Davante

8

BASE DE DATOS

El lenguaje SQL

ÍNDICE

ÍNDICE

OBJETIVOS

Familiarizarse con el lenguaje SQL y su importancia.

Conocer los diferentes sublenguajes de SQL.

Aprender a usar el lenguaje de descripción de datos (DDL).

Explorar las herramientas gráficas del SGBD para la descripción de datos.

/ 1. Introducción y contextualización práctica

El lenguaje SQL tiene sus raíces en la década de 1970, cuando IBM desarrolló el "Structured English Query Language" (SEQUEL) a partir de los trabajos de E.F. Codd, pionero en bases de datos relacionales. A finales de esa década, Oracle comenzó a usar SQL comercialmente, lo que ayudó a incrementar su popularidad.

En 1986, ANSI e ISO establecieron SQL como el estándar para bases de datos, lo que lo convirtió en uno de los lenguajes más importantes y usados en informática, principalmente en bases de datos.

Desde entonces, SQL ha evolucionado continuamente, con versiones como SQL89, SQL92, SQL2011, etc. Estas versiones se nombran según el año en que fueron lanzadas y muestran cómo SQL se ha adaptado para trabajar con otros lenguajes.

En este tema, estudiaremos los elementos y sentencias fundamentales de SQL. Aunque es fácil de aprender, su dominio requiere práctica constante, como cualquier otro lenguaje de programación.

Para practicar, se pueden usar herramientas gráficas como Oracle Live, que permite trabajar con SQL, crear bases de datos desde cero o a partir de modelos existentes, y seguir tutoriales.

```
CREATE TABLE tablaPrueba (
CampoPrueba1 number,
CampoPrueba2 varchar2(10));
```

Fig.1. Ejemplo de sentencia SQL.

A continuación, vamos a plantear un caso práctico a través del cual podremos aproximarnos de forma práctica a la teoría de este tema.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y Resolución del caso práctico.

Audio intro. "Vamos a practicar con SQL"

https://on.soundcloud.com/qYY5gVu93huUwDCR8

/ 2. El lenguaje SQL (Structured Query Language)

Es el lenguaje utilizado en la mayoría de los SGBD relacionales, de hecho, se ha convertido en uno de los lenguajes más empleados de la historia. Su uso es indispensable para cualquier trabajo o tarea relacionado con las bases de datos.

Cómo hemos visto en el tema de SGBD, SQL está compuesto por varios sublenguajes o tipos de lenguajes:

- **DDL (Data Definition Language):** Es el sublenguaje que permite definir la estructura de los datos.
- **DML (Data Manipulation Language)**: Es el sublenguaje que permite manipular los datos.
- **DCL (Data Control Language):** Es el sublenguaje que permite controlar los permisos de acceso a los datos.
- **DTL (Data Transaction Language):** Es el sublenguaje que permite controlar el ciclo de vida de las transacciones.

Podemos destacar las siguientes características principales de SQL:

- **Es un lenguaje declarativo**: Se asemeja al lenguaje natural ya que indica lo que se debe hacer, pero no cómo debe hacerse.
- **Soporta todas las funcionalidades necesarias para gestionar una base de datos,** lo que permite que sea utilizado por todos los perfiles que intervienen en ella: administradores, usuarios básicos o avanzados, etc.
- **Combinable con** otros lenguajes como **PHP, Java o Python.**

2.1. Elementos SQL

Dentro el lenguaje SQL debemos destacar los siguientes elementos:

ELEMENTO	DESCRIPCIÓN	EJEMPLOS
Comandos	Son las instrucciones para crear y gestionar la base de datos.	CREATE, ALTER, DROP, SELECT, INSERT, GRANT
Cláusulas	Criterios o condiciones para alterar el comportamiento de un comando.	WHERE, ORDER BY, FROM, GROUP BY, HAVING
Operadores	Permiten crear operaciones con cláusulas más complejas.	+,-,*,/,AND, OR
Funciones	Son expresiones predefinidas que generan resultados a partir de los valores de entrada.	AVG, COUNT, SUM(), MAX, MIN, SYSDATE
Literales	Son constantes o valores concretos.	Cualquier dato concreto: fechas, nombres
Metadatos	Identificación de un componente de la base de datos	Tablas, registros, atributos

Tabla 1. Elementos del lenguaje SQL.

2.2. Cláusulas SQL

En la siguiente tabla veremos el significado de las cláusulas más usadas.

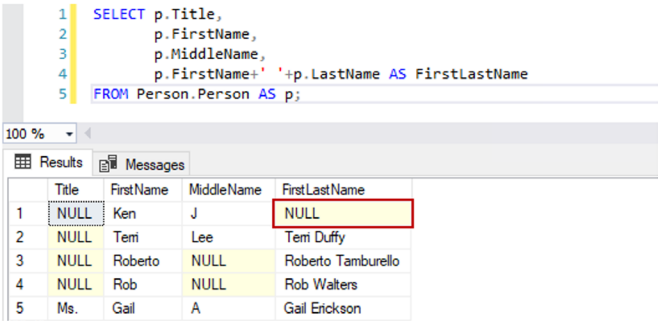
CLÁUSULA	DESCRIPCIÓN
FROM	Determina la tabla en la que se buscarán datos.
WHERE	Especifica las condiciones que debe cumplir la información que se desea extraer.
GROUP BY	Agrupa la información en base a un criterio.
HAVING	Permite indicar una condición que debe cumplir el grupo, similar a WHERE, pero sobre grupos.
ORDER BY	Ordena la información en base a un criterio.

