detailed activities related to tasks, providing a log of actions taken by users.
- Tags: Manages categorization of tasks through tags, facilitating filtering and searching of tasks based on their attributes.
- Task Tags: A junction table that implements a many-to-many relationship between tasks and tags.
- Comments: Allows users to add comments to tasks and activities, fostering communication and collaboration.

Rationale Behind the Design Decisions:

The design of the database schema focuses on efficiently managing task-related data. Each table serves a specific segment of task data, ensuring the database can handle diverse queries—from task progress tracking to user activity logs.

Table relations:

This section outlines the SQL commands used to create and configure the tables within the database. Each command includes a brief explanation of the table's purpose and its relationships with other tables.

1. Users table:

```
CREATE TABLE users (
  id NUMBER(12) NOT NULL,
  role_id NUMBER(4) NOT NULL,
  first_name VARCHAR2(50),
  middle_name VARCHAR2(50),
```

```
  last_name VARCHAR2(50),
  username VARCHAR2(50),
  mobile VARCHAR2(15),
  email VARCHAR2(50),
  password_hash VARCHAR2(32) NOT NULL,
  registered_at DATE NOT NULL,
  last_login DATE,
  intro VARCHAR2(4000),
  profile CLOB,
  CONSTRAINT pk_user PRIMARY KEY (id),
  CONSTRAINT uq_username UNIQUE (username),
  CONSTRAINT uq_mobile UNIQUE (mobile),
  CONSTRAINT uq_email UNIQUE (email)
);
```

2. Tasks table:

```
CREATE TABLE tasks (
  id NUMBER(12) NOT NULL,
  user_id NUMBER(12) NOT NULL,
  created_by NUMBER(12) NOT NULL,
  updated_by NUMBER(12) NOT NULL,
  title VARCHAR2(512) NOT NULL,
  description VARCHAR2(2048),
  status NUMBER(4) DEFAULT 0 NOT NULL,
  hours NUMBER DEFAULT 0,
  created_at DATE NOT NULL,
  updated_at DATE,
  planned_start_date DATE,
  planned_end_date DATE,
  actual_start_date DATE,
  actual_end_date DATE,
  content CLOB,
  CONSTRAINT pk_task PRIMARY KEY (id),
  CONSTRAINT fk_task_user FOREIGN KEY (user_id) REFERENCES
users(id) ON DELETE CASCADE,
  CONSTRAINT fk_task_creator FOREIGN KEY (created_by) REFERENCES
users(id) ON DELETE CASCADE,
  CONSTRAINT fk_task_updater FOREIGN KEY (updated_by) REFERENCES
users(id) ON DELETE CASCADE
);
```

3. Tags table:

```
CREATE TABLE tags (
  id NUMBER(12) NOT NULL,
  title VARCHAR2(75) NOT NULL,
  slug VARCHAR2(100) NOT NULL,
  CONSTRAINT pk_tag PRIMARY KEY (id)
);
```

4. Task_tags table:

```
CREATE TABLE task_tags (
  task_id NUMBER(12) NOT NULL,
  tag_id NUMBER(12) NOT NULL,
  CONSTRAINT pk_task_tag PRIMARY KEY (task_id, tag_id),
  CONSTRAINT fk_task_tag_task FOREIGN KEY (task_id) REFERENCES
tasks(id),
  CONSTRAINT fk_task_tag_tag FOREIGN KEY (tag_id) REFERENCES
tags(id)
);
```

5. Comments table:

```
CREATE TABLE comments (
  id NUMBER(12) NOT NULL,
  task_id NUMBER(12) NOT NULL,
  activity_id NUMBER(12),
  title VARCHAR2(100) NOT NULL,
  created_at DATE NOT NULL,
  updated_at DATE,
  content CLOB,
  CONSTRAINT pk_comment PRIMARY KEY (id),
  CONSTRAINT fk_comment_task FOREIGN KEY (task_id) REFERENCES
tasks(id) ON DELETE CASCADE,
  CONSTRAINT fk_comment_activity FOREIGN KEY (activity_id)
REFERENCES activities(id) ON DELETE CASCADE
);
```

