Bike Sharing App Database

02.05.2024

Shoab Ahamed
Roll 2007015
Student at Khulna Engineering & University
Dept Computer Science & Engineering

Project Overview

The bike sharing database system is designed to efficiently manage the operations and data associated with a bike sharing service. This system aims to provide a seamless experience for users, administrators, and managers involved in the bike sharing process.

Throughout this report, we will delve deeper into the various aspects of this database system, including its design, implementation, functionalities, and potential future enhancements.

Introduction

The popularity of bike-sharing systems has been on the rise in urban areas around the world. These systems provide an eco-friendly, convenient, and cost-effective alternative for short-distance commuting and leisure riding.

In response to this growing trend, we have developed a comprehensive bike sharing database system designed to efficiently manage the operations and data associated with a bike sharing service.

This report provides an overview of the bike sharing database system, including its design, implementation, functionalities, and potential future enhancements. We will discuss the database schema, the benefits of the system, and how it will contribute to the overall success of the bike sharing service.

Database Design

The bike sharing database system is designed using a star schema approach, which is well-suited for data warehousing and analytical purposes. This design allows for efficient querying and analysis of data, providing valuable insights into bike sharing operations.

In the star schema design, the central fact table represents the core data of the system, surrounded by dimension tables that provide additional context and details. This design simplifies queries and allows for fast aggregation of data.

Overview of Database Schema

In the database system there are following tables

I. Bike Rides

Tracks information about the rides taken by users, including rider id, start station id, end station id, byke type, start time, end time. It has a one to many relationship with *Riders table.*

II. Riders

Stores information about the users registered in the system, including rider ID, name, email, phone number, and other relevant details. It is the center table in the database system connected to all the tables other than *Bike Amounts table.*

III. Bike Types

Stores information about different types of bikes available in the system, including bike type ID, name. It has a one to many table *Bike Amounts table.*

