Final Project Report

CSE 3421

April 20, 2021

Cameron Jackson

Jake Kennedy

Josh Riebe

Matthew Rindler

Diagram

Description automatically generatedDatabase Description **ER Diagram**

**Table

Description automatically generatedRelational Schema**

**Functional Dependencies**

**Physical DVD:** ID# 🡪 {Stock, Location, Year, Genre, Title, Length, Rating, Order#, DirectorID#}

**Digital Movie:** ID# 🡪 {Stock, License, Year, Genre, Title, Length, Rating, Order#, DirectorID#}

**Physical Book:** ID# 🡪 {Stock, Location, ISBN, Order#, AuthorID#}

**Audiobook:** ID# 🡪 {Stock, License, ISBN, Order#, AuthorID#}

**Physical Disc:** ID# 🡪 {Stock, Title, Year, Length, Rating, Location, Order#, AristID#}

**Digital Album** ID# 🡪 {Stock, Title, Year, Length, Rating, License, Order#, AristID#}

**P. Book Info:** ISBN 🡪 {Year, Genre, Title, Length, #ofChapters}

**A. Book Info:** ISBN 🡪 {Year, Genre, Title, Time, #ofChapters}

**Artist:** ID# 🡪 ArtistName

**Track:** ID# 🡪 {Title, Number, Genre, AlbumID, DiscID}

**Employee:** SSN 🡪 DOB

**Library Card:** ID# 🡪 {Login, Password, Address, PaymentInfo}

**Orders:** Order# 🡪 {Order Date, ESSN}

**Ordered Media:** {SKU, OrderID} 🡪 {MediaMedium, #ofCopies, PricePerCopy, MediaType}

**Preference:** ID# 🡪 {FavoriteGenre, PatronID#}

**Feedback:** ID# 🡪 {Comment, PatronID#}

**Turns Into:** {Order#, SKU, MediaID} 🡪 ShippingDate

**Borrows:** {PatronID#, MediaID} 🡪 {BorrowDate, DueDate}

**Manages:** No Functional Dependency

**Staring:** No Functional Dependency

**Person:** ID# 🡪 {FirstName, LastName, PatronFlag, CardID, ActorFlag, AuthorFlag, DirectorFlag, EmployeeFlag, ESSN}

**Normalization of Tables**

**Physical DVD:** All attributes are atomic and functionally dependent on only the ID # attribute. Therefore, the table is in BCNF.

**Digital Movie:** All attributes are atomic and functionally dependent on only the ID # attribute. Therefore, the table is in BCNF.

**Physical Book:** All attributes are atomic and functionally dependent on only the ID # attribute. Therefore, the table is in BCNF.

**Audiobook:** All attributes are atomic and functionally dependent on only the ID # attribute. Therefore, the table is in BCNF.

**Physical Disc:** All attributes are atomic and functionally dependent on only the ID # attribute. Therefore, the table is in BCNF.

**Digital Album:** All attributes are atomic and functionally dependent on only the ID # attribute. Therefore, the table is in BCNF.

**P. Book Info:** All attributes are atomic and functionally dependent on only the ISBN attribute. Therefore, the table is in BCNF.

**A. Book Info:** All attributes are atomic and functionally dependent on only the ISBN attribute. Therefore, the table is in BCNF.

**Artist:** This table is in BCNF as the ID# determines the artist name attribute and those are the only two attributes in the table.

**Track:** All attributes of this table are atomic and fully dependent on the ID# of the track. Therefore, the table is in BCNF.

**Employee:** This table is BCNF since both attributes are atomic and the employee’s DOB is dependent on the Employee’s SSN.

**Library Card:** All the attributes of this table are atomic. All attributes are functionally dependent on the library card ID, however, since usernames are unique, they are also functionally dependent on the login attribute as well. However, since only one patron can have a username, creating a new table seems unnecessary since it would not save any repeated data from being stored and would create an extra table.

**Orders:** The atomic attributes Order Date and ESSN are fully dependent on only the Order number. Meaning this table is in BCNF.

**Ordered Media:** All attributes are atomic. The SKU and order number are needed to determine the ordered media within an order. The whole key is needed to identify the rest of the information. There is no other functional dependency, so the table is in BCNF.

**Preference:** The ID# is used to identify all other atomic attributes of the preference table. Thus, the table is in BCNF.

**Feedback:** All attributes of this table are atomic. The primary key, ID#, is the only functional dependency of the table, so it is in BCNF.

**Turns Into:** All attributes of this table are atomic. The order #, SKU, and media ID are all needed to correctly identify the shipment and determine its shipping date. This table is in BCNF.

**Borrows:** All attributes in this table are atomic. Since a patron can borrow multiple items, the media ID of the borrowed item is also needed to identify the desired row. There are no other functional dependencies, so the table is in BCNF.

**Person:** All attributes of this table are atomic. The ID # is the only unique attribute of every person in the system, The rest of the information is functionally dependent on it. This table is in BCNF.

**Indices**

**PersonID**: An index on the table Person that indexes the ID number of each person. This table is useful because Patrons, Authors, Directors, Actors, and Employees are all stored in the Person table. Since we have multiple foreign keys in other tables that reference the ID attribute I the person table, it will make the code more efficient to be able to have a constant look up time for the Person ID instead of searching through the entire table.

**TrackGenre**: An index on the Track table that indexes the genres of the tracks. Over time, as there become more and more albums that are added to the database, as these albums are added, they will bring anywhere from 10 – 20 songs with them that need to be added to the database. This means that the Track table could potentially grow astronomically. Because of this, it would be useful to have an index over the genre attribute of the track. This would make it easier to determine songs that may have similar styles or to find songs that could be grouped together and recommended to patrons.

**UserPreference:** Over time, patrons that frequent the library could add more and more personal preferences to the database to help the library recommend new media that they may enjoy. Because of the large number of preferences that may be submitted by patrons of the library, it would be beneficial to have an index based on the patronID of the preference for the table. This would allow the database to quickly find the preferences of a specific user to return items that they may like.

**Views**

**AlbumCount:** This view displays the number of albums that the library has by each band in the Database. This would be useful because it can help to see what bands are popular in the library or what bands are as represented in the library music collection.

