Creating Compound, DDL, and Event Database Triggers

Which Trigger will be fired First ????

```
create or replace trigger t1
before
insert
on emp
begin
insert into which_fired_first values ( s1.nextval,'t1');
end;
```

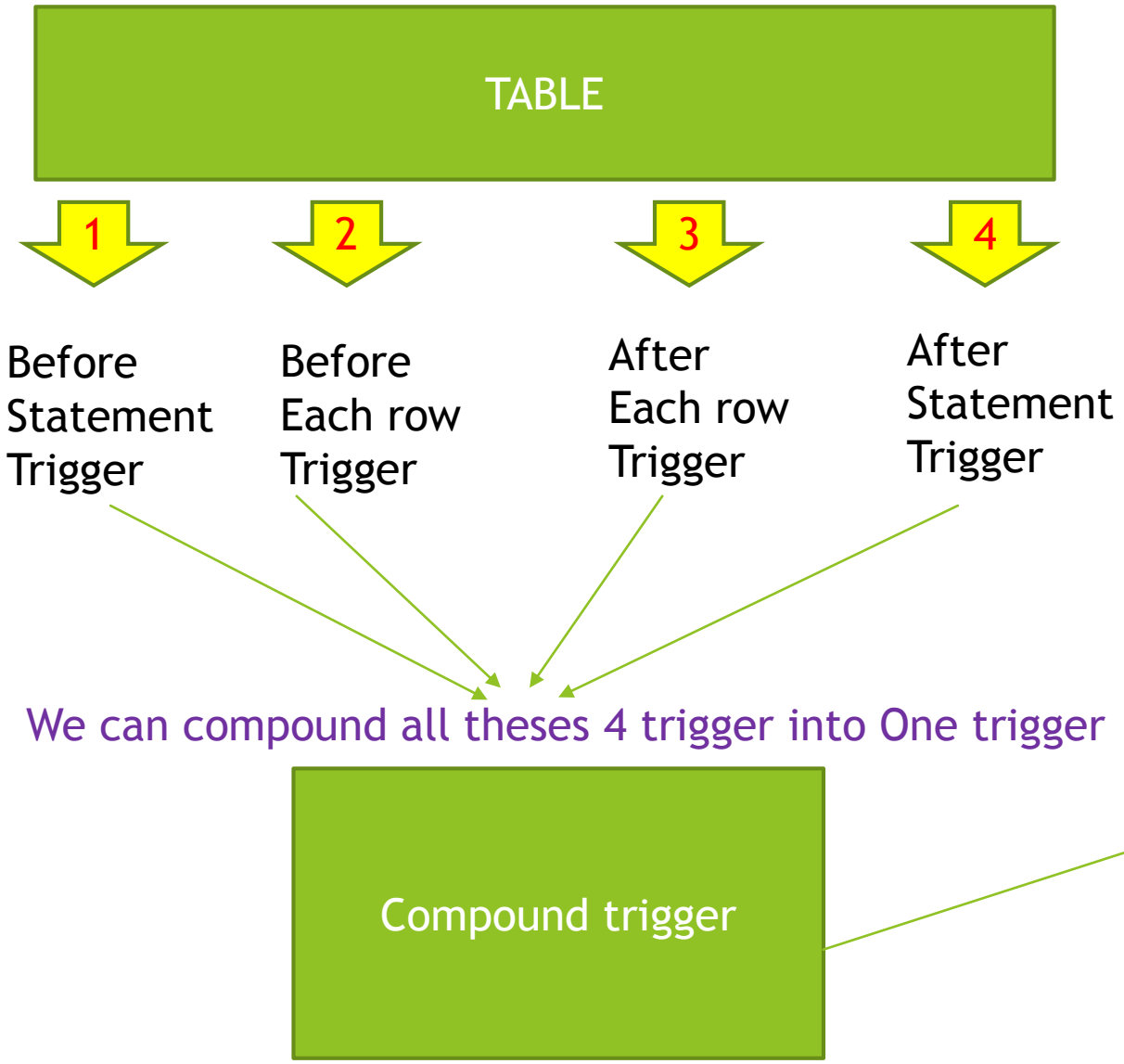
```
create or replace trigger t2
before
insert
on emp
begin
insert into which_fired_first values ( s1.nextval,'t2');
end;
```

In this case you can not guarantee because both are same level.

