Curso
Desarrollo de aplicaciones móviles Android Trainee

Módulo 2:

Lenguaje de consultas a una base de datos (4 unidades)

TALENTO DIGITAL
INTELIGENCIA HUMANA

AIEP

Instituto Profesional AIEP Spa.
Dirección Nacional de Educación Contínua
Programa Talento Digital 2021

• • • Temario de unidades

1.1) Las Bases de Datos Relacionales

El rol de las bases de datos relacionales

Una base de datos relacional es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para implementar bases de datos ya planificadas. Permiten establecer interconexiones (relaciones) entre los datos (que están guardados en tablas), y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre: "Modelo Relacional". Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma.

· Una base de datos relacional se compone de varias tablas o relaciones.
· No pueden existir dos tablas con el mismo nombre ni registro.
· Cada tabla es a su vez un conjunto de registros (filas y columnas).
· La relación entre una tabla padre y un hijo se lleva a cabo por medio de las llaves primarias y foráneas.
· Las llaves primarias son la llaves principal de un registro dentro de una tabla y éstas deben cumplir con la integridad de datos.
· Las llaves foráneas se colocan en la tabla hija, contienen el mismo valor que la llaves primaria del registro padre; por medio de éstas se hacen las relaciones.

Características de un RDBMS

Un **sistema de gestión de bases de datos relacionales (RDBMS)** es un programa que te permite crear, actualizar y administrar una base de datos relacional. La mayoría de los RDBMS comerciales utilizan el lenguaje de consultas estructuradas (SQL) para acceder a la base de datos, aunque SQL fue inventado después del desarrollo del modelo relacional y no es necesario para su uso.

El RDBMS se refiere a la base de datos programa sí mismo. Es el software que ejecuta consultas sobre los datos, incluida la adición, actualización y búsqueda de valores. Un RDBMS también puede proporcionar una representación visual de los datos. Por ejemplo, puede mostrar datos en tablas como un hoja de cálculo, lo que le permite ver e incluso editar valores individuales en la tabla. Algunos programas RDMBS le permiten crear formularios que pueden agilizar el ingreso, la edición y la eliminación de datos.

Las aplicaciones mas conocidas incluyen a Oracle Database, MySQL, Microsoft SQL Server e IBM DB2. Algunos de estos programas admiten bases de datos no relacionales, pero se utilizan principalmente para la gestión de bases de datos relacionales.

Conociendo las herramientas para consultar una base de datos

La herramienta SQL o Editor de consultas nos permite ejecutar sentencias DML (Lenguaje de Manipulación de Datos) y DDL (Lenguaje de definición de datos, solo para usuarios con permisos administrativos).

Principalmente se divide en 3 secciones:

· Barra de herramientas o contenedor Ribbon
· Panel de herramientas
· Editor de consultas

Instalando Oracle SQL Developer

Oracle SQL Developer es una interfaz gráfica de usuario que permite a los usuarios y administradores de bases de datos usarla como una herramienta de productividad cuyo objetivo principal es ayudar al usuario final a ahorrar tiempo y maximizar el retorno de la inversión en el paquete de tecnología de Oracle Database.

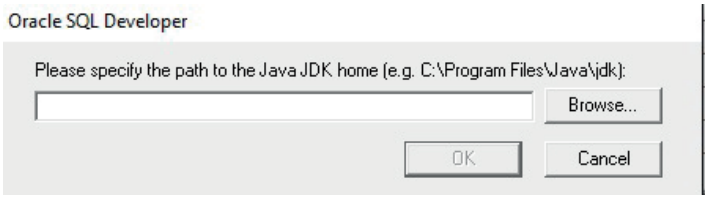
Este gestor es el equivalente de Oracle como pgAdmin III a Postgress o phpMyadmin para

MYSQL. Con el podremos realizar funcionalidades básicas como crear tablas, columnas, seleccionar tipos, insertar, modificar y eliminar filas entre otras de sus múltiples funcionalidades.

Esta herramienta la usaremos para gestionar nuestra base de datos de Oracle, no es obligatorio instalarla para poder trabajar con Oracle Database 11g Express Edition u otra versión de la base de datos de Oracle.

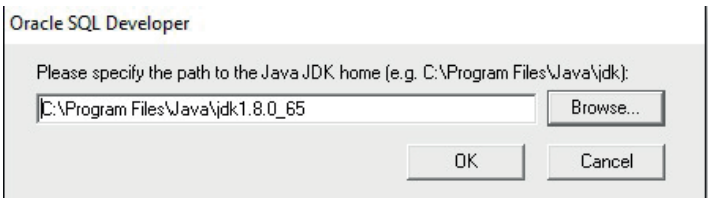
Una vez descargado (http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html) arrancamos el programa, para ello nos dirigimos a la carpeta descargada y pulsamos el ejecutable **sqldeveloper.exe**, automáticamente se lanzará la aplicación.