**Relation Algebra**: R1 🡨 Artist.Name(Artist  ⋈Artist.ID = PhysicalDisc.ArtistID PhysicalDisc)

R2 🡨 Artist.Name(Artist  ⋈Artist.ID = DigitalAlbum.ArtistID DigitalAlbum)

R3 🡨 R1 U R2

Name ℱCOUNT Artist.Name(R3)

**SQL Code**: Select Artist.Name, COUNT(Artist.Name) AS AlbumsInHouse

From Artist, PhysicalDisc

Where Artist.ID = PhysicalDisc.ArtistID

GROUP BY Artist.ID

UNION

Select Artist.Name, COUNT(Artist.Name) AS AlbumsInHouse

From Artist, DigitalAlbum

Where Artist.ID = DigitalAlbum.ArtistID

GROUP BY Artist.ID

ORDER BY Artist.Name

**Sample Output:**

|  |  |
| --- | --- |
| AC/DC | 2 |
| Accept | 2 |
| Aerosmith | 1 |
| Alanis Morrisette | 1 |
| Antonio Carlos Jobim | 1 |
| Eric Clapton | 1 |
| Frank Zappa & Captain Beefheart | 1 |
| Iron Maiden | 1 |
| Judas Priest | 1 |
| Led Zeppelin | 1 |

**CheckoutCount:** This view displays the current number of items checked out by patrons order by the most items checked out to the least items checked out. This view is useful because it lets the workers know which patrons have too many items checked out and which ones are allowed to check out more items.

**Relational Algebra:**

R1 🡨 Borrows ⋈Borrows.PatronID = LibraryCard.ID LibraryCard

R2 🡨 **σPatronFlag** = TRUE (Person)

R3 🡨 R1⋈R1.PatronID = R2.CardIDR2

**SQL Code:**

SELECT Person.FName, Person.LName, COUNT(Borrows.PatronID) AS ItemsCheckedOut

FROM Borrows, LibraryCard, Person

WHERE Borrows.PatronID = LibraryCard.ID

AND Person.PatronFlag = TRUE

AND Person.CardID = LibraryCard.ID

Group by LibraryCard.ID

ORDER BY ItemsCheckedOut DESC

**Sample Output:**

|  |  |  |
| --- | --- | --- |
| FName | LName | ItemsCheckedOut |
| Riley | Durham | 4 |
| Russell | Bailley | 4 |
| Eliza | Potts | 2 |
| John | Smith | 2 |
| Cameron | Jackson | 1 |
| Josh | Riebe | 1 |
| Matthew | Rindler | 1 |

**Sample Transactions**

Hiring New Worker

One transaction that could be done would be hiring a new worker to the library. To do this, the employee’s SSN and birthday would need to be added to the Employee table. Assuming that the worker was already a person in the database, the transaction would then have to update the Employee Flag and ESSN attribute in the person table.

SQL Code:

BEGIN TRANSACTION;

INSERT OR ROLLBACK INTO Employee

VALUES ('654893146', '03-01-2000');

UPDATE OR ROLLBACK Person

SET EmployeeFlag = TRUE, ESSN = '654893146'

WHERE Person.ID = 10;

COMMIT;

Decreasing Item Stock

Another transaction that could be done is changing the stock number for a given media if a worker counts it and notices it is different than what is in the data base. The transaction would involve a read and a write on the desired table. The write would retrieve the MediaID for the item and the write would update the stock to the correct value.

BEGIN TRANSACTION;

UPDATE OR ROLLBACK AudioBook

SET Stock = 5

WHERE AudioBook.ID = 72;

COMMIT;

Updating User Password

If a user forgets their password or wishes to set it to something else, their password could be updated in a transaction. The transaction would be a single new write into the password attribute to update the password to the new password.

BEGIN TRANSACTION;

UPDATE OR ROLLBACK LibraryCard

SET Password = 'ColoradoBlues'

Where ID = 10;

COMMIT;

**User Manual**

**Entity Descriptions**

**ABookInfo –** Represents the info that corresponds to an ISBN.

ISBN – 13 characters, corresponding to book’s ISBN.

Year – INT of length 4, year the book was published.

Genre – string up to length 50, represents genre of book.

Title – Title of the book, up to 50 characters.

Time – Int, Time in seconds for the Audiobook to play.

Chapters – Int, number of chapters in the audiobook.

**Artist –** Represents an Artist that may or may not have songs in Library.

ID – INT, ID number for artist.

Name – Name of Artist, up to 50 characters.

**AudioBook –** Represents an audiobook that can be borrowed from Library.

ID – INT, ID number for Audiobook.

Stock – INT, Number of copies currently available.

License – String, up to 30 chars, represents online license for audiobook.

ISBN – 13 char string, Represents ISBN of book.

OrderNumber – INT, Order Number that the book was a part of.

AuthorID – INT, ID number of the Author of book.

**Borrows –** Represents the items that are borrowed by patrons.

PatronID – INT, ID Number of the patron that borrowed an item.

MediaID – INT, ID Number of the borrowed Item.

BorrowDate – DATE, Date the borrowed item was taken.

DueDate – DATE, Date the borrowed Item must be returned.

**DigitalAlbum –** Represents a digital album that is available for checkout.

ID – INT, ID number for album.

Stock – INT, number of copies currently available.

Title – String up to 50, title of the album.

Year - INT, year the album was released.

Length – INT, length of album in seconds.

Rating - String up to 15, content rating of album.

License – String, up to 30 chars, represents online license for Album.

OrderNum – INT, Order number that the album was a part of.

ArtistID – INT, ID number of the artist of the album.

**DigitalMovie –** Represents a digital movie that can be borrowed.

ID – INT, ID Number for movie.

Stock – INT, number of copies currently available.

License – String up to 30 chars, represents digital License of movie.

Year – INT, year movie was released.

Genre – String up to 30, genre of movie.

Title – String up to 50, title of movie.

Length – INT, length in seconds of movie.

Rating – String up to 4 chars, content rating of movie.

OrderNumber -INT, Order number that movie was a part of.

