

Data Advanced

H3. Triggers

Koen Bloemen
Sander De Puydt



**DE HOGESCHOOL
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be





Triggers

- DML triggers
- DDL triggers
- Beheer database triggers
- Database triggers

Database triggers

- Wat bedoelen we met een trigger?
 - *Hebben we in een ander vak al eens met triggers gewerkt?*
- Waarom zouden we triggers willen gebruiken?
 - *Wat zijn de voordelen?*



Event code
PXLDATA

Op [wooclap.com](https://app.wooclap.com)

<https://app.wooclap.com/PXLDATA?from=instruction-slide>

Database triggers – Waarom?

- Use case: Webshop (WPL 2)
 - *Als een klant een bestelling plaatst*
 - *Hoe gaan we automatisch de beschikbare hoeveelheid aanpassen in de PRODUCTEN tabel?*
 - *Als klanten hun bestelling nog kunnen aanpassen*
 - *Hoe gaan we een totaalprijs herberekenen van een order?*
 - *Als we een spoor willen bijhouden van aanpassingen in de database*
 - *Hoe zetten we een audit trail op?*



Database triggers – Waarom?

- **Voordelen** van triggers
 - We kunnen de correctheid/integriteit van data controleren en aanpassen indien nodig
 - *Data validatie **voor** en **na** insert of update*
 - Triggers kunnen ons helpen met logs bij te houden
 - *Welke aanpassingen gebeuren wanneer?*
 - *Wie doet wat?*
 - We kunnen logica verwerken in hoe onze records zich moeten gedragen (denk aan totaal prijs, updaten van product stock)
 - Client-side code wordt verminderd met triggers.

Triggers



Database triggers

- Opgebouwd uit PL/SQL code
 - Eerste stappen PL/SQL
- Gekoppeld aan een tabel/view
- Opgeslagen in de DB
- Wordt automatisch uitgevoerd (kan niet manueel opgeroepen worden)
 - **DDL** commando's (CREATE, DROP, RENAME, ...)
 - **DML** commando's (INSERT, UPDATE, DELETE)
- Wordt uitgevoerd ONGEACHT de omgeving (SQLPLUS, SQL Developer, applicatie, ...)
- COMMIT en ROLLBACK zijn niet toegelaten
- Triggers op SELECT zijn niet mogelijk

DML triggers

- Syntax
 - Timing
 - Event
 - Frequency
 - Declare
 - Voorbeeld
- Errors
- Naming conventions
- Gebruik
- Row triggers



DML triggers - syntax

CREATE {OR REPLACE} TRIGGER *triggernaam*

{BEFORE / AFTER}

timing

{DELETE / INSERT / UPDATE [OF *kolom* [,*kolom*]] } [OR ...]

event

ON *tabelnaam*

[FOR EACH ROW [WHEN (*voorwaarde*)]]

frequency

[DECLARE]

trigger body

BEGIN

*/*uitvoerbare commando's*/*

[EXCEPTION]

END [*triggernaam*];

DML triggers - timing

- **BEFORE** event
 - Wanneer de trigger moet controleren of een actie is toegestaan of niet (voorkomt onnodige rollbacks)
 - Wanneer je zeker wil zijn dat deze trigger altijd afgaat (want deze wordt als eerste uitgevoerd)
- **AFTER** event
 - Wanneer het event zeker moet uitgevoerd worden (vooral bij row triggers)
 - Is de trigger afhankelijk van de informatie van het event?
 - Wordt pas uitgevoerd **nadat** alle constraints op de tabel gecontroleerd zijn

```
CREATE OR REPLACE TRIGGER triggernaam
    {BEFORE / AFTER}
    {DELETE / INSERT / UPDATE [OF kolom [,kolom] ] } [ OR ...]
    ON tabelnaam
    [FOR EACH ROW [WHEN (voorwaarde) ] ] [DECLARE ]
BEGIN
    /*uitvoerbare commando's*/
[EXCEPTION]
END [triggernaam];
```

DML triggers - event

- {DELETE / INSERT / UPDATE [OF *kolom* [,*kolom*]] } [OR ...]
 - Bij het verwijderen (DELETE), toevoegen (INSERT) of wijzigen (UPDATE) van één of meerdere kolommen
 - Je kan DELETE, INSERT en UPDATE combineren met OR
- ON *tabelnaam*
 - Op welke tabel het event van toepassing is

