# Database READ ME

Jonathan Thompson

This README describes the design of the ERD for assignment 2. In particular this README will outline each entity, pertinent columns in each entity, relationships with associated cardinalities, and any indices that may be assigned to an entity. Additionally this README will provide an overview of the library system.

**Overview**
This design is to meet the requirements for the library system and associated use cases. In general this database ERD is intended to allow for all the expected actions such as item checkout, user accounts, renewing items, tracking fines, transferring items, cataloguing items, purchasing items, and reserving items. The database design provides the necessary storage and retrieval for these to be performed but it would be the application that uses the database design to actually implement the needed functionality. Therefore it is assumed that there will be an application layer that performs the functionality related to many of these times but much of the functionality can be aided by database actions such as triggers, stored procedures, and table view creations. As is the case with most database systems, it is assumed that primary key columns will be automatically indexed and therefore it is not explicitly added in this ERD. Also all indices are non-clustered and the generic index selection is assumed to be BTree indexing.

**Branch**
The library network is composed of multiple branches. Each branch has an unique id, a name that would identify the branch, and fields for its address. There is an index put on the name column since it is likely users will search for a branch first by name. Additionally there is a bitmap index on the zipCode column since users will also search for branches closest to them. Since there is a low cardinality for numeric responses, the zip code index is a bitmap.

**BranchBook/BranchCD/BranchDVD**
Each branch can have zero or more books, CDs, and/or DVDs. This points to the fact that each branch can have their own copy of a book, cd, or dvd. Additionally a branch may have many copies of the same book, therefore each of these entities have their own unique id.

**Book**,**CD**,**DVD**
The Book, CD, and DVD tables represents inventory that the library system has, once had, or can order from. In short it is the “world” of books, CDs, and DVDs that the library system knows about. Depending on the item, there is pertinent information related to describing that item in various columns(eg. title, rating ,year etc). There is an index on the name column for each table so that one can search the database by name for an item. This could be done by either searching directly on the Book, CD, or DVD table and represents a library system employee searching the “known” world of books to purchase or research. This could also be combined into a view table and joined on the appropriate branch item table so that users can search the library inventory by name. For example creating a view by joining Book, BranchBook, and Branch a searching can be performed to find all books in inventory.

**BookByAuthor/CdByArtistOrBand/DvdByThespian**
These tables represent that a book, a cd, or a dvd can be the result of many contributors. Additionally it makes the assumption that a CD means music and a DVD means movies, as the practice commonly is. Then a book can have many authors, a CD many band members, and a DVD can have many thespians. Therefore these tables are intersection entities that allow this type of detail and cataloging and each have as a primary key the combined keys that were migrated from the participating tables. Additionally for CD since a CD can be described both by band members or by the band itself, each entry has an id that will be populated with either an artist id or a band id. In order to know which id is populated, an enum data type will say either “artist” or “band”. Finally the relationship is 1 to zero-or-many. This is mainly due to database entry allowances but also for such works that may not be associated with a particular author, artist, or actor. For example the Bible has no specified author and in this case there would be no BookByAuthor entry for the Bible but there would be an entry in the Book table for the Bible.

**Author/Artist/Band/Thespian**
These tables represent the producers of the items catalogued in the library system. Each has appropriate columns describing each entry. The BookByAuthor table must have an author by virtue of its existence but an author may have no books produced. So the relationship is 1 to zero-to-many between Author and BookByAuthor. Between Artist/Band and CdByArtistOrBand there is a zero to zero-to-many relationship since only one table will be present in the particular relationship at a time but the not null requirement on the CdByArtistOrBand column artist_BandID guarantees must participate in the relationship. And similar to the relationship between Author and BookByAuthor, there is a one to zero-to-many relationship between Thespian and DvdByThespian. There is an index over the name component of these tables so that an item may be searched by their creator. For example a user may search for all books by a certain author. Again a view could be created to facilitate this.

**Purchase**
This table holds all purchases for a branch. A branch may have zero-to-many purchases but a purchase row must be associated to only one branch. The columns are the foreign key to Branch table, the purchase price of the total order, and the purchase date. The unique identifier, the purchase id, for the table is also passed as a foreign key to the tables DvdPurchase, CdPurchase, and/or BookPurchase. This is because a purchase may include zero to multiple book, cd, and/or dvd purchases. The Purchase table also has an index on the foreign key branchID. Additionally it is worth mentioning that this database design does not account for storing purchase information as it relates to credit card or bank account information. While this database does show purchase information such as amount or product information it does not store credit card or bank account numbers. This is to purposely reduce risk from storing sensitive information and allowing purchases to be done through the application.