DirectorId – INT, ID number of the director of movie.

**Employee –** Library Employee’s personal info

SSN – String of 9 chars, employees Social Security Number.

DOB - DATE, Employees Date of Birth.

**Feedback –** Feedback comment left by a patron

ID – INT, ID number for feedback.

Comment – String up to 100 chars, feedback comment left by patron.

PatronID – INT, ID Number of patron that left comment.

**LibraryCard** – Library card and owner information

ID – INT, Library Card ID Number.

Login – String up to 20 chars, username of library card owner.

Password – String, up to 20 chars, password of library card owner.

Address – String up to 30 chars, email address of library card owner.

PaymentInfo – 16 char string, credit card number of card owner.

**Manages –** Shows what employees manage what library cards.

ESSN – String of 9 chars, employee social security number.

CardID – INT, library card number .

**OrderedMedia -**  Shows items that have been ordered to the library.

SKU – INT, SKU of ordered media.

OrderID – INT, order number media is a part of.

MediaMedium – String, Digital or Physical.

NumberOfCopies – INT, number of copies ordered.

PricePerCopy – DECIMAL, price per copy of ordered media.

MediaType – string, Movie, Album, Book.

**Orders –** Shows orders that have been placed that contain ordered Media.

Number – INT, Order ID number.

Date – DATE, date order was placed.

ESSN – String of 9 chars, ordering employees SSN.

**PBookInfo –** Information that corresponds to a physical Book’s ISBN.

ISBN – 13 characters, corresponding to book’s ISBN.

Year – INT of length 4, year the book was published.

Genre – string up to length 50, represents genre of book.

Title – Title of the book, up to 50 characters.

Length – Int, number of pages in the book.

Chapters – Int, number of chapters in the book.

**Person –** Represents information for people, can be patrons, actors, authors, directors, or employees.

ID – INT, ID number for person.

Fname – String up to 15 chars, first name of person.

Lname – String up to 15 chars, last name of person.

PatronFlag – BOOLEAN, true is person is a patron, false otherwise.

CardID – INT, LibraryCardID that corresponds to patron.

ActorFlag – BOOLEAN, true is person is an actor, false otherwise.

AuthorFlag – BOOLEAN, true is person is an author, false otherwise.

DirectorFlag – BOOLEAN, true is person is a director, false otherwise.

EmployeeFlag - BOOLEAN, true is person is an employee, false otherwise.

ESSN – String of 9 chars, Employees Social Security Number.

**PhysicalBook –** Represents a physical book that can be checked out.

ID – INT, ID number for Audiobook.

Stock – INT, Number of copies currently available.

Location – String, up to 30 chars, location of book in library.

ISBN – 13 char string, Represents ISBN of book.

OrderNumber – INT, Order Number that the book was a part of.

AuthorID – INT, ID number of the Author of book.

**PhysicalDisc –** Represents a physical album that can be checked out.

ID – INT, ID number for album.

Stock – INT, number of copies currently available.

Title – String up to 50, title of the album.

Year - INT, year the album was released.

Length – INT, length of album in seconds.

Rating - String up to 15, content rating of album.

Location – String, up to 30 chars, location of album in Library.

OrderNum – INT, Order number that the album was a part of.

ArtistID – INT, ID number of the artist of the album.

**PhysicalDVD –** Physical movie that is available for checkout

ID – INT, ID Number for movie.

Stock – INT, number of copies currently available.

Location – String up to 30 chars, location of movie in Library.

Year – INT, year movie was released.

Genre – String up to 30, genre of movie.

Title – String up to 50, title of movie.

Length – INT, length in seconds of movie.

Rating – String up to 4 chars, content rating of movie.

OrderNumber -INT, Order number that movie was a part of.

DirectorId – INT, ID number of the director of movie.

**Preference –** Patron’s genre preference.

ID – INT, ID number of preference.

FavoriteGenre – string up to 20 chars, patron’s favorite genre.

PatronID – INT, ID number of patron that created preference.

**Starring –** Shows what actors star in what movies.

MovieID – INT, movie ID number that actor stars in.

ActorID – INT, Actor ID number that corresponds to an actor.

**Track –** Songs that are on albums

ID – INT, ID number of track.

Title – String up to 50 chars, title of track.

Number – INT, track number of the song in the album.

Genre – String up to 20 chars, genre of the track.

AlbumID – INT, Album ID number of Digital Album if this track is on it.

DiscID – INT, DISC ID number of physical disc if this track is on it.

**TurnsInto –** Shows orders turning into available media.

OrderNum – INT, Order Number of order that contains item.

SKU – INT, SKU of Item in ORDER.

MediaID – INT, ID number of Media in order.

ShippingDate – DATE, Date that order was received.

**Sample SQL Queries**

1. Find the titles of all tracks by ARTIST (AC/DC) released before YEAR (2000)
2. R1 ← Track ⋈Track.DiscID = DigitalAlbum.ID DigitalAlbum

R2 ← Track ⋈Track.DiskID = PhysicalDisc.ID PhysicalDisc

R3 ← R1 U R2

R4 ← R3 ⋈ID = ArtistID Artist

𝜋 Title (σArtist.Name = ‘AC/DC’ AND Year < 2000 R4)

1. SELECT Track.Title

FROM Track,Artist,PhysicalDisc

WHERE Track.DiscID=PhysicalDisc.ID

AND PhysicalDisc.ArtistID=Artist.ID AND Artist.Name='AC/DC'

AND PhysicalDisc.Year<2000

UNION

SELECT Track.Title

FROM Track,Artist,DigitalAlbum

WHERE Track.AlbumID=DigitalAlbum.ID

AND DigitalAlbum.ArtistID=Artist.ID AND Artist.Name='AC/DC'

AND DigitalAlbum.Year<2000;

1. Give all the movies and their date of their checkout from a single patron (patron designated by LibraryCard ID Number=1)
2. D ← Person ⋈ID=PatronIDBorrows

S← D⋈MediaID=IDPhysicalDVD

R← D⋈MediaID=IDDigitalMovie

B ← S U R

πName,BorrowDate(σID = 1(B))

