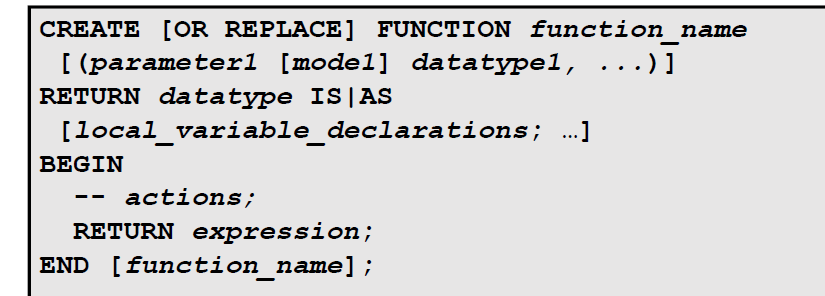
# BEGIN voor elke keer opstarten van sqlplus

SET SERVEROUT ON

# Functions

De syntax van een functie is als het volgt



Hier kan je create doen of een bestaane functie vervangen

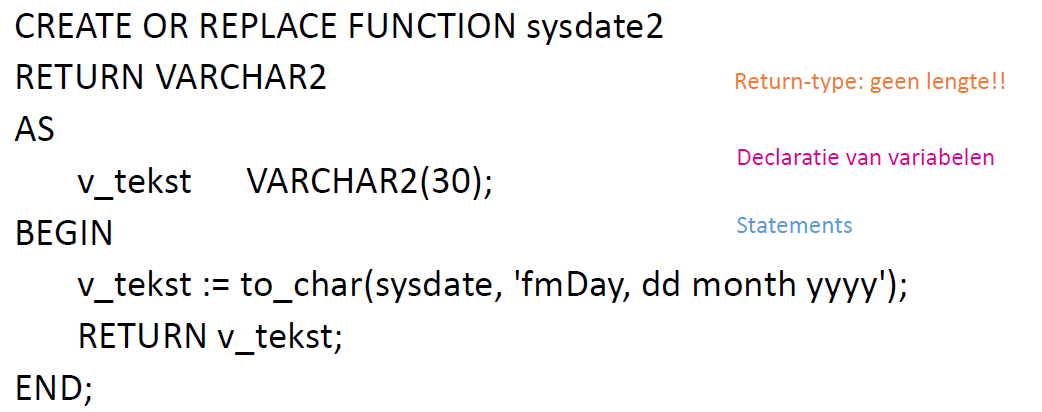
Dan de parameter geef je mee met de naam p\_iets.

Dit komt of uit de database of geef je mee als data.

Daarna zeg je wat er terug moet gegeven worden

Daarna zeg je welke acties moet gebeuren en geef je een waarde terug en zeg je op het einde end

Bv



Elke instructie eindigt met een ;

Elke toekenning gebeurt door :=

Je kan ook andere functies gebeuren door ze toe te kennen aan een andere variabele die he hebt aangemaakt bv v\_lengte := LENGTH (v\_naam)

Commentaar kan getypt worden door of

Door twee liggende streepjes – voor één lijn

of voor meerdere lijnen /\* commentaar \*/

Als er de melding komt created with comilation errors kan je dit eenvoudig opvragen met het commando “show errors”. Deze zegt waar er ergens de fout zit ipv totaal geen idee te hebben. Vaak zit het op de rij zelf of de vorige rij als je bijvoorbeeld een ; vergeten bent.

De voorwaarden van het aanmaken van de naamgeving van een variabele is:

Moet beginnen meet een letter

Mag letters en getallen bevatten

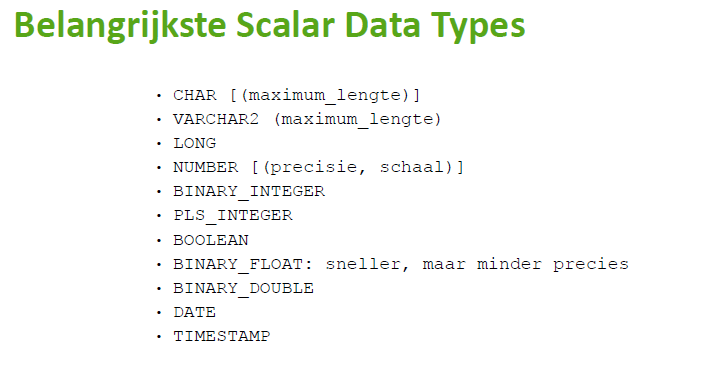
Mag bevatten dollar teken ($), underscore, pond teken(alle andere tekens mag het dus niet bevatten)

Maximale lengte is 30 tekens

Geen gereserveerde woorden

Moet beginnen met v\_

De datatypes die wij gebruiken



## Declaratie met %TYPE

Dit maakt een verwijzing naar een object in de database. Dit zorgt er voor dat het type

Vb:

V\_getal NUMBER(4,1);

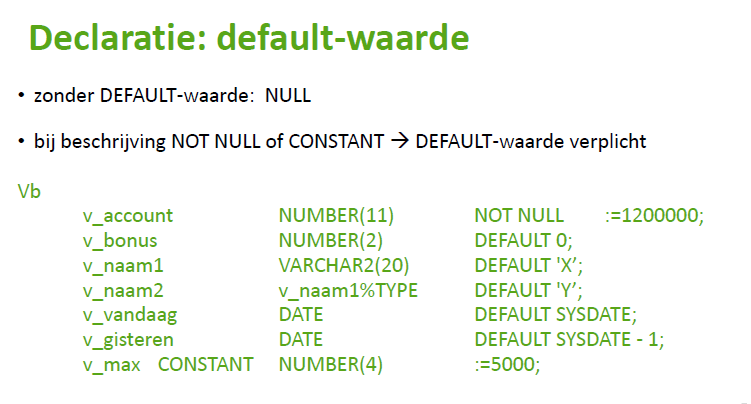
V\_getal2 v\_getal%TYPE;

