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

Bases de Datos

Curso 2016-2017

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 (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.
- **1999:** Se aprueba el estándar SQL99 que incorpora mejoras.
- Posteriormente, se aprueban revisiones del estándar: **2003**, **2006**, **2008**.

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 consultas.

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).
  - **`LONG`:** cadena de caracteres de longitud variable hasta 2 gigabytes.
- Tipos numéricos:
  - **`NUMBER`:** cualquier valor numérico 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. La escala puede ser negativa.
  - **`INTEGER, SHORTINTEGER, LONGINTEGER`:** enteros de 32, 16 y 64 bits.
  - **`DECIMAL`:** números decimales con 15 dígitos significativos en el rango $-10^{308}.,10^{308}$.

DDL – Dominios de columnas – Tipos de datos básicos

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 almacena información del día, mes, año, hora, minuto y segundo.
- 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'
'1-JAN-2000' - 1                   =  '31-DEC-1999'
'10-MAY-2000' - '1-MAY-2000'       =  9
```

- **`TIMESTAMP[(fs)]:`** igual que DATE, con fracciones de segundo. El parámetro fs indica el número de dígitos de dicha fracción. Por defecto 6.
- Oracle dispone de más tipos de datos: `http://docs.oracle.com`

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 que se pueden especificar (entre otras):
  - **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).

DDL – Creación de tablas – Restricciones

- Las **restricciones** son **condiciones de obligado cumplimiento** para una o más columnas de una tabla.
- Por defecto Oracle asigna un nombre a cada restricción, pero Es recomendable indicar un nombre de restricción utilizando unas normas comunes:
  - Un prefijo o sufijo que indique el tipo de restricción: `PK` (primary key), `FK` (foreign key), `UK` (unique), `CK` (check).
  - el nombre de la tabla y la(s) columnas afectadas.
- Cuando afectan a una única columna, las restricciones pueden indicarse junto con las propiedades de la columna.
- Si afectan a varias columnas, se deben indicar después de la descripción de todas las columnas.
- Tipos de restricciones: **`UNIQUE`**, **`PRIMARY KEY`**, **`CHECK`**, **`FOREIGN KEY`**

DDL – Creación de tablas – Restricciones

- **`CONSTRAINT` *`nombre`* `UNIQUE(`**$A_{j1}, \ldots, A_{jp}$**`)`**
  La combinación de valores de los atributos es única.
- **`CONSTRAINT` *`nombre`* `PRIMARY KEY(`**$A_{j1}, \ldots, A_{jp}$**`)`**
  La combinación de valores de los atributos es única y no pueden tener valor nulo.
- **`CONSTRAINT` *`nombre`* `CHECK(`*`predicado`*`)`**
  El predicado debe ser satisfecho por todas las filas de la tabla.
- **`CONSTRAINT` *`nombre`* `FOREIGN KEY (`**$A_{j1}, \ldots, A_{jp}$**`) REFERENCES` *`tabla`***
  `(`$B_{j1}, \ldots, B_{jp}$**`)`**
  Clave externa: define un conjunto de atributos cuyos valores coinciden con los de los atributos de la clave primaria (o única) de la misma tabla o de otra.
  - Si uno de los atributos de la clave externa toma valor nulo no se comprueba el valor en la tabla referenciada.

DDL – Creación de tablas – Restricciones

- **Ejemplos:**

```
CREATE TABLE sucursal
 (nombre_sucursal VARCHAR2(15) CONSTRAINT suc_PK 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

- **Ejemplos:**

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

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

DDL – Creación de tablas – Restricciones

- Determinadas operaciones de modificación de datos pueden **incumplir las restricciones de clave externa:**
  - Cuando se **modifica una clave primaria** (o única) referenciada en otra tabla con una clave externa.
  - Cuando se **elimina una fila** cuyos valores de la clave primaria (o única) son referenciados mediante 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 una fila de la tabla padre es borrada, todas las filas de las tabla hijas que la referencian también son borradas.
  - **`ON DELETE SET NULL:`** Cuando una fila de la tabla padre es borrada, los atributos que referencian en las tablas hijas se actualizan a **`NULL`.** (Estos atributos deben aceptar valores nulos).

