Yash Pankhania (pankhania.y@northeastern.edu)

Notes on Triggers:

Triggers

• Trigger: procedure that starts automatically if specified changes occur to the DBMS

A trigger has three parts:

• Event

Change to the database that activates the trigger

• Condition

Query or test that is run when the trigger is activated

• Action

Procedure that is executed when the trigger is activated and its condition is true

Trigger Options

• Event can be insert, delete, or update on DB table

• Condition:

Condition can be a true/false statement

All employee salaries are less than $100K

• Condition can be a query

Interpreted as true if and only if answer set is not empty

• **Action** can perform DB queries and updates that depend on:

• Answers to query in condition part

• Old and new values of tuples modified by the statement that activated the trigger

• Action can also contain data-definition commands, e.g., create new tables

When to Fire the Trigger?

• Triggers can be executed once per modified record or once per activating statement

• Row-level trigger versus a Statement Level Trigger

• Trigger looking at the set of records that are modified versus the actual individual values of the old and the new values
• Should trigger action be executed before or after the statement that activated the trigger?
• Consider triggers on insertions
• Trigger that initializes a variable for counting how many new tuples are inserted: execute trigger before insertion
• Trigger that updates this count variable for each inserted tuple: execute after each tuple is inserted (might need to examine values of tuple to determine action)
• Trigger can also be run in place of the action

Trigger Example

```
• CREATE TRIGGER YoungSailorUpdate
AFTER INSERT ON SAILORS
REFERENCING NEW TABLE NewSailors
FOR EACH STATEMENT
INSERT
INTO YoungSailors(sid, name, age, rating)
SELECT sid, name, age, rating
FROM NewSailors N
WHERE N.age <= 18
```

Trouble with Triggers

• Action can trigger multiple triggers
• Execution of the order of the triggers is arbitrary
• Challenge: Trigger action can fire other triggers
• Very difficult to reason about what exactly will happen
• Trigger can fire “itself” again
• Unintended effects possible
• Introducing Triggers leads you to deductive databases
• Need rule analysis tools that allow you to deduce truths about the data

MY SQL limits the use of triggers

• Triggers not introduced until 5.0
• Not activated for foreign key actions

• No triggers on the mysql system database
• Active triggers are not notified when the meta data of the table is changed while it is running
• No recursive triggers
• Triggers cannot modify/alter the table that is already being used For example the table that triggered it

MY SQL Trigger

CREATE TRIGGER <trigger-name> trigger_time trigger_event
ON table_name
FOR EACH ROW
BEGIN
END

• Syntax
• Trigger_time is [BEFORE | AFTER]
• Trigger_event [INSERT|UPDATE|DELETE]
• Other key words – OLD AND NEW
• Naming convention for a trigger
trigger_time_tablename_trigger_event
• Found in the directory associated with the database
• File tablename.tdg – maps the trigger to the correspnoding table
• Triggername.trn contains the trigger definition

Reviewing your trigger

• Go to the trigger directory and read the file (.trg)
Program Data\MySQL\MySQL5.5\data\<db-name>*.trg
• Use the DBMS to locate the trigger for you
Triggers in current schema
SHOW TRIGGERS;
ALL Triggers in DBMS using the System Catalog
SELECT * FROM Information_Schema.Triggers
WHERE Trigger_schema = 'database_name' AND
Trigger_name = 'trigger_name';
select trigger_schema, trigger_name, action_statement
from information_schema.triggers;

Changing your trigger

• There is no edit of a trigger
• CREATE TRIGGER ...
• DROP TRIGGER <TRIGGERNAME>;
• CREATE TRIGGER ...

Events

• MySQL Events are tasks that run according to a schedule.
• An event performs a specific action
• This action consists of an SQL statement, which can be a compound statement in a BEGIN END block
• An event's timing can be either one-time or recurrent
• If recurrent can state an interval that determines how often it gets run
• Can specify a time window to state when the event is active
• an event is uniquely identified by its name and the schema to which it is assigned
• an event is executed with the privileges of its definer/author
• Errors and warnings from an event are written to the log
Events
• CREATE EVENT `event_name`
ON SCHEDULE schedule
[ON COMPLETION [NOT] PRESERVE]
[ENABLE | DISABLE | DISABLE ON SLAVE] --CLUSTERdb
• DO BEGIN
• -- event body
• END;
• DROP EVENT `event_name`
• ALTER EVENT `event_name`

Options for a Schedule

• Run once on a specific date/time:
AT ‘YYYY-MM-DD HH:MM.SS’
e.g. AT ‘2011-06-01 02:00.00′
• Run once after a specific period has elapsed:

AT CURRENT_TIMESTAMP + INTERVAL n
[HOUR|MONTH|WEEK|DAY|MINUTE]
e.g. AT CURRENT_TIMESTAMP + INTERVAL 1 DAY
• Run at specific intervals forever:
EVERY n [HOUR|MONTH|WEEK|DAY|MINUTE]
e.g. EVERY 1 DAY
• Run at specific intervals during a specific period:
EVERY n [HOUR|MONTH|WEEK|DAY|MINUTE] STARTS date
ENDS date
e.g. EVERY 1 DAY STARTS CURRENT_TIMESTAMP + INTERVAL 1
• WEEK ENDS '2012-01-01 00:00.00′

Summary

• Triggers respond to changes in the database
• Allows you to define constraints on the data
• Events allow you to schedule tasks to be done by a calendar
date or an interval

Youtube Link : Database Triggers - React to Table Changes