Tabla 2. Cláusulas SQL.

2.3. Proceso de ejecución de sentencia SQL

Al ejecutarse, una instrucción SQL pasa un proceso compuesto por los siguientes pasos:

1. Comprobación de la sintaxis de la instrucción.
2. Si la sintaxis es correcta, se realiza la validación de los metadatos (tablas, registros, atributos, etc.) que contiene.
3. Si también son correctos, se realiza la optimización de la consulta para minimizar el uso de recursos necesarios.
4. Se procede a ejecutar la sentencia y mostrar los resultados.



```
SELECT p.Title,
       p.FirstName,
       p.MiddleName,
       p.FirstName+' '+p.LastName AS FirstLastName
FROM Person.Person AS p;
```

100 %

Results | Messages

	Title	FirstName	MiddleName	FirstLastName
1	NULL	Ken	J	NULL
2	NULL	Terri	Lee	Terri Duffy
3	NULL	Roberto	NULL	Roberto Tamburello
4	NULL	Rob	NULL	Rob Walters
5	Ms.	Gail	A	Gail Erickson

Fig.2. Ejemplo de sentencia SQL y sus resultados.

2.4. Componentes del entorno de ejecución

En un entorno de trabajo con SQL, existen varios componentes que intervienen en la ejecución de consultas:

- **Agente SQL**: Es el **componente que ejecuta y procesa las instrucciones SQL enviadas por el cliente**. Generalmente, es el software usado por el usuario para enviar peticiones SQL al servidor.
- **Implementación SQL**: Recibe, procesa y envía instrucciones SQL. Está formado por **el servidor SQL,** que procesa las instrucciones y devuelve los resultados (puede haber varios servidores) y **el cliente SQL,** que es el software que actúa de interfaz entre el agente SQL y el servidor SQL.

/ 3. Criterios de notación en SQL. Sintaxis

En SQL, se emplea una **notación específica para escribir las distintas sentencias**. Esta notación está formada por el siguiente conjunto de reglas básicas:

- Las **palabras clave** deben estar en **mayúsculas**, como por ejemplo los **comandos** o **cláusulas** (SELECT, FROM, WHERE, etc.).
- Los **corchetes** (**[]**) se utilizan para indicar que un atributo o parte de la sentencia es opcional.
- Las **llaves** (**{}**) encierran diferentes alternativas, separadas por el carácter "|".
- Los **puntos suspensivos** (**...**) indican que la sección anterior puede repetirse múltiples veces.

En la siguiente imagen, puedes ver un ejemplo de sentencia en la que aparecen combinadas las diferentes reglas de la notación SQL que hemos visto.

```
SELECT * | {DISTINCT columna | expresión [alias], ...}
FROM tabla1;
```

Fig.3. Ejemplo de notación de sentencia SQL.

En la siguiente imagen se muestra la **sintaxis** que se debe seguir **para construir una sentencia en SQL**.

Fig.4. Estructura de una sentencia SQL.

En la siguiente imagen, se muestra un ejemplo de una sentencia SQL para listar todas las filas de la tabla 'CLIENTES' donde el nombre es Javier.

```
SELECT * FROM CLIENTES WHERE NOMBRE = 'JAVIER';
```

Fig.5. Ejemplo de sentencia SQL.

Hay ciertas **normas que deben considerarse en SQL**:

- **No** se **diferencia** entre letras **mayúsculas** y **minúsculas**.
- Es necesario poner un **';' al final de cada instrucción**.
- Se pueden agregar **espacios** y **saltos de línea** dentro del comando para mejorar su legibilidad.
- Los **comentarios**, que no serán ejecutados y sirven para aclaraciones, se encierran entre '/' y '/'.

Enlaces de interés...

En el siguiente enlace podrás informarte sobre estándares de formato en SQL, sobre el uso de mayúsculas y minúsculas, sangrías, comentarios y paréntesis: https://solutioncenter.apexsql.com/es/estandares-de-formato-sql-mayusculas-y-minusculas-sangria-comentarios-parentesis/

/ 4. DDL: Lenguaje de definición de datos

El **Lenguaje de Definición de Datos (DDL)** permite realizar operaciones de **creación**, **modificación** y **eliminación** de **elementos de la base de datos**. Esto incluye la definición de estructuras para almacenar información, como **tablas**, **registros**, **índices**, **vistas**, y **claves**.

Por lo tanto, esta es la primera actividad que debe realizarse al implementar una base de datos, siendo fundamental entender y manejar estas operaciones. En la siguiente imagen se pueden ver los comandos DDL más frecuentes.

