iSQLplus:
http://uadoem02.uad.ac.uk:5560/isqlplus.

Oracle PL/SQL

TRIGGERS

What is a trigger?

- A block of code that is attached to an event. When that event occurs the trigger code is fired.
- A stored block with [Declare], Begin, End.
- Associated with a database table or action
- Fires **automatically** when certain DML action is carried out on the table
  - Before or after an event
  - Change may be INSERT, DELETE, UPDATE
  - Do we want to perform on multiple rows?
  - If not, then Statement level trigger fires once only
  - If so, then Row level trigger fires for each affected row

Trigger Uses

1. Auditing
   - Write information about (sensitive) data modifications to an audit table
   - May include old and new values, user, timestamp
   - E.g. new and old salary
2. Data Integrity
   - Implement checks on data against business rules
   - Can compare with live database values
   - NEW and OLD values can be compared
   - E.g. prices must not go down

Trigger Uses

3. Referential integrity
   - Allows implementation of a "cascade update"
   - E.g. if author ID (aID) is changed, appropriately change authID in foreign key
4. Derived data
   - Update any stored derived data when base data changes
   - E.g. if total number of employees is stored, add 1 if new employee added

Trigger Uses

5. Security
   - Logging of database access
   - E.g. date and time each user logs on
   - E.g. deny access at weekend
6. Maintaining synchronous replicates
   - In a distributed database
7. Generating statistics on table access

Types of Trigger and Naming

- Statement or Row triggers
- So 4 types can be triggered:
  - Before Statement, Before Row
  - After Statement, After Row
- Naming them
  - Must be unique within a schema
  - Can have same name as table on which it is defined but this may be confusing to developers

Statement Triggers

Inserting, Deleting and Updating can be used to find out which event is occurring

```
CREATE OR REPLACE TRIGGER trig_testTable
AFTER INSERT or UPDATE ON Personnel
BEGIN
If Inserting
    Then INSERT into testTable values ('insert done', SYSDATE) ;
Else
     INSERT into testTable values ('update done', SYSDATE) ;
End If;
END;
```

Test_trigger_1 tests this trigger

The statement level trigger is useful for INSERT as it only involves a single row whereas DELETE and UPDATE often deal with many rows at once

Row trigger

- Operates on many rows potentially

```
CREATE OR REPLACE TRIGGER trig_test
AFTER UPDATE OF SNUM ON PERSONNEL
FOR EACH ROW
BEGIN
    null;          -- write operations here
END;
```

A Row trigger fires once for every row affected by the DML operation

:OLD and :NEW

- When a DML statement changes a column the old and new values are visible to the executing code
- This is done by prefixing the table column with :old or :new
- **:new** is useful for INSERT and UPDATE
- **:old** is useful for DELETE and UPDATE
- triggers may fire other triggers in which case they are CASCADING. Try not to create too many interdependencies with triggers!

Book

ISBN	title	publisher	year	pages	price	authID
0201403803	Data Mining	Addison_Wesley	1996	158		1
0077095855	Database Design and Programming	McGraw-Hill	2000	487		3
0773481435	Essays on Fiction	Edwin Mellen Press	1999	156	£39.99	4
0201674769	Database Solutions	Longman Higher	1999	256	£29.99	5
0201342871	Database Systems	Addison_Wesley	1998	1094	£31.99	5
0130402648	Database System Implementation	Prentice-Hall	2000	653	£29.99	6
0201694719	Database Design for mere Mortals	Addison_Wesley	1997	440	£22.99	7
0760010900	Database Systems	Course Technology	1999	765	£28.99	5

Author

aID	last_name	forename
1	Adriaans	Pieter
3	Carter	John
4	Connolly	Thomas E
5	Connolly	Thomas M
6	Garcia-Molina	Hector
7	Hernandez	Michael J
8	Rob	Peter

Publisher

name	country
Addison_Wesley	UK
Course Technology	
Edwin Mellen Press	
Longman Higher	
McGraw-Hill	UK
Prentice-Hall	USA