La primera vez que la arranquemos nos saldrá la siguiente imagen:

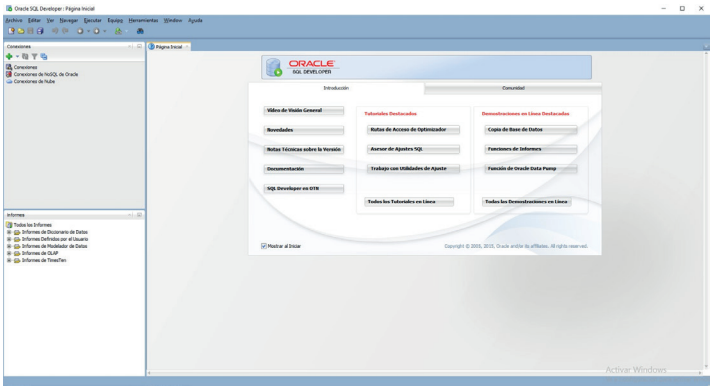


Buscamos la ruta donde tengamos instalado el software **JDK**, si no lo tenemos instalado debes descargarlo e instalarlo: **Revisar Modulo 1** - Unidad 2

Una vez tengamos la carpeta localizada pulsamos OK y seguidamente se abrirá el programa.

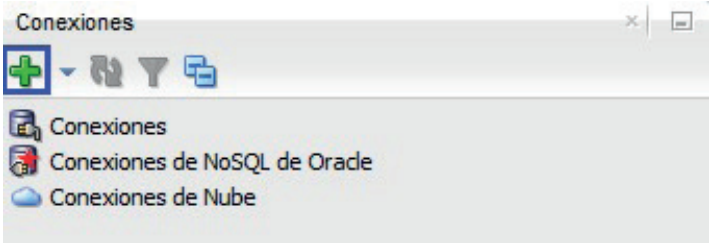


Siempre que se quiera utilizar **SQL Developer** debemos acceder mediante el ejecutable sqldeveloper.exe.
Cuando arranque SQL Developer visualizaremos la siguiente pantalla:

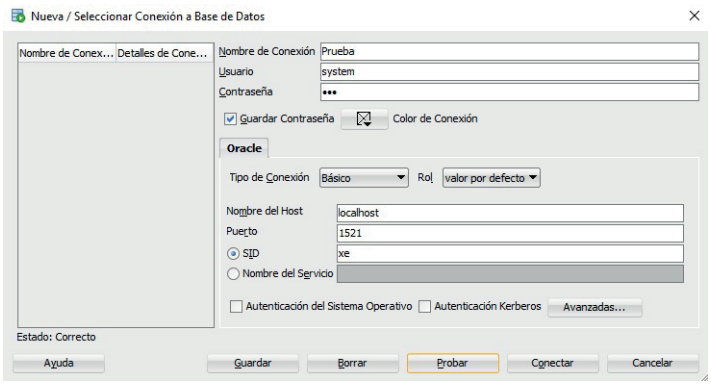


Creando una conexión a la base de datos

Para comenzar a trabajar lo primero que debemos hacer es conectarnos a nuestra base de datos, para ello pulsaremos al **icono** + y se abrirá un panel, a continuación deberemos rellenar los **datos** que se requieren para realizar la **conexión.**



La ventana emergente donde nos pedirá los datos de conexión será la siguiente:



Ahora que ya tenemos creada la conexión podemos trabajar sobre la base de datos de Oracle.

Los principales objetos de una base de datos

Una base de datos Oracle está compuesta principalmente por tres partes:

Memoria

Puede ser memoria asociada a los procesos de usuario (PGA – Process Global Area) y sería propia de cada cliente Oracle, o por memoria asociada al servidor Oracle (SGA – System Global Area).

La SGA son buffers que Oracle toma del sistema operativo cuando arranca la base de datos. La porción de memoria, que toma inicialmente y como lo distribuye en sus diferentes componentes, está determinado por el fichero de inicialización de la base de datos. En la SGA se sitúan datos, información del diccionario de datos, gestión de bloqueos, etc.

Procesos

Existen dos tipos de procesos:
· Procesos de usuario, sqlplus, sqlforms
· Procesos background o procesos de Oracle como DBWR, LGWR

Ficheros de Datos

Existen varios tipos de ficheros:

· Ficheros de Parametros: spfile/pfile, listener, tnsnames.

· Ficheros de Control (control files): Contienen información para poder arrancar la base de datos, su nombre, fecha de creación, ficheros asociados a la base de datos.

· Ficheros de Datos (datafiles): Son los ficheros donde se almacena la información de la base de datos.

Redo LOG: Sirven para mantener la consistencia en caso de fallo.

Una base de datos contempla varios aspectos:

· Estructuras: Almacenan la información de una base de datos. El acceso a las estructuras y los datos contenidos en ellas, se realiza mediante operaciones.
· Operaciones: Son las acciones que nos permiten acceder y manipular los datos y las estructuras. Las operaciones en una base de datos suelen ajustarse a una serie de reglas de integridad predefinidas.
· Reglas de Integraidad: Son las leyes que rigen las operaciones que permiten acceder a los datos y manipularlos. Sirven para proteger los datos y las estructuras que los contienen.

1.2) Consultando información de una tabla

Una consulta sirve para extraer información de una base de datos. Permite manipular datos: agregar, eliminar y cambiar. Así es como usaremos esta palabra. Para esto se debe escribir la consulta basándose en un conjunto de códigos predefinidos, de modo que la base de datos pueda entender la instrucción. Nos referimos a este código como el lenguaje de consulta SQL.

El Lenguaje Estructurado de Consultas SQL

La programación SQL permite interactuar con una base de datos. El lenguaje de consulta estructurado (SQL) es el lenguaje de base de datos más implementado y valioso para cualquier persona involucrada en la programación informática o que usa bases de datos para recopilar y organizar información.

La programación SQL se puede usar para compartir y administrar datos, en particular la información organizada en tablas que se encuentra en los sistemas de administración de bases de datos relacionales.

