SQL Databases

Lesson: Using a Relational Database management System and DDL

Task 1 - Create Database and Table using DB Browser

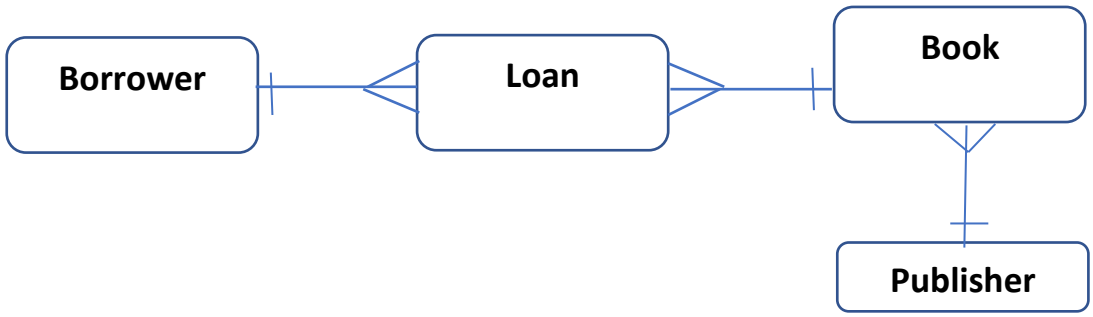
You will now create a library database with four tables using DB Browser for SQLite.

The library contains books that can be on loan to borrowers.

- A borrower can take one or more loans.
- Each loan record belongs to only one borrower.
- A book can be loaned many times.
- A publisher publishes one or more books.
- A book can be published by zero or one publisher.

  For example, school lecture notes are not published by an official publishing house.

The Entity-Relationship (E-R) diagram below is provided to show the tables and the relationships between them.



For the relations below,

- underline the Primary Key and add a * on the Foreign Key
- Write the order in which the tables should be populated in the DBMS

Tables	Order to populate in DB
Borrower (ID, FirstName, Surname, Contact)	3 1
Book(ID, Title, PublisherID*, Damaged)	2
Publisher(ID, Name)	1
Loan(ID, BorrowerID*, BookID*, DateBorrowed)	4

create the one with no foreign keys first (primary key)

the order is important (affects the foreign key)

SQL Databases

The first table you will create is the Borrower table as shown below. After the creation of the table, you will apply some constraints on the table.

Borrower

Column Name	Type
ID	INTEGER
FirstName	TEXT
Surname	TEXT
Contact	TEXT

Table Constraints:

- ID is the **PRIMARY KEY** of the Borrower table
  This means that ID is used to identify a Borrower.

- The value of ID should be **AUTOINCREMENT**
  This means that the ID value increases automatically with each new record inserted.

- All fields are **NOT NULL**
  Each field cannot be empty.

1. Create a folder called **DBTASK**. You will save all your files inside this folder.
2. Open DB Browser for SQLite.
3. Click **File**, then **New Database**.
4. Save your database file as library. The default extension is .db.

   Note: other database file extensions are sqlite/sqlite3/db3
5. Create a table called Borrower with the fields and constraints listed above.
6. Click **Write Changes** or **CTRL + S** to save changes to the database.

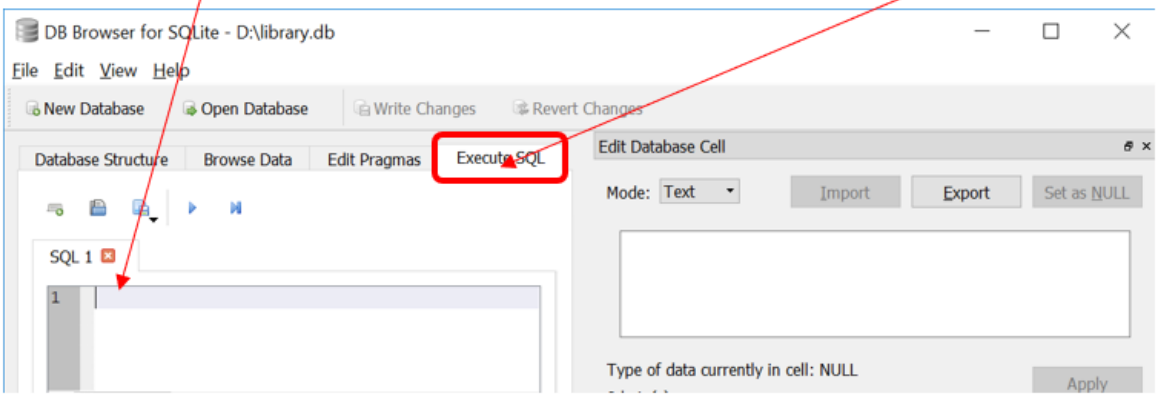
COMMIT in DB Browser for SQLite
In DB Browser for SQLite, the equivalent of the COMMIT command in SQLite is **Write Changes.** This feature saves changes but does not close the database file.

