DATABASE DESIGN

Assignment 1

Abstract

Redesigning the Movies Database Scenario.

Venkata Sai Praneeth Kandula
Yashwanth Lingareddy
Saikumar Nanjala
Sanketh Bhagavanthi
Vinay Murthy

University of South Florida

The following Critical Assumptions were made in the redesign process:

- A movie can have more than one director, writer, and genre
- A movie can have one or more than one cast members
- A movie can have one or more than one Locales
- A movie can be linked with one or more than one keywords and taglines
- A movie/theater can be rated/reviewed by one or more than one Fans
- A fan can rate/review one or more than one movies/theaters
- A theater group should have at least one physical theater location
- A theater should have at least one screen
- The rates for add-on and patrons are same for all movie screens
- No of Tickets in transaction details should be always 1 or greater than 1
- No transaction details will be recorded for zero tickets in 'No of Tickets' column
- Each transaction detail should be selected with a patron. Default patron will be linked in case of no selection
- Each show time can should be selected with an add-on. Default add-on will be linked in case of no selection

Changes done to existing database:

With the assumptions above, we have made following changes to the existing database as part of the normalization process. The entire Database Design now adheres to the Third- Normal Form (3F):

- Writer table will have only list of writers and MOVIE_WRITER table will have relationship between Movie and Writer
- Cast table will have only list of Cast members and MOVIE_CAST table will have relationship between Movie and Cast
- Director table will have only list of Directors and MOVIE_DIRECTOR table will have relationship between Movie and Director
- Genre table will have only list of Genres and MOVIE_GENRE table will have relationship between Movie and Genre

- KEY_WORD table will have only list of key words and MOVIE_KEYWORD table will have relationship between Movie and Key words
- Locale table will have only list of locales and MOVIE_LOCALE table will have relationship between Movie and Locale
- Tagline column table has been removed from the movie table.
- TAG_LINE table will have only list of tag lines and MOVIE_TAGLINE table will have relationship between Movie and TagLine
- LOGIN table will have login details of all the fans and Login Id will linked to Fan table

We have added the following new tables to the existing database design as part of the redesign process to address the 3 design specifications provided in the question:

1) To allow users to view the name and location of theatres, we added two new tables.
   - THEATER_GROUP which has a unique identifier for each theater group/Chain.
   - THEATER which contains the theater details for each theatre group, including the no. of screens and the address of each theater location.
   - SCREEN which contains data about the names of the screens in each theater.

2) To allow users to see the show times for movies playing at different theaters including historical show times.
   - MOVIE_SHOWTIME which contains the show time, movie and screen id for each show time, also contains the add on available of any for each show.
   - ADD_ON which contains data about the add-on name and price.

3) To allow users to sign-up for a free account so that they can review movies and theaters as well as purchase tickets.
   - LOGIN which stores the userid, email and password of users who have signed-up. These users can now review theaters and movies.
   - FAN_MOVIE_RATING which stores the ratings and reviews of movies given by users.
   - THEATER_RATING which has the ratings and reviews of theaters given by the users.
   - TRANSACTION_DETAIL which keeps track of all transactions, it contains purchase id and details about the type of patron and the no. of tickets purchased in each transaction.

- PURCHASE_HISTORY which contains the historical purchase data of each user who has an account.
- PATRON which has details regarding the type of patron and the fare for each type.

