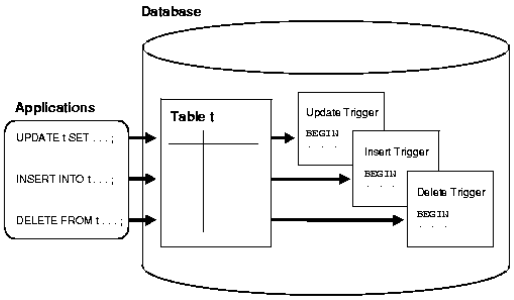
Chapter 11:
Database Triggers

- Triggers are procedures written in PL/SQL, Java, or C that run (fire) implicitly whenever a table or view is modified or when some user actions or database system actions occur.
- Are similar to a stored procedure.

- Program units that are attached to a specific table or view
- Execute in response to the following table operations:
  - INSERT
  - UPDATE
  - DELETE



Uses For Database Triggers

- Automatically generate derived column values
- Prevent invalid transactions
- Enforce complex security authorizations
- Enforce referential integrity across nodes in a distributed database
- Enforce complex business rules
- Provide transparent event logging
- Provide auditing
- Maintain synchronous table replicates
- Gather statistics on table access
- Modify table data when DML statements are issued against views
- Publish information about database events, user events, and SQL statements to subscribing applications

Creating Database Triggers

- Code is similar to all PL/SQL program unit blocks
- Database triggers cannot accept parameters

Defining Triggers

- To define a trigger, you must specify:
  - Statement type that causes trigger to fire
    - INSERT, UPDATE, DELETE
  - Timing
    - BEFORE or AFTER
  - Level
    - STATEMENT or ROW

Trigger Timing

- **BEFORE**: trigger fires before statement executes
  - Example: for audit trail, records grade value before it is updated
- **AFTER**: trigger fires after statement executes
  - Example: update QOH after item is sold

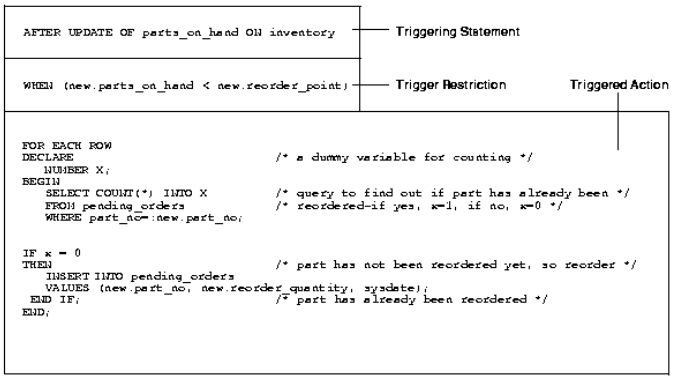
Trigger Levels

- **ROW**: trigger fires once for each row that is affected
  - Example: when updating multiple item prices, inventory value has to be updated for each item.
- **STATEMENT**: trigger fires once, regardless of how many rows are updated
  - Example: for audit trail, you just want to record that someone updated a table, but you don't care how many rows were updated

Parts of a Trigger

- A trigger has three basic parts:
  - A triggering event or statement
  - A trigger restriction
  - A trigger action

Parts of a Trigger: The REORDER Trigger



Creating a Trigger in SQL*Plus

CREATE OR REPLACE TRIGGER *trigger_name*

[BEFORE|AFTER] [INSERT|UPDATE|DELETE] ON *table_name*
[FOR EACH ROW]

[WHEN (condition)]

BEGIN
trigger body
END;

Triggering Event or Statement

- A triggering event or statement is the SQL statement, database event, or user event that causes a trigger to fire. A triggering event can be one or more of the following:
  - An INSERT, UPDATE, or DELETE statement on a specific table (or view, in some cases)
  - A CREATE, ALTER, or DROP statement on any schema object
  - A database startup or instance shutdown
  - A specific error message or any error message
  - A user logon or logoff

- For example:
- ... UPDATE OF parts_on_hand ON inventory ...

- This statement means that when the parts_on_hand column of a row in the inventory table is updated, fire the trigger.
- When the triggering event is an UPDATE statement, you can include a column list to identify which columns must be updated to fire the trigger.
- You cannot specify a column list for INSERT and DELETE statements, because they affect entire rows of information.

- A triggering event can specify multiple SQL statements:
- ... INSERT OR UPDATE OR DELETE OF inventory ...

- This part means that when an INSERT, UPDATE, or DELETE statement is issued against the inventory table, fire the trigger.

Trigger Restriction

- A trigger restriction specifies a Boolean expression that must be true for the trigger to fire.
- The trigger action is not run if the trigger restriction evaluates to false or unknown.
- For example:

  :new.parts_on_hand < :new.reorder_point

- The trigger does not fire unless the number of available parts is less than a present reorder amount.

Row-Level Trigger Syntax

- WHEN (condition):
  - Optional
  - Specifies to fire only when a row satisfies a certain search condition
- Referencing old and new values in the trigger body:
  - :OLD.*field_name*
  - :NEW.*field_name*

Trigger Action

- A trigger action is the procedure (PL/SQL block, Java program, or C callout) that contains the SQL statements and code to be run when the following events occur:
  - A triggering statement is issued.
  - The trigger restriction evaluates to true.

Creating a Row-Level Trigger in SQL*Plus

```
CREATE OR REPLACE TRIGGER qoh_update
  AFTER UPDATE ON order_line
  FOR EACH ROW
BEGIN
  UPDATE inventory SET qoh = qoh + :OLD.order_quantity -
:NEW.order_quantity;
END;
/
```

Trigger Restrictions

- You can only create triggers on tables that you own
- You must have the CREATE TRIGGER system privilege
- You cannot execute a COMMIT command in a trigger

Disabling and Dropping Triggers

- Syntax to drop a trigger:
  DROP TRIGGER *trigger_name*;

- Syntax to enable or disable a trigger:
  ALTER TRIGGER *trigger_name* [ENABLE | DISABLE];

Trigger Problems

- Potentially infinite loop
  - Trigger A: On insertion into Person, insert into Population
  - Trigger B: On insertion into Population, insert into Person
- Mutating tables
  - Trigger A: On insertion into Person, insert into Person!
  - Disallowed!
  - Trigger cannot make changes to table that trigger is defined on

Cautionary Notes

- Although triggers are useful for customizing a database, use them only when necessary.
- Excessive use of triggers can result in complex interdependencies, which can be difficult to maintain in a large application.
  - For example, when a trigger fires, a SQL statement within its trigger action potentially can fire other triggers, resulting in **cascading triggers**.

Summary

- Examined uses of triggers
- Trigger timing
  - BEFORE
  - AFTER
- Trigger level
  - ROW
  - STATEMENT
- Parts of a trigger
  - Event or statement
  - Restriction
  - Action
- Creating and dropping database triggers
- Trigger problems