SQL – Triggers

Carla Teixeira Lopes

Bases de Dados
Mestrado Integrado em Engenharia Informática e Computação, FEUP

Based on Jennifer Widom slides

Agenda

Triggers

“Event-Condition-Action Rules”

- When event occurs, check condition; if true, do action

Move monitoring logic from apps into DBMS

Enforce constraints

- Beyond what constraint system supports
- Automatic constraint “repair”

Implementations vary significantly

- Introduction: SQL standard
- Examples: SQLite

Triggers in SQL

Create Trigger name

Before | After | Instead Of events

[referencing-variables]

[For Each Row]

When (condition)

action

Triggers in SQL - events

insert on T

delete on T

$\text{update } \underbrace{[C_1, \ldots, C_n]}_{\text{columns are optional}} \text{ on T}$

```
Create Trigger name
Before | After | Instead Of events
[ referencing-variables ]
[ For Each Row ]
When ( condition )
action
```

Triggers in SQL – [For Each Row]

```
Create Trigger name
Before | After | Instead Of events
[ referencing-variables ]
[ For Each Row ]
When ( condition )
action
```

Optional clause

States that the trigger is activated once for each modified tuple

If, for example, a delete command deletes 10 tuples

- With “for each row” -> trigger will run 10 times, once for each deleted tuple
- Without “for each row” -> trigger runs once for the entire statement

The trigger is always activated at the end of the statement

- Either x times or once if “for each row” is not present

Triggers in SQL – Referencing variables

To reference the data that was modified and caused the trigger to be activated

old row as var

new row as var

old table as var

new table as var

New variables for inserts and updates

Old variables for deletes and updates

```
Create Trigger name
Before | After | Instead Of events
[ referencing-variables ]
[ For Each Row ]
When ( condition )
action
```

Triggers in SQL – Referencing variables

```
Create Trigger name
Before | After | Instead Of events
[ referencing-variables ]
[ For Each Row ]
When ( condition )
action
```

old row as var / new row as var
- only for row-level triggers
- refer to the specific affected tuple

old table as var / new table as var
- for row-level or statement-level triggers
- refer to the set of all the affected tuples

What referencing variables can we use in these cases?
- Row-level delete
- Statement-level insert
- Row-level update

Triggers in SQL – condition and action

Condition

- Tests the condition, if it is true, action will be performed
- Like a general assertion

```
Create Trigger name
Before | After | Instead Of events
[ referencing-variables ]
[ For Each Row ]
When ( condition )
action
```

Action

- SQL statement
- Systems vary much here
  - set of simple statements + begin-end bracket
  - stored procedures
  - set of simple statements + begin/end

Referential integrity – row-level trigger

R.A references S.B, cascaded delete

```
Create Trigger Cascade
After Delete On S
Referencing Old Row As O
For Each Row
[ no condition ]
Delete From R Where A = O.B
```

Referential integrity – statement-level trigger

R.A references S.B, cascaded delete

Not the old state of the table, just the values of the tuples that have been deleted

Create Trigger Cascade

After Delete On S

Referencing Old Row As O $\longrightarrow$ Referencing $\boxed{\text{Old Table}}$ As OT

[For Each Row] $\longrightarrow$ Eliminate [For Each Row]

[no condition]

Delete From R Where A = O.B $\longrightarrow$ Delete From R Where A in (select B from OT)

Row-level versus statement-level triggers

```
Create Trigger Cascade
After Delete On S
Referencing Old Row As O
For Each Row
[ no condition ]
Delete From R Where A = O.B
```

```
Create Trigger Cascade
After Delete On S
Referencing Old Table As OT
[ For Each Row ]
[ no condition ]
Delete From R Where A in (select B from OT)
```

Which version to use?

- In this case, probably the statement-level trigger would be more efficient
- Some systems don't support both types of triggers -> no choice

Tricky issues

Row-level vs. Statement-level

- New/Old Row and New/Old Table
- Before, Instead Of

Multiple triggers activated at same time $\longrightarrow$ Which goes first?

