# Bases de Datos Activas

## Resumen

# Bases de Datos Activas

- **Objetivo**
  - El uso de los disparadores en los DBMS
  - Definición y características de los disparadores
  - Los disparadores en Oracle 10g

# Bases de Datos Activas

- **Modelo ECA**
  - Las reglas que especifican acciones que son activadas de forma automática por determinados eventos han sido consideradas en las bases de datos, principalmente, para la definición de restricciones (constraints)
  - El modelo que se ha empleado para especificar las reglas de las bases de datos activas se denomina:
    - Evento-Condición-Acción (**ECA**)
  - Estas reglas se denominan, generalmente, disparadores "Trigger"
    - Su finalidad es convertir una base de datos "pasiva" en "activa"

# Bases de Datos Activas

- **Modelo ECA**
  - **Evento**
    - Generalmente son operaciones de actualización de la base de datos que se aplican explícitamente a la base de datos
      - En un amplio sentido podrían considerarse los eventos temporales (consideración del tiempo en la activación de la regla)

# Bases de Datos Activas

- **Modelo ECA**
  - **Condición**
    - Declaración que determina si la regla debe o no ejecutarse
      - Una vez que se ha producido el evento activador puede evaluarse una condición opcional
      - Si no se especifica una condición, la acción se ejecuta una vez que se produzca el evento
      - Si se especifica una condición, primero se evalúa ésta, y si su evaluación es verdadera se ejecuta la acción asociada a la regla

# Bases de Datos Activas

- **Modelo ECA**
  - **Acción**
    - Operación que se realiza cuando se evalúa afirmativamente la regla
      - Suele ser un procedimiento SQL, aunque puede ser una función en otro lenguaje, programa externo, etc.

# Bases de Datos Activas

- **Características de los disparadores**
  - Son reglas simples
  - No consideran un orden de evaluación
  - No incluyen conocimiento
  - No son escalables
  - Un número excesivo genera confusión
  - Pueden generarse ciclos de ejecución
  - Los eventos están limitados a INSERT, UPDATE y DELETE
  - Se pueden definir a dos niveles: filas y sentencia
  - No soportan la consideración del tiempo
  - Sobrecargan al gestor de bases de datos
  - Se ejecutan antes, después o en lugar de el evento
  - No son estándar para diferentes DBMS
  - No existe conocimiento externo de su ejecución
  - Complejidad para diseñar y verificar la consistencia
  - Pueden invocar a otros procedimientos y disparar otros *triggers*, pero no admiten parámetros y no pueden ser invocados desde otros procedimientos.

# Bases de Datos Activas

- **Tipos de los Disparadores**
  - Auto-Generados
    - Son generados de forma automática por el gestor de la base de datos cuando se define el esquema
      - Restricciones de dominio
      - Restricciones de clave
      - Restricciones de referencia
      - Alertas
      - Otros
  - Generados por el DBA
    - Correspondientes a las reglas de negocio

# Bases de Datos Activas

- **Uso de los disparadores**
  - Cuando los datos de una tabla son generados desde otro tipo de procedimientos y se necesita controlar los valores que toman algunos campos determinados de la tabla en cuestión.
  - Para duplicar los contenidos de una tabla automáticamente y en tiempo real.
  - Para implementar complejas restricciones sobre los valores que pueden tomar los campos de una tabla Oracle, es decir, cuando los *CONSTRAINTS* que se pueden definir sobre una tabla son insuficientes.
  - Para controlar las modificaciones de los valores de los campos de una tabla (auditorías).
  - Para incrementar automáticamente los valores de un campo.
  - Para realizar actualizaciones de una tabla en cascada.
  - Para modificar campos o registros de una tabla que un usuario no puede modificar directamente.

# Bases de Datos Activas

- **Restricciones de los disparadores**
  - Un disparador no puede emitir ninguna orden de control de transacciones: COMMIT, ROLLBACK o SAVEPOINT. El disparador se activa como parte de la ejecución de la orden que provocó el disparo, y forma parte de la misma transacción que dicha orden.
  - Por razones idénticas, ningún procedimiento o función llamado por el disparador puede emitir órdenes de control de transacciones.
  - El cuerpo del disparador no puede contener ninguna declaración de variables LONG o LONG RAW.
  - Existen restricciones acerca de a qué tablas puede acceder el cuerpo de un disparador. Dependiendo del tipo de disparador y de las restricciones que afecten a las tablas.
  - INSTEAD OF es una cláusula válida solo para vistas; no se puede especificar un disparador INSTEAD OF en una tabla. Si una vista tiene un disparador INSTEAD OF, cualquier vista creada sobre ésta debe tener a su vez un disparador INSTEAD OF.

