UT4 – El lenguaje SQL

Parte 2

- Tipos de sentencias
- DDL
- DROP TABLE
- TRUNCATE TABLE
- ALTER TABLE
- RENAME
- Índices
- Vistas
- Sinónimos

TIPOS DE SENTENCIAS SQL

- **TIPOS DE LENGUAJES**

DML (Data Manupulation Language)

Permite manipular y consultar la información que contienen las tablas. (SELECT, INSERT, DELETE, UPDATE,...)

DDL (Data Description Language)

Crea y mantiene la estructura de la BD. Usada por programadores y administradores. (CREATE, DROP, ALTER,...)

DCL (Data Control Language)

Permite gestionar el acceso –confidencialiad- (GRANT, REVOKE,..) y las transacciones a la BD (COMMIT, ROLLBACK,...)

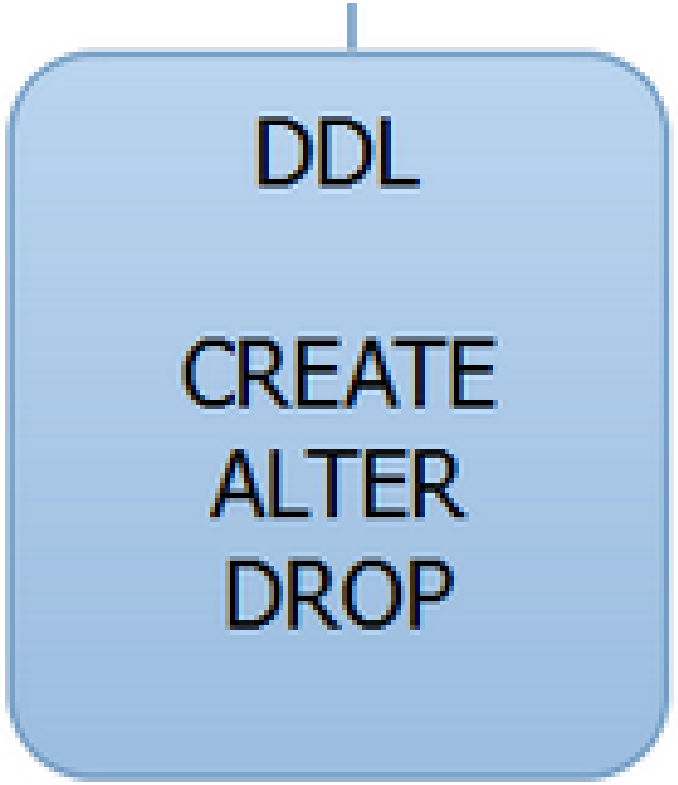
- **LENGUAJE DESCRIPCIÓN/DEFINICION DE DATOS**

Permite crear, modificar y eliminar objetos de la base de datos (es decir, los metadatos).

En un principio, un usuario puede crear sus propios objetos, a no ser que el administrador le retire los privilegios que permitan crear distintos objetos

En Oracle, cada usuario de una base de datos tiene sus propios objetos (tablas, vistas, índices,…), y este conjunto de objetos se denomina ESQUEMA, que tendrá el mismo nombre que el usuario con el que se ha accedido.

NOTA: *Las instrucciones DDL generan acciones que no se pueden deshacer, por eso es conveniente usarlas con precaución y tener copias de seguridad cuando manipulamos la base de datos.*



DROP TABLE

- Permite suprimir una tabla de la base de datos.

  Un usuario solo puede borrar sus propias tablas

  Solo el administrador o algún usuario al que se le otorgara privilegios puede borrar las tablas de otro usuario

  Al borrar una tabla se suprimen los índices y privilegios asociados a ella.

