FUNDAMENTOS DE BASES DE DATOS y SQL

"¡Los datos! ¡Los datos! ¡Los datos!, gritó con impaciencia. ¡No puedo hacer ladrillos sin arcilla!"

(Sherlock Holmes)

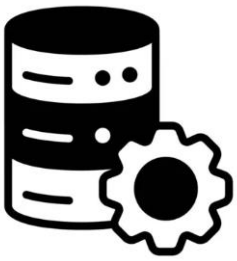
DATOS e INFORMACIÓN

En la actualidad, la cantidad de datos e información que recopilan las organizaciones es más grande que nunca, pero también es cada vez más compleja. Esta información debe estar disponible para quienes la necesiten. Actualmente las organizaciones utilizan sistemas de información para almacenar y procesar esos volúmenes de información. Un sistema de información podría ser un fichero metálico de oficina que contienen cajones donde se almacenan y recuperan carpetas. Sin embargo, la mayoría de las empresas hoy en día utilizan una base de datos para automatizar sus sistemas de información. El propósito de una base de datos es recopilar, almacenar y recuperar información para ser usada por las aplicaciones.

Pero tener muchos datos no significa contar con información. Los términos datos e información se utilizan a veces indistintamente, pero no son lo mismo. Los datos representan elementos en bruto o hechos no procesados, desde valores numéricos y representaciones simbólicas hasta texto e imágenes. Por ejemplo, los datos podrían incluir precios individuales, pesos, direcciones, edades, nombres, temperaturas, fechas o likes de redes sociales. Hay dos tipos principales de datos:

- Los datos cuantitativos se proporcionan en forma numérica, como el peso, volumen o precio de un artículo.
- Los datos cualitativos son descriptivos, pero no numéricos, como el nombre, el genero, el color de los ojos de una persona o las calificaciones de satisfacción del cliente frente a un producto.

Los datos suelen ser abundantes y fácilmente accesibles, pero pueden resultar abrumadores si no se interpretan. La información es algo más preciso, consiste en darle un significado a los datos bajo un contexto determinado. La información en una organización puede contribuir a la toma de decisiones. Por ejemplo, podemos tener un sistema que vaya capturando los montos de compras de cada cliente, pero una empresa podría utilizar esos datos sin procesar para convertirlos en información, con estos datos podría predecir las ventas que tendría a futuro o segmentar los clientes basado en su historial de compras, con esta información la empresa podría tomar mejores decisiones, mejorando sus estrategias de marketing, optimizar ciertas operaciones y desarrollar un plan de ofertas de productos según la temporada. Los datos y la información también pueden clasificarse según su forma de almacenamiento es decir en datos estructurados y no estructurados . Los datos estructurados tienen un formato predefinido y siguen una estructura clara. Se presentan en tablas con filas y columnas, como las que se encuentran en una base de datos relacional. Por ejemplo, considere los datos transaccionales de una compra on-line. En estos datos, cada registro tendrá una fecha de compra, una hora de compra, un importe total, un numero de cliente, los códigos de los artículos comprados, la información de pago y un código de confirmación de compra. Debido a que cada campo tiene una finalidad definida, facilita la consulta manual de estos datos. También es fácil para los algoritmos de aprendizaje automático identificar patrones y, en muchos casos, identificar anomalías fuera de esos patrones. Por el contrario, los datos no estructurados son aquellos que no tienen una estructura predefinida y que, por lo tanto, no se pueden organizar fácilmente en tablas o en bases de datos relacionales. Estos datos pueden ser desde e-mails, imágenes, audios, videos, publicaciones en redes sociales y un largo etcétera. Dada su naturaleza tan diversa, los datos no estructurados son mucho más complejos de analizar y procesar. Hay disponible una gran variedad de herramientas y tecnologías que te permiten gestionar y analizar datos no estructurados entre ellos el NLP (Procesamiento de Lenguaje Natural) y ML (Aprendizaje Automático).

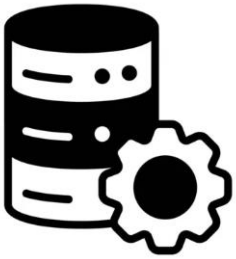


Nota

Las consultas de los datos mediante el lenguaje de consulta estructurado (SQL) son la base fundamental del análisis de datos estructurado.

BASES DE DATOS

Las Bases de Datos (del termino en Ingles database, a menudo abreviada DB) son un componente esencial de la vida cotidiana en la sociedad moderna. Actualmente, la mayoría de nosotros nos enfrentamos a diversas actividades que implican cierta interacción con una base de datos. Por ejemplo, comprar a través un sitio de comercio electrónico, realizar reservas de pasajes de avión, visualizar mapas a través de un GPS, compartir información en una red social u obtener turnos de atención medica son todas actividades que implican la interacción con bases de datos.



Nota

Las aplicaciones de streaming de música en Internet como Spotify o Deezer hacen un uso intensivo de las bases de datos, por ejemplo, para almacenar los amplios catálogos de archivos de música procedentes de diferentes álbumes de diferentes artistas.

De una manera simple, una base de datos es un repositorio que permite almacenar la información de forma ordenada con diferentes propósitos y usos. Junto con las bases de datos, el elemento fundamental para el aprovechamiento de estas son los Sistemas Gestores de Bases de Datos (SGBD o DBMS, del inglés DataBase Management System) o simplemente Motor de la base datos. Este software representa un elemento intermedio entre los propios datos y los programas que van a hacer uso de ellos, facilitando las operaciones a realizar sobre aquellos. El trabajo de un motor de base de datos no es sencillo piense en una base de datos puede tener miles de millones de datos y ser utilizada simultáneamente por miles de usuarios, que a su vez pueden utilizar diversos programas, no todos ellos del mismo tipo. El SGBD debe proporcionar a todos ellos la metodología adecuada para extraer del conjunto de datos completo cuanto sea necesario en cada caso. Los SGBD o DBMS tienen ciertas características:

Independencia: Los datos son independientes de las aplicaciones que los usan, así como de los usuarios.

Disponibilidad: Se facilita el acceso a los datos desde contextos, aplicaciones y medios distintos, haciéndolos útiles para un mayor número de usuarios.

Mayor seguridad (protección de los datos): Los datos se alojan en una ubicación central segura y no todos los usuarios de un sistema deberían poder acceder a dichos datos, se les debe asignar distintos permisos a los grupos de usuarios.

Menor redundancia: La repetición de datos no controlada produce un costo de almacenamiento y de acceso mayor a los necesarios. En segundo lugar, y más importante, la redundancia introduce la posibilidad de inconsistencia (Es decir, tener diversas copias de los mismos datos donde la información de los mismos puede no coincidir).

Copias de seguridad de datos: Hacer copias de seguridad de datos desde un solo lugar es simple.

Sin dependencia de ningún lenguaje de programación: Otro beneficio más de las bases de datos es que es independiente de cualquier tipo de lenguaje de programación. Esto significa que no es necesario conocer ningún lenguaje de programación específico para acceder a una base de datos. Escribir consultas SQL o NoSQL sería suficiente independientemente del lenguaje de programación que se utilice en la aplicación.

USUARIOS DE SGBD

Hay distintos tipos de usuarios de los SGBD entre ellos podemos mencionar:

- DBA (Administrador de Bases de Datos): Son aquellos que están a cargo de la administración y mantenimiento de los SGBD.
- Software developer (Desarrolladores de Software): Son los que construyen los productos de Software que involucran bases de datos.
- Database developer (Desarrolladores de Base de Datos): Son los que diseñan, implementan y administran las bases de datos.
- Data analyst (Analistas de Datos): Trabaja con los datos de la base de datos para obtener información procesable para el negocio.

TIPOS DE SGBD

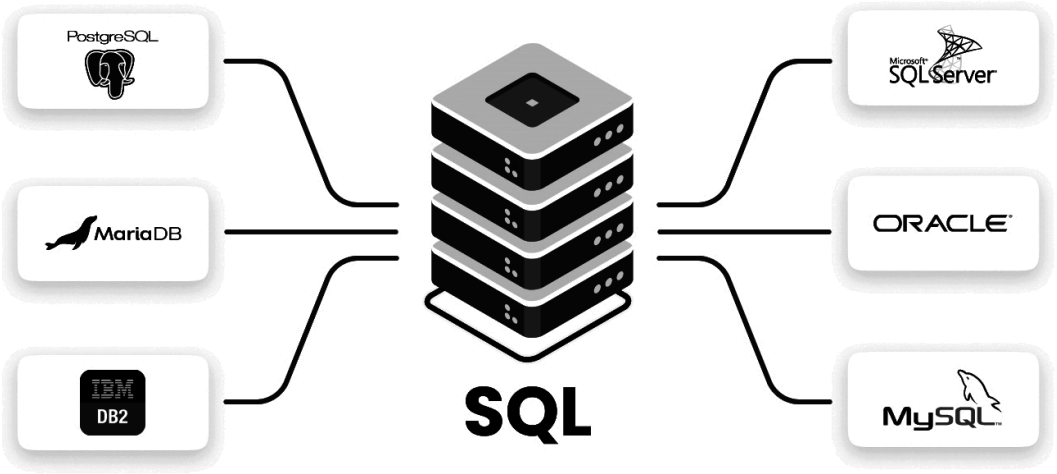
Dentro del sector de las bases de datos nos encontraremos con bases de datos relacionales (las más usadas actualmente) y por otro lado las bases de datos NoSQL. Junto a estos dos tipos de bases de datos aparecen las bases de datos híbridas (SQL/NoSQL) y bases de datos as a service.