V\_mndsal employees.salary%TYPE

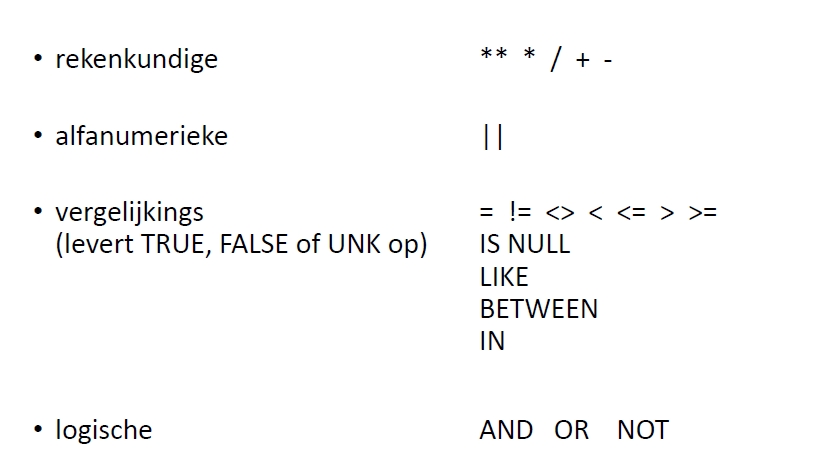
In dit voorbeeld is V\_getal dus een nummer met maximum 4 cijfers en één cijfer na de komma. V\_getal2 is juist het zelfde als V\_getal dus ook een nummer met maximum 4 cijfers en één na de komma.

En V\_mndsal heeft de zelfde waarde als het salary van employees in de database.

Dit is handig als je een varialbe gelijk stelt aan iets uit de database.

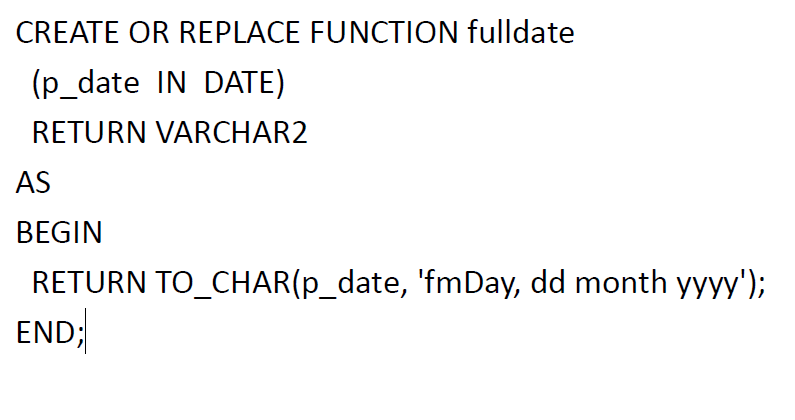


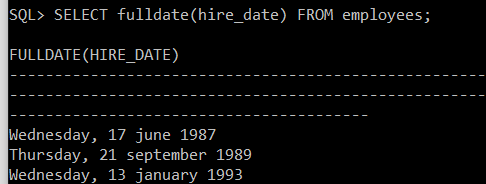
Enkele wiskundige operatoren



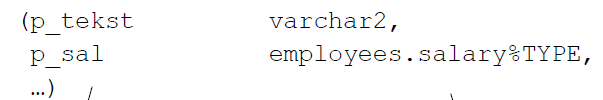
(\*\* nog uit te zoeken wat dit juist doet)

Voorbeeld om een dagen te krijgen in een andere manier uit de database:

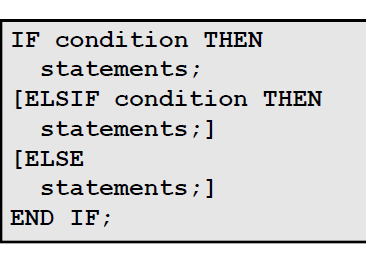




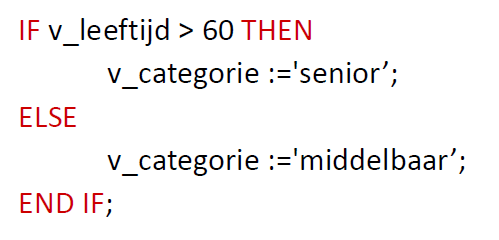
Als je meerdere parameters wilt ingeven kan je dit doen door deze te scheiden met een komma. Dus je schrijft geen twee keer een haakje maar zoals hier onder.



