

Introducción a SQL

La estructuración y gestión de grandes volúmenes de datos es un desafío cada vez más común en el mundo empresarial y tecnológico. Para manejar estos datos de manera efectiva, se utilizan diferentes tipos de sistemas de gestión de bases de datos, y uno de los más populares es **SQL (Structured Query Language)**. SQL es un *lenguaje de consulta utilizado para administrar y manipular grandes bases de datos, permitiendo la creación, modificación y eliminación de datos de manera eficiente.*

En este documento se presentará una introducción a SQL, sus principales características y funcionalidades, así como algunas de las mejores prácticas para su uso efectivo.

Temas principales a tratar en este documento:

- Introducción a SQL: conocer los fundamentos de SQL, su sintaxis y cómo se utiliza para interactuar con las bases de datos.
- Creación de tablas y campos: aprender a crear tablas y campos para almacenar y organizar los datos en una base de datos.
- Consultas **SELECT**: entender cómo se utiliza la sentencia SELECT para recuperar datos específicos de una o varias tablas.
- Modificación de datos: aprender a utilizar las sentencias **UPDATE**, **INSERT** y **DELETE** para modificar y agregar datos a una base de datos.
- Restricciones y claves: comprender las restricciones y claves, como las restricciones de integridad referencial y las claves primarias y externas.
- Funciones y operadores: conocer las funciones y operadores comunes de SQL, como las funciones de agregación y los operadores lógicos.
- Uniones y subconsultas: entender cómo se utilizan las uniones y subconsultas para combinar y filtrar datos de varias tablas.
- Transacciones y bloqueos: aprender a trabajar con transacciones y bloqueos para asegurar la integridad de los datos en la base de datos.
- Optimización de consultas: conocer las técnicas para optimizar consultas SQL para mejorar el rendimiento de la base de datos.

Introducción

SQL (Structured Query Language) es un lenguaje de programación utilizado para interactuar con bases de datos relacionales. Los fundamentos de SQL incluyen las siguientes características:

Declarativo: SQL es un lenguaje declarativo, lo que significa que le dice al motor de la base de datos lo que quiere hacer, no cómo hacerlo. El motor de la base de datos se encarga de procesar la consulta y encontrar la mejor manera de ejecutarla.

Independencia del sistema: SQL es independiente del sistema, lo que significa que funciona en cualquier sistema de base de datos relacional, independientemente del fabricante o la plataforma.

Estructurado: SQL es un lenguaje estructurado, lo que significa que utiliza una sintaxis específica para estructurar las consultas y comandos.

Basado en conjuntos: SQL es un lenguaje basado en conjuntos, lo que significa que trabaja con conjuntos de datos, en lugar de operar con un solo registro a la vez.

La sintaxis de SQL consta de comandos y cláusulas que se utilizan para interactuar con la base de datos. Algunos de los comandos SQL más comunes incluyen:

SELECT: se utiliza para recuperar datos de una o varias tablas.

INSERT: se utiliza para agregar nuevos datos a una tabla.

UPDATE: se utiliza para modificar datos existentes en una tabla.

DELETE: se utiliza para eliminar datos de una tabla.

CREATE: se utiliza para crear una nueva tabla, vista, índice o procedimiento almacenado.

DROP: se utiliza para eliminar una tabla, vista, índice o procedimiento almacenado.

Para utilizar SQL, se utiliza un cliente de base de datos, como MySQL Workbench o Microsoft SQL Server Management Studio, para conectarse a una base de datos y escribir consultas SQL. La consulta se envía al motor de la base de datos, que la procesa y devuelve los resultados.

Existen distintos tipos de declaraciones en SQL que son categorizadas en seis tipos:

1. DDL- Data Definition Language.

Estas declaraciones son usadas para construir objetos en la base de datos. Específicamente son usados para:

- Crear, alterar y borrar tablas y otros objetos de la base de datos
- Agregar comentarios en un objeto en particular que se encuentre alojado en la base de datos y asociarlo con otros objetos.
- Otorgar privilegios a otros usuarios dentro de la base de datos.
- Analizar objetos usando herramientas.

2. DML- Data Manipulation Language.

Se refiere a las declaraciones que son usadas para trabajar con los datos en los objetos. Estas declaraciones se usan para añadir, modificar, ver y eliminar datos de los objetos en la base de datos, como las tablas.

3. TCL – Transaction Control Language.

Esta declaraciones complementan a las DML sin embargo no forman parte de ellas. Se utilizan para controlar los datos dentro de nuestra base de datos, guardan y recuperan datos.

4. Sesión Control Statements
5. System Control Statatements
6. Embedded SQL Statements

Schemas

Los esquemas(schemas) son una colección de ciertos objetos de una base de datos, como las tablas, índices y vistas, todos estos pertenecen a la cuenta de usuario. En si es una colección lógica de los objetos en una base de datos.

A un usuario le “pertenecen” uno o mas objetos en una base de datos y juntos esos objetos constituyen un esquema (schema). Un schema tiene es mismo nombre que la cuenta del usuario y el usuario se identifica por un conjunto de privilegios y roles.

Los objetos dentro de una base de datos se categorizan en 2 tipos: **schema** y **nonschema**

Schema Objects	Nonschema Objects
Tables	Users
Constraints	Roles
Indexes	Public synonyms
Views	
Sequences	
Private synonyms	

Consideraciones de la creación de objetos

- Nombramiento de los objetos: Los nombres de los objetos en nuestra base de datos deben de ser únicos y no se pueden llamar igual que alguna palabra reservada existente.

- Los nombres de nuestros objetos pueden ir entrecomillados (usando “ ”), el hecho de que estén entrecomillados los hace sensibles, quiere decir que si se llamo a un objeto “Nombre” después no se puede llamar como “NOMBRE” o “nombre”.
- De la misma manera hay que tomar en cuenta no deben de existir objetos con los mismos nombres aunque en algunos casos esta permitido, por orden y evitar confusión, cada objeto debe d tener un nombre único.

Para saber si nuestra tabla fue creada exitosamente se usa la declaración **DESCRIBE** (también abreviada **DESC**) que muestra los detalles y estructura de nuestra tabla. Por ejemplo para la tabla ALUMNO anterior:

DESCRIBE ALUMNO	
Nombre	Nulo Tipo

NOMBRE	VARCHAR2(10)
APELLIDO	VARCHAR2(10)
NUMERO_CUENTA	NUMBER
FECHA_INGRESO	DATE

Tipos de datos

Tipos de datos.

Como se vio anteriormente los tipos de datos se asignan a las columnas al momento de crear una nueva tabla, esto para especificar que los datos incorporados a esta columna cumplan con ese tipo de dato. En caso de que el dato no cumpla con el tipo de dato entonces marcara error. Existen gran cantidad de tipos de datos.

- **CHAR(n)**: “Carácter”, acepta cualquier carácter alfanumérico. La ‘n’ indica la longitud deseada.
- **VARCHAR2(n)**: “carácter variable”, acepta cualquier variable alfanumérica. La ‘n’ indica la longitud máxima y es necesario indicar la longitud. El valor mínimo para ‘n’ es 1 y el valor máximo es 4,000 bytes.
- **NUMBER(m,n)**: Este tipo de dato acepta datos numéricos incluidos el cero, negativos y números positivos. La ‘n’ representa la precisión que es el numero máximo de números o dígitos significantes (o bien los dígitos antes del punto decimal), y la ‘m’ es la escala que quiere decir el total de dígitos a la después del puto decimal, si ‘m’ se omite entonces se dará por default que es cero.
- **DATE**: Este tipo de dato consiste en campos, cada campo es un componente de fecha, tiempo, horas, minutos, meses. Y todos estos campos generan varias combinaciones para este tipo de dato.

En general acepta información de fecha y tiempo. Los campos que incluye son año, mes, fecha, hora, minutos y segundos.

Large Objects.

