Lecture 7 SQL-Trigger

CS211 - Introduction to Database

Trigger

- Trigger
  - When define a dynamic constraint
  - When the constraint involves more than one table

- Dynamic
  - Object
  - Predicate
  - Trigger
  - Response

Define all 4 components in a trigger

Trigger - syntax

CREATE TRIGGER trigger_name BEFORE|AFTER INSERT|DELETE|UPDATE

[OF col, {col…}] ON table

[referencing variables]

[FOR EACH ROW]

[WHEN condition]

action

Object

Trigger_time Trigger_event

Predicate

Response

Trigger - MariaDB

```
CREATE TRIGGER trigger BEFORE|AFTER
  INSERT|DELETE|UPDATE
  [OF col, {col…}  ON table
  [referencing variables]
  [FOR EACH ROW]
  [WHEN condition]
    action
```

MariaDB mySQL

```
CREATE TRIGGER trigger BEFORE|AFTER
  INSERT|DELETE|UPDATE ON table
  FOR EACH ROW
  BEGIN
    IF(condition)
      action
    ENDIF;
  END
```

Trigger – referencing variables

```
DELIMITER //
CREATE TRIGGER t1 BEFORE INSERT ON student
FOR EACH ROW
BEGIN
  IF((SELECT count(*) FROM student where sid=new.sid)>0) THEN
      /* reject the insert statement */
     signal sqlstate '45000' set message_text= ‘duplicate’;
  END IF;
END //
DELIMITER ;
```

Issue a statement: INSERT INTO student VALUES('4001', 'Amy', 'F', 20, '03', 'Sophomore');

Trigger – referencing variables

```
CREATE TRIGGER t1 BEFORE INSERT ON student
FOR EACH ROW
BEGIN
  IF(SELECT count(*) FROM student where sid=new.sid>0) THEN
      /* reject the insert statement */
     signal sqlstate '45000' set message_text= ‘duplicate’;
  ENDIF
END
```

INSERT	
Old	New
Not applicable	('4001', 'Amy', 'F', 20, '03', 'Sophomore')

Trigger – referencing variables

sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

INSERT INTO student VALUES('4001', 'Amy', 'F', 20, '03', 'Sophomore');	
Old	New
Not applicable	('4001', 'Amy', 'F', 20, '03', 'Sophomore')

DELETE FROM student WHERE sID = '4001';	
Old	New
('4001', 'Amy', 'F', 20, '03', 'Sophomore')	Not applicable

Update student SET Age=22 WHERE sID = '4001';	
Old	New
('4001', 'Amy', 'F', 20, '03', 'Sophomore')	('4001', 'Amy', 'F', 22, '03', 'Sophomore')

Trigger – For each row

sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

```
CREATE TRIGGER t2 BEFORE UPDATE ON student
FOR EACH ROW
BEGIN
  IF(old.age>39) THEN
      /* action*/
  END IF;
END
```

Row-level triggers execute 5 times

Issue a statement: Update student SET age = age+1;

Trigger – row-level vs. statement-level

- Row-level trigger
  - Row-level triggers execute **once for each row**
  - Used for *data-related* activities, e.g. check if old.age > 39
- Statement-level trigger
  - Statement-level triggers execute **once**
  - Therefore they are not often used for *data-related* activities
  - They are normally used to enforce additional security measures.

Issue a statement: Update student SET age = age+1;

MariaDB/mySQL does not support statement-level trigger

Trigger – before/after

sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

```
CREATE TRIGGER t1 BEFORE|AFTER INSERT ON student
FOR EACH ROW
BEGIN
  IF(SELECT count(*) FROM student where sid=new.sid>0) THEN
      /* reject the insert statement */
     signal sqlstate '45000' set message_text= 'duplicate';
  ENDIF;
END
```

Issue a statement:

```
INSERT INTO student VALUES('4001', 'Amy', 'F', 20, '03', 'Sophomore');
```