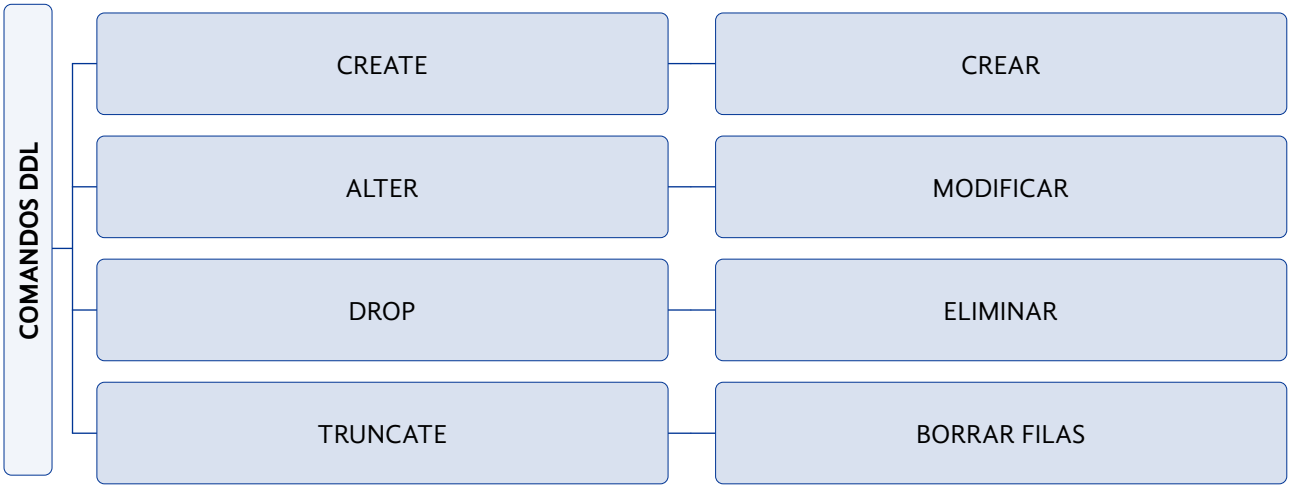


Fig.6. Comandos DDL más habituales.

Estas instrucciones tienen un gran impacto en los datos almacenados en la base de datos y, en la mayoría de las ocasiones, no se pueden revertir sin restaurar una copia de seguridad anterior. Por tanto, **generalmente, estas operaciones son realizadas por administradores o usuarios propietarios de las tablas.** Es esencial decidir cuidadosamente qué usuarios tendrán privilegios para realizar cambios de esta magnitud.

Los sistemas de gestión de bases de datos modernos **ofrecen herramientas gráficas que facilitan estas acciones** a través de interfaces intuitivas y fáciles de usar, que ejecutan el código SQL en segundo plano. Sin embargo, es muy importante comprender las instrucciones y su funcionamiento, independientemente de estas interfaces gráficas. Ahondaremos en el análisis de estas sentencias en próximos temas.

Enlaces de interés...

En el siguiente enlace, encontrarás un vídeo donde podrás ampliar información sobre DDL, su uso, y su importancia en la gestión de bases de datos: https://www.youtube.com/watch?v=XJb6qflbsx4

Al crear una tabla en una base de datos, es necesario definir los campos y **sus tipos de datos correspondientes**, así como establecer las **claves primarias** y **foráneas**, además de los **índices** o **restricciones** que puedan aplicarse.

/ 5. Tipos de datos del lenguaje. Funciones

La información que se quiere almacenar en la base de datos suele muy variada. Por ello, SQL permite crear las tablas y registros indicando, para cada campo el tipo de dato que más apropiado, de los que podemos destacar:

TIPO DE DATO	CARACTERÍSTICAS
Numérico (NUMBER)	En Oracle se representa como NUMBER y pueden almacenar tanto números enteros (int) como números decimales (float, double). Además, existen otros como: • INTEGER • BIGINT • FLOAT • DOUBLE • REAL
Texto	Secuencias de caracteres que pueden incluir números, letras o símbolos. Los tipos más comunes son CHAR (para textos de longitud fija) y VARCHAR (para textos de longitud variable). El tipo VARCHAR2 es similar a VARCHAR, pero permite limitar la longitud máxima de un texto.
Datos de gran tamaño	Para almacenar datos de gran tamaño existen los tipos **CLOB y BLOB**, que permiten almacenar hasta 4GB y **BFILE, que permite almacenar hasta 8GB**.
DATE	Almacena fechas sin las horas, minutos o segundos. La función SYSDATE proporciona la fecha actual.
TIMESTAMP	Es una ampliación de DATE. Guarda el valor del día, mes y año, así como horas, minutos y segundos.

Tabla 3. Tipos de datos más habituales y utilizados en SQL.