Referential integrity trigger example

```
UPDATE author SET aID='9' WHERE aID='5';

ERROR at line 1:
ORA-02292: integrity constraint
   (MCTPL.BOOK_FK) violated - child record found
```

Without trigger –
referential integrity prevents
changes to aID

```
CREATE OR REPLACE TRIGGER author_trg
AFTER UPDATE OF aID ON author
FOR EACH ROW
BEGIN
UPDATE Book SET authID = :new.aID WHERE authID= :old.aID;
END;
```

Trigger
automatically applies
corresponding changes
to aID in child table

```
UPDATE author SET aID='9' WHERE aID='5';
1 row updated.
```

With trigger –
changes to aID are now allowed!

Referring to Values which fire the trigger

- So remember ………….
- While trigger is running, it knows the old <u>and</u> the new values of the record you're updating
- You need to specify which you want to refer to
- Prefix column name with **:new.** or **:old.**

```
…
AFTER UPDATE OF name ON publisher
…
INSERT INTO publish_audit
VALUES(SYSDATE, :new.name, :old.name);
…
```

A record of what has changed

Example trigger: business rules

```
CREATE OR REPLACE TRIGGER publish_trg
BEFORE INSERT OR UPDATE ON book
FOR EACH ROW
DECLARE
   how_many NUMBER;
BEGIN
   SELECT COUNT(*) INTO how_many FROM book
           WHERE publisher = :new.publisher;
   IF how_many >= 3 then
     Raise_application_error(-20000,'Publisher' ||
          :new.publisher || 'already has 3 books');
   END IF;
END;
```

Trigger header

PL/SQL Block

Example explained

```
CREATE OR REPLACE TRIGGER publish_trg
BEFORE INSERT OR UPDATE ON book
FOR EACH ROW
```

- Creates a trigger called publish_trg
- Executes before data in book table is added/updated
  - Can prevent the change if necessary
  - Not fired if deleting from book table
- Following block will execute once for each row
  - Whenever book table data is changed or added

Example explained

```
DECLARE
   how_many NUMBER;
BEGIN
   SELECT COUNT(*) INTO how_many FROM book
          WHERE publisher =  :new.publisher;
   IF how_many >= 3 THEN
     Raise_application_error(-20000,
              ('Publisher ' ||:new.publisher ||
               ' already has 3 books'));
   END IF;
END;
```

- Count how many records already exist with the same publisher value as the new record (the one being added or updated)
- If count is 3 or more, error fires - Output message and abort transaction

Raise_application_error

- Used inside trigger
- Purpose:
  - output an error message and
  - immediately **stop** the event that fired the trigger
  - For example, data insertion
- Can include variables/trigger values, see previous slide
- E.g.

```
RAISE_APPLICATION_ERROR(-20000,'trigger violated')
```

label

Message text

The Trigger in use

This should happen

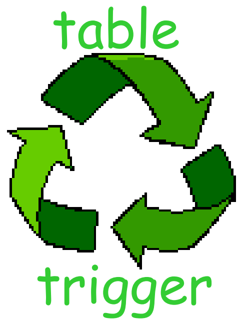
```
insert into book (ISBN,publisher) values('012345678','McGraw-Hill')
            *
ERROR at line 1:
ORA-20000: Publisher 4233 already has 3 books
ORA-06512: at "MCTPL.PUBLISH_TRG", line 7
ORA-04088: error during execution of trigger 'MCTPL.PUBLISH_TRG'
```

This is a common problem
When updating

```
update book set publisher='McGraw-Hill' where ISBN='0077095855'
       *
ERROR at line 1:
ORA-04091: table MCTPL.BOOK is mutating,
           trigger/function may not see it
ORA-06512: at "MCTPL.PUBLISH_TRG", line 4
ORA-04088: error during execution of trigger 'MCTPL.PUBLISH_TRG'
```

Trigger error: Table is mutating

- Example:
  Update to table x fires trigger
  trigger queries/ modifies table x



- This is a problem!
  - happens only for update, not for insert
  - Can be fixed by using two triggers, or dummy table, but quite tricky
    (see Casteel, J. (2003). Oracle 9i Developer: PL/SQL Programming. p.318)