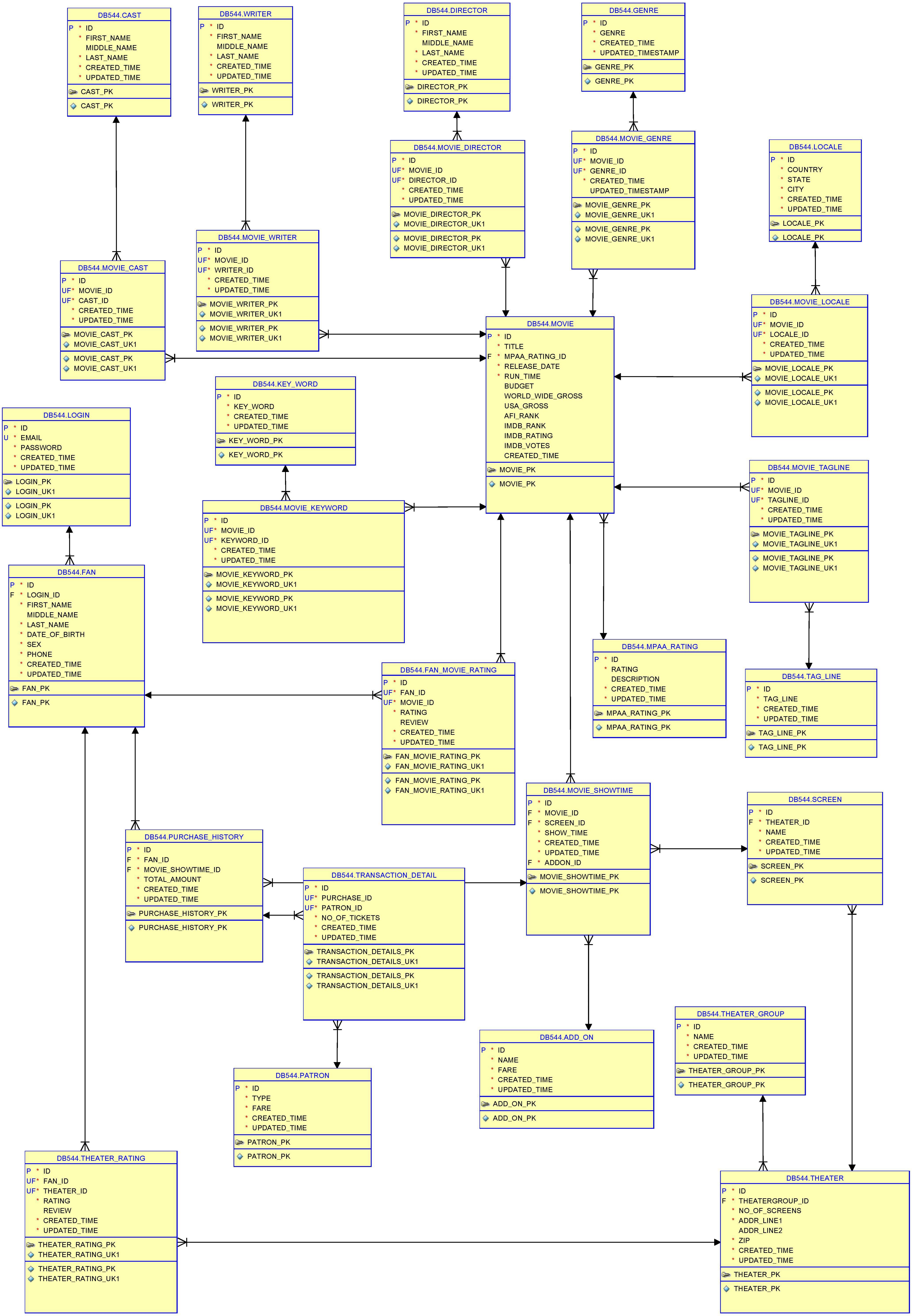
Interesting queries

1. If a user who has already provided a rating and review for a movie tries to do it again, this query will update the rating of the movie and append the new review to the existing review.

```
MERGE INTO FAN_MOVIE_RATING dest
  USING (SELECT 1 FAN_ID, 1 MOVIE_ID, 4.2 RATING, 'Good Movie' REVIEW FROM dual) src
    ON (dest.FAN_ID = src.FAN_ID AND dest.MOVIE_ID = src.MOVIE_ID)
WHEN MATCHED THEN
  UPDATE SET REVIEW = REVIEW || src.REVIEW, RATING = src.RATING
WHEN NOT MATCHED THEN
  INSERT (FAN_ID, MOVIE_ID, RATING, REVIEW)
    VALUES (src.FAN_ID, src.MOVIE_ID, src.RATING, src.REVIEW);
```