DDL – Creación de tablas – Restricciones

- **Ejemplos:**

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

CREATE TABLE impositor
 (dni CHAR(9) CONSTRAINT imp_dni_FK REFERENCES cliente ON
    DELETE CASCADE,
  numero_cuenta CHAR(20),
  CONSTRAINT imp_PK PRIMARY KEY (dni, numero_cuenta),
  CONSTRAINT imp_ct_FK FOREIGN KEY (numero_cuenta)
    REFERENCES cuenta ON DELETE CASCADE
 );
```

DDL – Modificaciones sobre tablas creadas

- Añadir atributos a una tabla:

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

- Eliminar atributos de una tabla:

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

  No se puede eliminar la única columna de una tabla: Si la columna interviene en una constraint dará error:

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

- Modificar atributos de una tabla:

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

- Renombrar atributos 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 n]
  [START WITH n] [MAXVALUE n|NOMAXVALUE]
  [MINVALUE n|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 valor actual.
  - **CURRVAL** devuelve el valor de la secuencia, pero sin incrementar la misma.
- No se puede utilizar como valor para la cláusula `DEFAULT` de un campo de tabla.
- Se pueden modificar las secuencias, pero la modificación sólo puede afectar a los futuros valores de la secuencia.
- **Ejemplo:**

```
CREATE SEQUENCE numHabitacion INCREMENT 100
  STARTS WITH 100 MAXVALUE 20000;
SELECT numHabitacion.CURRVAL FROM DUAL;
INSERT INTO Habitacion(numero,tipo)
  VALUES(numHabitacion.NEXTVAL,'Suite');
```

DDL – Índices

- Los **índices** permiten acelerar las operaciones de consulta y ordenación sobre los campos a los que el índice hace referencia.
  - Por otra parte, hacen menos eficientes las operaciones de modificación de datos (`UPDATE`, `INSERT`).
- La mayoría de los índices se crean de manera implícita, como consecuencia de las **restricciones** **`PRIMARY KEY`** y **`UNIQUE`**.
- Se pueden crear explícitamente para aquellos campos sobre los cuales se realizarán búsquedas e instrucciones de ordenación frecuente.

```
CREATE [unique] INDEX NombreIndice
  ON NombreTabla(col1,...,colk);
```

- Aunque no se crean implícitamente, puede ser conveniente crear índices sobre las claves externas.

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 filas existentes en una tabla.
- **`SELECT`** para realizar consultas en tablas.

DML – Inserción de filas en una tabla.

- La sentencia **INSERT** inserta una fila nueva en una tabla.
- La **sintaxis básica** de la sentencia de inserción es:

```
INSERT INTO NombreTabla [(listaColumnas)]
  VALUES (ListaValores);
```

- La lista de valores debe contener los valores a insertar.
- El **orden de los valores** debe coincidir con el orden de las columnas `listaColumnas` (si no se incluye, el orden de `CREATE TABLE`).
- **Ejemplo:**

```
INSERT INTO Prestamo
  VALUES ('Navacerrada','Pepe Pérez',125000);
```

- 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 – Borrado de filas de una tabla.

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

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

- La cláusula **WHERE** es **opcional**. Si no se especifica, se borran todas las filas de la tabla.
- La cláusula **WHERE** puede contener condiciones complejas, incluso otras consultas anidadas.
  - Cuando veamos la sentencia **SELECT** estudiaremos en detalle la cláusula `WHERE`.
- **Ejemplo:**

```
DELETE FROM Prestamo WHERE NumPrestamo='P-260';
```

DML – Modificación de filas de una tabla.

- 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=valor1,..., colk = valork
  WHERE condición;
```

- Como en el caso anterior, la cláusula **WHERE** es opcional y, si no se especifica, se actualizan todas las filas de la tabla.
- La cláusula **WHERE** puede contener condiciones complejas, incluso otras consultas anidadas.
- **Ejemplo:**

```
UPDATE Prestamo SET importe=200000
  WHERE NumPrestamo='P-170';
```