Voorbeeld:

```
CREATE TRIGGER bdus_emp  
  BEFORE DELETE OR UPDATE  
  ON employees  
BEGIN ... (trigger_body)
```

```
CREATE OR REPLACE TRIGGER triggernaam  
  {BEFORE / AFTER}  
  {DELETE / INSERT / UPDATE [OF kolom [,kolom] ] } [ OR ...]  
  ON tabelnaam  
  [FOR EACH ROW [WHEN (voorwaarde) ] ] [DECLARE ]  
BEGIN  
  /*uitvoerbare commando's*/  
[EXCEPTION]  
END [triggernaam];
```

DML triggers - frequency

- [FOR EACH ROW [WHEN (*voorwaarde*)]]
 - Is optioneel!
 - Wanneer niet gedefinieerd => 1x uitgevoerd
 - Eventueel ook voorwaarden op te leggen

```
CREATE OR REPLACE TRIGGER triggernaam
    {BEFORE / AFTER}
    {DELETE / INSERT / UPDATE [OF kolom [,kolom] ] } [ OR ...]
    ON tabelnaam
    [FOR EACH ROW [WHEN (voorwaarde) ] ] [DECLARE ]
BEGIN
    /*uitvoerbare commando's*/
[EXCEPTION]
END [triggernaam];
```

DML triggers - declare

- [DECLARE]
 - Is optioneel!
 - Om variabelen te declareren

Voorbeeld:

```
CREATE TRIGGER ads_emp_count
    AFTER DELETE ON employees
    DECLARE
        n    INTEGER;
BEGIN
    SELECT COUNT(*) INTO n
    FROM employees;
END;
```

```
CREATE OR REPLACE TRIGGER triggernaam
    {BEFORE / AFTER}
    {DELETE / INSERT / UPDATE [OF kolom [,kolom] ]} [OR ...]
    ON tabelnaam
    [FOR EACH ROW [WHEN (voorwaarde) ] ] [DECLARE ]
BEGIN
    /*uitvoerbare commando's*/
[EXCEPTION]
END [triggernaam];
```


DML triggers - voorbeeld

```
CREATE OR REPLACE TRIGGER bds_mdw
    BEFORE DELETE
    ON medewerkers
BEGIN
    IF USER != 'JAN' THEN
        RAISE_APPLICATION_ERROR(-20000,
            'U heeft geen rechten voor deze actie');
    END IF;
END;
```

DML triggers – Raise_Application_Error

- Met het SQL command **RAISE_APPLICATION_ERROR** kan een fout geraised worden
- De foutmelding wordt op het scherm afgedrukt
- De foutcode kan gebruikt worden in elke applicatie
- **Het “programma” wordt afgebroken**
- **Automatische rollback** voor het triggerende DML-statement (in het voorbeeld: het DELETE statement)
- De foutcode moet liggen in de user-defined range van -20000 tot -20999



DML triggers – Naming conventions

- De naam van de trigger geeft best het soort trigger weer
Voorbeeld: **bds_mdw**
- b = before of a = after
De trigger gaat af vóór het DML statement wordt uitgevoerd
- d = delete of u = update of i = insert, ...
De trigger gaat af bij een delete-statement
- s = statement trigger of r = rows
Deze trigger gaat slechts 1x af per DML-statement, ongeacht hoeveel rijen door het DML-statement worden bewerkt



DML triggers – creatie uitvoeren

- Zoals elk commando wordt de trigger aangemaakt bij uitvoer van het commando (**r(un) of /**)
- Indien foutloos wordt de gecompileerde versie van de trigger opgeslagen in de database
- Broncode wordt sowieso opgeslagen in de data dictionary van de database
- In geval van fouten (melding “created with compilation errors”) => *show errors* om fouten te tonen

DML triggers – gebruiken

- Een DML trigger gaat automatisch af bij het uitvoeren van een DML-statement
- Er kunnen meerdere triggers op 1 tabel gemaakt worden (let op volgorde van uitvoering!)