# Bases de Datos Activas

**-- Formato general de los Disparadores en Oracle 10g**

```
create [or replace] trigger [schema .] trigger
{ [FOLLOWS nombre-otro-trigger] }
{ before | after | instead of }
{ dml_event_clause
        | { ddl_event [or ddl_event]...
        | database_event [or database_event]…}
on { [schema .] schema | database }
[referencing_clause] [for each row | for each statement]

[when ( condition ) ]
{ pl/sql_block | call_procedure_statement }

-- La sintaxis depende del tipo de trigger

{ delete | insert | update [of column [, column]...] }
[or { delete | insert | update [of column [, column]...] }]...
on { [schema .] table | [nested table nested_table_column of]
[schema .] view }
```

# Bases de Datos Activas

## -- Un ejemplo de trigger

```
drop table BOOKSHELF_AUDIT;
create table BOOKSHELF_AUDIT
(Title VARCHAR2(100), Publisher VARCHAR2(20), CategoryName VARCHAR2(20),
Old_Rating VARCHAR2(2), New_Rating VARCHAR2(2), Audit_Date DATE);

create or replace trigger BOOKSHELF_BEF_UPD_ROW
         before update on BOOKSHELF
         for each row
         when (new.Rating < old.Rating)
begin
         insert into BOOKSHELF_AUDIT
         (Title, Publisher, CategoryName,
         Old_Rating, New_Rating, Audit_Date)
         values
                  (:old.Title, :old.Publisher, :old.CategoryName,
                  :old.Rating, :new.Rating, Sysdate);
end;
/
```

# Bases de Datos Activas

## -- Un ejemplo de trigger

```
drop trigger BOOKSHELF_BEF_UPD_ROW;
create or replace trigger BOOKSHELF_BEF_UPD_INS_ROW
          before insert or update of Rating on BOOKSHELF
          for each row
begin
          if INSERTING then
                    insert into BOOKSHELF_AUDIT
                    (Title, Publisher, CategoryName, New_Rating, Audit_Date)
                    values
                    (:new.Title, :new.Publisher, :new.CategoryName,:new.Rating,
                    Sysdate);
          else -- if not inserting then we are updating the Rating
                    insert into BOOKSHELF_AUDIT
                    (Title, Publisher, CategoryName, Old_Rating, New_Rating,
                    Audit_Date)
                    values
                    (:old.Title, :old.Publisher, :old.CategoryName,
                    :old.Rating, :new.Rating, Sysdate);
          end if;
end; /
```

# Bases de Datos Activas

-- Un ejemplo de trigger

```
alter table BOOKSHELF_CHECKOUT
       add (UpperName VARCHAR2(25));

create or replace trigger BOOKSHELF_CHECKOUT_BUI_ROW
before insert or update of Name on BOOKSHELF_CHECKOUT
for each row
begin
       :new.UpperName := UPPER(:new.Name);
end;
/
```

# Bases de Datos Activas

## -- Un ejemplo de trigger

```
create or replace trigger BOOKSHELF_BEF_DEL
          before delete on BOOKSHELF
declare
          weekend_error EXCEPTION;
          not_library_user EXCEPTION;
begin
          if TO_CHAR(SysDate,'DY') = 'SAT' or
                    TO_CHAR(SysDate,'DY') = 'SUN' THEN
                    RAISE weekend_error;
          end if;
          if SUBSTR(User,1,3) <> 'LIB' THEN
                    RAISE not_library_user;
          end if;
          EXCEPTION
          WHEN weekend_error THEN
                    RAISE_APPLICATION_ERROR (-20001, Borrado no permitido');
          WHEN not_library_user THEN
                    RAISE_APPLICATION_ERROR (-20002,
                    Borrado permitido para usuarios bibliotecarios');
end; /
```

# Bases de Datos Activas

-- Llamando a procedimientos en los triggers

```
create or replace trigger BOOKSHELF_AFT_INS_ROW
       after insert on BOOKSHELF_AUDIT
       for each row
begin
       call INSERT_BOOKSHELF_AUDIT_DUP(:new.Title,
       :new.Publisher, :new.CategoryName,
       :new.Old_Rating, :new.New_Rating,
       :new.Audit_Date);
end;
/
```

