

# TBE: Writing Trigger Rules Visually

Wesley W. Chu  
Dept. of Computer Science  
UCLA

## Introduction

- Triggers:
  - A useful feature for push technology (e.g., active DB, sensors).
  - Difficult to understand and compose trigger rules.
- QBE (Query-By-Example)
  - Visual query interface.
  - Guide users to write only *admissible* SQL queries in an intuitive and visual manner.
- TBE (Trigger-By-Example)
  - Use QBE idea in writing trigger rules.
  - Based on SQL3 specification.

## QBE (Query-By-Example)

- SQL query is represented within two-dimensional skeleton tables by filling examples of the answers.
  - Variables names are lowercase alphabets with prefix “\_”.
  - System commands are uppercase alphabets with suffix “.”.
  - Constants are denoted without quote (unlike SQL3).
- Example Schema:
  - emp and dept relations.
  - key attributes: Eno and Dno (underlined).
  - foreign key attributes: emp.*DeptNo* references to dept.Dno and dept.*MgrNo* references to emp.Eno.

```
emp   (Eno, Ename, DeptNo, Sal)
dept  (Dno, Dname, MgrNo)
```

VDB5

3

## QBE Example: “Who is being managed by the manager Tom?”

```
SELECT E2.Ename
FROM   emp E1, emp E2, dept D
WHERE  E1.Ename = 'Tom' AND E1.Eno = D.MgrNo
AND    E2.DeptNo = D.Dno
```

emp	Eno	Ename	DeptNo	Sal
	<u>_e</u>	Tom		
		P.	<u>_d</u>	

→ E1  
→ E2

print

dept	Dno	Dname	MgrNo
	<u>_d</u>		<u>_e</u>

→ D

VDB5

4

## TBE Model

- TBE = ECA rules + QBE
- TBE has 3 distinct skeleton tables and condition boxes.
- Each E, C, A rule in trigger rule maps to the corresponding skeleton table with the same prefix **E.**, **C.**, **A.**, respectively.
- INSERT, DELETE, UPDATE are denoted by **I.**, **D.**, **U.** system commands.
- Since **I.** and **D.** affects the whole tuple, they must be filled in the table name column (i.e., leftmost) of the skeleton tables.

VDB5

5

## Event Skeleton Table Examples

(1)				(2)			
E.dept	Dno	Dname	MgrNo	E.dept	Dno	Dname	MgrNo
I.				D.			

  

(3)				(4)			
E.dept	Dno	Dname	MgrNo	E.dept	Dno	Dname	MgrNo
		U.	U.	U.			

- (1) & (2): INSERT and DELETE events on dept table.
- (3): UPDATE event of columns Dname and MgrNo.
- (4): UPDATE event of any columns on dept table.

VDB5

6



## Activation Time & Granularity

- SQL3 trigger has two activation time modes
  - BEFORE: triggers execute before their events. (BFR.)
  - AFTER: triggers execute after their events. (AFT.)
- SQL3 trigger has two granularities
  - Row-level: triggers are executed once for each modification to tuple. (R.)
  - Statement-level: triggers are executed once for an event regardless of the number of tuples affected. (S.)



## Transition Values

- When an event occurs and values change, trigger rules need to refer to the *before* or *after* values (i.e., transition values) of the triggered attributes.
- SQL3
  - Row-level: OLD and NEW.
  - Statement-level: OLD\_TABLE and NEW\_TABLE.
- TBE provides equivalent built-in functions
  - Row-level: OLD() and NEW().
  - Statement-level: OLD\_TABLE() and NEW\_TABLE().
    - OLD\_TABLE() returns a set of tuples with values *before* the changes.
    - NEW() returns a single tuple with value *after* the change.

## TBE Event Example:

“Every time more than 10 employees are inserted (statement-level)”

E.emp	Eno	Ename	DeptNo	Sal
AFT.I.S.	_n			

E.conditions
CNT.ALL.NEW_TABLE(_n) > 10

- The rule is activated *after* activation time (AFT.), after *insertion* event (I.), for each *statement* (S.).
- Use a built-in function CNT. to count the number of employee tuples inserted.
- ALL. keeps duplicates in counting.
- Two skeleton tables are linked by the same variable \_n.

## TBE Statement Box

- SQL3 trigger allows arbitrary SQL procedural statements (e.g., IF, CASE, assignment statements) in the action part of the rules.
- TBE uses a special box similar to QBE condition box denoted as *statement box* with A. prefix.
  - Fill in arbitrary SQL statements delimited by “;”
  - Fill in action part of the trigger rules.