Mediante el uso de SQL, se puede:

· Consultar, actualizar y reorganizar datos.
· Crear y modificar la estructura de un sistema de base de datos.
· Controlar el acceso a sus datos.

Aunque pueda parecer algo similar al funcionamiento de una hoja de cálculo, el objetivo de SQL es diferente, ya que permite compilar y administrar datos en volúmenes mucho mayores. Mientras que las hojas de cálculo pueden volverse complicadas con demasiada información que llena demasiadas celdas, las bases de datos SQL permiten gestionar hasta miles de millones de celdas de datos.

Recuperando información de una tabla

Una consulta funciona de la misma manera en que hacemos una solicitud en nuestro lenguaje nativo; es necesario darle sentido al código utilizado en cualquier lenguaje de consulta. Ya sea SQL o cualquier otro, tanto el usuario como la base de datos pueden intercambiar información en cualquier momento, siempre que "hablen" el mismo lenguaje.

Las operaciones básicas de manipulación de datos que podemos realizar con SQL se les denomina operaciones **CRUD** (de **C**reate, **R**ead, **U**pdate and **D**elete)

Hay cuatro instrucciones para realizar estas tareas:
• **INSERT:** Inserta filas en una tabla. Se corresponde con la "C" de CRUD.
• **SELECT:** muestra información sobre los datos almacenados en la base de datos. Dicha información puede pertenecer a una o varias tablas. Es la "R".
• **UPDATE:** Actualiza información de una tabla. Es, obviamente, la "U".
• **DELETE:** Borra filas de una tabla. Se corresponde con la "D".

Para realizar consultas sobre las tablas de las bases de datos disponemos de la instrucción SELECT. Con ella podemos consultar una o varias tablas. Es sin duda el comando más versátil del lenguaje SQL. Existen cláusulas asociadas a la sentencia SELECT (GROUP BY, ORDER, HAVING, UNION). También es una de las instrucciones en la que con más frecuencia los motores de bases de datos incorporan cláusulas adicionales al estándar

El resultado de una consulta SELECT nos devuelve una tabla lógica. Es decir, los resultados son una relación de datos, que tiene filas/registros, con una serie de campos/columnas. Igual que cualquier tabla de la base de datos. Sin embargo esta tabla está en memoria mientras la utilicemos, y luego se descarta. Cada vez que ejecutamos la consulta se vuelve a calcular el resultado.
La sintaxis básica de una consulta SELECT es la siguiente (los valores opcionales van entre corchetes):

SELECT [ALL / DISTINC] [*] / [ListaColumnas_Expresiones] **AS** [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [**ASC / DESC**]

SELECT:
Permite seleccionar las columnas que se van a mostrar y en el orden en que lo van a hacer. Simplemente es la instrucción que la base de datos interpreta como que vamos a solicitar información.

ALL/DISTINCT:
ALL es el valor predeterminado, especifica que el conjunto de resultados puede incluir filas duplicadas. Por regla general nunca se utiliza.

DISTINCT especifica que el conjunto de resultados sólo puede incluir filas únicas. Es decir, si al realizar una consulta hay registros exactamente iguales que aparecen más de una vez, éstos se eliminan. Muy útil en muchas ocasiones.

Nombre de Campos:
Se debe especificar una lista de nombres de campos de la tabla que nos interesan y que por tanto queremos devolver. Normalmente habrá más de uno, en cuyo caso separamos cada

nombre de los demás mediante comas.
Se puede anteponer el nombre de la tabla al nombre de las columnas, utilizando el formato Tabla.Columna. Además de nombres de columnas, en esta lista se pueden poner constantes, expresiones aritméticas, y funciones, para obtener campos calculados de manera dinámica.

Si queremos que nos devuelva todos los campos de la tabla utilizamos el comodín "*" (asterisco).
Los nombres indicados deben coincidir exactamente con los nombre de los campos de la tabla

AS:
Permite renombrar columnas si lo utilizamos en la cláusula SELECT, o renombrar tablas si lo utilizamos en la cláusula FROM. Es opcional. Con ello podremos crear diversos alias de columnas y tablas. Enseguida veremos un ejemplo.

FROM:
Esta cláusula permite indicar las tablas o vistas de las cuales vamos a obtener la información. De momento veremos ejemplos para obtener información de una sola tabla.

WHERE:
Especifica la condición de filtro de las filas devueltas. Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones. Lo habitual es utilizar esta cláusula en la mayoría de las consultas.

Condiciones:
Son expresiones lógicas a comprobar para la condición de filtro, que tras su resolución devuelven para cada fila TRUE o FALSE, en función de que se cumplan o no. Se puede utilizar cualquier expresión lógica y en ella utilizar diversos operadores como:

- **>** (Mayor)
- **>=** (Mayor o igual)
- **<** (Menor)
- **<=** (Menor o igual)
- **=** (Igual)
- **<>** o **!=** (Distinto)

IS [NOT] NULL (para comprobar si el valor de una columna es o no es nula, es decir, si contiene o no contiene algún valor)

ORDER BY:
Define el orden de las filas del conjunto de resultados. Se especifica el campo o campos (separados por comas) por los cuales queremos ordenar los resultados.

ASC/DESC:
ASC es el valor predeterminado, especifica que la columna indicad en la cláusula ORDER BY se ordenará de forma ascendente, o sea, de menor a mayor. Si por el contrario se especifica DESC se ordenará de forma descendente (de mayor a menor).