If the order is important for your code then
You can use follows

```
create or replace trigger t2
before
insert
on emp
follows t1
begin
insert into which_fired_first values ( s1.nextval,'t2');
end;
```

What is Compound Triger




```
-- Optional section
BEFORE STATEMENT IS ...;
```

```
-- Optional section
BEFORE EACH ROW IS ...;
```

```
-- Optional section
AFTER EACH ROW IS ...;
```

```
-- Optional section
AFTER STATEMENT IS ...;
```

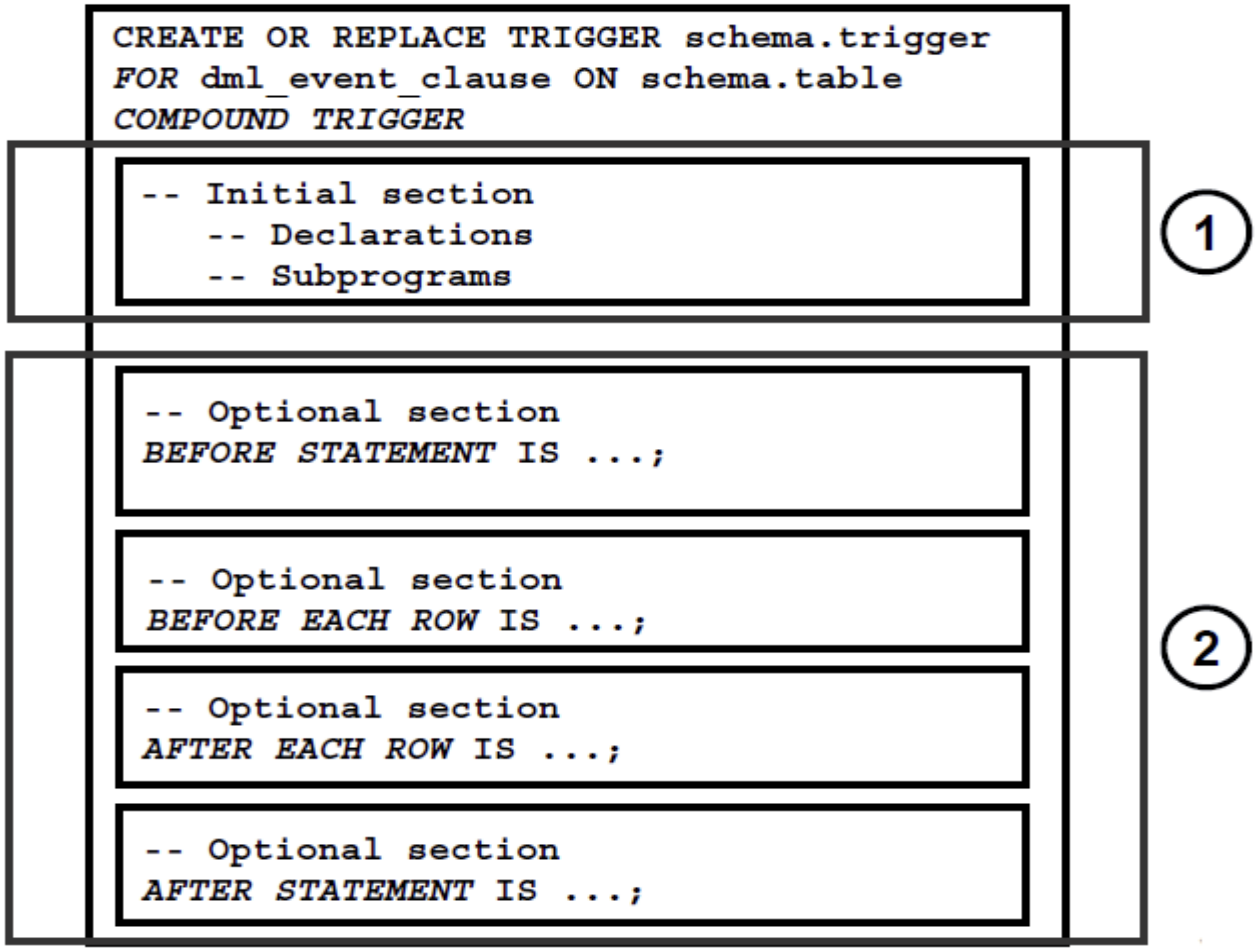
What Is a Compound Trigger?