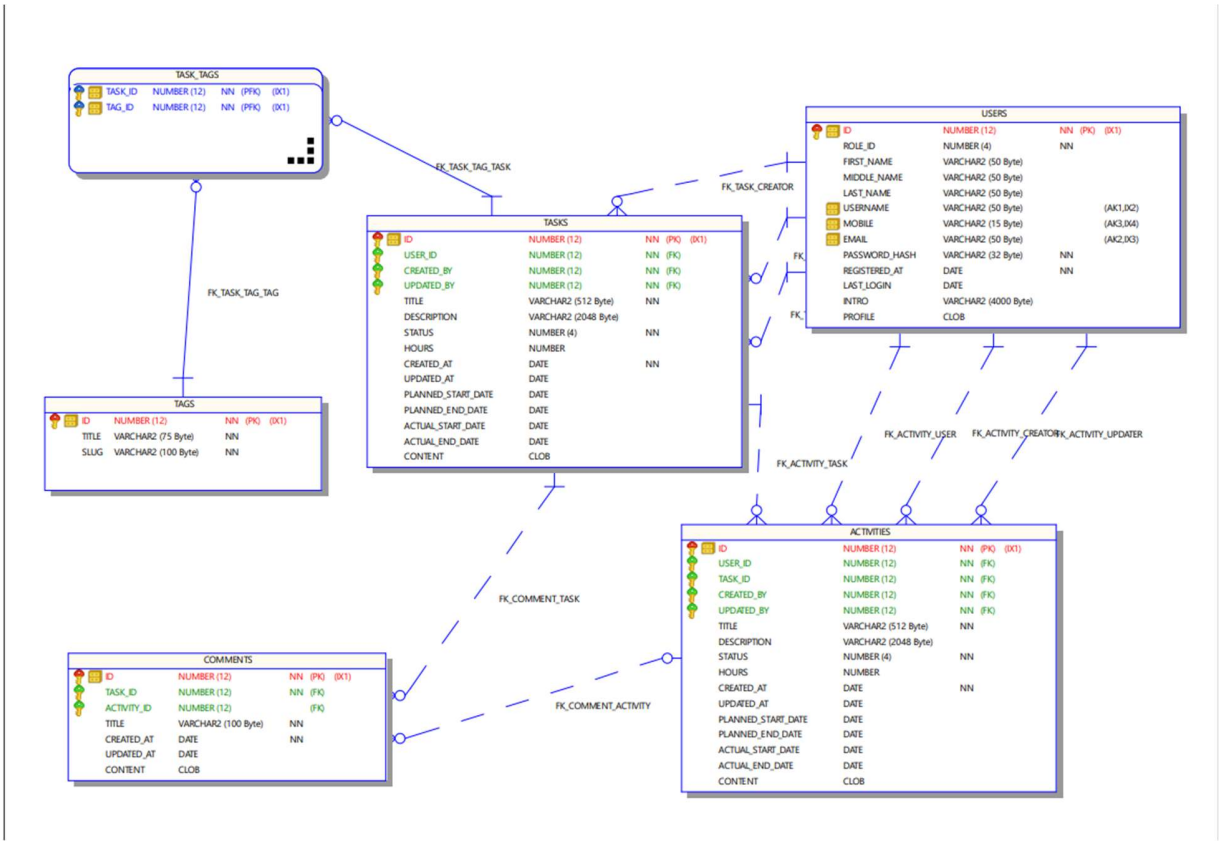
6. Activities table:

```
CREATE TABLE activities (
  id NUMBER(12) NOT NULL,
  user_id NUMBER(12) NOT NULL,
  task_id NUMBER(12) NOT NULL,
  created_by NUMBER(12) NOT NULL,
  updated_by NUMBER(12) NOT NULL,
  title VARCHAR2(512) NOT NULL,
  description VARCHAR2(2048),
  status NUMBER(4) DEFAULT 0 NOT NULL,
  hours NUMBER DEFAULT 0,
  created_at DATE NOT NULL,
  updated_at DATE,
  planned_start_date DATE,
  planned_end_date DATE,
  actual_start_date DATE,
  actual_end_date DATE,
  content CLOB,
  CONSTRAINT pk_activity PRIMARY KEY (id),
  CONSTRAINT fk_activity_user FOREIGN KEY (user_id) REFERENCES
users(id) ON DELETE CASCADE,
  CONSTRAINT fk_activity_task FOREIGN KEY (task_id) REFERENCES
tasks(id) ON DELETE CASCADE,
  CONSTRAINT fk_activity_creator FOREIGN KEY (created_by)
REFERENCES users(id) ON DELETE CASCADE,
  CONSTRAINT fk_activity_updater FOREIGN KEY (updated_by)
REFERENCES users(id) ON DELETE CASCADE
);
```

Normalization:

Applied up to the third normal form (3NF) to eliminate redundancy and maintain data integrity across the system. This ensures data is stored in a logically organized manner, enhancing both retrieval efficiency and scalability.

ER Diagram:


