Course work
on the topic Database for
Board games Saas app

Implemented by:
Heorhi Bachyla

Project Documentation

1. Introduction

1.1 Project Description

The project is a service for creating and booking online/offline events, as well as reserving tables in board game clubs.

The main goal of the project is to provide users with an easy way to organize and attend various events, as well as leave reviews.

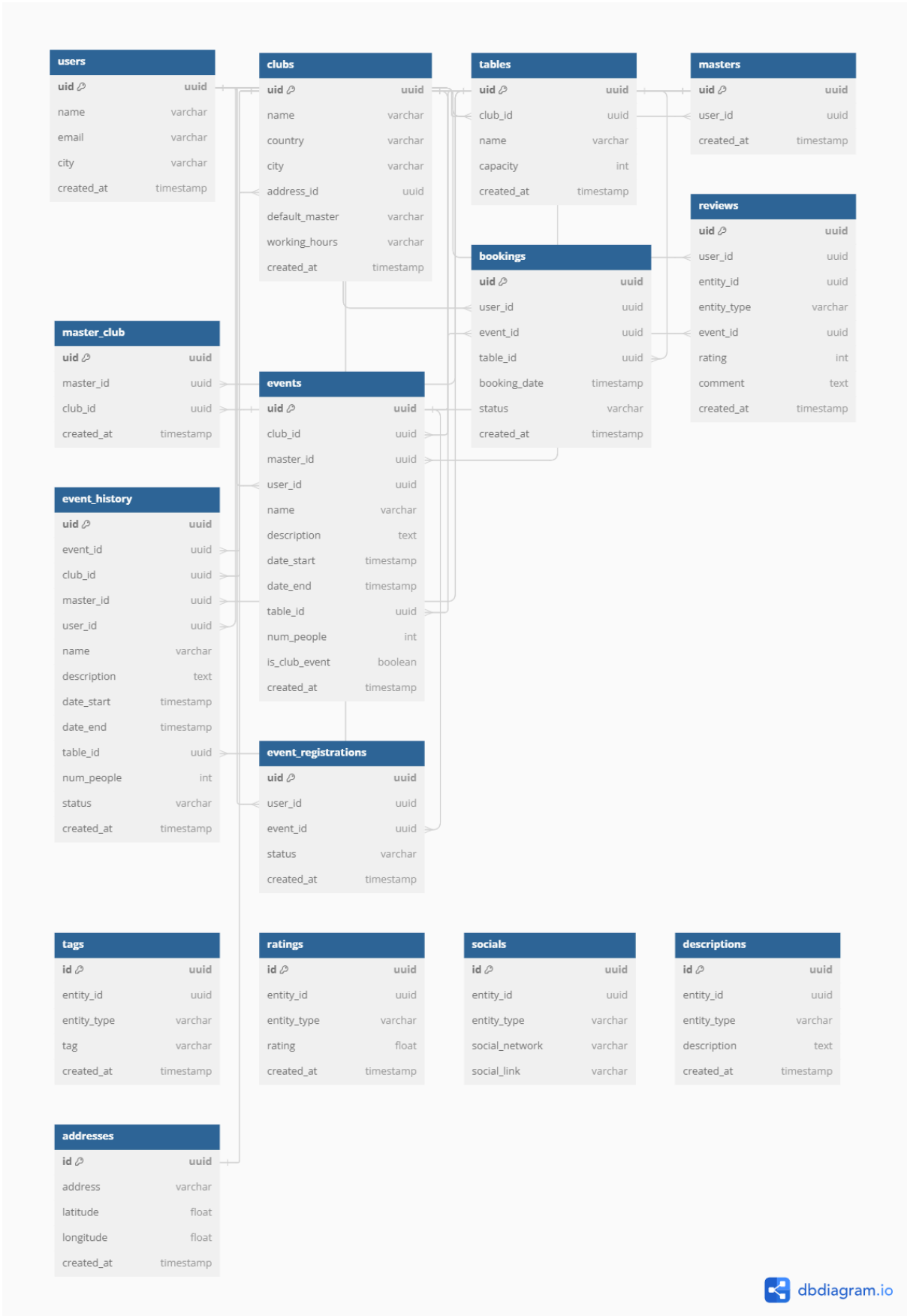
2. Task Definition

Goals and Objectives:

