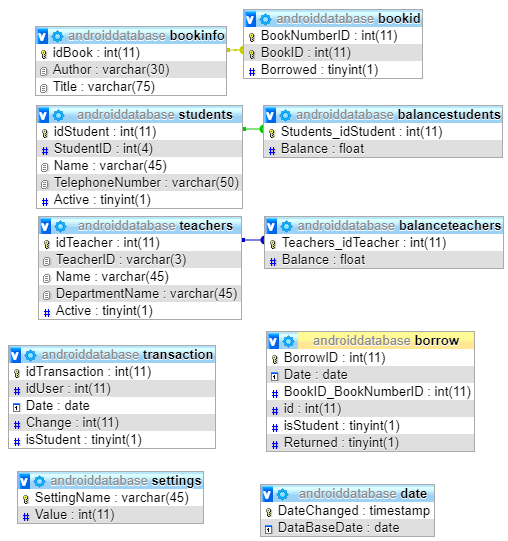
# Table discussion databases

Above you can see the table structure for the databases assignment.

I will start discussing from the top

bookinfo This table holds the main information about types of books

bookid This table holds specific copies of books

students This table holds all the information about a student except for financial data, this to allow different users to have different kinds of access, an open database portal might only be able to show the users balance, but can make changes to the users phone number for example, whereas the computers used for in and out checking of books can have more access to the balance

balancestudents This table holds just the balance of a student, a line is automatically generated as soon as a new student is made by use of a trigger

teachers This table holds all the information about a teacher, except the balance

balanceTeachers This holds the balance of a teacher, is filled by the use of a trigger

transaction This holds all the financial transactions, if someone deposits money, it goes in here, if someone pays a fine, it goes in here, using this table it should always be possible to calculate all the balances again if need be

borrow This is were books go to be borrowed. There is a Boolean to mark if they have returned, this would allow a program to find the history of a book.  
The id here is the internal idStudent or idTeacher, in order to figure out what type of user it is I have added the Boolean isStudent

date Holds the current date that the database program needs to use for testing.  
Automatically gets a timestamp to allow for easy sorting by day added

settings This holds the settings of the database, each field needs to be inserted by a database admin. You can find settings like the maximum amount of books a student can borrow at the same time here.

## Triggers

There are 2 triggers, when a student or teacher gets added it will automatically get the internal student id and use that to make a corresponding row in the balance tables.

# Detailed look into tables

Whilst most fields are self-explanatory, such as names or telephone numbers, there are also other that take a second look

Each table has a unique id, which is either an auto incremented id set by the database itself, or a foreign key linking to such a unique id

An example of this can be seen when looking at the books:

bookinfo holds the information about a type of book, so the author and the title. When you create a book you can then add multiple books, each of which gets inserted in the bookid table using the id from bookinfo.idBook.

In order to then find out which book you have when given a bookid.BookNumberID you can either use separate querries, or better yet just join on the key.

An example of this is in the getBorrowedBooks method where we see the following querry

String **querry** = "SELECT `idBook`,`Author`,`Title`,`BookNumberID`,`Date`,`isStudent`,`id` FROM bookinfo A JOIN bookid B ON A.idBook = B.BookID JOIN borrow C ON B.BookNumberID = C.BookID\_BookNumberID WHERE C.Returned='0' ";

In the above case we use the idbook to join bookinfo and bookid together, and then use the bookNumberID to join the borrow table to this. This allows us to get all the information we want of all the books that are currently on loan.

The settings table is an odd one:

Inserts are not needed in this table, the settings are predefined by me.

| [**SettingName**](http://localhost/phpmyadmin/sql.php?db=androiddatabase&table=settings&sql_query=SELECT+%2A+FROM+%60settings%60++%0AORDER+BY+%60settings%60.%60SettingName%60+ASC&session_max_rows=25&is_browse_distinct=0) | [**Value**](http://localhost/phpmyadmin/sql.php?db=androiddatabase&table=settings&sql_query=SELECT+%2A+FROM+%60settings%60++%0AORDER+BY+%60settings%60.%60Value%60+ASC&session_max_rows=25&is_browse_distinct=0) |
| --- | --- |
| StudentBookBorrowLimit | 6 |
| StudentBookDayLimit | 8 |
| TeacherBookBorrowLimit | 10 |
| TeacherBookDayLimit | 10 |

These values get updated when a setting needs to be altered.

In the transaction and borrow tables I had the problem of either having multiple tables, one for students and one for teachers, or find a way to make the students.idStudent and teachers.idTeachers unique. Neither was a very attractive solution so I opted to simply store a Boolean which allows me to know if an id is from a teacher or a student.

# The program itself

As I mentioned the program was made with a httpserver in mind, therefore the program starts the httpserver, this does mean that the port can be bound when restarting the program.

The console is in its own thread in order to stop hangups due to either thread waiting on the other.

When the console is opened it shows all the commands.

This then communicates to the Communication class in order to talk to the database, the communications class is designed in such a way that it can be used by either the console or an a different client like on an android app.

The communication class has its open and close globally, so when you are inside a connection to the database it is wise not to connect again.

The code has included Javadoc and here and there inline documentation.

When preparing any communication to the database I am opening a preparedStatement with the querry were the values are replaced by ? , then by manually adding the values I can make sure that when I enter an int, it will send an int and not a String from someone trying to do sql injection.

I have added functions that were not part of the assignment since I needed part of the code for them inside the Communication class anyway and it helps with navigation. Examples of these are the print commands and the get available book command.