Los Large Objects o LOB pueden ser usados como los otros tipos de datos. Una tabla puede tener múltiples columnas con datos tipo LOB, las columnas con tipo de dato LOB no pueden ser PRIMARY KEY y tampoco se pueden usar en **DISTINC, GROUP BY, ORDER BY y JOIN.**

Dentro de los tipos de dato LOB entran:

- **BLOB:** "Binary Large Object". Acepta objetos binarios, imágenes o archivos de video.
- **CLOB:** "Carácter Large Object". Acepta elementos de texto largos.
- **NCLOB.**

Creación de tablas y campos

1. Para crear tablas y campos en Oracle SQL, sigue estos pasos:
2. Abre Oracle SQL Developer y conéctate a la base de datos donde deseas crear la tabla.
3. Haz clic derecho en el nombre de la base de datos y selecciona "Nueva consulta".
4. En la ventana de la consulta, escribe el siguiente código SQL para crear una nueva tabla:

```
CREATE TABLE nombre_tabla (  
    campo1 tipo_dato1 [restricciones],  
    campo2 tipo_dato2 [restricciones],  
    campo3 tipo_dato3 [restricciones],  
    ...  
);
```

5. Reemplaza "nombre_tabla" con el nombre que deseas darle a la tabla.
6. Define cada campo que deseas incluir en la tabla, especificando su nombre, tipo de datos y cualquier restricción que desees aplicar.
7. Separa cada campo con una coma.
8. Ejecuta la consulta haciendo clic en el botón "Ejecutar" o presionando la tecla F5.

9. Si la consulta se ejecuta con éxito, verás un mensaje que indica que la tabla se ha creado.

Una consideración importante al crear nuevos objetos es la asignación de los nombres, se sugiere que se asigne de manera única, simple, sin espacios o caracteres especiales, ya que será más fácil su uso para futuras declaraciones.

Aquí hay un ejemplo de código SQL para crear una tabla de empleados:

```
CREATE TABLE empleados (  
  id_empleado NUMBER(10) PRIMARY KEY,  
  nombre VARCHAR2(50) NOT NULL,  
  apellido VARCHAR2(50) NOT NULL,  
  fecha_nacimiento DATE,  
  salario NUMBER(10,2) DEFAULT 0,  
  departamento VARCHAR2(50) REFERENCES departamentos(nombre)  
);
```

En este ejemplo, la tabla "empleados" tiene seis campos:

id_empleado: campo numérico que es la clave principal de la tabla.

nombre: campo de texto que almacena el nombre del empleado.

apellido: campo de texto que almacena el apellido del empleado.

fecha_nacimiento: campo de fecha que almacena la fecha de nacimiento del empleado.

salario: campo numérico que almacena el salario del empleado y tiene un valor predeterminado de 0.

departamento: campo de texto que almacena el nombre del departamento al que pertenece el empleado y se referencia a una tabla departamentos mediante una clave foránea.

Consultas SELECT

Para recuperar datos específicos de una o varias tablas en SQL Oracle, se utiliza la sentencia **SELECT**. Sigue estos pasos para escribir una consulta SELECT:

1. Abre Oracle SQL Developer y conéctate a la base de datos que deseas consultar.
2. Haz clic derecho en el nombre de la base de datos y selecciona "Nueva consulta".
3. En la ventana de la consulta, escribe el siguiente código SQL:

```
select COLUMN1, COLUMN2, ... from TABLA1 [join TABLA2 on CONDICION] where CONDICION;
```

4. Reemplaza "columna1, columna2, ..." con el nombre de las columnas que deseas recuperar de la tabla.
5. Reemplaza "tabla1" con el nombre de la tabla que deseas consultar.

Si deseas combinar dos o más tablas, puedes usar la cláusula JOIN seguida del nombre de la tabla y la condición de unión. La condición de unión especifica cómo se relacionan las tablas. Por ejemplo, si tienes una tabla de empleados y una tabla de departamentos, puedes combinarlas con la siguiente cláusula JOIN:

```
SELECT empleados.nombre, departamentos.nombre  
FROM empleados  
JOIN departamentos  
on EMPLEADOS.DEPARTAMENTO_ID = DEPARTAMENTOS.id;
```

Si deseas aplicar una condición para filtrar los resultados, utiliza la cláusula WHERE seguida de la condición que deseas aplicar. Por ejemplo, si deseas recuperar todos los empleados que ganan más de \$50,000 al año, puedes usar la siguiente consulta:

```
SELECT nombre, salario  
FROM empleados  
where SALARIO > 50000;
```

Modificación de datos

En SQL Oracle, existen tres sentencias principales que se utilizan para modificar y agregar datos a una base de datos: UPDATE, INSERT y DELETE. A continuación, se explican brevemente estas tres sentencias:

1. Sentencia **UPDATE**:

La sentencia **UPDATE** se utiliza para actualizar datos en una o varias filas de una tabla. Sigue estos pasos para utilizar la sentencia **UPDATE**:

- Especifica la tabla que deseas actualizar y las columnas que deseas modificar.
- Utiliza la cláusula **WHERE** para identificar las filas que deseas actualizar.
- Especifica los nuevos valores que deseas asignar a las columnas.

Aquí hay un ejemplo de cómo usar la sentencia **UPDATE** para actualizar el salario de un empleado:

```
UPDATE empleados  
SET salario = 60000  
where ID_EMPLEADO = 123;
```

2. Sentencia **INSERT**:

La sentencia INSERT se utiliza para agregar nuevas filas a una tabla. Sigue estos pasos para utilizar la sentencia INSERT:

- Especifica la tabla a la que deseas agregar datos y las columnas correspondientes.

- Especifica los valores que deseas agregar a cada columna.

Aquí hay un ejemplo de cómo usar la sentencia INSERT para agregar un nuevo empleado:

```
INSERT INTO empleados (id_empleado, nombre, apellido, fecha_nacimiento, salario, departamento)
values (124, 'Juan', 'Pérez', '01-01-1980', 50000, 'Marketing');
```

3. Sentencia **DELETE**:

La sentencia DELETE se utiliza para eliminar filas de una tabla. Sigue estos pasos para utilizar la sentencia DELETE:

- Especifica la tabla de la que deseas eliminar filas.
- Utiliza la cláusula WHERE para identificar las filas que deseas eliminar.

Aquí hay un ejemplo de cómo usar la sentencia DELETE para eliminar un empleado:

```
DELETE FROM empleados
where ID_EMPLEADO = 124;
```

Restricciones y claves

En SQL Oracle, las restricciones y claves son elementos importantes que se utilizan para garantizar la integridad de los datos en las tablas de la base de datos. A continuación se describen brevemente algunas de las restricciones y claves más comunes en SQL Oracle:

Restricciones de integridad:

Las restricciones de integridad son reglas que se aplican a los datos en una tabla para garantizar que sean precisos y coherentes. Algunas de las restricciones de integridad más comunes en SQL Oracle incluyen:

NOT NULL: especifica que una columna no puede contener valores nulos.

UNIQUE: especifica que los valores en una columna deben ser únicos.

CHECK: Define una regla en particular para futuras filas que se agrega en, solo se modifica una pequeña parte del código. Esta restricción en particular **no se puede incluir**:

- Columnas en otras tablas
- Pseudocolumnas **CURRVAL, NEXTVAL, LEVEL, ROWNUM**
- Subconsultas o expresiones de consultas escalares
- Funciones definidas de usuario
- Ciertas funciones de tiempo.

FOREIGN KEY: especifica una relación entre dos tablas en la que la columna de una tabla se relaciona con la clave principal de otra tabla.

Claves:

Las claves se utilizan para identificar de manera única cada fila en una tabla. Algunas de las claves más comunes en SQL Oracle incluyen:

Clave principal(Primary Key): una columna o combinación de columnas que se utiliza para identificar de manera única cada fila en una tabla.

Clave foránea(Foreign Key): una columna o combinación de columnas que se utiliza para establecer una relación entre dos tablas.

Es importante tener en cuenta que el uso de restricciones y claves puede afectar el rendimiento de las consultas en la base de datos, especialmente en tablas grandes. Es importante equilibrar la necesidad de integridad de los datos con la necesidad de un rendimiento óptimo de las consultas.

Creación de CONSTRAINTS en creación de tablas.