A single trigger on a table that allows you to specify actions for each of the following four timing points:

- Before the firing statement
- Before each row that the firing statement affects
- After each row that the firing statement affects
- After the firing statement

- Starting from 11g

Compound Trigger Structure for Tables



```
create or replace trigger comp_test
for insert or update or delete
on test_emp
compound trigger

    before statement is
    begin
    insert into test_emp_sequence values (s.nextval,'before_insert_stat');
    end  before statement;

    before each row is
    begin
    insert into test_emp_sequence values (s.nextval,'before_insert_each_row');
    end before each row;

    after each row is
    begin
    insert into test_emp_sequence values (s.nextval,'after_insert_each_row');
    end after each row;

    after statement is
    begin
    insert into test_emp_sequence values (s.nextval,'after_insert_stat');
    end after statement;
end;
```

Same order

Compound Trigger Restrictions

- A compound trigger must be a DML trigger and defined on either a table or a view.
- The body of a compound trigger must be compound trigger block, written in PL/SQL.
- A compound trigger body cannot have an initialization block; therefore, it cannot have an exception section.
- An exception that occurs in one section must be handled in that section. It cannot transfer control to another section.
- `:OLD` and `:NEW` cannot appear in the declaration, `BEFORE STATEMENT`, or the `AFTER STATEMENT` sections.
- Only the `BEFORE EACH ROW` section can change the value of `:NEW`.
- The firing order of compound triggers is not guaranteed unless you use the `FOLLOWS` clause.

Compound Trigger Structure for Views

```
CREATE OR REPLACE TRIGGER
schema.trigger

FOR dml_event_clause ON schema.view

COMPOUND TRIGGER

-- Initial section
   -- Declarations
   -- Subprograms

-- Optional section (exclusive)
INSTEAD OF EACH ROW IS
...;
```

Trigger Restrictions on Mutating Tables

- A mutating table is:
  - A table that is being modified by an `UPDATE`, `DELETE`, or `INSERT` statement, or
  - A table that might be updated by the effects of a `DELETE CASCADE` constraint
- The session that issued the triggering statement cannot query or modify a mutating table.
- This restriction prevents a trigger from seeing an inconsistent set of data.
- This restriction applies to all triggers that use the `FOR EACH ROW` clause.
- Views being modified in the `INSTEAD OF` triggers are not considered mutating.

