Maggie Baumgardner, Sruti Chigurupati, Trey Hakanson, Elizabeth Gilbert

CSE 3421

04/20/2018

Final Project User Guide

User Guide

Database Overview

All primary keys are unique, and cannot be null.

Entity: Album

Attributes:

- album_id
- album_title
- release_date
- genre

The `album_id` is a unique integer ID assigned to each album, used as the primary key of the album relation. The `album_title` is the 155 character or less string title of the album and cannot be null. The `release_date` is the date time of the album's release and cannot be null. The `genre` is a 155 character or less string label describing the album genre and cannot be null.

Entity: artist_albums

Attributes:

- artist_id
- album_id

The `artist_id` is unique integer ID assigned to each artist, used as a primary key to the `artist_albums` relation and foreign key to the artist relation and cannot be null. The `album_id` is a unique integer ID assigned to each album, used as a primary key to the `artist_albums` relation and foreign key to the album relation and cannot be null.

Entity: artist

Attributes:

- artist_id

- name

The `artist_id` is unique integer ID assigned to each artist, used as the primary key of the artist relation. The `name` is a 155 character or less string name of the artist and cannot be null.

Entity: track

Attributes:

- title
- album_id
- number
- length
- size_bytes

The `title` is the 155 character or less string title of the track and cannot be null. The `album_id` is a unique small integer ID assigned to each album, used as a primary key to the track relation and foreign key to the album relation and cannot be null. The `number` is the integer track number of the track in its album. The `length` is the decimal length of the track playtime in minutes and cannot be null. The `size_bytes` is a big integer representing the size of the track in bytes.

Entity: media

Attributes:

- media_id
- type
- album_id

The `media_id` is a unique integer ID assigned to each media, used as a primary key in the media relation and foreign key to the checkout relation. The `type` is an 8 character or less string labeling the media as either "physical" or "digital" and cannot be null. The `album_id` is a unique integer ID assigned to each album, used as a foreign key to the album relation and cannot be null.

Entity: checkout

Attributes:

- checkout_id
- checkout_date
- return_date
- media_id
- card_number
- due_date

The `checkout_id` is a unique integer ID assigned to the checkout session, used as a primary key to the checkout relation. The `checkout_date` is the date time at which the media was checked out, and

cannot be null. The `return_date` is the date time at which the media was returned, and is null until returned. The `media_id` is a unique integer ID assigned to each media and cannot be null. The `card_number` is a 36 character or less string identifying the person's library card, used as a foreign key to the person relation and cannot be null. The `due_date` is the date time at which the media is due to be returned to the library and cannot be null.

Entity: person

Attributes:

- card_number
- email
- first_name
- last_name
- activation_date
- num_cards
- card_number_active

The `card_number` is a 36 character or less string identifying the person's library card, used as a primary key in the person relation. The `email` is a unique 155 character or less string that is the email address of the person. The `first_name` is a 155 character or less string that is the first name of the person and is not null. The `last_name` is a 155 character or less string that is the last name of the person and is not null. The `activation_date` is the date time of when the person was created in the library system and is not null. The `num_cards` is an integer that is not null and defaults to 1 identifying the number of cards a person has. The `card_number_active` is a non-null boolean field defaulting to 1 where 1 indicates that a card number is active and 0 indicates that it is inactive.

Entity: employee

Attributes:

- start_date
- salary
- position
- card_number

The `start_date` is the date time that the employee started working for the library and cannot be null. The `salary` is a decimal representing the salary of the employee and cannot be null. The `position` is a 155 character or less string of the employee's job title. The `card_number` is a 36 character or less string identifying the person's library card, used as a primary key in the employee relation and foreign key to the person relation.

Entity: feedback

Attributes:

- feedback_id
- description
- date
- category
- card_number

The `feedback_id` is a unique integer identifying the feedback submission, used as a primary key of the feedback relation. The `description` is a 500 character or less string of the person's feedback and cannot be null. The `date` is the date time that the feedback was submitted at. The `category` is a 155 character or less string of the category the feedback falls under and cannot be null. The `card_number` is a 36 character or less string identifying the person's library card, used as a foreign key to the person relation.

Entity: review

Attributes:

- review_id
- stars
- title
- description
- album_id
- card_number

The `review_id` is a unique integer identifying the review submission, used as a primary key of the review relation. The `stars` is a small integer representing the number of the stars of the review. The `title` is a 155 character or less title of the review. The `description` is a 500 character or less description of the review. The `album_id` is a unique integer ID assigned to each album, used as a foreign key to the album relation and cannot be null. The `card_number` is a 36 character or less string identifying the person's library card, used as a foreign key to the person relation and cannot be null.