BASES DE DATOS RELACIONALES

Las bases de datos relacionales se hicieron predominantes en la década de 1980 y hoy en día constituyen el modelo de bases de datos más utilizado en la actualidad. Utiliza para ello un esquema basado en tablas. Las tablas (relaciones) contienen un número dado de registros (filas en la tabla), así como campos (columnas), lo que da lugar a una correcta estructuración y un acceso eficiente. Por ejemplo, una base de datos de un negocio incluiría una tabla que describiera a un cliente con columnas para el nombre, dirección, número de teléfono, y así sucesivamente. Así mismo el motor de la base de datos provee un lenguaje especial denominado SQL (Structured Query Language, Lenguaje Estructurado de Consultas) que permiten a los usuarios de las bases de datos realizar consultas. Actualmente, SQL es el lenguaje estándar de los SGBD relacionales. Los principales motores de bases de datos relacionales son Microsoft SQL, MySQL, y Oracle Database .

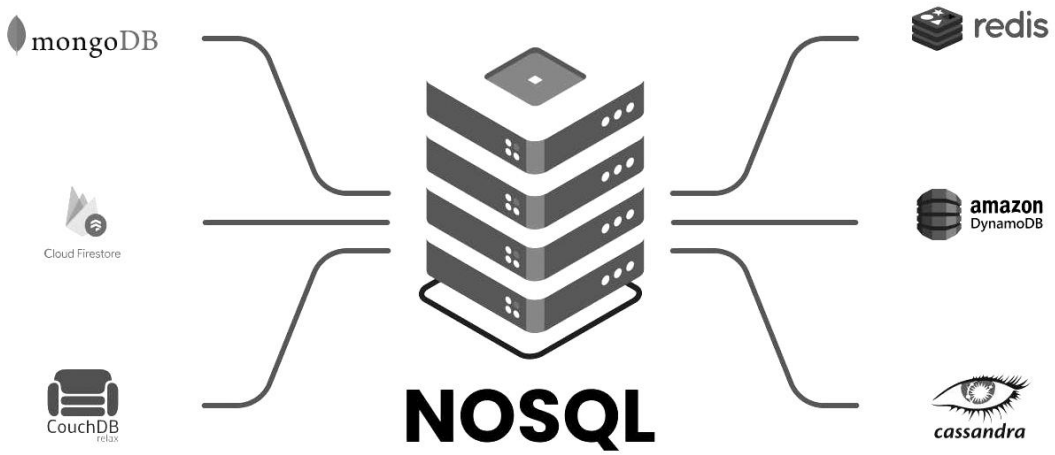
¿QUÉ ES SQL?

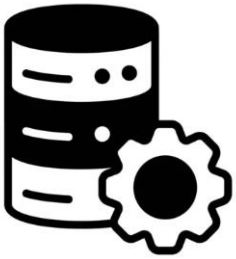
SQL (Structured Query Language, Lenguaje Estructurado de Consultas) es un lenguaje de programación que utilizan casi todas las bases de datos relacionales para consultar, manipular y definir los datos, además de para proporcionar control de acceso. SQL se desarrolló por primera vez en IBM en la década de 1970 con Oracle como uno de los principales contribuyentes, lo que dio lugar a la implementación del estándar ANSI SQL. El SQL ha propiciado muchas ampliaciones de empresas como IBM, Oracle y Microsoft.



BASE DE DATOS NoSQL

En la década de 1980, se hicieron populares las bases de datos relacionales, No fue sino hasta mediados y finales de la década del 2000 que otros modelos de datos flexibles comenzaron a adoptarse y aumentó su uso significativamente. Para diferenciar y categorizar estas nuevas clases de bases de datos y modelos de datos, se acuñó el término NoSQL (Not Only SQL, No solo SQL), estas base de datos surgieron como respuesta al crecimiento de Internet y la necesidad de acelerar la velocidad y el procesamiento de los datos no estructurados. En este tipo de bases de datos los sistemas de almacenamiento de información no cumplen con el esquema relacional ya que no imponen una estructura de datos en forma de tablas y relaciones entre ellas, en ese sentido son más flexibles, ya que suelen permitir almacenar información en otros formatos como clave-valor, Mapeo de Columnas, Documentos o Grafos.
Las bases de datos NoSQL más populares son MongoDB, CouchDB, Apache Cassandra, Amazon DynamoDB, Firestore y Redis.





Nota

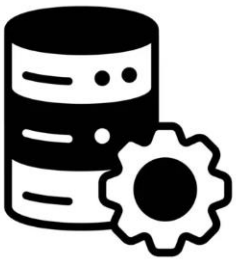
Disney+ ofrece una amplia biblioteca de contenido digital a sus subscriptores mediante el uso de bases de datos NoSQL.

MODELO HÍBRIDO

Algunas bases de datos SQL están adoptando características NoSQL y viceversa. Esto es lo que llamamos modelo hibrido. Cada vez más sistemas usan este tipo de modelo. Por ejemplo, IBM ha extendido su base de datos DB2 para ofrecer la posibilidad de utilizar bases de datos NoSQL

BASES DE DATOS COMO SERVICIOS

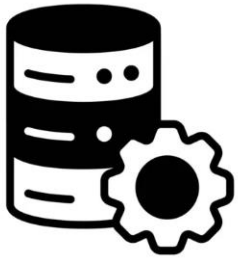
En términos simples, las bases de datos como servicios (DBaaS, Database as a Service) son bases de datos que operan en Datacenters en la nube con una gran cantidad de recursos, que pueden ser consumidos por personas o empresas desde cualquier parte del mundo, pagando sólo por lo que usan.



Nota

El concepto de SaaS (Software as a Service, Software como un servicio) ha existido desde hace mucho tiempo. Se trata de cualquier servicio basado en la Web. Tales servicios se acceden a través del navegador sin ocuparnos del desarrollo, mantenimiento, actualizaciones o copias de seguridad ya que todo ello es responsabilidad del proveedor. Típico ejemplo de este concepto es Gmail, Dropbox, Google Docs, etc.

Las DbaaS son simples de usar ya que, con un pago por el servicio el proveedor gestiona la automatización de la base de datos, la copia de seguridad, la escalabilidad, la alta disponibilidad, la seguridad de la misma, la aplicación de parches y la supervisión del estado. De esta manera no requiere que nuestra empresa contrate un conjunto de administradores para su gestión. Los DBaaS ofrecen tanto bases de datos relacionales (MySql, PostgreSQL), como base de datos no relacionales o noSQL cómo CouchDB o MongoDB. Dentro de esta categoría podemos mencionar a Oracle Cloud , Microsoft SQL Azure, Amazon RDS.



Nota

Existe un ranking de popularidad de motores de bases de datos. Se pueden encontrar en el link https://db-engines.com/en/ranking

OLTP vs OLAP

Todas las industrias del mundo actual utilizan algún sistema OLTP (OnLine Transactional Processing) para registrar sus datos transaccionales. Los sistemas OLTP cubren todas las operaciones del día a día como datos de compras, de ventas, de fabricación, de contabilidad y de otros tipos. Los sistemas OLTP tienen un gran número de usuarios que realizan transacciones breves y admiten consultas de base de datos simples, por lo que el tiempo de respuesta ante cualquier acción del usuario es muy rápida. Los sistemas OLTP utilizar un modelo relacional con bases de datos normalizadas evitando redundancias e inconsistencias. Por otro lado, el procesamiento analítico en línea (OLAP, OnLine Analytical Processing) es un sistema de procesamiento en línea que realiza análisis multidimensionales en grandes volúmenes de datos a alta velocidad. Este tipo de datos proviene de un data mart, un almacén de datos centralizado o almacenamiento de datos. Los OLAP son usados por los Data Warehouse. Un Data Warehouse es un sistema pensado para almacenar y procesar grandes cantidades de datos. De hecho, cuando hablamos de base de datos analítica, generalmente nos referimos a un Data Warehouse. Un almacén de datos centralizado con grandes cantidades de datos de múltiples fuentes. Gracias a sus capacidades analíticas, las organizaciones pueden obtener información empresarial valiosa a partir de los datos y mejorar las decisiones. Los almacenes de datos son entornos relacionales que se utilizan para el análisis de datos, sobre todo para datos históricos. Los sistemas OLAP suelen utilizar modelos multidimensionales desnormalizados, aunque también podrían utilizar modelos relacionales. Usualmente los tipos OLTP suelen usarse en bases de datos con gran necesidad transacciones no se basan en datos históricos. De hecho, en los entornos OLTP, los datos históricos a menudo se archivan o simplemente se eliminan para mejorar el rendimiento, en cambio los sistemas OLAP recogen esos datos mediante procesos ETL que se encargar de la extracción, trasformación y carga de datos para poder posteriormente analizar los datos.

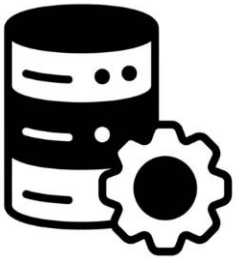
¿QUE ES EL ANALISIS DE DATOS O DATA ANALYTICS?