Para poder trabajar con los diferentes tipos de datos, a veces puede ser necesario realizar conversiones entre ellos. Por eso, S**QL incluye las siguientes funciones**:

- Conversión número a cadena: **TO_CHAR (nº,[formato])**.
- Conversión cadena a número: **TO_NUMBER (cadena ,[formato])**.
- Conversión fecha a cadena: **TO_CHAR (fecha,[formato])**.
- Conversión cadena a fecha: **TO_DATE (cadena,[formato])**.

```
SQL> SELECT TO_NUMBER('123.456',999.999) Con_Separador_Decimal from dual;

CON_SEPARADOR_DECIMAL
---------------------
               123456

SQL> select TO_NUMBER('12,7') Numero from dual;

    NUMERO
----------
      12,7
```

Fig.7. Ejemplo de uso de TO_NUMBER.

En el siguiente vídeo puedes ver algunos tipos de datos que no hemos mencionado:

Vídeo 1. "Tipos de datos SQL"
https://bit.ly/3MZ8WCw

/ 6. Creación, modificación y eliminación de bases de datos

Ahora veremos las operaciones necesarias para manejar una BD.

6.1. Creación de una base de datos

Crear una base de datos es una tarea complicada que requiere diferentes tareas relacionadas con el almacenamiento y la infraestructura. Para crear una base de datos, **se necesitan permisos de administración**, como los que tiene el usuario SYSDBA.

El proceso de creación puede diferir según el SGBD utilizado. En el caso del SGBD Oracle, este procedimiento puede ser complicado, **aunque las herramientas gráficas**, como Oracle Express Edition, **facilitan esta tarea**, como se muestra en el siguiente vídeo.

Vídeo 2: "Creación de tablas y base de datos en Oracle Express"
https://bit.ly/2OJjf0y

Para **crear una base de datos en SQL**, se emplea la instrucción **'CREATE DATABASE'**, junto con los parámetros necesarios, tal como se muestra en la siguiente imagen.

```
CREATE DATABASE prueba
LOGFILE prueba.log
MAXLOGFILES 25
MAXINSTANCES 10
ARCHIVELOG
CHARACTER SET WIN1234
NATIONAL CHARACTER SET UTF8
DATAFILE prueba1.dbf AUTOEXTEND ON MAXSIZE 500MB;
```

Fig.8. Ejemplo de creación de una base de datos con parámetros de Oracle.

Enlaces de interés...

En el siguiente enlace, tendrás acceso a la documentación proporcionada por Oracle en la que se explican los pasos a seguir para crear una base de datos primaria: https://docs.oracle.com/cd/E40205_01/html/E23227/chdicjgd.html#chdfeeij

6.2. Modificación de una base de datos

La instrucción utilizada para la modificación de la base de datos es **'ALTER DATABASE'**. Esta instrucción permite modificar propiedades generales, como el nombre, el conjunto de caracteres predeterminado, la definición de grupos, los archivos, entre otros.

```
ALTER DATABASE Cliente
MODIFY NAME = Clientes;
```

Fig.9. Ejemplo de utilización de ALTER.

```
ALTER DATABASE prueba
SET COMPATIBILITY_LEVEL ) {150 | 140 | 130 | 100}
```

Fig.10. Ejemplo de utilización del comando 'ALTER'. 'COMPATIBILITY_ LEVEL' hace referencia a la versión de SQL Server con la que la base de datos es compatible.

Para ejecutar esta sentencia, es necesario disponer del permiso 'ALTER' en la base de datos.

6.3. Eliminación de una base de datos

La instrucción utilizada para llevar a cabo la eliminación de una base de datos es **'DROP DATABASE'**.

```
DROP DATABASE Cliente;
```

Fig.11. Ejemplo de utilización del comando DROP.

/ 7. Creación de tablas

Es necesario conocer la información que se quiere almacenar en cada tabla antes de construirla. Esto nos permitirá elegir un nombre adecuado para la table, a las columnas (atributos), el tipo de dato más apropiado y su longitud, las claves, y las restricciones que se aplicarán. El concepto de restricción (*constraint*) lo estudiaremos en profundidad más adelante.

En la siguiente imagen, podemos ver un ejemplo de la sintaxis que se debe seguir en SQL para crear una tabla:

```
CREATE TABLE nombreTabla (
Columna1 Tipo_Dato [DEFAULT valor],
...
columnaN tipo_dato
[restricciones de tabla]
);
```

Fig.12. Sintaxis necesaria para crear una tabla en SQL.

A la hora de nombrar tablas, es necesario tener en cuenta algunas reglas. Escucha el siguiente audio para conocer las principales:

Audio 1. "Recomendaciones para nombrar tablas"

https://on.soundcloud.com/xtcXC8USWmxzf2NH7

En las siguientes imágenes se puede ver un ejemplo del código de creación de una tabla y el resultado obtenido tras su ejecución.

```
create table COMPRAS (
    ID_PRODUCTO number not null constraint compras_pk primary key,
    CONCEPTO varchar2(20),
    UNIDADES number,
    PRECIO number not null,
    FECHA date not null,
    PROVEEDOR varchar2(20)
);
/

alter table COMPRAS add constraint
compras_id_producto_uq unique (ID_PRODUCTO);
```

Fig.13. Ejemplo de código de creación de tabla.