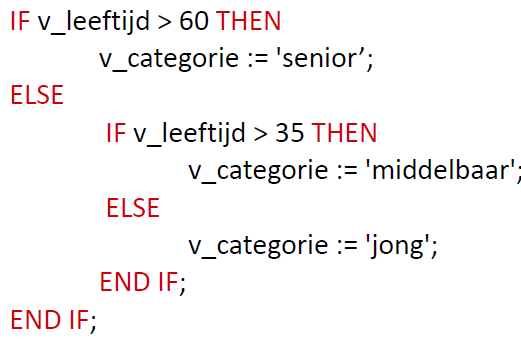
## IF THEN ELSIF ENDIF



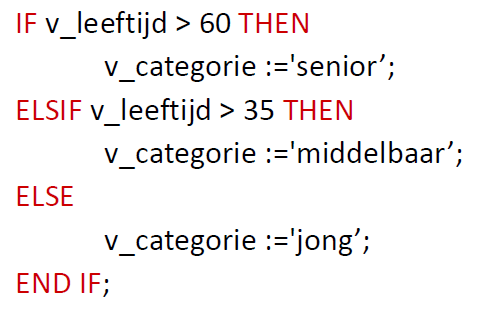
eenvoudig Vb:



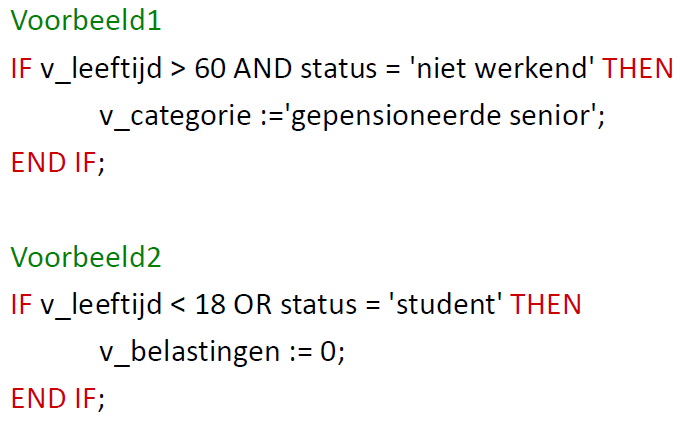
Geneste IF structuur vb:



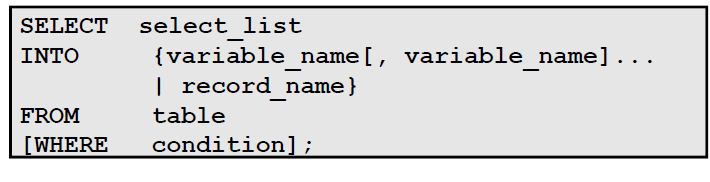
ElsIF structuur vb:



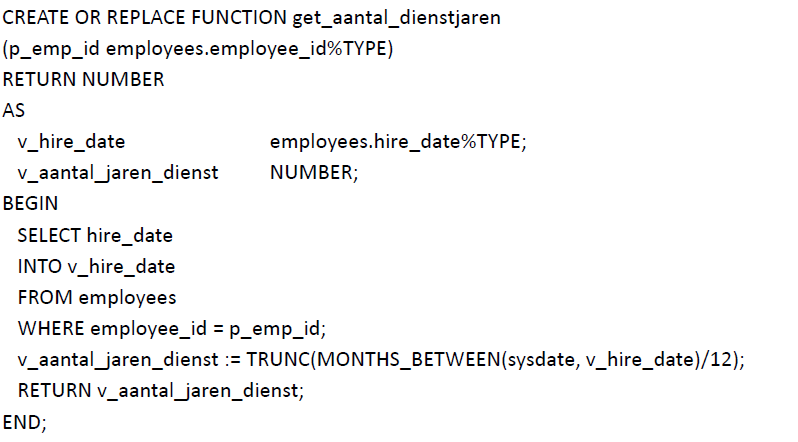
AND en OR structuur vb:



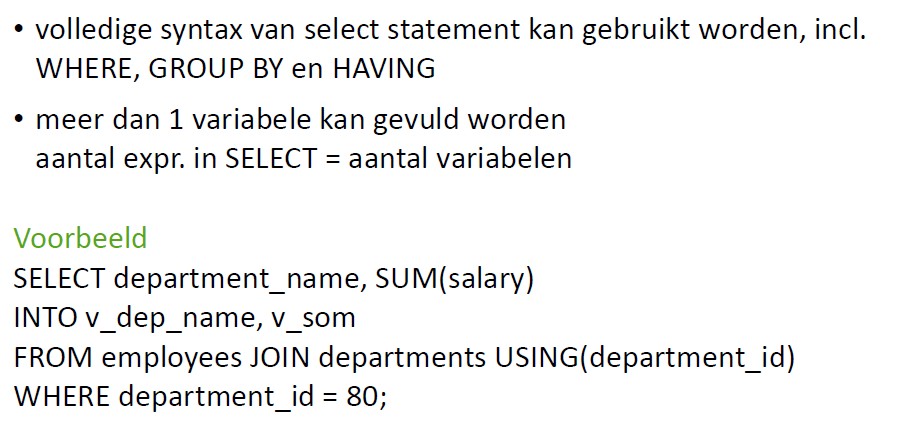
Je kan ook in je functie een SELECT statement gebruiken zodat deze functie iets terug geeft waar deze gelijk aan iets dat je hebt meegeven.