Se pueden crear CONSTRAINTS(restricción) NULL, NOT NULL, PRIMARY KEY, FOREIGN KEY, considerando que para la FOREIGN KEY debe existir una PRIMARY KEY creada desde que se crea una tabla, esta debe de ir en seguida de que sea crea una columna.

```
CREATE TABLE ALUMNO
(
  NOMBRE          VARCHAR(10) CONSTRAINT NOMBRE_PK PRIMARY KEY,
  APELLIDO        VARCHAR(10),
  NUMERO_CUENTA   NUMBER ,
  FECHA_INGRESO   DATE
);
```

**La restricción NOT NULL no puede ser declarada fuera de línea.*

Funciones y operadores

SQL Oracle cuenta con una gran cantidad de funciones y operadores que se pueden utilizar para manipular y consultar datos en una base de datos. A continuación se describen brevemente algunas de las funciones y operadores más comunes en SQL Oracle:

Funciones de agregación:

Las funciones de agregación se utilizan para realizar cálculos en grupos de filas. Algunas de las funciones de agregación más comunes en SQL Oracle incluyen:

SUM: devuelve la suma de los valores en una columna.

AVG: devuelve el promedio de los valores en una columna.

COUNT: devuelve el número de filas en una tabla o el número de valores no nulos en una columna.

MAX: devuelve el valor máximo en una columna.

MIN: devuelve el valor mínimo en una columna.

Funciones de fecha y hora:

Las funciones de fecha y hora se utilizan para manipular y consultar datos de fecha y hora. Algunas de las funciones de fecha y hora más comunes en SQL Oracle incluyen:

SYSDATE: devuelve la fecha y hora actuales del sistema.

MONTHS_BETWEEN: devuelve el número de meses entre dos fechas.

NEXT_DAY: Retorna una fecha correspondiente al primer día especificado en formato día(lunes, martes, miércoles,...).

```
select next_day('01/01/2023','LUNES') from dual;
```

Se muestran la siguiente fecha que será 'LUNES'.

ADD_MONTHS: agrega un número específico de meses a una fecha.

```
select add_months('12/12/1990', 9) from dual;
```

 devuelve

```
12/09/1991
```

TO_CHAR: convierte una fecha en una cadena de caracteres en un formato específico.

LAST_DAY: Retorna el ultimo día del mes de la fecha dada como argumento.

EXTRACT: Retorna la parte de una fecha. Se puede retornar, día, mes, año, hora, minutos, segundos, etc.

SYSDATE: Fecha y hora actual del servidor de Oracle.

**Los operadores aritméticos (+,-) funcionan con fechas.*

Operadores de comparación:

Los operadores de comparación se utilizan para comparar valores en una consulta. Algunos de los operadores de comparación más comunes en SQL Oracle incluyen:

= (igual): compara si dos valores son iguales.

!= o <> (distinto): compara si dos valores son diferentes.

< (menor que): compara si un valor es menor que otro valor.

>(mayor que): compara si un valor es mayor que otro valor.

BETWEEN: compara si un valor se encuentra entre dos valores especificados.

Operadores lógicos:

Los operadores lógicos se utilizan para combinar múltiples condiciones en una consulta. Algunos de los operadores lógicos más comunes en SQL Oracle incluyen:

AND: combina múltiples condiciones y devuelve verdadero si todas las condiciones son verdaderas.

OR: combina múltiples condiciones y devuelve verdadero si al menos una de las condiciones es verdadera.

NOT: invierte una condición y devuelve verdadero si la condición es falsa.

conjuntos:

Operadores aritméticos: Son los que utilizan valores numéricos y que retornan un valor numérico.

Símbolo	Significado	Ejemplo
+	Operación suma	1 + 2
-	Operación resta	1 - 2
*	Operación multiplicación	1 * 2
/	Operador división	1 / 2

Es importante tener en cuenta que SQL Oracle cuenta con muchas más funciones y operadores que los mencionados anteriormente. La elección de la función o operador correcto dependerá de los requisitos específicos de la consulta.

Como complemento, existen los siguientes:

		[TRUE]
ANY SOME	a algún elemento del arreglo de resultados (derecha). Debe ser estar precedido por =, !=, <, <=, >, >= Hace un OR lógico entre todos los elementos.	10 >= ANY (1,2,3,10) [TRUE]
ALL	a todos los elementos del arreglo de resultados (derecha), Debe ser estar precedido por =, !=, <, <=, >, >= Hace un AND lógico entre todos los elementos.	10 <= ALL (1,2,3,10) [TRUE]
BETWEEN x AND y	Operando de la izquierda entre x e y. Equivalente a op >= x AND op <= y	10 BETWEEN 1 AND 100
EXISTS	Si la retorna al menos una fila	EXISTS(SELECT 1 FROM DUAL)
LIKE(*)	Es como	'pepe' LIKE 'pe%'
IS NULL	Si es nulo	1 IS NULL
IS NOT NULL	Si es No nulo	1 IS NOT NULL
NOT cond.	Niega la condición posteriores	NOT EXISTS... NOT BETWEEN NOT IN NOT =
cond AND cond	Hace un Y lógico entre dos condiciones	1=1 AND 2 IS NULL
Cond OR cond	Hace un O lógico entre dos condiciones	1=1 OR 2 IS NULL

% : Devuelve un conjunto de N caracteres (de 0 a ∞)

| : Devuelve un solo carácter

De concatenación: Para unir cadenas de caracteres

|| : Concatena una cadena con otra o también se puede usar **CONCAT** para solo 2 cadenas de caracteres.

Funciones matemáticas:

- **ABS(X)**: Devuelve el valor absoluto del argumento 'X'.
- **CEIL(X)**: Redondea a entero, hacia arriba(valor mas alto tomando como referencia los decimales).
- **FLOOR(X)**: Redondea a entero, hacia abajo(valor pequeño, tomando como referencia los decimales.)
- **MOD(X,Y)**: Devuelve la división de X/Y.

Borrar y agregar columnas a una tabla

Borrar columnas es muy sencillo y nos sirve para liberar espacio en la memoria y tener un mejor aprovechamiento de las tablas. Cualquier restricción o índice puede ser borrada cuando se corra una columna. Para borrar una columna se utiliza la declaración **DROP**.

A continuación se muestran dos posibles maneras para borrar la columna, en una se utiliza **DROP COLUMN** que especifica que se borrara una COLUMNA y en la otra un **DROP()**, con los parentesis se especifica qué queremos borrar.

```
ALTER TABLE nombre_tabla DROP COLUMN nombre_columna;  
ALTER TABLE nombre_tabla DROP (nombre_columna);
```

- **UNUSED:** De igual manera que la declaración **DROP** su uso es muy sencillo. Una vez que se elije la columna a eliminar, esta se borra. **Una columna que fue borrada con la declaración UNUSED nunca podrá ser recuperada, inclusive si se usa la declaración ROLLBACK.** Para declarar UNUSED se puede hacer de las siguientes 2 maneras:

```
ALTER TABLE nombre_tabla SET UNUSED COLUMN nombre_columna;  
ALTER TABLE nombre_tabla SET UNUSED (nombre_columna);
```

Creación de tablas externas.

Una tabla externa es una tabla de solo lectura que se define en la base de datos pero existe fuera de la base de datos.

En términos más técnicos, una tabla externa se almacena dentro de la base de datos y los datos que contiene están fuera de la base de datos.

Las tablas externas tienen un número de restricciones dentro de ellas, se pueden consultar con **SELECT** pero no se pueden usar ninguna otra declaración DML. Se pueden crear **INDEX** en ella pero no aceptan restricciones(**CONSTRAINTS**).

- No se pueden crear columnas con el tipo de dato LOB, CLOB, BLOB, NCLOB, etc.
- No se pueden agregar restricciones.
- No se puede usar **UNUSED, DIRECTORY**.

Pasos para construir una Tabla Externa.

1. Crear un directorio
2. Después se crea una tabla con referencia al directorio previamente creado

Las tablas externas pueden ser consultadas como cualquier otra tabla en la base de datos.

TRUNCATE

Si por alguna circunstancia se requieren eliminar todas las filas de una tabla, la declaración TRUNCATE es la solución. Por el contrario si requiere eliminar fila por fila o no requiere de eliminar todos los datos entonces funciona la función **DELETE**.

Para utilizar la declaración **TRUNCATE** en una tabla se usa **TRUNCATE TABLE**

```
TRUNCATE TABLE nombre_tabla;
```

y esta declaración sigue los siguientes parámetros:

- Remueve todas la filas de una tabla
- Remueve todos los datos asociados a un índice.
- No funciona con DML pero considera las declaraciones DDL ya que es DDL
- Declara un **COMMIT** de manera implícita
- No funciona con **FLASHBACK_TABLE**
- Mantiene las dependencias y estructuras intactas.