ID_PRODUCTO	CONCEPTO	UNIDADES	PRECIO	FECHA	PROVEEDOR

Tabla 4. Tabla resultante de aplicar la sentencia de la imagen anterior.

En este ejemplo se crea la tabla 'COMPRAS', con varias columnas ('ID_PRODUCTO', 'CONCEPTO', 'UNIDADES', etc.). Además, se especifica que la columna ID_PRODUCTO será la clave primaria de la tabla.

Por otro lado, podemos observar que cada columna lleva el nombre y el tipo de dato, además de su longitud máxima (en el caso de *Varchar2*). También podemos ver varios campos definidos como 'NOT NULL', lo que les obliga a tener un valor no nulo.

Enlaces de interés...

En el siguiente enlace, podrás encontrar un ejemplo de creación de una tabla en SQL. Además, podrás encontrar también otros contenidos de interés relacionados con las bases de datos: https://www.srcodigofuente.es/curso-sql/crear-tablas-create-sql

8. Borrado y modificación de tablas

Otras dos operaciones imprescindibles en el mantenimiento de bases de datos son el borrado y la modificación de tablas.

8.1. Borrado de tablas

El borrado de tablas se hace a través del comando 'DROP TABLE'. Este comando provoca la eliminación total de la tabla. Es una operación peligrosa que debemos ejecutar con sumo cuidado, ya que por lo general es irreversible.

¡! Sabías que...

Algunos sistemas gestores de bases de datos como Oracle ofrecen la opción de habilitar una papelera de reciclaje, que permite recuperar una tabla eliminada, o incluyen una operación TRUNCATE, que solo elimina el contenido.

Es importante, a la hora de eliminar una tabla, tener en cuenta si existen columnas que sean claves foráneas. Si es así, conviene utilizar la opción **'CASCADE CONSTRAINTS'** para que se borren también las restricciones entre tablas.

8.2. Modificación de tablas

Con el comando **'ALTER TABLE'** podemos añadir o eliminar columnas, modificar el tipo de datos y propiedades, modificar *constraints*. A continuación, vamos a ver algunos ejemplos.

La siguiente imagen, muestra la adición y eliminación de columnas.

```
ALTER TABLE EMPLEADOS DROP (PUESTO);
ALTER TABLE EMPLEADOS ADD (fecha DATE);
```

Fig.14. Ejemplo uso ALTER TABLE.

En la siguiente imagen, se muestra como el comando **'ALTER TABLE' con 'MODIFY'** permite **modificar el tipo de dato de una columna**:

```
ALTER TABLE EMPLEADOS MODIFY(fecha TIMESTAMP);
```

Fig.15. Ejemplo de 'ALTER TABLE' y 'MODIFY'.

Otra utilización 'ALTER TABLE' sería definir una tabla como solo lectura (**READ ONLY**) o lectura/escritura (**READ WRITE**), o modificar el nombre de las columnas (**RENAME COLUMN**):

```
ALTER TABLE EMPLEADOS READ ONLY;
ALTER TABLE EMPLEADOS READ WRITE;
```

Fig.16. Ejemplo uso 'READ ONLY' y 'READ WRITE'.

```
ALTER TABLE EMPLEADOS RENAME COLUM
fecha TO fechaNueva
```

Fig.17. Ejemplo uso 'RENAME COLUMN'.

En esta imagen, se ejemplifica la creación de la tabla Empleados así:

```
create table EMPLEADOS (
    ID_EMPLEADO number not null constraint empleados_pk primary key,
    NOMBRE varchar2(10),
    APELLIDOS varchar2(20),
    SALARIO number(6),
    DIRECCION varchar2(20)
);
/

alter table EMPLEADOS add constraint empleados_id_empleado_uq unique (ID_EMPLEADO);
```

Fig.18. Ejemplo uso ALTER TABLE.

En la siguiente imagen se puede ver un ejemplo de cómo restringir el sueldo entre 100.000 y 300.000.

```
alter table EMPLEADOS modify (SALARIO number(6) CHECK
(SALARIO BETWEEN 100000 AND 300000));
```

Fig.19. Ejemplo modificación tablas.

/ 9. Restricciones

Las restricciones, o **constraints**, **establecen reglas que limitan el uso de ciertos datos**. Pueden aplicarse a una o varias columnas, o a toda la tabla. Si una restricción afecta a múltiples columnas, debe crearse como una restricción de tabla. Por ejemplo, una clave primaria compuesta requiere una restricción a nivel de tabla. La sintaxis se muestra en la siguiente imagen.

```
CONSTRAINT nombre_restriccion{
    PRIMARY KEY (columna1 [,columna2]...)
    | UNIQUE (columna1 [,columna2]...)
    | FOREIGN KEY (columna1 [,columna2]...)
       REFERENCES nombre_tabla (columna1 [,columna2]...)
       [ON DELETE {CASCADE | SET NULL}]
    | CHECK (condicion)
}
```

Fig.20. Sintaxis que se utiliza para crear restricciones de tabla.

