Writing
Database Triggers

Michael McLaughlin
Faculty, BYU - Idaho

Outline

- Why Triggers?
- System & Database Triggers?
- DDL Triggers
- DML Triggers
- Compound Triggers
- Instead Of Triggers
- Trigger Restrictions
- Query Pseudo Triggers

Why Triggers?

- Implement Event Driven Handlers
- Implement Business Logic Guarantees
- Hide Business Process & Logic

Event-Driven Handler

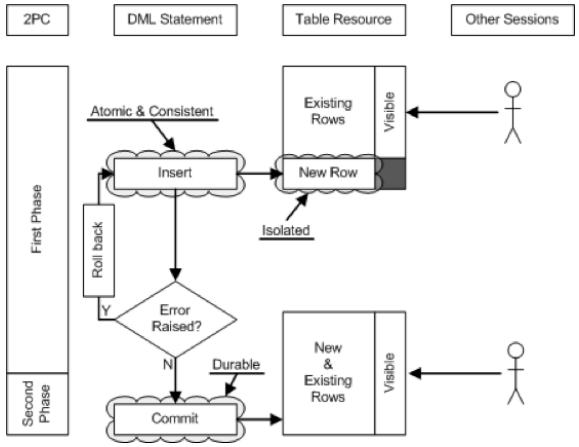
- Triggers listen for events
- Sources for the events
  - System logins
  - Server errors
  - Startups and shutdowns
  - DDL commands
  - DML commands

Business Logic Guarantees

- Manage Critical Business Events
  - Events fire the trigger
  - Stored logic manages event handling process
  - Raises an exception and stops the event
- Manage Non-critical Business Events
  - Events fire trigger
  - Stored logic manages event handling process
  - Log event and allow event to continue successfully