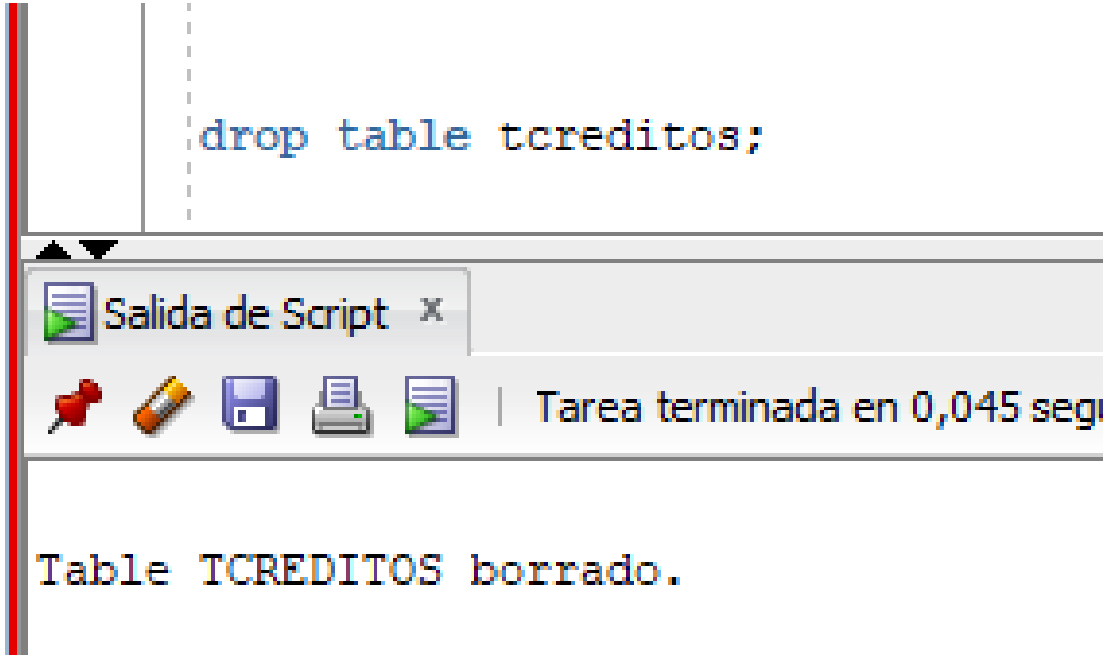
  Las vistas y sinónimos asociados a una tabla dejan de funcionar pero no desaparecen, por lo que habría que eliminarlos.

DROP TABLE

- Formato:

```
DROP TABLE [esquema_usuario.]nombre_tabla
                    [CASCADE CONSTRAINTS];
```

La opción CASCADE CONSTRAINTS permite suprimir todas las restricciones de integridad referencial que se refieran a claves de la tabla borrada.



DROP TABLE

```
CREATE TABLE tprovincias (
    cd_prov NUMBER(2) PRIMARY KEY,
    nom_prov VARCHAR2(15)
);

CREATE TABLE tpersonas (
    cd_pers NUMBER(2) PRIMARY KEY,
    dni VARCHAR2(10),
    nombre VARCHAR2(20),
    direccion VARCHAR2(30),
    poblacion VARCHAR2(10),
    cd_prov NUMBER(2) NOT NULL,
       CONSTRAINT FK_pers_prov FOREIGN KEY (cd_prov) RE
);

DROP TABLE tprovincias;
```

```
da de Script x
Tarea terminada en 0,044 segundos
orme de error -
-02449: claves únicas/primarias en la tabla referidas por
49. 00000 -  "unique/primary keys in table referenced by
use:    An attempt was made to drop a table with unique
        primary keys referenced by foreign keys in anothe
tion:   Before performing the above operations the table,
```

Note: Be careful before deleting a table. Deleting a table results in loss of all information stored in the table!

El borrado de una tabla es irreversible y no hay ninguna petición de confirmación, por lo que conviene ser muy cuidadoso con esta operación. Al borrar una tabla se borran todos los datos que contiene.

Se eliminará la tabla tprovincias aunque su pk sea fk de alguna tabla de la bbdd.

Automáticamente se borrará la restricción de fk asociada.

```
17
18  DROP TABLE tprovincias CASCADE CONSTRAINTS;
19
20
```

```
Salida de Script x
Tarea terminada en 0,059 segundos

Table TPROVINCIAS borrado.
```

CASCADE CONSTRAINTS

Specify `CASCADE CONSTRAINTS` to drop all referential integrity constraints that refer to primary and unique keys in the dropped table. If you omit this clause, and such referential integrity constraints exist, then the database returns an error and does not drop the table.

Truncate table

- Existe también la posibilidad de eliminar las filas de una tabla y los índices, manteniendo la definición de la tabla y los objetos asociados, si se utiliza la sentencia

TRUNCATE TABLE *nombreTabla*

Note: Be careful before deleting a table. Deleting a table results in loss of all information stored in the table!

