Trinity Evans and Emma Hughson

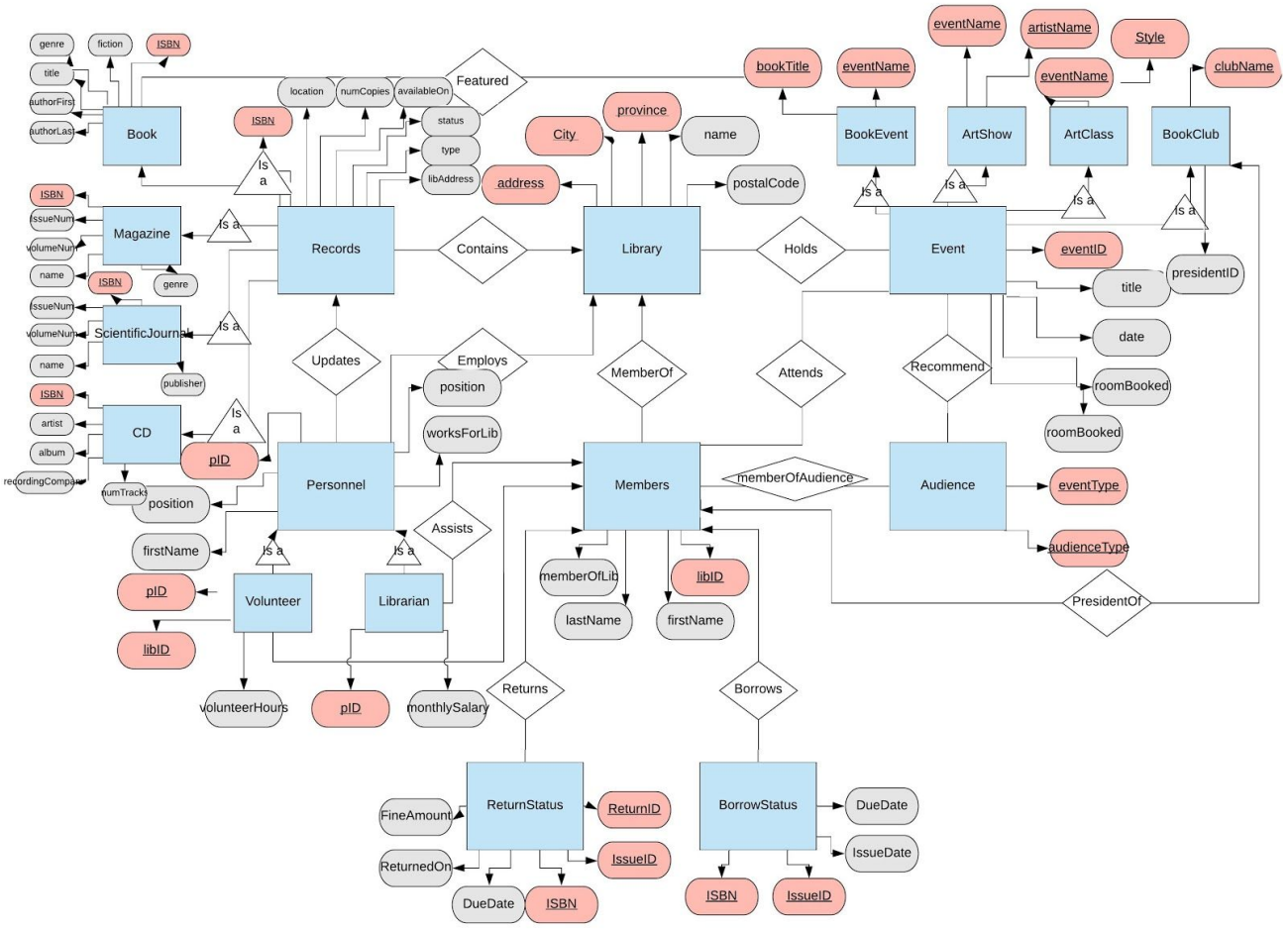
Library Mini Project

Simon Fraser University

PART 2: Project Specifications

- Library has print books, online books, magazines, scientific journals, CDs, records, etc.
- People can borrow the items from library and return by the due date.
- People may be subject to fines if they do not return items by the due date.
- Library also holds book clubs, book related events, art shows, film screenings, etc.
- Library events are recommended for specific audiences.
- Library events are held on library social rooms.
- People can attend library events for free.
- Library also has personnel and record keeping for personnel.
- Library also keeps records of items (books, etc.) that might be added to library in the future.
- A book is known by its ISBN. We also store its title, author publisher
- A magazine is known by its ISBN. We also store its volume and issue number, along with the genre of magazine, magazine name and publisher
- A scientific journal is known by its ISBN. We also store its volume and issue number, along with the journal's name and publisher.
- A cd is known by its ISBN. We also store the name of the artist and album, recording company and the number of tracks on the album.
- A record stores management information. A record stores number of copies, type of book, availability, when it is available (is it available in the future?), and shelf location. Record is known by a unique ID(ISBN).
- We also store information about who has borrowed a record, date borrowed and due date
- A event holds information about the type of event, title, when it is, and location.
- A person is known by their unique library card number. Also store the books they have, and first and last name.
- A librarian is known by their unique personnel number. We also store employee first name, last name and the records they manage.
- Library is known by its address, city and province. We also store postal code information and name of the library.

PART 3: E/R diagram



PART 4 and 5: Does your design allow anomalies and SQL Schemas?

Entity Schemas:

· Members(LibID, firstName, lastName, memberOF)
· Records(ISBN, locationNumber, numCopies, availableOn, status, recordType, LibraryAddress)
· BorrowStatus(IssueID, ISBN[FK-Records], IssueDate, DueDate)
· Returns(ReturnID, IssueID[FK-BorrowStatus], ISBN[FK-Records], DueDate, ReturnedOn, fineAmount)
