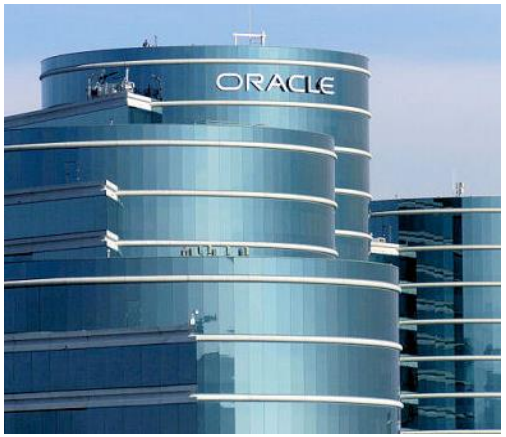
Tema 4

SQL: El lenguaje estándar de los SGBD comerciales

Grado en Ingeniería Informática



Bases de Datos

2021/22

Departamento de Tecnologías de la Información

Universidad de Huelva

SQL: El lenguaje estándar de los SGBD Relacionales

Objetivos

- Saber utilizar el lenguaje SQL para crear esquemas de bases de datos relacionales
- Saber utilizar, de forma eficiente, el lenguaje SQL para consultar bases de datos relacionales
- Conocer las sentencias SQL que permiten establecer y denegar permisos a un esquema relacional
- Aprender algunas extensiones del lenguaje SQL, concretamente el lenguaje de procedimientos PL/SQL de Oracle, para proporcionar mayor funcionalidad a las bases de datos

Contenidos

SQL: El lenguaje estándar de los SGBD Relacionales

Duración

- 2 clases

Bibliografía

- Capítulo 8, 9 y 10 [Elmasri 02]
- Capítulos 3, 4 y 5 [Silberschatz 07]
- Capítulos 5, 6 y 7 [Connolly 05]

LEER: ANEXO "INTRODUCCIÓN A PL/SQL"

SQL: El lenguaje estándar de los SGBD Relacionales

4.1 Introducción

- A comienzos de los años 70 empieza a surgir la necesidad de diseñar lenguajes que soporten el modelo relacional propuesto por Codd
- Se han instrumentado lenguajes puros de álgebra como el ISBL o de cálculo como el ALPHA y el QBE, pero ha sido el **SQL** (*Structured Query Language*) el lenguaje relacional que ha acabado imponiéndose
- La versión original de SQL se desarrolló en los laboratorios de investigación de IBM en San José (USA)
- Originariamente se denominó SEQUEL (*Structured English QUEry Language*)
- El primer SGBD comercial basado en SQL fue ORACLE
- A partir de éste, aparecieron numerosos productos que implementaban este lenguaje (DB2, Sybase, Informix, Ingres, Access, SQL Server, etc.)
- SQL es un lenguaje de base de datos global o completo
  - Cuenta con sentencias de definición, consulta y actualización de datos.
  - Así pues, es tanto un LDD como un LMD

SQL: El lenguaje estándar de los SGBD Relacionales

- Con idea de unificar criterios, las organizaciones **ISO** (*Internacional Standards Organization*) y **ANSI** (*American National Standards Institute*), están publicando normas desde 1986, para estandarizar el lenguaje SQL

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision that added integrity constraints, adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries (e.g. transitive closure), triggers, support for procedural and control-of-flow statements, non-scalar types (arrays), and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice versa (SQL/JRT).
2003	SQL:2003		Introduced XML-related features (SQL/XML), *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006		ISO/IEC 9075-14:2006 defines ways that SQL can be used with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database, and publishing both XML and conventional SQL-data in XML form. In addition, it lets applications integrate queries into their SQL code with XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.[28]
2008	SQL:2008		Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers, TRUNCATE statement,[29] FETCH clause.
2011	SQL:2011		Adds temporal data (PERIOD FOR)[30] (more information at: Temporal database#History). Enhancements for *window functions* and FETCH clause.[31]
2016	SQL:2016		Adds row pattern matching, polymorphic table functions, JSON.