- Develop a data model for storing information about users, clubs, tables, game masters, events, and reservations.
- Implement functions and procedures for working with the data.
- Develop an ETL script for loading data into an OLAP database.
- Create visual reports based on the data in Power BI.

3. ER Diagram

3.1 Logical Data Model


users
uid uuid
name varchar
email varchar
city varchar
created_at timestamp
clubs
uid uuid
name varchar
country varchar
city varchar
address_id uuid
default_master varchar
working_hours varchar
created_at timestamp
tables
uid uuid
club_id uuid
name varchar
capacity int
created_at timestamp
masters
uid uuid
user_id uuid
created_at timestamp
reviews
uid uuid
user_id uuid
entity_id uuid
entity_type varchar
event_id uuid
rating int
comment text
created_at timestamp
bookings
uid uuid
user_id uuid
event_id uuid
table_id uuid
booking_date timestamp
status varchar
created_at timestamp
master_club
uid uuid
master_id uuid
club_id uuid
created_at timestamp
events
uid uuid
club_id uuid
master_id uuid
user_id uuid
name varchar
description text
date_start timestamp
date_end timestamp
table_id uuid
num_people int
is_club_event boolean
created_at timestamp
event_history
uid uuid
event_id uuid
club_id uuid
master_id uuid
user_id uuid
name varchar
description text
date_start timestamp
date_end timestamp
table_id uuid
num_people int
status varchar
created_at timestamp
event_registrations
uid uuid
user_id uuid
event_id uuid
status varchar
created_at timestamp
tags
id uuid
entity_id uuid
entity_type varchar
tag varchar
created_at timestamp
ratings
id uuid
entity_id uuid
entity_type varchar
rating float
created_at timestamp
socials
id uuid
entity_id uuid
entity_type varchar
social_network varchar
social_link varchar
descriptions
id uuid
entity_id uuid
entity_type varchar
description text
created_at timestamp
addresses
id uuid
address varchar
latitude float
longitude float
created_at timestamp
dbdiagram.io