· Personnel(pID, position, firstName, lastName, worksForLib)
· Event(eventID, title, date, eventType, roomBooked)
· Audience(eventType[FK-Event], audienceType)
· Library(address, province, city, name, postalCode)
· Book(ISBN[FK-Records], fiction, genre, title, authorFirstName, authorLastName, publisher)
· ScientificJournal(ISBN[FK-Records], IssueNumber, VolumeNumber, JournalName, publisher)
· Magazine(ISBN[FK-Records], IssueNumber, VolumeNumber, magazineName, genre, publisher)
· Volunteer(pID[FK-Personnel], LibID[FK-Members], VolunteerHours)
· Librarian(pID[FK-Personnel], monthlySalary)
· CD(ISBN[FK-Records], artistName, albumName, RecordingCompany, numTracks)
· BookClub(clubName[FK-Event], presidentID[FK-Members])
· BookEvent(eventName[FK-Event], BookTitle[FK-Book])
· ArtShow(eventName[FK-Event],artistName)
· FilmScreening(eventName[FK-Event] , movieName)
· ArtClass(eventName[FK-Event], style)

Relationship Schemas:

· memberOfAudience(eventType[FK-Audience], audienceType[FK-Audience], memberID[FK-Members])
· Attends(eventID[FK-Event], LibID[FK-Members])
· Updates(pID[FK-Personnel], ISBN[FK-Records],)
· Recommend(eventID[FK-Event], eventType[FK-Audience], audienceType[FK-Audience])
· Holds(eventID[FK-Event], address[FK-Library], city[FK-Library], province[FK-Library])
· Returned(ISBN[FK-Records], returnID [FK-Returns], LibID[FK-Members])
· Borrows(ISBN[FK-Records], LibID[FK-Members], IssueID[FK-BorrowStatus])
· memberOf(LibID[FK-Members], city[FK-Library], address [FK-Library], province [FK-Library],)
· HelpRequest(pID[FK-Personnel], LibID[FK-Members])

Proof of No Bad-FDS:

- As shown below, the primary key chosen for each entity set was associated with a Non-trivial functional dependency. Each of these non-trivial functional dependencies show that every primary key will have one value for each of the attributes on the

right hand side. This proves that there can be no update anomalies because each tuple is identified by a key that is unique in value that is not shared by any other tuple in the specific table and each key has one value for each attribute that is associated with it. All the resulting non-trivial functional dependencies are shown in BCNF's at the beginning but are also deduced in each of the non-trivial functional dependencies below.

- Each functional dependency below has a paragraph associated with it explaining why the key was chosen and why it does not have any bad FD's.