Trigger actions activating other triggers (chaining)

- Also self-triggering, cycles, nested invocations

Conditions in When vs. as part of action

Implementations vary significantly

Row-level versus statement-level triggers

T(K,V) – K key, V value

```
Create Trigger IncreaseInserts
After Insert On T
Referencing New Row As NR, New Table As NT
For Each Row
When (Select Avg(V) From T) < (Select Avg(V) From NT)
Update T set V=V+10 where K=NR.K
```

stable value, NT is always the set of inserted tuples

No statement-level equivalent

Nondeterministic final state

Agenda

~~Introduction~~

Examples

- Before and After
- Insert, Delete, and Update
- New and Old
- Conditions and actions

Triggers enforcing constraints

SQL standard

Previous slides used SQL standard

No DBMS implements exact standard

- Some deviate considerably in syntax and behavior

SQL standard

Postgres

- Expressiveness/behavior = full standard row-level + statement-level, old/new row & table
- Cumbersome & awkward syntax

SQLite

- Row-level only, immediate activation -> no old/new table

MySQL

- Row-level only, immediate activation -> no old/new table
- Only one trigger per event type
- Limited trigger chaining

Triggers in SQLite

Row-level triggers, immediate activation

For Each Row implicit if not specified

No Old Table or New Table

No Referencing clause

- Old and New predefined for Old Row and New Row

Trigger action: SQL statements in begin-end block

College Admission Database

Apply

sID	cName	major	dec
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	Y
876	MIT	biology	Y
876	MIT	marine biology	N
765	Stanford	history	Y
765	Cornell	history	N
765	Cornell	psychology	Y
543	MIT	CS	N

College

cName	state	enr
Stanford	CA	15000
Berkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000

Student

sID	sName	GPA	sizeHS
123	Amy	3.9	1000
234	Bob	3.6	1500
345	Craig	3.5	500
456	Doris	3.9	1000
567	Edward	2.9	2000
678	Fay	3.8	200
789	Gary	3.4	800
987	Helen	3.7	800
876	Irene	3.9	400
765	Jay	2.9	1500
654	Amy	3.9	1000
543	Craig	3.4	2000

College(<u>cName</u>, state, enr)

Student(<u>sID</u>, sName, GPA, sizeHS)

Apply(<u>sID</u>, <u>cName</u>, <u>major</u>, decision)

Trigger 1 – after insert

```
Create Trigger R1
After Insert On Student
For Each Row
When New.GPA > 3.3 AND New.GPA <= 3.6
Begin
    Insert into Apply values (New.sID, ‘Stanford’, ‘geology’, null);
    Insert into Apply values (New.sID, ‘MIT’, ‘biology’, null);
End;
```

A first test to trigger 1

College(<u>cName</u>, state, enr)

Student(<u>sID</u>, sName, GPA, sizeHS)

Apply(<u>sID</u>, <u>cName</u>, <u>major</u>, decision)

Insert into Student values ('111', 'Kevin', 3.5, 1000);

Insert into Student values ('222', 'Lori', 3.8, 1000);

Student

sID	sName	GPA	sizeHS
…	…	…	…
765	Jay	2.9	1500
654	Amy	3.9	1000
543	Craig	3.4	2000
111	Kevin	3.5	1000
222	Lori	3.8	1000

Apply

sID	cName	major	dec
…	…	…	…
765	Cornell	history	N
765	Cornell	psychology	Y
543	MIT	CS	N
111	Stanford	geology	NULL
111	MIT	biology	NULL

College(<u>cName</u>, state, enr)

Student(<u>sID</u>, sName, GPA, sizeHS)

Apply(<u>sID</u>, <u>cName</u>, <u>major</u>, decision)

A second test to trigger 1

```
Insert into Student
    select sID+1, sName, GPA, sizeHS from Student;
```

Student

<u>sID</u>	**sName**	**GPA**	**sizeHS**
…	…	…	…
111	Kevin	3.5	1000
222	Lori	3.8	1000
…	…	…	…
766	Jay	2.9	1500
655	Amy	3.9	1000
544	Craig	3.4	2000
112	Kevin	3.5	1000
223	Lori	3.8	1000

Apply

<u>sID</u>	**<u>cName</u>**	**<u>major</u>**	**dec**
…	…	…	…
111	Stanford	geology	NULL
111	MIT	biology	NULL
…	…	…	…
112	Stanford	geology	NULL
112	MIT	biology	NULL

Trigger 2 – after delete

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

```
Create Trigger R2
After Delete On Student
For Each Row
Begin
    Delete from Apply where sID = Old.sID;
End;
```

What does it do?

