SQL (I)
DDL y sentencias DML de modificación de datos

Bases de Datos

Curso 2018-2019

Jesús Correas – jcorreas@ucm.es

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

.

Lenguajes de consulta de BD

- **SQL** es el lenguaje estándar para consultar BD relacionales (respecto a consultas de datos es básicamente un superconjunto del algebra relacional).
  - desarrollado por IBM a mediados de los 70.
  - Otros lenguajes son **DATALOG**, basado en programación lógica, y **XQUERY**, lenguaje funcional para consultar documentos XML.
- **1979:** Oracle presentó la primera implementación comercial de SQL.
- **1986:** ANSI adopta SQL como lenguaje estándar de los SGBD relacionales. Un año después lo adopta ISO.
- **1992:** Aparece SQL92, la versión más popular del lenguaje.
- Posteriormente, se aprueban revisiones del estándar con diversas mejoras: **2003**, **2006**, **2008**, **2011**, **2016**.

Introducción SQL

- SQL en realidad está formado por varios lenguajes:
  - Lenguaje de **definición de datos** (**DDL**): creación de tablas, índices, modificación de tablas, etc.
  - Lenguaje de **Manipulación de Datos** (**DML**): `SELECT`, `INSERT`, `UPDATE`, `DELETE`.
  - Lenguaje de **Control de Datos** (**DCL**): control de acceso de usuarios.
  - Lenguaje de **Control de Transacciones** (**TCL**): `COMMIT`, `ROLLBACK`.
- Modos de ejecución:
  - **Directa:** Las instrucciones se introducen en un cliente conectado directamente al servidor SQL.
  - **Embebido:** (embedded) El código SQL forma parte del código fuente de otro lenguaje anfitrión (C, Java).
    - ⋆ Se utiliza un **precompilador** para traducir las sentencias SQL a llamadas a funciones de librería para conectar con el SGBD.
  - **Dinámico:** Las instrucciones SQL se generan durante la ejecución del programa anfitrión y se envían al SGBD como un string.

DDL – Lenguaje de Definición de Datos

- El **DDL** permite la especificación de la estructura de la BD:
- Estructura de las **tablas:** corresponden a los **esquemas de relación** del Modelo Relacional.
- Los dominios de las **columnas:** los dominios de los **atributos** del Modelo Relacional.
- Especificación de las **restricciones de integridad:** claves primarias, claves externas, unicidad, valores nulos, otras restricciones.
- Además, se pueden crear **índices** sobre las tablas para acelerar algunas consultas (aunque esto lo hacen los administradores de BD para ajustar el rendimiento de la BD a bajo nivel).
- Otros objetos de la BD, como por ejemplo **secuencias.**

DDL – Dominios de columnas – Tipos de datos básicos

- Tipos de cadenas de caracteres:
  - **`CHAR(n)`:** cadena de caracteres de longitud fija n. Hasta 2000 caracteres, por defecto 1.
  - **`VARCHAR2(n)`:** cadena de caracteres de longitud variable, máximo n (hasta 4000 caracteres).
- Tipos numéricos:
  - **`NUMBER`:** en el rango $-10^{125}..10^{125}$ con 38 dígitos significativos.
  - **`NUMBER(p,s)`:** números decimales, donde p es el número **total** de dígitos y s es el número de decimales.
  - **`INTEGER`:** enteros de 32 bits.
  - Ejemplos:

Valor asignado	tipo numérico	Valor almacenado
7,456,123.89	NUMBER(9)	7456124
7,456,123.89	NUMBER(9,2)	7456123.89
7,456,123.89	NUMBER(9,1)	7456123.9
7,456,123.89	NUMBER(6)	No valido, supera la precisión
7,456,123.89	NUMBER(7,-2)	7456100

DDL – Dominios de columnas – Tipos de datos básicos

- **`DATE:`** fecha y hora en una sola columna: año, mes, día, hora, minuto y segundo (y hasta milisegundos).
- Desde 1 de Enero del 4712 AC, hasta 31 de Diciembre 9999 DC.
- El formato por defecto viene dado por el parámetro `NLS_DATE_FORMAT`.
- Internamente una fecha se almacena como el número de días desde cierto punto de inicio. Se pueden realizar **operaciones aritméticas:**

```
'1-JAN-2001' + 10                 =  '11-JAN-2001'
'27-FEB-2000' + 2                 =  '29-FEB-2000'
'10-MAY-2000' - '1-MAY-2000'      =  9
```

- Oracle dispone de más tipos de datos: `http://docs.oracle.com`
  - **`DECIMAL, SHORTDECIMAL, SHORTINTEGER, LONGINTEGER, NCHAR, NVARCHAR2, TIMESTAMP, BLOB, CLOB, BFILE,`** etc.

