**Database Project Final Report**

**CSE 3241 – Introduction to**

**Database Systems**

**Leon Madrid**

**Tuesday/Thursday, 9:35-10:55 am**

Tyler Cingel, Lily Driscoll, Isaac Mattern, Jacob Woodhouse

Section 1: Database Description

**Figure 1: Entity-Relation Diagram**

Diagram

Description automatically generated

**Figure 2: Relational Schema**

Diagram

Description automatically generatedFor each table, give a brief description of the level of normalization achieved for the table, if it is not in BCNF, explain why.

**Normalization achieved for each table:**

Media Item – This is in 2NF because values like Console are functionally dependent on GameFlag

Media Genres –

Artist Genres – This is in BCNF because, while there may be a trend, there is no dependence between Name and Genres

Branch – Because this table only has two attributes which make its key it is in BCNF

Order – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Patron – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Chapter – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Author – This table only has two attributes, one of which is the key, and the other of which is dependent on the key, meaning this table is in BCNF

Actor – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Game Studio – Because this table only has one attribute (Name) it is in BCNF

Artist – Because this table only has one attribute (Name) it is in BCNF

Track – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Author Writes – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Actor Stars – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Studio Creates – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Artist Authors – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Album Contains – This is in BCNF because non-key attributes are dependent on the keys, and a non-loss decomposition exists for each relation

Description of at least three indexes added and rational for each

**Indexes included:**

1. CREATE INDEX track\_artist

ON Track (Artist\_Name);

There will likely be many instances where we want to retrieve tracks only by a specific artist.

1. CREATE INDEX patron\_checkouts

ON Checkouts (Email\_Address);

It would be useful for the checkouts of specific patrons to be indexed so that when individual (or all) checkouts of a specific patron are needed, they are quickly accessible.

1. CREATE INDEX best\_available\_media

ON Media\_Item (Rating, Status, City);

Many patrons will only care to see the media items which are available and in a specific city. Additionally, many patrons would like to only see media items above a certain rating.

**Views included:**

1. Total number of chapters in each book to help users gauge the size of the book
   1. CREATE VIEW BookChaptersTotal As

SELECT Media\_Item.Name, Author\_Writes.Name as Author, COUNT (DISTINCT Chapter.Name) as Total\_Chapters

FROM Media\_Item, Author\_Writes, Chapter

WHERE Author\_Writes.ID=Chapter.ID AND Audiobook\_Flag=1 AND Chapter.ID=Media\_Item.ID

GROUP By Media\_Item.Name

* 1. π media\_item.name, author\_writes.name → author, COUNT (\delta name) → total\_chapters

γ name, COUNT (chapter.name)

σ author\_writes.id = chapter.id AND audiobook\_flag = 1 AND chapter.id = media\_item.id (media\_item × author\_writes × chapter)

* 1. Table

     Description automatically generated

1. Total number of checkouts by patron to help users see the most active patrons
   1. CREATE VIEW CheckoutsMade AS

SELECT First\_Name, Last\_Name, COUNT(Checkout\_ID) as Total\_Checkedout

FROM Patron, Checkouts

Where Patron.Email\_Address=Checkouts.Email\_Address

GROUP BY Last\_Name;

* 1. δ π first\_name, last\_name, COUNT (checkout\_id) → total\_checkedout

ρ last\_name, COUNT (checkout\_id)

σ patron.email\_address = checkouts.email\_address (patron × checkouts)

* 1. Table

     Description automatically generated

Description of three sample transactions useful for our database. Should include sample SQL code for each as well as English description of what “unit of work” the transaction represents.

**Transaction samples:**

1. [Transaction 1]
2. [Transaction 2]
3. [Transaction 3]

Section 2: User Manual

**Table descriptions:**

Media Item – A single piece of any media (audiobook, album, movie, game)

ID – Numeric ID of the media item, CHAR(10), NOT NULL, Primary Key

Copy# - Which copy of media item out of all copies, INT, NOT NULL

Name – Name of media item, VARCHAR (50)

Year – Year of release of item, INT

Length – Length of media item (minutes, pages), INT

%Flag – Boolean representation of type of media, INT

Publisher – Publishing company of media, VARCHAR(15)

Director – If movie, who directed it, VARCHAR(20)

Console – If game, console it is for, VARCHAR(20)

Rating – Rating of media item, VARCHAR(5)