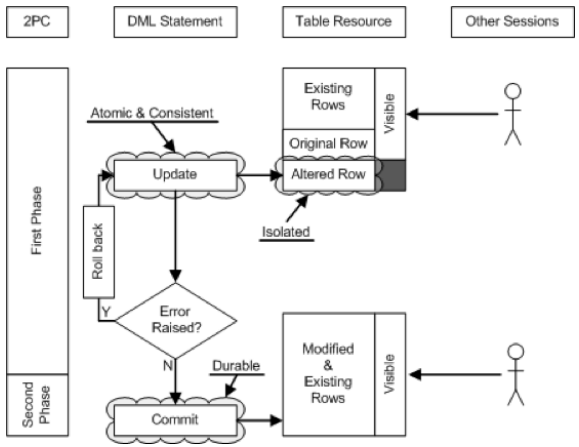
Hide Business Process & Logic

`INSERT` - 2 Phase Commit (PC) Cycle



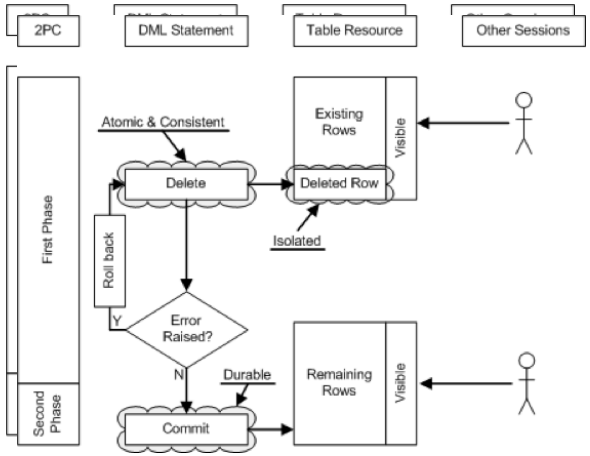
Hide Business Process & Logic

`UPDATE` - 2 Phase Commit (PC) Cycle



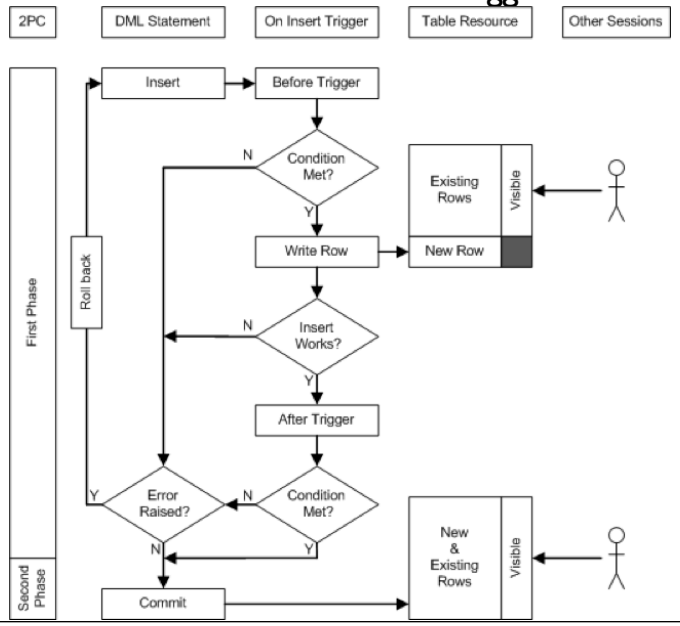
Hide Business Process & Logic

`DELETE` - 2 Phase Commit (PC) Cycle



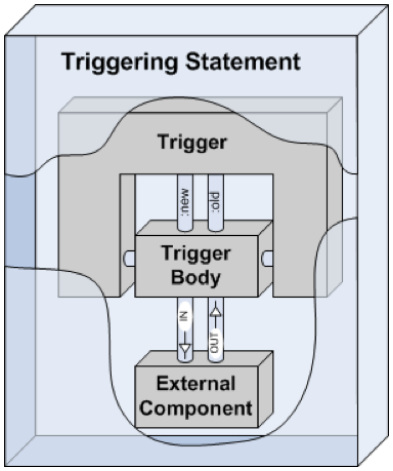
Hide Business Process & Logic

`INSERT` - 2 PC with Trigger



Hide Business Process & Logic

Oracle Trigger Architecture



System & Database Triggers

- Triggers listen for system or database events
- Sources for the events
  - System logins
  - Server errors
  - Startups and shutdowns

Login Trigger
Call an Obfuscated Package

```
CREATE OR REPLACE TRIGGER connecting_trigger
  AFTER LOGON ON DATABASE
BEGIN
  user_connection.connecting(ora_login_user);
END;
/

CREATE OR REPLACE TRIGGER disconnecting_trigger
  BEFORE LOGOFF ON DATABASE
BEGIN
  user_connection.disconnecting(ora_login_user);
END;
/
```

Login & Logout Triggers
Package Specification

```
CREATE OR REPLACE PACKAGE user_connection AS

  PROCEDURE Connecting
  (user_name       IN VARCHAR2);

  PROCEDURE Disconnecting
  (user_name       IN VARCHAR2);

END user_connection;
/
```

Login & Logout Triggers
Package Body

```
CREATE OR REPLACE PACKAGE BODY user_connection AS
  PROCEDURE connecting
  (user_name       IN VARCHAR2) IS
  BEGIN
    INSERT INTO connection_log
    (event_user_name, event_type, event_date)
    VALUES (user_name,'CONNECT',SYSDATE);
  END connecting;
  PROCEDURE disconnecting
  (user_name       IN VARCHAR2) IS
  BEGIN
    INSERT INTO connection_log
    (event_user_name, event_type, event_date)
    VALUES (user_name,'DISCONNECT',SYSDATE);
  END disconnecting;
END user_connection;
/
```

Login & Logout Triggers
Package Body

```
CREATE OR REPLACE PACKAGE user_connection AS

  PROCEDURE Connecting
  (user_name       IN VARCHAR2);

  PROCEDURE Disconnecting
  (user_name       IN VARCHAR2);

END user_connection;
/
```

DDL Triggers
Available DDL Events

- Triggers listen for DDL events
- Sources for the events
  - The **`ALTER`**, **`CREATE`**, **`DROP`**, **`RENAME`**, **`TRUNCATE`**, and **`COMMENT`** Statements
  - The **`GRANT`** and **`REVOKE`** Statements
  - The **`ANALYZE`**, **`AUDIT`**, **`ASSOCIATE STATISTICS`**, and **`DISASSOCIATE STATISTICS`** Statements
  - The **`AUDIT`**, and **`NOAUDIT`** Statements