DDL – Creación de tablas

- La creación de tablas se realiza mediante la sentencia **CREATE TABLE:**

```
CREATE TABLE nombreTabla
 (columna_1 tipo_1 [propiedades],
  columna_2 tipo_2 [propiedades],
  ......
  columna_n tipo_n [propiedades],
  restricción_integridad1,
  ......
  restricción_integridadk );
```

- Propiedades de columnas:
  - **DEFAULT** *valor*
    Valor por defecto cuando se inserta una nueva fila en la tabla.
  - **NOT NULL**
    Si la columna no permite valores nulos (por defecto sí se permiten).
  - **Restricciones** de una columna (ver siguiente transparencia).
  - ... y otras propiedades.

DDL – Creación de tablas – Restricciones

- Las **restricciones** son **condiciones de obligado cumplimiento** para una o más columnas de una tabla.
  - Cuando afectan a una única columna, las restricciones **pueden indicarse en la definición de la columna.**
  - Si afectan a varias columnas, se deben indicar **al final de la definición de la tabla.**
  - Por defecto Oracle asigna un nombre a cada restricción, pero se puede indicar un nombre de restricción.
- Sintaxis general para las restricciones:

[CONSTRAINT *nombre*] *restricción*

(Los paréntesis cuadrados no son parte del lenguaje: indican elementos **opcionales**).

DDL – Creación de tablas – Tipos de restricciones

- Tipos de restricciones (los nombres de columnas solo son necesarios para las restricciones de varias columnas):
  - **PRIMARY KEY [(**$C_1, \ldots, C_j$**)]**
    La tabla no permite valores duplicados para $C_1, \ldots, C_j$ y tampoco pueden contener valores nulos.
  - **[FOREIGN KEY (**$C_1, \ldots, C_j$**)] REFERENCES** ***tabla*** **[(**$B_1, \ldots, B_j$**)]**
    Los valores de $C_1, \ldots, C_j$ de cualquier fila deben ser **null** o deben corresponder al valor de la clave primaria (o restricción `unique`) de de una fila de *`tabla`*.
  - **UNIQUE [(**$C_1, \ldots, C_j$**)]**
    La tabla no permite valores duplicados para $C_1, \ldots, C_j$.
  - **CHECK(*condition*)**
    La expresión Booleana `condition` debe ser cierta para todas las filas de la tabla.

DDL – Creación de tablas – Tipos de restricciones

- **Ejemplos:**

```
CREATE TABLE sucursal
 (nombre_sucursal VARCHAR2(15) PRIMARY KEY,
  ciudad CHAR(20) NOT NULL CONSTRAINT cl_UK UNIQUE,
  activos NUMBER(12,2) DEFAULT 0
 );

CREATE TABLE cliente
 (dni VARCHAR2(9) NOT NULL,
  nombre_cliente CHAR(35) NOT NULL,
  domicilio CHAR(50) NOT NULL,
  CONSTRAINT cl_PK PRIMARY KEY (dni)
 );
```

DDL – Creación de tablas – Restricciones

- **Más ejemplos:**

```
CREATE TABLE cuenta
 (numero_cuenta CHAR (20) PRIMARY KEY,
  nombre_sucursal VARCHAR2(15) REFERENCES sucursal,
  saldo NUMBER(12,2) DEFAULT 100,
  CHECK (saldo >= 100)
 );

Create table impositor
 (dni VARCHAR2(9) REFERENCES cliente,
  numero_cuenta CHAR(20) NOT NULL,
  PRIMARY KEY (dni, numero_cuenta),
  FOREIGN KEY (numero_cuenta) REFERENCES cuenta
 );
```

DDL – Creación de tablas – Manejo de restricciones

- Cuando se insertan o modifican filas de una tabla se debe **garantizar el cumplimiento de las restricciones de clave externa:**
  - Cuando se **modifica la clave primaria (o única) de una fila** a la que se refiere la clave externa de otra tabla.
  - Cuando se **elimina una fila** de una tabla a la que se refiere una clave externa de otra tabla.
- Como regla general, cuando se incumple una clave externa **se rechaza la acción y se produce un error.**
- Se puede modificar este comportamiento mediante cláusulas en la creación de la clave externa:
  - **`ON DELETE CASCADE:`** Cuando se elimina una fila de la tabla referida, todas las filas de las tablas que la referencian también son borradas. (¡este comportamiento no se recomienda!).
  - **`ON DELETE SET NULL | SET DEFAULT:`** las columnas que referencian una fila eliminada se actualizan a **`NULL`** o sus valores por defecto.
  - **`ON UPDATE CASCADE | SET NULL | SET DEFAULT.`**

