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.MgrName references to emp.Eno.

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

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
```
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.

Event Skeleton Table Examples

(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.
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)”

<table>
<thead>
<tr>
<th>E.emp</th>
<th>Eno</th>
<th>Ename</th>
<th>DeptNo</th>
<th>Sal</th>
<th>_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFT.I.S.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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._*

**E.conditions**

| CNT.ALL.NEW_TABLE(_n) > 10 |

**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.”

```sql
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)
```

---

**Event panel describes what causes the rule to be triggered.**

**Condition panel describes a condition to be checked when the rule is triggered.**

**Action panel describes what to do when the rule is triggered and condition is satisfied.**

**Output panel contains the trigger rule generated by the above E, C, A parts.**
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
```

Context-sensitive pop-up menu.

<table>
<thead>
<tr>
<th>Emp</th>
<th>Eno</th>
<th>Ename</th>
<th>DeptNo</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFT.R</td>
<td>_n</td>
<td>Name_</td>
<td>Dept_</td>
<td>U.</td>
</tr>
</tbody>
</table>

- Insert Row
- Delete Row
- Insert Example Variable: new one
- Insert Transition Variable: _c
- Insert Aggregation: _n
- Insert System Command

Writing condition.

<table>
<thead>
<tr>
<th>Emp</th>
<th>Eno</th>
<th>Ename</th>
<th>DeptNo</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFT.R</td>
<td>_n</td>
<td>Name_</td>
<td>Dept_</td>
<td>U.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C sal-change</th>
<th>Eno</th>
<th>Year</th>
<th>Cnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>_n</td>
<td>CURRENT_YEAR</td>
<td>_c</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C conditions</th>
<th>_c == 2</th>
</tr>
</thead>
</table>
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.