LENGUAJE DE DEFINICIÓN DE DATOS (DDL) VISTAS

- UT4 -

INTRODUCCIÓN **DDL**

Las sentencias **DDL** son sentencias que nos permiten definir, alterar, borrar objetos dentro de la base de datos. Las sentencias más importantes son:

CREATE: Permite crear objetos dentro de la base de datos y la propia base de datos. Estos objetos son: Procedimientos almacenados, tablas, base de datos, triggers (disparadores), funciones, vistas, índices, etc.

ALTER TABLE: Permite **modificar** la estructura de una tabla (añadir (ADD) o eliminar (DROP) columnas y sus tipos de datos, restricciones, etc)

DROP: La sentencia DROP me permite **eliminar** objetos de la base de datos (podríamos decir que es la operación contraria a CREATE).

TRUNCATE: Elimina todos los registros de una tabla.

COMMENT: Nos permite agregar comentarios al diccionario de datos de la BD.

RENAME

VISTAS

Por su relación con el DDL (consultas) trabajaremos inicialmente con las **vistas**.
La creación de vistas permite almacenar consultas como si se trataran de nuevas tablas con la finalidad de utilizar el resultado de las mismas en otras consultas más complejas.

- Cuando se crea una vista se genera una **'tabla lógica'** que permite asignar un nombre al resultado de una consulta y utilizar ésta más adelante y siempre actualizada.

- Hay que tener en cuenta que realmente la consulta que se ha creado como vista no se encuentra almacenada sino que tiene que ser generada cada vez que se deba utilizar.

VISTAS: Sintaxis

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW vista **[(**alias[, alias2...**])]**

AS consulta SELECT

[WITH CHECK OPTION [**CONSTRAINT** restricción**]]**

[WITH READ ONLY [CONSTRAINT restricción**]]**

DROP VIEW *nombre_de_la_vista*

ALTER VIEW *nombre_de_la_vista*

AS

Consulta

VISTAS: Sintaxis

OR REPLACE. Si la vista ya existía, la cambia por la actual (de esta forma podemos modificar una vista creada anteriormente).

FORCE. Crea la vista aunque los datos de la consulta SELECT no existan.

alias. Lista de alias que se establecen para las columnas devueltas por la consulta. El número de alias debe coincidir con el número de columnas devueltas por SELECT. No son necesarios si ya se ha puesto un nombre y/o un alias correcto en cada columna de la instrucción SELECT.

WITH CHECK OPTION. Hace que sólo las filas que se muestran en la vista puedan ser añadidas (**INSERT**) o modificadas (**UPDATE**).

VISTAS: Sintaxis

Las instrucciones DML (**INSERT**, **UPDATE** o **DELETE**) se pueden ejecutar sobre las vistas simples

https://www.tutorialesprogramacionya.com/oracleya/simulador/simulador.php?cod=246

VISTAS

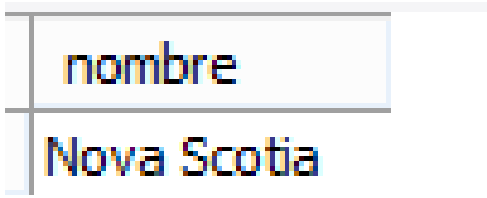
Vista que almacena el número de pistas que hay en cada polideportivo (base de datos "Reservas")

```
CREATE VIEW pistas_por_polideportivo
AS
SELECT PP.id, PP.nombre, COUNT(*) AS cantidad
FROM polideportivos PP, pistas P
WHERE PP.id = P.id_polideportivo
GROUP BY PP.id
```

id	nombre	cantidad
17	Nova Scotia	12
10	SI	11
15	Provence-Alpe...	10
1	BR	9
6	Brd	9
11	TA	9
27	CA	9
28	Minnesota	9
2	NA	8
3	British Columbia	8

VISTAS

Si ahora suponemos que nos pidieran conocer el polideportivo que más pistas tiene, sólo tendríamos que realizar una consulta utilizando la vista creada anteriormente.




```
SELECT nombre
FROM pistas_por_polideportivo
WHERE cantidad = (SELECT MAX(cantidad) FROM pistas_por_polideportivo)
```

Fíjate que puedes usar el identificador “cantidad” como si fuera realmente el nombre de un campo. Este nombre se origina en la vista creada y como se guarda como una tabla virtual, puedes hacer uso de él.

Desde luego esta consulta es mucho más legible que su homóloga sin

VISTAS

Nombre de los polideportivos que tienen más de 3 pistas: Inicialmente esta consulta debería construirse con GROUP BY y HAVING, sin embargo, haciendo uso de la vista que hemos creado:

```
SELECT nombre
FROM pistas_por_polideportivo
WHERE cantidad >3
```

Es mucho más sencillo contestar consultas relacionadas con los datos que se han preparado en la vista.

VISTAS

Para cada usuario calcular el número de reservas que ha hecho y el total que ha gastado en dichas reservas. Incluiremos también a los usuarios que no hayan realizado ninguna reserva.

- Calcular el gasto total del usuario llamado “Ima”
- Obtener el usuario que mayor gasto ha tenido en el club.
- Calcular el gasto total de los usuarios de Zaragoza

```
SELECT * FROM
USER_VIEWS;
```