The `TRUNCATE TABLE` command deletes the data inside a table, but not the table itself.

```
22  TRUNCATE TABLE tpersonas;
23
```

Salida de Script

Tarea terminada en 0,052 segundos

```
Table TPERSONAS truncado.
```

```
23  DESC tpersonas;
24
```

Salida de Script

Tarea terminada en 0,514 segundos

```
Nombre     ¿Nulo?   Tipo
---------- -------- ------------
CD_PERS    NOT NULL NUMBER(2)
DNI                 VARCHAR2(10)
NOMBRE              VARCHAR2(20)
DIRECCION           VARCHAR2(30)
POBLACION           VARCHAR2(10)
CD_PROV    NOT NULL NUMBER(2)
```

TAREA 4 – La de los autores, editoriales y libros

Genera los <> scripts para la creación de las siguientes tablas:

tautores

```
Nombre ¿Nulo?   Tipo
------ -------- ------------
CODIGO NOT NULL NUMBER(4)
NOMBRE NOT NULL VARCHAR2(30)
```

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
SYS_C007250	C	"CODIGO" IS NOT NULL
SYS_C007251	C	"NOMBRE" IS NOT NULL
CK_TAUTORES_CODIGO	C	codigo>=0
PK_TAUTORES_CODIGO	P	(null)
UQ_TAUTORES_NOMBRE	U	(null)

teditoriales

```
Nombre ¿Nulo?   Tipo
------ -------- ------------
CODIGO NOT NULL NUMBER(3)
NOMBRE          VARCHAR2(30)
```

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
SYS_C007248	C	"CODIGO" IS NOT NULL
PK_TEDITORIALES	P	(null)

tlibros

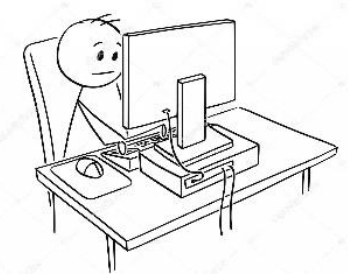
```
Nombre          ¿Nulo?   Tipo
--------------- -------- ------------
CODIGO          NOT NULL NUMBER(5)
TITULO                   VARCHAR2(40)
CODIGOAUTOR              NUMBER(4)
CODIGOEDITORIAL          NUMBER(3)
PRECIO                   NUMBER(5,2)
```

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
CK_TLIBROS_CODIGOEDITORIAL	C	codigoeditorial is NOT NULL
CK_TLIBROS_PRECIONONULO	C	precio is NOT NULL
CK_PRECIO_POSITIVO	C	precio>=0
PK_TLIBROS_CODIGO	P	(null)
UQ_TLIBROS_TITULOAUTOR	U	(null)
FK_TLIBROS_EDITORIAL	R	(null)
FK_TLIBROS_TAUTORES	R	(null)

TAREA 5 -- La de los medicamentos

Diseña el script de generación de una tabla que recoja datos sobre medicamentos.

- El medicamento viene identificado a través de una pegatina con valores como: SPI-01, PAR/92, DAS+30, que identificarán a un medicamento del resto.
- Los nombres de los medicamentos podrán tener hasta 20 caracteres, no pudiéndose repetir el nombre entre medicamentos.
- Recogeremos tb el nombre de los laboratorios que no podrán superar los 20 caracteres.
- Un medicamento puede costar hasta 10000€ - 0,01€.
- La cantidad de medicamentos que tengamos nunca puede ser inferior a 20, pudiendo almacenar hasta 9999 unidades.
- Trabajaremos con dos fechas (que nunca podrán ser nulas):
    - Fecha de compra
    - Fecha de caducidad
        - Inicialmente la fecha de compra será la del día en el que se ha comprado.
- Los medicamentos se almacenarán únicamente en tres almacenes que serán:
    - al_1
    - al_2
    - al_3



TAREA 5 -- La de los medicamentos -- PS

```
CREATE TABLE tmedicamentos(
 codigo VARCHAR2(6) PRIMARY KEY,
 nombre VARCHAR2(20) UNIQUE,
 laboratorio VARCHAR2(20),
 precio NUMBER(6,2),
 cantidad NUMBER(4) CHECK (cantidad > 20),
 fec_compra DATE DEFAULT SYSDATE NOT NULL,
 fec_cad DATE NOT NULL,
 almacen VARCHAR2(4) CHECK (almacen in ('al_1', 'al_2','al_3'))
);
```

ALTER TABLE

Permite modificar la estructura de las tablas.