# Bases de Datos Activas

## -- Triggers relacionados con el Esquema

```
create or replace trigger PREVENT_DROP
       before drop on Practice.schema
begin
       if ora_dict_obj_owner = 'PRACTICE'
              and ora_dict_obj_name like 'BOO%'
              and ora_dict_obj_type = 'TABLE'
       then
              RAISE_APPLICATION_ERROR (
              -20002, 'Operación no permitida.');
       end if;
end;
/
```

# Bases de Datos Activas

```
-- Habilitando y deshabilitando triggers

alter trigger BOOKSHELF_BEF_UPD_INS_ROW enable;

alter table BOOKSHELF enable all triggers;

alter trigger BOOKSHELF_BEF_UPD_INS_ROW disable;

alter table BOOKSHELF disable all triggers;

-- Borrando triggers
drop trigger BOOKSHELF_BEF_UPD_INS_ROW;
```

# Bases de Datos Activas

Atributos predefinidos por el sistema para los sucesos

| Attribute | Type | Description | Example |
|---|---|---|---|
| ora_client_ip_address | VARCHAR2 | Returns the IP address of the client in a LOGON event when the underlying protocol is TCP/IP. | `if (ora_sysevent = 'LOGON') then addr := ora_client_ip_address; end if;` |
| ora_database_name | VARCHAR2(50) | Database name. | `Declare db_name VARCHAR2(50); begin db_name := ora_database_name; end;` |
| ora_des_encrypted_password | VARCHAR2 | The DES encrypted password of the user being created or altered. | `if (ora_dict_obj_type = 'USER') then insert into event_table (ora_des_encrypted_password); end if;` |
| ora_dict_obj_name | VARCHAR(30) | Name of the dictionary object on which the DDL operation occurred. | `insert into event_table ('Changed object is ' \|\| ora_dict_obj_name');` |
| ora_dict_obj_name_list (name_list OUT ora_name_list_t) | BINARY_INTEGER | Returns the list of object names of objects being modified in the event. | `if (ora_sysevent = 'ASSOCIATE STATISTICS') then number_modified := ora_dict_obj_name_list (name_list); end if;` |
| ora_dict_obj_owner | VARCHAR(30) | Owner of the dictionary object on which the DDL operation occurred. | `insert into event_table ('object owner is' \|\| ora_dict_obj_owner');` |
| ora_dict_obj_owner_list(owner_list OUT ora_name_list_t) | BINARY_INTEGER | Returns the list of object owners of objects being modified in the event. | `if (ora_sysevent = 'ASSOCIATE STATISTICS') then number_of_modified_objects := ora_dict_obj_owner_list(owner_list); end if;` |

# Bases de Datos Activas

| Attribute | Type | Description | Example |
|---|---|---|---|
| ora_dict_obj_type | VARCHAR(20) | Type of the dictionary object on which the DDL operation occurred. | `insert into event_table ('This object is a ' \|\| ora_dict_obj_type);` |
| ora_grantee( user_list OUT ora_name_list_t) | BINARY_INTEGER | Returns the grantees of a grant event in the OUT parameter; returns the number of grantees in the return value. | `if (ora_sysevent = 'GRANT') then number_of_users := ora_grantee(user_list); end if;` |
| ora_instance_num | NUMBER | Instance number. | `if (ora_instance_num = 1) then insert into event_table ('1'); end if;` |
| ora_is_alter_column( column_name IN VARCHAR2) | BOOLEAN | Returns true if the specified column is altered. | `if (ora_sysevent = 'ALTER' and ora_dict_obj_type = 'TABLE') then alter_column := ora_is_alter_column('FOO'); end if;` |
| ora_is_creating_nested_table | BOOLEAN | Returns true if the current event is creating a nested table. | `if (ora_sysevent = 'CREATE' and ora_dict_obj_type = 'TABLE' and ora_is_creating_nested_table) then insert into event_tab values ('A nested table is created'); end if;` |
| ora_is_drop_column( column_name IN VARCHAR2) | BOOLEAN | Returns true if the specified column is dropped. | `if (ora_sysevent = 'ALTER' and ora_dict_obj_type = 'TABLE') then drop_column := ora_is_drop_column('FOO'); end if;` |
| ora_is_servererror | BOOLEAN | Returns true if given error is on error stack, FALSE otherwise. | `if (ora_is_servererror(error_number)) then insert into event_table ('Server error!!'); end if;` |
| ora_login_user | VARCHAR2(30) | Login username. | `select ora_login_user from dual;` |
| ora_partition_pos | BINARY_INTEGER | In an INSTEAD OF trigger for CREATE TABLE, the position within the SQL text where you could insert a PARTITION clause. | `-- Retrieve ora_sql_txt into -- sql_text variable first.`<br>`n := ora_partition_pos; new_stmt := substr(sql_text, 1, n-1) \|\| ' ' \|\| my_partition_clause \|\| ' ' \|\| substr(sql_text, n));` |