Fuente: http://en.wikipedia.org/wiki/SQL

SQL: El lenguaje estándar de los SGBD Relacionales

- SQL, cuenta con mecanismos para:
  - Definir vistas de la base de datos
  - Crear y desechar índices
  - Incorporar sus sentencias en lenguajes alto nivel (anfitriones) como C++ o Java
  - Permitir o denegar permisos a los distintos usuarios (lenguaje de control)

4.2. El Lenguaje SQL

4.2.1 Lenguaje de Definición de Datos

La documentación sobre el Lenguaje de Definición de Datos de SQL se encuentra en la sección "Práctica I" disponible en la página web de la asignatura.

4.2.2 Lenguaje de Manipulación de Datos

SQL proporciona cuatro instrucciones para la manipulación de datos:

- SELECT: seleccionar — Extracción de datos
- UPDATE: actualizar
- DELETE: eliminar
- INSERT: insertar

— Instrucciones de Actualización

La parte del LMD relacionada con la extracción de los datos se suele denominar **lenguaje de consulta**

La documentación sobre las Instrucciones de Actualización y el Lenguaje de Consulta de SQL se encuentra en las secciones "Práctica I“ y “Práctica II”, disponibles en la página web de la asignatura.

4.2.3 Vistas

- En SQL, una **vista** es una tabla derivada de otras tablas, que pueden ser a su vez *tablas de base* o *vistas* previamente definidas

  create table

- A las vistas se les considera *tablas virtuales*, en contraste con las tablas de base, cuyas tuplas se almacenan físicamente en la BD. Este detalle **limita las operaciones de actualización** aplicables en una vista, aunque no limita las posibles consultas

- En el caso de tener que hacer frecuentes consultas sobre la reunión de varias tablas, se puede crear una vista con la reunión de éstas y consultar sobre una sola tabla

  concatenación

- Cada una de las tablas utilizadas para crear la vista se denominan *tablas de definición* de la vista

Especificación de vistas en SQL

- La instrucción en SQL para crear una vista es **CREATE VIEW**

SQL: El lenguaje estándar de los SGBD Relacionales

- Una vista estará formada por:
  - **Un nombre de vista**
  - **Una lista de nombres de atributos.** Si ningún atributo es resultado de aplicar funciones aritméticas, no hace falta especificar sus nombres, ya que serán los mismos que los de las tablas de definición
  - **Una consulta** para especificar el contenido de la vista

- Una vista siempre *está al día* (actualizada), o sea, que si modificamos las tuplas de las tablas base sobre las que está definida, la vista refleja automáticamente estos cambios

- Es responsabilidad del SGBD controlar que la vista esté al día

SQL: El lenguaje estándar de los SGBD Relacionales

Ejemplos

V1: CREATE VIEW *Alumno_ordenador*
AS SELECT A.nombre, O.tipo, O.lugar
FROM Alumno A INNER JOIN Ordenador O
ON A.ordenador = O.IdOrd;

V2: CREATE VIEW *Prof_Asignatura*
(Nombre_prof, Num_Asigs, Suma_creditos)
AS SELECT P.nombre, count(*), Sum (A.créditos)
FROM Asignatura A INNER JOIN Profesor P
ON A.prof = P.npr
GROUP BY p.nombre;

- En V1 no se especifican nuevos atributos, por lo que no es necesario detallar la lista de atributos de la vista
- En V2 es necesario detallar los atributos de la vista ya que algunos de ellos (Num_asigs y Suma_creditos) son calculados.

- Se pueden especificar consultas SQL en términos de una vista, como si se tratase de una tabla base. Por ejemplo se podría escribir:

```
SELECT nombre, lugar
FROM Alumno_ordenador
WHERE tipo='PC Prácticas';
```

- Si se deja de necesitar una vista se puede eliminar con **DROP VIEW**

```
DROP VIEW Alumno_Ordenador;
```

4.2.4 Lenguaje de Control

- En un sistema multiusuario, el SGBD debe proporcionar técnicas que permitan a ciertos usuarios o grupos tener acceso sólo a las partes de cada BD que sea necesario

- El administrador de BD se encarga de otorgar privilegios a los usuarios y clasificarlos de acuerdo a la política de la organización

- Para ello dispone de una cuenta privilegiada con la que puede realizar operaciones tales como:

  - **Creación de cuentas.** A cada usuario se le asigna una cuenta con una contraseña para poder tener acceso a la BD

  - **Concesión y revocación de privilegios.** Con esta operación, se pueden conceder o revocar privilegios a ciertas cuentas

- Los mecanismos de seguridad basados en la concesión de privilegios a los usuarios para acceder a los datos en un determinado modo (lectura, escritura, actualización) se denominan **discrecionales**.

