### **1. Common Information**

The goal of this work is to design and implement a database for an application that processes user actions, such as track listens. The project involves creating two schemas: OLTP for storing transactional data, and OLAP for analytical processing. This work requires basic skills in relational databases, knowledge of SQL, understanding of ETL processes, and fundamentals of data visualization. The outcome includes a set of SQL scripts, schema descriptions, a ready-to-use storage structure, and a final implementation report.

## **2. OLTP Database Context**

The OLTP database is used for storing transactional data related to user activity in the system. It serves as a source of "raw" data necessary for subsequent analytical processing.

### **Key entities:**

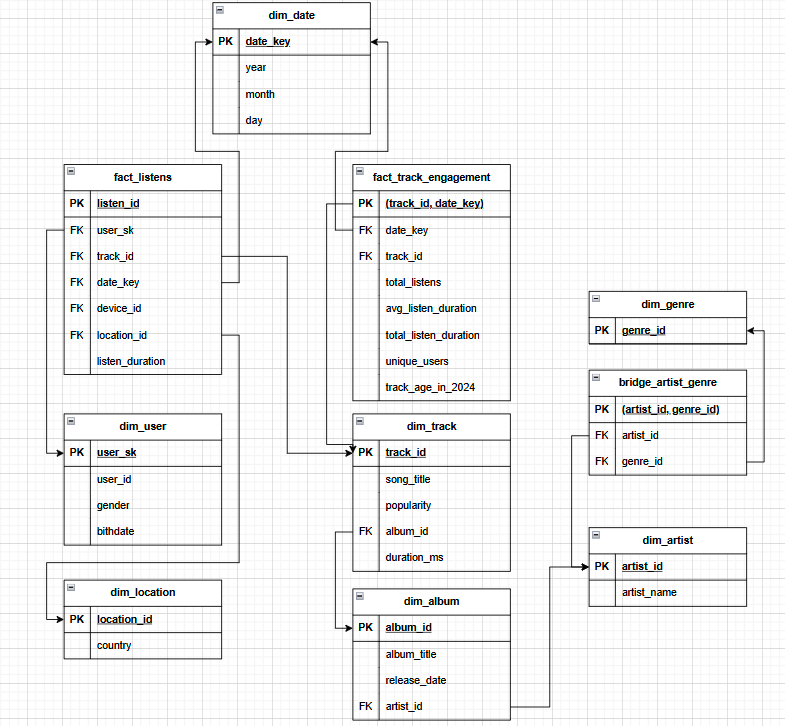
* **tracks**, **artists**, **albums**, **genres** — reference tables containing information about tracks, artists, albums, and music genres;
* **users**, **locations** — data about users and their geographical location;
* **listening\_sessions** — information about listening sessions: date and time, duration, device, geolocation, track;

### **Linking tables for many-to-many relationships:**

* **Track\_artists, track\_albums, artist\_genres.**

. Each table uses primary keys (**PRIMARY KEY**) and foreign keys (**FOREIGN KEY**) to ensure referential data integrity.

## **3. OLAP Database Context**



The analytical data warehouse (OLAP) is built using a star schema and is designed for quickly executing aggregated queries to analyze user behavior, content engagement, demographics, and geographical distribution of activity.

### **Examples of analytical queries:**

* Determining the number of unique users who listened to each track daily;
* Calculating the average listening duration by device type;
* Identifying the most active countries and devices.

Fact tables and dimensions allow for solving a wide range of business intelligence tasks due to their denormalized structure.

## **4. Schemas/tables/keys/constraints/relationships**

### **OLTP-schema**

* Each entity has a primary key (e.g., track\_id, artist\_id) and maintains referential integrity through foreign keys.
* Relationships between tables:
  + One user is associated with one region;
  + One track can belong to multiple albums and have multiple artists;
  + One track can belong to multiple albums and have multiple artists;

### **OLAP-schema**

The schema is organized according to the "star" principle and includes the following components:

**Fact tables**

* fact\_listens — each listening event;
* fact\_track\_engagement — aggregated activity metrics by track.

**Dimensions:**

* dim\_user — user profiles (implemented as SCD Type 2: columns valid\_from, valid\_to, is\_current);
* dim\_date, dim\_device, dim\_location, dim\_track, dim\_album, dim\_artist, dim\_genre — reference tables containing corresponding attribute information.

**Bridge table:**

* bridge\_artist\_genre — a linking table to represent many-to-many relationships between artists and genres.