1. SELECT Person.Fname, Person.Lname, PhysicalDVD.Title, Borrows.BorrowDate

FROM PhysicalDVD,Borrows,LibraryCard,Person

WHERE Person.CardID=LibraryCard.ID AND PhysicalDVD.ID = Borrows.MediaID

AND Borrows.PatronID=LibraryCard.ID

AND LibraryCard.ID=1

UNION

SELECT Person.Fname, Person.Lname, DigitalMovie.Title, Borrows.BorrowDate

FROM DigitalMovie,Borrows,LibraryCard,Person

WHERE Person.CardID=LibraryCard.ID AND DigitalMovie.ID = Borrows.MediaID

AND Borrows.PatronID=LibraryCard.ID

AND LibraryCard.ID=1;

1. List all the albums and their unique identifiers with less than 2 copies held by the library.
2. R1 ← 𝜋Title, ID (σStock < 2 PhysicalDisc)

R2 ← 𝜋Title, ID (σStock < 2 DigitalAlbum)

Result ← R1 U R2

1. SELECT PhysicalDisc.Title, PhysicalDisc.ID

FROM PhysicalDisc

WHERE PhysicalDisc.Stock<2

UNION

SELECT DigitalAlbum.Title, DigitalAlbum.ID

FROM DigitalAlbum

WHERE DigitalAlbum.Stock<2;

1. Give all the patron’s IDs who checked out a movie by ACTOR (ID=12 (Steven Spielberg)) and the movies they checked out.
2. D ← Person ⋈ID=PatronIDBorrows

M1← D⋈MediaID=IDPhysicalDVD

M2← D⋈MediaID=IDDigitalMovie

R ← M1 U M2

Q← R⋈Person.ID=ActorIDStarring

πID,Patron Name,Movie ID, Movie Name (σName=ACTOR(Q))

1. SELECT LibraryCard.ID,PhysicalDVD.Title

FROM LibraryCard,PhysicalDVD,Starring,Borrows,Person

WHERE LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID=PhysicalDVD.ID AND PhysicalDVD.ID=Starring.MovieID AND Starring.ActorID=Person.ID AND Person.ID=12

UNION

SELECT LibraryCard.ID,DigitalMovie.Title

FROM LibraryCard,DigitalMovie,Starring,Borrows,Person

WHERE LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID=DigitalMovie.ID AND DigitalMovie.ID=Starring.MovieID AND Starring.ActorID=Person.ID AND Person.ID=12;

2. Find the total number of albums checked out by a single patron (Patron designated by library card ID=15)
3. D ← Person ⋈ID = PatronID Borrows

A1 ← D ⋈MediaID = IDDigitalAlbum

A2 ← D ⋈MediaID = IDPhysicalDisc

R ← A1 U A2

ℱCOUNT MediaID (σPerson.ID = 15(R))

1. SELECT SUM(COUNT)

FROM(

SELECT COUNT(Borrows.PatronID) as COUNT

FROM Borrows, PhysicalDisc,LibraryCard

WHERE LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID = PhysicalDisc.ID AND LibraryCard.ID = 15

UNION ALL

SELECT COUNT(Borrows.PatronID) as COUNT

FROM Borrows, DigitalAlbum,LibraryCard

WHERE LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID = DigitalAlbum.ID AND LibraryCard.ID = 15);

2. Find the patron who has checked out the most videos and the total number of videos they have checked out.
3. D ← Person ⋈ID= PatronID Borrows

S← D⋈MediaID=IDPhysicalDVD

R← S⋈MediaID=ID DigitalMovie

Q← PatronID ℱ COUNT MovieID (R)

ℱMAX Count(MovieID) (Q)

1. SELECT Data.ID,Data.FName,Data.LName,SUM(Count)

FROM(

SELECT LibraryCard.ID,Person.FName,Person.LName, COUNT(Borrows.MediaID) as Count

FROM LibraryCard, Borrows ,PhysicalDVD,Person

WHERE Person.CardID=LibraryCard.ID AND LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID=PhysicalDVD.ID

GROUP BY LibraryCard.ID

UNION ALL

SELECT LibraryCard.ID,Person.FName,Person.LName, COUNT(Borrows.MediaID) as Count

FROM LibraryCard, Borrows,DigitalMovie,Person

WHERE Person.CardID=LibraryCard.ID AND LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID=DigitalMovie.ID

GROUP BY LibraryCard.ID

) AS data

GROUP BY data.ID

ORDER BY DESC

LIMIT 1;

1. List all of the favorite genres of the PATRON with ID = 7
2. D ← LibraryCard ⋈ID = LibraryCard Preference

πFavorite\_Genres(σID=7(D))

1. SELECT Preference.FavoriteGenre

FROM LibraryCard,Preference

WHERE Preference.PatronID=LibraryCard.ID AND LibraryCard.ID = 7;

2. Find the oldest employee that manages library cards
3. D← Person⋈ESSN = SSNEmployee

R← D⋈SSN =ESSNManages

πFName,LName(ℱMIN DOB (D))

1. SELECT Person.ID, Person.FName, Person.LName

FROM Person, Employee, Manages

WHERE Person.EmployeeFlag = TRUE AND Person.ESSN=Employee.SSN AND Employee.SSN = Manages.ESSN

GROUP BY Person.ID

ORDER BY Employee.DOB ASC

LIMIT 1;

2. Provide a list of patron names, along with the total combined running time of all the movies they have checked out.
3. M1 ← 𝜋 ID, Length (DigitalMovie)

M2 ← 𝜋 ID, Length (PhysicalDVD)

M3 ← M1 U M2

R ← Person ⋈ID = PatronID Borrows

Result ← 𝜋Fname, Lname, SUM(Length) (R)

1. SELECT Data.FName,Data.LName,SUM(Length)

FROM(

SELECT LibraryCard.ID,Person.FName, Person.LName, SUM(PhysicalDVD.Length) as Length

FROM LibraryCard, PhysicalDVD, Borrows,Person

WHERE Person.CardID=LibraryCard.ID AND LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID=PhysicalDVD.ID