4. Logical and Physical Schemas of the OLTP Database

4.1 Logical Schema

Description of the logical schema of the database, including all tables and its fields.

```
// Users
Table users {
  uid uuid [pk] // Primary key
  name varchar // User's name
  email varchar // User's email address
  city varchar // User's city
  created_at timestamp // Record creation timestamp
}

// Clubs
Table clubs {
  uid uuid [pk] // Primary key
  name varchar // Club's name
  country varchar // Country where the club is located
  city varchar // City where the club is located
  address_id uuid [ref: > addresses.id] // Foreign key
referencing addresses.id
  default_master varchar // Default master of the club
  working_hours varchar // Club's working hours
  created_at timestamp // Record creation timestamp
}

// Tags
Table tags {
  id uuid [pk] // Primary key
  entity_id uuid // ID of the user or club
  entity_type varchar // Type of the entity (e.g., 'user',
'club')
  tag varchar // Tag related to the entity
  created_at timestamp // Record creation timestamp
}

// Ratings
Table ratings {
  id uuid [pk] // Primary key
  entity_id uuid // ID of the user or club
```

```
  entity_type varchar // Type of the entity (e.g., 'user',
'club')
  rating float // Rating value
  created_at timestamp // Record creation timestamp
}

// Socials
Table socials {
  id uuid [pk] // Primary key
  entity_id uuid // ID of the user or club
  entity_type varchar // Type of the entity (e.g., 'user',
'club')
  social_network varchar // Name of the social network
  social_link varchar // Link to the user's or club's
profile
}

// Descriptions
Table descriptions {
  id uuid [pk] // Primary key
  entity_id uuid // ID of the user or club
  entity_type varchar // Type of the entity (e.g., 'user',
'club')
  description text // Description of the entity
  created_at timestamp // Record creation timestamp
}

// Addresses
Table addresses {
  id uuid [pk] // Primary key
  address varchar // Full address
  latitude float // Latitude coordinate
  longitude float // Longitude coordinate
  created_at timestamp // Record creation timestamp
}

// Tables
Table tables {
  uid uuid [pk] // Primary key
  club_id uuid [ref: > clubs.uid] // Foreign key referencing
clubs.uid
  name varchar // Table's name
```

```
  capacity int // Table's capacity (number of people it can
accommodate)
  created_at timestamp // Record creation timestamp
}

// Masters
Table masters {
  uid uuid [pk] // Primary key
  user_id uuid [ref: > users.uid] // Foreign key referencing
users.uid
  created_at timestamp // Record creation timestamp
}

// Master-Club Relationship
Table master_club {
  uid uuid [pk] // Primary key
  master_id uuid [ref: > masters.uid] // Foreign key
referencing masters.uid
  club_id uuid [ref: > clubs.uid] // Foreign key referencing
clubs.uid
  created_at timestamp // Record creation timestamp
}

// Events
Table events {
  uid uuid [pk] // Primary key
  club_id uuid [ref: > clubs.uid, null] // Foreign key
referencing clubs.uid, null if not a club event
  master_id uuid [ref: > masters.uid, null] // Foreign key
referencing masters.uid, null if the user is not a master
  user_id uuid [ref: > users.uid] // Foreign key referencing
users.uid
  name varchar // Event's name
  description text // Event's description
  date_start timestamp // Event start date and time
  date_end timestamp // Event end date and time
  table_id uuid [ref: > tables.uid] // Foreign key
referencing tables.uid
  num_people int // Number of people attending the event
  is_club_event boolean // Flag indicating if the event is a
club event
  created_at timestamp // Record creation timestamp
}
```

```
// Bookings
Table bookings {
  uid uuid [pk] // Primary key
  user_id uuid [ref: > users.uid] // Foreign key referencing
users.uid
  event_id uuid [ref: > events.uid] // Foreign key
referencing events.uid
  table_id uuid [ref: > tables.uid] // Foreign key
referencing tables.uid
  booking_date timestamp // Booking date
  status varchar // Booking status (active, completed,
canceled)
  created_at timestamp // Record creation timestamp
}

// Reviews
Table reviews {
  uid uuid [pk] // Primary key
  user_id uuid [ref: > users.uid] // Foreign key referencing
users.uid
  entity_id uuid // Foreign key referencing clubs.uid or
users.uid
  entity_type varchar // Type of the entity (e.g., 'club',
'user')
  event_id uuid [ref: > events.uid] // Foreign key
referencing events.uid
  rating int // Review rating (1-5)
  comment text // Review comment
  created_at timestamp // Record creation timestamp
}

// Event History
Table event_history {
  uid uuid [pk] // Primary key
  event_id uuid [ref: > events.uid] // Foreign key
referencing events.uid
  club_id uuid [ref: > clubs.uid] // Foreign key referencing
clubs.uid
  master_id uuid [ref: > masters.uid, null] // Foreign key
referencing masters.uid, null if no master
  user_id uuid [ref: > users.uid] // Foreign key referencing
users.uid
```

```
  name varchar // Event's name
  description text // Event's description
  date_start timestamp // Event start date and time
  date_end timestamp // Event end date and time
  table_id uuid [ref: > tables.uid] // Foreign key
referencing tables.uid
  num_people int // Number of people who attended the event
  status varchar // Event status (completed, canceled)
  created_at timestamp // Record creation timestamp
}

// Event Registrations
Table event_registrations {
  uid uuid [pk] // Primary key
  user_id uuid [ref: > users.uid] // Foreign key referencing
users.uid
  event_id uuid [ref: > events.uid] // Foreign key
referencing events.uid
  status varchar // Registration status (registered,
pending, rejected)
  created_at timestamp // Record creation timestamp
}
```

4.2 Physical Schema

Description of the physical schema of the database, including data types and constraints.

