CS 5200 Assignment 6

Code

Triggers are automated callback functions, that occur when events occur on the secified table or if defined by row using FOR EACH ROW. Such events included INSERT, DELETE, OR UPDATE.

Audit Triggers are used to log (to track) these type of functions mentioned above. In my trigger I will be creating an audit trail for every of inventory for the book of the Author table.

Hide

```
library(RSQLite)

fpath = "/Users/domschrein/Desktop/CS5200/"
dbfile = "commiteeDB.sqlite"

dbcon <- dbConnect(RSQLite::SQLite(), paste0(fpath,dbfile))
```

Hide

```
DROP TABLE IF EXISTS "Author"
```

Added a few columns to use for my Audit Trigger. “publishinghouse”, “titlespublished”, “inventory” to see if I can log the inventory and type of event occurring while maintaing the integrity of the other data.

This is the table specified and the columns involved are “aid”, “inventory”

Hide

```
CREATE TABLE "Author"(
  "aid" INTEGER PRIMARY KEY NOT NULL,
  "name" varchar(255) NOT NULL,
  "email" varchar(255) NOT NULL,
  "title" VARCHAR(255) NOT NULL,
  "publishinghouse" VARCHAR(255) NOT NULL,
  "titlespublished" INTEGER NOT NULL,
  "inventory" INTEGER NOT NULL

);
```

Hide

```
DROP TABLE IF EXISTS "Author_Audits"
```

Here we are creating the Audit table to track all the “inventory” log transactions. “aid” will be this table’s ID as it comes from Author.Id. “TRIGGER_AuthorAudits” will reflect the type of trigger “entry_date” will keep the timestamp when the record will be created.

Hide

```
CREATE TABLE "Author_Audits"(
  "aid" INTEGER NOT NULL,
  "TRIGGER_AuthorAudits" NOT NULL,
  "entry_date" TEXT NOT NULL
);
```

Here I am creating an insert trigger on Author Table, which will log the insertion of a record

Hide

```
CREATE TRIGGER insert_log AFTER INSERT

ON "Author"

BEGIN

   INSERT INTO "Author_Audits"("aid", "entry_date", "TRIGGER_AuthorAudits") VALUES
(new.aid, datetime('now'), 'INSERT');

END;
```

Here I am creating an update trigger on Author Table, which will log the update of a record

Hide

```
CREATE TRIGGER update_log AFTER UPDATE

ON "Author"

BEGIN

  INSERT INTO "Author_Audits"("aid", "entry_date", "TRIGGER_AuthorAudits") VALUES
(new.aid, datetime('now'), 'UPDATE');

END;
```

Here I am creating a delete trigger on Author Table, which will log the deletion of a record

Hide

```
CREATE TRIGGER delete_log AFTER DELETE

ON "Author"

BEGIN

   INSERT INTO "Author_Audits"("aid", "entry_date", "TRIGGER_AuthorAudits") VALUES
(new.aid, datetime('now'), 'DELETE');

END;
```

TESTING THE TRIGGER This is where I will insert a record into Author table which should result in creating an insert log record in Author_Audits table.

Hide

```
INSERT INTO "Author" (
  "aid",
  "name",
  "email",
  "title",
  "publishinghouse",
  "titlespublished",
  "inventory"
  )

VALUES (1184, 'Jones James', "JJs@JJwrites.com", 'Ph.D', "Morton", 3, 489)
```

Hide

```
SELECT * FROM "AUTHOR"
```

aid <int>	name <chr>	email <chr>	title <chr>	publishinghouse <chr>	titlespublished <int>	inventory <int>
1184	Jones James	JJs@JJwrites.com	Ph.D	Morton	3	489

1 row

Hide

```
SELECT * FROM "Author_Audits";
```

aid <int>	TRIGGER_AuthorAudits <chr>	entry_date <chr>

1184	INSERT	2021-06-22 17:37:47

1 row

Similarly we execute UPDATE or DELETE operations on the Author_Audits table, log record also be updated, using the code below I will test the UPDATE event:

Hide

```
UPDATE "Author" SET "inventory"= 488 WHERE "aid"= 1184;
```

Here we can see the UPDATE Sucessfully implemented along with the insert. Note the “aid” must match from the insert to make changes upon the record when testing the UPDATE fucntion.

Hide

```
SELECT * FROM "Author_Audits";
```

aid <int>	TRIGGER_AuthorAudits <chr>	entry_date <chr>
1184	INSERT	2021-06-22 17:37:47
1184	UPDATE	2021-06-22 17:39:45

2 rows

Here we can see all the triggers from sqlite_master table, using following code:

Hide

```
SELECT name FROM sqlite_master

WHERE type = 'trigger' AND tbl_name = "Author";
```

name <chr>
insert_log
update_log
delete_log

3 rows

One can also delete the triggers using DROP command which is as follows:

Hide

```
DROP TRIGGER 'insert_log';
```

We can also see all the triggers currently logged on Author table, then use AND clause with table name as follows:

Hide

```
SELECT name FROM sqlite_master

WHERE type = 'trigger' AND tbl_name = "Author";
```

name <chr>
update_log
delete_log

2 rows

Hide

```
dbDisconnect(dbcon)
```