SQL Databases

Task 2 - Insert Records [CREATE]

After creating the library database and Borrower table, you will now add four records to the table.

To enter SQL into DB Browser, after loading the database, click on the **Execute SQL** tab. There is a text area for you to type in your SQL commands.



The INSERT INTO command is used to insert a new record in a table.

For example, to insert a new borrowser, we can use the following SQL:

```
INSERT INTO Borrower(FirstName, Surname, Contact) VALUES
('Peter','Tan', 999)
```

This will insert a new borrower record into the table.

Insert the next 3 records into the table to get this:

Borrower

ID	FirstName	Surname	Contact
1	Peter	Tan	999
2	Sarah	Lee	81111123
3	Kumara	Ravi	94456677
4	Some	User	11111111

Write changes to the database.

SQL Databases

Task 3 – Creating More Tables [CREATE, INSERT]

After creating the library database and Borrower table, you will now create the **Publisher** and **Book** tables and apply the relevant constraints. You will need to take note of special constraints which help to maintain inter-table dependencies and the integrity of related data in different tables. **They will affect the order in which tables are created.**

1. Using DB Browser for SQLite, create the **Publisher** table with the following types and constraints.

   Publisher

Column Name	Type
ID	INTEGER
Name	TEXT

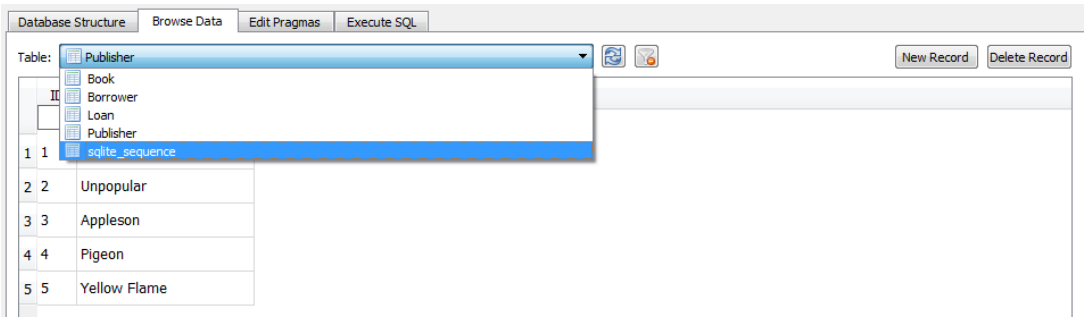
   Table Constraints:

   - ID is the **PRIMARY KEY** of the Publisher table
   - The value of ID should be **AUTOINCREMENT**
   - All fields are **NOT NULL**

2. Insert the following records into the **Publisher** table.

ID	Name
1	NPH
2	Unpop
3	Appleson
4	Squirrel
5	Yellow Flame

3. If you have successfully created the Publisher table, you can view it under the Browse Data tab.



4. Create the **Book** table with the following types and constraints.

> Note:
> The Publisher table has to be created before the Book table because of the foreign key reference to ID in the Publisher table.
>
> **Rule:**
> **Tables with foreign keys have to be created after the referenced tables are created.**

Book

Column Name	Type
ID	INTEGER
Title	TEXT
PublisherID	INTEGER
Damaged	INTEGER

Table Constraints:

- ID is the **PRIMARY KEY** of the Book table.
- The value of ID should be **AUTOINCREMENT**.
- ID, Title and Damaged fields are **NOT NULL**
  Damaged is an attribute that tracks the condition of the book.
  A value of 0 means that the book is not damaged, while a value of 1 means that the book is damaged.
- PublisherID is a **FOREIGN KEY** to ID in the Publisher table.

5. Insert records to Book table as follows:

ID	Title	PublisherID	Damaged
1	The Lone Gatsby	5	0
2	A Winter's Slumber	4	1
3	Life of Pie	4	0
4	A Brief History Of Primates	3	0
5	To Praise a Mocking Bird	2	0
6	The Catcher in the Eye	1	1
123	H2 Computing Ten Year Series	NULL	0

6. Write changes to the database.

Task 4 – Creating Table Using Import

You will now create the **Loan** table by importing a text file into the library database. The types and constraints are described below.