DELETE

Esta declaración remueve todas las filas de una tabla y permite restablecerlas con **ROLLBACK**.

Uniones y subconsultas

En SQL Oracle, las uniones y subconsultas son herramientas útiles para combinar y filtrar datos de varias tablas. A continuación se describen brevemente cada una de estas técnicas:

Uniones:

Las uniones se utilizan para combinar filas de dos o más tablas basadas en una condición de unión. Algunos tipos comunes de uniones en SQL Oracle incluyen:

INNER JOIN: combina filas de dos o más tablas en función de una condición de unión y devuelve solo las filas que cumplen con la condición.

LEFT JOIN: combina todas las filas de la tabla de la izquierda con las filas correspondientes de la tabla de la derecha en función de una condición de unión, y devuelve también las filas de la tabla de la izquierda que no tienen correspondencia en la tabla de la derecha.

RIGHT JOIN: similar al left join, pero devuelve también las filas de la tabla de la derecha que no tienen correspondencia en la tabla de la izquierda.

FULL OUTER JOIN: combina todas las filas de ambas tablas, devolviendo nulos para las filas que no tienen correspondencia en la otra tabla.

Subconsultas:

Las subconsultas se utilizan para realizar consultas dentro de otra consulta. Las subconsultas se pueden utilizar para filtrar datos en una consulta principal, realizar cálculos complejos o combinar datos de múltiples tablas. Algunos tipos comunes de subconsultas en SQL Oracle incluyen:

Subconsultas correlacionadas: utilizan valores de una consulta externa en una subconsulta.

Subconsultas escalares: devuelven un solo valor y se utilizan como parte de una expresión.

Subconsultas de tabla derivada: crean una tabla temporal que se utiliza en la consulta principal.

Es importante tener en cuenta que tanto las uniones como las subconsultas pueden ser costosas en términos de rendimiento si se utilizan en tablas grandes o complejas. Por lo tanto, es importante utilizar estas técnicas de manera cuidadosa y eficiente.

Optimización de consultas

En SQL Oracle, hay varias técnicas que se pueden utilizar para optimizar las consultas SQL y mejorar el rendimiento de la base de datos. Algunas de estas técnicas incluyen:

- **Utilizar índices:** Los índices(**INDEX**) se utilizan para mejorar la velocidad de las consultas SQL al permitir un acceso más rápido a los datos. Es importante crear índices en las columnas que se utilizan con frecuencia en las consultas y en las columnas que se utilizan como claves foráneas en las tablas relacionales.

Para crear un índice en SQL Oracle, se utiliza el comando **CREATE INDEX**, que tiene la siguiente sintaxis básica:
Donde:

```
CREATE INDEX nombre_indice  
on NOMBRE_TABLA (COLUMNA1 [asc|desc], COLUMNA2 [asc|desc], ...)
```

nombre_indice: es el nombre que se le asignará al índice.

nombre_tabla: es el nombre de la tabla sobre la que se creará el índice.

columna1, columna2, ...: son las columnas de la tabla sobre las que se creará el índice. Pueden ser una o varias columnas.

Además, se puede especificar la dirección de ordenamiento(**ORDER BY**) para cada columna, usando las palabras clave **ASC** (ascendente) o **DESC** (descendente).

Por ejemplo, para crear un índice llamado "idx_nombre" en la tabla "usuarios" sobre la columna "nombre" en orden ascendente, se puede utilizar el siguiente comando:

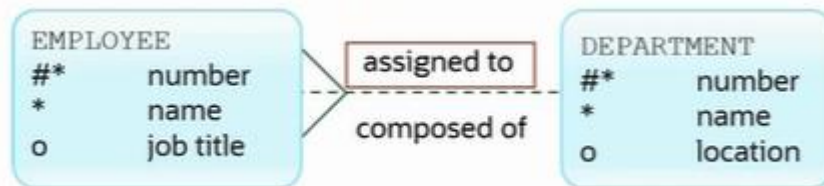
```
CREATE INDEX idx_nombre  
on USUARIOS (NOMBRE asc);
```

Es importante tener en cuenta que la creación de un índice puede tardar un tiempo significativo en tablas grandes, y que un índice mal diseñado puede tener un impacto negativo en el rendimiento de las consultas. Por lo tanto, es importante evaluar cuidadosamente la necesidad de crear un índice y su diseño antes de llevarlo a cabo.

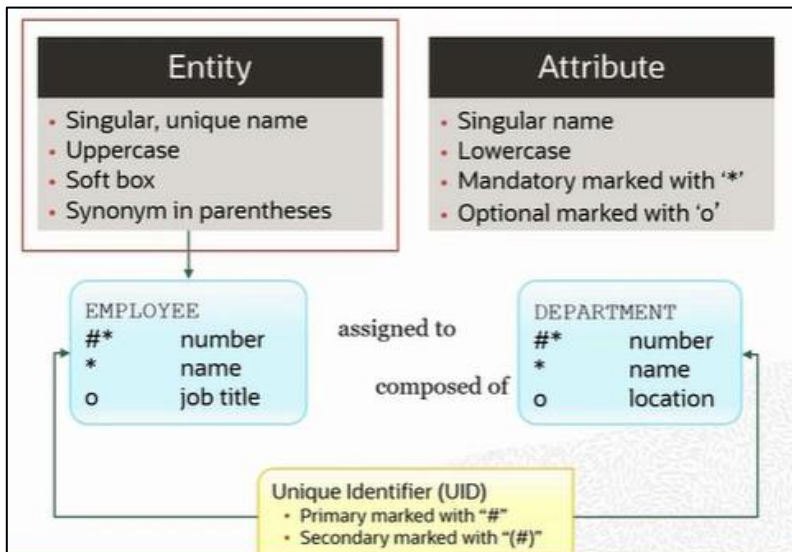
- Utilizar consultas eficientes: Es importante escribir consultas SQL eficientes y bien estructuradas. Se pueden utilizar técnicas como la agrupación, el filtrado y la ordenación para reducir la cantidad de datos que se procesan y mejorar el rendimiento de la consulta.
- Limitar el número de filas devueltas: Si una consulta devuelve un gran número de filas, puede ralentizar el rendimiento de la base de datos. Es importante limitar el número de filas devueltas utilizando técnicas como la cláusula WHERE y la cláusula TOP.
- Utilizar vistas materializadas: Las vistas materializadas son vistas que se almacenan en caché en la base de datos. Estas vistas pueden mejorar el rendimiento de las consultas SQL al permitir un acceso más rápido a los datos.
- Optimizar la configuración de la base de datos: La configuración de la base de datos también puede afectar el rendimiento de las consultas SQL. Es importante configurar correctamente la memoria y el almacenamiento de la base de datos, así como ajustar la configuración de los parámetros de rendimiento.
- Realizar pruebas y monitoreo: Es importante realizar pruebas de rendimiento y monitorear el rendimiento de las consultas SQL para identificar cuellos de botella y áreas de mejora. Se pueden utilizar herramientas como Oracle Enterprise Manager para realizar pruebas y monitorear el rendimiento de la base de datos.

Oracle Database

Para empezar a usar SQL Developer primero es sustancial conocer la estructura relacional que existe detrás de el manejo de una base de datos. Para esto se usara es esquema Human Resources (HR). Dentro de la base de datos existen relaciones entre distintos datos.