A test to trigger 2

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

Delete from Student where sID > 500;

Student

sID	sName	GPA	sizeHS
…	…	…	…
111	Kevin	3.5	1000
222	Lori	3.8	1000
…	…	…	…
457	Doris	3.9	1000
112	Kevin	3.5	1000
223	Lori	3.8	1000

12 tuples

Apply

sID	cName	major	dec
…	…	…	…
111	Stanford	geology	NULL
111	MIT	biology	NULL
…	…	…	…
112	Stanford	geology	NULL
112	MIT	biology	NULL

17 tuples

Trigger 3 – after update

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

```
Create Trigger R3
After Update Of cName on College
For Each Row
Begin
    Update Apply
    Set cName = New.cName
    Where cName = Old.cName;
End;
```

$\longrightarrow$ If we left out cName then any update to College would activate this trigger

What does it do?

College(<u>cName</u>, state, enr)

Student(<u>sID</u>, sName, GPA, sizeHS)

Apply(<u>sID</u>, <u>cName</u>, <u>major</u>, decision)

A test to trigger 3

Update College set cName = ‘The Farm’ where cName = ‘Stanford’;

Update College set cName = ‘Bezerkeley’ where cName = ‘Berkeley’;

College

cName	state	enr
The Farm	CA	15000
Bezerkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000

Apply

sID	cName	major	dec
123	The Farm	CS	Y
123	The Farm	EE	N
123	Bezerkeley	CS	Y
123	Cornell	EE	Y
234	Bezerkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	The Farm	history	Y
987	The Farm	CS	Y
987	Bezerkeley	CS	Y

Trigger 4 – Simulating key constraints

College(<u>cName</u>, state, enr)

Student(<u>sID</u>, sName, GPA, sizeHS)

Apply(<u>sID</u>, <u>cName</u>, <u>major</u>, decision)

```
Create Trigger R4
Before Insert on College
For Each Row
When exists (select * from College where cName = New.cName)
Begin
    Select raise(ignore);
End;
```

$\longrightarrow$ Ignores the operation that's underway

What does it do?

Trigger 5 – Simulating key constraints

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

```
Create Trigger R4
Before Update of cName on College
For Each Row
When exists (select * from College where cName = New.cName)
Begin
    Select raise(ignore);
End;
```

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A test to trigger 4

```
Insert into College values ('Stanford', 'CA', 15000);

Insert into College values ('MIT', 'hello', 10000);
```

What should happen?

College

cName	**state**	**enr**
The Farm	CA	15000
Bezerkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000

$\longrightarrow$

College

cName	**state**	**enr**
The Farm	CA	15000
Bezerkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000
Stanford	CA	15000

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A test to trigger 5

Update College set cName = ‘Berkeley’ where cName = ‘Bezerkeley’;

Update College set cName = ‘Stanford’ where cName = ‘The Farm’;

Update College set cName = ‘Standford’ where cName = ‘The Farm’;

What should happen?

College

cName	state	enr
The Farm	CA	15000
Bezerkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000
Stanford	CA	15000

$\longrightarrow$

College

cName	state	enr
Standford	CA	15000
Berkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000
Stanford	CA	15000

A test to trigger 5

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

Anything happened behind the scenes?

Apply

sID	cName	major	dec
123	The Farm	CS	Y
123	The Farm	EE	N
123	Bezerkeley	CS	Y
123	Cornell	EE	Y
234	Bezerkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	The Farm	history	Y
987	The Farm	CS	Y
987	Bezerkeley	CS	Y

$\xrightarrow{\text{trigger 3}}$

Apply

sID	cName	major	dec
123	Standford	CS	Y
123	Standford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Standford	history	Y
987	Standford	CS	Y
987	Berkeley	CS	Y

Kahoot time!

Any doubts?

Readings

Jeffrey Ullman, Jennifer Widom, A first course in Database Systems 3rd Edition

Section 7.5 – Triggers