DML triggers – uitbreiding

```
CREATE OR REPLACE TRIGGER bdus_mdw
  BEFORE DELETE OR UPDATE OF maandsal
  ON medewerkers
BEGIN
  IF USER != 'JAN' THEN
    IF DELETING THEN
      RAISE_APPLICATION_ERROR(-20000,
        'U heeft geen rechten om te verwijderen');
    ELSE
      RAISE_APPLICATION_ERROR(-20001,
        'U heeft geen rechten om het maandsal te wijzigen');
    END IF;
  END IF;
END;
```

DML triggers – uitbreiding

- Detecteren van DML operation
 - INSERTING – DELETING – UPDATING
- Te gebruiken indien er meer dan 1 triggerend event is op 1 tabel
- Geven een boolean terug
- Bij UPDATING kan er ook een parameter meegegeven worden
 - IF updating('maandsal') THEN...

Row triggers

```
CREATE OR REPLACE TRIGGER aur_mdw_maandsal
  AFTER UPDATE OF maandsal
  ON medewerkers
  FOR EACH ROW
BEGIN
  IF (:NEW.maandsal - :OLD.maandsal > 0.1 * :OLD.maandsal) THEN
    RAISE_APPLICATION_ERROR(-20000, 'Maandsal te veel verhoogd');
  END IF;
END;
```


Row triggers

- Bovenaan de trigger definitie: FOR EACH ROW
- Zorgt ervoor dat je trigger VOOR ELKE RIJ afgaat
- Voorbeelden:
 - Een delete command verwijdt 10 rijen
 - Statement-trigger gaat 1 keer af
 - Row-trigger gaat 10 keer af
 - Een delete command verwijdt 0 rijen
 - Statement-trigger gaat 1 keer af
 - Row-trigger gaat niet af

Row triggers

- Mogelijk om de oude en de nieuwe waarde op te vragen
 - **:OLD.kolomnaam**
 - **:NEW.kolomnaam**
- Enkel bij een update statement bevatten beiden een waarde
- Voorbeelden:
 - **:NEW.maandsal**
Bevat de nieuwe waarde voor maandsal na uitvoering van een insert of update statement.
Let op: bij DELETE is deze variabele leeg!
 - **:OLD.maandsal**
Bevat de oude waarde voor maandsal voor uitvoering van een update of delete statement.
Let op: bij INSERT is deze variabele leeg!

Row triggers - uitbreiding

```
CREATE OR REPLACE TRIGGER aur_mdw_maandsal
  AFTER UPDATE OF maandsal
  ON medewerkers
  FOR EACH ROW
  WHEN (OLD.mnr > 7500)
BEGIN
  IF (:NEW.maandsal - :OLD.maandsal > 0.1 * :OLD.maandsal) THEN
    RAISE_APPLICATION_ERROR(-20000, 'Maandsal te veel verhoogd');
  END IF;
END;
```

- WHEN conditie filtert op rij niveau
- ENKEL mogelijk voor row triggers!
- LET OP! **Geen ":" bij OLD** keyword

Volgorde van uitvoering

Indien meerdere triggers op 1 DML-statement:

1. Alle BEFORE STATEMENT triggers
2. Voor elke rij uit de ROW triggers
 - a. Alle BEFORE ROW triggers voor die rij
 - b. Triggerende DML-statement + integrity constraints checken voor die rij
 - c. Alle AFTER ROW triggers voor die rij
3. Alle AFTER STATEMENT triggers

Trigger keuze

- Gebruik een **row trigger** als de inhoud van de kolommen nodig is
- Gebruik een **before statement trigger** als de trigger MOET afgaan
- Gebruik eerder een **before statement-trigger** dan een after statement trigger (vooral bij controle of een actie toegestaan is, dit voorkomt rollbacks)
- Gebruik liever een **after row trigger** dan een before row trigger (Oracle controleert dan eerst de constraints)
- Gebruik een **before row trigger** als de inhoud van een kolom in de trigger gewijzigd wordt

DDL triggers

- Syntax
- Voorbeeld



DDL triggers - syntax

CREATE OR REPLACE TRIGGER *triggernaam*

{BEFORE / AFTER}

{DDL EVENT} →

ON {DATABASE / SCHEMA}

[WHEN (*voorwaarde*)]

