CS5200 Fall 2020: Practicum 2

Chandra Davis, Evan Douglass

Overview

We've decided to work with SQLite for this practicum. As such, to work with these files you will need SQLite installed on your machine. The data we are using is provided by IMDB at https://datasets.imdbws.com/

Download all the compressed tsv files into a folder called `data/`. You can leave them in their compressed format, as that is how we will read them in to this notebook.

Setup

Before working with the data we need to create a place to store it. This section will set up any constants needed later and connect to a SQLite database.

```
# Setup SQLite database
library(RSQLite)
DB_NAME <- "imdb-data.db"
conn <- dbConnect(RSQLite::SQLite(), DB_NAME)
```

!!!Need to download the files from the web before completing the next chunk!!!

Files were manually downloaded and saved to the data folder for testing

```
# Simplify data extraction for later
getAkasData <- function() {
  read.table("data/title.akas.tsv.gz",
             sep='\t', quote = "\"", fill=TRUE, header=TRUE, encoding="UTF-8")
}

getNameBasicsData <- function() {
  read.table("data/name.basics.tsv.gz",
             sep='\t', quote = "\"", fill=TRUE, header=TRUE, encoding="UTF-8")
}

getTitleBasicsData <- function() {
  read.table("data/title.basics.tsv.gz",
             sep='\t', quote = "\"", fill=TRUE, header=TRUE, encoding="UTF-8")
}

getCrewData <- function() {
  read.table("data/title.crew.tsv.gz",
             sep='\t', fill=TRUE, header=TRUE, encoding="UTF-8")
}

getEpisodeData <- function() {
```

```
 read.table("data/title.episode.tsv.gz",
            sep='\t', fill=TRUE, header=TRUE, encoding="UTF-8")
}

getPrincipalsData <- function() {
  read.table("data/title.principals.tsv.gz",
             sep='\t', fill=TRUE, header=TRUE, encoding="UTF-8")
}

getRatingsData <- function() {
  read.table("data/title.ratings.tsv.gz",
             sep='\t', fill=TRUE, header=TRUE, encoding="UTF-8")
}
```