SQL Databases

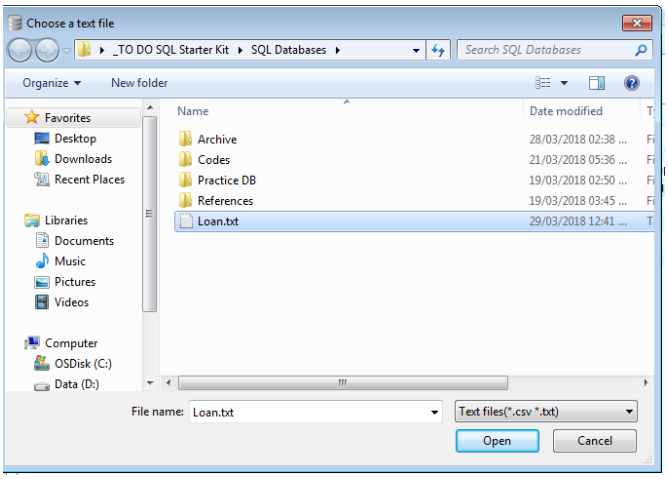
Loan

Column Name	Type
ID	INTEGER
BorrowerID	INTEGER
BookID	INTEGER
DateBorrowed	TEXT

1. Create the **Loan** table using the **Import** feature.

   This feature also allows importing of .TXT and .CSV files.

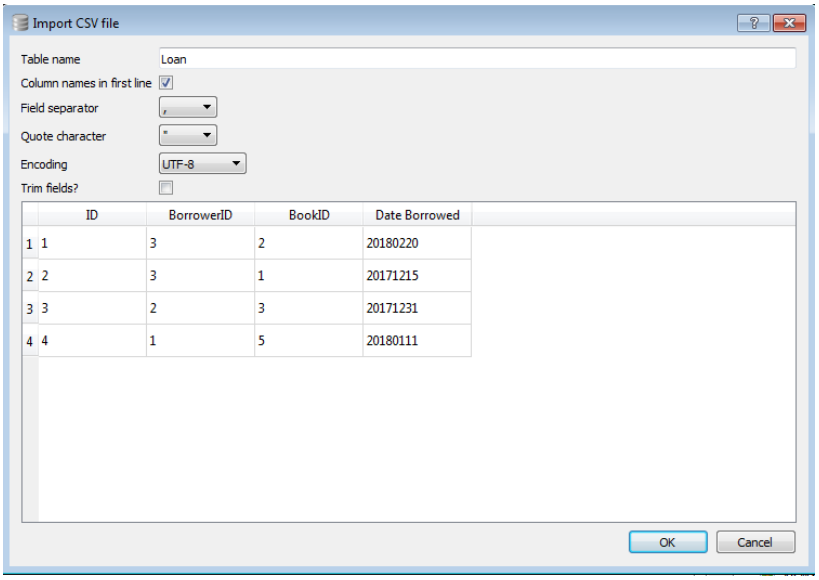
2. Select Loan.txt.(Data files are found in the Google drive data folder)



3. Click **Open**.

4. Tick the option **Column names in the first line**.

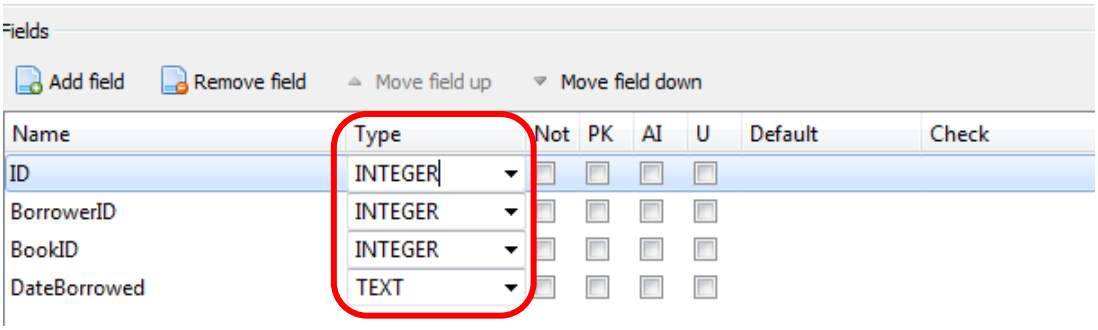
SQL Databases



5. Click OK.
6. Click **Modify Table**.

7. Edit the types according to the description above.

> Note:
> The types for every column in the table are defaulted to TEXT during an import. Hence it is important that you check on the types after an import.