## TBE Simple Example:

“When a manager is deleted, all employees in his dept are deleted too.”

```
CREATE TRIGGER ManagerDelRule AFTER DELETE On emp
FOR EACH ROW
DELETE FROM emp E WHERE E.DeptNo IN
(SELECT D.Dno FROM dept D WHERE D.MgrNo = OLD.Eno)
```

E.emp	Eno	Ename	DeptNo	Sal
AFT.D.R.	_e			

A.dept	Dno	Dname	MgrNo
	_d		_e

A.emp	Eno	Ename	DeptNo	Sal
D.			_d	

VDB5

11

The screenshot shows the TBE IDE interface for configuring a trigger rule. The window title is "TBE" and the menu bar includes "File", "Edit", "View", "Insert", and "Help". The main area is divided into three sections:

- Input:** Contains two panels:
  - Event panel:** Describes what causes the rule to be triggered.
  - Condition panel:** Describes a condition to be checked when the rule is triggered.
- Action:** Contains one panel:
  - Action panel:** Describes what to do when the rule is triggered and condition is satisfied.
- Output:** Contains one panel:
  - Output panel:** Contains the trigger rule generated by the above E, C, A parts.

The status bar at the bottom shows "Status:".

VDB5

12

## TBE Construction Process Example

- When an employee's salary is changed more than twice within the same year, record new values of Eno and Sal into the **log(Eno, Sal)** table. There is another table **sal-change(Eno, Year, Cnt)** that keeps track of the employee's salary changes.

```
CREATE TRIGGER TwiceSalaryRule AFTER UPDATE OF Sal ON emp
FOR EACH ROW
WHEN EXISTS (SELECT * FROM sal-change WHERE
    Eno = NEW.Eno AND Year = CURRENT_YEAR AND Cnt >= 2)
BEGIN ATOMIC
    UPDATE sal-change SET Cnt = Cnt + 1
    WHERE Eno = NEW.Eno AND Year = CURRENT_YEAR;
    INSERT INTO log VALUES (NEW.Eno, NEW.Sal);
END
```

VDB5

13

Context-sensitive pop-up menu.

E.emp	Eno	Ename	DeptNo	Sal
AFT.R.	_n			U.

Insert Row  
Delete Row  
Insert Example Variable ▶ new one  
Insert Transition Variable ▶ \_c  
Insert Aggregation ▶ \_n  
Insert System Command ▶

Writing condition.

E.emp	Eno	Ename	DeptNo	Sal
AFT.R.	_n			U.

C.sal-change	Eno	Year	Cnt	C.conditions
	_n	CURRENT_YEAR	_c	_c >= 2

VDB5

14

The screenshot shows a database tool window titled 'TwiceSalaryRule' with a 'Target' of 'SQL3'. It displays a visual representation of a trigger rule with three tables: 'E.emp', 'C.sal-change', and 'A.sal-change'. The 'E.emp' table has columns 'EmpNo', 'Eno', 'Ename', 'DeptNo', and 'Sal'. The 'C.sal-change' table has columns 'Eno', 'Year', 'Cnt', and 'C.conditions'. The 'A.sal-change' table has columns 'Eno', 'Year', 'Cnt', 'A.log', 'Eno', and 'Sal'. Below the visual representation is a text editor containing the following SQL code:

```

CREATE TRIGGER TwiceSalaryRule
AFTER UPDATE ON E.emp
FOR EACH ROW
WHEN EXISTS (
  SELECT *
  FROM C.sal-change
  WHERE C.sal-change.Cnt >= 2
  AND C.sal-change.Eno = E.Eno
  AND C.sal-change.Year = CURRENT_YEAR)
BEGIN ATOMIC
  UPDATE A.sal-change
  SET A.sal-change.Cnt = A.sal-change.Cnt + 1
  WHERE A.sal-change.Eno = E.Eno
  AND A.sal-change.Year = CURRENT_YEAR;
  INSERT INTO A.log VALUES (E.Eno, NEW.Sal);
END

```

The status bar at the bottom indicates 'Status: Done'.

## Summary

- TBE:
  - Easy in writing trigger rules.
    - Visual
    - Admit only valid input
  - Support SQL3 triggers.
  - Support Oracle, Sybase triggers by universal trigger mapping.
  - Statement box to support arbitrary action statements.
- Future work
  - Support for composite event triggers.
  - Support for interactions among multiple triggers.
- <http://www.cobase.cs.ucla.edu/projects/tbe/>