**CdPurchase/DvdPurchase/BookPurchase**
Each of these tables represent a component of a total purchase for a branch. Therefore each must be associated with a purchase row and each must be associated with an item. Additionally the item relationship to an item purchase is optional as that item in the “known” universe of items to the library system may not be purchased. Each item purchase table has the two migrated foreign keys as the primary key. Each table also has a cost associated to each item purchase and the quantity of that item purchased.

**Transfer**
This table represents the many parties involved with a transfer and the needed information to catalogue transfers. First each transfer has a unique id to identify it. Also each row has a branch id for both the origin and destination of the transfer, each being associated to one branch row. Additionally each transfer is associated to one reservation. For a transfer back to the owning branch the reservation id would be the original reservation that caused the book to be transferred away from the owing branch. Also each transfer is associated to one or zero users. The case of zero users is for when an item is being transferred back to its owning branch. Lastly the date column records the date of transfer. The index in this table are over the foreign key columns toBranchID, userID, and branchItemID. The index for toBranchID is a bitmap since the cardinality in the column will be low. Also there is no index over the fromBranchID column as most searches are for the destination and not the origin.

**Patron**
The user table is to record information of the patrons of the library and to store actions associated to them. The columns of the Patron table primarily is related to identifying information. There is a password column so that the user can use their id and the password to log into the system to view their information. Additionally there are user identification information. The relations relating to a user are a user may be involved in zero or many transfers, checkouts, reservations, or fines. Additionally there is a boolean indicating if the patron is a representative of a library from a different county. This value is defaulted to false and would be set through a special registration page through the application for library systems of other counties. While the library system will be mostly used by private individuals, patrons also will include other county libraries. This library design will not treat another library differently in that they still will be a patron who checks out and return items at a branch or be fined. The name portions of the table for a library in a different county is for the representative person for that library county who has been designated to access this library system. Finally there is a bitmap index on the boolean column remoteCountyLibrary since the cardinality would be low.

**Reservation**
The reservation table is for storing information related to a reservation of a branch item by a user. Each reservation has a unique id and must be associated with one user through the patronID foreign key. Lastly there is a column for the date the reservation was placed and a boolean column showing if the reservation has been fulfilled. There is an index for patronID so that a search can be done for all reservations by a user. Additionally there is a multicolumn index for branchItemID and date. This is to know which user made the next reservation for an item and to assign a transfer if necessary. Also there is a bitmap index for the fulfilled column to search for reservations not yet fulfilled. Note the fulfilled being set to true also is satisfied by a reservation being canceled.

**Fine**
This table is to record all fines related to users associated with a checkout. Since a user may check an item out of any branch and fines are to the library system, branch identification is not directly stored but can be obtained through the connectivity of the database. Therefore each row in this table must have a patron id and a checkout id, but the converse is not true. Additionally there are columns for the date the fine was paid, the amount, and a boolean indicating if it is paid. There is a multicolumn index for patronID and checkOutID to find a fine for a user for a certain checkout. Also there is an index for the paid column and because this a boolean column

with very low cardinality a bitmap index is used. In this way a search to find all unpaid fines is faster. Also just as in the Purchase table the Fine table only stores information related to the fine and if it was paid and it does not store information related to payment information such as credit card information and bank account information. Again this is to reduce the risk associated with storing sensitive information. Since payment will be done through the application and possibly 3rd party software, it is unnecessary to store sensitive information that may be compromised. Also this reduces complexity to store this information.

**Checkout**
This table is store the checkout information that happens in the library system. A checkout must be associated with one user and therefore there is a column for patronID as a foreign key. Also each checkout has a unique checkoutID that is used track all the components of a checkout. Additionally there is a date column to store the date of checkout, the number of items checked out, whether the checkout is overdue, and a boolean indicating if all items are returned. The overdue boolean value is changed to true if any of the component checked out items are overdue. There are indices for patronID, overdue, and itemsReturned. A bitmap index use fore overdue and itemsReturned, respectively, as the cardinality is low. By these indices one can search checkouts by a user, all checkouts that have items not returned, or checkouts that are overdue.

**BranchItem**
This table has foreign keys in each branch book, branch cd, and branch dvd. There for it is a many to one relationship. This allows for the library system to be tracking only branchitems but allows for getting back to the specific item checked out. Each branchitem entity has boolean values showing if it is checked out, reserved, or in transit. Additionally each item entity has 2 foreign keys: the branch to where the item belongs, the branch where it is at currently. Additionally there are booleans showing if the item is checked out, reserved, or in transit.