GROUP BY LibraryCard.ID

UNION ALL

SELECT LibraryCard.ID,Person.FName, Person.LName, SUM(DigitalMovie.Length) as Length

FROM LibraryCard, DigitalMovie, Borrows,Person

WHERE Person.CardID=LibraryCard.ID AND LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID=DigitalMovie.ID

GROUP BY LibraryCard.ID) AS Data

GROUP BY Data.ID;

2. Provide a list of patron names and addresses for patrons who have checked out more albums than the average patron.
3. P ← Person ⋈ ID = PatronID Borrows

L ← P ⋈ Person.ID = LibraryCard.ID LibraryCard

A1 ← 𝜋 ID (DigitalAlbum)

A2 ← 𝜋 ID (PhysicalDisc)

A ← A1 U A2

S ← σA.ID = L.MediaID (A)

B ← ℱ S > AVERAGE COUNT(S) (A)

R ← S U B

Result ← 𝜋 Fname, Lname, Address (R)

1. SELECT Data.ID,Data.Fname,Data.Lname,Data.Address,SUM(Data.AlbumCount) AS BorrowCount

FROM(

SELECT Person.ID,Person.Fname,Person.Lname, LibraryCard.Address, COUNT(Borrows.MediaID) AS AlbumCount

FROM Person,LibraryCard,Borrows,PhysicalDisc

WHERE Person.PatronFlag=TRUE AND Person.CardID=LibraryCard.ID AND LibraryCard.ID=Borrows.PatronID

AND Borrows.MediaID=PhysicalDisc.ID

GROUP BY LibraryCard.ID

UNION ALL

SELECT Person.ID,Person.Fname,Person.Lname, LibraryCard.Address, COUNT(Borrows.MediaID) AS AlbumCount

FROM Person,LibraryCard,Borrows,DigitalAlbum

WHERE Person.PatronFlag=TRUE AND Person.CardID=LibraryCard.ID AND LibraryCard.ID=Borrows.PatronID

AND Borrows.MediaID=DigitalALbum.ID

GROUP BY LibraryCard.ID

)AS Data

GROUP BY Data.ID;

2. Provide a list of the borrowed movies in the database and associated total copies lent to patrons, sorted from the movie that has been lent the most to the movies that has been lent the least.
3. D ← Person ⋈ID=PatronIDBorrows

M1← D⋈MediaID=IDPhysicalDVD

M2← D⋈MediaID=IDDigitalMovie

M ← M1 U M2

Result ← πTitle, COUNT(MediaID)(M)

1. SELECT Data.Title,SUM(Count) AS LoanCount

FROM(

SELECT PhysicalDVD.ID,PhysicalDVD.Title, COUNT(PhysicalDVD.ID) AS Count

FROM PhysicalDVD, Borrows

WHERE PhysicalDVD.ID=Borrows.MediaID

GROUP BY PhysicalDVD.ID

UNION

SELECT DigitalMovie.ID,DigitalMovie.Title, COUNT(DigitalMovie.ID) AS Count

FROM DigitalMovie, Borrows

WHERE DigitalMovie.ID=Borrows.MediaID

GROUP BY DigitalMovie.ID) AS Data

GROUP BY Data.Title

ORDER BY LoanCount DESC;

* 1. Provide a list of the borrowed albums in the database and associated totals for copies checked out to customers, sorted from the ones that have been checked out the highest amount to the ones checked out the lowest.
  2. A1 ← Borrows ⋈MediaID = ID PhysicalDisc

A2 ← A1 ⋈PatronID = ID LibraryCard

A3 ← σ Title, COUNT(PhysicalDisc.ID) (A2)

ρ(Title, Count) (A3)

B1 ← Borrows ⋈MediaID = ID DigitalAlbum

B2 ← B1 ⋈PatronID = ID LibraryCard

B3 ← σ Title, COUNT(DigitalAlbum.ID) (B2)

ρ(Title, Count) (B3)

C1 ← A3 U B3

C2 ← σ(Title, SUM(Count)) (C1)

ρ(Title, LoanCount) (C2)

𝜋(C2)

* 1. SELECT Data.Title,SUM(Count) AS LoanCount

FROM(

SELECT PhysicalDisc.Title, COUNT(PhysicalDisc.ID) AS Count

From PhysicalDisc, LibraryCard, Borrows

WHERE PhysicalDisc.ID=Borrows.MediaID AND Borrows.PatronID=LibraryCard.ID

GROUP BY PhysicalDisc.ID

UNION

SELECT DigitalAlbum.Title, COUNT(DigitalAlbum.ID) AS Count

From DigitalAlbum, LibraryCard, Borrows

WHERE DigitalAlbum.ID=Borrows.MediaID AND Borrows.PatronID=LibraryCard.ID

GROUP BY DigitalAlbum.ID) AS Data

GROUP BY Data.Title

ORDER BY LoanCount DESC;

* 1. Find the most popular actor in the database (i.e. the one who has had the most lent movies) and the number of movies they have had borrowed
  2. A1 ← Person ⋈ID = ActorID Starring

A2 ← A1 ⋈MovieID = Id PhysicalDVD

A3 ← A2 ⋈PhysicalDVD.ID = MediaID Borrows

A4 ← σ ActorFlag = TRUE A3

A5 ← σ Person.ID, Fname, Name, COUNT(MediaID)(A4)

ρ(Person.Id, Fname, Lnmae, Count) (A5)

B1 ← Person ⋈ID = ActorID Starring

B2 ← B1 ⋈MovieID = Id DigitalDVD

B3 ← B2 ⋈DigitalMovie.ID = MediaID Borrows

B4 ← σ ActorFlag = TRUE B3

B5 ← σ Person.ID, Fname, Name, COUNT(MediaID)(B4)

ρ(Person.Id, Fname, Lnmae, Count) (B5)

C1 ← A5 U B5

C2 ← σ(Fname, Name, SUM(Count)) (C1)

ρ(Fname, Lname, LentCount) (C2)

𝜋(C2)