Status – Whether it is available to checkout or not, INT

Order# - If currently being shipped, number for order, INT, Foreign Key

City – To which city the branch owning the item is in, VARCHAR(20), NOT NULL, Foreign Key

State – To which state the branch owning the item is in, VARCHAR(20), NOT NULL, Foreign Key

Media Genres – List of genres for any piece of media

ID – Numeric ID of the media item, CHAR(10), NOT NULL, Foreign Key

Copy# - Which copy of media item out of all copies, INT, NOT NULL, Foreign Key

Genres – Genre for this specific media item, VARCHAR(20), NOT NULL, Primary Key

Artist Genres – List of genres an artist makes music for

Name – Name of artist, VARCHAR(50), NOT NULL, Foreign Key

Genres – Specific Genre, VARCHAR(20), NOT NULL

Branch – A branch of this library represented by the city and state it’s in

City – City where library is located, VARCHAR(20), NOT NULL, Primary Key

State – State where library is located, VARCHAR(20), NOT NULL, Primary Key

Orders – An order that is in-transit to a given library

Order# - Number for the order, INT, NOT NULL, Primary Key

Price – Total cost of the order, VARCHAR(10)

Quantity – Number of items ordered, INT

Arrival Date – Date when items will arrive, DATE

City – City where destination library is located, VARCHAR(20), NOT NULL, Foreign Key

State – State where destination library is located, VARCHAR(20), NOT NULL Foreign Key

Patron – A customer of the library that can check out media

Email Address – Email address associated with individual, VARCHAR(50), NOT NULL, Primary Key

First Name – First name of individual, VARCHAR(20), NOT NULL

Last Name – Last name of individual, VARCHAR(20), NOT NULL

Address – Home address of individual, VARCHAR(50)

City – City where library branch is located, VARCHAR(20), NOT NULL, Foreign Key

State – State where library branch is located, VARCHAR(20), NOT NULL, Foreign Key

Chapter – A single chapter of a book

Name – Name of chapter, VARCHAR(50), NOT NULL, Primary Key

Number – Number of the chapter, INT, NOT NULL, Primary Key

ID – Numeric ID of the media item, CHAR(10), NOT NULL, Foreign Key

Copy# - Which copy of media item out of all copies, INT, NOT NULL, Foreign Key

Author – An author for a book

Name – Name of author, VARCHAR(50), NOT NULL, Primary Key

Age – Age of author, INT

Actor – An actor for a movie

Name – Name of actor, VARCHAR(50), NOT NULL, Primary Key

Sex – Gender of actor, CHAR(1)

Age – Age of actor, INT

Game Studio – A studio that specializes in game development

Name – Name of game studio, VARCHAR(50), NOT NULL, Primary Key

Track – A single song

Name – Name of song, VARCHAR(50), NOT NULL, Primary Key

Genre – Genre of the song, VARCHAR(20)

Artist Name – Name of artist who made song, VARCHAR(50), NOT NULL, Foreign Key

Author Writes – Link between a book and the author that wrote it

Name – Name of author, VARCHAR(50), NOT NULL

ID – Numeric ID of the book, CHAR(10), NOT NULL, Foreign Key

Copy# - Which copy of media item out of all copies, INT, NOT NULL, Foreign Key

Actor Stars – Link between an actor and the movie they’re starring in

Name – Name of actor, VARCHAR(50), NOT NULL

ID – Numeric ID of the movie, CHAR(10), NOT NULL, Foreign Key

Copy# - Which copy of media item out of all copies, INT, NOT NULL, Foreign Key

Studio Creates – Link between a studio and the game they’re creating

Name – Name of studio, VARCHAR(50), NOT NULL

ID – Numeric ID of the game, CHAR(10), NOT NULL, Foreign Key

Copy# - Which copy of media item out of all copies, INT, NOT NULL, Foreign Key

Artist Authors – Link between an artist and the song they’re creating

Name – Name of artist, VARCHAR(50) , NOT NULL

ID – Numeric ID of the song, CHAR(10), NOT NULL, Foreign Key

Copy# - Which copy of media item out of all copies, INT, NOT NULL, Foreign Key

Album Contains – List of tracks within an album

ID – Numeric ID of the song, CHAR(10), NOT NULL, Foreign Key

Copy# - Which copy of media item out of all copies, INT, NOT NULL, Foreign Key

Name – Name of track in album, VARCHAR(50), NOT NULL

Artist Name – Name of artist who made album, VARCHAR(50), NOT NULL, Foreign Key

**Sample SQL Queries:** π σ

Find the titles of all tracks by ARTIST released before YEAR

* SELECT DISTINCT Name FROM Album\_Contains, Media\_Item WHERE Media\_Item.Year<2000 AND Album\_Contains.ArtistName=’Billy Joel’;
* [Relational algebra query]