Users Table

CREATE TABLE users (
  uid UUID PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL,
  city VARCHAR(255),
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP

);

Field Descriptions:

uid: Universal unique identifier (UUID) for each user, primary key.
name: User name, a string up to 255 characters, required field.
email: User email, a string up to 255 characters, required field.
city: User city, a string up to 255 characters.
created_at: Record creation time, timestamp, default value is the current time
Clubs Table

```
CREATE TABLE clubs (
  uid UUID PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  country VARCHAR(255),
  city VARCHAR(255),
  address_id UUID REFERENCES addresses(id),
  default_master VARCHAR(255),
  working_hours VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Field Descriptions:

uid: Universal unique identifier (UUID) for each club, primary key.

name: Club name, a string up to 255 characters, required field.

country: Country where the club is located, a string up to 255 characters.

city: City where the club is located, a string up to 255 characters.

address_id: Universal unique identifier (UUID) for the address, reference to the addresses table.

default_master: Name of the club's main master, a string up to 255 characters.

working_hours: Club working hours, a string up to 255 characters.

created_at: Record creation time, timestamp, default value is the current time.

Addresses Table

```
CREATE TABLE addresses (
  id UUID PRIMARY KEY,
  address VARCHAR(255) NOT NULL,
  latitude FLOAT,
  longitude FLOAT,
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

id: Universal unique identifier (UUID) for each address, primary key.
address: Address, a string up to 255 characters, required field.
latitude: Latitude, floating-point number.
longitude: Longitude, floating-point number.
created_at: Record creation time, timestamp, default value is the current time.

Tags Table

```
CREATE TABLE tags (
  id UUID PRIMARY KEY,
  entity_id UUID NOT NULL,
  entity_type VARCHAR(50) NOT NULL,
  tag VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

id: Universal unique identifier (UUID) for each tag, primary key.
entity_id: Universal unique identifier (UUID) of the entity (user or club) to which the tag belongs.

entity_type: Type of entity (e.g., 'user' or 'club'), a string up to 50 characters.

tag: Tag, a string up to 255 characters, required field.

created_at: Record creation time, timestamp, default value is the current time.

Ratings Table

```
CREATE TABLE ratings (
  id UUID PRIMARY KEY,
  entity_id UUID NOT NULL,
  entity_type VARCHAR(50) NOT NULL,
  rating FLOAT NOT NULL,
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

id: Universal unique identifier (UUID) for each rating, primary key.

entity_id: Universal unique identifier (UUID) of the entity (user or club) to which the rating belongs.

entity_type: Type of entity (e.g., 'user' or 'club'), a string up to 50 characters.

rating: Rating, floating-point number, required field.

created_at: Record creation time, timestamp, default value is the current time.

Socials Table

```
CREATE TABLE socials (
  id UUID PRIMARY KEY,
  entity_id UUID NOT NULL,
  entity_type VARCHAR(50) NOT NULL,
  social_network VARCHAR(255) NOT NULL,
  social_link VARCHAR(255) NOT NULL
);
```

Field Descriptions:

id: Universal unique identifier (UUID) for each social media record, primary key.
entity_id: Universal unique identifier (UUID) of the entity (user or club) to which the social network belongs.
entity_type: Type of entity (e.g., 'user' or 'club'), a string up to 50 characters.
social_network: Name of the social network, a string up to 255 characters, required field.
social_link: Link to the social media profile, a string up to 255 characters, required field.