DDL Triggers
Available DDL Attribute Functions

- Helpful tools - DDL Attribute Functions
  - Let you simply access to frequently requested features
- Sources for the events
  - ORA_CLIENT_IP_ADDRESS, ORA_DATABASE_NAME, ORA_DICT_OBJ_NAME, ORA_DICT_OBJ_NAME_LIST, ORA_DICT_OBJ_OWNER, ORA_DICT_OWNER_LIST, ORA_DICT_OBJ_TYPE, ORA_GRANTEE, ORA_INSTANCE_NAME, ORA_IS_ALTER_COLUMN, ORA_IS_CREATING_NESTED_TABLE, ORA_IS_DROP_COLUMN, ORA_IS_SERVERERROR, ORA_LOGIN_USER, ORA_PARTITION_POS, ORA_PRIVILEGE_LIST, ORA_REVOKEE, ORA_SERVER_ERROR, ORA_SERVER_ERROR_DEPTH, ORA_SERVER_ERROR_MSG, ORA_SERVER_ERROR_NUM_PARAMS, ORA_SERVER_ERROR_PARAM, ORA_SQL_TXT, ORA_SYSEVENT, ORA_WITH_GRANT_OPTION, SPACE_ERROR_INFO

DDL Triggers

Using DDL Attribute Functions

```
CREATE OR REPLACE TRIGGER audit_creation_t
BEFORE CREATE ON SCHEMA
BEGIN
  INSERT INTO audit_creation VALUES
  ( audit_creation_s.nextval
  , ORA_DICT_OBJ_OWNER
  , ORA_DICT_OBJ_NAME
  , SYSDATE );
END audit_creation_t;
/
```

DML Triggers
Types of Behavior

- Statement-level Triggers
  - Fire one-time for any `INSERT`, `UPDATE`, or `DELETE` statement
  - Can't access the values of the DML event
- Row-level Triggers `[FOR EACH ROW]`
  - Fire one-time for any row inserted, updated, or deleted
  - Can access the values of the DML event by row
  - Can accept, reject, or alter the value of any change
- Synchronizing triggers with the `FOLLOWS` clause

DML Triggers

Trigger Firing Order

- Before statement triggers
- Before row triggers
- After row triggers
- After statement triggers

DML Triggers
Type of Use

- Non-critical Business Logic Triggers
  - Discover events
  - Log events for subsequent action
- Critical Business Logic Triggers
  - Discover events
  - Log events through external autonomous stored programs
  - Raise errors to stop the event

DML Triggers

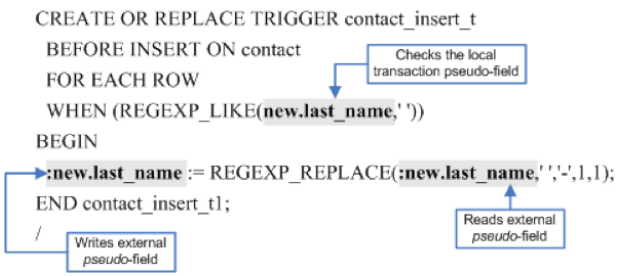
Non-Critical Business Logic Example

	:new	:old
`INSERT`	X	
`UPDATE`	X	X
`DELETE`		X

DML Triggers

Non-Critical Business Logic Example

```
CREATE OR REPLACE TRIGGER contact_insert_t
  BEFORE INSERT ON contact
  FOR EACH ROW
  WHEN (REGEXP_LIKE(new.last_name,' '))
BEGIN
:new.last_name := REGEXP_REPLACE(:new.last_name,' ','-',1,1);
END contact_insert_t1;
/
```



DML Triggers

Trigger Logic Sequencing - FOLLOWS

```
CREATE OR REPLACE TRIGGER member_insert_t1
BEFORE INSERT ON member
FOR EACH ROW
BEGIN
  INSERT INTO logger VALUES ('Who''s on first');
END member_insert_t1;
/

CREATE OR REPLACE TRIGGER member_insert_t2
BEFORE INSERT ON member
FOR EACH ROW
FOLLOWS member_insert_t1
BEGIN
  INSERT INTO logger VALUES ('What''s on second');
END member_insert_t2;
/
```

DML Triggers

Non-Critical Business Logic - INSERT

```
CREATE OR REPLACE TRIGGER contact_insert_t
BEFORE INSERT ON contact
FOR EACH ROW
BEGIN
    :new.last_name :=
      REGEXP_REPLACE(:new.last_name,' ','-',1,1);
END contact_insert_t;
/
```

DML Triggers

Non-Critical Business Logic - UPDATE

```
CREATE OR REPLACE TRIGGER contact_insert_t
BEFORE UPDATE ON contact
FOR EACH ROW
BEGIN
    :new.last_name :=
      REGEXP_REPLACE(:new.last_name,' ','-',1,1);
END contact_insert_t;
/
```