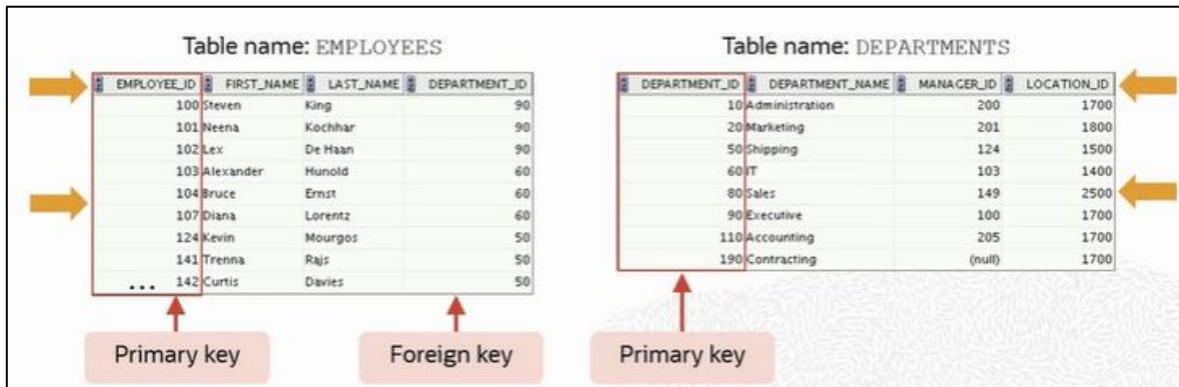


A un empleado le corresponde un departamento y a su vez un departamento esta compuesto de varios empleados.



En este caso cada una de las filas de la tabla employees esta relacionada con una columna de la tabla departamentos. Esta relación se lleva a cabo mediante

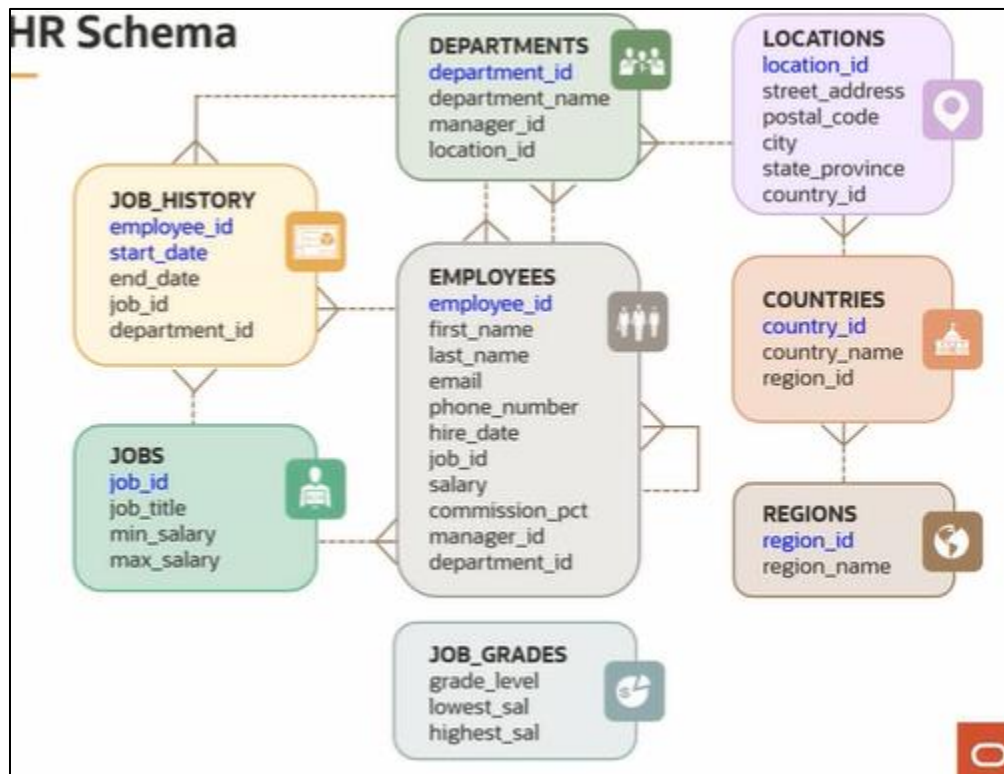
PRIMARY KEY, FOREIGN KEY, en donde se relaciona a cada uno de los valores de una tabla con los valores de otra tabla.



Por lo tanto lo que sucede y la manera en la que logran relacionarse es muy simple. Lo que nos indica la PRIMARY KEY es que los valores de una columna pueden ser relacionados con los valores de otra tabla (deben de ser los mismos para que exista una coincidencia) y lo que hace la FOREIGN KEY es que permite que esta relación se lleve a cabo. En este caso EMPLOYEE_ID es la PRIMARY KEY de la tabla EMPLOYEES mientras que DEPARTMENT_ID es su FOREIGN KEY que es la que relaciona con la PRIMARY KEY de la tabla DEPARTMENTS.

Esquema HR.

En esta capacitación se utilizará el esquema HR (Human resources) donde se tendrá acceso a distintas tablas de los empleados de una empresa. Se podrán consultar datos como nombre, email, dirección, departamento, jefes, cargos, salario, fecha de ingreso etc. Todos estos datos alojados de manera independiente en distintas tablas sin embargo relacionadas entre sí por datos en común.



Introducción a procedimientos en SQL Developer.

Oracle SQL Developer ofrece una hoja de trabajo en SQL donde se pueden consultar datos escribiendo instrucciones SQL sencillas o complejas.

1. Requisitos de software

- Tener instalado **Oracle Database**
- Tener acceso a la cuenta de usuario de ejemplo **HR(Human Resources)**.

2. Examinar los datos.

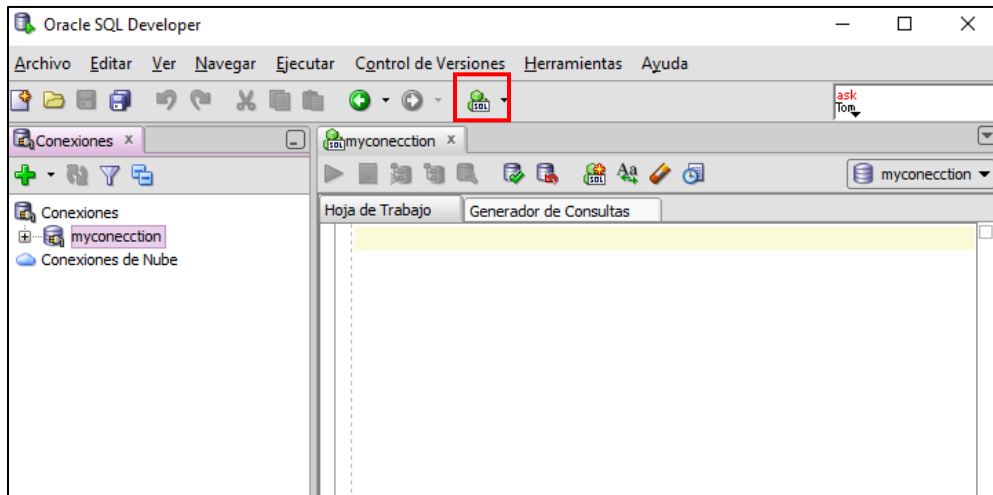
En una conexión de base de datos se puede explorar el esquema, realizar consultas y modificar los datos.

1. Una vez creada una conexión a la base de datos se debe asegurar de tener abierta la ventana de hoja de trabajo en SQL (Al ejecutar el programa, se arrojará la hoja de trabajo en SQL). En el caso de que la hoja no este abierta use el menú para poder abrirla o bien simplemente seleccionando el siguiente icono:



o también pulsar ALT + F9

2. Una vez conectado, deberá aparecer una ventana de la hoja de trabajo:



3. Entonces ya puede empezar. Por ejemplo puede consultar todos los datos de la tabla EMPLOYEES
 - Para poder ejecutar las instrucciones que introduce en su hoja de trabajo, puede dar clic en Execute Statement (Ejecutar instrucción) o pulsar directamente F9

```
SELECT * FROM EMPLOYEES;
```

- Muestra todos los datos de la tabla EMPLOYEES.