2. This query shows a list of all the users with their total amount spent in all theatres sorted in Decreasing order.

```
SELECT
  SUM(PH.TOTAL_AMOUNT) AS TOTAL_AMOUNT,
  PH.FAN_ID,
  F.FIRST_NAME || F.MIDDLE_NAME || F.LAST_NAME AS NAME
  FROM
  FAN F INNER JOIN PURCHASE_HISTORY PH ON F.FAN_ID = PH.FAN_ID
  GROUP BY
    PH.FAN_ID
  ORDER BY
    FAN_ID DESC;
```



```
DB544.CAST
P * ID
  * FIRST_NAME
    MIDDLE_NAME
  * LAST_NAME
  * CREATED_TIME
  * UPDATED_TIME
⚷ CAST_PK
◆ CAST_PK
```

```
DB544.MOVIE_CAST
P  * ID
UF * MOVIE_ID
UF * CAST_ID
   * CREATED_TIME
   * UPDATED_TIME
⚷ MOVIE_CAST_PK
◆ MOVIE_CAST_UK1
◆ MOVIE_CAST_PK
◆ MOVIE_CAST_UK1
```

```
DB544.LOGIN
P * ID
U * EMAIL
  * PASSWORD
  * CREATED_TIME
  * UPDATED_TIME
⚷ LOGIN_PK
◆ LOGIN_UK1
◆ LOGIN_PK
◆ LOGIN_UK1
```

```
DB544.FAN
P * ID
F * LOGIN_ID
  * FIRST_NAME
    MIDDLE_NAME
  * LAST_NAME
  * DATE_OF_BIRTH
  * SEX
  * PHONE
  * CREATED_TIME
  * UPDATED_TIME
⚷ FAN_PK
◆ FAN_PK
```

```
DB544.THEATER_RATING
P  * ID
UF * FAN_ID
UF * THEATER_ID
   * RATING
     REVIEW
   * CREATED_TIME
   * UPDATED_TIME
⚷ THEATER_RATING_PK
◆ THEATER_RATING_UK1
◆ THEATER_RATING_PK
◆ THEATER_RATING_UK1
```

```
DB544.WRITER
P * ID
  * FIRST_NAME
    MIDDLE_NAME
  * LAST_NAME
  * CREATED_TIME
  * UPDATED_TIME
⚷ WRITER_PK
◆ WRITER_PK
```

```
DB544.MOVIE_WRITER
P  * ID
UF * MOVIE_ID
UF * WRITER_ID
   * CREATED_TIME
   * UPDATED_TIME
⚷ MOVIE_WRITER_PK
◆ MOVIE_WRITER_UK1
◆ MOVIE_WRITER_PK
◆ MOVIE_WRITER_UK1
```

```
DB544.KEY_WORD
P * ID
  * KEY_WORD
  * CREATED_TIME
  * UPDATED_TIME
⚷ KEY_WORD_PK
◆ KEY_WORD_PK
```

```
DB544.MOVIE_KEYWORD
P  * ID
UF * MOVIE_ID
UF * KEYWORD_ID
   * CREATED_TIME
   * UPDATED_TIME
⚷ MOVIE_KEYWORD_PK
◆ MOVIE_KEYWORD_UK1
◆ MOVIE_KEYWORD_PK
◆ MOVIE_KEYWORD_UK1
```

```
DB544.PURCHASE_HISTORY
P * ID
F * FAN_ID
F * MOVIE_SHOWTIME_ID
  * TOTAL_AMOUNT
  * CREATED_TIME
  * UPDATED_TIME
⚷ PURCHASE_HISTORY_PK
◆ PURCHASE_HISTORY_PK
```

```
DB544.TRANSACTION_DETAIL
P  * ID
UF * PURCHASE_ID
UF * PATRON_ID
   * NO_OF_TICKETS
   * CREATED_TIME
   * UPDATED_TIME
⚷ TRANSACTION_DETAILS_PK
◆ TRANSACTION_DETAILS_UK1
◆ TRANSACTION_DETAILS_PK
◆ TRANSACTION_DETAILS_UK1
```

```
DB544.PATRON
P * ID
  * TYPE
  * FARE
  * CREATED_TIME
  * UPDATED_TIME
⚷ PATRON_PK
◆ PATRON_PK
```

```
DB544.DIRECTOR
P * ID
  * FIRST_NAME
    MIDDLE_NAME
  * LAST_NAME