Sample Queries

Sample queries for each checkpoint can be found as standalone files in the `sql` directory.

Relation Algebra Queries from Checkpoint 2

a. Find the titles of all songs by ARTIST released before YEAR

$$\pi_{track\_title}(\sigma_{release\_date < YEAR}(\sigma_{name = ARTIST} \text{ ARTIST*ARTIST\_ALBUMS*ALBUM*TRACK}))$$

b. Give all the albums and their date of their checkout from a single patron (you choose how to designate the patron)

$$\pi_{\text{album\_title, checkout\_date}}\sigma_{\text{card\_number = CARD NUMBER}}(\text{PERSON*MEDIA*ALBUM})$$

c. List all the albums and their unique identifiers with less than 5 copies held by the library.

$$\pi_{\text{album\_id, title}}(\sigma_{\text{count < 5}}({}_{\text{album\_id}}F_{\text{COUNT album\_id}}(\text{ALBUM*MEDIA})))$$

d. Give all the patrons who checked out an album by ARTIST and the albums they checked out.

$$\pi_{\text{first\_name, last\_name, album\_title}}(\sigma_{\text{checkout\_date != NULL}}(\sigma_{\text{name = ARTIST}}$$
$$\text{ARTIST*ARTIST\_ALBUMS*ALBUM*PERSON*MEDIA}))$$

e. Find the total number of albums checked out by a single patron (you choose how to designate the patron)

$$F_{\text{COUNT album\_id}}(\sigma_{\text{card\_number = CARD NUMBER}}\ \text{MEDIA})$$

f. Find the patron who has checked out the most albums and the total number of albums they have checked out.

$$F_{\text{MAX count}}({}_{\text{card\_number}}F_{\text{COUNT album\_id}}(\sigma_{\text{checkout\_date != NULL}}\ \text{MEDIA}))$$

Additional Relational Algebra Queries from Checkpoint 2

a. Number of feedbacks each patron as given

$${}_{\text{card\_number}}F_{\text{COUNT feedback\_id}}(\text{PERSON*FEEDBACK})$$

b. Average star rating for each album

$${}_{\text{album}}F_{\text{AVERAGE stars}}(\text{REVIEW*ALBUM})$$

c. How many copies does each album have (physical and digital)?

$${}_{\text{album\_id}}F_{\text{COUNT media\_id}}(\text{ALBUM*MEDIA})$$

Queries from Checkpoint 3

a. Find the titles of all tracks by ARTIST released before YEAR

```
SELECT t.title
    FROM track t
        JOIN album a ON a.album_id = t.album_id
        JOIN artist_albums aa ON aa.album_id = a.album_id
        JOIN artist art ON art.artist_id = aa.artist_id
    WHERE name = 'AC/DC' AND
          release_date < 1982;
```

b. Give all the albums and their date of their checkout from a single patron (you choose how to designate the patron)

```
SELECT album_title, release_date
   FROM Album AS a
    JOIN media AS m ON a.album_id = m.album_id
      JOIN checkout AS c ON m.media_id = c.media_id
    JOIN person AS p ON p.card_number = c.card_number
   WHERE p.card_number = '7eddeba0-c2cc-4d7a-a1b1-7248c9dbda63';
```

c. List all the albums and their unique identifiers with less than 5 copies held by the library.

```
SELECT a.album_id, a.album_title
    FROM album a
        JOIN media m ON a.album_id = m.album_id
    GROUP BY a.album_id
    HAVING COUNT(*) < 5;
```

d. Give all the patrons who checked out an album by ARTIST and the albums they checked out.

```
SELECT album_title, first_name, last_name
   FROM Album AS a
    JOIN Media AS m ON a.album_id = m.album_id
      JOIN checkout AS c ON m.media_id = c.media_id
    JOIN Person AS p ON p.card_number = c.card_number
    JOIN Artist_Albums aa ON a.album_id = aa.album_id
    JOIN Artist a2 ON aa.artist_id = a2.artist_id
   WHERE a2.name = "AC/DC";
```

e. Find the total number of albums checked out by a single patron (you choose how to designate the patron)

```
SELECT COUNT(*)
    FROM media AS m
        JOIN checkout AS c ON m.media_id = c.media_id
        JOIN person AS p ON p.card_number = c.card_number
        WHERE p.card_number = '7eddeba0-c2cc-4d7a-a1b1-7248c9dbda63';
```

f. Find the patron who has checked out the most albums and the total number of albums they have checked out.

```
SELECT full_name, MAX(count)
    FROM (
        SELECT first_name || ' ' || last_name full_name, COUNT(*) count
            FROM media m
                JOIN checkout AS c ON m.media_id = c.media_id
                JOIN person p ON p.card_number = c.card_number
            GROUP BY p.card_number
            ORDER BY COUNT(*) DESC
    );
```