SQL: El lenguaje estándar de los SGBD Relacionales

- Dentro de los mecanismos discrecionales, hay dos niveles de asignación de privilegios para acceder a una BD:
  - **Nivel de cuenta.** El administrador especifica los privilegios particulares que tiene cada cuenta, independientemente de las tablas de la BD. Ejemplos: permiso de creación de tablas, de conexión, etc.
  - **Nivel de tabla.** En este nivel se puede controlar los privilegios para tener acceso a cada tabla de la BD
- Para controlar la concesión y eliminación de privilegios de tabla, a cada tabla de una BD se le asigna un propietario (por defecto, es el usuario que la crea)
- Al propietario de la tabla se le conceden todos los privilegios y puede proporcionar privilegios, para cualquiera de sus tablas, a otros usuarios
- En SQL se pueden conceder los siguientes tipos de privilegios para cada relación R:

- **Privilegio SELECT (atributo).** Asigna a la cuenta el privilegio de obtener datos de R. En caso de especificar atributos, sólo se permitirá ver los especificados
- **Privilegio INSERT.** Asigna el privilegio de insertar tuplas en la relación R
- **Privilegio UPDATE (atributo).** Asigna el privilegio de modificar tuplas R. En caso de haber especificado uno/varios atributos, sólo se permitirá actualizar los especificados
- **Privilegio DELETE.** Asigna el privilegio de borrar las tuplas de la relación R
- **Privilegio REFERENCES.** Asigna el privilegio de hacer referencia a la relación R por medio de una clave ajena

SQL: El lenguaje estándar de los SGBD Relacionales

- La instrucción para asignar estos privilegios en SQL se denomina **GRANT**, y su sintaxis es:

GRANT {<lista de permisos>| ALL [PRIVILEGES]}
ON objeto
TO {usuario | PUBLIC}
[WITH GRANT OPTION]

- La cláusula *WITH GRANT OPTION* permite que el usuario que recibe los privilegios sobre la tabla también pueda concederlos a otros usuarios

- La instrucción para eliminar privilegios en SQL se denomina **REVOKE**, y su sintaxis es:

REVOKE {<lista de permisos>| ALL [PRIVILEGES]}
ON objeto
FROM {usuario | PUBLIC}

donde <lista de permisos> es:
{ DELETE | INSERT | REFERENCES | SELECT | UPDATE }

SQL: El lenguaje estándar de los SGBD Relacionales

Ejemplo

- El usuario **BD_125** es el propietario de la tabla Profesor, y quiere que todos los usuarios puedan consultar los datos, la sintaxis sería:

GRANT SELECT **ON** PROFESOR **TO** PUBLIC;

- Si después quiere quitar dicho privilegio a los usuarios **B_015** y **BD_234**, la sintaxis sería:

REVOKE SELECT **ON** PROFESOR **FROM** B_015, BD_234;

- Las **vistas** son en sí mismas un importante mecanismo de autorización discrecional
  - Por ejemplo, si el propietario de una determinada relación desea que otro usuario sólo tenga acceso a determinados atributos de esa relación, puede crear una vista que incluya sólo esos atributos y concederle el privilegio de SELECT para esa vista
  - Lo mismo se puede aplicar cuando sólo se desea limitar el acceso a ciertas tuplas de una relación. En este caso se puede crear una vista que seleccione sólo dichas tuplas, y posteriormente se concede el privilegio correspondiente

4.2.5 Procesamiento de transacciones

- Una **transacción** es un conjunto de sentencias SQL de manipulación de datos que conforman una unidad lógica de trabajo, que es procesada de manera atómica, lo que permite que se ejecuten operaciones complejas en la base de datos manteniendo la integridad. La transacción más simple en SQL es una única sentencia