Las tablas se pueden modificar:

- Cambiando la definición de una columna
- Añadiendo/borrando una columna

Formato:

```
ALTER TABLE nombre_tabla
{
        [ADD    (columna  tipo)]
        [DROP  COLUMN (columna [, columna ]...)]
        [MODIFY (columna tipo [, columna tipo]...)]
        [RENAME COLUMN NombreAntiguo TO NombreNuevo]
        [ADD    CONSTRAINT restricción]
        [DROP  CONSTRAINT restricción]
        [DISABLED CONSTRAINT restricción]
        [ENABLE  CONSTRAINT restricción]
};
```

ALTER TABLE

[ADD (columna tipo)]: Añade columnas a una tabla

Si la columna no está definida como NOT NULL, se puede añadir en cualquier momento

Si la columna está definida como NOT NULL, una opción:

- se define primero la columna sin especificar NOT NUL
- A continuación, se le asigna un valor para cada fila
- Se modifica la columna a NOT NULL

ALTER TABLE

[MODIFY (columna tipo [, columna tipo].....)]: Modifica una o más columnas existentes. Habrá que tener en cuenta:

Se puede aumentar la longitud de la columna en cualquier momento

Al disminuir la longitud de una columna con datos no se puede dar menor tamaño que el máximo valor almacenado

Si la columna es NULL en todas las filas de la tabla, se puede disminuir y modificar el tipo de dato

La opción MODIFY … NOT NULL sólo será posible cuando la tabla no contenga ninguna fila con valor nulo en la columna

ALTER TABLE

[DROP COLUMN (columna [, columna]...)]: Borra una columna de una tabla.

Recordad que no se pueden borrar todas las columnas de una tabla

Tampoco se pueden eliminar las claves primarias referenciadas por claves ajenas

ALTER TABLE

[ADD CONSTRAINT restricción]: Permita añadir una restricción

[DROP CONSTRAINT restricción]: Permite borrar una restricción

[DISABLE CONSTRAINT restricción]: Permita desactivar una restricción

[ENABLE CONSTRAINT restricción]: Permite activar una restricción

Ejemplos

ALTER TABLE TEJEMPLO ADD (categoria CHAR(1), importe NUMBER(4));

ALTER TABLE TEJEMPLO ADD (categoria CHAR(1) NOT NULL, importe NUMBER(4));

La tabla debería estar vacía, sino lo esta la restricción NOT NULL produciría un error.

ALTER TABLE TEJEMPLO MODIFY (categoria CHAR(1) NOT NULL, NOMBRE VARCHAR(5));

Podría dar error si la columna nombre tiene valores y son mayores de 5

ALTER TABLE TEJEMPLO DROP COLUMN (categoria);

ALTER TABLE

Ejemplos:

- Para añadir una restricción de valor único

```
ALTER TABLE TEMPLE ADD CONSTRAINT apellido_uq UNIQUE (apellido);
```

- Para añadir la opción no nula al apellido

```
ALTER TABLE TEMPLE ADD CONSTRAINT apell_nonulo CHECK (APELLIDO NOT NULL);
ALTER TABLE TEMPLE ADD CONSTRAINT emple_ck (NUM_DEP IN(10,20,30,40,50));
```

- Para borrar una restricción

```
ALTER TABLE TEMPLE DROP CONSTRAINT apellido_uq;
ALTER TABLE TEMPLE DROP CONSTRAINT SYS_C0095;
```

UT 4 -- TAREA 6

TAREA 6 – La de la escuela

Partiendo del siguiente modelo de datos :

ESCUELA
Código_Escuela
Nombre_Escuela
Domicilio_Escuela

TELEFONO_ESCUELA
Código_Escuela
Teléfono_Escuela

RESERVA
Número_Reserva
Fecha_Visita_Reservada
Hora_Visita_Reservada
Código_Escuela

TIPO_VISITA
Código_Tipo_Visita
Descripción_Tipo_Visita
Arancel_por_Alumno

RESERVA_TIPO_VISITA
Número_Reserva
Código_Tipo_Visita
Cantidad_Alumnos_Reservados
Cantidad_Alumnos_Reales
Código_Guía

GUIA
Código_Guía
Nombre_Guía
Apellido_Guía

RESERVA_POR_GRADO
Número_Reserva
Código_Tipo_Visita
Grado

1. Crear la tabla tescuela y definir su clave principal en la misma instrucción de creación. Continuar con tablas tguia, treserva y ttipo_visita.
2. Crear la tabla ttelefono_escuela con su clave principal. Define esta añadiendo la CONSTRAINT al final del script.
3. Crear la tabla treserva_por_grado con su clave principal. Hacer las correspondientes restricciones.
4. Crear la tabla treserva_tipo_visita con sus campos propios y los referenciados. Sin generar claves.
5. Completar el ejercicio anterior, con la creación de las claves correspondientes. Haz uso del comando ALTER TABLE. (FKs)
6. Hacer que no pueda haber dos escuelas con el mismo nombre.
7. Por defecto la fecha de visitas en la tabla de reservas será la del sistema
8. Como mínimo cada reserva tendrá 15 alumnos. Tabla treserva_tipo_visita.
9. Tanto el apellido como el nombre del guía no pueden ser nulos.
10. No se podrán repetir los apellidos.