Additional Queries from Checkpoint 3

a. Number of feedbacks each patron as given

```
SELECT first_name || ' ' || last_name, COUNT(*)
    FROM person p
        JOIN feedback f ON p.card_number = f.card_number
    GROUP BY p.card_number;
```

b. Average star rating for each album

```
SELECT album_title, AVG(stars)
   FROM review r
      JOIN album a ON a.album_id = r.album_id
   GROUP BY a.album_id;
```

c. How many copies does each album have (physical and digital)?

```
SELECT a.album_title, COUNT(*)
    FROM album a
        JOIN media m ON a.album_id = m.album_id
    GROUP BY a.album_id;
```

Queries from Checkpoint 4

a. Provide a list of patron names, along with the total combined running time of all the albums they have checked out.

```
SELECT full_name, SUM(length)
    FROM (
        SELECT first_name || ' ' || last_name full_name,
               p.card_number card_number,
               m.album_id media_album_id
```

```
            FROM media m
                JOIN checkout c ON c.media_id = m.media_id
                JOIN person p ON p.card_number = c.card_number
    ) JOIN track t on media_album_id = t.album_id
    GROUP BY card_number;
```

b. Provide a list of patron names and email addresses for patrons who have checked out more albums than the average patron.

```
SELECT first_name || ' ' || last_name full_name, email
   FROM person p
       JOIN checkout c ON c.card_number = p.card_number
   GROUP BY first_name, last_name, email
   HAVING COUNT(*) > (
       SELECT AVG(count)
           FROM (
               SELECT email, COUNT(*) count
                   FROM person p
                       JOIN checkout c ON c.card_number = p.card_number
                   GROUP BY email
           )
   );
```

c. Provide a list of the albums in the database and associated total copies lent to patrons, sorted from the album that has been lent the most to the album that has been lent the least.

```
SELECT album_title, COUNT(*)
    FROM album a
        JOIN media m ON a.album_id = m.album_id
        JOIN checkout c ON c.media_id = m.media_id
    WHERE return_date IS NULL
    GROUP BY album_title
    ORDER BY COUNT(*) DESC;
```

d. Provide a list of the titles in the database and associated totals for copies checked out to customers, sorted from the title that has been checked out the highest amount to the title checked out the smallest.

```
SELECT a.album_title, COUNT(*)
    FROM album a
        JOIN media m ON a.album_id = m.album_id
        JOIN checkout c ON m.media_id = c.media_id
    GROUP BY a.album_title
    ORDER BY COUNT(*) DESC;
```

e. Find the most popular artist in the database (i.e. the one who has had the most lent albums)

```
SELECT name
```

```
    FROM album a
        JOIN media m ON a.album_id = m.album_id
        JOIN checkout c ON c.media_id = m.media_id
        JOIN artist_albums aa ON aa.album_id = a.album_id
        JOIN artist art ON art.artist_id = aa.artist_id
    WHERE return_date IS NULL
    GROUP BY name
    ORDER BY COUNT(*) DESC
    LIMIT 1;
```

f. Find the most listened to artist in the database (use the running time of the album and number of times the album has been lent out to calculate)

```
SELECT name
    FROM album a
        JOIN media m ON a.album_id = m.album_id
        JOIN checkout c ON c.media_id = m.media_id
        JOIN artist_albums aa ON aa.album_id = a.album_id
        JOIN artist art ON art.artist_id = aa.artist_id
        JOIN track t ON t.album_id = a.album_id
    GROUP BY name
    ORDER BY SUM(length) DESC
    LIMIT 1;
```

g. Provide a list of customer information for patrons who have checked out anything by the most listened to artist in the database.

```
SELECT p.card_number, email, first_name || ' ' || last_name full_name
    FROM person p
        JOIN checkout c ON p.card_number = c.card_number
        JOIN media m ON c.media_id = m.media_id
        JOIN album a ON a.album_id = m.album_id
        JOIN artist_albums aa ON a.album_id = aa.album_id
        JOIN artist art ON aa.artist_id = art.artist_id
    WHERE (
        SELECT name
            FROM album a
                JOIN media m ON a.album_id = m.album_id
                JOIN checkout c ON c.media_id = m.media_id
                JOIN artist_albums aa ON aa.album_id = a.album_id
                JOIN artist art ON art.artist_id = aa.artist_id
                JOIN track t ON t.album_id = a.album_id
            GROUP BY name
            ORDER BY SUM(length) DESC
            LIMIT 1
    ) LIKE art.name;
```