BCNF:

LibID->firstName, lastName, memberOf
ISBN-> LocationNumber, numCopies, availableOn, status, recordType, LibraryAddress
IssueID, ISBN->DueDate, IssueID
ReturnID, IssueID, ISBN->DueDate, returnedOn, fineAmount
pID->position, firstName, lastName, worksForLib
eventID-> title, date, eventType, roomBooked
address, city, province->name, postalCode
pID, LibID->volunteerHours
pID->monthlySalary
ISBN-> fiction, genre, authorLN, publisher, title
ISBN, volumeNumber, IssueNumber->magazineName, genre, publisher
ISBN, volumeNumber, IssueNumber->JournalNumber, publisher
ISBN-> artistName, albumName, RecordingCompany, numTracks
clubName->presidentID

Non-trivial Function Dependencies for Members Entity:

- LibID -> firstName
- LibID -> lastName
- LibID-> memberOf
- LibID ->firstName, lastName
- LibID->firstName, memberOf
- LibID->lastName, memberOf
- LibID->firstName, lastName, memberOf

Since a Library ID is a unique number it can only be associated with one name in the library, as well each library has library ID's that are all unique. Each library has a unique library ID, and so every library id will have a unique value for the attributes of firstName, lastName and memberOf. As such, it will not allow anomalies to be entered because each tuple is unique.

Non-trivial Functional Dependencies for Records Entity:

- ISBN -> locationNumber

- ISBN-> numCopies
- ISBN -> availableOn
- ISBN -> status
- ISBN -> recordType
- ISBN-> libraryAddress
- ISBN -> LocationNumber, numCopies
- ISBN -> LocationNumber, availableOn
- ISBN -> LocationNumber, status
- ISBN -> LocationNumber, recordType
- ISBN -> LocationNumber, LibraryAddress
- ISBN-> numCopies, availableOn
- ISBN-> numCopies, status
- ISBN-> numCopies, recordType
- ISBN-> numCopies, LibraryAddress
- ISBN->availableOn, status
- ISBN->availableOn, recordType
- ISBN->availableOn, libraryAddress
- ISBN->status, recordType
- ISBN->status, LibraryAddress
- ISBN->recordType, LibraryAddress
- ISBN-> LocationNumber, numCopies, availableOn
- ISBN-> LocationNumber, numCopies, status
- ISBN-> LocationNumber, numCopies, recordType
- ISBN-> LocationNumber, numCopies, libraryAddress
- ISBN-> LocationNumber, availableOn, status
- ISBN-> LocationNumber, availableOn, recordType
- ISBN-> LocationNumber, availableOn, LibraryAddress
- ISBN-> LocationNumber, status, recordType
- ISBN-> LocationNumber, status, LibraryAddress
- ISBN-> LocationNumber, recordType, LibraryAddress
- ISBN->numCopies, availableOn, status
- ISBN->numCopies, availableOn, recordType
- ISBN->numCopies, availableOn, LibraryAddress
- ISBN->numCopies, status, recordType
- ISBN->numCopies, status, libraryAddress
- ISBN->numCopies, recordType, LibraryAddress
- ISBN->status, recordType, LibraryAddress
- ISBN->status, availableOn, LibraryAddress
- ISBN->status, availableOn, recordType
- ISBN-> availableOn, recordType, LibraryAddress
- ISBN->LocationNumber,numCopies, availableOn, status
- ISBN->LocationNumber,numCopies, availableOn, LibraryAddress

- ISBN->LocationNumber,numCopies, availableOn, recordType
- ISBN->LocationNumber,numCopies, LibraryAddress, recordType
- ISBN->LocationNumber,numCopies, status, recordType
- ISBN->LocationNumber, numCopies, status, LibraryAddress
- ISBN->LocationNumber, availableOn, status, recordType
- ISBN->LocationNumber, availableOn, status, LibraryAddress
- ISBN->LocationNumber, availableOn, recordType, LibraryAddress
- ISBN->LocationNumber, status, LibraryAddress, recordType
- ISBN->numCopies, status, LibraryAddress, recordType
- ISBN->numCopies, status, availableOn, recordType
- ISBN->numCopies, status, availableOn, LibraryAddress
- ISBN->numCopies, availableOn, LibraryAddress, recordType
- ISBN-> status, availableOn, LibraryAddress, recordType
- ISBN-> LocationNumber,numCopies, availableOn, status, LibraryAddress
- ISBN-> LocationNumber,numCopies, availableOn, status, recordType
- ISBN-> LocationNumber,availableOn, status, recordType, LibraryAddress
- ISBN-> LocationNumber, numCopies, status, recordType, LibraryAddress
- ISBN-> numCopies, availableOn, status, recordType, LibraryAddress
- ISBN-> LocationNumber, numCopies, availableOn, status, recordType, LibraryAddress