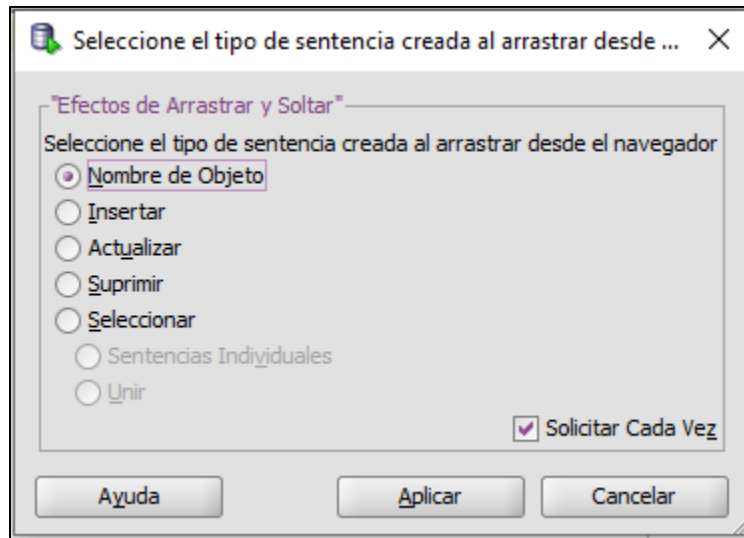
Podrá ver que los resultados (numero de filas recuperadas) aparecerá debajo en la pestaña results (resultados)

Resultado de la Consulta x											
Todas las Filas Recuperadas: 20 en 0.115 segundos											
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	100	Steven	King	SKING	515.123.4567	17/06/2011	AD_PRES	24000	(null)	(null)	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21/09/2009	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13/01/2009	AD_VP	17000	(null)	100	90
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03/01/2014	IT_PROG	9000	(null)	102	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21/05/2015	IT_PROG	6000	(null)	103	60
6	107	Diana	Lorentz	DLORENTZ	590.423.5567	07/02/2015	IT_PROG	4200	(null)	103	60
7	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16/11/2015	ST_MAN	5800	(null)	100	50
8	141	Trenna	Rajs	TRAJS	650.121.8009	17/10/2011	ST_CLERK	3500	(null)	124	50
9	142	Curtis	Davies	CDAVIES	650.121.2994	29/01/2013	ST_CLERK	3100	(null)	124	50
10	143	Randall	Matos	RMATOS	650.121.2874	15/03/2014	ST_CLERK	2600	(null)	124	50
11	144	Peter	Vargas	PVARGAS	650.121.2004	09/07/2014	ST_CLERK	2500	(null)	124	50

4. Recuperar columnas.

Para evitar consultar todos los datos de una tabla, puede simplemente seleccionar detalladamente los datos que necesita.

 - Puede escribir el nombre de cada columna que necesita o arrastrar el nombre de la columna desde el navegador de conexiones.
 - Por ejemplo, arrastre la tabla DEPARTMENTS a la hoja de trabajo. Al momento de arrastrar pedirá que especifique como desea pegar la tabla indicada en este caso solo insertaremos el nombre:



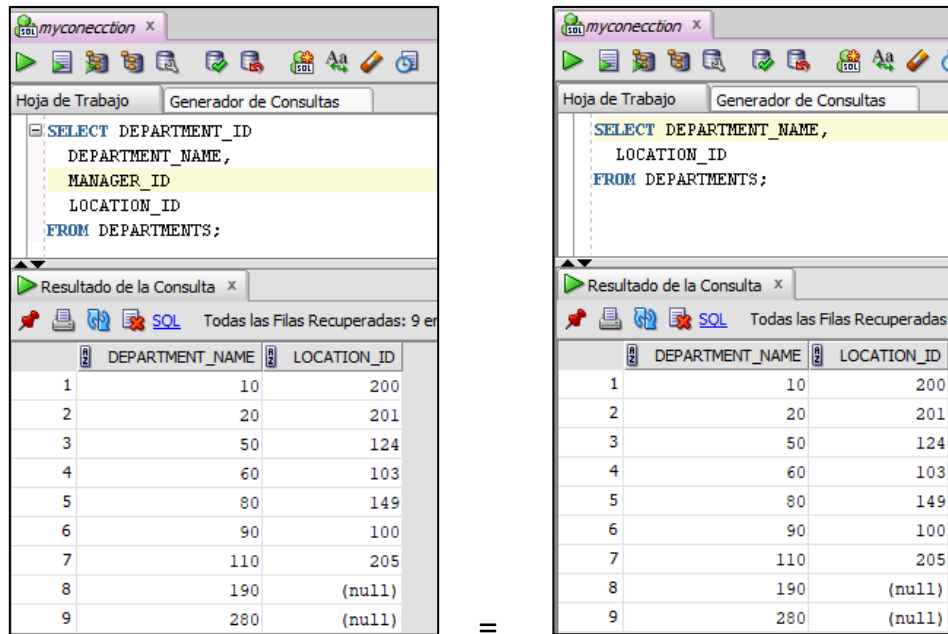
Y lo arroja en la hoja de trabajo: **DEPARTMENTS**

Esta función puede tener distintos usos y nos ahorra escribir un par de instrucciones sencillas.

- A continuación con ayuda de la instrucción anterior vamos a seleccionar los elementos de la tabla **DEPARTMENTS**:

```
SELECT DEPARTMENT_ID,  
       DEPARTMENT_NAME,  
       MANAGER_ID,  
       LOCATION_ID  
FROM DEPARTMENTS ;
```

Si se desean conocer menos columnas dentro de la instrucción se pueden borrar por completo o quitar la ',' que la separa de la siguiente. Por ejemplo si solo se quisiera saber el **DEPARTMENT_NAME** y **LOCATION_ID**, lo puede consultar de dos maneras distintas y arroja el mismo resultado:



myconnection x

Hoja de Trabajo Generador de Consultas

```
SELECT DEPARTMENT_ID,
DEPARTMENT_NAME,
MANAGER_ID,
LOCATION_ID
FROM DEPARTMENTS;
```

Resultado de la Consulta x

Todas las Filas Recuperadas: 9 en

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	200	
2	20	201	
3	50	124	
4	60	103	
5	80	149	
6	90	100	
7	110	205	
8	190	(null)	
9	280	(null)	

=

myconnection x

Hoja de Trabajo Generador de Consultas

```
SELECT DEPARTMENT_NAME,
LOCATION_ID
FROM DEPARTMENTS;
```

Resultado de la Consulta x

Todas las Filas Recuperadas

	DEPARTMENT_NAME	LOCATION_ID
1	10	200
2	20	201
3	50	124
4	60	103
5	80	149
6	90	100
7	110	205
8	190	(null)
9	280	(null)

5. Seguido se requerirá de restringir la cantidad de registros devueltos. Para hacer esto se requiere de la cláusula **WHERE**. Sin embargo por el momento estas clausulas pueden ser complejas sin embargo se comenzará con una sencilla:
 - La cláusula **WHERE** va después de la clausula **FROM** y de la misma manera se terminara con un ';' para indicar que finaliza la instrucción. En el siguiente ejemplo se mostraran solo los EMPLOYEES que tienen como DEPARTMENT_ID = 90 :

```
SELECT EMPLOYEE_ID,
FIRST_NAME,
LAST_NAME,
EMAIL,
PHONE_NUMBER,
HIRE_DATE,
JOB_ID,
SALARY,
COMMISSION_PCT,
MANAGER_ID,
DEPARTMENT_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 90;
```

6. La hoja de trabajo SQL permite utilizar los comandos **SQL*Plus** es necesario que interprete dichos comandos antes de pasarlos a la base de datos. Existen algunos que no son compatibles por lo cual se ignoran y la instrucción no se envia a la base de datos.

- El comando **DESCRIBE** pertenece a los comandos de **SQL*Plus** y su fin es describir un objeto de esquema.

Por ejemplo, se describirá la tabla EMPLOYEES:

DESCRIBE EMPLOYEES		
Nombre	Nulo	Tipo

EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Como resultado se obtienen los nombres de la columna de la tabla, si es **NOT NULL** o en su defecto **NULL** y también su tipo de dato.

7. En el caso de que se requieran datos mas específicos se puede consultar la tabla de diccionario **USER_OBJECTS** que contiene todos los objetos a los que tiene acceso el usuario.