Bv:



Opmerking



Je kan ook een functie verwijderen door het commando

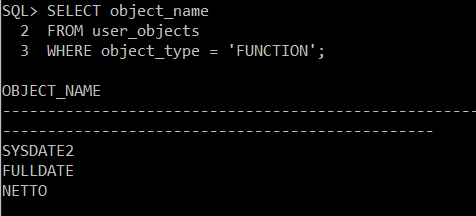
DROP FUNCTION function\_name

Vb

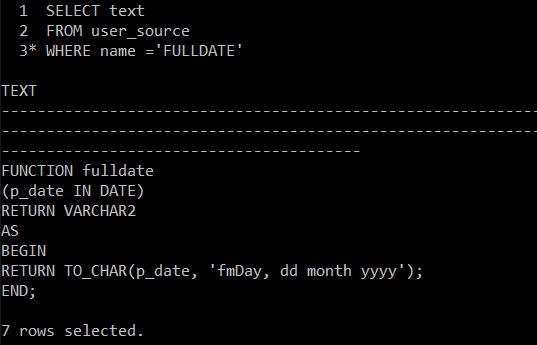
DROP FUNCTION get\_jaarsal;

## Opvragen kenmerken

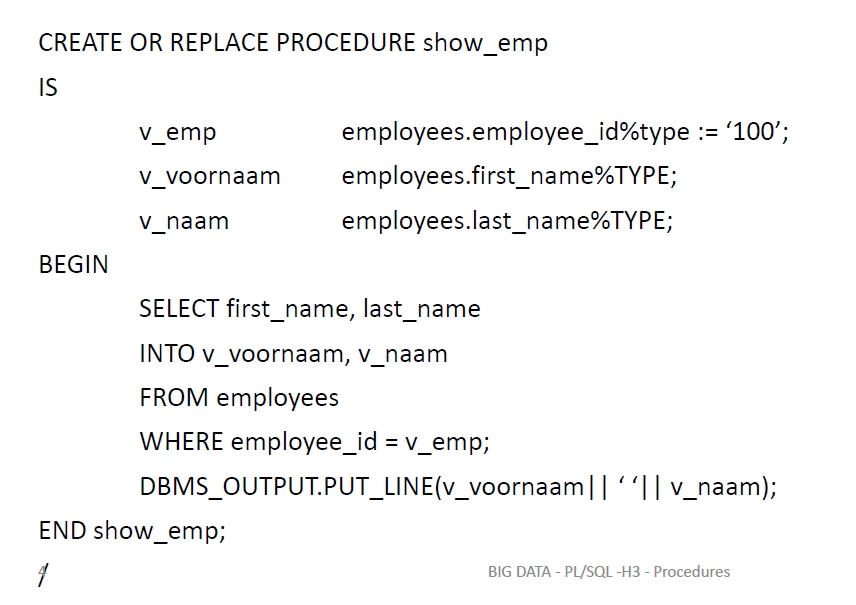
Je kan de functies die geschreven zijn opvragen door het commando



Opvragen van een functie kan je doen door het commando:



# Procedures

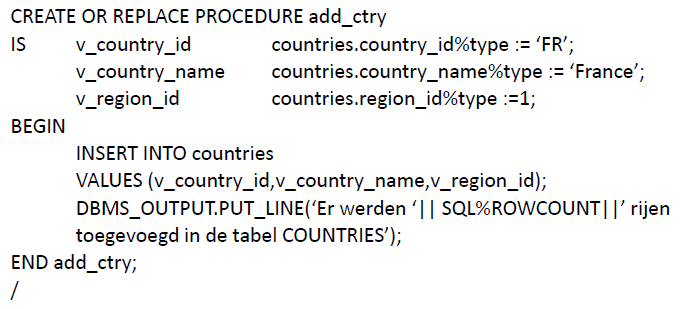


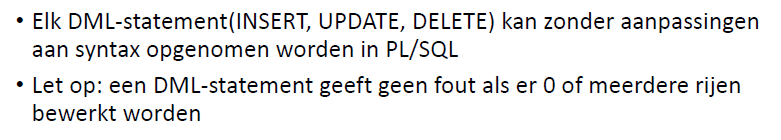
Je kan bij procedures ook alles wat er juist gebeurd in een lijn laten afdrukken door het commando “DBMS\_OUTUT.PUT\_LINE(‘iets’);”

Als je dit wilt gebruiken moet je in je sql plus eerst “SET SERVEROUTPUT ON” typen.

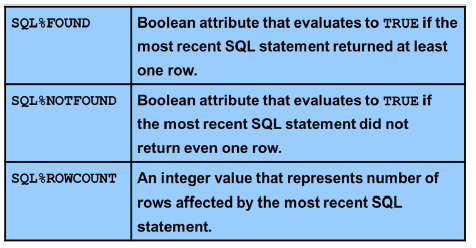
Je kan dit natuurlijk ook automatisch doen door dit commando in LOGIN.SQL te plaatsen

Een voorbeeld zonder parameters

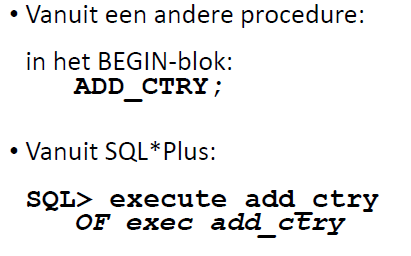




Om dit op te vangen kan je gebruik maken van sommige van volgende functies.

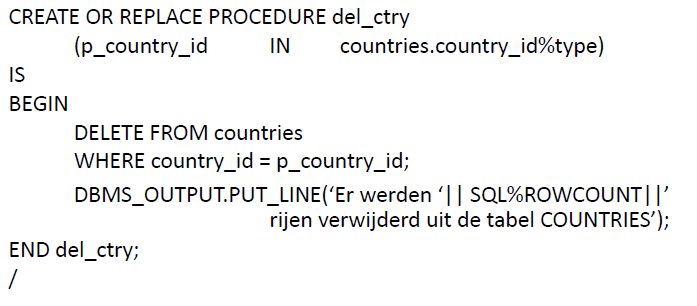


Je kan ook een prdcedure aanroepen in een andere precedure of gewoon in sql plus

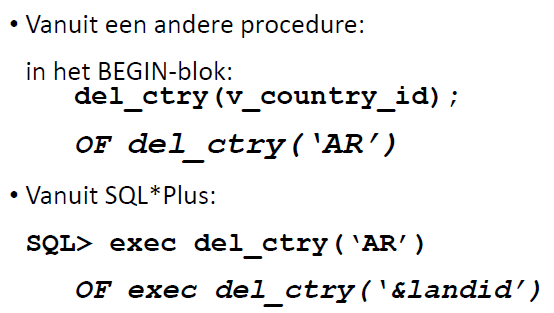


## IN

Als je in een procedure een parameter wilt gebruiken kan je dit gebruiken doen door een parameter mee te geven en het woord IN. Bv.