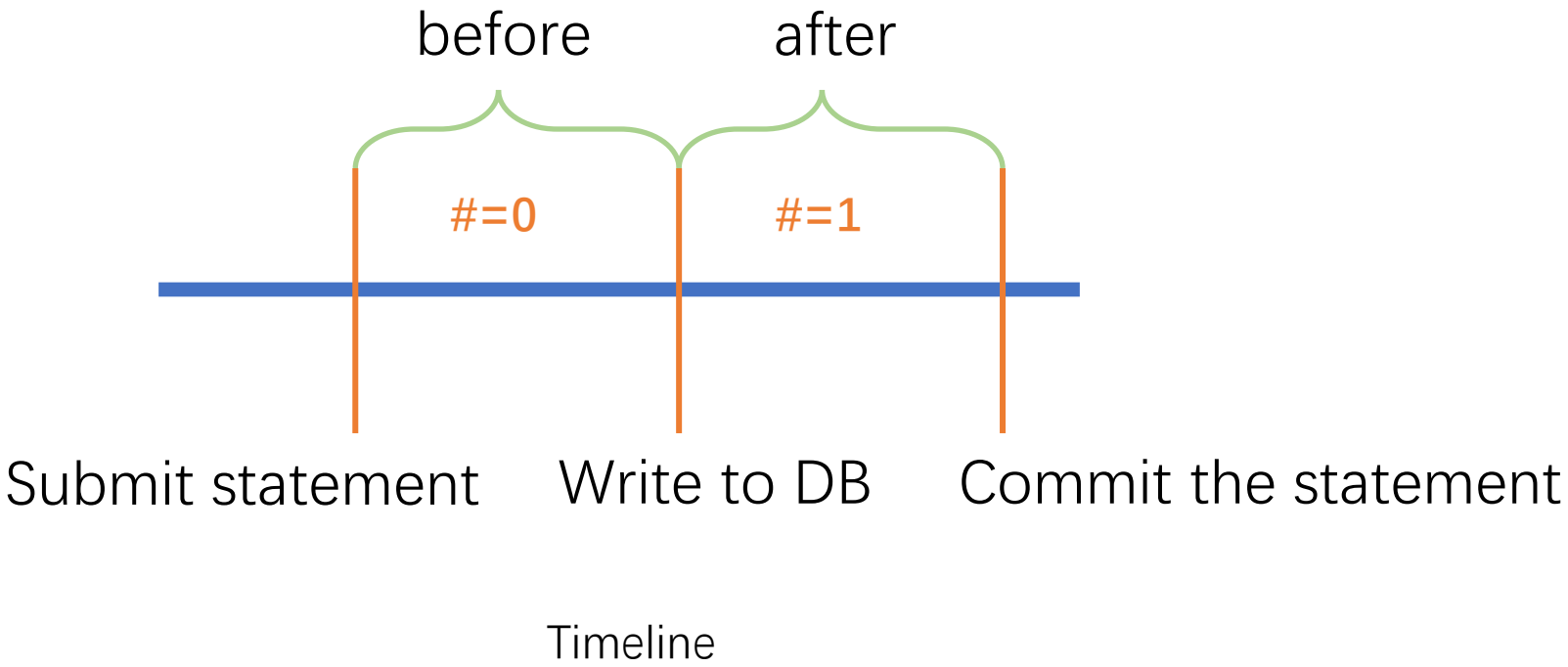
Trigger – before/after

sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

Issue a statement:

INSERT INTO student VALUES('4006', 'Kate', 'F', 20, '03', 'Sophomore');

#: the result of condition in the trigger: SELECT count(*) FROM student where sid=new.sid



Trigger time and event combination

- Until MariaDB 10.2.3, a table could have only one trigger defined for each event/timing combination.

- For example, a table could only have one BEFORE INSERT trigger.

Trigger – example1

- Workload can only be increased
- Design a trigger when *instructor.workload* is updated

```
DELIMITER //
CREATE TRIGGER tWorkload BEFORE UPDATE ON instructor
FOR EACH ROW
BEGIN
  IF(new.workload < old.workload) THEN
    signal sqlstate '45000' set message_text= 'tWorkload';
  END IF;
END//
DELIMITER ;
```

Trigger – example2

- Workload can not be more than 10. Set 10 if it is larger than 10.
- Design a trigger when *instructor.workload* is updated

```
DELIMITER //
CREATE TRIGGER tWorkload BEFORE UPDATE ON instructor
FOR EACH ROW
BEGIN
  IF(new.workload > 10 ) THEN
    set new.workload = 10;
  END IF;
END//
DELIMITER ;
```

Trigger – example3

- Relation stuNCourse(sID, sumCourse)
- Design a trigger when *rc* is inserted, automatically update *stuNCourse*

```
DELIMITER //
CREATE TRIGGER tIncCourse AFTER INSERT ON rc
FOR EACH ROW
BEGIN
  update stuNCourse set sumCourse=sumCourse+1
  where sID = new.sID;
END//
DELIMITER ;
```

sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89

Trigger – example4

sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

- Primary Key *sID* in *student* is allowed to be updated
- Design a trigger when *siD* is updated, automatically update *rc.sID*

```
DELIMITER //
CREATE TRIGGER tSID AFTER UPDATE ON student
FOR EACH ROW
BEGIN
  update rc set sID=new.sID where sID = old.sID;
END//
DELIMITER ;
```

Trigger – example5

sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

- Design a trigger when a student is deleted from *student*, automatically update *sID* as NULL in *rc*

```
DELIMITER //
CREATE TRIGGER tDelStu AFTER DELETE ON student
FOR EACH ROW
BEGIN
  update rc set rc.sID = NULL where sID = old.sID;
END//
DELIMITER ;
```

Trigger – example6

sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

- Design a trigger when a student is deleted from *student*, automatically delete all records of the student in *rc*

```
DELIMITER //
CREATE TRIGGER tDelStu AFTER DELETE ON student
FOR EACH ROW
BEGIN
  delete from rc where sID = old.sID;
END//
DELIMITER ;
```

Trigger – example7

- Inserting a senior into *student* is not allowed

```
CREATE TRIGGER tSenior AFTER INSERT ON student
FOR EACH ROW
BEGIN
  IF(new.grade='senior') THEN
    delete from student where sId=new.sID;
  END IF;
END;
```

×

Within a trigger, it is not permitted to modify a table that is already being used (for reading or writing) by the statement that invoked the trigger.

Trigger – chaining effect

- There is a trigger

```
DELIMITER //
CREATE TRIGGER tDelStu AFTER DELETE ON student
FOR EACH ROW
BEGIN
  delete FROM rc where sID = old.sID;
END//
DELIMITER ;
```

Trigger – chaining effect

- Then add a new trigger

```
DELIMITER //
CREATE TRIGGER tDecCourse AFTER DELETE ON rc
FOR EACH ROW
BEGIN
  update stuNCourse set sumCourse=sumCourse-1
  where sID = old.sID;
END//
DELIMITER ;
```

Trigger – chaining effect

- Issue the statement below

```
DELETE FROM student WHERE sID='4001';
```

- What will happen?

In stuNCourse