- Por ejemplo, para hacer un traspaso de dinero entre cuentas bancarias, el incremento y decremento en las cuentas implicadas debe hacerse como un todo

- Las operaciones que conforman una transacción podrían ocasionar inconsistencia en los datos si no terminasen completamente

- Por tanto, es conveniente controlar que toda transacción termine correctamente y sus efectos sean efectivos en la base de datos, o que sea anulada y ninguna de sus operaciones quede almacenada

SQL: El lenguaje estándar de los SGBD Relacionales

- Las sentencias básicas de Oracle para gestionar las transacciones son: **COMMIT** para confirmar las operaciones de la transacción y **ROLLBACK** para anular las operaciones

- Se pueden establecer puntos intermedios en las operaciones mediante SAVEPOINT y anular las operaciones realizadas a partir de esos puntos

Sintaxis

```
COMMIT ;
```

```
SAVEPOINT nombre_savepoint ;
```

```
ROLLBACK [ TO nombre_savepoint ] ;
```

SQL: El lenguaje estándar de los SGBD Relacionales

Ejemplos de transacciones

Esquemas de las tablas usadas

R (atrA: integer, atrB: integer)

S (atrX: integer, atrY: integer)

Ejemplo 1

- INSERT INTO **R** VALUES (1,2)
- ROLLBACK
- INSERT INTO **R** VALUES (3,4)
- COMMIT

⟹ **R** contiene …

Ejemplo 2

- INSERT INTO **S** VALUES (5,6)
- SAVEPOINT SP1
- INSERT INTO **S** VALUES (7,8)
- SAVEPOINT SP2
- INSERT INTO **S** VALUES (9,10)
- ROLLBACK TO SP1
- INSERT INTO **S** VALUES (11,12)
- COMMIT

⟹ **S** contiene …

LEER PREVIAMENTE: INTRODUCCIÓN A PL/SQL

4.2.6 Disparadores

- **Disparador (trigger)**. Mecanismo que se utiliza para dotar a la base de datos de un conjunto de reglas activas

- Su función principal es ejecutar ("disparar") una **acción** cuando ocurre un determinado **evento** (inserción, eliminación o actualización de tuplas, eventos del sistema, operación LDD, etc.)

- Los disparadores se almacenan en la Base de Datos, y el SGBD asume la responsabilidad de ejecutarlo cada vez que ocurra el evento especificado y se satisfaga la condición correspondiente

SQL: El lenguaje estándar de los SGBD Relacionales

- Utilidades:
  - **Mantenimiento de restricciones de integridad complejas**

    si la política de la empresa especifica que no puede aumentar el precio de un artículo, habrá que controlar las actualizaciones de dicho campo

  - **Mantenimiento de valores calculados**

    si en una tabla tenemos valores parciales de ventas y en otra los totales, al insertar, eliminar o modificar un valor parcial, habrá que actualizar automáticamente el valor total

  - **Auditoría**

    registrar en una tabla los cambios producidos en determinadas tablas críticas, almacenando el usuario que los llevó a cabo así como la fecha y la hora

  - **Facilidad de codificación de las aplicaciones que trabajan sobre la base de datos**

    si aumentamos el número de restricciones de integridad en la base de datos conseguimos reducir, en los programas de aplicación, el número de comprobaciones necesarias para mantener la integridad de los datos

SQL: El lenguaje estándar de los SGBD Relacionales

- Básicamente están formados por tres elementos: *Evento, Condición y Acción*, por lo que también se conocen como **reglas ECA**:

  - **Evento:** la acción puede ser llevada a cabo cuando se producen uno o varios eventos
  - **Condición:** determina si la acción debe ejecutarse cuando se produce el evento. Esta condición es opcional
  - **Acción:** suele ser una secuencia de sentencias SQL

- Los disparadores se implementaron en muchos SGBD antes de que formaran parte del estándar de SQL.
- Por desgracia, cada sistema de bases de datos implementó su propia sintaxis para los disparadores, conduciendo a incompatibilidades