UT 4 -- TAREA 7

TAREA 7 – La de la sencilla bbdd

Se trata de una sencilla base de datos relacional para una pequeña empresa de distribución. La base de datos almacena la información necesaria para implementar una pequeña aplicación de procesamiento de pedidos. Tenemos:

- Los *productos* de la empresa
- Los *clientes* que compran dichos productos
- Los *pedidos* remitidos por esos clientes
- Los *vendedores* que venden los productos a los clientes
- Las *oficinas* de ventas donde trabajan los vendedores.

Se pide hacer las sentencias SQL necesarias para crear las tablas según el modelo relacional dado y las restricciones indicadas Modelo relacional

TOFICINAS (oficina, ciudad, región, director (FK), objetivo, ventas)

TPEDIDOS (num_pedido, fecha_pedido, cliente (FK) vendedor (FK), fabricante (FK), producto (FK), cantidad, importe)

TCLIENTES (num_cliente, empresa, vendedor_cliente (FK), limite_credito)

TVENDEDORES (num_empleado, nombre, edad, oficina_vendedor (FK), puesto, contrato, director (FK), cuota, ventas)

TPRODUCTOS (id_fab, id_producto, descripción, precio, existencias)

1 Crear las tablas

Crear las cinco tablas según la descripción que se da a continuación.

NOTA: Previamente hay que ver el orden en que deben crearse para evitar los inconvenientes que puede causar la definición de claves ajenas sin haber definido previamente las tablas en las que son primarias.

-Tabla **TOFICINAS**

Campos:

OFICINA numérico (2 dígitos), no admite nulos, clave principal
CIUDAD alfanumérico 15 caracteres, no admite nulos
REGION alfanumérico 10 caracteres, no admite nulos, valor por defecto '***Este'***,
DIRECTOR numérico (3 dígitos), sí admite nulos
OBJETIVO numérico (9 dígitos, 2 decimales), sí admite nulos
VENTAS numérico (9 dígitos, 2 decimales), no admite nulos, valor por defecto 0.00

RENAME

Permite cambiar el nombre a una tabla, vista o un sinónimo.

Oracle invalida todos los objetos que dependan del objeto renombrado, como las vistas, los sinónimos que hacen referencia a la tabla renombrada.

Formato:

RENAME nombre_old TO nombre_new;

Ejemplo:

```
create synonym ALUM from ALUMNOS;      //sinónimo para tabla ALUMNOS
rename ALUMNOS to TALUMNOS;            // al renombrar la tabla, el sinónimo
                                       ALUM deja de funcionar
```

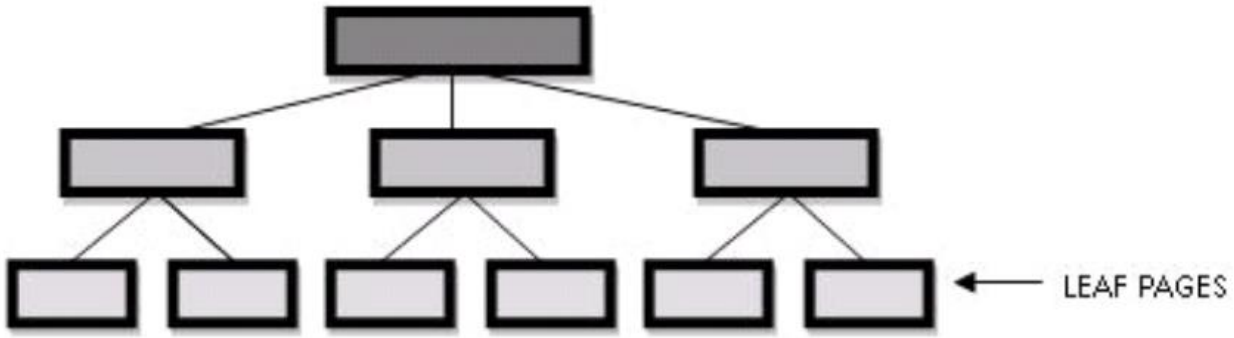
ÍNDICES

- Permiten acelerar el tiempo de respuesta en las consultas.
- Es un objeto de la base de datos que se asocia a una tabla y al que se asocia una o varias columnas de la tabla

Formato para crear:

