Trigger Definition, Use Cases and Explanation

Trigger: A named block of code that is executed or fired automatically when a certain type of SQL statement is executed in a particular database.

In SQLite, Triggers are driven(fired) by EVENT statements such as INSERT, UPDATE, or DELETE. And their order is defined by an execution constraint such as Trigger BEFORE an event or AFTER an EVENT.

TRIGGER : 1

Here we are using our Trigger set 1 to create an Audit Trail Log.

- Our main purpose here is to keep a track record of any topic additions, switches or removals from any given courses.
- By keeping an Audit of what was deleted and when it was deleted. We could efficiently trace accidental edits, deletes OR provide proper information about the changes that were made to the database, in case it is needed.
- Every time a topic is deleted from a course, we will update the AuditCourseTopic Table which contains all essential information, needed to track the changes such as ids of Course and Topic, Name of Course and Topic, Timestamp of events, All these columns are acquired from a join of three tables: CourseTopic, Course and Topic.

For Example: If a new topic say "Duffing's Oscillator" was introduced in our course "Mathematical Modelling", and the trigger can be used to notify users of this new addition.

Code and Operation:

1. The trigger `after_add_topic_to_course` fires after every addition to the linking table CourseTopic which links a course with a topic.

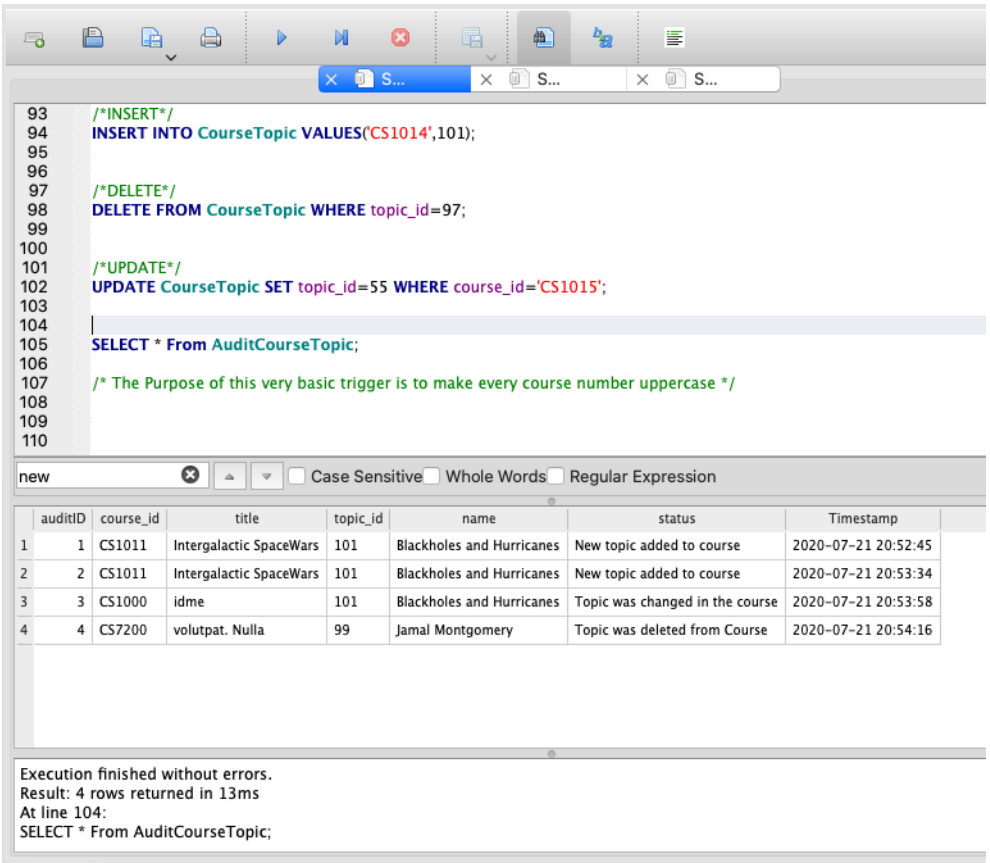
2. Every time a new value in entered. The trigger is fired and 'NEW' references can be exploited to point towards the new values, according to the following table. A NEW value is valid when AFTER statement is used with INSERT, UPDATE, and DELETE Statements

INSERT	NEW references are valid
UPDATE	NEW and OLD references are valid
DELETE	OLD references are valid

3. The specifics of the NEW Course/Topic are fetched from their respective tables via a Select statement, By using "AS" keyword we create an alias to the input value, this alias is then used to populate our Audit Table via INSERT statement.

```
CREATE TRIGGER if not EXISTS after_add_topic_to_course
AFTER
  INSERT ON CourseTopic FOR EACH ROW
  BEGIN
  INSERT INTO AuditCourseTopic (
    course_id, title, topic_id, name, status,
    Timestamp
  )
select
  NEW.course_id as course_id,
  c.title as title,
  NEW.topic_id as topic_id,
  t.name as name,
  "New topic added to course" as status,
  CURRENT_TIMESTAMP as Timestamp
from
  CourseTopic ct,
  Course c,
  topic t
where
  ct.course_id = NEW.course_id
  AND ct.course_id = c.number
  AND ct.topic_id = t.tid;
END;
```

Screenshots of execution and output from the set of triggers for INSERT UPDATE, and DELETE.



Trigger 2: A very basic trigger to update the column containing alphanumeric Course Codes to uppercase.

- SQLite is case insensitive and our Course Number attribute is an alphanumeric primary key. So SQLite perceives “CS500”, “cs500” and “cS500” etc. differently. If we use a check constraint while declaration that will mean, that the user is required to entered it in uppercase, or use the application logic.
- By using a very basic trigger after every INSERT in the course, we can fix this and enforce the constraint on this domain.
- The trigger will make sure that there are no duplicate values for the same course name and also help user input without any case constraints.

```
CREATE TRIGGER if not EXISTS update_course_title_to_uppercase
AFTER INSERT ON Course
FOR EACH ROW
BEGIN

UPDATE Course

  SET

  number = UPPER(number);

END;
```

Screenshot of operation: Cs1014 and cs1015 converted to uppercase: