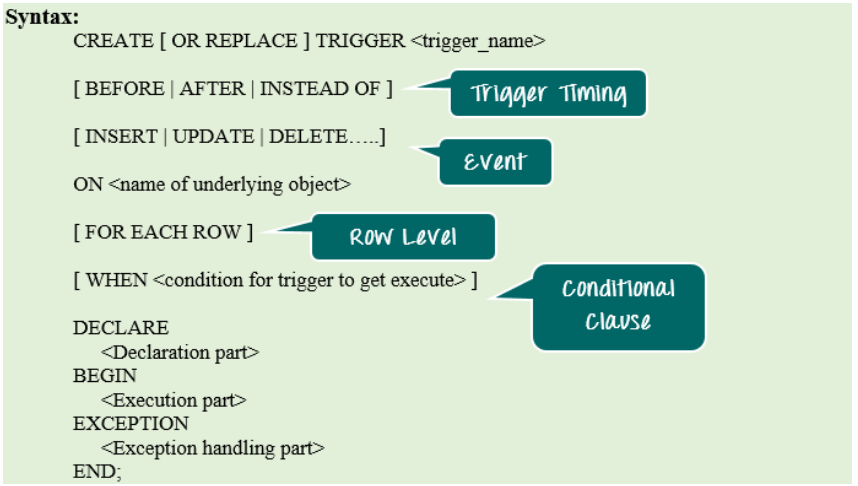
Triggers

Definició	Format

Un **trigger** és un bloc de PL/SQL amb nom emmagatzemat a la BD, aquest bloc de PL/SQL s'executa de manera automàtica quan es produeix **l'esdeveniment** que el dispara. Aquests esdeveniments poden ser:
• A data manipulation language (DML) statement executed against a table e.g., INSERT, UPDATE, or DELETE.
For example, if you define a trigger that fires before an INSERT statement on the customers table, the trigger will fire once before a new row is inserted into the customers table.
• A data definition language (DDL) statement executes e.g., CREATE or ALTER statement. These triggers are often used for auditing purposes to record changes of the schema.
• A system event such as startup or shutdown of the Oracle Database.
• A user event such as login or logout.
	A continuació la sintaxis de la creació d'un trigger amb les seves diferents opcions:
	`CREATE [OR REPLACE] TRIGGER trigger_name`
	`{BEFORE | AFTER} dml_event ON table_name`
	`[FOR EACH ROW]`
	`[DECLARE variables]`
	`BEGIN`
	`pl_sql_code`
	`[EXCEPTION exception_code]`
	`END;`
	 

Definició	Format
Hi ha un tipus particular de trigger el **Compound trigger**. Es un trigger que permet especificar accions per a cadascun dels quatre nivells temporals que poden provocar que es dispari un trigger. Els quatre punts temporals que podem definir sobre actuacions que es fan a la BD són: •BEFORE STATEMENT •BEFORE ROW •AFTER ROW •AFTER STATEMENT The compound trigger is useful when you want to accumulate facts that characterise the “for each row” changes and then act on them as a body at “after statement” time. Two popular reasons to use compound trigger are: 1.To accumulate rows for bulk-insertion. 2.To avoid the infamous ORA-04091: mutating-table error. Compound Triggers in Oracle	CREATE OR REPLACE TRIGGER compound_trigger_name FOR [INSERT|DELETE]UPDATE [OF column] ON <table> -- Declarative Section (optional) -- Variables declared here have firing-statement duration. --Executed before DML statement BEFORE STATEMENT IS BEGIN NULL; END BEFORE STATEMENT; --Executed before each row change- :NEW, :OLD available BEFORE EACH ROW IS BEGIN NULL; END BEFORE EACH ROW; --Executed aftereach row change- :NEW, :OLD available AFTER EACH ROW IS BEGIN NULL; END AFTER EACH ROW; --Executed after DML statement AFTER STATEMENT IS BEGIN NULL; END AFTER STATEMENT; END compound_trigger_name;

Definició	Format
Un **trigger** ORACLE és útil en molts casos com per exemple: • Ajudar al compliment de regles de negoci que no poden ser implementades mitjançant restriccions, *constraints*, com : ◦ UNIQUE, NOT NULL, CHECK. • Prevenir transaccions no permeses. Per exemple: no es poden fer modificacions fora d'horari, només de 8 a 15h. • Recollir informació estadística de l'accés a taules i modificació de dades de la BD. Per exemple: registrar quin usuari ha modificat una dada. • Generació d'informació per columnes que són derivades d'altres. Per exemple columnes de total ... • Auditar dades sensibles. Per exemple: guardar imatges d'una fila abans de modificar-la.	

Paràmetres d'un trigger	Sintaxis

Cadascun dels paràmetres que podem especificar en la creació d'un trigger, del tipus que es mostra a la dreta, es descriu a continuació:
• trigger_name: el nom amb que es guardarà l'objecte a la BD.
• BEFORE|AFTER: indica que volem que el codi del trigger s'executi abans o després de la sentència d'actualització de BD que el dispara.
• dml_event: El tipus de sentència DML que el dispararà , els tipus poden ser: INSERT, UPDATE, o DELETE.
• table_name: El nom de la taula associada al trigger, es a dir quan es facin canvis a la taula provocarà que es dispari el trigger.
• FOR EACH ROW: Això vol dir que el trigger s'executarà per a cada fila afectada per la sentència ex: si un update modifica 5 files , el trigger s'executarà per a cadascuna d'aquestes files. Si no s'indica que es a nivell de fila llavors només s'executa una vegada per sentència.
• DECLARE: En aquesta secció es poden definir variables que es faran servir dins el bloc de pl_sql_code
• pl_sql_code: Aquest es el cos del trigger i on s'indiquen les accions que es volen dur a terme.
• EXCEPTION exception_code: També pot haver-hi un bloc de d'excepcions si es necessari fer gestió d'excepcions.
El trigger name ha de ser únic, dins els objectes de tipus trigger, del mateix esquema. Un trigger es pot dir igual que un procedure, però es recomanable no repetir noms encara que es tracti de tipus d'objectes diferents.
	Els triggers relacionats amb actualitzacions de dades d'una taula són els més habituals.
	El codi per crea un trigger d'aquest tipus , contempla els següents conceptes:
	`CREATE [OR REPLACE] TRIGGER trigger_name`
	`{BEFORE | AFTER} dml_event ON table_name`
	`[FOR EACH ROW]`
	`[DECLARE variables]`
	`BEGIN`
	`pl_sql_code`
	`[EXCEPTION exception_code]`
	`END;`

Clàusula :NEW and :OLD

En un trigger a nivell de fila , el trigger es dispara per a cada fila que compleixi la condició. En aquests casos a vegades cal saber quins eren els valors abans i després de la sentència que ha disparat el trigger.

Per saber els valors abans i després, ORACLE proporciona les clàusules:

- :NEW i :OLD

Aquestes clàusules només es poden fer servir en el codi d'un trigger, serveixen per:

- :NEW – conté el nou valor que prendrà una columna d'una taula una vegada es completar la sentència que ha disparat el trigger.
- :OLD – conté el valor que té una columna d'una taula abans de completar la sentència que ha disparat el trigger.

El camp :NEW i el camp :OLD poden ser iguals si l'actualització que anem a fer a la BD no canvia el valor de la columna.

Hi ha restriccions a l'aplicació d'aquesta clàusula en funció de quin tipus de sentència estem executant:

	INSERT	UPDATE	DELETE
:NEW	VALID	VALID	INVALID, no hi ha new value quan s'esborra
:OLD	INVALID, no hi ha old value quan es fa insert	VALID	VALID

Exemples

Actualitzar les hores que ha dedicat un treballador a un projecte:

```
CREATE OR REPLACE TRIGGER comptarHoresProj
after UPDATE ON participar
for EACH ROW
BEGIN
    update projecte
    set hores= hores + (:new.num_hores - :old.num_hores)
    where num_projecte = :new.num_projecte;
END comptarHoresProj;
```

En aquest exemple suposem que hi ha les taules

- PARTICIPAR, on es guarda quantes hores treballa una persona en un projecte
- PROJECTE, on es manté el total de les hores dedicades a un projecte

quan algú modifica la participació cal actualitzar les hores totals del projecte.

Clàusula :NEW and :OLD	Exemples

	Reflectir en un històric de sous, els canvis de sou de cada treballador:
	
	`CREATE OR REPLACE TRIGGER guardarSous`
	`before UPDATE ON employee`
	`for EACH ROW`
	`BEGIN`
	`if :old.salary <> :new.salary then`
	`insert into salary_his`
	`values (:old.employee_id,:old.salary, sysdate);`
	`end if;`
	`END guardarSous;`
	
	En aquest exemple podem veure que només s'executa una operació quan els valors :NEW i :OLD són diferents.

Si més d'un tipus d'operació DML pot activar un trigger (per exemple, ON INSERT OR DELETE OR UPDATE OF ...), en el cos del trigger es poden utilitzar els següents predicats condicionals:
• INSERTING
• DELETING
• UPDATING
per comprovar quin tipus de declaració activa el trigger.
	`create or replace trigger histproveedor_tr after`
	`insert or update or delete on proveedor for each row`
	`declare`
	`idprov proveedor.clvprov%type;`
	`nombre proveedor.nombprov%type;`
	`operac hist_proveedor.tpop%type;`
	`begin`
	`if inserting then`
	`operac := 'I';`
	`idprov := :new.clvprov;`
	`nombre := :new.nombprov;`
	`elsif deleting then`
	`operac := 'D';`
	`idprov := :old.clvprov;`
	`nombre := :old.nombprov;`
	`elsif updating then`
	`operac := 'U';`

Clàusula :NEW and :OLD	Exemples

	idprov := :new.clvprov;
	nombre := :new.nombprov;
	end if;
	insert into hist_proveedor values (
	idhistopsq.nextval, idprov, nombre,
	sysdate, user, operac);
	end histpiezas_tr;
	
	En aquest exemple podem veure que segons l'opció es posen uns valors o uns altres al registre d'històric que es guarda cada vegada que es fa una modificació.

Webgrafia

Enllaços web	
.../oracle-triggers/	Oracle Triggers – The Complete Guide
inserting, deleting, updating	DML Trigger Conditional Predicates: INSERTING, UPDATING, DELETING