CREATE INDEX nombre_índice ON nombre_tabla (columna [,columna]...)
[STORAGE clausula_almacenamiento]
[TABLESPACE nombre_tablespace]
[otras cláusulas];

Estructura interna de un índice:



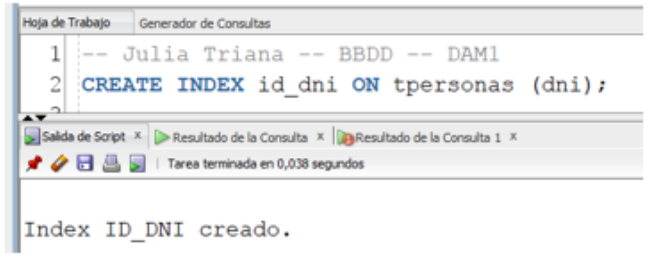
Formato para borrar:

DROP INDEX NombreIndice;

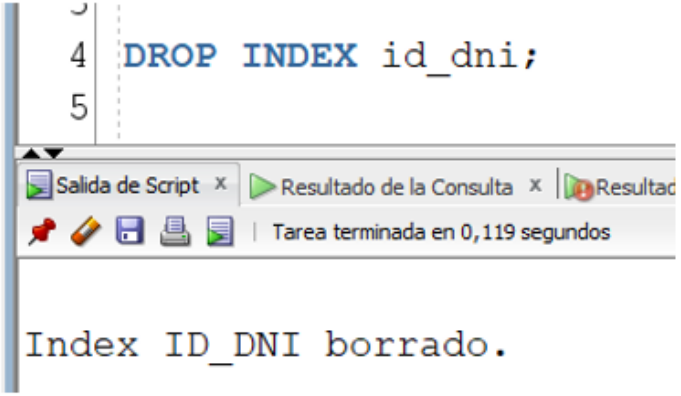
ÍNDICES: CREACIÓN Y BORRADO

Pero el formato básico es el siguiente:

CREATE INDEX nombreIndice ON nombreTabla (nombreColumna)

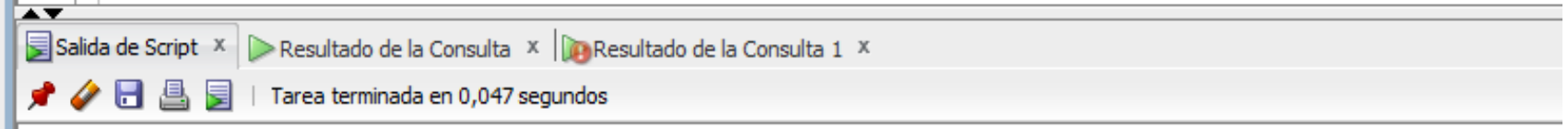


Se puede borrar un índice (DROP INDEX)



TRABAJANDO CON ÍNDICES

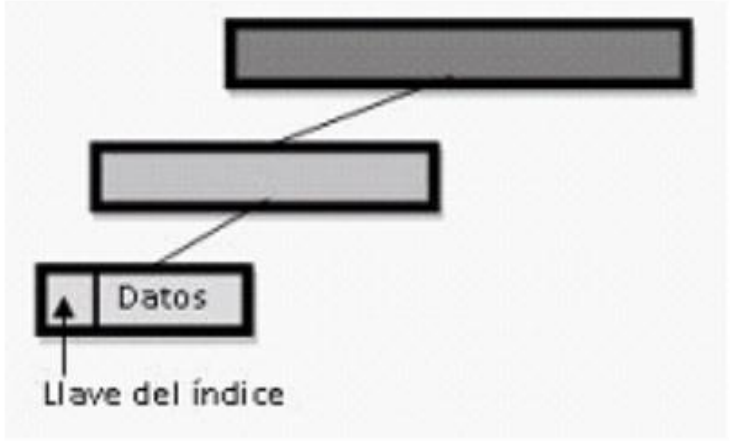
```
-- Julia Triana -- BBDD -- DAM1
-- La primera es borrar el índice y generarlo de nuevo
-- cuando se vuelva a necesitar. Dependiendo del volumen de
-- datos con los que trabajemos es siempre una opción viable.

-- La segunda es la siguiente:
-- "Deshabilitar" el índice
ALTER INDEX ID_DNI UNUSABLE;

-- "Habilitar" el índice
ALTER INDEX ID_DNI REBUILD;
```