Give all the movies and their date of their checkout from a single patron

* SELECT Name, Checkout\_Date FROM Media\_Item WHERE email\_address=’librarypatron@gmail.com’ AND Movie\_Flag=1;
* [Relational algebra query]

List all the albums and their unique identifiers with less than 2 copies held by the library

* SELECT Name, ID FROM Media\_Item WHERE Album\_Flag=1 AND Copy\_Number<2;
* [Relational algebra query]

Give all the patrons who checkout out a movie by ACTOR and the movies they checked out

* SELECT First\_Name, Last\_Name, Media\_Item.Name FROM Patron, Media\_Item, Actor\_Stars WHERE Patron.Email\_Address = Media\_Item.Email\_Address AND Actor\_Stars.Name = ACTOR;
* [Relational algebra query]

Find the total number of albums checked out by a single patron

* SELECT COUNT(Media\_Item) FROM Media\_Item WHERE Media\_Item.Email\_Address=’librarypatron@gmail.com’ AND Album\_Flag=1;
* [Relational algebra query]

Find the patron who has checked out the most videos and the total number of videos they have checked out

* SELECT Patron, COUNT(Media\_Item) FROM Patron, Media\_Item WHERE Media\_Item.Email\_Address = ‘librarypatron@gmail.com’ AND Movie\_Flag=1;
* [Relational algebra query]

List all the games from a game studio

* SELECT Media\_Item.Name FROM Media\_Item, Studio\_Creates WHERE Game\_Flag=1 AND Studio\_Creates.Name=’Electronic Arts’;
* σ[Game\_Flag=1]Media\_Item (join)[ID=ID] σ[Name=’EA’]StudioCreates

Find how many movies an actor/actress has played in that are RATING and are in stock

* SELECT COUNT(\*) FROM Media\_Item, Actor\_Stars WHERE Rating=5 AND Movie\_Flag=1 AND Actor\_Stars.Name = ‘Brad Pitt’;
* F\_COUNT(σ[Rating=RATING](σ[Movie\_Flag=1]Media\_Item (join)[ID=ID] σ[Name=ACTOR]Actor\_Stars))

**INSERT Statement descriptions/examples**

Albums

Syntax – INSERT INTO Media\_Item VALUES ([ID], [copy#], [name], [year], [length], 0, 1, [publisher], 0, NULL, 0, NULL, [rating], [status], [order#], [city], [state]);

Dependencies – Branch

Example – INSERT INTO Media\_Item VALUES (‘1265472654’, 1, ‘Greatest Hits’, 1985, 113, 0, 1, ‘Columbia’, 0, NULL, 0, NULL, NULL, 1, NULL, ‘Cleveland’, ‘Ohio’);

Movies/Videos

Syntax – INSERT INTO Media\_Item VALUES ([ID], [copy#], [name], [year], [length], 0, 0, NULL, 1, [director], 0, NULL, [rating], [status], [order#], [city], [state]);

Dependencies – Branch

Example – INSERT INTO Media\_Item VALUES (‘1358594289’, 3, ‘Star Wars IV: A New Hope’, 1977, 121, 0, 0, NULL, 1, ‘George Lucas’, 0, NULL, NULL, 1, NULL, ‘Phoenix’, ‘Arizona’);

Audiobooks

Syntax – INSERT INTO Media\_Item VALUES ([ID], [copy#], [name], [year], [length], 1, 0, [publisher], 0, NULL, 0, NULL, [rating], [status], [order#], [city], [state]);

Dependencies – Branch

Example – INSERT INTO Media\_Item VALUES (‘1234567890’, 1, ‘Harry Potter and the Sorcerer’s Stone’, 1997, 269, 0, 0, ‘Bloomsbury’, 0, NULL, 0, NULL, NULL, 1, NULL, ‘Columbus’, ‘Ohio’);

Artists

Syntax – INSERT INTO Artist VALUES ([name], [gender(F/M)], [age]);

Dependencies – N/A

Example – INSERT INTO Artist VALUES (‘Bruno Mars’, ‘M’, ‘36’);

Patrons

Syntax – INSERT INTO Patron VALUES ([email], [first name], [last name], [address], [city], [state]);