8. Tick the constraints as according to below.

Table Constraints:

- ID is the **PRIMARY KEY** of the Loan table
- The value of ID should be **AUTOINCREMENT**
- ID, BorrowerID and BookID fields are **NOT NULL**

For BorrowerID and BookID of the Loan table, identify the **FOREIGN KEY** constraints.

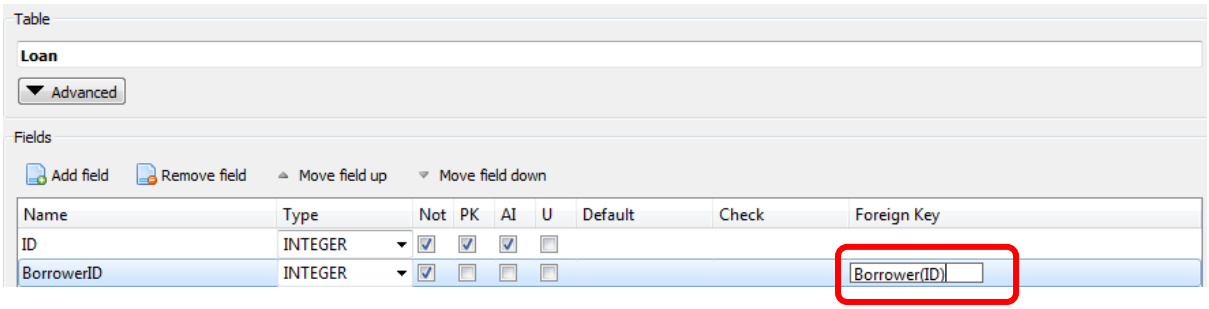
- BorrowerID is a **FOREIGN KEY** to ID in the Borrower table

SQL Databases

- BookID is a **FOREIGN KEY** to ID in the Book table.

9. To create the foreign key for BorrowerID, highlight the BorrowerID attribute. Type **Borrower(ID)** under Foreign Key column.

   This creates a foreign key reference to ID in the Borrower table.



10. Repeat the above step for BookID to create the foreign key reference.

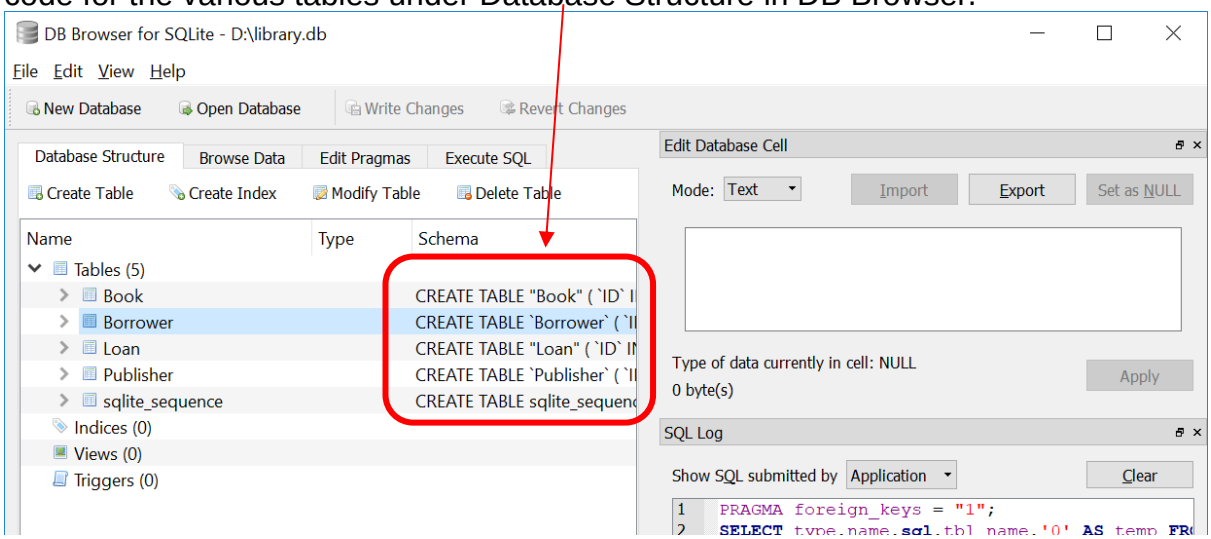
11. View the Loan table from **Browse Data** tab. You should see the following data:

ID	BorrowerID	BookID	DateBorrowed
1	3	2	20180220
2	3	1	20171215
3	2	3	20171231
4	1	5	20180111

12. Write changes to the database.

TASK 5: CREATE TABLE USING SQL COMMAND

The CREATE TABLE command allows you to create a table. You can see the SQL code for the various tables under Database Structure in DB Browser.