**BranchCheckout**
Intersection table connecting multiple branch items  and checkouts. This allows for a checkout to track individual items, such if it is returned, overdue, or renewed. This table has two foreign keys as the primary key.

**BookCheckout/CdCheckout/DvdCheckout**
These tables represent a branch item checked out by a user during a checkout. As this is an intersection entity, the primary key is made up of the checkoutID and the branch item ID. Since a user may checkout multiple items in one checkout there is a zero-to-many relationship from Checkout to any of the item checkout tables and an item checkout must be associated to one and only one checkout row. Also there is a zero-to-many relationship from a branch item to its corresponding item checkout table as a branch item can be involved in zero or many checkouts. Additionally each item checkout table has booleans for if a checkout item is overdue, renewed, or returned. Each row also has a date for when a checked out item is due, when it was renewed, and the return date. A checkout item can only be renewed once and if it is renewed the boolean for renewed is turned to true and the due date is then extended by two weeks. For each of these tables there are indices for the boolean columns overdue, renew, and returned have bitmap indexes, respectively, due to their low cardinality. This allows for searches for items in a checkout that have been renewed, are overdue, or returned.

**Librarian**

This table holds the information for librarians of the library system. This allows for librarians at each branch to checkin to the system and navigate the system with enhanced privileges. This would include viewing fines, user accounts, and making purchases, among other things. This table holds the needed information for a librarian to be verified and have some information about which librarian is performing an action. This table is not meant to be a HR table but one related to the activity and maintenance of the library database and application. The columns in this table are first the librarianID. Also there is a foreign key branchID from the Branch table. This is part of a one to zero-or-many relationship as a branch may not have any librarians who are approved to use the system at a given moment or may have many, but a librarian must be from one and only one branch. Also there is a password column, date created column, first name column, and last name column.

**Use Cases**

1. Ordering new items for Library inventory - the library system of books, CDs, and DVDs will be updated by an employee or possibly populated by a vendor. This is what would be the Book, CD, or DVD table. Each represent the books, CDs, and DVDs the Library system can purchase and/or has purchased. A library employee would search the tables, select a book, and purchase it through the application layer. Upon purchase a new row is inserted in the appropriate item purchase table, the Purchase table, and the appropriate branch item table.
2. Searching for items within Library inventory - This search can be done in different ways, depending on who is doing the search. If it is a patron of the library, this can be done by performing a search for a specific item on the appropriate branch item table and the item table joined together. This can table can also be joined to the Branch table to know the name and information of the branch the item is from or located at. This all could be done with either view tables or queries through the application layer.
3. Reserve item for patrons - After searching and finding the item in step 2, the patron can reserve that item through the Reservation table. Since the item information is now on hand from step 2, this information is entered in the Reservation table to know which branch item is being reserved and by which user. Since there is a timestamp field called date associated with each reservation, for an item with multiple reservations the reservations are fulfilled in the order they were reserved. As a reservation is fulfilled or canceled, the fulfilled column is set to true.
4. Transferring items between branches - This is done primarily by the Transfer table. A transfer is uniquely associated with a reservation. This reservation is usually by a user but can be from a branch for a branch. The transfer table then is associated with a branch item, a branch for the destination, and the branch for the source. The transfer back is another transfer row but has the same reservation id associated with it as was with the original transfer that caused it to leave the owning branch or is the last reservation on the item, if necessary. This is facilitated by each branch item having a currentLocation column for the branchID for where the item currently is residing and a boolean showing if the branch item is in transfer.
5. Checking out items - This is fulfilled by the Checkout table and the associated branch item checkout table. This allows for storing all items related to a checkout, as well as the user who did the checkout. There is also pertinent information detailing the checkout.
6. Returning items - When an item is returned the specific row in the item checkout has the boolean column returned set to true. When all items are returned that are associated with a checkout the itemsReturned boolean is set to true.

7. Renewing items - An item will only be allowed to be renewed once. When a specific item of a checkout is renewed, the boolean for renew is set to true and the date on which it was renewed is populated. Additionally the date date column is extended by two weeks.
8. Paying fines - Fines are associated with a checkout and with a user. There is a total amount due and a boolean indicating if it was paid. Additionally there is a date column specifying when the fine was paid. These fields would be populated after the user pays through the payment system via the application.
9. Managing patrons accounts - Library admins and the users are able to log into the User table and view information. There can be many views through this interface that are appropriate for users and most likely will be most of the information related to a user's activities. Users and library admins will be able to update user information. Inserts will be done either by the system or a library admin manually. Deletes can only be done by an admin.