Descriptions Table

```
CREATE TABLE descriptions (
  id UUID PRIMARY KEY,
  entity_id UUID NOT NULL,
  entity_type VARCHAR(50) NOT NULL,
  description TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

id: Universal unique identifier (UUID) for each description, primary key.
entity_id: Universal unique identifier (UUID) of the entity (user or club) to which the description belongs.
entity_type: Type of entity (e.g., 'user' or 'club'), a string up to 50 characters.
description: Description, text, required field.
created_at: Record creation time, timestamp, default value is the current time.

Tables Table

```
CREATE TABLE tables (
  uid UUID PRIMARY KEY,
  club_id UUID REFERENCES clubs(uid),
  name VARCHAR(255) NOT NULL,
  capacity INT NOT NULL,
```

```
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

uid: Universal unique identifier (UUID) for each table, primary key.

club_id: Universal unique identifier (UUID) of the club where the table is located, reference to the clubs table.

name: Table name, a string up to 255 characters, required field.

capacity: Table capacity, integer, required field.

created_at: Record creation time, timestamp, default value is the current time.

Masters Table

```
CREATE TABLE masters (
  uid UUID PRIMARY KEY,
  user_id UUID REFERENCES users(uid),
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

uid: Universal unique identifier (UUID) for each master, primary key.

user_id: Universal unique identifier (UUID) of the user who is a master, reference to the users table.

created_at: Record creation time, timestamp, default value is the current time.

Master-Club Relationship Table

```
CREATE TABLE master_club (
  uid UUID PRIMARY KEY,
  master_id UUID REFERENCES masters(uid),
  club_id UUID REFERENCES clubs(uid),
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

uid: Universal unique identifier (UUID) for each master-club relationship record, primary key.

master_id: Universal unique identifier (UUID) of the master, reference to the masters table.

club_id: Universal unique identifier (UUID) of the club, reference to the clubs table.

created_at: Record creation time, timestamp, default value is the current time.

Events Table

```
CREATE TABLE events (
  uid UUID PRIMARY KEY,
  club_id UUID REFERENCES clubs(uid),
  master_id UUID REFERENCES masters(uid),
  user_id UUID REFERENCES users(uid),
  name VARCHAR(255) NOT NULL,
  description TEXT NOT NULL,
  date_start TIMESTAMP NOT NULL,
  date_end TIMESTAMP NOT NULL,
  table_id UUID REFERENCES tables(uid),
  num_people INT NOT NULL,
  is_club_event BOOLEAN,
  status VARCHAR(50),
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

uid: Universal unique identifier (UUID) for each event, primary key.

club_id: Universal unique identifier (UUID) of the club, reference to the clubs table.

master_id: Universal unique identifier (UUID) of the master, reference to the masters table.

user_id: Universal unique identifier (UUID) of the user, reference to the users table.

name: Event name, a string up to 255 characters, required field.

description: Event description, text, required field.

date_start: Event start date and time, timestamp, required field.

date_end: Event end date and time, timestamp, required field.

table_id: Universal unique identifier (UUID) of the table, reference to the tables table.

num_people: Number of people, integer, required field.

is_club_event: Boolean indicating whether the event is a club event.

status: Event status, a string up to 50 characters.

created_at: Record creation time, timestamp, default value is the current time.

Bookings Table

```
CREATE TABLE bookings (
  uid UUID PRIMARY KEY,
  user_id UUID REFERENCES users(uid),
  event_id UUID REFERENCES events(uid),
  table_id UUID REFERENCES tables(uid),
  booking_date TIMESTAMP NOT NULL,
  status VARCHAR(50) NOT NULL,
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

uid: Universal unique identifier (UUID) for each booking, primary key.
user_id: Universal unique identifier (UUID) of the user, reference to the users table.
event_id: Universal unique identifier (UUID) of the event, reference to the events table.
table_id: Universal unique identifier (UUID) of the table, reference to the tables table.
booking_date: Booking date and time, timestamp, required field.
status: Booking status, a string up to 50 characters, required field.
created_at: Record creation time, timestamp, default value is the current time.