Por ejemplo se usa para encontrar detalles mas específicos:

DESCRIBE USER_OBJECTS		
Nombre	Nulo	Tipo

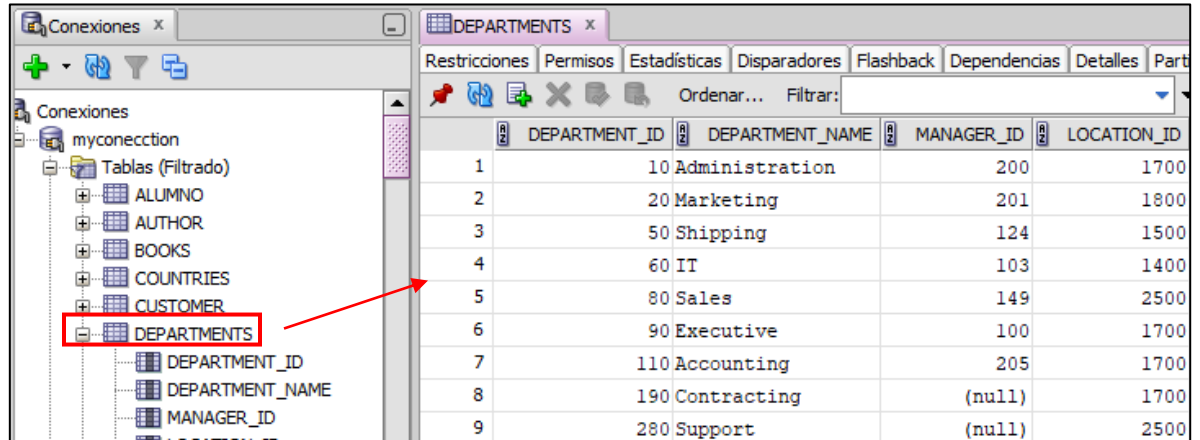
OBJECT_NAME		VARCHAR2(128)
SUBOBJECT_NAME		VARCHAR2(128)
OBJECT_ID		NUMBER
DATA_OBJECT_ID		NUMBER
OBJECT_TYPE		VARCHAR2(23)
CREATED		DATE
LAST_DDL_TIME		DATE
TIMESTAMP		VARCHAR2(19)
STATUS		VARCHAR2(7)
TEMPORARY		VARCHAR2(1)
GENERATED		VARCHAR2(1)
SECONDARY		VARCHAR2(1)
NAMESPACE		NUMBER
EDITION_NAME		VARCHAR2(128)

Ahora que se conoce las columnas que definen la tabla **USER_OBJECTS** se puede escribir una instrucción **SELECT** para conocer los datos que la componen.

Por ejemplo se consultaran los nombres y status de los objetos:

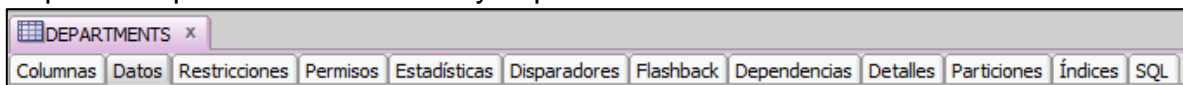
```
SELECT OBJECT_NAME, STATUS FROM USER_OBJECTS;
```

O directamente consultarlas, al seleccionar desde nuestra conexión alguna tabla, se mostrarán los datos de la misma. Por ejemplo en la tabla DEPARTMENTS



	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700
9	280	Support	(null)	2500

Una vez que se desglosa la tabla podemos observar una serie de opciones disponibles para visualizar datos y especificaciones de la tabla.



DEPARTMENTS x
Columnas
Datos
Restricciones
Permisos
Estadísticas
Disparadores
Flashback
Dependencias
Detalles
Particiones
Índices
SQL

Columnas: Muestra las respectivas columnas de la tabla junto con su tipo de dato y posibles comentarios.

Datos: Todos los datos que contiene la tabla.

Restricciones: Muestra las Primary Key, Foreign key, Checks en caso de contener y los parámetros en los que fueron creadas.

Permisos: Privilegios que se otorgaron a otro usuario respecto a esa tabla.

Estadísticas: Estadísticas de las columnas de la tabla.

Dependencias: Vistas creadas.

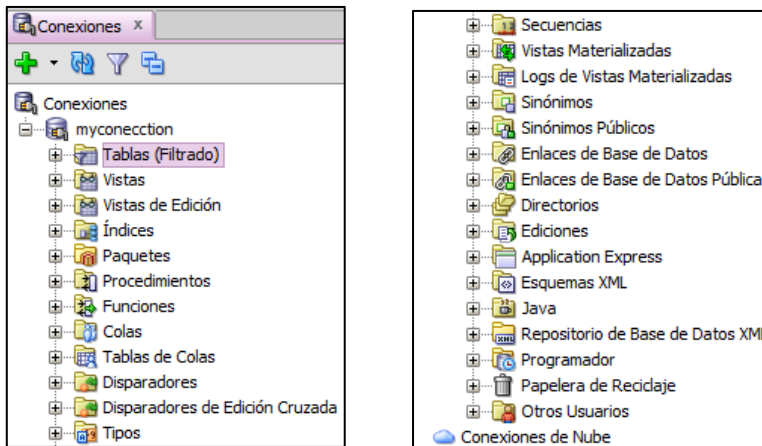
Particiones: En caso de tener muestra si la tabla fue particionada

índices: índices creados

SQL: Código DDL de la tabla seleccionada.

Acceso a los objetos de schema.

Para poder acceder a los objetos del schema basta con simplemente seleccionar nuestra colección dentro de la base de datos. Dentro de estos objetos se encuentran las Tablas que están almacenadas dentro de la BD.



Para cada uno de los objetos que son parte del esquema es posible crearlos a partir de un script (o sea código directamente) eso abre la posibilidad de modificaciones y alteraciones en el código, esto para especificar lo que se quiere agregar o hacer dependiendo el procedimiento.

Se pueden agregar parámetros. Los objetos se almacenan dentro de la BD y pueden utilizarse. Algunos se diseñan para esquemas específicos, sin embargo otros están abiertos a modificaciones.

Comando DESCRIBE

Este comando muestra la estructura de una tabla

```
DESCRIBE tablename
```

Por ejemplo: Se desea conocer la estructura de la tabla EMPLOYEES

```
DESCRIBE employees
```

```
DESCRIBE Employees
Name          Null    Type
-----
EMPLOYEE_ID   NOT NULL NUMBER(6)
FIRST_NAME    VARCHAR2(20)
LAST_NAME     NOT NULL VARCHAR2(25)
EMAIL         NOT NULL VARCHAR2(25)
PHONE_NUMBER  VARCHAR2(20)
HIRE_DATE     NOT NULL DATE
JOB_ID        NOT NULL VARCHAR2(10)
SALARY        NUMBER(8,2)
COMMISSION_PCT NUMBER(2,2)
MANAGER_ID    NUMBER(6)
DEPARTMENT_ID NUMBER(4)
```

v

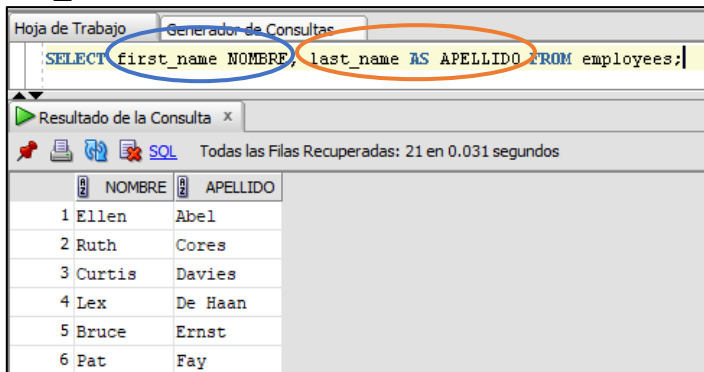
Alias y operador de concatenación '||'

Cuando se consulta una tabla a través de SELECT, en el resultado el script se muestran las columnas consultadas por el usuario y su nombre genérico, sin embargo a los nombres de las columnas se le pueden poner alias para que al

momento de consultarlas sea mas fácil su visualización y entendimiento con el usuario.

El alias puede ir seguido de el nombre de la columna o le añadimos el conector AS para indicar el alias. El alias puede contener espacios siempre y cuando este entre comillas(“ ”) y no puede llamarse igual a otras columnas o variables declaradas.