Consultas utilizando la llave primaria

Las consultas a base de datos pueden ser de forma general donde obtenemos una gran cantidad de datos y puede ser selectiva dependiendo del tiempo de parámetro que entreguemos en nuestra consulta SQL. Esta realizará la evaluación y filtrará a la respuesta o retorno final los datos que coincidan con la solicitud.

Paises

id_pais (int)	nom_pais (varchar)	continente	tipo_gobierno
9	Perú	America del Sur	Republica Democrática
10	Chile	America del Sur	Republica Democrática
11	España	Europa	Monarquía Parlamentaria
12	Estados Unidos	America del Norte	Republica Federal

SELECT nom_pais FROM Paises WHERE id_pais = 10
Resultado Consulta: Chile

Consultas utilizando condiciones de selección

Las condiciones de selección permiten filtrar los registros con el fin de recuperar solamente aquellos que cumplan unas condiciones preestablecidas.

SELECT nom_pais
FROM Paises
WHERE tipo_gobierno = "Republica Democratica"
AND continente = "America del Sur"

Resultado Consulta: Perú, Chile

Utilización de funciones en las consultas

SQL tiene funciones incorporadas para hacer cálculos sobre los datos. Las funciones se pueden dividir en dos grupos (existen muchas mas, que dependen del sistema de bases de datos que se utilice):

Funciones agregadas SQL, devuelven un sólo valor, calculado con los valores de una columna.
· AVG() - La media de los valores
· COUNT() - El número de filas
· MAX() - El valor más grande
· IN() - El valor más pequeño
· SUM() - La suma de los valores
· GROUP BY - Es una sentencia que va muy ligada a las funciones agregadas

Funciones escalares SQL, devuelven un sólo valor basándose en el valor de entrada
· UCASE() - Convierte un campo a mayúsculas
· LCASE() - Convierte un campo a minúsculas
· MID() - Extrae caracteres de un campo de texto
· LEN() - Devuelve la longitud de un campo de texto
· NOW() - Devuelve la hora y fecha actuales del sistema
· FORMAT() - Da formato a un formato para mostrarlo

Consultas de selección con funciones de agrupación

La sentencia GROUP BY se utiliza junto con las funciones agregadas
para agrupar en un resultado una o más columnas.

SELECT nombreColumna, funcion_agregada(nombreColumna)
FROM nombreTabla
WHERE nombreColumna operador valor
GROUP BY nombreColumna;

Si tenemos las tablas **Productos, Pedidos y Clientes:**

Productos:

ProductoID	Nombre Producto	Descripción	Precio	Stock
1	Camiseta	Camiseta negra simple de talla única	10	16
2	Pantalón	Pantalón largo azul tipo chino	20	24
3	Gorra	Gorra azul con el logo de los Yankees	15	32
4	Zapatillas	Zapatillas de running de color blanco y verde	35	13

Pedidos:

PedidoID	ClienteID	Factura
234	4	160
235	5	48
236	12	64
237	4	92

Clientes:

cliente

id_cliente (PK)
nom_cliente

Ventas

id_venta (PK)
id_cliente (FK)
valor_venta

ClienteID	Nombre Cliente	Contacto
1	Lorena Higgins	456443552
2	Raúl González	445332221
3	Carmen Smith	488982635
4	Fernando Stewart	412436773

Vamos a obtener el número de pedidos realizados por cada cliente:

```
SELECT Clientes.NombreCliente, Count(Pedidos.PedidoID) AS NumeroPedidos FROM Pedidos
LEFT JOIN Clientes
ON Pedidos.ClienteID=Clientes.ClienteID
GROUP BY NombreCliente;
```

Podemos utilizar GROUP BY en más de una columna:

```
SELECT Clientes.NombreCliente, Productos.NombreProducto,
COUNT (Pedidos.PedidoID) AS NumeroPedidos
FROM ((Pedidos
INNER JOIN Clientes ON Pedidos.ClienteID=Clientes.ClienteID)
INNER JOIN Productos ON Pedidos.ProductoID=Productos.ProductoID)
GROUP BY NombreCliente, NombreProducto;
```

1.3) Consultando información relacionada en varias tablas

Qué es un modelo de datos y cómo leerlo

Un modelo de datos es un conjunto de herramientas conceptuales para la descripción de los datos y las relaciones entre ellos, su semántica y las restricciones de consistencia. Ahora bien, el modelo de datos relacional facilita el trabajo del programador en comparación con otros modelos. Sin embargo, primero se desarrollará un modelo basado en Entidad-Relación el cual después será trasformado finalmente en el Relacional, ya que E-R sirve para modelar el problema y reglas de negocio.

Este modelo es solamente un método que se aprovecha para diseñar los esquemas que posteriormente se deben implementar en la Base de datos. Este modelo se representa a través de diagramas y está formado por varios elementos:

Entidad: Cada entidad representa cosas y objetos ya sean reales o abstractos que se diferencian entre sí. En un diagrama las entidades se representan con rectángulos.

Atributos: Los atributos definen las características de las entidades, son las propiedades de cada una. Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha...). Cada una de las columnas de una tabla. En un diagrama los atributos se representan con círculos que descienden de la entidad.

Tupla: La tupla son elementos formados por una fila de una tabla. Cada fila está de la entidad estaría compuesta por los atributos de la entidad correspondiente.