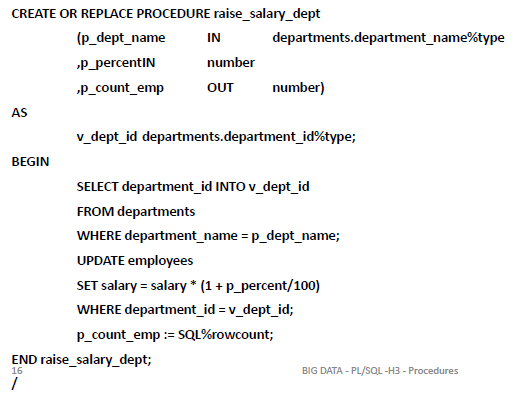


Je kan dan later of in het begin blok of in sql plus direct deze procedure oproepen. Hier geef je dan ook een waarde mee voor deze parameter die je in de procedure geschreven hebt.

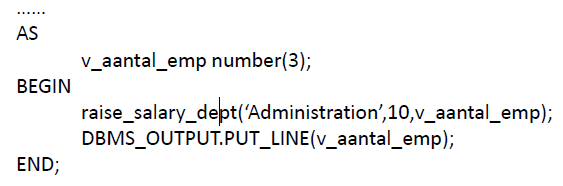


## OUT

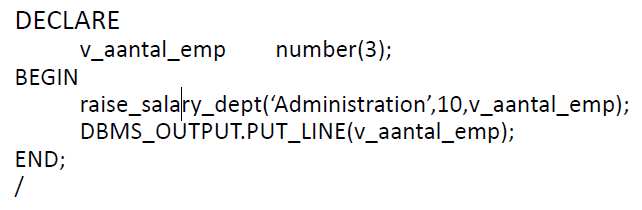
Het in woord gebruik je in sql om iets in te geven in het begin van je functie. Nu kan je ook het woord OUT gebruiken. Dit spreekt redelijk voor zich zelf en is om een woord te outputen op het einde van je procedure.



Hieronder is een voorbeeld dat een andere procedure uitgevoerd word in een andere procedure.

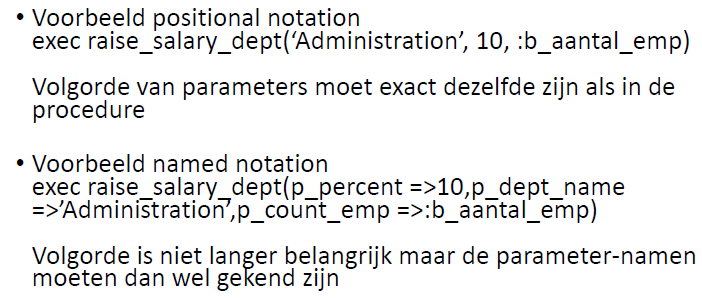


Je kan dit natuurlijk ook doen via een anoniem blok bv.



## Notations

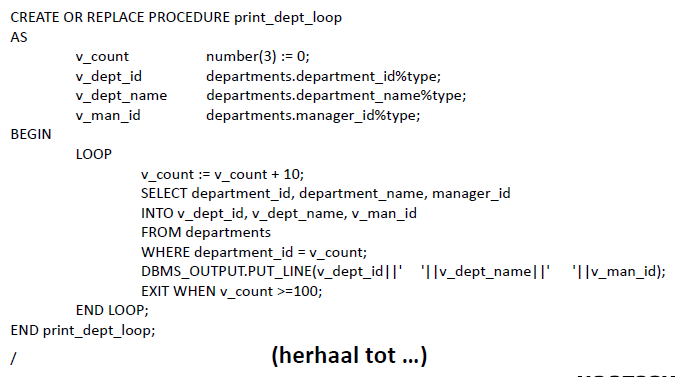
Je kan een procedure op verschillende manieren je informatie meegeven. Je kan dit doen zoals je al lang gewoon ben in de volgorde zoals je die hebt ingegeven of je kan dit doen door named notation. (persoon vind ik dit overbodig om dat je het al lang gewoon bent om in de juiste volgorde te zetten en het maakt je meer typen alleen de volgorde speelt hier geen rol maar kost natuurlijk meer tijd om te typen)



## LOOP

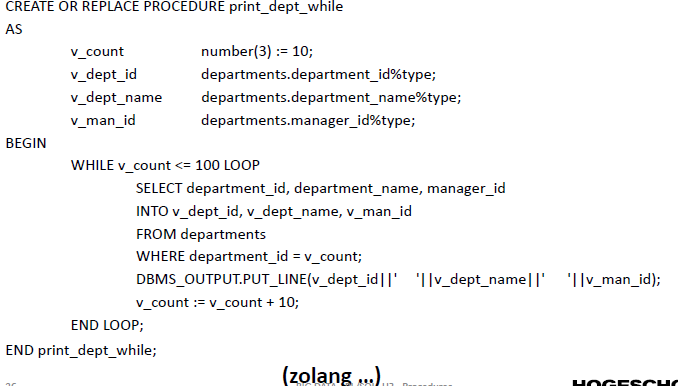
Net als in veel andere programeer talen kan je ook hier met een loop werken. Je kan dit best vergelijken met do while. Je start dit met LOOP en eindigt dit met EXIT WHEN varialbenaam >=< of zo iets dan iets ; en dan END LOOP;

Bv.