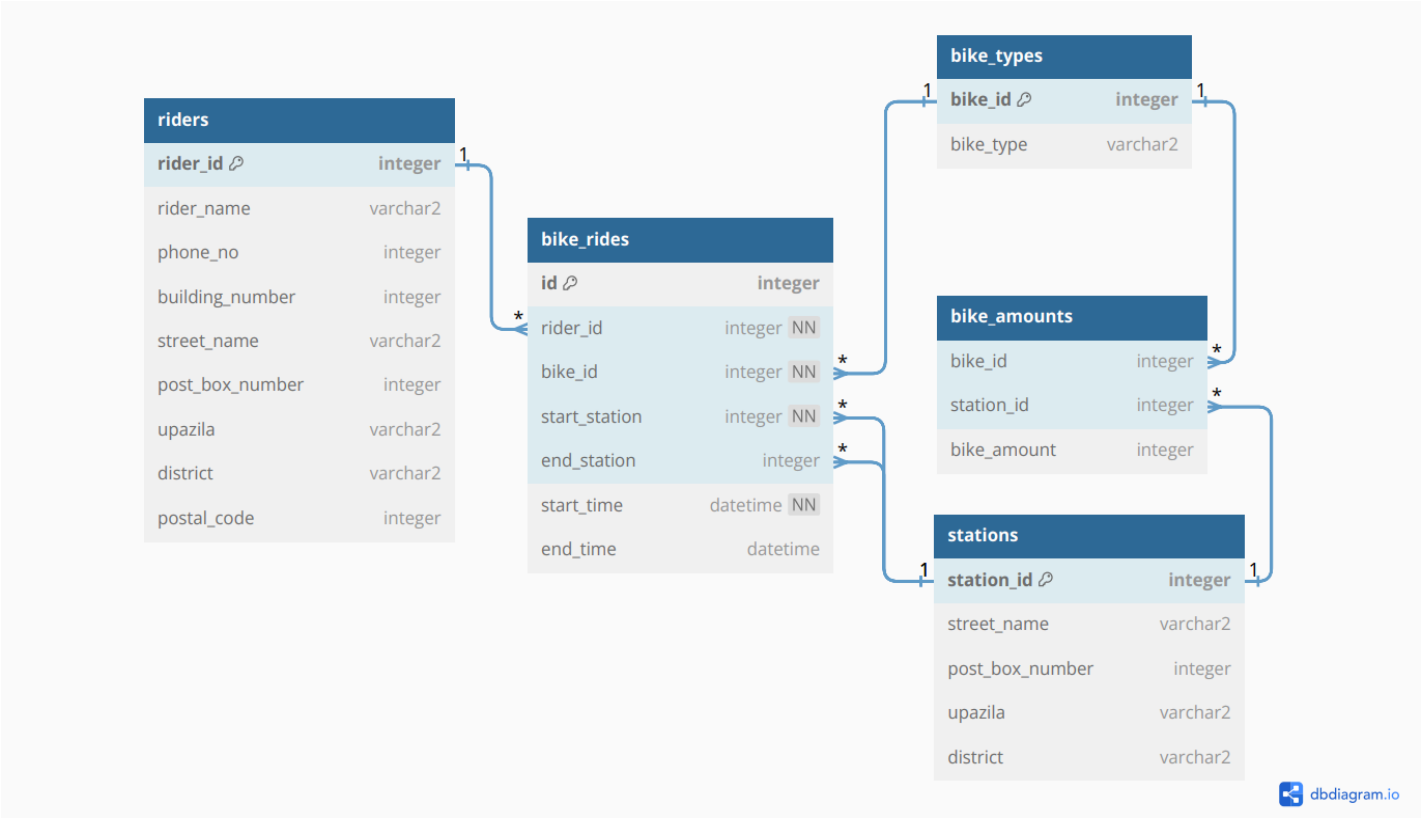
IV. Stations

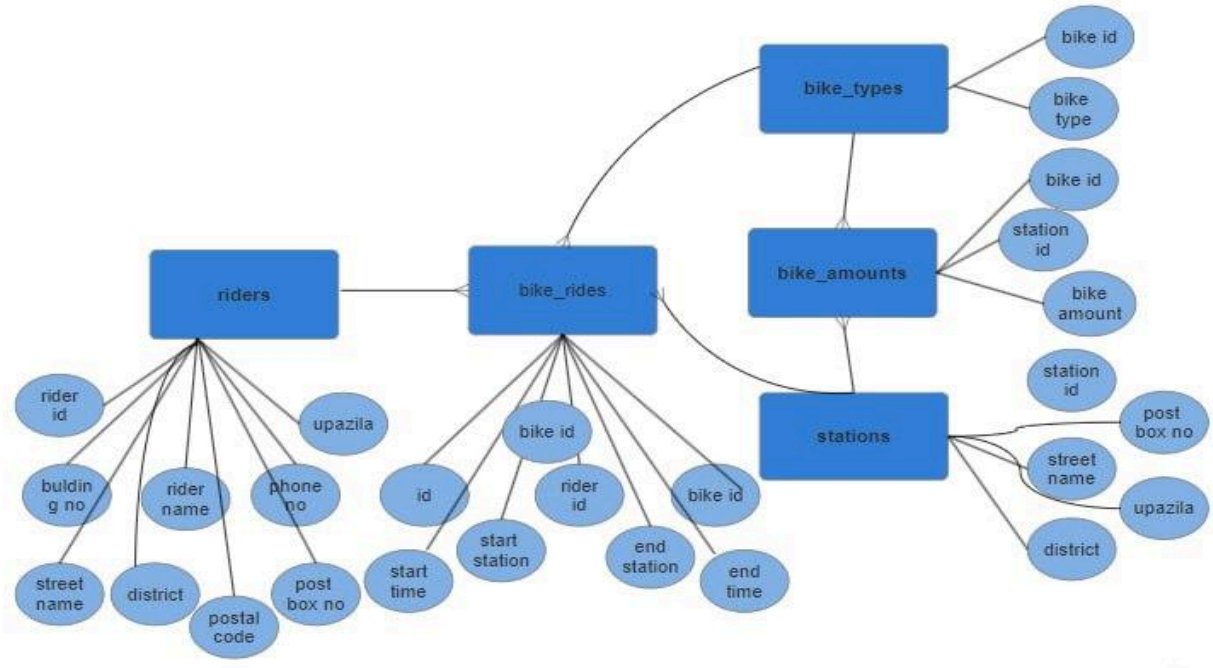
Contains information about the bike stations, including station ID, name, location etc. It has a one to many table *Bike Amounts table.*