Los sistemas OLAP se utilizan generalmente para el análisis de datos. El análisis de datos (Data Analyst) se encarga de buscar en fuentes específicas de datos que están sin procesar tratando de encontrar tendencias y métricas que sirvan a las compañías para tomar decisiones más acertadas y obtener mejores resultados. Su aplicación más común es en sectores como la salud, ya que permite a los centros sanitarios atender a los pacientes de una forma más eficiente. También es bastante usual el uso de esta disciplina en otras industrias como la energía, ya que en base al análisis de los datos se puede optimizar el uso que se hace de los recursos en distintos lugares e incluso apostar por la automatización de distintos servicios, para así evitar gastos innecesarios. También el sector hotelero demanda los perfiles de analista, puesto que permite conocer mejor las preferencias de los viajeros y ofrecerles alternativas que se ajusten a sus gustos.

ANÁLISIS DE DATOS y CIENCIA DE DATOS

Aunque los términos se pueden usar de manera indistinta, el análisis de datos es un subconjunto de la ciencia de datos. Como vimos anteriormente el análisis de datos es el proceso de examinar, limpiar, transformar y utilizar los datos para extraer información útil y apoyar la toma de decisiones en una organización. El objetivo principal del análisis de datos es identificar patrones y tendencias en los datos, y comunicar los resultados de manera efectiva. La ciencia de datos (Data Scientist) es un campo interdisciplinario que utiliza técnicas de análisis de datos, estadística y aprendizaje automático para extraer

conocimientos y crear modelos predictivos a partir de grandes conjuntos de datos. En cuanto a su campo de aplicación, un Data Analyst aborda única y exclusivamente problemas de negocios. El Data Scientist, en cambio, actúa más allá de este campo. En pocas palabras, un analista da sentido a los datos existentes, mientras que un científico crea nuevos métodos y herramientas para procesarlos y que los usen los analistas.

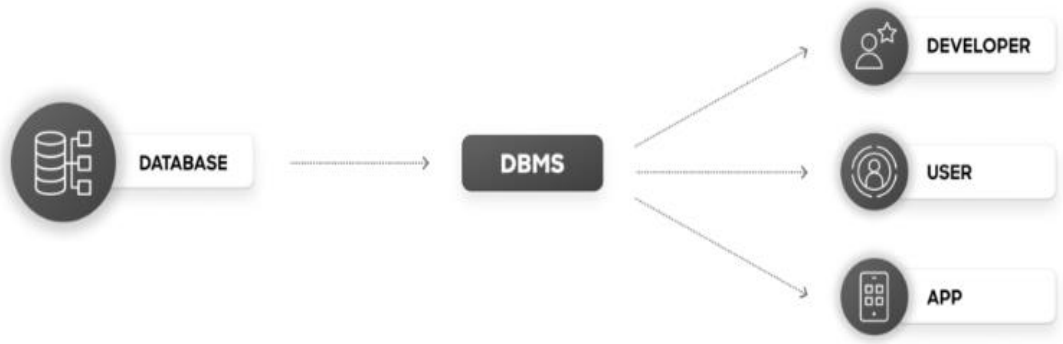


Nota

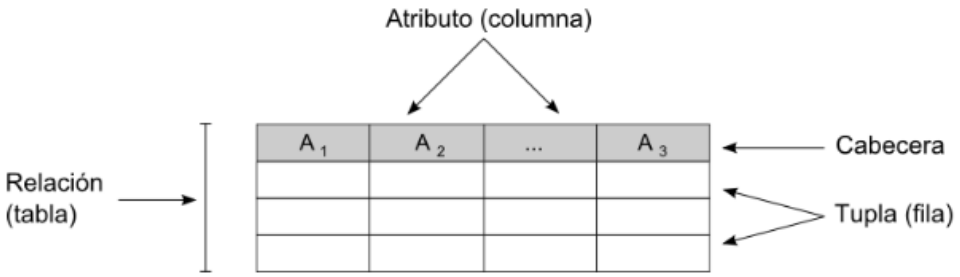
Los analistas de datos necesitan saber SQL ya que la mayoría de las empresas usan bases de datos relacionales para almacenar sus datos e información.

MODELO RELACIONAL

El modelo relacional hace la siguiente propuesta: Tómenos el conjunto de toda la información que es relevante a la operación de una organización. Para un banco, será información relativa a cuentas, clientes, empleados, equipos, etc. Para un comercio será los productos, los clientes, los proveedores, su logística etc, toda esta información apropiadamente organizada y codificada, se colocará en un gran repositorio denominado Base de Datos. Ningún programa de aplicación tendrá acceso directo a los archivos que componen la base de datos, sino que interponemos entre estos archivos y los programas de aplicación un nuevo nivel de software llamado sistema de gestión de base de datos, que abreviaremos SGBD (Sistema de Gestor de Base de Datos, también denominado DBMS, Database Management System o Motor de Base de Datos). El SGBD provee acceso a la información a un alto nivel de abstracción (es decir, el SGBD esconde ciertos detalles de cómo se almacenan y se mantiene la información), en lugar de manipular archivos, registros, índices, cilindros, el programa de aplicación se maneja enteramente en términos de clientes, cuentas, saldos, transportes etc., que son traducidos por el SGBD a su implementación física. La base de datos no estará bajo el control de ninguno de los grupos encargados de construir sistemas de aplicaciones, sino que tal control estará centralizado en un grupo o persona, el administrador de la base de datos (DBA), quien es el responsable de definir, mantener, y garantizar la integridad y seguridad de la base de datos. Finalmente, el SGBD proveerá generalmente una aplicación especial llamada lenguaje de consulta (SQL) que permite a los usuarios finales interrogar la base de datos a discreción sin tener que pasar por el analista o programador como intermediarios.



En el modelo relacional los datos se organizan en entidades también denominada relaciones. Las entidades son las unidades de almacenamiento principal dentro de las bases de datos. Dentro de estas entidades los datos están organizados a su vez en filas y columnas. Las columnas representan los distintos atributos asociados a la entidad, mientras que las filas conforman los distintos registros. Una fila se forma con un conjunto de n atributos, constituyendo una tupla.



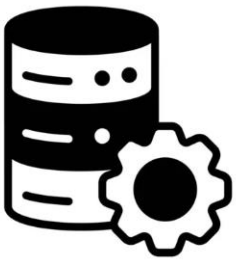
Un esquema contiene la definición de una estructura de la relación, es decir, el nombre de la relación, sus atributos y el dominio de cada atributo. El dominio especifica el tipo de dato a contener en cada columna. Por ejemplo, si se cargamos el nombre de una persona el atributo será de tipo alfanumérico, mientras que si el atributo es un código deberá ser de tipo entero. El número de filas de la tabla (el número de tuplas) se conoce como cardinalidad. Las relaciones son, por tanto, un conjunto de tuplas asociadas a un esquema. No debe existir dos tuplas iguales dentro de la misma relación y cada atributo puede solo tomar un único valor del dominio, es decir, no pueden contener listas de valores. En una relación, tanto el orden de las filas como el de las columnas son irrelevantes, pero es importante que cada atributo sea del tipo correspondiente a la columna a la que pertenece. Es decir, que sea coherente con el esquema. Una forma abreviada de definir las relaciones que forman parte de una base de datos es mediante su nombre y su esquema expresado como una lista de los atributos que lo constituyen. Por ejemplo, podemos definir una relación denominada TWEETS como:

TWEETS (Id, Texto, Fecha_Creacion, Id_Usuario)

Una base de datos contiene normalmente más de una tabla, ya que suelen ser muchos los tipos de datos a almacenar y resulta conveniente dividirlos en distintas tablas. Además de las relaciones que la tabla en sí implica, es necesario definir relaciones entre las distintas tablas, y para ello se emplean los denominados atributos clave. Un atributo clave es aquel que tiene valor único para cada tupla, pudiendo servir para representar a esta plenamente. Por ejemplo, en la tabla anterior se podría haber usado como clave el Id del tweet ya que en su carga no puede repetirse. Un esquema de relación puede contener varios atributos clave, que se conocen como claves candidatas. Normalmente, de estas se elige una como representante principal de las tuplas, y se conoce como clave primaria. Si una de las columnas de la interrelación es clave principal de la otra entidad a dicha columna se la denomina clave externa. Es decir, una clave externa es una columna (o combinación de columnas) que identifica de forma única una fila en otra tabla. Ambos conceptos de clave son extremadamente importantes. Por ejemplo, en la relación TWEETS el campo Id_Usuario puede ser clave externa en dicha entidad ya que es clave primaria en la Tabla USUARIOS (Los usuarios son los que generan los tweets). Por convención, las claves se escriben subrayadas al definir el esquema de la tabla, de tal modo que el de la tabla TWEETS quedaría de la siguiente forma:

TWEETS (Id, Texto, Fecha_Creacion,Id_Usuario)

Las interrelaciones entre tablas pueden ser de distintos tipos en función del número de elementos distintos que se vinculan de cada tabla. Por ejemplo, si tenemos la entidad USUARIOS y la queremos interrelacionar con TWEETS tendremos que un TWEET puede ser generado solo por un USUARIO de la tabla USUARIOS, mientras que un USUARIO de la tabla USUARIOS puedo haber generados muchos TWEETS distintos. Es decir, cada tupla de la tabla TWEET se relaciona con una única de la tabla USUARIOS, y cada tupla de esta última se relaciona con una o varias de la primera. Este tipo de relación se conoce como de uno a muchos. Existen otros dos tipos de relaciones además de esta: las denominadas de uno a uno y las de muchos a muchos. Por ejemplo, puede haber una relación de uno a uno entre el usuario de una red social y su perfil.