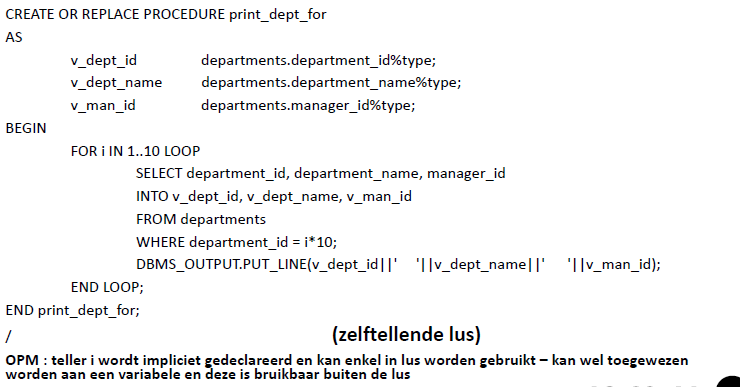
## WHILE

Hetzelfde als bovenstaande voorbeeld maar in plaats van dat er eerst iets gedaan word en dan pas gekeken word of er al aan de voorwaarde voldaan is. Met een while word eerst gekeken of er al aan de voorwaarde voldaan is als dit zo is gaat er niet in de blok anders wel tot er aan voldaan is.



## FOR

Je kan natuurlijk ook een for lus gebruiken dit werkt zoals je het kent in andere programeer talen. Dit telt zelf elke keer plus een van het begin getal dat je hebt ingegeven tot aan het laaste getal dat je hebt ingegeven.



Als je wilt dat je lus met twee verhoogt of met een ander getal dan met een moet je dit doen door ipv gewoon i te pakken kan je dan doen door (i+(i+1)) voor als je i wilt verhogen met twee. Als je dit wilt doen voor bv 3 dan doe je i+i+i+1. Natuurlijk pas je i niet echt aan maar haal je gewoon data uit i en steek dit in een variabel. Natuurlijk moet je dan wel bij de lus ook wat aanpassen. Je moet dit beginnen met BEGINGETAL-1..(AANTALKEREN/AANTALPERSPRONG JE WILT)-1.

Dit zou dus willen zeggen dat het zelfde voorbeeld zo word

CREATE OR REPLACE PROCEDURE print\_dept\_for

AS

v\_dept\_id departments.department\_id%type;

v\_dept\_name departments.department\_name%type;

v\_man\_id departments.manager\_id%type;

BEGIN

FOR i IN 0..4 LOOP

SELECT department\_id, department\_name, manager\_id

INTO v\_dept\_id, v\_dept\_name, v\_man\_id

FROM departments

WHERE department\_id = (i+i+1)\*10;

DBMS\_OUTPUT.PUT\_LINE(v\_dept\_id||' '||v\_dept\_name||' '||v\_man\_id);

END LOOP;

END print\_dept\_for;

/

## Alaternatieve FOR

CREATE OR REPLACE PROCEDURE print\_dept\_cursorloop

AS

BEGIN

FOR rec IN (SELECT department\_id, department\_name, manager\_id

FROM departments

WHERE department\_id between 10 and 100)

LOOP

DBMS\_OUTPUT.PUT\_LINE(rec.department\_id||'

'||rec.department\_name||' '||rec.manager\_id);

END LOOP;

END print\_dept\_cursorloop;

/

## Procedures verwijderen

Net als bij functies kan je ook een procedure verwijderen. Dit doe je door bv.

DROP PROCEDURE raise\_salary\_dept;

## Opvragen kenmerken

Net als bij functies kan je ook dit doen met een procedure. Dit doe je door

**SELECT object\_name**

**FROM user\_objects**

**WHERE object\_type = ‘PROCEDURE';**

Zo kan je alle Prcedures opvragen.

Als je de fucntie van een proceduere wilt opvragen kan je dit doen door dit commando.

