Project 1

Movie Database

CmpE 321, Introductin to Database Systems, Spring 2023

Simar Achment Kechagia - 2020400378
Bahadır Gezer - 2020400039

2 April 2023

Contents

1 Introduction

1.1 Definitions

2 Program Execution

1. **Data distribution:** The master thread reads in the input document file and evenly distributes the data to the slave threads. The slave threads each receive a portion of the data and print their rank and the number of sentences they have received.

2. **Bigram and unigram counting:** Each slave thread counts the bigrams and unigrams in the sentences it received. The slave threads perform this calculation in parallel.

3. **Data merging:** The slave threads send their calculated bigram and unigram counts back to the master thread, which merges the data by summing the counts of the same bigrams and unigrams. There are two options for merging the data, specified by the ”–merge_method” flag:

   - **MASTER:** After each slave thread has finished calculating the data, they send it to the master thread. The master thread is responsible for merging the data by summing the counts of the same bigrams and unigrams.
   - **WORKERS:** Instead of passing the calculated data to the master thread, each slave thread gathers the data from the previous slave thread, merges that data with its own data, and passes it to the next slave thread. The last slave thread passes the final data to the master thread.

4. **Conditional probability calculation:** The master thread uses the merged bigram and unigram counts to calculate the conditional probabilities for each pair of words in the test data file.

5. **Program termination:** The program terminates once all calculations have been completed and the output has been generated.

2.1 Input

2.2 Output

3 Program Structure

4 Examples

```
a
b
c
d
e
f
```

5 Improvements and Extensions

6 Difficulties Encountered

7 Conclusion

A SQL DDL Statements

A.1 createTables.sql

```sql
CREATE TABLE Audience (
    username VARCHAR(255),
    password VARCHAR(255) NOT NULL,
    name VARCHAR(255),
    surname VARCHAR(255),
    PRIMARY KEY (username)
);
CREATE TABLE Genre (
    id INT,
    name VARCHAR(255) UNIQUE,
    PRIMARY KEY (id)
);
CREATE TABLE Database_Managers (
    username VARCHAR(255),
    password VARCHAR(255),
    PRIMARY KEY (username)
);
CREATE TABLE Rating_Platform (
    id INT,
    name VARCHAR(255) UNIQUE,
    PRIMARY KEY (id)
);
CREATE TABLE Theater (
    id INT,
    name VARCHAR(255),
    capacity INT,
    district VARCHAR(255),
    PRIMARY KEY (id)
);
CREATE TABLE Director (
    username VARCHAR(255),
    password VARCHAR(255) NOT NULL,
    name VARCHAR(255),
    surname VARCHAR(255),
    nation VARCHAR(255) NOT NULL,
    platform_id INT,
    PRIMARY KEY (username),
    FOREIGN KEY (platform_id)
        REFERENCES Rating_Platform (id)
        ON DELETE SET NULL ON UPDATE CASCADE
);
CREATE TABLE Movie (
    id INT,
    name VARCHAR(255),
    duration INT,
    overall_rating REAL,
    director_name VARCHAR(255) NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (director_name)
        REFERENCES Director (username)
        ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Movie_Session (
    id INT,
    movie_id INT NOT NULL,
    theater_id INT NOT NULL,
    time_slot INT NOT NULL,
    date DATE NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (movie_id)
        REFERENCES Movie (id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (theater_id)
        REFERENCES Theater (id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Ticket (
    username VARCHAR(255),
    session_id INT,
    PRIMARY KEY (username , session_id),
    FOREIGN KEY (username)
        REFERENCES Audience (username)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (session_id)
        REFERENCES Movie_Session (id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Platform_Subscription (
    username VARCHAR(255),
    platform_id INT,
    PRIMARY KEY (username , platform_id),
    FOREIGN KEY (username)
        REFERENCES Audience (username)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (platform_id)
        REFERENCES Rating_Platform (id)
        ON DELETE CASCADE ON UPDATE CASCADE
```

```
);
CREATE TABLE Movie_Ratings (
    username VARCHAR(255),
    movie_id INT,
    rating REAL,
    PRIMARY KEY (username , movie_id),
    FOREIGN KEY (username)
        REFERENCES Audience (username)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (movie_id)
        REFERENCES Movie (id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Movie_Genre (
    movie_id INT,
    genre_id INT,
    PRIMARY KEY (movie_id , genre_id),
    FOREIGN KEY (movie_id)
        REFERENCES Movie (id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (genre_id)
        REFERENCES Genre (id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Movie_Predecessor (
    predecessor_id INT,
    successor_id INT,
    PRIMARY KEY (predecessor_id , successor_id),
    FOREIGN KEY (predecessor_id)
        REFERENCES Movie (id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (successor_id)
        REFERENCES Movie (id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

A.2 dropTables.sql

```sql
DROP TABLE Movie_Predecessor;
DROP TABLE Movie_Genre;
DROP TABLE Movie_Ratings;
DROP TABLE Platform_Subscription;
DROP TABLE Ticket;
DROP TABLE Movie_Session;
DROP TABLE Movie;
DROP TABLE Director;
DROP TABLE Theater;
DROP TABLE Rating_Platform;
DROP TABLE Database_Managers;
DROP TABLE Genre;
DROP TABLE Audience;
```