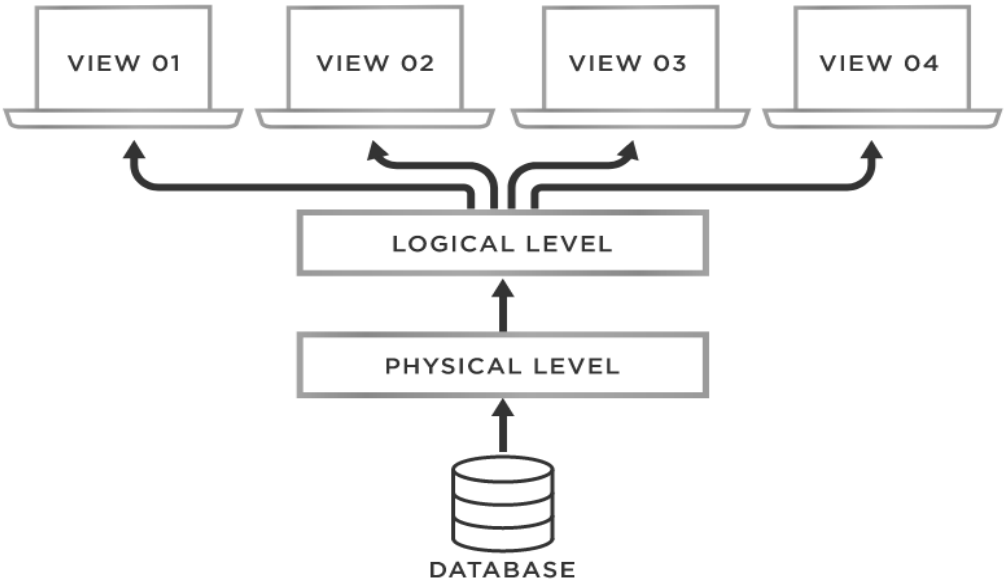
Nota

Piense que sucede en el momento de que usted agrega un nuevo contacto a su teléfono Smartphone. Si el contacto ya existiese (Porque hay otra persona con el mismo nombre y apellido) la aplicación no permitiría el ingreso del nuevo contacto. Es decir que la clave principal de la aplicación contactos es la combinación de los atributos nombre y apellido.

ARQUITECTURA DE LOS SISTEMAS DE BASES DE DATOS

En 1975, el comité ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal era el de separar los programas de aplicación de la Base de datos física. En esta arquitectura el esquema de una Base de datos se define en tres niveles de abstracción distintos:

Nivel interno o físico: El más cercano al almacenamiento físico, es decir, tal y como están almacenados en la computadora. Describe la estructura física de la Base de Datos mediante un esquema interno. Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, etcétera.
Nivel conceptual: Describe la estructura de toda la Base de Datos para un grupo de usuarios mediante un esquema conceptual. Este esquema describe las entidades, atributos, relaciones, operaciones de los usuarios y restricciones, ocultando los detalles de las estructuras físicas de almacenamiento.
Nivel externo o vista: Es el más cercano a los usuarios, es decir, es donde se describen varios esquemas externos o vistas. Cada esquema describe la parte de la Base de Datos que interesa a un grupo de usuarios en este nivel se representa la visión individual de un usuario o de un grupo de usuarios.



La arquitectura en tres niveles garantiza la independencia física y lógica de los datos de forma que si se modifica el esquema en uno de los 3 niveles no afectará a los otros dos.

DATA MODELING

El data modeling es el proceso de crear una representación visual de un sistema de información completo o partes de él para comunicar conexiones entre puntos de datos y estructuras. El objetivo es ilustrar los tipos de datos utilizados y almacenados dentro del sistema, las relaciones entre estos tipos de datos, las formas en que los datos se pueden agrupar y organizar y sus formatos y atributos. Los modelos de datos se basan en las necesidades del negocio. Las reglas y los requisitos se definen por adelantado mediante el feedback de las partes interesadas del negocio para que puedan incorporarse al diseño de un nuevo sistema o adaptarse en la iteración de uno existente. El proceso comienza con la recopilación de información sobre los requerimientos del negocio de las partes interesadas y de los usuarios finales. Estas reglas comerciales se traducen luego en estructuras de datos para formular un diseño de base de datos concreto. Idealmente, los modelos de datos son documentos “vivos” que evolucionan junto con las necesidades cambiantes del negocio. El modelo relacional contribuye en el proceso del data modeling agregando un modelo denominado entidad-relacion (ER) donde muestra no solo las entidades involucradas en la base de datos sino que también sus relaciones, en ingeniería de software, un diagrama ER a menudo es un primer paso para determinar los requisitos de un proyecto de sistemas de información. Por lo general, el flujo de trabajo para el data modeling es el siguiente:

1. **Identifica las entidades.** El proceso de modelado de datos comienza con la identificación de las cosas, eventos o conceptos que están representados en el conjunto de datos que se va a modelar. Cada entidad debe ser consistente y lógicamente diferenciada de todas las demás.
2. **Identificar atributos de cada entidad.** Cada tipo de entidad se puede diferenciar de todos los demás porque tiene una o más propiedades únicas, llamadas atributos. Por ejemplo, una entidad llamada "cliente" podría poseer atributos tales como nombre, apellido y un número de teléfono, mientras que una entidad llamada "dirección" podría incluir el nombre y número de una calle, una ciudad, provincia, país y código postal.

3. **Identificar relaciones entre entidades.** El primer borrador de un modelo de datos especificará la naturaleza de las relaciones que cada entidad tiene con las demás. En el ejemplo anterior, cada cliente "vive en" una dirección. Si ese modelo se expandiera para incluir una entidad llamada "pedidos", cada pedido se enviaría y facturaría a una dirección también.
4. **Asigne claves según sea necesario y decida un grado de normalización que equilibre la necesidad de reducir la redundancia con los requisitos de rendimiento.** La normalización es una técnica para organizar modelos de datos (y las bases de datos que representan) en la que se asignan identificadores únicos llamados claves, a grupos de datos para representar relaciones entre ellos sin repetir los datos. Por ejemplo, si a cada cliente se le asigna un id, esa clave se puede vincular tanto a su dirección como a su historial de pedidos sin tener que repetir esta información en la tabla de nombres de clientes. La normalización tiende a reducir la cantidad de espacio de almacenamiento que requerirá una base de datos, pero puede costar el rendimiento de las consultas.
5. **Finalizar y validar el modelo de datos.** El data modeling es un proceso iterativo que debe repetirse y perfeccionarse a medida que cambian las necesidades cmerciales.

LOS COMPONENTES Y LAS CARACTERÍSTICAS DE UN DIAGRAMA ER

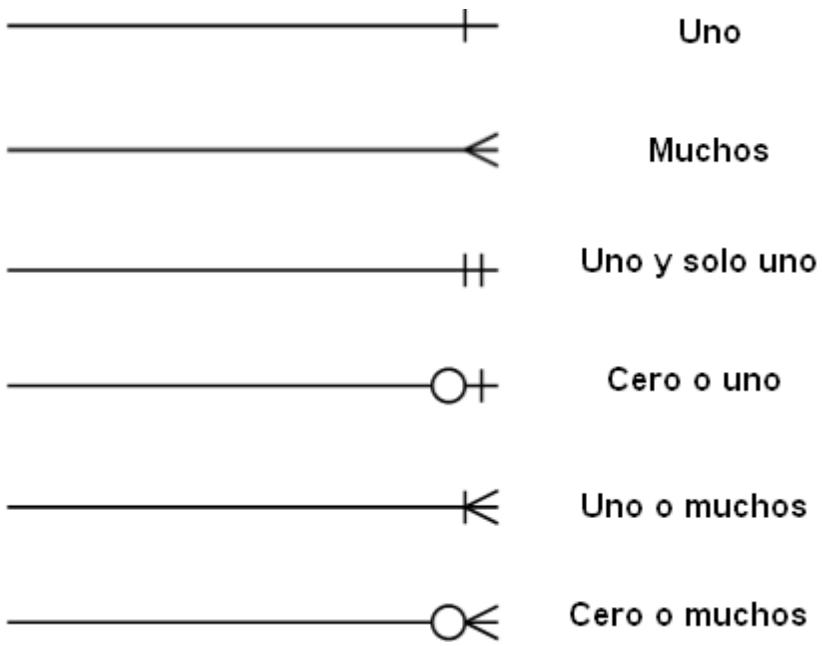
Los diagramas ER se componen de entidades, relaciones y atributos. También representan la cardinalidad, que define las relaciones en términos de números. A continuación, algunos conceptos:

Entidad: Algo que se puede definir, como una persona, objeto, concepto u evento, que puede tener datos almacenados acerca de este. Piensa en las entidades como si fueran sustantivos. Por ejemplo: un cliente, estudiante, auto o producto.