SELECT text

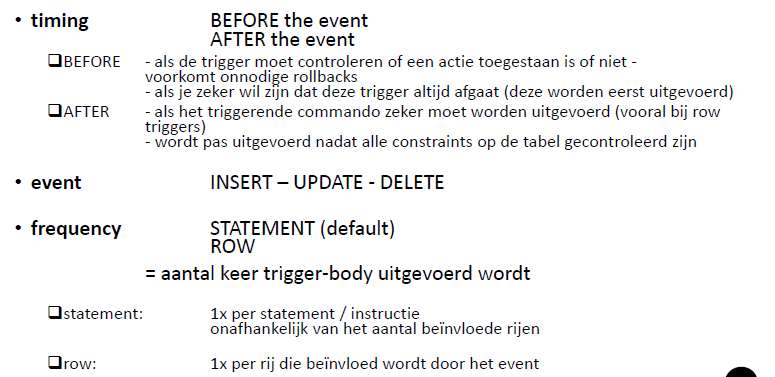
FROM user\_source

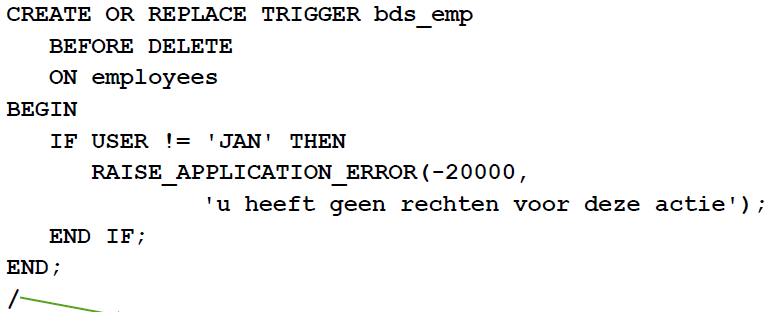
WHERE name = ‘RAISE\_SALARY\_DEPT';

Dit moet altijd met hoofdletters.

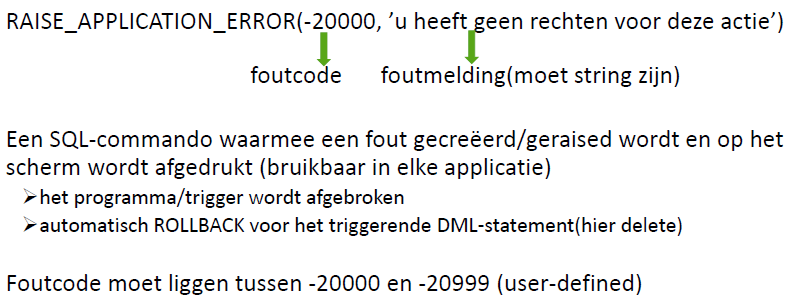
# Database triggers

Als je een trigger maakt dan word er een timing een event en hoevaak deze dit moet uitgevoerd worden bv.





## Foutmelding



## Naamgeving

B=before

A=after

I=insert

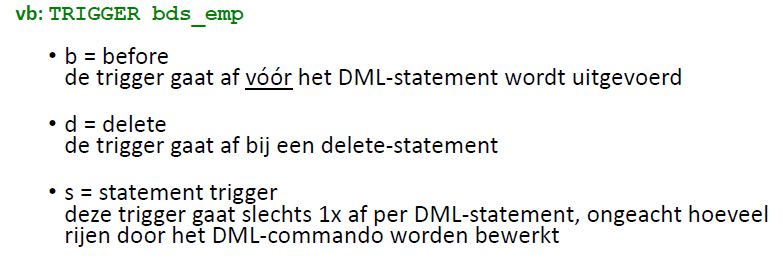
U=Update

D=Delete

S=Statement

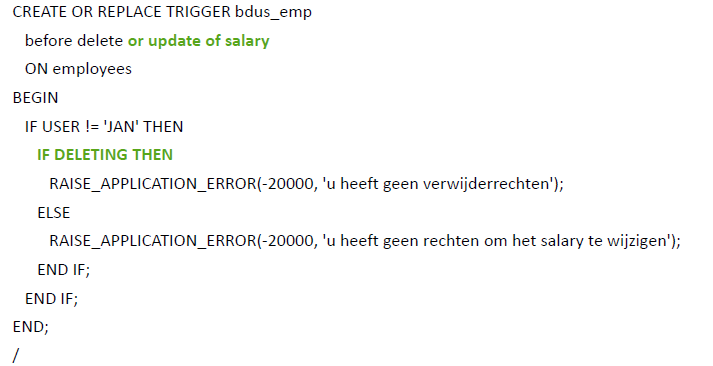
R=Row

Het eerste en het laatste kan je maar een nemen. Maar van de middelste blok kan je meer tegelijk in een trigger.



## Multi statement trigger

Je kan natuurlijk ook meerdere triggers in een schrijven dit kan je doen door bv.



Zoals je hierboven ziet staat hier DELETING. Er zijn 3 van deze functies:

INSERTING

DELETING

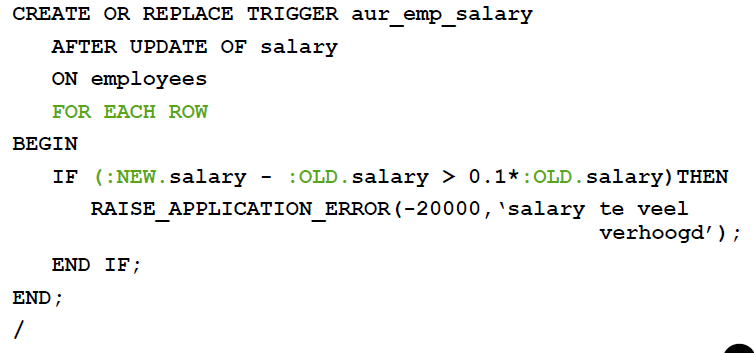
UPDATING

In dit laaste geval kan er ook nog een extra parameter dus bv.

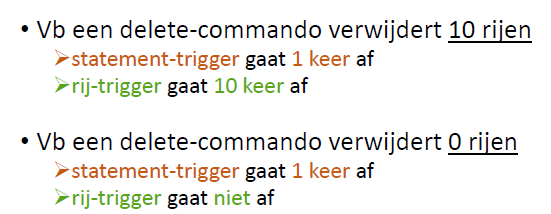
IF updating(‘salary’) THEN

## Row Trigger

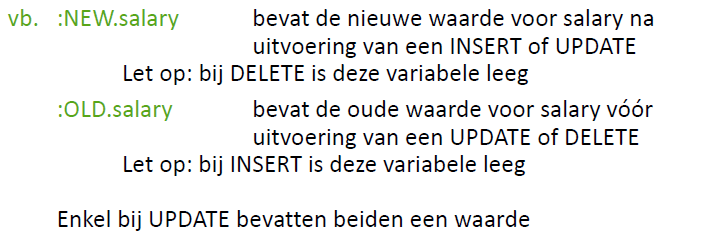
In plaats van een statment kan je ook een row trigger hebben bv.