Reviews Table

```
CREATE TABLE reviews (
  uid UUID PRIMARY KEY,
  user_id UUID REFERENCES users(uid),
  entity_id UUID NOT NULL,
  entity_type VARCHAR(50) NOT NULL,
  event_id UUID REFERENCES events(uid),
  rating INT NOT NULL,
  comment TEXT,
```

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

Field Descriptions:

uid: Universal unique identifier (UUID) for each review, primary key.
user_id: Universal unique identifier (UUID) of the user, reference to the users table.
entity_id: Universal unique identifier (UUID) of the entity (user or club) to which the review belongs.
entity_type: Type of entity (e.g., 'user' or 'club'), a string up to 50 characters.
event_id: Universal unique identifier (UUID) of the event, reference to the events table.
rating: Rating, integer, required field.
comment: Comment, text.
created_at: Record creation time, timestamp, default value is the current time.

Event History Table

```
CREATE TABLE event_history (
  uid UUID PRIMARY KEY,
  event_id UUID REFERENCES events(uid),
  club_id UUID REFERENCES clubs(uid),
  master_id UUID REFERENCES masters(uid),
  user_id UUID REFERENCES users(uid),
```

```
    name VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    date_start TIMESTAMP NOT NULL,
    date_end TIMESTAMP NOT NULL,
    table_id UUID REFERENCES tables(uid),
    num_people INT NOT NULL,
    status VARCHAR(50) NOT NULL,
    created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

uid: Universal unique identifier (UUID) for each event history record, primary key.

event_id: Universal unique identifier (UUID) of the event, reference to the events table.

club_id: Universal unique identifier (UUID) of the club, reference to the clubs table.

master_id: Universal unique identifier (UUID) of the master, reference to the masters table.

user_id: Universal unique identifier (UUID) of the user, reference to the users table.

name: Event name, a string up to 255 characters, required field.

description: Event description, text, required field.

date_start: Event start date and time, timestamp, required field.

date_end: Event end date and time, timestamp, required field.

table_id: Universal unique identifier (UUID) of the table, reference to the tables table.

num_people: Number of people, integer, required field.

status: Event status, a string up to 50 characters, required field.

created_at: Record creation time, timestamp, default value is the current time.

Event Registrations Table

```
CREATE TABLE event_registrations (
  uid UUID PRIMARY KEY,
  user_id UUID REFERENCES users(uid),
  event_id UUID REFERENCES events(uid),
  status VARCHAR(50) NOT NULL,
  created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

Field Descriptions:

uid: Universal unique identifier (UUID) for each event registration, primary key.

user_id: Universal unique identifier (UUID) of the user, reference to the users table.

event_id: Universal unique identifier (UUID) of the event, reference to the events table.

status: Registration status, a string up to 50 characters, required field.

created_at: Record creation time, timestamp, default value is the current time.

Indexes

```
-- Users table indexes
CREATE INDEX idx_users_email ON users(email);

-- Clubs table indexes
CREATE INDEX idx_clubs_name ON clubs(name);

-- Tags table indexes
CREATE INDEX idx_tags_entity ON tags(entity_id, entity_type);

-- Ratings table indexes
CREATE INDEX idx_ratings_entity ON ratings(entity_id, entity_type);

-- Socials table indexes
CREATE INDEX idx_socials_entity ON socials(entity_id, entity_type);

-- Descriptions table indexes
```

```
    CREATE INDEX idx_descriptions_entity ON
descriptions(entity_id, entity_type);

    -- Tables table indexes
    CREATE INDEX idx_tables_club ON
tables(club_id);

    -- Masters table indexes
    CREATE INDEX idx_masters_user ON
masters(user_id);

    -- Master-Club Relationship table indexes
    CREATE INDEX idx_master_club ON
master_club(master_id, club_id);

    -- Events table indexes
    CREATE INDEX idx_events_club ON
events(club_id);
    CREATE INDEX idx_events_user ON
events(user_id);

    -- Bookings table indexes
    CREATE INDEX idx_bookings_user ON
bookings(user_id);
    CREATE INDEX idx_bookings_event ON
bookings(event_id);

    -- Reviews table indexes
```