Las restricciones pueden definirse de dos maneras:

- Al crear la tabla, utilizando el comando 'CREATE TABLE'.
- Después de crear la tabla, con el comando 'ALTER TABLE'.

En la imagen siguiente, al crear la tabla 'AUTORES', se han definido las restricciones 'PRIMARY KEY' (PK_autor_codigo), 'UNIQUE' (UQ_autor_nombre) y 'CHECK' (CK_autor_codigo). **Cada restricción lleva un nombre asociado, lo cual es recomendable.** Generalmente, reflejan el tipo de restricción ('PK', 'UQ', 'CK', etc.). Sin embargo, si no se asigna un nombre, el sistema genera uno por defecto, lo que complica su gestión, excepto para 'NOT NULL'.

```
CREATE TABLE AUTORES(
    codigo NUMBER(4) NOT NULL
    CONSTRAINT CK_autor_codigo
    CHECK (codigo >= 0),
    nombre VARCHAR2(50) NOT NULL,
    CONSTRAINT PK_autor_codigo
    PRIMARY KEY (codigo),
    CONSTRAINT UQ_autor_nombre
    UNIQUE (nombre)
);
```

Fig.21. Ejemplo de restricciones al crear una tabla.

9.1. NOT NULL

Cuando **el valor de un campo en una tabla no se conoce o no es necesario, se asigna 'NULL'**. 'NULL' no es lo mismo que 0; la base de datos los interpreta de manera diferente. **La restricción 'NOT NULL' garantiza que una columna no tenga valores 'NULL'**, por lo que **el campo siempre debe tener un valor.** Si no se define al crear la columna, la orden fallará. Esta restricción puede añadirse al crear la tabla o después. Generalmente, no se le asigna nombre porque se aplica a nivel de columna. En la imagen siguiente, se muestra un ejemplo del uso de restricciones 'NOT NULL'.

```
CREATE TABLE PERSONAS(
    id_persona INT NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    edad INT
);

ALTER TABLE PERSONAS(
MODIFY edad INT NOT NULL;
```

Fig.22. Ejemplo de utilización de restricciones tipo 'NOT NULL'.

9.2. UNIQUE

La restricción **'UNIQUE' garantiza que todos los valores en los campos de una columna sean únicos.** Así, no existirán dos filas con valores idénticos para una misma columna. Es aconsejable darle un nombre a la restricción al aplicarla a múltiples columnas en la tabla.

En la siguiente imagen, se muestra un ejemplo de esta restricción.

```
CREATE TABLE PERSONAS(
    id_persona INT NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    edad INT
    CONSTRAINT uc_persona UNIQUE(id_persona, apellidos)
);
```

Fig.23. Ejemplo restricción 'UNIQUE'.

9.3. PRIMARY KEY y FOREIGN KEY

La restricción **'PRIMARY KEY' identifica cada registro de la tabla de manera única**. Una tabla tendrá sólo una clave primaria, compuesta por una o varias columnas. Implica que se apliquen también las restricciones 'NOT NULL' y 'UNIQUE'. La sintaxis para especificar esta restricción es la misma que para 'NOT NULL' y 'UNIQUE'.

```
ALTER TABLE PERSONAS
DROP PRIMARY KEY;

ALTER TABLE PERSONAS(
ADD CONSTRAINT pk_personas PRIMARY KEY (id_persona, apellidos);
```

Fig.24. Ejemplo de modificación de restricción 'PRIMARY KEY'. Esta restricción también se puede eliminar con el comando 'DROP'.

La restricción **'FOREIGN KEY' define una clave foránea en una tabla que referencia la clave primaria 'PRIMARY KEY' de otra tabla.**

Enlaces de interés...

En el siguiente enlace, podrás observar y comprobar algunos ejemplos de uso de la restricción Foreign Key, que es un pilar fundamental para la consistencia de los datos y la relación entre las tablas: https://www.w3schools.com/sql/sql_foreignkey.as

9.4. CHECK y DEFAULT

La restricción **'CHECK' se emplea para imponer condiciones que los valores de una columna específica o en toda la tabla deben seguir**.

```
CREATE TABLE PERSONAS(
    id_persona INT NOT NULL UNIQUE,
    nombre VARCHAR(50) NOT NULL,
    edad INT CHECK (edad>=18)
);
ALTER TABLE PERSONAS(
ADD CONSTRAINT ck_personas CHECK (edad>=18);
```

Fig.25. Ejemplos de restricciones CHECK con los comandos CREATE TABLE y ALTER TABLE, con y sin nombre de restricción.

La restricción **'DEFAULT' define un valor por defecto para un campo**.

```
PAÍS varchar2(10) not null DEFAULT 'España'
```

Fig.26. Ejemplo de uso de la restricción 'DEFAULT'.