[DECLARE]

BEGIN

*/*uitvoerbare commando's*/*

[EXCEPTION]

END [*triggernaam*];

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

DDL triggers - voorbeeld

```
CREATE OR REPLACE TRIGGER before_create_trigger
    BEFORE CREATE
    ON schema
BEGIN
    IF to_number(to_char(sysdate,'hh24')) not between 8 and 12 THEN
        RAISE_APPLICATION_ERROR(-20000,
            'U mag enkel creëren tussen 8 en 12 u');
    END IF;
END;
```

DDL triggers - voorbeeld

```
CREATE TABLE ddl_log (operation VARCHAR2(30), obj_owner VARCHAR2(30),  
object_name VARCHAR2(30), attempt_by VARCHAR2(30), attempt_dt DATE);
```

```
CREATE OR REPLACE TRIGGER after_ddl_trigger
```

```
    AFTER DDL
```

```
    ON schema
```

```
BEGIN
```

```
    INSERT INTO ddl_log
```

```
    SELECT ora_sysevent, ora_dict_obj_owner, ora_dict_obj_name, USER,  
    SYSDATE
```

```
    FROM DUAL;
```

```
END;
```

DDL triggers - voorbeeld

- De nieuwe tabel `ddl_log` is nu nog leeg
- Voer een DDL command uit
- Kijk terug in de `ddl_log` tabel
- Verwijder de `ddl_log` tabel

Beheer van triggers

- Triggers opvragen
- Broncode opvragen
- Triggers verwijderen
- Triggers (de)-activeren
- Invalid triggers



Beheer van triggers

- **Triggers opvragen**

```
SELECT object_name, created, status  
FROM user_objects  
WHERE object_type = 'TRIGGER';
```

```
SELECT trigger_name, trigger_type, triggering_event, table_name, status  
FROM user_triggers;
```

- **Broncode opvragen**

```
SELECT line, text  
FROM user_source  
WHERE name = 'AUR_MDW_MAANDSAL';
```

```
SELECT trigger_type, trigger_body  
FROM user_triggers  
WHERE trigger_name = 'AUR_ MDW_MAANDSAL';
```

Beheer van triggers

- **Trigger verwijderen**

DROP TRIGGER triggernaam

- **1 specifieke trigger activeren/desactiveren**

ALTER TRIGGER triggernaam ENABLE

ALTER TRIGGER triggernaam DISABLE

- **Alle triggers van 1 specifieke tabel activeren/desactiveren**

ALTER TABLE tabelnaam ENABLE ALL TRIGGERS

ALTER TABLE tabelnaam DISABLE ALL TRIGGERS

Beheer van triggers

- Een trigger die **invalid** geworden is kan opnieuw gecompileerd worden

ALTER TRIGGER triggernaam **COMPILE**

- **Hoe** kan een trigger invalid worden?

Wanneer er bijvoorbeeld een **wijziging** is in de **structuur** van de gerefereerde tabel



INVALID

Database triggers

- Database operations
- Syntax



Database triggers – Database operations

- Er zijn een aantal database operations waarvoor we triggers kunnen schrijven:
 - **LOGON**
 - **LOGOFF**
 - **STARTUP**
 - **SHUTDOWN**
 - **SERVERERROR**

Database triggers - Syntax

```
CREATE OR REPLACE TRIGGER after_logon_trigger
```

```
    AFTER LOGON
```

```
    ON database
```

```
DECLARE
```

```
    osUser VARCHAR2(30);
```

```
    machine VARCHAR2(100);
```

```
    prog VARCHAR2(100);
```

```
    ip_user VARCHAR2(15);
```

```
BEGIN
```

```
    SELECT OSUSER, MACHINE, PROGRAM, ora_client_ip_address
```

```
    INTO osUser, machine, prog, ip_user
```

```
    FROM v$session
```

```
    WHERE SID = SYS_CONTEXT('USERENV', 'SID');
```

```
    IF (User = 'JOHN') THEN
```

```
        RAISE_APPLICATION_ERROR(-20000, 'Denied!  You are not allowed to logon as ' || User || ' ' || prog || ' using ' || osUser);
```

```
    END IF;
```

```
END;
```

Oefeningen!

