

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.

***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)

);

1. 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

);

1. 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)

);

1. 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)

);

1. 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

);

1. 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,

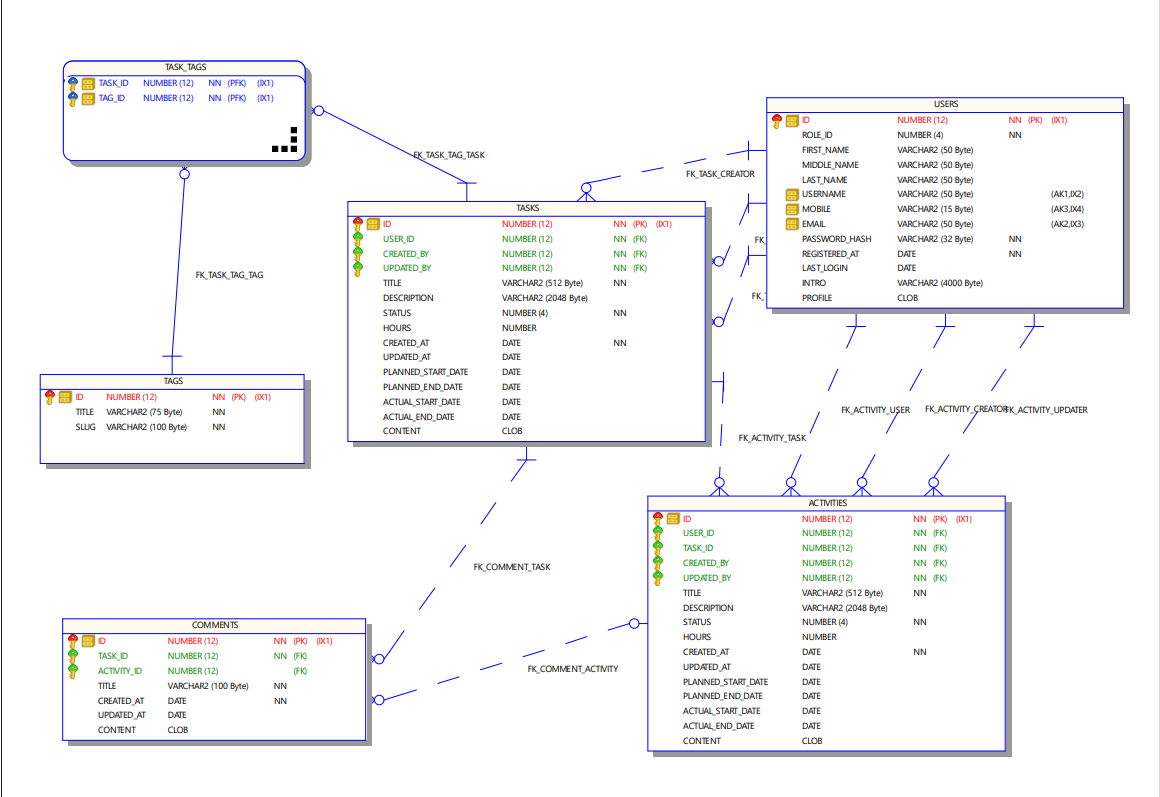
CONSTRAINT fk\_activity\_updater FOREIGN KEY (updated\_by) REFERENCES users(id) ON DELETE CASCADE

);

**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:**



***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.