V. Bike Amounts

Keeps track of the number of bikes available, broken down by bike type and stations. The linking table which connects *Stations and Bike Types* table

Entity Relationship Diagram





Implementation

The bike sharing database system has been implemented using Oracle 11g, a robust and reliable relational database management system.

```
--Bike Types
CREATE TABLE bike_types(
   bike_id INTEGER PRIMARY KEY,
   bike_type VARCHAR2(40) NOT NULL
   );

--Stations
CREATE TABLE stations(
    station_id INTEGER PRIMARY KEY,
    street_name VARCHAR2(40) NOT NULL,
    post_box_number INTEGER,
    upazila VARCHAR2(40),
    district VARCHAR2(40)
   );
```

```
--Riders
CREATE TABLE riders(
    rider_id INTEGER PRIMARY KEY,
    rider_name VARCHAR2(40) NOT NULL,
    phone_no INTEGER UNIQUE NOT NULL,
    building_number INTEGER,
    street_name VARCHAR2(60),
    post_box_number INTEGER,
    upazila VARCHAR2(40),
    district VARCHAR2(40),
    postal_code INTEGER
   );
```

```
--Bike Amounts
CREATE TABLE bike_amounts(
    bike_id INTEGER NOT NULL,
    station_id INTEGER NOT NULL,
    bike_amount INTEGER NOT NULL CHECK(bike_amount>=0),
    FOREIGN KEY(bike_id) REFERENCES bike_types(bike_id),
    FOREIGN KEY(station_id) REFERENCES stations(station_id)
    );
```

```
--Bike Rides
CREATE TABLE bike_rides(
    id INTEGER PRIMARY KEY,
    rider_id INTEGER NOT NULL,
    bike_id INTEGER NOT NULL,
    start_station INTEGER NOT NULL,
    end_station INTEGER NOT NULL,
    start_time DATE NOT NULL,
    end_time DATE NOT NULL,
    FOREIGN KEY(rider_id) REFERENCES riders(rider_id) ON DELETE CASCADE,
    FOREIGN KEY(bike_id) REFERENCES bike_types(bike_id),
    FOREIGN KEY(start_station) REFERENCES stations(station_id),
    FOREIGN KEY(end_station) REFERENCES stations(station_id),
    CONSTRAINT check_date_time CHECK(
    end_time > start_time
    )
  );
```

Triggers

Mainly there are two kind of trigger. One is for auto incrementation of primary keys in Bike Rides table and Riders Table as they would be updated all the time with new rides or user data. Only in oracle we have to do this. In other sql languages like mysql you do not have go through this lengthy process

```
--creating sequences for auto incrementation primary key in bike_rides and riders table
CREATE SEQUENCE bike_rides_sequence;
CREATE SEQUENCE riders_sequence;
CREATE OR REPLACE TRIGGER bike_rides_on_insert
  BEFORE INSERT ON bike_rides
  FOR EACH ROW
BEGIN
  SELECT bike_rides_sequence.nextval
  INTO :new.id
  FROM dual;
END;
CREATE OR REPLACE TRIGGER riders_on_insert
  BEFORE INSERT ON riders
  FOR EACH ROW
BEGIN
  SELECT riders_sequence.nextval
  INTO :new.rider_id
  FROM dual;
END;
```

Another trigger is when a new row is inserted in the rides table. The amount bike would increase in end station and decrease in start station for corresponding bike type

```
-- a trigger which will update the bike_amounts depending on the new bike_rides record
SET SERVEROUTPUT ON
CREATE OR REPLACE TRIGGER update_bike_amounts
AFTER INSERT ON bike_rides
FOR EACH ROW
Declare
start_station_id bike_rides.start_station%type;
end_station_id bike_rides.end_station%type;
bike_id BIKE_RIDES.BIKE_ID%type;
BEGIN
    start_station_id := :new.start_station;
    end_station_id := :new.end_station;
    bike_id := :new.bike_id;

    UPDATE bike_amounts set bike_amount=bike_amount+1 WHERE station_id=end_station_id AND bike_id = bike_id;
    UPDATE bike_amounts set bike_amount=bike_amount-1 WHERE station_id=start_station_id AND bike_id = bike_id;
--   dbms_output.put_line('START STATION: '||start_station_id|| ' END STATION: ' || end_station_id);
END;
/
```