Relación: Una relación es lo que nos permite definir qué tipo de dependencia se debe dar entre entidades, es decir nos permite decir que ciertas entidades comparten ciertos atributos de manera necesaria.

Enunciado: los empleados de un taller tienen un cargo asociado.

Es decir, si un atributo de la entidad "Empleados" especifica que cargo tiene en el taller. El cargo que ejerce debe coincidir con el que existe en la tabla de "cargos".

Llaves en las relaciones: Una llave es un atributo de una entidad, a la cual hace distinguir a esta entidad de los demás registros. Primeramente, debemos saber cuáles son los tipos de llaves que existen en una entidad y como se pueden relacionar con otras entidades.

Una forma de representar en diagramas o textos la llave primaria es poniendo "#" como pre-

fijo de la llave primaria (PK) o subrayando el nombre de la llave primaria.
Ejemplo:

· Estudiante (#ID, Nombre, Apellido)
· Estudiante (ID, Nombre, Apellido)

Llave primaria: Es un atributo (o columna) que restringe y distingue a las tuplas para que no se repitan en la misma entidad. La clave primaria es única. En algunos casos puede ser dos atributos lo cual conlleva al siguiente concepto.

Llave foránea: es el atributo de una entidad que existe como dependencia en otra entidad, cuyos valores en las tuplas deben coincidir con valores de una llave que debe ser primaria de las tuplas de otra relación.

Consultas de selección con tablas relacionadas

Los diseños de la base de datos están estrechamente relacionados con las relaciones de la base de datos, la asociación entre dos columnas en una o más tablas. Los tipos de relaciones no deben confundirse con los tipos de datos en SQL, siendo conceptos distintos.

Las relaciones se definen sobre la base de columnas de clave coincidentes.En SQL, estas relaciones se definen utilizando restricciones de clave principal a clave externa.

Se crea un enlace entre dos tablas donde la clave principal de una tabla se asocia con la clave externa de otra tabla utilizando las relaciones de la base de datos.

Integridad referencial

La integridad referencial es un sistema de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que los registros de tablas relacionadas son válidos y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.

Entre dos tablas de cualquier base de datos relacional pueden haber dos tipos de relaciones, relaciones uno a uno y relaciones uno a muchos:

Relación Uno a Uno: Cuando un registro de una tabla sólo puede estar relacionado con un único registro de la otra tabla y viceversa.

Por ejemplo:
tenemos dos tablas una de profesores y otra de departamentos y queremos saber qué profesor es jefe de qué departamento, tenemos una relación uno a uno entre las dos tablas ya que un departamento tiene un solo jefe y un profesor puede ser jefe de un solo departamento.

Relación Uno a Varios: Cuando un registro de una tabla (tabla secundaria) sólo puede estar relacionado con un único registro de la otra tabla (tabla principal) y un registro de la tabla principal puede tener más de un registro relacionado en la tabla secundaria, en este caso se suele hacer referencia a la tabla principal como tabla 'padre' y a la tabla secundaria como tabla 'hijo', entonces la regla se convierte en 'un padre puede tener varios hijos pero un hijo solo tiene un padre (regla más fácil de recordar).

Por ejemplo:
Tenemos dos tablas una con los datos de diferentes poblaciones y otra con los habitantes, una población puede tener más de un habitante, pero un habitante pertenecerá (estará empadronado) en una única población. En este caso la tabla principal será la de poblaciones y la tabla secundaria será la de habitantes. Una población puede tener varios habitantes pero un habitante pertenece a una sola población. Esta relación se representa incluyendo en la tabla ‹hijo› una columna que se corresponde con la clave principal de la tabla ‹padre›, esta columna es lo denominamos clave foránea (o clave ajena o clave externa).

Una clave foránea es pues un campo de una tabla que contiene una referencia a un registro de otra tabla. Siguiendo nuestro ejemplo en

la tabla habitantes tenemos una columna población que contiene el código de la población en la que está empadronado el habitante, esta columna es clave ajena de la tabla habitantes, y en la tabla poblaciones tenemos una columna codigo de poblacion clave principal de la tabla.

Query's anidadas

Es una subconsulta, una consulta que se lanza dentro de otra consulta.

Por ejemplo:

```
SELECT nombre, edad FROM Jugadores
WHERE NomEquipo in(SELECT Nombre
FROM Equipos WHERE titulos=2);
```

Esto nos da el nombre y edad de los jugadores cuyos equipos tengan dos titulos. Primero se resuelve la subconsulta o consulta anidada dando los nombres de los aquipos que tienen dos titulos y luego este dato se compara con la tabla jugadores resolviendo la otra consulta.

Desventajas

Son lentas al tener que resolver una por una cada consulta, por eso se recomienda usar consultas multitabla que hacen la consulta directamente en mas de una tabla a la vez.

Query's con distintos tipos de JOIN (INNER, LEFT, OUTER)

Los JOINs en SQL sirven para combinar filas de dos o más tablas basándose en un campo común entre ellas, devolviendo por tanto datos de diferentes tablas. Un JOIN se produce cuando dos o más tablas se juntan en una sentencia SQL.

Existen más tipos de joins en SQL que los que aquí se explican, como CROSS JOIN, O SELF JOIN, pero no todos ellos están soportados por todos los sistemas de bases de datos. Los más importantes son los siguientes:

- INNER JOIN: Devuelve todas las filas cuando hay al menos una coincidencia en ambas tablas.
- LEFT JOIN: Devuelve todas las filas de la tabla de la izquierda, y las filas coincidentes de la tabla de la derecha.
- RIGHT JOIN: Devuelve todas las filas de la tabla de la derecha, y las filas coincidentes de la tabla de la izquierda.
- OUTER JOIN: Devuelve todas las filas de las dos tablas, la izquierda y la derecha. También se llama FULL OUTER JOIN.