SQL Error: ORA-04091 table xxxx is mutating

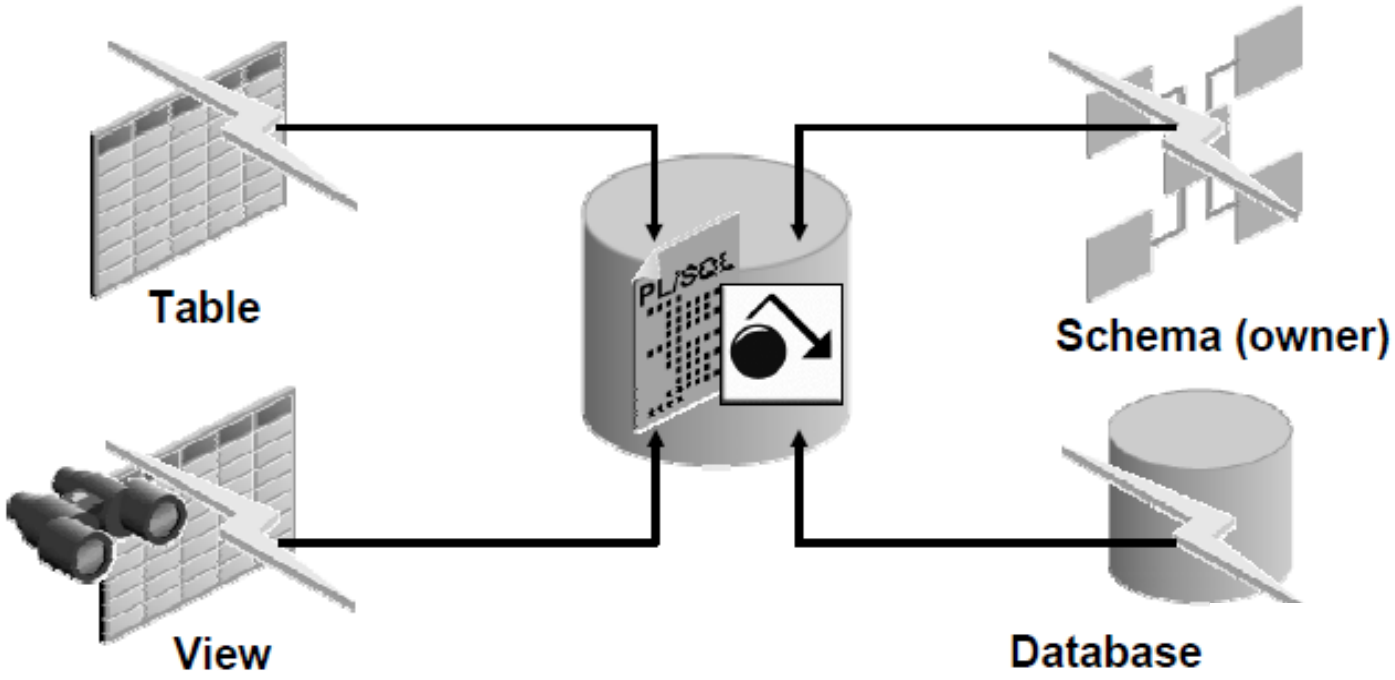
Always come form: for each row triggers

Comparing Database Triggers to Stored Procedures

Triggers	Procedures
Defined with `CREATE TRIGGER`	Defined with `CREATE PROCEDURE`
Data dictionary contains source code in `USER_TRIGGERS`	Data dictionary contains source code in `USER_SOURCE`
Implicitly invoked by DML	Explicitly invoked
`COMMIT`, `SAVEPOINT`, and `ROLLBACK` are not allowed	`COMMIT`, `SAVEPOINT`, and `ROLLBACK` are allowed

Defining Triggers

A trigger can be defined on the table, view, schema (schema owner), or database (all users).



Creating Triggers on DDL Statements

```
CREATE [OR REPLACE] TRIGGER trigger_name
BEFORE | AFTER -- Timing
[ddl_event1 [OR ddl_event2 OR ...]]
ON {DATABASE | SCHEMA}
trigger_body
```

Sample DDL Events	Fires When
CREATE	Any database object is created using the CREATE command.
ALTER	Any database object is altered using the ALTER command.
DROP	Any database object is dropped using the DROP command.

ALTER
ANALYZE
ASSOCIATE STATISTICS
AUDIT
COMMENT
CREATE
DISASSOCIATE STATISTICS
DROP
GRANT
NOAUDIT
RENAME
REVOKE
TRUNCATE

DDL (ALL the above)

Sample DDL Triger on HR schema

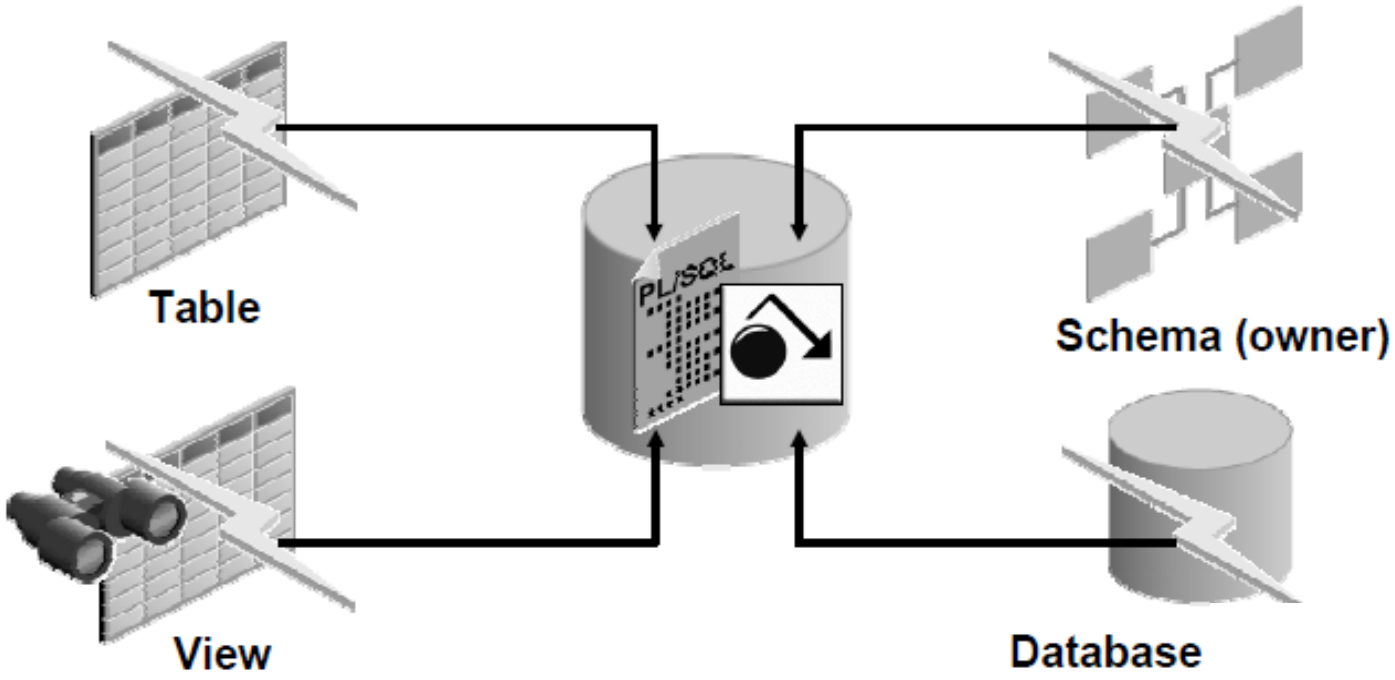
```
CREATE OR REPLACE TRIGGER before_create_trigger
BEFORE CREATE
 ON SCHEMA
BEGIN
if  to_number(to_char(sysdate,'hh24')) not between 8 and 16 then
raise_application_error(-20001, 'Create table not Allowed now');
end if;
END;

create table  t2 (value1 number );
```