SINTAXIS

```
CREATE [OR REPLACE] TRIGGER nombreDisparador
                                  evento
{BEFORE | AFTER} sucesoDisparador ON Tabla
                                        condición
[FOR EACH ROW [WHEN condiciónDisparador]]
          acción
cuerpoDisparador;
```

sucesoDisparador: operación LMD que lanza el cuerpo del disparador cuando se efectúa sobre la tabla. Puede haber varios sucesos separados por el operador **OR**

SQL: El lenguaje estándar de los SGBD Relacionales

- Un disparador LMD se activa con una operación INSERT, UPDATE o DELETE realizada sobre una tabla de la base de datos.

evento

- Puede realizarse antes o después de la ejecución de la instrucción.

- Puede ejecutarse una vez por cada fila afectada o una vez por cada instrucción.

Tipo de disparadores:

Categoría	Valores	Comentarios
Instrucción	INSERT DELETE UPDATE	Indica el tipo de instrucción LMD que causa la activación del disparador
Temporización	Antes o después	Indica si el disparador se activa antes o después de la ejecución de la instrucción LMD
Nivel	Fila o instrucción	- Si es a nivel de fila, se dispara una vez por cada fila afectada. - Si es a nivel de instrucción, se dispara una única vez

SQL: El lenguaje estándar de los SGBD Relacionales

- **Ejemplo 1:** se desea almacenar en una tabla de control la fecha y el usuario que modifica una nota en la tabla MATRICULA

```
CREATE TABLE modifNotas
( numero number(5),
  usuario varchar2(15),
  fecha date,
  constraint CP_modifNotas PRIMARY KEY(numero));
```

Tabla de control

Disparador

```
CREATE OR REPLACE TRIGGER controlNotas
  AFTER UPDATE OF feb_jun, sep, dic ON matricula
BEGIN
  INSERT INTO modifNotas (numero, usuario, fecha)
      VALUES (secuencia_control_notas.NEXTVAL, USER, SYSDATE);
END;
```

- Sólo se ejecuta una vez, después de la actualización, aunque una sola operación UPDATE puede actualizar muchas filas de la tabla matrícula

Código para crear la secuencia *secuencia_control_notas*

```
create sequence secuencia_control_notas
start with 1
increment by 1;
```

SQL: El lenguaje estándar de los SGBD Relacionales

Disparadores de fila

- Se ejecutan una vez por cada fila procesada en la sentencia LMD disparadora
- Para acceder a la fila procesada se usan dos identificadores de correlación: **:old** y **:new**
- Un identificador de correlación es un tipo especial de variable de PL/SQL
- El compilador de PL/SQL los trata como si fuesen registros de tipo **tabla_disparo%ROWTYPE**

Orden LMD	:old	:new
INSERT	Indefinido. Todos los campos contienen el valor NULL	Valores que se insertarán en la tabla cuando finalice la instrucción
UPDATE	Valores originales antes de la actualización	Valores actualizados después de la instrucción
DELETE	Valores originales antes de la eliminación	Indefinido. Todos los campos contienen el valor NULL

SQL: El lenguaje estándar de los SGBD Relacionales

- **Ejemplo 2:** se desea controlar que el primer carácter del código de la asignatura sea una 'A' y tenga 4 caracteres en total

```
CREATE OR REPLACE TRIGGER codigoAsigValido
  BEFORE INSERT OR UPDATE OF idAsig ON asignatura
  FOR EACH ROW
BEGIN
  IF SUBSTR(:new.idAsig,1,1)<>'A'
    THEN RAISE_APPLICATION_ERROR(-20001,'Código de asignatura debe empezar por A');
  END IF;
  IF LENGTH(TRIM(:new.idAsig))<>4
    THEN RAISE_APPLICATION_ERROR(-20002,'Código de asignatura debe tener 4 caracteres');
  END IF;
END;
```

SUBSTR(cad, ini, ncar) → Extrae *ncar* caracteres de la cadena *cad* a partir del carácter *ini. Por ejemplo SUBSTR ("PRODUCTO", 2, 3)= "ROD"*

TRIM(cad) → Devuelve la cadena *cad*, sin contar los blancos. Por ejemplo, si IdProd="MO" y esta definido como CHAR(4), LENGTH(TRIM (IdProd)=2), mientras que LENGTH(IdProd)=4.