Constraints

There are different types of constraint used here

1. Primary Constraint:

The primary key constraint ensures that each record in the table is uniquely identified by a primary key. It prevents duplicate and null values in the specified column(s). The tables that used this constraints are:

- riders
- bike_rides
- bike_types
- Stations

2. Unique Constraint:

The unique constraint ensures that each record in the table is uniquely identified. Only phone attribute used this constraint.

3. Non Null Constraint:

The not null constraint ensures that the specified column(s) do not contain null values. It requires that these columns have a value for every record in the table, preventing the insertion of null values.The tables that used this constraints are:

- riders (rider_name, phone_no)
- bike_rides (rider_id, bike_id, start_station, end_station, start_time, end_time)
- bike_amounts (bike_id, station_id, bike_amount)
- stations (street_name)

4.Check Constraint:

The check constraint specifies a condition that must be satisfied for a row to be inserted or updated. In this case, it ensures that the bike_amount column contains only non-negative values. The tables that used this constraint are

- bike_amount(bike_amount has to be greater than or equal to zero)
- rides(end_time must be greater than start time)

SQL Queries and Functionality

To facilitate a wide range of informational needs for users, several SQL queries have been designed to extract useful information from the database. Here are examples illustrating common queries:

4. Most Frequent Riders

```
SELECT rider_id, COUNT(*) AS ride_count
FROM bike_rides
GROUP BY rider_id
ORDER BY ride_count DESC;
```

5. Amount of bike available in station 1

```
SELECT bike_amount
FROM bike_amounts
WHERE station_id = 1;
```

6. Total amount bikes available to the company

```
SELECT SUM(bike_amount) AS total_bikes_available
FROM bike_amounts;
```

7. Total amount of time spent by each rider

```
SELECT rider_id, AVG((end_time - start_time) * 24 * 60 * 60) AS "Average Duration (in seconds)"
FROM bike_rides GROUP BY rider_id;
```

8. Address Of Rider with id=1

```
SELECT *
FROM riders
WHERE rider_id = 1;
```

Targeted Customers/Users

The bike sharing database system is designed to cater to the needs of various stakeholders involved in bike sharing operations. The target users of this system include:

1. End Users (Riders):

Individuals who use the bike sharing service to commute or travel short distances.

They use the system to locate, rent, and return bikes.

2. Administrators:

Personnel responsible for managing bike sharing operations.

They use the system to monitor bike availability, track user activity, manage stations, and perform administrative tasks.

3. Managers:

Higher-level personnel responsible for overseeing bike sharing operations.

They use the system to analyze data, generate reports, make strategic decisions, and optimize operations.

Maintenance Personnel:

Personnel responsible for bike maintenance and repairs.

They use the system to track bike status, schedule maintenance tasks, and manage bike inventory.

Customer Support:

Personnel responsible for addressing user inquiries, issues, and complaints.

They use the system to respond to user queries, resolve issues, and provide support.

Data Analysts:

Personnel responsible for analyzing data and generating insights.

They use the system to extract and analyze data, identify trends, and make data-driven decisions.

Conclusion

The bike sharing database system represents a comprehensive solution for efficiently managing bike sharing operations and data. By implementing a star schema design using Oracle 11g, we have created a robust and scalable database system that meets the needs of various stakeholders involved in bike sharing services.

Throughout this report, we have explored the various components of the database system, including its design, implementation, functionalities, and potential future enhancements. We have discussed the database schema, the benefits of the system, and how it contributes to the overall success of bike sharing services.

By leveraging the capabilities of the bike sharing database system, users, administrators, managers, and other stakeholders can effectively manage bike inventory, optimize operations, enhance the user experience, and make data-driven decisions.