SQL Databases

For example, to create the Borrower table, you can key in:

```
CREATE TABLE 'Borrower' (
      'ID' INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
      'FirstName' TEXT NOT NULL,
      'Surname' TEXT NOT NULL,
      'Contact' INTEGER NOT NULL
)
```

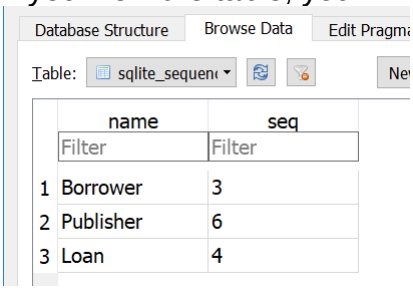
Let's look at the SQL statement carefully.
INTEGER and TEXT are data types, 'ID', 'FirstName', 'Surname' and 'Contact' are field names. The ID field is the primary key of the table Borrower, while Autoincrement means the ID value is automatically given by the database. NOT NULL means that the fields do not accept Null values.

Thus, the syntax for creating tables in SQL is

CREATE TABLE *table_name*(
column1_name COLUMN1_TYPE COLUMN1_CONSTRAINTS,
column2_name COLUMN2_TYPE COLUMN2_CONSTRAINTS,
…
PRIMARY KEY (*column1_name, column2_name,…*),
FOREIGN KEY (*column_name*) **REFERENCES** *table_name*(*column_name*)
)

Also, note that there is an additional table sqlite_sequence in the database.

If you view the table, you will see the following.



This table is used by SQLite so as to keep track of the next number to give for tables with AUTOINCREMENT. It is generated automatically when you have an AUTOINCREMENT field used in SQLite.

Let us look at the Book table next. The SQL code to create the table is:

```
CREATE TABLE 'Book' (
      'ID' INTEGER NOT NULL,
      'Title' TEXT NOT NULL,
      'PublisherID' INTEGER,
      'Damaged' INTEGER NOT NULL,
      FOREIGN KEY('PublisherID') REFERENCES 'Publisher'('ID'),
      PRIMARY KEY('ID')
)
```

Instead of putting the PRIMARY KEY constraint with the ID, you can also put it here

Notice the line starting with FOREIGN KEY. It defines the foreign key, which is the field linking to the primary key of another table. For this example, it is linking the PublisherID field in the Book table to the ID field in the Publisher table.

Quiz

1. Key in SQL statement to create a table Testing with fields name, tag_no and remarks. The fields name and remarks should accept text, while tag_no is an integer automatically incremented. The primary key is tag_no. Name field should not be NULL.

```
CREATE TABLE "Testing" (
    "Name" TEXT NOT NULL,
    "Tag_No" INTEGER,
    "Remarks" TEXT,
    PRIMARY KEY("Tag_No"  AUTOINCREMENT)
);
```

Tip!
You can click ‘Revert Changes’ in DB Browser to remove the recent changes made.

TASK 6: CREATE tables for a new entity and relationship

Write the SQL command to create a table based on the following relation:
`Author (ID, Name, email)`

```
CREATE TABLE Author (
    ID INTEGER PRIMARY KEY,
    Name TEXT,
    Email TEXT
    );
```

A book can have 1 or more authors. An author can write 1 or more books. Write the SQL command to create another table to implement this relationship.

```
CREATE TABLE BookAuthor (
BookID INTEGER,
AuthorID INTEGER,
PRIMARY KEY (BookID, AuthorID),
FOREIGN KEY (BookID) REFERENCES Book(ID),
FOREIGN KEY (AuthorID) REFERENCES Author(ID)
)
```

SQL Databases

TASK 6: DROP TABLE

The DROP TABLE command deletes the entire table.
For example, to remove the loan table, you can key in:

```
DROP TABLE Loan
```

What is the difference between `DELETE FROM Loan` and `DROP TABLE Loan`?

With `DELETE FROM Loan`, you delete all entries from the Loan table, but the Loan table remains there. With `DROP TABLE Loan`, the Loan table will be removed. You cannot insert any entries into Loan table anymore.

Quiz

1. Try keying `DROP TABLE Publisher` to remove the table containing the publishers. Is it possible? Why?
2. What feature in the DBMS is preventing you from deleting the table

The Foreign Key Constraint or referential integrity in the DBMS prevents a table to be deleted when there are records in another table with a foreign key referencing it