The following chunk creates the database schema using CREATE TABLE statements. This schema was designed for the relational model based on the above datasets.

```
# First we need to remove any existing data,
# for example, if this has been run before.
drop_sql <- function(table_name) {
  paste("DROP TABLE IF EXISTS ", table_name, ";", sep="")
}

# Saving rows affected to a var prevents output below
rows_aff <- dbExecute(conn, "PRAGMA foreign_keys = OFF;") # Avoid FK checks
curr_tables <- dbListTables(conn)
for (table in curr_tables) {
  dbExecute(conn, drop_sql(table))
}
rows_aff <- dbExecute(conn, "PRAGMA foreign_keys = ON;")

# Now we can create the tables
build_table <- function(table_def) {
  CREATE <- "CREATE TABLE IF NOT EXISTS"
  paste(CREATE, table_def)
}

# Build table definition list
tables <- c(
  build_table(
    "Title_Type (
      format_id INTEGER PRIMARY KEY,
      format TEXT NOT NULL
    );"
  ),
  build_table(
    "Media (
      tconst TEXT PRIMARY KEY,
      format_id INTEGER NOT NULL,
      primaryTitle TEXT,
      originalTitle TEXT,
      isAdult INTEGER, -- 0=false, else true
      startYear INTEGER,
```

```
    endYear INTEGER,
    runtimeMins INTEGER,
    FOREIGN KEY (format_id) REFERENCES Title_Type(format_id)
  );"
),
build_table(
  "Ratings (
    tconst TEXT PRIMARY KEY,
    averageRating REAL
    numVotes INTEGER,
    FOREIGN KEY (tconst) REFERENCES Media(tconst)
  );"
),
build_table(
  "Episode (
    tconst TEXT PRIMARY KEY,
    parentTconst TEXT NOT NULL,
    seasonNumer INTEGER,
    episodeNumber INTEGER,
    FOREIGN KEY (tconst) REFERENCES Media(tconst),
    FOREIGN KEY (parentTconst) REFERENCES Media(tconst)
  );"
),
build_table(
  "Genres (
    genre_id INTEGER PRIMARY KEY,
    genre TEXT NOT NULL
  );"
),
build_table(
  "Media_Genres (
    mg_id INTEGER PRIMARY KEY,
    tconst TEXT NOT NULL,
    genre_id INTEGER NOT NULL,
    FOREIGN KEY (tconst) REFERENCES Media(tconst),
    FOREIGN KEY (genre_id) REFERENCES Genres(genre_id)
  );"
),
build_table(
  "People (
    nconst TEXT PRIMARY KEY,
    primaryName TEXT,
    birthYear INTEGER,
    deathYear INTEGER,
    age INTEGER,
    numMovies INTEGER
  );"
),
build_table(
  "Known_For_Titles (
    kt_id INTEGER PRIMARY KEY,
    nconst TEXT NOT NULL,
    tconst TEXT NOT NULL,
```

```
    FOREIGN KEY (tconst) REFERENCES Media(tconst),
    FOREIGN KEY (nconst) REFERENCES People(nconst)
  );"
),
build_table(
  "Professions (
    prof_id INTEGER PRIMARY KEY,
    prof_title TEXT NOT NULL
  );"
),
build_table(
  "Primary_Profession (
    pp_id INTEGER PRIMARY KEY,
    nconst TEXT NOT NULL,
    prof_id INTEGER NOT NULL,
    FOREIGN KEY (nconst) REFERENCES People(nconst),
    FOREIGN KEY (prof_id) REFERENCES Professions(prof_id)
  );"
),
build_table(
  "Crew (
    tconst TEXT NOT NULL,
    ordering INTEGER NOT NULL,
    nconst TEXT NOT NULL,
    category TEXT,
    job TEXT,
    characters TEXT,
    PRIMARY KEY (tconst, ordering),
    FOREIGN KEY (tconst) REFERENCES Media(tconst),
    FOREIGN KEY (nconst) REFERENCES People(nconst)
  );"
),
build_table(
  "Media_Directors (
    md_id INTEGER PRIMARY KEY,
    tconst TEXT NOT NULL,
    nconst TEXT NOT NULL,
    FOREIGN KEY (tconst) REFERENCES Media(tconst),
    FOREIGN KEY (nconst) REFERENCES People(nconst)
  );"
),
build_table(
  "Media_Writers (
    mw_id INTEGER PRIMARY KEY,
    tconst TEXT NOT NULL,
    nconst TEXT NOT NULL,
    FOREIGN KEY (tconst) REFERENCES Media(tconst),
    FOREIGN KEY (nconst) REFERENCES People(nconst)
  );"
),
build_table(
  "Also_Known_As (
    tconst TEXT NOT NULL,
```

```
      ordering INTEGER NOT NULL,
      title TEXT,
      region TEXT,
      language TEXT,
      isOriginalTitle INTEGER, -- 0=false, else true
      PRIMARY KEY (tconst, ordering),
      FOREIGN KEY (tconst) REFERENCES Media(tconst)
    );"
  ),
  build_table(
    "Title_Types (
      type_id INTEGER PRIMARY KEY,
      type TEXT NOT NULL
    );"
  ),
  build_table(
    "Aka_Types (
      akt_id INTEGER PRIMARY KEY,
      tconst TEXT NOT NULL,
      ordering INTEGER NOT NULL,
      type_id INTEGER NOT NULL,
      -- Media table PK tconst has ref. integrity through AKA Table
      FOREIGN KEY (tconst, ordering) REFERENCES Also_Known_As(tconst, ordering),
      FOREIGN KEY (type_id) REFERENCES Title_Types(type_id)
    );"
  ),
  build_table(
    "Attributes (
      att_id INTEGER PRIMARY KEY,
      att_name TEXT NOT NULL
    );"
  ),
  build_table(
    "Aka_Attributes (
      aka_id INTEGER PRIMARY KEY,
      tconst TEXT NOT NULL,
      ordering INTEGER NOT NULL,
      att_id INTEGER NOT NULL,
      -- Media table PK tconst has ref. integrity through AKA Table
      FOREIGN KEY (tconst, ordering) REFERENCES Also_Known_As(tconst, ordering),
      FOREIGN KEY (att_id) REFERENCES Attributes(att_id)
    );"
  )
)

# Actually create the tables
for (table_stmt in tables) {
  dbExecute(conn, table_stmt)
}

# Remove later - for testing
dbListTables(conn)
```

```
##  [1] "Aka_Attributes"     "Aka_Types"          "Also_Known_As"
##  [4] "Attributes"         "Crew"               "Episode"
##  [7] "Genres"             "Known_For_Titles"   "Media"
## [10] "Media_Directors"    "Media_Genres"       "Media_Writers"
## [13] "People"             "Primary_Profession" "Professions"
## [16] "Ratings"            "Title_Type"         "Title_Types"
```