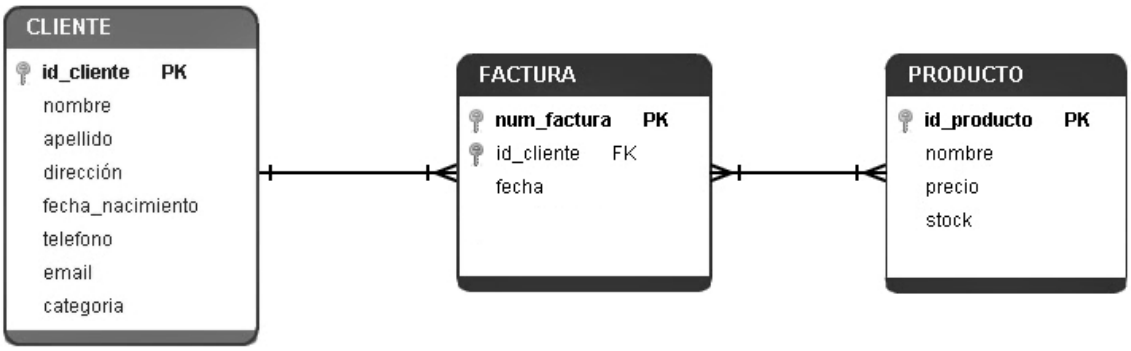
Atributo: Una propiedad o característica de una entidad. Ejemplo para la entidad estudiante puede ser su número de legajo. Una entidad puede tener tantos atributos como sean necesarios.

Relación: Indica cómo las entidades interactúan o se asocian entre sí. Piensa en las relaciones como si fueran verbos. Por ejemplo, un estudiante podría inscribirse en un curso. Las dos entidades serían el estudiante y el curso, y la relación representada es el acto de inscribirse, que conecta ambas entidades.

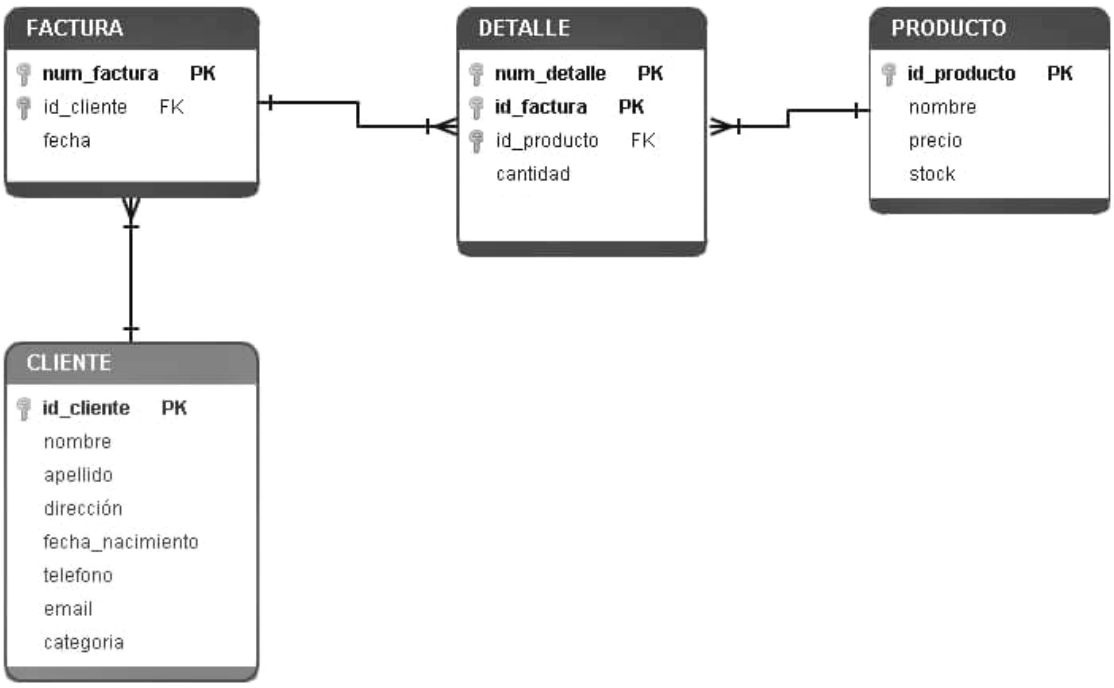
Cardinalidad: Cardinalidad se refiere al número máximo de veces que una instancia en una entidad se puede relacionar con instancias de otra entidad. Por otra parte, la ordinalidad es el número mínimo de veces que una instancia en una entidad se puede asociar con una instancia en la entidad relacionada. La cardinalidad y la ordinalidad se muestran a través del estilo de una línea y su extremo, según el estilo de notación seleccionado.



A través de un sistema de facturación vamos a resolver muchas dudas que nos hayan surgido. Los sistemas de facturación están en la mayoría de las organizaciones estos sistemas se caracterizan por almacenar los datos de las facturas de nuestros clientes que compran en nuestra empresa. La primera tarea es identificar todas las entidades posibles que vayan a intervenir en el proceso de facturación de la empresa. Para el objetivo de esta parte nos centraremos en las principales. En principio tendremos al cliente que se encarga de realizar la compra. También estará el producto que se la facturo al cliente. Y también es necesario entregar una factura para constatar la entrega del producto. Por el momento el cliente, la factura y el producto son las entidades más relevantes, cuya información debe ser almacenada en la base de datos. Luego debemos recopilar los atributos de cada entidad.



Podemos observar que hay una relación de muchos entre factura y producto. En una relación muchos a muchos debemos reemplazar esa relación por una tabla intermediaria que reciba las claves principales de ambas tablas referenciadas. Además de eso, las tablas originales deben tener una relación uno a muchos con la tabla intermedia. Esto permite optimizar y normalizar los datos. Por eso se crea la tabla detalle, en alusión a cada renglón detallado en la factura.



La clave num_detalle y id_factura son la nueva clave principal de la entidad detalle esto permite identificar a cada renglón dentro de cada factura.

PROPIEDADES DE ACID DE LOS SGBD RELACIONALES

Una transacción es un conjunto de operaciones que se ejecutan en una base de datos, y que son tratadas como una única unidad lógica por el Sistema de gestión de base de datos. Si una transacción tiene éxito, todas las modificaciones de los datos realizadas durante la transacción se confirman y se convierten en una parte permanente de la base de datos. Si una transacción encuentra errores y debe cancelarse o revertirse, se borran todas las modificaciones de los datos realizadas. En el manejo de transacciones de un SGBD se debe cumplir cuatro propiedades cruciales: atomicidad, consistencia, aislamiento y durabilidad. Estas suelen conocerse como ACID, su acrónimo en inglés.

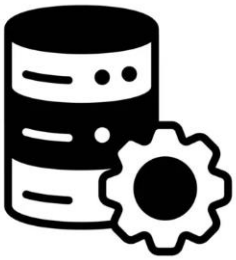
- La **atomicidad (atomicity)** garantiza que todos los pasos de una transacción se han realizado correctamente como grupo. Es decir, si falla algún paso entre las transacciones, todos los demás pasos también deben fallar o revertirse. La finalización correcta de una transacción se denomina confirmación o commit. El fallo de una transacción se denomina abortar o rollback.
- La **consistencia (consistency)** toda transacción conserva la consistencia interna de la base de datos. Si ejecuta la transacción por sí misma en una base de datos que es inicialmente consistente, cuando la transacción termina de ejecutar la base de datos es consistente de nuevo.
- El **aislamiento (Isolation)** impide que el efecto de una transacción sea visible a otros hasta que se establezca la confirmación, a fin de evitar confusiones.
- La **durabilidad (Durability)** una vez completado el cambio (confirmación), éste debe conservarse, aunque se produzcan fallos en la base de datos o el sistema completo.

¿QUE ES ORACLE DATABASE?

Oracle Corporation se fundó en 1977 y en 1979 fue el primer proveedor en ofrecer un sistema de base de datos relacional comercial. Oracle Database es su producto más conocido, su enfoque en funciones avanzadas, rendimiento, seguridad y confiabilidad la ha convertido en una opción para industrias donde la integridad de los datos y el tiempo de actividad son fundamentales. Una de las ventajas de su producto es que es multiplataforma y esto garantiza independencia de cualquier Hardware desde PCs hasta grandes mainframes. Mayormente se encuentra ejecutándose en Linux como sistema operativo, pero eso no quiere decir que pueda ejecutarse en otros sistemas operativos como Microsoft Windows. A lo largo de los años, Oracle ha evolucionado con tendencias como la computación en la nube, el big data y la inteligencia artificial para seguir siendo relevante en un panorama tecnológico cambiante.

Dentro de las características esenciales del SGBD de Oracle podemos encontrar:

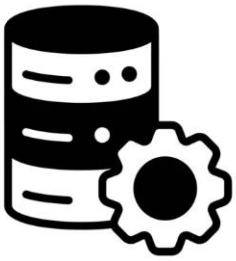
- El cumplimiento de ACID es la característica principal, ya que garantiza la seguridad y la integridad, que es de suma importancia para las empresas.
- Oracle DB utiliza estructuras de datos lógicos para almacenar datos; por lo tanto, la ubicación física de los datos se puede ubicar fácilmente.
- Es multiplataforma; por lo tanto, es compatible con múltiples hardware y varios sistemas operativos.
- El diccionario de datos de Oracle permite una fácil administración y mantenimiento.
- El rendimiento y la escalabilidad de las bases de datos de Oracle son lo mejor del mercado de base de datos.



