Triggers

Tom Kelliher, CS 318

Apr. 5, 2002

1 Administrivia

Announcements

Assignment

Read Chapter 10. Refer to the PostgreSQL online documentation for triggers and stored procedures Chapter 24 of the *Programmer's Guide*. Project implications: Your integrity constraints and FDs should be maintained via triggers and PL/pgSQL stored procedures.

From Last Time

Normal forms and schema synthesis/decomposition.

Outline

1. Trigger concepts.
2. PostgreSQL triggers.
3. Controlling cascading triggers.

Coming Up

Stored procedures in PostgreSQL.

2 Trigger Concepts

1. A trigger is an element of a database schema with the following form:

   ```
   ON Event
      IF PreCondition THEN
         Action
   ```

   where:

   (a) Event refers to a database modification operation.

   (b) PreCondition must be true for the trigger to "fire."

   (c) Action is a list of steps to take if the trigger fires.

2. Informal example:

   ```
   ON Insertion on Transcript
      IF Course is Full THEN
         Abort Transaction
   ```

3. Trigger consideration

   (a) When is the PreCondition evaluated, relative to the event?

   What is an event? A single SQL statement or an entire transaction?

   (b) Options: Immediate or deferred consideration.

4. Trigger execution

   (a) When is the Action executed, relative to the event.

(b) Possibilities: immediate or deferred.

Obviously, immediate execution can't be paired with deferred consideration.

(c) Action options with immediate execution:

i. Before event.

ii. In place of event.

iii. After event.

5. Trigger granularity

(a) Is the trigger associated with the occurrence of the event or each tuple affected by the event?

(b) Statement-level triggers:

i. Trigger occurs once per event.

ii. Occurs even if nothing is modified.

iii. Good for computing aggregate data.

iv. Trigger is passed "before" and "after" images of the affected tuples as two relations.

(c) Row-level triggers:

i. Trigger occurs multiple times per event.

ii. Won't occur if nothing is modified.

iii. Good for checking each affected tuple.

iv. Trigger is passed "before" and "after" images of the affected tuples.

6. Multiple enabled triggers

(a) Ordering of consideration?

(b) Concurrent or consecutive evaluation/execution.

7. Triggers and integrity constraints

   (a) How do triggers interact with `ON DELETE` and `ON CASCADE` clauses of `UPDATE` and `DELETE` statements and foreign constraints.

3 PostgreSQL Triggers

1. `CREATE TRIGGER` syntax in PostgreSQL:

```
CREATE TRIGGER name { BEFORE | AFTER } { event [ OR ... ] }
   ON table FOR EACH { ROW | STATEMENT }
   EXECUTE PROCEDURE func ( arguments );
```

   `EVENT` is one of `INSERT`, `DELETE`, or `UPDATE`. `STATEMENT` level triggers are *not* currently supported.

   Of course, the PL/pgSQL `func` must appear before the trigger naming it.

2. Example:

```
CREATE FUNCTION RaiseCheck () RETURNS OPAQUE AS '
   BEGIN
      IF NEW.Salary > 1.05 * OLD.Salary THEN
         -- Excessive salary raise - abort transaction.
         RAISE EXCEPTION ''% given an excessive raise'', OLD.EmpName;
      END IF;
      -- Salary increase OK, proceed with transaction.
      RETURN NEW;
   END;
' LANGUAGE 'plpgsql';

CREATE TRIGGER LimitRaises BEFORE UPDATE ON Employee
   FOR EACH ROW EXECUTE PROCEDURE RaiseCheck();
```

3. Introduction to the anatomy of a PL/pgSQL trigger function.

4 Controlling Cascading Triggers

1. "Safe" triggers.

2. Dependency graphs of triggers.

3. Eliminating cycles in the dependency graph.

4. Some cycles are safe. Self-limiting triggers: LimitSalaryRaise.