INNER JOIN selecciona todas las filas de las dos columnas siempre y cuando haya una coincidencia entre las columnas en ambas tablas. Es el tipo de JOIN más común.

```
SELECT nombreColumna(s)
FROM tabla1
INNER JOIN tabla2
ON tabla1.nombreColumna=table2.nombreColumna;
```

LEFT JOIN mantiene todas las filas de la la tabla1. Las filas de la tabla derecha se mostrarán si hay una coincidencia con las de la izquierda. Si existen valores en la tabla izquierda pero no en la tabla derecha, ésta mostrará null.

```
SELECT nombreColumna(s)
FROM tabla1
LEFT JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;
```

RIGHT JOIN es igual que LEFT JOIN pero al revés. Ahora se mantienen todas las filas de la tabla2. Las filas de la tabla izquierda se mostrarán si hay una coincidencia con las de la derecha. Si existen valores en la tabla derecha pero no en la tabla izquierda, ésta se mostrará null.

```
SELECT nombreColumna(s)
FROM tabla1
RIGHT JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;
```

OUTER JOIN o **FULL OUTER**

JOIN devuelve todas las filas de la tabla1 y de la tabla2. Combina el resultado de los joins **LEFT** y **RIGHT.** Aparecerá null en cada una de las tablas alternativamente cuando no haya una coincidencia.

```
SELECT nombreColumna(s)
FROM tabla1
OUTER JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreCo-
lumna;
```

2.1) Sentencias para la manipulación de datos

Data Manipulación Languaje (DML)

Las sentencias de lenguaje de manipulación de datos **(DML)** son utilizadas para gestionar datos dentro de los schemas. Algunos ejemplos:

- **SELECT -** para obtener datos de una base de datos.
- **INSERT -** para insertar datos a una tabla.
- **UPDATE -** para modificar datos existentes dentro de una tabla.
- **DELETE -** elimina todos los registros de la tabla; no borra los espacios asignados a los registros.

Actualizando la información de una tabla

Update es la instrucción del lenguaje SQL que nos sirve para modificar los registros de una tabla. Como para el caso de Delete, necesitamos especificar por medio de Where cuáles son los registros en los que queremos hacer efectivas nuestras modificaciones. Además, obviamente, tendremos que especificar cuáles son los nuevos valores de los campos que deseamos actualizar.

La sintaxis es de este tipo:

UPDATE nombre_tabla
SET nombre_campo1 = valor_campo1, nombre_campo2 = valor_campo2,...
WHERE condiciones_de_selección

Es importante la utilización del **WHERE** de lo contrario la modificación se realizará a todas las filas de la tabla.

Borrando información de una tabla

La instrucción **DELETE** permite eliminar uno o múltiples registros. Incluso todos los registros de una tabla, dejándola vacía.

La sintaxis es de este tipo:

DELETE [FROM] nombre_tabla
WHERE condiciones_de_selección

La condición, como siempre, define las condiciones que deben cumplir los registros que se desean eliminar. Se puede aplicar todo lo visto para esta cláusula anteriormente, incluidas las sub-consultas.

Ingresando información a una tabla

INSERT es una instrucción que permite la inserción de filas en una tabla determinada con valores recibidos o generados.

La sintaxis es de este tipo:

INSERT INTO nombre_tabla (nombre_campo1, nombre_campo2)
VALUES (valor_campo1,valor_campo2);

Insertar, actualizar y borrar datos con integridad referencial

Podemos decir de manera simple que integridad referencial significa que cuando un registro en una tabla haga referencia a un registro en otra tabla, el registro correspondiente debe existir.

Por ejemplo, consideremos la relación entre una tabla cliente y una tabla venta.

Para poder establecer una relación entre dos tablas, es necesario asignar un campo en común a las dos tablas. Para este ejemplo, el campo id_cliente existe tanto en la tabla cliente como en la tabla venta. La mayoría de las veces, este campo en común debe ser una clave primaria en alguna de las tablas. Vamos a insertar algunos datos en estas tablas.

Hay dos registros en la tabla cliente, pero existen 3 id_cliente distintos en la tabla venta. Habíamos dicho que las dos tablas se relacionan con el campo id_cliente, por lo tanto, po-

demos decir que Juan Penas tiene una cantidad de 23, y Pepe el Toro 81, sin embargo, no hay un nombre que se corresponda con el id_cliente 3. Las relaciones de claves foráneas se describen como relaciones padre/hijo (en nuestro ejemplo, cliente es el padre y venta es el hijo), y se dice que un registro es huérfano cuando su padre ya no existe. Cuando en una base de datos se da una situación como esta, se dice que se tiene una integridad referencial pobre (pueden existir otra clase de problemas de integridad). Generalmente esto va ligado a un mal diseño, y puede generar otro tipo de problemas en la base de datos, por lo tanto debemos evitar esta situación siempre que sea posible.