En el vídeo siguiente puedes ver otros ejemplos de creación de restricciones:

Vídeo 3. "Ejemplos de creación de restricciones"

https://bit.ly/3ifK7Ej

/ 10. Caso práctico 1: "Primeros pasos en Oracle Live SQL"

Planteamiento: Continuando con el caso de los temas anteriores, vamos a seguir asistiendo a nuestro amigo en la creación de una base de datos sencilla para gestionar su tienda.

Nudo: A lo largo de los últimos temas, hemos ido definiendo y construyendo la estructura de nuestra base de datos: tablas, atributos, relaciones, etc. Quizá haya llegado el momento de profundizar en el uso de Oracle para darle forma a nuestro planteamiento, ahora que estamos estudiando SQL.

Una herramienta que nos puede ayudar a seguir aprendiendo SQL es Oracle Live, en la que únicamente es necesario registrarse para poder acceder a múltiples contenidos. Uno de ellos es la posibilidad de crear tablas con varias opciones de manera gráfica para, posteriormente, comprobar qué sentencias SQL se utilizarían para ello. Pero, ojo, estas tablas no se guardan permanentemente, solo se pueden utilizar mientras tenemos la sesión abierta en la plataforma.

Vamos a probar a crear en la misma las tablas que hemos definido para la base de datos de nuestro amigo, y poder contrastar y estudiar, así, las expresiones SQL necesarias para ello. También nos ayudará a interiorizar y comprender mejor los aspectos vistos hasta ahora en el tema.

Desenlace: En la siguiente imagen se muestra una posible tabla de clientes, junto con el código generado para su creación:

Create Table

* Table Name: CLIENTES — Add a Trigger

Column Name	Data Type	Length	Nullable?	Primary Key?	Unique?
ID_CLIENTE	NUMBER	5	No	☑	☑
NOMBRE	VARCHAR2	20	No	☐	☐
APELLIDOS	VARCHAR2	50	Ye	☐	☐
DIRECCIÓN	VARCHAR2	50	Ye	☐	☐
TELÉFONO	NUMBER	9	No	☐	☐
FECHA_ALTA	VARCHAR2	10	Ye	☐	☐
DNI	VARCHAR2	9	No	☐	☐

Fig.27. Tabla de clientes.

A continuación podrás ver el código de creación de la tabla de clientes.

```
create table CLIENTES (
    ID_CLIENTE number not null constraint clientes_pk primary key,
    NOMBRE varchar2(20) not null,
    APELLIDOS varchar2(50),
    DIRECCIÓN varchar2(50),
    TELÉFONO number not null,
    FECHA_ALTA varchar2(10),
    DNI varchar2(9) not null
);
/
alter table CLIENTES add constraint clientes_id_cliente_uq
unique (ID_CLIENTE);
```

Fig.28. Código de creación de la tabla de clientes.

/ 11. Caso práctico 2: "Primeros pasos en Oracle Express"

Planteamiento: Oracle Express es un SGBD gratuito y potente que podemos utilizar para implementar la base de datos de la tienda de informática de nuestro amigo. Hemos visto durante el curso como instalar Oracle Express y en este tema hemos visto cómo crear una base de datos utilizando este sistema y como definir las tablas.

Nudo: Con la estructura de nuestras tablas y campos ya decidida, procederemos a crearlas en el SGBD Oracle Express. Primero, crearemos la base de datos global llamada "Tienda Informática". A continuación, utilizaremos lo aprendido en el tema, los vídeos de ayuda y el Caso Práctico 1 con Oracle SQL Live para crear las tablas definidas en los temas anteriores. Como diseñadores de la base de datos, este proceso nos permitirá poner en práctica nuestros conocimientos.

Desenlace: Podemos realizar todo este proceso utilizando la herramienta gráfica del sistema, tanto la creación de la base de datos como de los demás elementos. Este proceso puede verse en los vídeos del tema. En el vídeo 2 podemos ver el proceso para la creación de la base de datos y las tablas necesarias y en el vídeo 3 podemos ver cómo definir las restricciones necesarias para asegurar la integridad y consistencia de los datos almacenados y sus relaciones.

Fig.29. Oracle Express Edition.

/ 12. Resumen y resolución del caso práctico de la unidad

Hemos empezado el tema con una introducción al lenguaje SQL, sus sublenguajes, características, elementos y los pasos para la ejecución de consultas.

Para continuar, hemos visto en qué consiste el sublenguaje DDL y todas las opciones que nos ofrece para crear, modificar y eliminar tanto bases de datos como tablas, terminando con las diferentes restricciones de columna o de tabla y cómo definirlas.

A continuación podrás ver el esquema resumen del tema.