- **Ejemplo 3:** supongamos que un ordenador no lo pueden manejar más de 3 alumnos

- La cláusula **REFERENCING** sirve para asignar un nombre diferente a los pseudoregistros :old y :new

```
CREATE OR REPLACE TRIGGER solo3ordenadores
  BEFORE INSERT ON ALUMNO
  REFERENCING new AS nuevo_alumno
  FOR EACH ROW
DECLARE
  NUM_ORD INTEGER;
BEGIN
  SELECT COUNT(*) INTO NUM_ORD FROM ALUMNO
  WHERE ordenador=:nuevo_alumno.ordenador;
  IF NUM_ORD >= 3
  THEN
    :nuevo_alumno.ordenador:=NULL;
    DBMS_OUTPUT.PUT_LINE('El ordenador está asignado a 3 alumnos.');
    DBMS_OUTPUT.PUT_LINE('No olvide asignarle un ordenador.');
  END IF;
END;
```

SQL: El lenguaje estándar de los SGBD Relacionales

- **Disparadores de fila (la cláusula WHEN)** **- CONDICIÓN -**
  - Hace que sólo se ejecute el cuerpo del disparador en las filas donde se cumple la condición especificada en la cláusula WHEN
  - Dentro de la cláusula WHEN se puede hacer referencia a los pseudoregistros :old y :new pero sin anteponerle los dos puntos(:)

- **Ejemplo 4:** queremos asegurar que no se actualiza una nota de septiembre o diciembre si el alumno ya ha aprobado en junio

```
CREATE OR REPLACE TRIGGER aprobadosJunio
  BEFORE INSERT OR UPDATE OF sep, dic ON MATRICULA
  FOR EACH ROW
  WHEN (new.feb_jun>=5)
BEGIN
    DBMS_OUTPUT.PUT_LINE('El alumno tiene aprobada la asignatura en junio.');
    :new.sep:=NULL;
    :new.dic:=NULL;
END;
```

SQL: El lenguaje estándar de los SGBD Relacionales

- Los disparadores para múltiples comandos de LMD sobre una tabla se pueden combinar en un único disparador. Para ello se utilizan las cláusulas **INSERTING**, **DELETING** y **UPDATING**

```
CREATE OR REPLACE TRIGGER Auditoria
  BEFORE INSERT OR  UPDATE OF feb_jun, sep, dic ON matricula
  FOR EACH ROW
BEGIN
  IF INSERTING THEN
        /* sentencias a realizar cuando el suceso es una inserción */
  ELSE
        /* sentencias a realizar cuando el suceso es una actualización */
END;
```

SQL: El lenguaje estándar de los SGBD Relacionales

- Las **tablas mutantes** son aquellas que están siendo modificadas por una operación LMD

- Para un disparador, la tabla mutante es la tabla sobre la cual está definido así como las tablas que referencien a la mutante y que, por tanto, pueden ser actualizadas como consecuencia de la integridad referencial (por ejemplo, DELETE CASCADE)

FOR EACH ROW

- Dentro del cuerpo de los disparadores a nivel de fila **no** pueden existir:
  - lecturas o modificaciones de tablas mutantes
  - lecturas o modificaciones de claves primarias, ajenas o alternativas de tablas referenciadas por la tabla de disparo. Sin embargo se pueden modificar el resto de las columnas
  - HAY UNA EXCEPCIÓN: no se produce error de tabla mutante en los disparadores BEFORE INSERT de nivel de fila

- Dentro del cuerpo de los disparadores a **nivel de sentencia** no existen problemas de tablas mutantes