Restricciones en una tabla

Las CONSTRAINTS son restricciones que se utilizan para limitar el tipo de dato que puede recibir una columna de una tabla.
Las restricciones se puede definir cuando creamos la tabla (CREATE TABLE) o posteriormente con la sentencia ALTER TABLE.
Las posibles restricciones son:

- **NOT NULL:** Sirve para especificar que una columna no acepta el valor NULL, es decir, que esa columna siempre tiene que tener algún valor, no puede estar vacía.
- **UNIQUE:**Identifica de manera única a cada fila de una tabla.
- **PRIMARY KEY:** Identifica de manera única cada fila de una tabla.
- **FOREIGN KEY:** Es una columna o varias columnas, que sirven para señalar cual es la clave primaria de otra tabla.
- **CHECK:** Se utiliza para limitar el rango de valores que puede tener una columna.
- **DEFAULT:** Se utiliza para establecer un valor por defecto a una columna.

2.2) Transaccionalidad en las operaciones

Qué es una transacción

Las transacciones en SQL son unidades o secuencias de trabajo realizadas de forma ordenada y separada en una base de datos. Normalmente representan cualquier cambio en la base de datos, y tienen dos objetivos principales:

· Proporcionar secuencias de trabajo fiables que permitan poder recuperarse fácilmente ante errores y mantener una base de datos consistente incluso frente a fallos del sistema.
· Proporcionar aislamiento entre programas accediendo a la vez a la base de datos.

Una transacción es una propagación de uno o más cambios en la base de datos, ya sea cuando se crea, se modifica o se elimina un registro. En la práctica suele consistir en la agrupación de consultas SQL y su ejecución como parte de una transacción.

Propiedades de las transacciones

Las transacciones siguen cuatro propiedades básicas, bajo el acrónimo ACID (Atomicity, Consistency, Isolation, Durability):

· Atomicidad: aseguran que todas las operaciones dentro de la secuencia de trabajo se completen satisfactoriamente. Si no es así, la transacción se abandona en el punto del error y las operaciones previas retroceden a su estado inicial.
· Consistencia: aseguran que la base de datos cambie estados en una transacción exitosa.
· Aislamiento: permiten que las operaciones sean aisladas y transparentes unas de otras.
· Durabilidad: aseguran que el resultado o efecto de una transacción completada permanezca en caso de error del sistema.

Confirmación de una transacción

La sentencia COMMIT confirma los cambios realizados en la base de datos durante la transacción actual y hace que los cambios sean permanentes.

Vuelta atrás de una transacción

Puede usar ROLLBACK TRANSACTION para borrar todas las modificaciones de datos realizadas desde el inicio de la transacción o hasta un punto de retorno. También libera los recursos que mantiene la transacción.

La cancelación de una transacción es restituir los cambios a un punto de respaldo. Un punto de respaldo es una entidad con nombre que representa el estado de los datos en un determinado momento durante una transacción.

Puede utilizar la sentencia ROLLBACK para restituir los cambios únicamente a un punto de respaldo dentro de la transacción sin finalizar la transacción.

3.1) Sentencias para la definición de tablas

El lenguaje de definición de datos DDL

Un lenguaje de definición de datos (Data Definition Language, DDL por sus siglas en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.

La definición de la estructura de la base de datos incluye tanto la creación inicial de los diferentes objetos que formarán la base de datos, como el mantenimiento de esa estructura. Las sentencias del DDL utilizan unos verbos que se repiten para los distintos objetos.

Las principales funcionalidades de SQL como lenguaje de definición (DDL) son la creación, modificación y borrado de las tablas que componen la base de datos, así como de los índices, vistas, sinónimos, permisos, etc. que pudieran definirse sobre las mismas.

Creación de una tabla

CREATE TABLE crea una tabla con el nombre dado. Se debe tener el permiso CREATE para la tabla.
Por defecto, la tabla se crea en la base de datos actual. Ocurre un error si la tabla existe, si no hay base de datos actual o si la base de datos no existe.

La sintaxis es de este tipo:

CREATE TABLE “nombre_tabla”
(“columna 1” “tipo_de_datos_para_columna_1”,
“columna 2” “tipo_de_datos_para_columna_2”,
...);

Ejemplo:

CREATE TABLE Cliente
(PrimerNombre char(50),
Apellido char(50),
Dirección char(50),
Ciudad char(50),
Pais char(25),
FechaNacimiento datetime);

La restricción de nulidad

Por defecto, las columnas pueden contener valores NULL. Se usa una restricción **NOT NULL** en SQL para evitar insertar valores NULL en la columna especificada, considerándolo entonces como un valor no aceptado para esa columna. Esto significa que debe proporcionar un valor válido SQL **NOT NULL** a esa columna en las instrucciones INSERT o UPDATE, ya que la columna siempre contendrá datos.

Ejemplo:

CREATE TABLE Cliente
(PrimerNombre varchar(50) **NOT NULL,**
Apellido varchar(50),
Dirección varchar(50),
Ciudad varchar(50),
Pais varchar(25),
FechaNacimiento datetime);

Definición de la llave primaria

La restricción **PRIMARY KEY** consta de una columna o varias columnas con valores que identifican de forma única cada fila de la tabla.

La restricción PRIMARY KEY de SQL se combina entre las restricciones UNIQUE y SQL NOT NULL, donde la columna o el conjunto de columnas que participan en PRIMARY KEY no pueden aceptar el valor NULL. Si la CLAVE PRIMARIA se define en varias columnas, entonces se puede insertar valores duplicados en cada columna individualmente, pero es importante mencionar que los valores de combinación de todas las columnas de CLAVE PRIMARIA deben ser únicos.