* 1. SELECT Data.Fname,Data.Lname,SUM(Count) AS LentCount

FROM(

SELECT Person.ID,Person.Fname, Person.Lname, COUNT(Borrows.MediaID) AS Count

FROM Person, Borrows, PhysicalDVD,Starring

WHERE Person.ActorFlag = TRUE AND Person.ID=Starring.ActorID

AND Starring.MovieID=PhysicalDVD.ID AND Borrows.MediaID=PhysicalDVD.ID

GROUP BY Person.ID

UNION ALL

SELECT Person.ID,Person.Fname, Person.Lname, COUNT(Borrows.MediaID) AS Count

FROM Person, Borrows, DigitalMovie,Starring

WHERE Person.ActorFlag = TRUE AND Person.ID=Starring.ActorID

AND Starring.MovieID=DigitalMovie.ID AND Borrows.MediaID=DigitalMovie.ID

GROUP BY Person.ID) AS Data

GROUP BY Data.ID

ORDER BY LentCount DESC

LIMIT 1;

* 1. 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)
  2. A1 ← Artist ⋈ID = ArtistID PhysicalDisc

A2 ← A1 ⋈PhysicalDisc.ID = MediaID Borrows

A3 ← A2 ⋈Patron ID = ID LibraryCard

A4 ← σ Artist.ID, Artist.name, Length \* COUNT(MediaID) (A3)

ρ(Artist.ID, Artist.name, Count) (A4)

B1 ← Artist ⋈ID = ArtistID DigitalAlbumDisc

B2 ← B1 ⋈DigitalAlbum.ID = MediaID Borrows

B3 ← B2 ⋈Patron ID = ID LibraryCard

B4 ← σ Artist.ID, Artist.name, Length \* COUNT(MediaID) (B3)

ρ(Artist.ID, Artist.name, Count) (B4)

C1 ← A4 U B4

C2 ← σ(Name, SUM(Count)) (C1)

ρ(Name, ListenTime) (C2)

𝜋(C2)

* 1. SELECT Data.Name,SUM(Count) AS ListenTime

FROM(

SELECT Artist.ID,Artist.Name,PhysicalDisc.Length\*COUNT(Borrows.MediaID) AS Count

FROM Artist, Borrows,PhysicalDisc,LibraryCard

WHERE Artist.ID=PhysicalDisc.ArtistID AND PhysicalDisc.ID=Borrows.MediaID AND Borrows.PatronID=LibraryCard.ID

GROUP BY PhysicalDisc.ID

UNION ALL

SELECT Artist.ID,Artist.Name,DigitalAlbum.Length\*COUNT(Borrows.MediaID) AS Count

FROM Artist, Borrows,DigitalAlbum,LibraryCard

WHERE Artist.ID=DigitalAlbum.ArtistID AND DigitalAlbum.ID=Borrows.MediaID AND Borrows.PatronID=LibraryCard.ID

GROUP BY DigitalAlbum.ID

) AS Data

GROUP BY Data.ID

ORDER BY ListenTime DESC

LIMIT 1;

* 1. Provide a list of customer information for patrons who have checked out anything by the most watched actor in the database.
  2. A1 ←Person ⋈ID = ActorID Starring

A2 ←A1 ⋈MovieID = ID PhysicalDVD

A3 ←A2 ⋈PhysicalDVD.ID = MediaID (Borrows)

A4 ←σActorFlag = TRUE (A3)

A5 ←σPerson.ID, Fname, Name, COUNT(MediaID) (A4)

ρ(Person.ID, Fname, Name, Count) (A5)

B1 ←Person ⋈ID = ActorID Starring

B2 ←B1 ⋈MovieID = ID DigitalMovie

B3 ←B2 ⋈DigitalMovie.ID = MediaID (Borrows)

B4 ←σActorFlag = TRUE (B3)

B5 ←σPerson.ID, Fname, Name, COUNT(MediaID) (B4)

ρ(Person.ID, Fname, Name, Count) (B5)

C1 ←A5 U B5

C2 ←σ(ID, Fname, Lname, SUM(Count) (C1)

ρ(ID, Fname, LName, LentCount )(C2)

D1 ←Borrows ⋈MediaID = ID PhysicalDisc

D2 ←D1 ⋈MediaID = ID LibraryCard

D3 ←D2 ⋈ActorID = ID C2

𝜋(D3)

* 1. SELECT LibraryCard.\*

FROM LibraryCard,Borrows,Starring,PhysicalDVD,(SELECT Data.ID,Data.Fname,Data.Lname,SUM(Count) AS LentCount

FROM(

SELECT Person.ID,Person.Fname, Person.Lname, COUNT(Borrows.MediaID) AS Count

FROM Person, Borrows, PhysicalDVD,Starring

WHERE Person.ActorFlag = TRUE AND Person.ID=Starring.ActorID

AND Starring.MovieID=PhysicalDVD.ID AND Borrows.MediaID=PhysicalDVD.ID

GROUP BY Person.ID

UNION ALL

SELECT Person.ID,Person.Fname, Person.Lname, COUNT(Borrows.MediaID) AS Count

FROM Person, Borrows, DigitalMovie,Starring

WHERE Person.ActorFlag = TRUE AND Person.ID=Starring.ActorID

AND Starring.MovieID=DigitalMovie.ID AND Borrows.MediaID=DigitalMovie.ID

GROUP BY Person.ID) AS Data

GROUP BY Data.ID

ORDER BY LentCount DESC

LIMIT 1) AS FinalData

WHERE LibraryCard.ID=Borrows.PatronID AND Borrows.MediaID=PhysicalDVD.ID AND PhysicalDVD.ID=Starring.MovieID

AND Starring.ActorID=FinalData.ID

GROUP BY LibraryCard.ID;

* 1. Provide a list of artists who authored the albums checked out by customers who have checked out more albums than the average customer.
  2. A1 ← Person ⋈ CardID = ID LibraryCard
  3. A2 ← A1 ⋈ LibraryCard.ID = PatronID Borrows

A3 ← A2 ⋈ MediaID = ID PhysicalDisc

A4 ← σ PatronFlag = TRUE (A3)

A5 ← σ ArtistID, Fname, Name, Address, COUNT(MediaID) (A4)

ρ(ArtistID, Fname, Name, Address, AlbumCount) (A5)