| Attribute | Type | Description | Example |
|---|---|---|---|
| ora_privilege_list( privilege_list OUT ora_name_list_t) | BINARY_INTEGER | Returns the list of privileges being granted by the grantee or the list of privileges revoked from the revokees in the OUT parameter; returns the number of privileges in the return value. | `if (ora_sysevent = 'GRANT' or ora_sysevent = 'REVOKE') then number_of_privileges := ora_privilege_list(priv_list); end if;` |
| ora_revokee ( user_list OUT ora_name_list_t) | BINARY_INTEGER | Returns the revokees of a revoke event in the OUT parameter; returns the number of revokees in the return value. | `if (ora_sysevent = 'REVOKE') then number_of_users := ora_revokee(user_list);` |
| ora_server_error | NUMBER | Given a position (1 for top of stack), it returns the error number at that position on error stack. | `insert into event_table ('top stack error ' \|\| ora_server_error(1));` |
| ora_server_error_depth | BINARY_INTEGER | Returns the total number of error messages on the error stack. | `n := ora_server_error_depth; -- This value is used with -- other functions such as -- ora_server_error` |
| ora_server_error_msg (position in binary_integer) | VARCHAR2 | Given a position (1 for top of stack), it returns the error message at that position on error stack. | `insert into event_table ('top stack error message' \|\| ora_server_error_msg(1));` |
| ora_server_error_num_params (position in binary_integer) | BINARY_INTEGER | Given a position (1 for top of stack), it returns the number of strings that have been substituted into the error message using a format like "%s". | `n := ora_server_error_num_params(1);` |
| ora_server_error_param (position in binary_integer, param in binary_integer) | VARCHAR2 | Given a position (1 for top of stack) and a parameter number, returns the matching "%s", "%d", and so on substitution value in the error message. | `-- E.g. the 2nd %s in a message -- like "Expected %s, found %s" param := ora_server_error_param(1,2);` |

# Bases de Datos Activas

Atributos predefinidos por el sistema para los sucesos

| Attribute | Type | Description | Example |
|---|---|---|---|
| ora_sql_txt (sql_text out ora_name_list_t) | BINARY_INTEGER | Returns the SQL text of the triggering statement in the OUT parameter. If the statement is long, it is broken up into multiple PL/SQL table elements. The function return value specifies how many elements are in the PL/SQL table. | `sql_text ora_name_list_t;`<br>`stmt VARCHAR2(2000);`<br>`...`<br>`n := ora_sql_txt(sql_text);`<br>`for i in 1..n loop`<br>`  stmt := stmt \|\| sql_text(i);`<br>`end loop;`<br>`insert into event_table ('text of triggering statement: ' \|\| stmt);` |
| ora_sysevent | VARCHAR2(20) | System event firing the trigger: Event name is same as that in the syntax. | `insert into event_table (ora_sysevent);` |
| ora_with_grant_option | BOOLEAN | Returns true if the privileges are granted with grant option. | `if (ora_sysevent = 'GRANT' and ora_with_grant_option = TRUE)`<br>`  then insert into event_table ('with grant option');`<br>`end if;` |
| space_error_info( error_number OUT NUMBER, error_type OUT VARCHAR2, object_owner OUT VARCHAR2, table_space_name OUT VARCHAR2, object_name OUT VARCHAR2, sub_object_name OUT VARCHAR2) | BOOLEAN | Returns true if the error is related to an out-of-space condition, and fills in the OUT parameters with information about the object that caused the error. | `if (space_error_info(eno, typ, owner, ts, obj, subobj) = TRUE) then.put_line('The object ' \|\| run out of space.');`<br>`dbms_outputobj \|\| ' owned by ' \|\| owner \|\| ' has`<br>`end if;` |