DDL – Creación de tablas – Manejo de restricciones

- **Ejemplos:**

```
CREATE TABLE cuenta
 (numero_cuenta CHAR (20) PRIMARY KEY,
  nombre_sucursal char(15)
    REFERENCES sucursal ON DELETE SET NULL,
  saldo NUMBER(12,2) DEFAULT 100,
  CHECK(saldo >=100)
 );

CREATE TABLE impositor
 (dni CHAR(9) REFERENCES cliente ON DELETE CASCADE,
  numero_cuenta CHAR(20) NOT NULL,
  PRIMARY KEY (dni, numero_cuenta),
  FOREIGN KEY (numero_cuenta)
    REFERENCES cuenta ON DELETE CASCADE
 );
```

DDL – Cambio de la estructura de tablas existentes

Es posible cambiar la estructura de una tabla existente y con datos (hasta cierto punto):

- Añadir columnas a una tabla:

  ```
  ALTER TABLE tabla ADD columna dominio [propiedades];
  ```

- Eliminar columnas de una tabla:

  ```
  ALTER TABLE tabla DROP COLUMN columna;
  ```

  No se puede eliminar una columna que aparece en una restricción; se puede utilizar lo siguiente:

  ```
  ALTER TABLE tabla DROP columna CASCADE CONSTRAINTS;
  ```

- Modificar columnas de una tabla (p. ej. para extender el dominio):

  ```
  ALTER TABLE tabla MODIFY (columna dominio [propiedades]);
  ```

- Renombrar columnas de una tabla:

  ```
  ALTER TABLE tabla RENAME COLUMN columna TO nuevoNombre;
  ```

DDL – Modificaciones sobre tablas creadas

- Añadir restricciones a una tabla:

  ```
  ALTER TABLE tabla ADD CONSTRAINT nombre Tipo (columnas);
  ```

- Eliminar restricciones de una tabla:

  ```
  ALTER TABLE tabla DROP PRIMARY KEY;
  ALTER TABLE tabla DROP UNIQUE(campos);
  ALTER TABLE tabla DROP CONSTRAINT nombre [CASCADE];
  ```

  La opción **CASCADE** hace que se eliminen las restricciones de integridad que dependen de la eliminada.

- Desactivar restricciones:

  ```
  ALTER TABLE tabla DISABLE CONSTRAINT nombre [CASCADE];
  ```

- Activar restricciones:

  ```
  ALTER TABLE tabla ENABLE CONSTRAINT nombre;
  ```

DDL – Modificaciones sobre tablas creadas

- **Ejemplos:**

```
ALTER TABLE cuenta ADD comision NUMBER(4,2);

ALTER TABLE cuenta ADD fecha_apertura DATE;

ALTER TABLE cuenta DROP COLUMN nombre_sucursal;

ALTER TABLE cuenta MODIFY comision DEFAULT 1.5;

ALTER TABLE cliente MODIFY nombre_cliente NULL;

ALTER TABLE sucursal ADD CONSTRAINT cd_UK UNIQUE(ciudad);
```

DDL – Otras operaciones sobre tablas

- Descripción de una tabla.

  DESCRIBE *tabla*;

- Eliminación de una tabla.

  DROP TABLE *tabla* [CASCADE CONSTRAINTS];

  La opción **CASCADE** hace que se eliminen las restricciones de integridad que dependen de la tabla eliminada.

- Renombrar una tabla.

  RENAME TABLE *tabla* TO *nuevoNombre*;

- Borrar contenido de una tabla.

  TRUNCATE TABLE *tabla*;

DDL – Secuencias

- Las **secuencias** permiten generar automáticamente números distintos.
- La generación de cada valor es atómica y se puede realizar desde distintas sesiones concurrentemente.
- Las secuencias son independientes de las tablas donde se utilizan.
- Para crear una secuencia:

```
CREATE SEQUENCE secuencia [INCREMENT BY m]
  [START WITH n] [MAXVALUE p|NOMAXVALUE]
  [MINVALUE q|NOMINVALUE] [CYCLE|NOCYCLE];
```