Nota

Larry Ellison (fundador de Oracle) desarrolló un sistema de base de datos relacional que se basó en el artículo de Codd (creador del modelo relacional en base de datos). Aunque IBM tenía un prototipo de base de datos relacional (System R) realmente no creía en su éxito. Por eso Ellison confió en su motor de bases de datos y lo lanzo dos años antes que la misma IBM.

Oracle Database tiene su propia versión de SQL, Oracle SQL, que es una implementación del estándar ANSI que se extiende más allá de las características estándar de SQL. Oracle también ofrece su propio lenguaje de procedimientos denominado PL/SQL, para crear programas del lado del servidor que se almacenan en la base de datos en forma compilada.



Nota

Oracle Database Cloud Service es la base de datos en la nube de la empresa Oracle.

EL LENGUAJE SQL

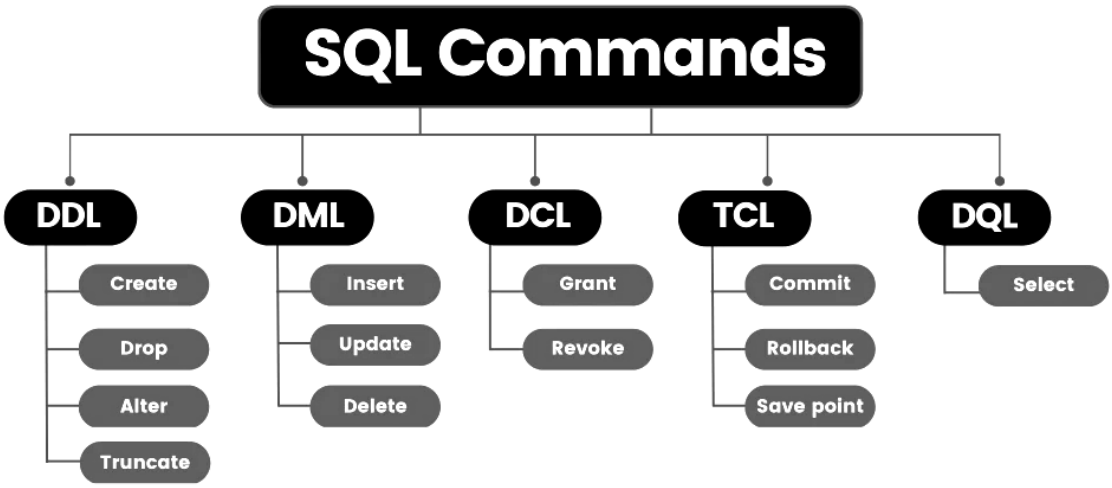
La historia de SQL comenzó en 1969, cuando el investigador de IBM, Edgar F. Codd definió el modelo de base de datos relacional. Este modelo se basa en la asociación de claves con varios datos. Por ejemplo, un nombre de usuario se puede asociar con el nombre real del usuario y un número de teléfono. Unos años más tarde, IBM creó un lenguaje para bases de datos relacionales basado en el trabajo de Codd. Este lenguaje primero se llamó SEQUEL, por Lenguaje de consulta en inglés estructurado. Después de varias implementaciones y revisiones, finalmente se llamó SQL (Abreviatura de Structured Query Language, Lenguaje Estructurado de Consulta). Las pruebas comenzaron en 1978, y luego IBM comenzó a desarrollar productos comerciales como SQL/DS en 1981 y DB2 en 1983. Aunque en su denominación se hace llamar lenguaje de consulta sus posibilidades van más allá de la simple de recuperación de datos. Normalmente, los comandos del lenguaje SQL se divide en cuatro categorías:

Lenguaje de definición de datos (DDL) : SQL usa comandos para crear y mantener bases de datos como CREATE, DROP, ALTER, TRUNCATE y COMENT.

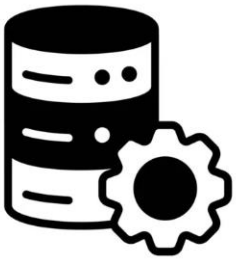
Lenguaje de consulta de datos (DQL) : Esto se refiere a los comandos SQL utilizados para recuperar datos de bases de datos. El comando SQL comúnmente utilizado aquí es SELECT.

Lenguaje de manipulación de datos (DML) : Estos se refieren a los comandos SQL utilizados para manipular datos y realizar operaciones críticas en las bases de datos, como INSERT, UPDATE y DELETE.

Lenguaje de control de datos (DCL) : Estos se refieren a los comandos de control SQL que otorgan a los usuarios permiso o acceso para realizar ciertas operaciones.



Por ejemplo, REVOKE es el comando de control que revoca el permiso de acceso otorgado a los usuarios. Oracle SQL es una implementación del estándar ANSI. Oracle SQL admite numerosas características que se extienden más allá del SQL estándar.



Nota

Las aplicaciones que usamos normalmente se pueden programar con diferentes lenguajes como Python , PHP o Ruby. Sin embargo, históricamente, las bases de datos relacionales no entienden estos lenguajes. Es por eso que aprender SQL es fundamental para los desarrollos de aplicaciones que manejan bases de datos.

¿QUÉ SON LOS ESTÁNDARES DE SQL?

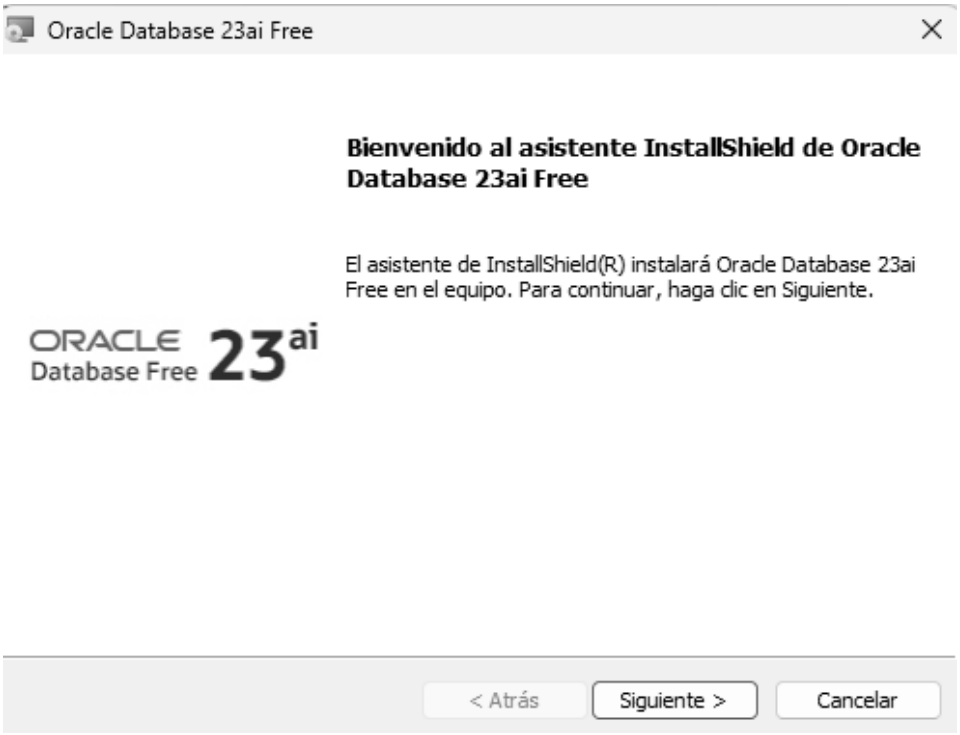
Los estándares SQL son un conjunto de pautas definidas formalmente del lenguaje SQl. El Instituto Nacional Estadounidense de Estándares (ANSI) y la Organización Internacional de Normalización (ISO) adoptaron las normas SQL en 1986. Los proveedores de software usan los estándares ANSI de SQL con el fin de crear software de base de datos SQL para los desarrolladores.

¿QUE ES PL/SQL?

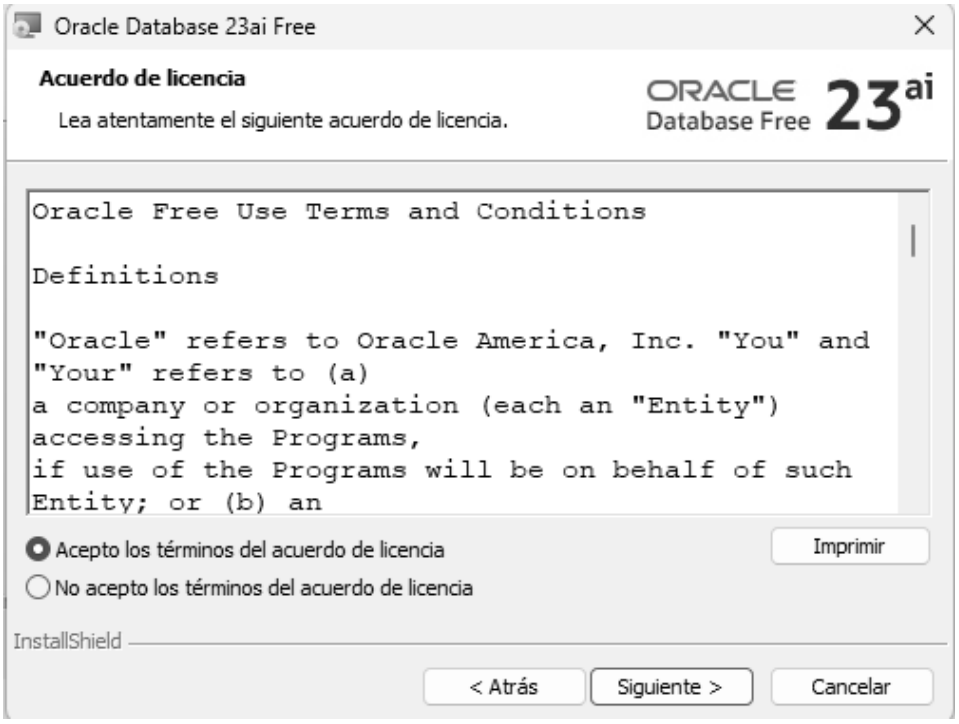
PL/SQL (Procedural Language/Structured Query Language) es un lenguaje de programación desarrollado por Oracle en 1988 como extensión de SQL. Ambos son lenguajes de bases de datos relacionales, pero entre ellos existen varias diferencias. SQL es un lenguaje declarativo, es decir, describe que hacer y no como hacerlo. PL/SQL va más allá, es un lenguaje de programación por procedimientos que posee la versatilidad de los lenguajes de programación tradicionales como C y Java. Entre sus principales características destacamos el manejo de variables, estructuras modulares, estructuras de control y toma de decisiones y el control de excepciones.