  * CREATED_TIME
  * UPDATED_TIME
⚷ DIRECTOR_PK
◆ DIRECTOR_PK
```

```
DB544.MOVIE_DIRECTOR
P  * ID
UF * MOVIE_ID
UF * DIRECTOR_ID
   * CREATED_TIME
   * UPDATED_TIME
⚷ MOVIE_DIRECTOR_PK
◆ MOVIE_DIRECTOR_UK1
◆ MOVIE_DIRECTOR_PK
◆ MOVIE_DIRECTOR_UK1
```

```
DB544.MOVIE
P * ID
  * TITLE
F * MPAA_RATING_ID
  * RELEASE_DATE
  * RUN_TIME
    BUDGET
    WORLD_WIDE_GROSS
    USA_GROSS
    AFI_RANK
    IMDB_RANK
    IMDB_RATING
    IMDB_VOTES
    CREATED_TIME
⚷ MOVIE_PK
◆ MOVIE_PK
```

```
DB544.FAN_MOVIE_RATING
P  * ID
UF * FAN_ID
UF * MOVIE_ID
   * RATING
     REVIEW
   * CREATED_TIME
     UPDATED_TIME
⚷ FAN_MOVIE_RATING_PK
◆ FAN_MOVIE_RATING_UK1
◆ FAN_MOVIE_RATING_PK
◆ FAN_MOVIE_RATING_UK1
```

```
DB544.MOVIE_SHOWTIME
P * ID
F * MOVIE_ID
F * SCREEN_ID
  * SHOW_TIME
  * CREATED_TIME
  * UPDATED_TIME
F * ADDON_ID
⚷ MOVIE_SHOWTIME_PK
◆ MOVIE_SHOWTIME_PK
```

```
DB544.ADD_ON
P * ID
  * NAME
  * FARE
  * CREATED_TIME
  * UPDATED_TIME
⚷ ADD_ON_PK
◆ ADD_ON_PK
```

```
DB544.GENRE
P * ID
  * GENRE
  * CREATED_TIME
  * UPDATED_TIMESTAMP
⚷ GENRE_PK
◆ GENRE_PK
```

```
DB544.MOVIE_GENRE
P  * ID
UF * MOVIE_ID
UF * GENRE_ID
   * CREATED_TIME
     UPDATED_TIMESTAMP
⚷ MOVIE_GENRE_PK
◆ MOVIE_GENRE_UK1
◆ MOVIE_GENRE_PK
◆ MOVIE_GENRE_UK1
```

```
DB544.MPAA_RATING
P * ID
  * RATING
    DESCRIPTION
  * CREATED_TIME
  * UPDATED_TIME
⚷ MPAA_RATING_PK
◆ MPAA_RATING_PK
```

```
DB544.LOCALE
P * ID
  * COUNTRY
  * STATE
  * CITY
  * CREATED_TIME
  * UPDATED_TIME
⚷ LOCALE_PK
◆ LOCALE_PK
```

```
DB544.MOVIE_LOCALE
P  * ID
UF * MOVIE_ID
UF * LOCALE_ID
   * CREATED_TIME
   * UPDATED_TIME
⚷ MOVIE_LOCALE_PK
◆ MOVIE_LOCALE_UK1
◆ MOVIE_LOCALE_PK
◆ MOVIE_LOCALE_UK1
```

```
DB544.MOVIE_TAGLINE
P  * ID
UF * MOVIE_ID
UF * TAGLINE_ID
   * CREATED_TIME
   * UPDATED_TIME
⚷ MOVIE_TAGLINE_PK
◆ MOVIE_TAGLINE_UK1
◆ MOVIE_TAGLINE_PK
◆ MOVIE_TAGLINE_UK1
```

```
DB544.TAG_LINE
P * ID
  * TAG_LINE
  * CREATED_TIME
  * UPDATED_TIME
⚷ TAG_LINE_PK
◆ TAG_LINE_PK
```

```
DB544.SCREEN
P * ID
F * THEATER_ID
  * NAME
  * CREATED_TIME
  * UPDATED_TIME
⚷ SCREEN_PK
◆ SCREEN_PK
```

```
DB544.THEATER_GROUP
P * ID
  * NAME
  * CREATED_TIME
  * UPDATED_TIME
⚷ THEATER_GROUP_PK
◆ THEATER_GROUP_PK
```

```
DB544.THEATER
P * ID
F * THEATERGROUP_ID
  * NO_OF_SCREENS
  * ADDR_LINE1
    ADDR_LINE2
  * ZIP
  * CREATED_TIME
  * UPDATED_TIME
⚷ THEATER_PK
◆ THEATER_PK
```