CREATE INDEX idx_reviews_entity ON reviews(entity_id, entity_type);

-- Event History table indexes
CREATE INDEX idx_event_history_event ON event_history(event_id);

-- Event Registrations table indexes
CREATE INDEX idx_event_registrations_user ON event_registrations(user_id);

5. Scripts for Creating Tables, Indexes, Functions, and Procedures can be found in functions_and_triggersevent_booking_oltp.sql

5.1 Scripts for Creating Tables and Indexes can be found in event_booking_oltp.sql

6. Instructions for Running Scripts for Datasets Loading and ETL Process

Overview

This document provides step-by-step instructions for loading datasets into the OLTP database and executing the ETL process to transfer data into the OLAP database. The instructions include creating databases, creating tables, loading data from CSV files, creating necessary functions and triggers, and performing the ETL operations.

Prerequisites

PostgreSQL installed and running.
pgAdmin4 or another PostgreSQL client.
CSV files with initial data located at C:\src\Data\.
Python installed with necessary libraries for ETL script execution.

Step-by-Step Instructions:

1. Create OLTP Database

2. Open your PostgreSQL client and execute the following command to create a new OLTP database:

   CREATE DATABASE event_booking_oltp;

3. Create OLAP Database

Open your PostgreSQL client and execute the following command to create a new OLAP database:

CREATE DATABASE event_booking_olap;

Connect to the OLTP Database

Connect to the newly created event_booking_oltp database.

4. Create OLTP Tables and Load Data

Open a new SQL editor (Query Tool) in your PostgreSQL client and run the script event_booking_oltp.sql to create tables and load data. Follow these steps:

Open pgAdmin4 and connect to your PostgreSQL server.
Select the event_booking_oltp database.
Open a new Query Tool.
Load the event_booking_oltp.sql script and execute it.
Create OLAP Tables

5. Connect to the newly created event_booking_olap database.

Open a new SQL editor (Query Tool) in your PostgreSQL client and run the script event_booking_olap.sql to create OLAP tables. Follow these steps:

Open pgAdmin4 and connect to your PostgreSQL server.
Select the event_booking_olap database.
Open a new Query Tool.
Load the event_booking_olap.sql script and execute it.
Create Functions and Triggers in OLTP

6. Connect to the event_booking_oltp database.

Open a new SQL editor (Query Tool) in your PostgreSQL client and run the script functions_and_triggers_event_booking_oltp.sql to create necessary functions and triggers. Follow these steps:

Open pgAdmin4 and connect to your PostgreSQL server.
Select the event_booking_oltp database.
Open a new Query Tool.
Load the functions_and_triggers_event_booking_oltp.sql script and execute it.
Run OLTP to OLAP ETL Script

7. Connect to the event_booking_oltp database.

Open a new SQL editor (Query Tool) in your PostgreSQL client and run the script oltp_ETL.sql to perform the ETL process from OLTP to OLAP. Follow these steps:

Open pgAdmin4 and connect to your PostgreSQL server.
Select the event_booking_oltp database.
Open a new Query Tool.
Load the oltp_ETL.sql script and execute it.
Verify Data in OLAP Database

8. Connect to the event_booking_olap database.

Open a new SQL editor (Query Tool) in your PostgreSQL client and run select queries to verify the data has been correctly loaded into the OLAP tables. Example:

```
SELECT * FROM dim_users;
SELECT * FROM dim_clubs;
SELECT * FROM fact_events;
SELECT * FROM fact_event_registrations;
SELECT * FROM fact_bookings;
```