h. Provide a list of artists who authored the albums checked out by customers who have checked out more albums than the average customer.

```
SELECT DISTINCT name
    FROM checkout c
        JOIN media m ON m.media_id = c.media_id
        JOIN artist_albums aa ON aa.album_id = m.album_id
        JOIN artist art ON art.artist_id = aa.artist_id
    WHERE c.card_number IN (
        SELECT p.card_number
           FROM person p
               JOIN checkout c ON c.card_number = p.card_number
           GROUP BY first_name, last_name, email
           HAVING COUNT(*) > (
               SELECT AVG(count)
                   FROM (
                       SELECT email, COUNT(*) count
                           FROM person p
                               JOIN checkout c ON c.card_number = p.card_number
                           GROUP BY email
                   )
           )
    );
```

Insertion and Deletion

Album

Insert

```
INSERT INTO album
   VALUES(1,'For Those About To Rock We Salute You',1981,'Rock');
```

Delete

```
DELETE FROM album AS a
  WHERE a.album_id = 1;
```

Must delete artist_album tuple, all tracks, all media records.

Person

Insert

```
INSERT INTO person
   VALUES('18108a24-4eac-498d-a2d3-4dc4cccf405a','jasonmorris5660@gmail.com',
          'Jason','Morris','2017-10-24 21:08:23');
```

Delete

```
DELETE FROM person AS p
  WHERE p.card_number = '18108a24-4eac-498d-a2d3-4dc4cccf405a';
```

Must delete checkout history, all feedback, all reviews, employee record if applicable.

Artist

Insert

```
INSERT INTO artist
   VALUES(1,'AC/DC');
```

Delete

```
DELETE FROM artist AS a
  WHERE a.artist_id = 1;
```

Must delete artist_album tuple.

Track

Insert

```
INSERT INTO track
   VALUES('Breaking The Rules',1,NULL,4.3799999999999998934,1,8596840);
```

Album must exist.

Delete

```
DELETE FROM track AS t
  WHERE t.title = 'Breaking The Rules' AND t.album_id = 1;
```

Media

Insert

```
INSERT INTO media
   VALUES(1,'physical',1);
```

Album must exist.

Delete

```
DELETE FROM media AS m
  WHERE m.media_id = 1;
```

Must delete checkout history.

Checkout

Insert

```
INSERT INTO checkout
   VALUES(2,'2017-08-24 03:43:51',NULL,1,'fdb627b1-23f4-4d43-8492-e5a5780d66fb',
      '2017-12-09 13:12:44');
```

Media and person must exist.

Delete

```
DELETE FROM checkout AS c
  WHERE c.checkout_id = 2;
```

Artist_Albums

Insert

```
INSERT INTO artist_albums VALUES(1,1);
```

Album and artist must exist.

Delete

```
DELETE FROM artist_albums AS aa
  WHERE aa.album_id = 1 AND aa.artist_id = 1;
```

Employee

Insert

```
INSERT INTO employee
   VALUES('2017-10-29 12:55:42',51055.999999999999998,'librarian',
      '18108a24-4eac-498d-a2d3-4dc4cccf405a');
```

Person must exist.

Delete

```
DELETE FROM employee AS e
  WHERE e.card_number = '18108a24-4eac-498d-a2d3-4dc4cccf405a';
```

Feedback

Insert

```
INSERT INTO feedback
   VALUES(1, replace('Ok among class too. Fund organization throughout too
      when. Media green certain line.\nWest value campaign personal address
      recent already. Meeting worker ball east leave or.','\n',char(10)),
   '2017-08-31 14:12:30','books','5bdb012a-f823-4c8b-83e8-60c36c61743b');
```

Person must exist.

Delete

```
DELETE FROM feedback AS f
  WHERE f.feedback_id = 1;
```

Review

Insert

```
INSERT INTO review
   VALUES(1,0,replace('Outside hard discuss subject wind dark simply. Market
      big specific enter upon left sea.\nWatch that everyone mouth wrong. Play
      community agent particularly e','\n',char(10)),replace('Marriage minute
         again rather nice design unit. Area would scientist focus.\nFrom best
         experience our paper quite value. Sea yourself cause environmental
         account he.','\n',char(10)),57,'284388ff-f6d6-417e-b8de-be17cd8c909c');
```

Person must exist.

Delete

```
DELETE FROM review AS r
  WHERE r.review_id = 1;
```