ORACLE DATABASE 23ai

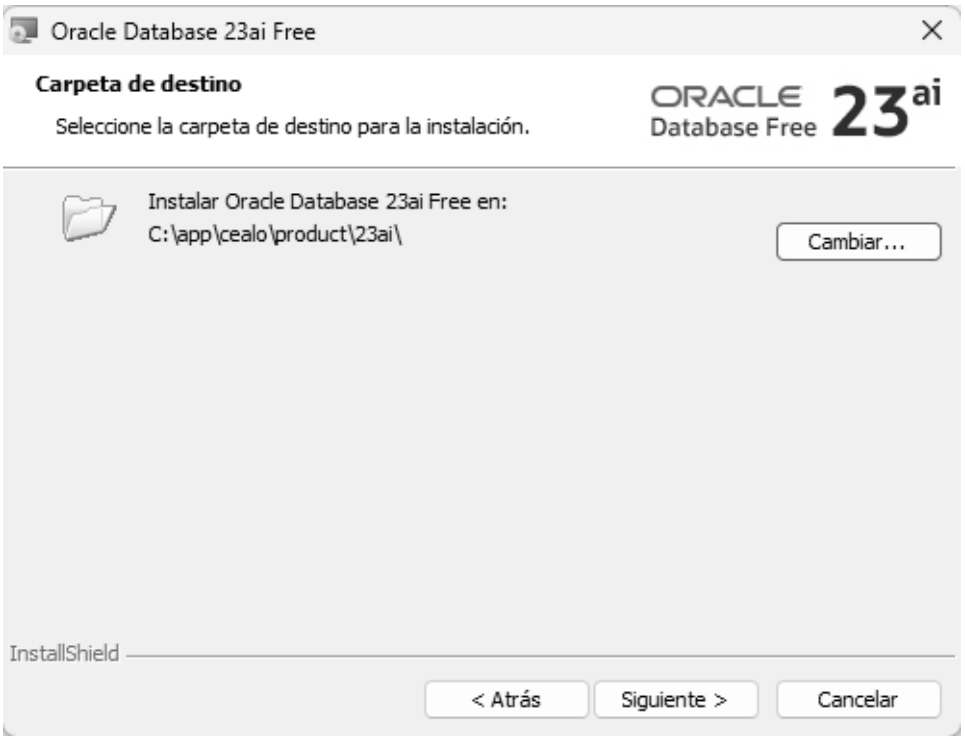
Oracle Database 23ai es la versión gratuita de Oracle Database. Es una versión reducida de la versión Oracle Database Enterprise Edition (EE) que se utiliza para iniciarse en el mundo de desarrollo de bases de datos Oracle. También se utiliza para desarrollo de pequeñas y medianas aplicaciones de escritorios y Webs. Los límites de recursos para Oracle Database Free son hasta 2 CPU para procesos en primer plano, 2 GB de RAM y 12 GB de datos de usuario en disco. Para descargar el producto debe dirigirse a la siguiente dirección: https://www.oracle.com/database/free/get-started/ (Seleccione la opción Microsoft Windows x64). Para poder instalar Oracle Database 23ai asegúrese de contar con permisos de Administrador.
Luego sobre el paquete descargado descomprímalo en un directorio y ejecute el archivo setup.exe con permisos de administrador (Haga clic derecho en setup.exe y seleccione la opción "Ejecutar como administrador" para iniciar el proceso de instalación). Al ejecutar el archivo y esperar unos segundos aparecerá la ventana con el siguiente mensaje:



Por ahora simplemente elija el botón Siguiente para comenzar la instalación en el equipo. En la siguiente ventana acepte el acuerdo de licencia del producto y seleccione el botón siguiente.

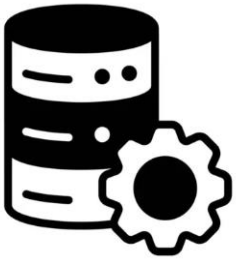


Al instante se abrirá una nueva ventana indicando la ruta de instalación del software. Haga nuevamente click en Siguiente.



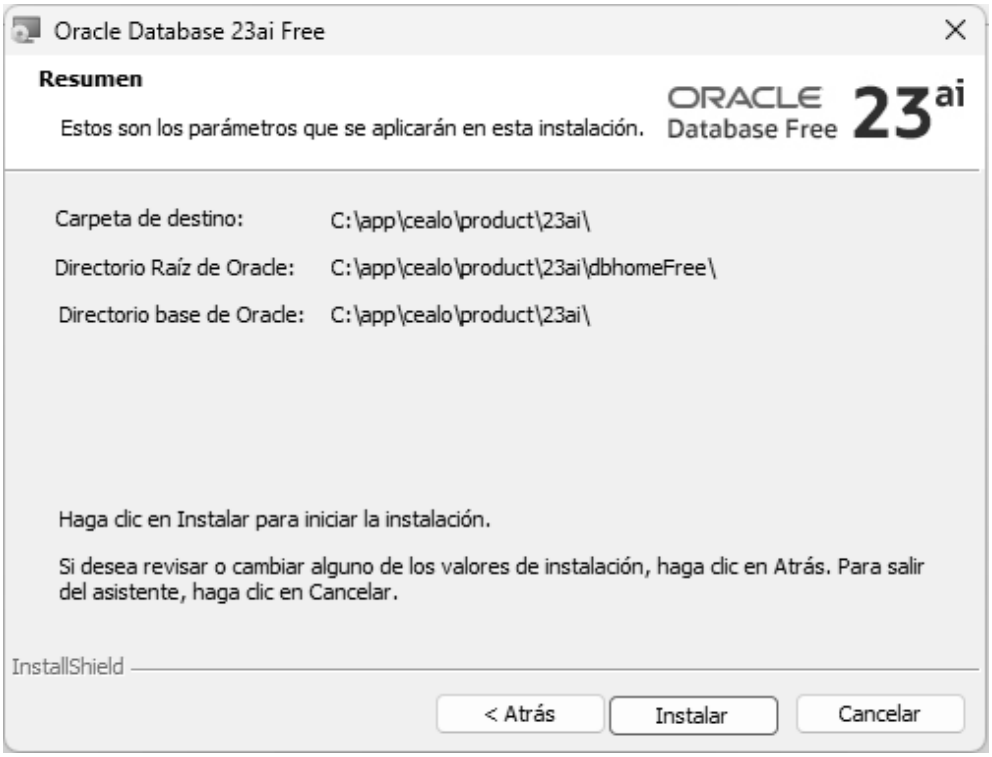
En la nueva ventana deberá colocar una contraseña (deberá repetirla nuevamente) para poder luego realizar procesos de administración sobre el Motor de Base de datos.

Una vez introducida la contraseña mostrara los directorios de instalación de elementos de nuestros paquetes.