B1 ← Person ⋈ CardID = ID LibraryCard

B2 ← B1 ⋈ LibraryCard.ID = PatronID Borrows

B3 ← B2 ⋈ MediaID = ID DigitalAlbum

B4 ← σ PatronFlag = TRUE (B3)

B5 ← σ ArtistID, Fname, Name, Address, COUNT(MediaID) (B4)

ρ(ArtistID, Fname, Name, Address, AlbumCount) (B5)

C1 ← A5 U B5

D1 ← Artist ⋈ID = ArtistID C1

𝜋Name(D1)

* 1. SELECT Artist.Name

FROM Artist,(

SELECT PhysicalDisc.ArtistID,Person.Fname,Person.Lname, LibraryCard.Address, COUNT(Borrows.MediaID) AS AlbumCount

FROM Person,LibraryCard,Borrows,PhysicalDisc

WHERE Person.PatronFlag=TRUE AND Person.CardID=LibraryCard.ID AND LibraryCard.ID=Borrows.PatronID

AND Borrows.MediaID=PhysicalDisc.ID

GROUP BY PhysicalDisc.ID

UNION ALL

SELECT DigitalAlbum.ArtistID,Person.Fname,Person.Lname, LibraryCard.Address, COUNT(Borrows.MediaID) AS AlbumCount

FROM Person,LibraryCard,Borrows,DigitalAlbum

WHERE Person.PatronFlag=TRUE AND Person.CardID=LibraryCard.ID AND LibraryCard.ID=Borrows.PatronID

AND Borrows.MediaID=DigitalALbum.ID

GROUP BY DigitalAlbum.ID

)AS Data

WHERE Artist.ID=Data.ArtistID

GROUP BY Artist.ID;

**GUIDELINES FOR INSERTING AND DELETING ENTITIES**

Inserting an Album:

INSERT INTO Orders VALUES(int Order#,date OrderDate,ESSN (FK))

INSERT INTO Artist VALUES(int ID#, 'Artist Name');

INSERT INTO PhysicalDisc VALUES(int ID#,int stock,'Title',int year,int length,'Rating','Location’,int OrderNumber,int ArtistID# (FK));

SAMPLE:

INSERT INTO Orders VALUES(300,'2021-04-20',123456789);

INSERT INTO Artist VALUES(65,'Billy Joel');

INSERT INTO PhysicalDisc VALUES(200,1,'The Stranger',1977,2554,'Everyone','Music',300,65);

Assuming that the Album comes from an order, and that the employee who made the order is already in the system, first the order in which the album arrived must be inserted, then the artist that sings the album must be inserted if not already in the system, then the album can be inserted into the system in digital or physical format (the physical format is shown above, digital would simply mean switching PhysicalDisc for DigitalAlbum). It is this order because an Album needs an order number and an artistID as its foreign keys. NOTE ALBUM ID# MUST BE UNIQUE FROM EVERY OTHER MEDIA ID#.

Inserting a Track:

INSERT INTO Orders VALUES(int Order#,date OrderDate,ESSN(FK))

INSERT INTO Artist VALUES(int ID#, 'Artist Name');

INSERT INTO PhysicalDisc VALUES(int ID#,int stock,'Title',int year,int length,'Rating','Location,int OrderNumber(FK),int ArtistID# (FK));

INSERT INTO Track VALUES(int ID#,'Title',int TrackNumber,'Genre',int AlbumID#(FK),int DiscID# (FK))

SAMPLE:

INSERT INTO Orders VALUES(301,'2021-04-20',123456789);

INSERT INTO Artist VALUES(66, 'Meat Loaf');

INSERT INTO PhysicalDisc VALUES(201,1,'Bat Out of Hell',1977,2554,'Everyone','Music',301,66);

INSERT INTO Track VALUES(101,'Paradise By The Dashboard Light',1,'Rock',201,NULL);

Assuming that the track exists on an album that was ordered, and the employee who made the order is already in the system, first the order in which the album containing the track arrived must be inserted, then the artist who sings the album must be inserted if not already in the system, then the album can be inserted into the system, followed by inserting the track.  If one wanted to insert the track to a digital album one would simply switch PhysicalDisc for Digital Album.  It is this order because the track needs an album ID which needs the aforementioned other foreign keys described in the album section above. NOTE TRACK AND ALBUM ID# MUST BE UNIQUE FROM OTHER TRACKS AND MEDIA RESPECTIVELY.

Inserting a Movie:

INSERT INTO Person VALUES(int ID#, ‘First Name’,’Last Name’,bool PatronFlag,int CardID# (FK), bool ActorFlag,bool AuthorFlag,bool DirectorFlag(TRUE), bool EmployeeFlag, int ESSN (FK))

INSERT INTO PhysicalDVD VALUES(int ID#, int Stock, ‘Location’, int Year, ‘Genre’, ‘Title’, int Length, ‘rating’, int OrderNumber(FK), int DirectorID(FK))

SAMPLE:

INSERT INTO Person VALUES(100, 'Quentin','Tarantino',FALSE,NULL, FALSE,FALSE,TRUE, FALSE, FALSE);

INSERT INTO PhysicalDVD VALUES(1000, 1, 'Upstairs', 1992, 'Noir', 'Reservoir Dogs', 7200, 'R', 1, 100);

Assuming that the movie was obtained not through an order (for information if it had been ordered consult the similar approach in the album and track section), the PhysicalDVD itself can be directly inserted into the database.  If the director is not already in the database, they must be inserted as a person with a director flag set to TRUE.  Then the movie can be inserted.  If the movie is a digital copy, one simply has to change PhysicalDVD to DigitalMovie. NOTE MOVIE ID MUST BE UNIQUE FROM EVERY OTHER MEDIA.