DDL – Secuencias

- Los métodos **NEXTVAL** y **CURRVAL** se utilizan para obtener el siguiente número y el valor actual de la secuencia respectivamente.
  - **NEXTVAL** incrementa la secuencia y devuelve el nuevo valor.
  - **CURRVAL** devuelve el valor de la secuencia sin incrementarla.
- Se pueden utilizar en cualquier sitio en el que se espere una expresión numérica (excepto en cláusulas `DEFAULT`).
- Se pueden modificar las características de las secuencias después de utilizarlas, pero la modificación sólo puede afectar a los valores posteriores al cambio.
- **Ejemplo:**

```
CREATE SEQUENCE sq_id_cliente INCREMENT BY 1
  START WITH 10 MAXVALUE 200000;
SELECT sq_id_cliente.CURRVAL FROM DUAL;
INSERT INTO cliente (id_cliente,nombre)
  VALUES (sq_id_cliente.NEXTVAL,'John Doe');
```

DDL – Índices

- Los **índices** permiten acelerar las operaciones de consulta y ordenación sobre los campos a los que el índice hace referencia. (internamente se representan mediante árboles B o bitmaps).
  - Sin embargo, hacen menos eficientes las operaciones de modificación de datos (`UPDATE`, `INSERT`).
- La mayoría de los índices se crean de manera automática, como consecuencia de las **restricciones** **`PRIMARY KEY`** y **`UNIQUE`**.
- Se pueden crear índices adicionales para aquellas combinaciones de columnas sobre las que se realizarán búsquedas e instrucciones de ordenación frecuentemente.

```
CREATE [UNIQUE] INDEX nombreIndice
  ON NombreTabla(col1,...,colk);
```

- Por ejemplo, puede ser conveniente crear índices sobre las claves externas si se espera que se realicen consultas sobre esas combinaciones de columnas.
- En cualquier caso, la creación de índices es tarea del administrador.

DML – Lenguaje de Manipulación de Datos

- El **lenguaje de manipulación de datos (DML)** está formado por cuatro sentencias:
- **`INSERT`** para insertar filas en una tabla.
- **`DELETE`** para eliminar filas existentes de una tabla.
- **`UPDATE`** para modificar el contenido de filas existentes en una tabla.
- **`SELECT`** para realizar consultas en tablas.

DML – INSERT.

- La sentencia **INSERT** inserta una o varias filas nuevas en una tabla.
- La **sintaxis básica para insertar una fila** es:

```
INSERT INTO NombreTabla [(col1,...,colk)]
  VALUES (val1,...,valk);
```

- La lista de valores debe contener los valores a insertar.
- El orden de la lista de nombres de columnas *col1,...,colk* debe coincidir con el **orden de la lista de valores val1,...,valk**.
  Si se omite la lista de nombres de columnas, se utiliza el orden de definición de columnas en la creación de la tabla.
- **Ejemplo:**

```
INSERT INTO client VALUES (37,'Juan García','C/Pez,5');
```

- Los **literales** de caracteres deben encerrarse entre **comillas simples.**
- **Sintaxis alternativa:** se puede utilizar una consulta SELECT para generar las filas a insertar (Veremos detalles más adelante):

```
INSERT INTO Prestamo SELECT * FROM NuevosPrestamos;
```

DML – DELETE.

- La sentencia **DELETE** elimina filas existentes en una tabla que cumplan determinada condición.
- La **sintaxis** es la siguiente:

```
DELETE FROM NombreTabla WHERE condición;
```

- La condición de la cláusula **WHERE** se evalúa **para cada fila.** Si es cierta, se elimina la fila.
- La cláusula WHERE es **opcional**, pero si se omite, se borran todas las filas de la tabla.
- La cláusula **WHERE** puede contener condiciones complejas, incluso otras consultas anidadas.
  - La estudiaremos en detalle cuando veamos la sentencia **SELECT**.
- **Ejemplo:**

```
DELETE FROM cliente WHERE id_cliente = 37;
```

DML – UPDATE.

- la sentencia **UPDATE** modifica los valores de las filas de una tabla que cumplan determinada condición.
- La **sintaxis** es la siguiente:

```
UPDATE NombreTabla SET col1=expr1,..., colk = exprk
  WHERE condición;
```

- La condición de la cláusula **WHERE** se evalúa **para cada fila.** Si es cierta, se modifican las columnas especificadas en la cláusula **SET.** Se pueden usar
- Si no se especifica cláusula **WHERE**, se actualizan todas las filas de la tabla.
- La cláusula **WHERE** puede contener condiciones complejas, incluso otras consultas anidadas.
- **Ejemplo:**

```
UPDATE empleado SET sueldo = sueldo*1.1 WHERE deptId='IS';
```