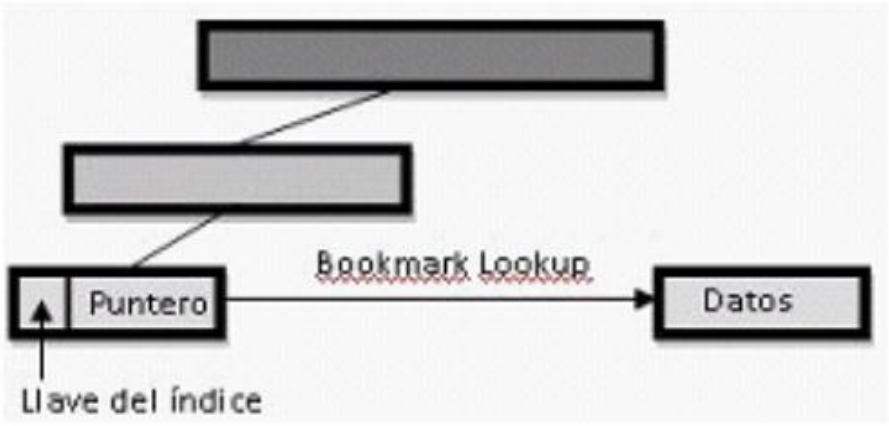

```
Index ID_DNI alterado.


Index ID_DNI alterado.
```

ÍNDICES

Búsqueda por clustered index:



Búsqueda por non-clustered index:



VISTA

VISTA: es una tabla lógica que permite acceder a la información de una o varias tablas

No contiene información por sí misma, sino que su información está basada en la que contienen otras tablas

Tiene por tanto la misma estructura que una tabla y se trata de igual forma

Una vista es una sentencia SQL

Al borrar una tabla la vista asociada deja de funcionar

CREACIÓN DE UNA VISTA

CREATE [OR REPLACE] **VIEW** nombre_vista (nom_col [,nom_col])]
AS consulta_sql;

Nombre_vista: nombre de la vista

Nom_col: nombre de las columnas de la vista, si no se especifican se adoptarán los mismos que los de la consulta

Consulta: determina las columnas y las tablas que aparecerán en la vista

OR REPLACE: crea de nuevo la vista si ya existía.

CREACION DE UNA VISTA

Aunque nos adelantamos en el concepto de consulta SQL (SELECT), nos centramos en las consultas más básicas para entender lo que es una vista. Ya en la unidad 5 al meternos en las consultas veremos la verdadera utilidad de las vistas.

```
CREATE VIEW vdep30 AS
        SELECT apellido, oficio, salario, dept_no
          FROM templeados
          WHERE dept_no = 30;

CREATE OR REPLACE VIEW vdep30 (ape, ofi, sal, dep) AS
            SELECT apellido, oficio, salario ,dept_no
             FROM templeados
             WHERE dept_no = 30;
```

consulta

consulta

SINÓNIMOS

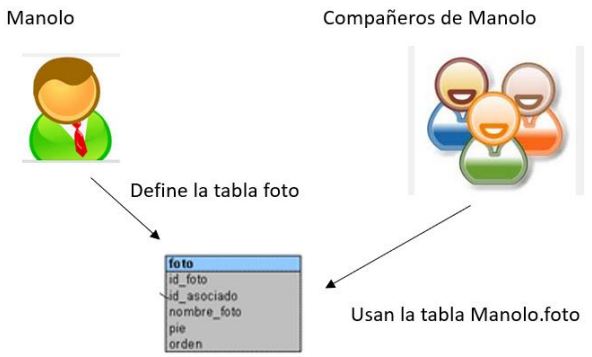
Nombre completo o totalmente cualificado de un objeto de la base de datos

Notación:

[[[servidor.] [base de datos] .] [nombre propietario] .] nombre objeto

Ejemplo, si el objeto fuese una tabla:

servidor.base de datos.usuario.tabla



Es un nombre que se puede dar a una tabla o una vista, permitiendo de esta forma **abreviar** a la hora de escribir el acceso a la misma.

Utilidad: Acceso a tablas de otro usuario

CREACIÓN DE SINÓNIMOS

Formato:

CREATE [public] SYNONYM nom_sinonimo
FOR [usuario.]nombre_tabla;

Public: hace que el sinónimo esté disponible para todos los usuarios. Solo puede ser usado por el DBA o usuarios con el privilegio CREATE PUBLIC SYNONYM

```
CREATE  SYNONYM temp_lrc FOR  templeados;
CREATE  SYNONYM tdepto_lrc FOR  tdepartamentos;
CREATE  SYNONYM tsal_lrc FOR  tsalarios;
CREATE  SYNONYM tpob_lrc FOR  tpoblaciones;
```