Dependencies – N/A

Example – INSERT INTO Patron VALUES (‘email@yahoo.com’, ‘John’, ‘Doe’, ‘404 Missing Lane, Las Vegas, NV 44720’, ‘Las Vegas’, ‘Nevada’);

**Miscellaneous INSERT Examples**

Branch – INSERT INTO Branch VALUES ('Cleveland', 'Ohio');

Orders – INSERT INTO Orders VALUES (2011, '$15', 1, 2022-01-04, 'Columbus', 'Ohio');

Checkouts – INSERT INTO Checkouts VALUES ('1111111111', '2021-11-08', '2021-10-08', 'librarypatron@gmail.com', '6421358790', 1);

Author – INSERT INTO Author VALUES ('Joanne K. Rowling', 56);

Actor – INSERT INTO Actor VALUES ('Mark Hamill', 'M', 70);

Game Studio – INSERT INTO Game\_Studio VALUES ('NineHertz');

Artist Genre – INSERT INTO Artist\_Genres VALUES ('Coldplay', 'Rock');

Track – INSERT INTO Track VALUES ('Politik', 'Rock', 'Coldplay');

Media Genres – INSERT INTO Media\_Genres VALUES ('1234567890',1, 'Fantasy');

Chapter – INSERT INTO Chapter VALUES ('The Boy Who Lived', 1,'1234567890',1);

Author Writes – INSERT INTO Author\_Writes VALUES ('Patrick James Rothfuss', '4563153215',1);

Actor Stars – INSERT INTO Actor\_Stars VALUES ('Mark Hamill', '7678906543',1);

Studio Creates – INSERT INTO Studio\_Creates VALUES ('Nintendo', '6421358790',1);

Album Contains – INSERT INTO Album\_Contains VALUES ('Politik', '2469762156', 1, 'Coldplay');

**DELETE Statement descriptions/examples**

Albums

Syntax – DELETE FROM Orders WHERE Orders\_Number=?;

Dependencies – Track, Album\_Contains

Example – DELETE FROM Orders WHERE Orders\_Number=10;

Movies/Videos

Syntax – DELETE FROM Media\_Item WHERE MovieFlag=1 AND ID=?;

Dependencies – Actor\_Stars

Example – DELETE FROM Media\_Item WHERE MovieFlag=1 AND ID=’7678906543’;

Audiobooks

Syntax – DELETE FROM Media\_Item WHERE AudiobookFlag=1 AND ID=?;

Dependencies – Author\_Writes

Example – DELETE FROM Media\_Item WHERE AudiobookFlag=1 AND ID=’1234567890’;

Artists

Syntax – DELETE FROM Artists WHERE Name=?;

Dependencies – Track, Artist\_Authors, Artist\_Genres

Example – DELETE FROM Artists WHERE Name=’Bruno Mars’;

Patrons

Syntax – DELETE FROM Patron WHERE Email\_Address=?;

Dependencies – N/A

Example – DELETE FROM Patron WHERE Email\_Address=’libraryuser@yahoo.com’;

**Miscellaneous DELETE Examples**

Branch – DELETE FROM Branch WHERE City=’Houston’ AND State=’Texas’;

Orders – DELETE FROM Orders WHERE Arrival\_Date<DATE.current;

Checkouts – DELETE FROM Checkouts WHERE Checkout\_Date+[7 days]<DATE.current;

Author – DELETE FROM Author WHERE Name=’Dr. Seuss’;

Actor – DELETE FROM Actor WHERE Name=’Carrie Fisher’;

Game Studio – DELETE FROM Game\_Studio WHERE Name=’Electronic Arts’;

Artist Genre – DELETE FROM Artist\_Genre WHERE Genres=’Electronic Rock’;

Track – DELETE FROM Track WHERE Name=’Greatest Hits’;

Media Genres – DELETE FROM Media\_Genres WHERE ID=’1234567890’;

Chapter – DELETE FROM Chapter WHERE Name=’Chapter 1’ AND Number=1;

Author Writes – DELETE FROM Author\_Writes WHERE ID=’1234567890’;

Actor Stars – DELETE FROM Actor\_Stars WHERE Name=’Carrie Fisher’;

Studio Creates – DELETE FROM Studio\_Creates WHERE ID=’1234567809’ AND Copy#=’4’;

Album Contains – DELETE FROM Album\_Contains WHERE Name=’Holiday’ AND ArtistName=’Green Day’;