TASK_TAGS
TASK_ID NUMBER (12) NN (PFK) (IX1)
TAG_ID NUMBER (12) NN (PFK) (IX1)
FK_TASK_TAG_TASK
FK_TASK_TAG_TAG
TAGS
ID NUMBER (12) NN (PK) (IX1)
TITLE VARCHAR2 (75 Byte) NN
SLUG VARCHAR2 (100 Byte) NN
TASKS
ID NUMBER (12) NN (PK) (IX1)
USER_ID NUMBER (12) NN (FK)
CREATED_BY NUMBER (12) NN (FK)
UPDATED_BY NUMBER (12) NN (FK)
TITLE VARCHAR2 (512 Byte) NN
DESCRIPTION VARCHAR2 (2048 Byte)
STATUS NUMBER (4) NN
HOURS NUMBER
CREATED_AT DATE NN
UPDATED_AT DATE
PLANNED_START_DATE DATE
PLANNED_END_DATE DATE
ACTUAL_START_DATE DATE
ACTUAL_END_DATE DATE
CONTENT CLOB
FK_TASK_CREATOR
USERS
ID NUMBER (12) NN (PK) (IX1)
ROLE_ID NUMBER (4) NN
FIRST_NAME VARCHAR2 (50 Byte)
MIDDLE_NAME VARCHAR2 (50 Byte)
LAST_NAME VARCHAR2 (50 Byte)
USERNAME VARCHAR2 (50 Byte) (AK1,IX2)
MOBILE VARCHAR2 (15 Byte) (AK3,IX4)
EMAIL VARCHAR2 (50 Byte) (AK2,IX3)
PASSWORD_HASH VARCHAR2 (32 Byte) NN
REGISTERED_AT DATE NN
LAST_LOGIN DATE
INTRO VARCHAR2 (4000 Byte)
PROFILE CLOB
FK_ACTIVITY_USER
FK_ACTIVITY_CREATOR
FK_ACTIVITY_UPDATER
FK_ACTIVITY_TASK
ACTIVITIES
ID NUMBER (12) NN (PK) (IX1)
USER_ID NUMBER (12) NN (FK)
TASK_ID NUMBER (12) NN (FK)
CREATED_BY NUMBER (12) NN (FK)
UPDATED_BY NUMBER (12) NN (FK)
TITLE VARCHAR2 (512 Byte) NN
DESCRIPTION VARCHAR2 (2048 Byte)
STATUS NUMBER (4) NN
HOURS NUMBER
CREATED_AT DATE NN
UPDATED_AT DATE
PLANNED_START_DATE DATE
PLANNED_END_DATE DATE
ACTUAL_START_DATE DATE
ACTUAL_END_DATE DATE
CONTENT CLOB
FK_COMMENT_TASK
FK_COMMENT_ACTIVITY
COMMENTS
ID NUMBER (12) NN (PK) (IX1)
TASK_ID NUMBER (12) NN (FK)
ACTIVITY_ID NUMBER (12) (FK)
TITLE VARCHAR2 (100 Byte) NN
CREATED_AT DATE NN
UPDATED_AT DATE
CONTENT CLOB

SQL Queries and Functionality

1. List All Tasks Assigned to a User

```
SELECT title, description, status, created_at, planned_end_date FROM tasks WHERE user_id = (SELECT id FROM users WHERE username = 'johndoe');
```

This query retrieves all tasks assigned to a specific user, showing task details including the creation date and planned end date.

2. Search for Tasks by Tag

```
SELECT t.title, t.description FROM tasks t JOIN task_tags tt ON t.id = tt.task_id JOIN tags tg ON tt.tag_id = tg.id WHERE tg.title = 'Urgent';
```

This query returns all tasks labeled with the "Urgent" tag, displaying the task title and description.

3. Retrieve Recent Activities for a Task

```
SELECT a.title, a.description, a.created_at FROM activities a WHERE a.task_id = 1 ORDER BY a.created_at DESC;
```

This query fetches the most recent activities related to a specific task, sorted by the date they were created.

4. Update Task Description and Status

```
UPDATE tasks SET description = 'Updated description here', status = 2 WHERE id = 1;
```

This command updates the description and status of a specific task.

5. Add a Comment to an Activity

```
INSERT INTO comments (task_id, activity_id, title, created_at, content) VALUES (1, 1, 'New Comment', SYSDATE, 'This is a new comment on the activity.');
```

This query adds a new comment to a specific activity, noting the time it was created.

6. List Tasks with Their Tags

```
SELECT t.title, LISTAGG(tg.title, ', ') WITHIN GROUP (ORDER BY tg.title) AS tags
FROM tasks t JOIN task_tags tt ON t.id = tt.task_id JOIN tags tg ON tt.tag_id =
tg.id GROUP BY t.title;
```

This query lists tasks along with their associated tags, grouped and ordered by task title.

7. Delete a Task and Its Related Entries

```
DELETE FROM tasks WHERE id = 1;
```

This command deletes a specific task. Due to the "ON DELETE CASCADE" constraints, all related activities, task_tags, and comments will also be automatically deleted.

8. Count of Tasks by Status

```
SELECT status, COUNT(*) AS task_count FROM tasks GROUP BY status;
```

This query provides a count of tasks grouped by their status, useful for reporting and status overviews.

These examples cover a wide range of operations, from basic data retrieval to more complex joins and aggregations, reflecting diverse use cases within the task management system.

Conclusion

The development of the Task Manager database involved integrating complex SQL functionalities and managing a dynamic schema that adapts to the evolving needs of task management. Challenges included ensuring high performance with increasing data volume and integrating user feedback into the database design effectively. The Task Manager database is crucial for improving organizational efficiency and individual productivity by providing a structured and reliable platform for managing tasks and activities. It supports detailed tracking of task progress and user engagement, making it an invaluable tool in various professional and personal settings.