### **Main table fields:**

#### **fact\_listens:**

| **Field** |  | **Description** |
| --- | --- | --- |
| listen\_id |  | Unique event identifier (surrogate key) |
| user\_sk |  | Foreign key to dim\_user.user\_sk |
| track\_id |  | Foreign key to dim\_track.track\_id |
| start\_time |  | Start time of listening |
| listen\_duration |  | Duration of listening |
| device\_id |  | Foreign key to dim\_device.device\_id |
| location\_id |  | Foreign key to dim\_location.location\_id |

#### **fact\_track\_engagement:**

| **Field** | **Description** |
| --- | --- |
| track\_id | Track identifier |
| date\_key | Date key in YYYYMMDD format |
| total\_listens | Total number of listens |
| avg\_listen\_duration | Average listening duration |
| total\_listen\_duration | Cumulative listening duration |
| unique\_users | Number of unique users |
| track\_age\_in\_2024 | Track age as of early 2024 |
| device\_id | Foreign key to dim\_device.device\_id |
| location \_id | Foreign key to dim\_location.location\_id |

## 

## **5. Data Loading and ETL Instructions**

To launch the entire system and fully load data, it is sufficient to execute two scripts:

oltpscript\_etl\_fromcsv\_script.sql.sql

olapscript\_etl\_fromoltp\_script.sql

These scripts will automatically perform all stages: schema creation, data loading from temporary 1. tables, populating OLTP and OLAP structures, configuring relationships, and aggregating analytical indicators. No additional manual configuration is required, provided the source files are correctly located and the DBMS connection is configured.

Detailed implementation of each stage is described below.

### **Step 1. Loading temporary tables from CSV files**

Ensure that the files are located at C:\csv. If they are saved elsewhere, modify the commands to suit your requirements. Execute the following commands to load data into temporary tables:

copy temp\_tracks\_albums\_artists from 'c:\csv\tracks\_albums\_artists.csv' DELIMITER ',' CSV HEADER;

copy temp\_users from 'c:\csv\users.csv' DELIMITER ',' CSV HEADER;

copy temp\_generated\_sessions from 'c:\csv\generated\_sessions.csv' DELIMITER ',' CSV HEADER;

copy temp\_locations from 'c:\csv\locations.csv' DELIMITER ',' CSV HEADER;

### **Step 2. Loading data into OLTP tables**

s Based on the temporary tables, execute INSERT INTO ... SELECT FROM temp\_... s for the following tables:

* tracks, artists, albums, genres, artist\_genres
* track\_artists, track\_albums
* locations, users
* listening\_sessions

Each query is implemented with support for ON CONFLICT ... DO NOTHING or ON CONFLICT ... DO UPDATE, making the process re-executable and safe with duplicate content.

### **Step 3. Configuring Foreign Data Wrapper**

CREATE EXTENSION IF NOT EXISTS postgres\_fdw;

Its further configuration is specified in the script.

### **Step 4. Populating OLAP Dimensions**

INSERT INTO dim\_artist;

INSERT INTO dim\_genre;

INSERT INTO bridge\_artist\_genre;

INSERT INTO dim\_album;

INSERT INTO dim\_track;

INSERT INTO dim\_location;

INSERT INTO dim\_device;

INSERT INTO dim\_date;

INSERT INTO dim\_user; -- включает логику SCD Type 2

### **Step 5. Populating Fact Tables**

INSERT INTO fact\_listens; -- загрузка по сессиям из OLTP

INSERT INTO fact\_track\_engagement; -- агрегированная аналитика

## **4. Описание визуализации в Power BI**

The report consists of several interconnected visual blocks displaying music listening analytics by users. Data is segmented by day of the week, age, artists, tracks, users, countries, and devices.

### **1.Page 1:**

#### **Line chart: : Count of listen\_id by day\_of\_week**

* Displays the total number of listens by day of the week.
* X-axis – days of the week, Y-axis – number of listens (listen\_id).

#### **Pie chart : Sum of total\_listens by artists**

* Shows the total number of listens by artists.
* Visualized as a percentage of the total listens.

#### **Histogram: Sum of total\_listens by username**

* Displays top users by total number of listens.
* Data provides insight into the activity of individual users.

#### **Slicers :**

* **Month and day**
* **Device type** (desktop, mobile, tablet)
* **Country** (e.g., Germany, Japan, Ukraine, etc.)

### **2. Page 2:**

#### **Pie chart : Count of total\_listens by age**

* Displays the distribution of listens by user age.

#### **Table : Song listen durations**

* Song listen durations Table with tracks, average listening duration (avg\_listen\_duration) and total duration (duration\_ms).

#### **Comparison of user age and track age :**

* Separate panel for all users.
* Displays track age (in 2024) and listener age.
* Used for preference analysis: do young people listen to old tracks or vice versa.

## **5. Conclusion**

During the coursework, a complete data storage and processing architecture for a music streaming service was implemented. OLTP ensures structured storage of raw user and reference data, while OLAP allows for solving a wide range of business intelligence tasks. The final system provides answers to key analytical questions without additional configuration, and its architecture can be adapted for real business needs.

The presented approach demonstrates the practical application of data warehouse design principles and ETL processes and can be used as a basis for more complex systems.