DML Triggers

Non-Critical Business Logic - INSERT/UPDATE

```
CREATE OR REPLACE TRIGGER contact_insert_t
BEFORE INSERT OR UPDATE ON contact
FOR EACH ROW
BEGIN
    :new.last_name :=
      REGEXP_REPLACE(:new.last_name,' ','-',1,1);
END contact_insert_t;
/
```

DML Triggers

Non-Critical Business Logic - Logging Statement

```
CREATE OR REPLACE TRIGGER price_t
AFTER UPDATE OF amount ON price
BEGIN
  INSERT INTO price_log
  VALUES
  ( price_log_s1.NEXTVAL
  ,'Change of amount value'
  , USER
  , SYSDATE);
END price_t;
/
```

DML Triggers

Non-Critical Business Logic - Logging Data Change

```
CREATE OR REPLACE TRIGGER price_t
AFTER UPDATE OF amount ON price
FOR EACH ROW
BEGIN
  INSERT INTO price_log
  VALUES
  ( price_log_s1.NEXTVAL  -- Not allowed before 11g
  , price_id
  , :old.amount
  , :new.amount
  , USER
  , SYSDATE);
END price_t;
/
```

DML Triggers

Critical Business Logic - Logging Data Change

```
CREATE OR REPLACE TRIGGER price_t
AFTER UPDATE OF amount ON price
FOR EACH ROW
BEGIN
  autonomous_logging( :old.price_id
                    , :old.amount
                    , :new.amount
                    , USER );
  RAISE_APPLICATION_ERROR(-20000,'Not enough data.');
END price_t;
/
```

Autonomous procedures enable you to send data to another context where it can be written when you terminate the current trigger's context.

DML Triggers
Compound Trigger

```
CREATE OR REPLACE TRIGGER compound_price_t
AFTER UPDATE ON price
COMPOUND TRIGGER
  declaration_statements;  -- Global data types and variables
BEGIN STATEMENT IS
BEGIN
  statements;              -- Processing statements
END BEFORE STATEMENT;
BEGIN EACH ROW IS
BEGIN
  statements;              -- Processing statements
END EACH ROW;
BEGIN STATEMENT IS
BEGIN
  statements;              -- Processing statements
END BEFORE STATEMENT;
END compound_price_t;
/
```

Instead Of Triggers

Triggers that work with Views

- Instead Of means
  - You use the Instead Of trigger to write non-updatable views
  - You embed logic in the trigger to unwind logic, like aggregation
- Instead Of functions
  - The `INSERTING` function identifies an `INSERT` statement
  - The `UPDATING` function identifies an `UPDATE` statement
  - The `DELETING` function identifies a `DELETE` statement

Instead Of Triggers

Triggers that work with Views

```
CREATE OR REPLACE TRIGGER customer_t
INSTEAD OF INSERT OR UPDATE OR DELETE ON customer_v
FOR EACH ROW
DECLARE
  declaration_statements;
BEGIN
  IF INSERTING THEN           -- Only on INSERT
    RAISE_APPLICATION_ERROR(-20000,'Not enough data.');
  ELSIF UPDATING THEN         -- Change new values on UPDATE.
    processing_statements;
  ELSIF DELETING THEN         -- Only on DELETE
    processing_statements;
  END IF;
END customer_t;
/
```

Query Pseudo Triggers

Functions that Support Discrete DML Statements

```
CREATE OR REPLACE FUNCTION object_table
(pv_search_key  VARCHAR2) RETURN object_table_view IS
  lv_counter       NUMBER := 1;
  lv_object_table  OBJECT_LIST := object_list();
  CURSOR c (cv_search_key) IS
    SELECT * FROM target_table
    WHERE  column_name = cv_search_key;
BEGIN
  FOR i IN c(pv_search_key) LOOP
    lv_object_table(lv_counter) := object_list(i.column_x
                                              ,i.column_y);
    lv_counter := lv_counter + 1;
  END LOOP;
  call_autonomous_routine;
  RETURN lv_object_table;
END;
/
```

A query against the function within a TABLE function call returns the UDT list, and dispatches information to an autonomous routine that writes data.

Query Pseudo Triggers

Functions that Support Discrete DML Statements

A query against the function within a TABLE function call returns the UDT list, and dispatches information to an autonomous routine that writes data.

```
SELECT *
FROM   TABLE(object_table('search_key'));
```

Review

- Why Triggers?
- System & Database Triggers?
- DDL Triggers
- DML Triggers
- Compound Triggers
- Instead Of Triggers
- Trigger Restrictions
- Query Pseudo Triggers

Questions?