CREACIÓN DE SINÓNIMOS

Dependiendo de los roles del usuario (UT 5) podrá o no crear sinónimos.

```
SELECT USER
  FROM dual;
```

Salida de Script | Resultado de la Co | SQL | Todas las Filas Recu

	USER
1	DAM1

```
CREATE  SYNONYM temp FOR  templeados;
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2

Tarea terminada en 0,037 segundos

```
Error que empieza en la línea: 42 del comando :
CREATE  SYNONYM temp FOR  templeados
Informe de error -
ORA-01031: privilegios insuficientes
01031. 00000 -  "insufficient privileges"
*Cause:    An attempt was made to perform a database operation without
           the necessary privileges.
*Action:   Ask your database administrator or designated security
           administrator to grant you the necessary privileges
```

```
SELECT USER
  FROM dual;
```

Salida de Script | Resultado de la Cor | SQL | Todas las Filas Recu

	USER
1	SYSTEM

```
CREATE  SYNONYM temp FOR  templeados;
```

Salida de Script | Resultado de la Consulta

Tarea terminada en 0,068 segundos

```
Synonym TEMP creado.
```

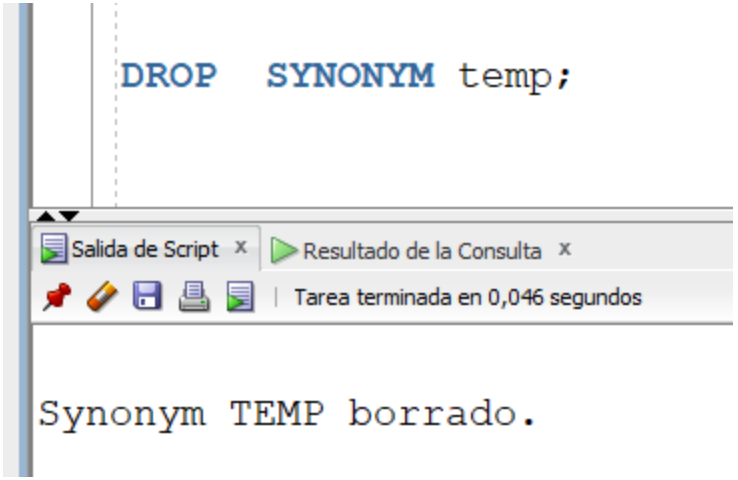
BORRADO DE SINÓNIMOS

Formato:

DROP [public] SYNONYM [usuario].nom_sinonimo ;

Al igual que antes, solo el DBA y los usuarios con el privilegio DROP PUBLIC SYNONYM pueden suprimir sinónimos públicos.

Los usuarios con el privilegio DROP ANY SYNONYM pueden borrar los sinónimos de otros usuarios



SINÓNIMOS DE UN USUARIO

Para ver los sinónimos, existe una vista **USER_SYNONYMS** que permite ver los sinónimos que son propiedad del usuario.

```
DESC user_synonyms;
```

Salida de Script | Resultado de la Consulta

Tarea terminada en 0,212 segundos

Nombre	¿Nulo?	Tipo
SYNONYM_NAME	NOT NULL	VARCHAR2(128)
TABLE_OWNER		VARCHAR2(128)
TABLE_NAME	NOT NULL	VARCHAR2(128)
DB_LINK		VARCHAR2(128)
ORIGIN_CON_ID		NUMBER

```
SELECT synonym_name, table_owner table_name
  FROM user_synonyms;
```

9	TEMP_LRC	SYSTEM
10	TDEPTO_LRC	SYSTEM
11	TSAL_LRC	SYSTEM
12	TPOB_LRC	SYSTEM

RENAME

Permite cambiar además del nombre de una tabla el de una vista o un sinónimo.

Oracle invalida todos los objetos que dependan del objeto renombrado, como las vistas, los sinónimos que hacen referencia a la tabla renombrada.

Formato:

RENAME nombre_old TO nombre_new;

```
-- Trabajando con tablas
DESC tprovincias;

RENAME tprovincias TO tprovincias_otra;

DESC tprovincias;
DESC tprovincias_otra;
```

```
Nombre    ¿Nulo?   Tipo
--------- -------- ------------
CD_PROV   NOT NULL NUMBER(2)
NOM_PROV           VARCHAR2(15)

Nombre de tabla cambiado.


ERROR:
ORA-04043: el objeto tprovincias no existe
Nombre    ¿Nulo?   Tipo
--------- -------- ------------
CD_PROV   NOT NULL NUMBER(2)
NOM_PROV           VARCHAR2(15)
```