Every ISBN is unique. There are no other books with the same ISBN number unless they are a copy. But since the copies is an attribute, ISBN will always have one value for number of copies. There won't be Multiple ISBN's with different number of copies. This idea is also related to each of the other attributes in the Records table because a ISBN can only have one location (i.e., rack number), it will always have one availableOn date (the date the library got their first copy) and the status will be either unavailable or available depending on the number of copies the database has. As well, ISBN will always have the same record type and will always be part of the same library.

<u>Non-trivial Functional Dependencies for BorrowStatus:</u>

- IssueID, ISBN->DueDate
- IssueID, ISBN->IssueDate
- IssueID, ISBN->DueDate, IssueDate

IssueID by itself cannot be the key because Issue ID is associated with the date the borrow was issued and a person can borrow many books on the same day. However, the combination of IssueID and ISBN uniquely determine the other attributes. Because a specific ISBN is associated with an IssueID, it will only have one value for DueDate and issueDate because a person can borrow multiple books on a specific day but one book will have a unique combination of ISBN and IssueID and that has a specific due date and Issue date associated with it.

<u>Non-trivial Functional Dependencies for Returns:</u>

- ReturnID, IssueID, ISBN->DueDate
- ReturnID, IssueID, ISBN->ReturnedOn
- ReturnID, IssueID, ISBN->fineAmount
- ReturnID, IssueID, ISBN->DueDate, returnedOn
- ReturnID, IssueID, ISBN->DueDate, fineAmount
- ReturnID, IssueID, ISBN->returnedOn, fineAmount
- ReturnID, IssueID, ISBN->DueDate, returnedOn, fineAmount

This is similar to BorrowStatus except with returns. A combination of returnID, IssueID and ISBN is required because a returnID and IssueID can be associated with multiple ISBN's but including ISBN in the key makes it so one book is chosen out of the group of returned books that are associated with the ReturnID and IssueID.

Non-trivial Functional Dependencies for Personnel:

- pID->positon
- pID->firstName
- pID->lastName
- pID->worksForLib
- pID->position, firstName
- pID->positon, lastName
- pID->position,worksForLib
- pid->firstName,lastName
- pID->firstName, worksForLib
- pID->lastName,worksForLib
- pID->position, firstName, lastName
- pID->position, firstName, worksForLib
- pID->positon,lastName,worksForLib
- pID->firstName, lastName, worksForLib
- pID->position, firstName, lastName, worksForLib

This is straightforward because a unique pID is associated with one personnel and so no other personnel in that library can have the same pID. As well, this unique identification number will never have multiple values for any of the other attributes.

Non-trivial Functional Dependencies for Event:

- eventID->title
- eventID->date
- eventID->eventType
- eventID->roomBooked
- eventID->title, date
- eventID->title, eventType
- eventID->title, roomBooked
- eventID->date, eventType

- eventID->date, roomBooked
- eventID->eventType, roomBooked
- eventID->title, date, eventType
- eventID->title, date, roomBooked
- eventID->title, eventType, roomBooked
- eventID-> date, eventType, roomBooked
- eventID-> title, date, eventType, roomBooked

Again a unique id is associated with every event and no event can have the same ID even if it is the same type of event, or on the same day with the same title.

Non-trivial Functional Dependencies for Library:

- address, city, province->name
- address, city, province->postalCode
- address, city, province->name, postalCode

This is self-explanatory because in every city and province there is a unique address and no address can be the same in a city and no province can have two cities with the same name. Therefore these determine the postal code and name of the library.

NOTE: Last minute we realized that the library entity is not required and that the database can be made for one individual library but we made it so all libraries across canada are housed in this database. Therefore, this database is for all libraries across the country and can be viewed by all libraries.