If the User HR tried to create table at 18:00 for example
It will give Error -20001 create table not allowed now

Defining Triggers

A trigger can be defined on the table, view, schema (schema owner), or database (all users).



Creating Database-Event Triggers

- **Triggering a user event:**
  - CREATE, ALTER, or DROP
  - Logging on or off
- **Triggering database or system event:**
  - Shutting down or starting up the database
  - A specific error (or any error) being raised

```
CREATE OR REPLACE TRIGGER before_create_trigger
BEFORE CREATE
 ON database
BEGIN
if  to_number(to_char(sysdate,'hh24')) not between 8 and 16 then
raise_application_error(-20001, 'Create table not allword now');
end if;
END;

create table  t2 (value1 number );
```

```
CREATE [OR REPLACE] TRIGGER trigger_name
BEFORE | AFTER -- timing
[database_event1 [OR database_event2 OR ...]]
ON {DATABASE | SCHEMA}
trigger_body
```

Database Event	Triggers Fires When	
AFTER SERVERERROR	An Oracle error is raised	
AFTER LOGON	A user logs on to the database	Can be on schema also
BEFORE LOGOFF	A user logs off the database	Can be on schema also
AFTER STARTUP	The database is opened	Only Database
BEFORE SHUTDOWN	The database is shut down normally	Only Database

Sample Database event Trigger

Sys DBA create this trigger on specific pluggable database

```
create or replace trigger logon_t
after
logon
on database
begin
insert into log_table values (user,sysdate,'logon');
end;
```

```
create or replace trigger logoff_t
before
logoff
on database
begin
insert into log_table values (user,sysdate,'logoff');
end;
```

CALL Statements in Triggers

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing
event1 [OR event2 OR event3]
ON table_name
[REFERENCING OLD AS old | NEW AS new]
[FOR EACH ROW]
[WHEN condition]
CALL procedure_name
/
```

```
CREATE OR REPLACE PROCEDURE log_execution IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('log_exection: Employee Inserted');
END;
/
CREATE OR REPLACE TRIGGER log_employee
BEFORE INSERT ON EMPLOYEES
CALL log_execution  -- no semicolon needed
/
```

You can also pass parameters to the procedure
Call p1(:new.emp_id)

maxvlearn

Benefits of Database-Event Triggers

- Improved data security:
  - Provide enhanced and complex security checks
  - Provide enhanced and complex auditing
- Improved data integrity:
  - Enforce dynamic data integrity constraints
  - Enforce complex referential integrity constraints
  - Ensure that related operations are performed together implicitly

System Privileges Required to Manage Triggers

The following system privileges are required to manage triggers:

- The `CREATE/ALTER/DROP (ANY) TRIGGER` privilege that enables you to create a trigger in any schema
- The `ADMINISTER DATABASE TRIGGER` privilege that enables you to create a trigger on `DATABASE`
- The `EXECUTE` privilege (if your trigger refers to any objects that are not in your schema)

Guidelines for Designing Triggers

- You can design triggers to:
  - Perform related actions
  - Centralize global operations
- You must not design triggers:
  - Where functionality is already built into the Oracle server
  - That duplicate other triggers
- You can create stored procedures and invoke them in a trigger, if the PL/SQL code is very lengthy.
- Excessive use of triggers can result in complex interdependencies, which may be difficult to maintain in large applications.

Thank You