Ejemplo:

```
CREATE TABLE Cliente
(idCliente int PRIMARY KEY,
PrimerNombre varchar(50) NOT NULL,
Apellido varchar(50),
Dirección varchar(50),
FechaNacimiento datetime);
```

Definición de llaves foráneas

Una **clave foránea** es una columna o grupo de columnas de una tabla que contiene valores que coinciden con la **clave** primaria de otra tabla. Las claves foráneas se utilizan para unir tablas.

Controla que el valor de un atributo (o la combinación de atributos) exista en otra tabla (el valor). Este atributo (o la combinación de atributos) debe ser clave primaria en la otra tabla (CLAVE EXTERNA).

```
CREATE TABLE Cliente (
        idCliente int PRIMARY KEY,
        PrimerNombre varchar(50) NOT NULL,
        GrupoClientes int NOT NULL,
        CONSTRAINT fk_idGrupo FOREIGN
KEY (idGrupo)
);
```

4.1) El modelo Entidad-Relación

El proceso de abstracción

La abstracción de datos es una técnica o metodología que permite diseñar estructuras de datos. La abstracción consiste en representar bajo ciertos lineamientos de formato las características esenciales de una estructura de datos. Este proceso de diseño evita los detalles específicos de implementación de los datos.

El modelo conceptual de Entidad-Relación

Es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Permite diseñar esquemas que posteriormente debemos de implementar en un gestor de BBDD (bases de datos). Este modelo se representa a través de diagramas y está formado por distintos elementos.

Entidades

Tipo de objeto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso (coches, casas, empleados, clientes, empresas, oficios, diseños de productos, conciertos, excursiones, etc.).

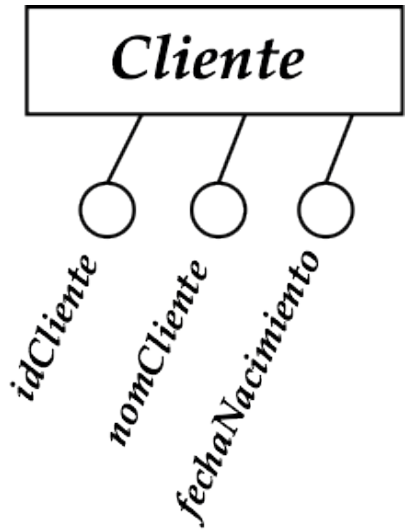
· Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior.
· Un nombre de entidad sólo puede aparecer una vez en el esquema.

Definición de atributos e identificadores únicos

Definición de atributos

Los atributos definen o identifican las características de entidad. Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha...).

Los atributos para la entidad que se muestra serían los siguientes: idCliente, nombreCliente, fechaNacimiento y muchos otros que complementen la información de cada cliente.



Identificadores Únicos

Es el atributo de una entidad, al que le aplicamos una restricción que lo distingue de los demás registros (no permitiendo que el atributo específico se repita en la entidad) o le aplica un vínculo. Estos son los distintos tipos:

Clave primaria: Identifica inequívocamente un solo atributo no permitiendo que se repita en la misma entidad. Como sería la matrícula o el número de chasis de un coche (no puede existir dos veces el mismo).

Clave externa o clave foránea: este campo tiene que estar estrictamente relacionado con la clave primaria de otra entidad, para así exigir que exista previamente ese clave. Anteriormente hemos hablado de ello cuando comentábamos que un empleado indispensablemente tiene que tener un cargo, por lo cual si intentásemos darle un cargo inexistente el gestor de bases de datos nos devolvería un error.

Tipos de relación entre entidades

La relación o interpelación es un elemento del modelo Entidad/Relación que permite relacionar datos entre sí. En una relación se asocia un elemento de una entidad con otro de otra entidad.

Entidades débiles

Son aquellas cuya existencia depende de la existencia de otras instancias de entidad. Por ejemplo, consideremos las entidades EDIFICIO y SALA. Supongamos que puede haber salas identificadas con la misma numeración pero en edificios diferentes. La numeración de cada sala no identificará completamente cada una de ellas. Para poder identificar completamente una sala es necesario saber también en qué edificio está localizada. Por tanto, la existencia de una instancia de una entidad débil depende de la existencia de una instancia de la entidad fuerte con la que se relaciona.

Entidades Fuertes

Son aquellas que tienen existencia por sí mismas, es decir, su existencia no depende de la existencia de otras entidades. Por ejemplo, en una base de datos hospitalaria, la existencia de instancias concretas de la entidad DOCTOR no depende de la existencia de instancias u objetos de la entidad PACIENTE. En el modelo E/R las entidades fuertes se representan como hemos indicado anteriormente, con el nombre de la entidad encerrado dentro de un rectángulo.

ENLACES Y BIBLIOGRAFIA

Unidad 1

"¿Qué es una base de datos relacional?"
https://www.oracle.com/cl/database/what-is-a-relational-database/

Unidad 2

"Sintaxis del lenguaje de manipulación de datos"
https://cloud.google.com/bigquery/docs/reference/standard-sql/dml-syntax?hl=es-419

Unidad 3

"Lenguaje de definición de datos"
https://es.wikipedia.org/wiki/Lenguaje_de_definici%C3%B3n_de_datos

Unidad 4

"Qué es un diagrama entidad-relación"
https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

TALENTO DIGITAL
INTELIGENCIA HUMANA

AIEP

Instituto Profesional AIEP Spa.
Dirección Nacional de Educación Continua
Programa Talento Digital 2021