WHEN clause

- Optional additional statement to control the trigger
- Takes a BOOLEAN SQL expression
  - Trigger fires if TRUE and not if FALSE
- Operates on a ROW level trigger
  - To prevent the trigger from firing in specific row cases
- WHEN (expression)

WHEN Example

```
create or replace trigger only_nulls
after update on BOOK
for each row
when (old.price is null)  -- notice the colon with OLD is not used here
begin
   insert into PriceChange values(:old.isbn,:new.price);
end;
```

Compilation errors

- When you create a trigger, Oracle responds
  - "Trigger created" or
  - "Warning: Trigger created with compilation errors."
- Type in the command

  SHOW ERRORS

  on its own to make the error messages visible

What to think about

- BEFORE or AFTER?
- INSERT, DELETE, UPDATE?
- FOR EACH ROW or once only?
- Any error conditions/messages?

- How can I test that the trigger works?

What else can we do with triggers

- DROP TRIGGER trigger_name;
  - No further firing will occur when dropped

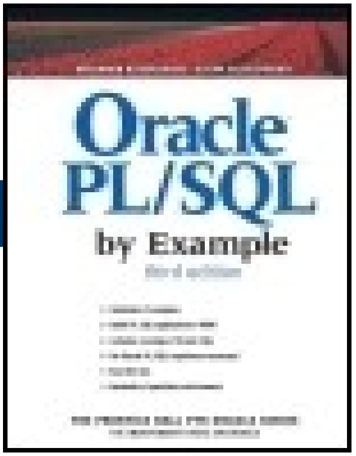
- ALTER TRIGGER trigger_name DISABLE;
- ALTER TRIGGER trigger_name ENABLE;

- ALTER TABLE table_name DISABLE ALL TRIGGERS;
- ALTER TABLE table_name ENABLE ALL TRIGGERS;

Trigger problems

- Easy to confuse := with = with = :old
- Compilation errors:
  use SHOW ERRORS to see them
- Errors may need to be solved in a different place from where they occur
- "table is mutating"

Reading & References



- In Safari e-books:
  Rosenzweig and Silvestrova (2003). Oracle® PL/SQL™ by Example, 3rd ed.

- Lots of Oracle/SQL books have sections on PL/SQL, e.g. (in library)
  - Shah, N. (2002). Database Systems Using Oracle. A simplified guide to SQL and PL/SQL. Ch. 8 onwards
  - Morris-Murphy chapters 15 & 16
  - Earp/Bagui Ch. 12, 13
- This goes way beyond our coverage:
  Casteel, J (2003). Oracle 9i Developer: PL/SQL Programming
- Connolly/Begg (3rd ed) Sections 8.2.5, 8.2.7, 27.4.11
- Elmasri section 24.1

Using the Data Dictionary to list stored programs

```
SELECT object_type, object_name
FROM user_objects
WHERE object_type
    IN ( 'PROCEDURE','FUNCTION','TRIGGER')
ORDER BY object_type;
```

Test_2

Test_3

Getting the code

```
SELECT text FROM user_source
  WHERE name ='SAL_TRG';
```

```
trigger sal_trg
after update on personnel_copy
for each row
when (old.salary > 20000 and old.bonus is not null)
begin
if inserting then
  insert into audit_salaries values(null,:new.salary);
  elsif deleting then
    insert into audit_salaries values(:old.salary, null);
    else
      insert into audit_salaries values(:old.salary,:new.salary);
end if;
end;
```

Triggers are in the Data Dictionary

- Info about your triggers is held in USER_TRIGGERS etc.

Trigger_name	Name of trigger
Triggering_event	INSERT, UPDATE, DELETE
Table_owner	Owner of the table attached to
Table_name	Name of the table attached to
Referencing_names	Names used for OLD and NEW
Status	Disabled or Enabled
Description	Trigger description
Trigger_body	Statements executed when fired