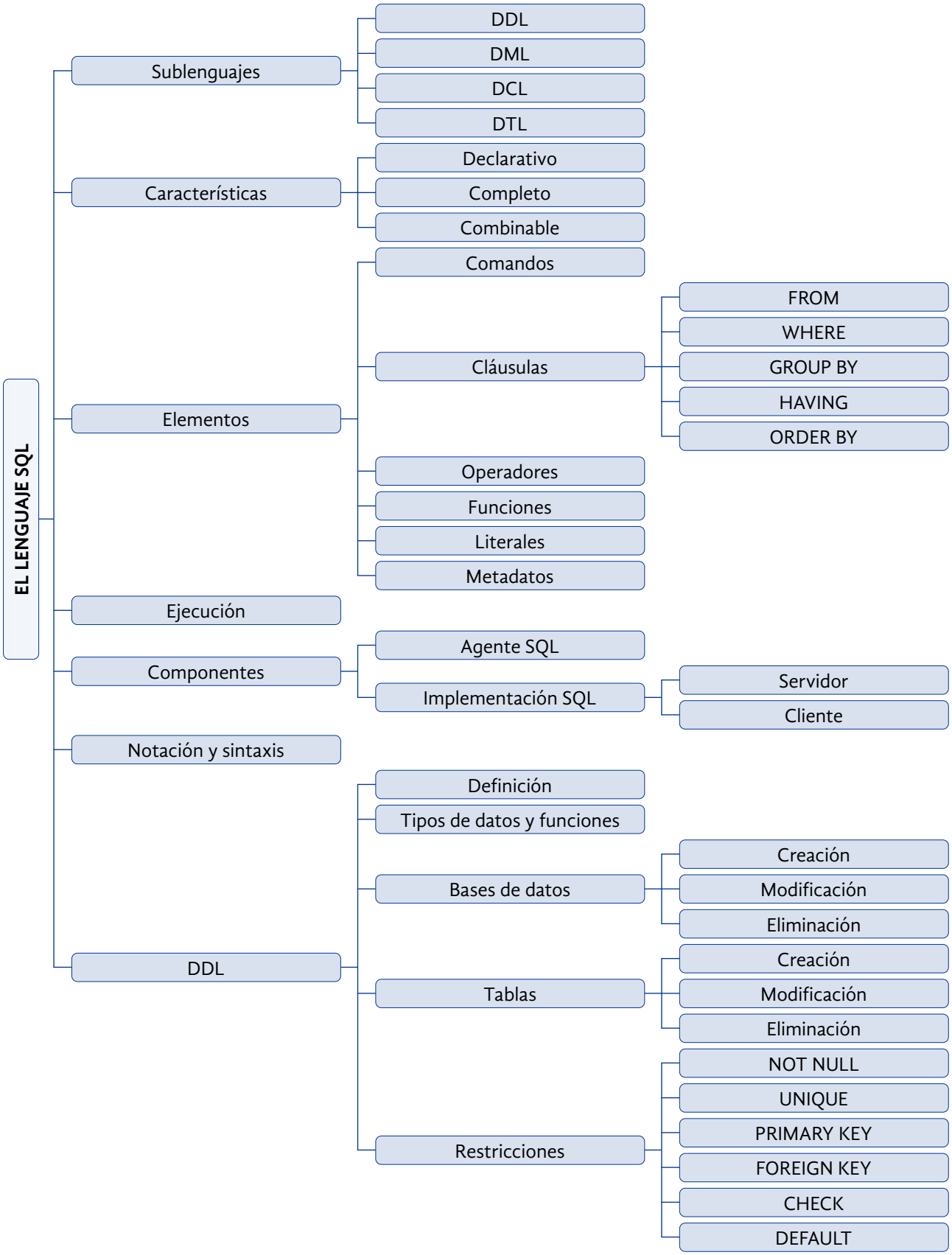


Fig.30. Esquema resumen del tema.

Resolución del caso práctico de la unidad

Para implementar la base de datos de la tienda de informática, comenzaríamos instalando Oracle Express.

Para los próximos pasos, podemos optar por utilizar las herramientas gráficas que nos ofrece Oracle Express o por escribir sentencias SQL directamente. Esta última opción nos permitirá obtener un conocimiento más profundo y práctico del lenguaje.

A continuación, debemos crear la base de datos utilizando la instrucción 'CREATE DATABASE'. Posteriormente, definiremos las tablas necesarias para la gestión de la información, como cliente, compra, productos y proveedores.

Debemos asegurarnos de incluir todas las restricciones necesarias, como claves primarias y únicas, para mantener la integridad de los datos.

Después, estableceremos las relaciones entre las tablas mediante claves foráneas. Esto garantizará que los datos estén vinculados correctamente y que las relaciones entre ellos se mantengan de manera coherente.

/ 13. Bibliografía

Connolly, T. y Begg, C. (2005). *Sistemas de Bases de Datos* (5ª ed.). Madrid, España. Addison – Wesley.

Elmasri, R., & Navathe, S. (2007). *Fundamentos de sistemas de bases de datos*. ADDISON WESLEY.

López, I., Castellano, M.J. y Ospino, J. (2011). *Bases de datos.* Garceta.

Oppel, A. (2009). *Databases A Beginner's Guide.* McGraw-Hill Education.

Pérez Marqués, M. (2016). *Administración básica de bases de datos con Oracle 12c SQL*. Madrid, España: Alfaomega.

Sánchez, G. C. (2001). *Sistemas gestores de bases de datos.* Paraninfo.