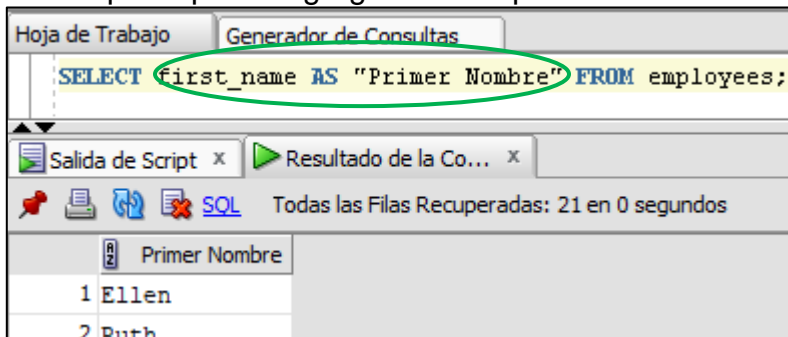
Por ejemplo: Se consulta de la tabla EMPLOYEES la columna first_name y last_name. Para estas columnas se les añadira el alias NOMBRE y APELLIDO.



The screenshot shows the SQL Developer interface. The query editor contains the following SQL statement: `SELECT first_name NOMBRE, last_name AS APELLIDO FROM employees;`. The words `NOMBRE` and `AS APELLIDO` are circled in orange. Below the query editor, the 'Resultado de la Consulta' (Query Result) window shows the results of the query. It displays a table with two columns: 'NOMBRE' and 'APELLIDO'. The results are as follows:

	NOMBRE	APELLIDO
1	Ellen	Abel
2	Ruth	Cores
3	Curtis	Davies
4	Lex	De Haan
5	Bruce	Ernst
6	Pat	Fay

Si se requiere que la columna tenga mas especificaciones, se agregan comillas dobles para que se agreguen los espacios



The screenshot shows the SQL Developer interface. The query editor contains the following SQL statement: `SELECT first_name AS "Primer Nombre" FROM employees;`. The phrase `AS "Primer Nombre"` is circled in green. Below the query editor, the 'Resultado de la Co...' (Query Result) window shows the results of the query. It displays a table with one column: 'Primer Nombre'. The results are as follows:

	Primer Nombre
1	Ellen
2	Ruth

El operador de concatenación || combina dos o mas columnas para dar como resultado en una fila o bien en una línea de texto. Si se quiere añadir texto recordar que se incorporan comillas simples (' ')

Por ejemplo: Usando de referencia el ejemplo anterior se mostrara "El empleado se llama: " seguido de los datos de las columnas last_name y first_name de la taba employees.

EXISTS y NO EXISTS:

Los operadores EXISTS y NO EXISTS se emplean para determinar si hay datos o no hay datos en una lista determinada de valores

Estos operadores pueden emplearse con subconsultas correlacionadas para restringir el resultado de una consulta exterior a los registros que cumplen con la subconsulta.

Estos operadores retornan un "TRUE" si la subconsultas retornan registros o "false" si no se regresan registros

Cuando se utiliza el operador EXISTS Oracle analiza si hay datos que coinciden con la subconsulta, no se devuelve ningún registro, es como una prueba de existencia. Oracle termina con la recuperación de registros cuando por lo menos un registro cumple con la condición WHERE de la subconsulta.

Por ejemplo:

```
where exists (SUBCONSULTA);
```

ERD and Data Modeling

ERD (Entity Relationship Diagram) es una representación lógica del sistema en el mundo real.




El mecanismo para identificar una única fila en una lista PRIMARY KEY es usando una FOREIGN KEY para relacionar los datos de otra tabla.

Para el diagrama ERD existen distintos tipos de relaciones y elementos.

Los modelos de datos conceptuales establecen una visión amplia de lo que debería incluirse en el conjunto de modelos. Los diagramas ERD se pueden utilizar como la base de modelos de datos lógicos.



- **Símbolos de entidades.**

Las entidades son objetos o conceptos que representan datos importantes .
Existen tres tipos de entidades comúnmente utilizados en ERD

Símbolo de entidad	Nombre	Descripción
	Entidad fuerte	Estas figuras son independientes de otras entidades y con frecuencia se les denomina entidades matriz ya que a menudo tienen entidades débiles que dependen de ellas. También tendrán una clave primaria, que distinga a cada suceso de la entidad.
	Entidad débil	Las entidades débiles dependen de algún otro tipo de entidad. No tienen claves primarias y no tienen significado en el diagrama sin su entidad matriz.
	Entidad asociativa	Las entidades asociativas relacionan las instancias de varios tipos de entidades. También contienen atributos que son específicos a la relación entre esas instancias de entidades.





- Símbolos de relación.

Las relaciones se usan para documentar la interacción entre dos entidades.

Símbolo de relación	Nombre	Descripción
	Relación	Las relaciones son asociaciones entre dos o más entidades.
	Relación débil	Las relaciones débiles son conexiones entre una entidad débil y su propietario.

- Símbolos de atributos.

Los atributos son características de la entidad que ayudan a los usuarios a entender mejor la base de datos. Los atributos se incluyen para agregar detalles de las diversas entidades que se sdestacan en ERD.

Símbolo de atributo	Nombre	Descripción
	Atributo	Los atributos son las características de una entidad, una relación de muchos a muchos, o una relación de uno a uno.
	Atributo de varios valores	Los atributos de valores múltiples son aquellos que pueden tomar más de un valor.
	Atributo derivado	Los atributos derivados son atributos cuyos valores se pueden calcular a partir de valores de atributos relacionados.
	Relación	Las relaciones son asociaciones entre dos o más entidades.

- **Símbolos Físicos.**

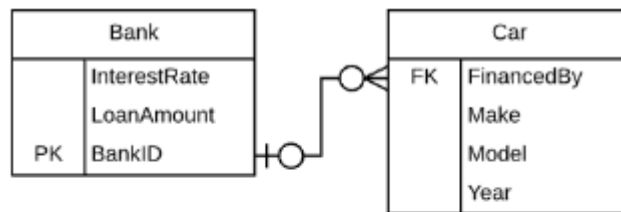
El modelos de datos físicos es el mas detallado de ERD y representa el proceso de agregar información a la base de datos. Los modelos ERD físicos muestran todas las estructuras de las tablas, incluidos *nombre de columna*, *primary key (clave primaria)*, *foreign key (clave foranea)* y relaciones entre tablas

Campos: Representan la parte de una tabla que establece lo atributos de una entidad. Generalmente son vistos como columnas.

Claves: Son una forma de categorizar a los atributos. La claves se utilizan para vincular diversas tabals en una base de datos entre si, de manera que sea lo mas eficiente posible.

- **Clave primaria(PRIMARY KEY):** Son un atributo o combinación de atributos que identifican de forma única una y solo una instancia de una entidad.
- **Clave foránea (FOREIGN KEY):** Son creadas siempre que un atributo se relacione con otra entidad en una relación de uno a uno o de uno a muchos.

Por ejemplo:

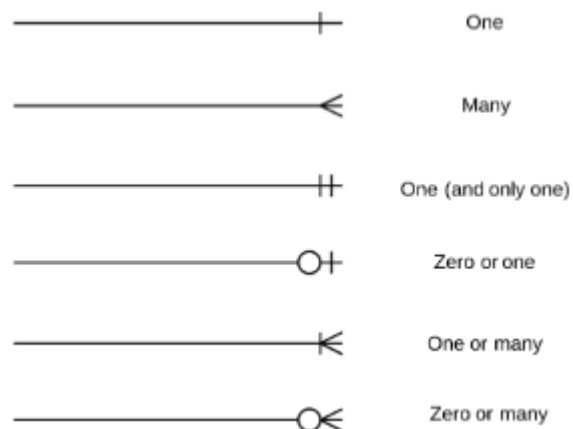


Cada auto solo puede ser financiado por un banco, por lo tanto la clave primaria IdBanco de la tabla Banco se usa como la clave extranjera FinanciadoPor en la tabla Auto. Este IdBanco se puede usar como la clave extranjera para múltiples autos.

- Cardinalidad y ordinalidad

Se refiere al número máximo de veces que una instancia en una entidad se puede relacionar con instancias de otra entidad. Por otro lado la ordinalidad es el numero mínimo de veces que una instancia en una entidad se puede asociar con una instancia en la entidad relacionada.

Estas se muestran a través de un estilo de una línea y su extremo según el estilo de una línea:



Curso de SQL avanzado y PL/SQL básico para Oracle, Cédric Simon, 2008.
Recuperado el 20/12/2022. "PL/SQL Básico".
"Temario básico de SQL", Obtenido de
<https://www.tutorialesprogramacionya.com/oracleya/> el 10/11/2022