Inserting a Book:

INSERT INTO Person VALUES(int ID#, “First Name”,”Last Name”,bool PatronFlag,int CardID# (FK), bool ActorFlag,bool AuthorFlag(TRUE),bool DirectorFlag, bool EmployeeFlag, int ESSN (FK))

INSERT INTO PhysicalBook VALUES(int ID#, int Stock, ‘Location’, int ISBN, int OrderNumber, int AuthorID (FK))

INSERT INTO PBookInfo VALUES(int ISBN (FK), int Year, ‘Genre’, ‘Title’, int Length, int Chapters)

SAMPLE:

INSERT INTO Person VALUES(2000, 'Alduous','Huxley',FALSE,NULL, FALSE,TRUE,FALSE, FALSE, NULL);

INSERT INTO PhysicalBook VALUES(1983, 1, 'Aisle 34', 9999888877776, 1, 2000);

INSERT INTO PBookInfo VALUES(9999888877776, 1932, 'Dystopia', 'Brave New World', 311, 15);

Assuming that the book was obtained not through an order (for information if it had been ordered consult the similar approach in the album and track section), the book itself can be directly inserted into the database, followed by information about the book.  If the author is not already in the database, they must be inserted as a person with the author flag set to TRUE.  Then the book can be inserted along with its info.  If the book is a digital copy, one simply has to change PhysicalBook to AudioBook and PBookInfo to ABookInfo. NOTE: BOOK ID MUST BE UNIQUE ACROSS ALL MEDIA TYPES.

Inserting an Artist:

INSERT INTO Artist VALUES(int ID#, ‘Artist Name')

SAMPLE:

INSERT INTO Artist VALUES(808,’Drake’);

Artists can be directly inserted into the database.

Inserting a Patron:

INSERT INTO LibraryCard VALUES(int ID#, 'login', 'password', 'email address', 'Payment Info')

INSERT INTO Person VALUES(int ID#, 'First Name', 'Last Name', bool Patron Flag=TRUE, int CardID# (FK), bool Actor Flag, bool Author Flag, bool Director Flag, bool Employee Flag, int Employee SSN (FK))

INSERT INTO LibraryCard VALUES(24601, 'jvaljean', 'password', 'miserab@gmail.com', '1233211233321');

INSERT INTO Person VALUES(24601, 'Jean', 'Valjean', TRUE, 24601, FALSE, FALSE, FALSE, FALSE, NULL);

To insert a patron into the database, one must first insert a new library card into the system and then insert them as a person as well, with the patron flag set to TRUE.

Deleting a Track:

DELETE FROM TRACK WHERE ID=(int ID#)

SAMPLE:

DELETE FROM Track WHERE ID=101;

Since Track does not cascade to other entities it can be deleted without an null reference worries.

Deleting an Album:

DELETE FROM TurnsInto WHERE MediaID=(int ID#)

DELETE FROM Track WHERE ID=(int ID#)

DELETE FROM PhysicalDisc WHERE ID=(int ID#)

SAMPLE:

DELETE FROM TurnsInto WHERE MediaID=200;

DELETE FROM Track WHERE ID=100;

DELETE FROM PhysicalDisc WHERE ID=200;

For an Album to be deleted, its order transformation must be deleted, then all of its tracks must also be deleted beforehand, then once its tracks are deleted, the album itself can be deleted.  In the instance of a digital album, one would simply replace PhysicalDisc with DigitalAlbum. Orders is not deleted because the order could contain multiple different media.

Deleting a Movie:

DELETE FROM TurnsInto WHERE MediaID=(int ID#)

DELETE FROM Starring WHERE MovieID=(int ID#)

DELETE FROM PhysicalDVD WHERE ID=(int ID#)

SAMPLE:

DELETE FROM TurnsInto WHERE MediaID=1000;

DELETE FROM Starring WHERE MovieID=1000;

DELETE FROM PhysicalDVD WHERE ID=1000;

For a movie to be deleted, its order transformation must be deleted, then all references of starring actors must be deleted, then once those are deleted, the movie itself can be deleted. In the instance of a digital movie, one would simply replace PhysicalDVD with DigitalMovie. Orders is not deleted because the order could contain multiple different media.

Deleting an Audiobook:

DELETE FROM TurnsInto WHERE MediaID=(int ID#)

DELETE FROM PhysicalBook WHERE ID=(int ID#)

DELETE FROM PBookInfo WHERE ISBN=(int ISBN)

SAMPLE:

DELETE FROM TurnsInto WHERE MediaID=1983;

DELETE FROM PhysicalBook WHERE ID=1983;

DELETE FROM PBookInfo WHERE ISBN=9999888877776

For an audiobook to be deleted, its order transformation must be deleted, then the book can be deleted, then the information about the book can be deleted.  In the instance of a digital audiobook, simply change PhysicalBook to AudioBook and PBookInfo to ABookInfo. Orders is not deleted because the order could contain multiple different media.

Deleting an Artist:

DELETE FROM TurnsInto WHERE MediaID=(int ID#)

DELETE FROM Track WHERE ID=(int ID#)

DELETE FROM PhysicalDisc WHERE ID=(int ID#)

DELETE FROM ARTIST WHERE ID=(int ID#)

SAMPLE:

DELETE FROM TurnsInto WHERE MediaID=200;

DELETE FROM Track WHERE ID=100;

DELETE FROM PhysicalDisc WHERE ID=200;

DELETE FROM ARTIST WHERE ID=65;

For an artist to be deleted, first every album by them must be deleted, once all that encompasses that is complete, the artist can be removed.

Deleting a Patron:

DELETE FROM Preference WHERE PatronID=(int ID#)

DELETE FROM Feedback WHERE PatronID=(int ID#)

DELETE FROM Borrows WHERE PatronID=(int ID#);

DELETE FROM LibraryCard WHERE ID=(int ID#);

UPDATE Person

SET PatronFlag=FALSE, CardID=NULL

WHERE CardID=(int ID#);

SAMPLE:

DELETE FROM Preference WHERE PatronID=24601;

DELETE FROM Feedback WHERE PatronID=24601;

DELETE FROM Borrows WHERE PatronID=24601;

DELETE FROM LibraryCard WHERE ID=24601;

UPDATE Person

SET PatronFlag=FALSE, CardID=NULL

WHERE CardID=24601;

For a patron to be deleted, first all of their preference, feedback, and borrowed book information must be deleted, then their library card must be deleted, then their person must be changed to not a patron anymore, the person itself is not deleted in case they remain an employee or any other job and need to stay in the database.