SQL: El lenguaje estándar de los SGBD Relacionales

☑ **Ejemplo 6:** se desea lanzar un disparador que no permita más de 20 ordenadores del mismo tipo cada vez que se realice una modificación

```
CREATE OR REPLACE TRIGGER mutante1
  BEFORE UPDATE OF tipo ON ORDENADOR
  FOR EACH ROW
DECLARE
  num_ord NUMBER;
BEGIN
   SELECT COUNT(*) INTO num_ord
   FROM ORDENADOR
    WHERE tipo=:new.tipo;
    IF num_ord > 20 then
       RAISE_APPLICATION_ERROR(-20000,'Demasiados
                                        ordenadores');
    END IF;
END;
```

se está accediendo a la misma tabla que lanza el disparador

```
update ordenador set tipo='Servidor de Impresión' where idOrd='Ord400';
```

ORA-04091: la tabla USER.ORDENADOR está mutando, puede que el disparador/la función no puedan verla

ORA-06512: en "USER.MUTANTE1", línea n

ORA-04088: error durante la ejecución del disparador 'USER.MUTANTE1'

SQL: El lenguaje estándar de los SGBD Relacionales

Otras funciones:

- **Activar/desactivar un disparador**

  ALTER TRIGGER nombre_disparador ENABLE;

  ALTER TRIGGER nombre_disparador DISABLE;

- **Eliminar un disparador**

  DROP TRIGGER nombre_disparador;

- **Ver todos los disparadores y su estado**

  SELECT TRIGGER_NAME, STATUS FROM USER_TRIGGERS;

- **Ver el cuerpo de un disparador**

  SELECT TRIGGER_BODY FROM USER_TRIGGERS
  WHERE TRIGGER_NAME='nombre_disparador';

- **Ver la descripción de un disparador**

  SELECT DESCRIPTION FROM USER_TRIGGERS

  WHERE TRIGGER_NAME='nombre_disparador';

4.2.7 Restricciones generales de integridad

- En SQL se pueden especificar restricciones más generales, esto es, aquellas que no pueden ser expresadas en la instrucción CREATE TABLE (CHECK, NOT NULL, etc.)
- La sentencia **CREATE ASSERTION** del LDD permite definir este tipo de restricciones
- Cada aserción recibe un nombre y se especifica mediante una condición similar a la cláusula WHERE de una consulta SQL
- **EJEMPLO:** "Sólo pueden ser responsable de asignatura aquellos profesores que tengan más de dos años de antigüedad"

```
CREATE ASSERTION profesores_noveles (
CHECK (NOT EXISTS
      (SELECT * FROM PROFESOR P INNER JOIN ASIGNATURA A
               ON P.nPr=A.prof
      WHERE (TO_CHAR(SYSDATE,'YYYY') - P.ant) <= 2 ));
```

- Cuando alguna tupla de la base de datos hace que la condición de una sentencia ASSERTION se evalúa a *falso*, significa que no se cumple la restricción.
- Los disparadores van más allá de las aserciones ya que, además de impedir una inserción o actualización incorrecta, permiten realizar diferentes acciones
- **Oracle no implementa las aserciones**

4.3 Otros lenguajes relacionales: QBE

- **QBE** (*Query By Example* o consulta mediante ejemplos) es un lenguaje de alto nivel no procedural basado en el cálculo de dominios y desarrollado por IBM

- Las consultas se realizan usando variables de dominio y constantes

- Características:

  - No hay que especificar una consulta estructurada. Las consultas se diseñan completando unas **plantillas** de relaciones

  - El usuario no tiene que seguir reglas de sintaxis rígidas para especificar las consultas. Éstas se construyen introduciendo las constantes y variables en las columnas de las plantillas, para construir un ejemplo relacionado con la solicitud que se desea realizar

- Un ejemplo de **plantilla** de la relación ALUMNO sería:

Alumno	Nal	DNI	Nombre	FechaNac	Lugar	nH	Ordenador

SQL: El lenguaje estándar de los SGBD Relacionales

Un ejemplo de QBE: Microsoft Access

- QBE de Access está diseñado para un entorno gráfico, por lo que se suele denominar **GQBE** (Graphical Query By Example)
- Una característica especial es que usa una línea para unir los atributos de las tablas que cumplen una condición de combinación o concatenación
- **Ejemplo**: mostrar el nombre del alumno y el lugar donde realiza las prácticas, pero sólo de aquellos alumnos nacido en Huelva y que utilizan un PC de Prácticas