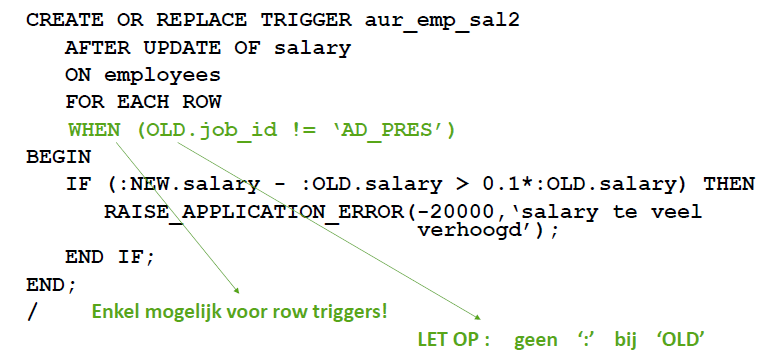
Het grote verschil tussen een statement trigger en een rij trigger is dat een statement 1 keer runt en een rij trigger zo vaak triggert als er rijen worden gemanipuleerd(d/u/i) bv.



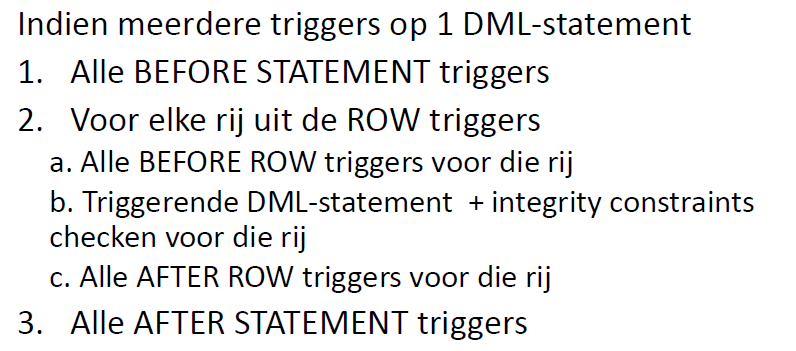
Hier kan je ook de keywords :NEW.kolomnaam en :OLD.kolomnaam dit spreekt denk ik wel redelijk voor zich zelf.



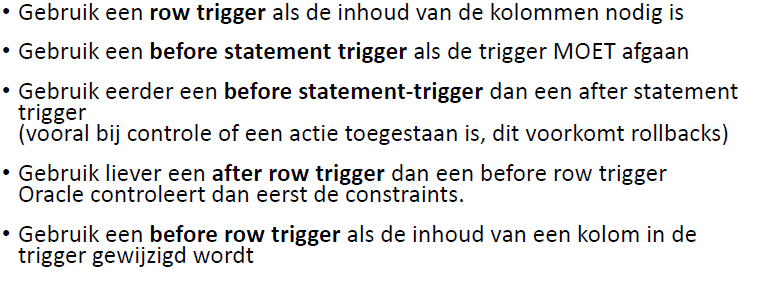
Je kan ook dit uitbreiden met een when. Hierboven zeg je gewoon op elke lijn. Maar je kan ook het woord WHEN gebruiken. Dus stel waar de vorige job\_id niet gelijk is aan iets dan voer je iets uit dus zoals in dit voorbeeld.



## Volgorde van triggers



## Welke trigger moet je kiezen in welke situatie



## Opvragen kenmerken

Net als bij functies en Procedures kan je ook dit doen met een trigger. Dit doe je door

**SELECT object\_name, created, status**

**FROM user\_objects**

**WHERE object\_type = ‘TRIGGER';**

Zo kan je alle Triggers opvragen en kijken of deze actief staan.

Als je de fucntie van een trigger wilt opvragen kan je dit doen door dit commando.

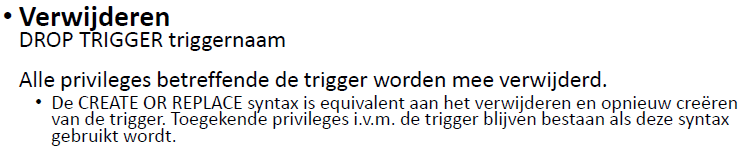
SELECT line,text

FROM user\_source

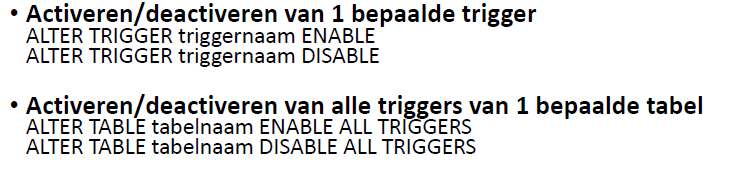
WHERE name = ‘AUR\_EMP\_SALARY';

Dit moet altijd met hoofdletters.

## Verwijderen van een trigger



## Activeren/deactiveren van triggers



Natuurlijk standaard staan triggers aan.

Een trigger kan ook invalid worden als de structuur van een databas aangepast word. Dit pas je dan zo aan.