Non-trivial Functional Dependencies for Volunteer:

- pID, LibID->volunteerHours
- pID->volunteerHours
- LibID->volunteerHours

There is a unique pID and a unique LibID and therefore these two combined determine volunteer hours. As well these two keys alone can also determine volunteer hours. But a key will never have multiple values for volunteer hours.

Non-trivial Functional Dependencies for Librarian:

- pID->monthlySalary

This is self explanatory because of the unique pID, there is only one possible salary that can be associated with it. A librarian cannot have more than one salary at a given library.

For all item types below that can be held in a library, the ISBN uniquely determines the attributes associated with each attribute in the ISA tables for the records entity. This is because ISBN is associated with one item in the library and one item only and as such determines the other attributes.

Non-trivial Functional Dependencies for Book:

- ISBN ->fiction
- ISBN ->genre
- ISBN ->title
- ISBN ->authorFN
- ISBN ->authorLN
- ISBN->publisher
- ISBN->fiction, genre
- ISBN->fiction, authorFN
- ISBN->fiction, authorLN
- ISBN->fiction, publisher
- ISBN->fiction, title
- ISBN->genre, authorFN
- ISBN->genre, authorLN
- ISBN->genre, publisher
- ISBN->genre, title
- ISBN-> authorFN, authorLN
- ISBN-> authorFN, publisher
- ISBN-> authorFN, title
- ISBN-> authorLN, publisher
- ISBN-> authorLN, title
- ISBN-> publisher, title
- ISBN->fiction, genre, authorFN
- ISBN->fiction, genre, authorLN
- ISBN->fiction, genre, publisher
- ISBN->fiction, genre, title
- ISBN-> genre, authorFN, authorLN
- ISBN-> genre, authorFN, publisher
- ISBN-> genre, authorFN ,title
- ISBN-> genre, authorLN ,publisher
- ISBN-> genre, authorLN ,title
- ISBN-> genre, publisher ,title
- ISBN-> authorFN, authorLN ,publisher
- ISBN-> authorFN, authorLN ,title
- ISBN-> authorFN, publisher,title
- ISBN-> authorLN, publisher,title
- ISBN->fiction, genre, authorFN, authorLN
- ISBN->fiction, genre, authorFN, publisher
- ISBN->fiction, genre, authorFN, title
- ISBN->fiction, genre, authorLN, publisher
- ISBN->fiction, genre, authorLN, title
- ISBN->fiction, genre, publisher, title
- ISBN-> genre, authorFN, authorLN, publisher

- ISBN-> genre, authorFN, authorLN, title
- ISBN-> genre, authorFN, publisher, title
- ISBN-> authorFN, authorLN, publisher, title
- ISBN-> fiction, genre, authorLN, publisher, title

Non-trivial Functional Dependencies for Magazine:

- ISBN, volumeNumber, IssueNumber->magazineName
- ISBN, volumeNumber, IssueNumber->genre
- ISBN, volumeNumber, IssueNumber->publisher
- ISBN, volumeNumber, IssueNumber->magazineName, genre
- ISBN, volumeNumber, IssueNumber->magazineName, publisher
- ISBN, volumeNumber, IssueNumber->genre, publisher
- ISBN, volumeNumber, IssueNumber->magazineName, genre, publisher

Non-trivial Functional Dependencies for Scientific Journal:

- ISBN, volumeNumber, IssueNumber->JournalNumber
- ISBN, volumeNumber, IssueNumber->publisher
- ISBN, volumeNumber, IssueNumber->JournalNumber, publisher

Non-trivial Functional Dependencies for CD:

- ISBN->artistName
- ISBN-> albumName
- ISBN-> RecordingCompany
- ISBN-> numTracks
- ISBN->artistName, albumName
- ISBN->artistName, RecordingCompany
- ISBN->artistName, numTracks
- ISBN-> albumName, RecordingCompany
- ISBN-> albumName, numTracks
- ISBN-> RecordingCompany, numTracks
- ISBN->artistName, albumName, RecordingCompany
- ISBN->artistName, albumName, numTracks
- ISBN->artistName, RecordingCompany, numTracks
- ISBN-> albumName, RecordingCompany, numTracks
- ISBN-> artistName, albumName, RecordingCompany, numTrack

Non-trivial Functional Dependencies for BookClub:

- clubName->presidentID

This is self explanatory because each club has exactly one president and so there will always be one value for every club.