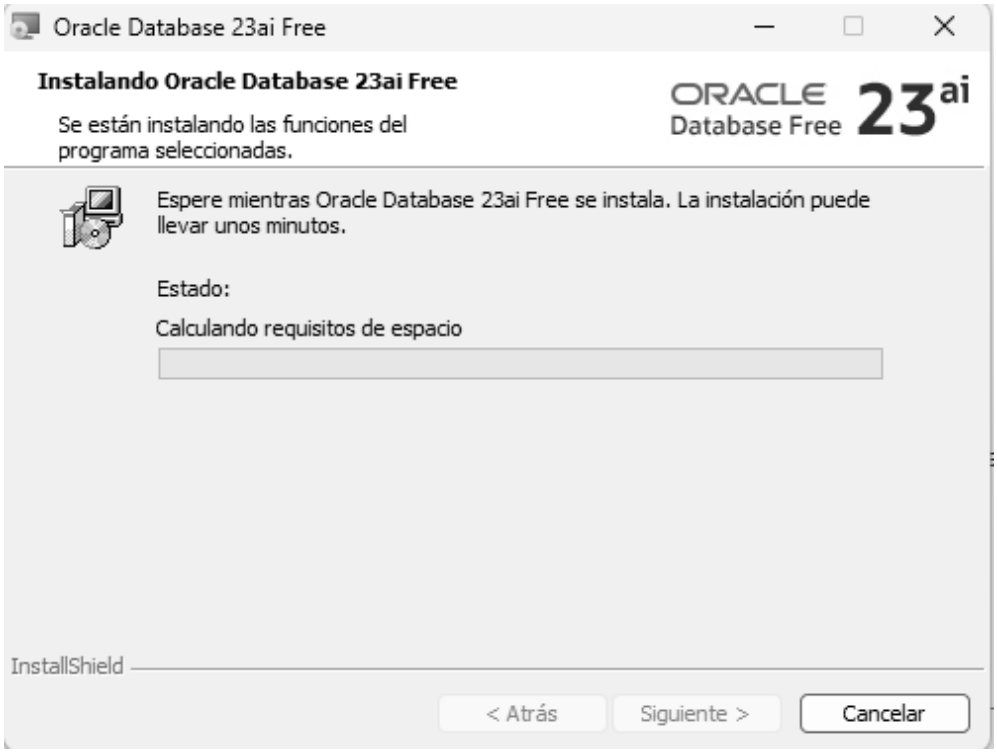


Nota

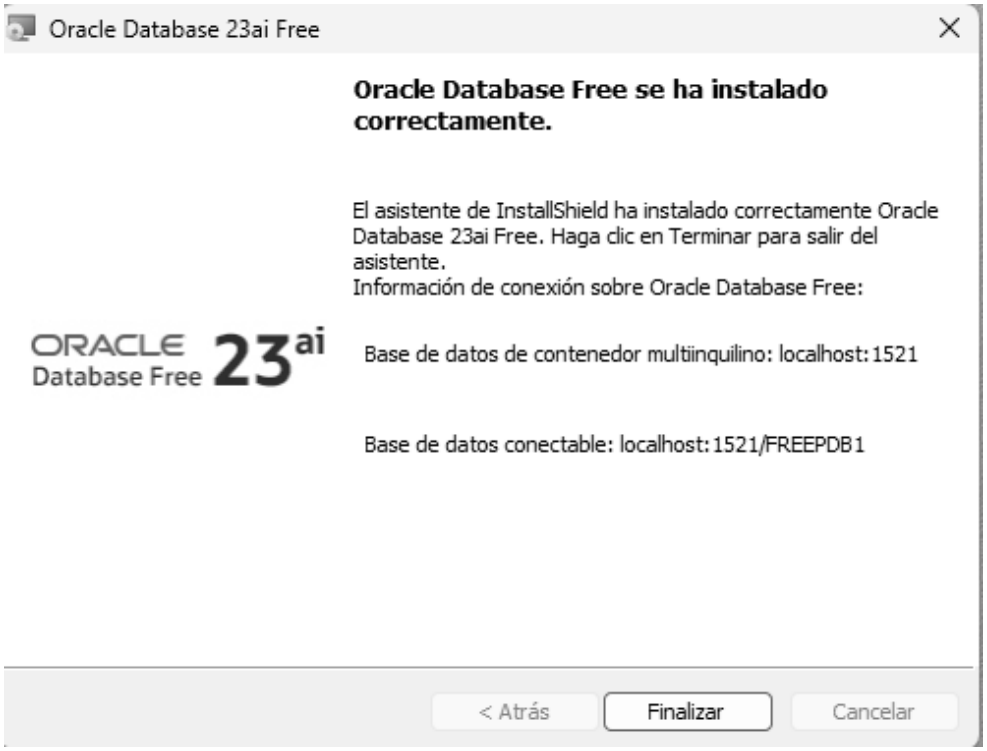
No olvide la contraseña ya que la necesitara más adelante.



Deje todo como esta y presione Instalar. Ahora empezara la copia de archivos este proceso tardara bastante tiempo por lo tanto deberá tener paciencia.



Finalmente llegara a la última ventana en la misma presionara Terminar para finalizar la instalación.



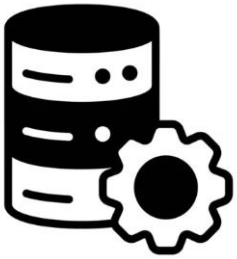
El próximo paso es verificar que todo funcione perfectamente para ello vamos a abrir una ventana de línea de comandos. En el menú de Windows tipee cmd y aparecerá una opción que indica que es el símbolo de sistema. Ejecútelo y desde dicha ventana tipee el siguiente comando:

```
sqlplus
```

sqlplus es una utilidad que viene con el motor Oracle. Con este comando nos conectamos al servidor como usuario administrador (system) dicho usuario se crea automáticamente al instalar el producto. Cuando le solicite la contraseña coloque la contraseña que coloco en la instalación. Al momento tendrá la siguiente línea de comando:

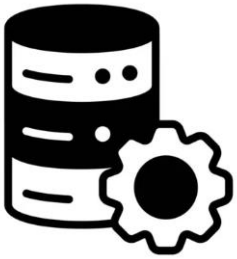
```
SQL*Plus: Release 23.0.0.0.0 - Production on Lun Feb 24 22:04:38 2025
Version 23.6.0.24.10

Copyright (c) 1982, 2024, Oracle.  All rights reserved.

Introduzca el nombre de usuario:
```

Desde esta línea de comandos puede lanzar cualquier consulta SQL que veremos en el curso.



Nota

Puede probar con otro usuario admin como /as sysdba. Para este usuario no es necesario la contraseña.



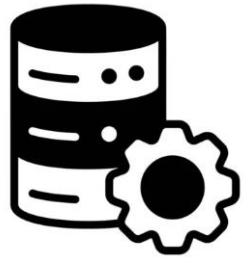
Nota

Si no desea instalar Oracle Database 23ai puede usar la herramienta en línea gratuita: https://livesql.oracle.com/next/.

EL LISTENER DE ORACLE

La conexión típica a Oracle comienza con una petición de acceso desde el lado del cliente. Cuando un cliente intenta conectarse a una base de datos de Oracle, en lugar de conectarse directamente a la base de datos, hay un servicio de intermediario que interviene y maneja la solicitud de conexión para el cliente. Esta proceso se lo conoce como listener y realiza la tarea de escuchar las solicitudes de los clientes entrantes. Cuando un cliente desea establecer una conexión con la base de datos, envía una solicitud al listener a través del protocolo de red TCP/IP. El listener verifica si la base de datos está disponible y si el cliente tiene los permisos adecuados para acceder a ella. Si se cumplen las condiciones, el listener establece la conexión entre el cliente y la base de datos, permitiendo así el intercambio de datos y consultas. El listener está configurado para escuchar la conexión en un puerto específico en el servidor de base de datos. Por defecto ese puerto es 1521. Cuando una pide una conexión a la base de datos, el listener devuelve la información relativa a la conexión. La información de una conexión para una instancia de una base de datos debe tener el nombre de usuario, la contraseña y el SID de la base de datos. Si estos datos no son correctos se devolverá un mensaje de error. El comando lsnrctl permite verificar el estado del listener. Para ello ejecute:

```
lsnrctl status
```



Nota

Puede detener el listener usando el comando lsnrctl stop, para levantar nuevamente el listener puede usar el comando lsnrctl start

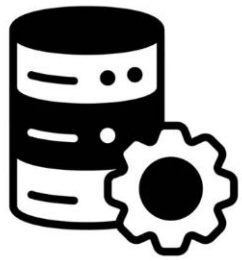
El archivo listener.ora contiene parámetros de configuración de red del lado del servidor. La ubicación del archivo listener.ora en Windows es la siguiente (C:\app\<username>\\product\23ai\dbhomeFree\NETWORK\ADMIN\listener.ora): Ejemplo de configuración del archivo listener.ora:

```
DEFAULT_SERVICE_LISTENER = FREE

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = name_host)(PORT = 1521))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )
```

Observe la referencia al nombre de host name_host. Si esto es incorrecto, el listener no funcionará correctamente. El archivo tnsnames.ora (ubicado en C:\app\<username>\\product\23ai\dbhomeFree\NETWORK\ADMIN\tnsnames.ora), contiene parámetros de configuración de red del lado del cliente. Este archivo en nuestro caso lo encontrara en el mismo directorio mencionado anteriormente.

```
FREE =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = name_host)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = FREE)
    )
  )

LISTENER_FREE =
  (ADDRESS = (PROTOCOL = TCP)(HOST = name_host)(PORT = 1521))
```

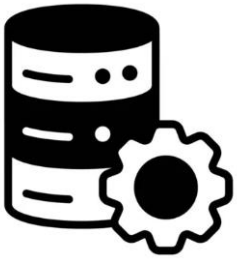
Observe la referencia al nombre de host name_host. Si esto es incorrecto, el listener no funcionará correctamente.



Nota

El Listener es uno de los elementos fundamentales de Oracle Database. Su labor es gestionar el tráfico de las peticiones del cliente. Una vez que el Listener detecta la nueva petición, se establece la conexión y comenzará a comunicarse el proceso de usuario con su proceso servidor correspondiente.

Para poder ejecutar nuestros comandos SQL usaremos una aplicación de Oracle gratuita denominada SQL Developer. La misma se descargará de la página https://www.oracle.com/database/sqldeveloper/technologies/download/ esta herramienta ofrece una Interfaz Gráfica (GUI) para poder trabajar con nuestras bases de datos. Una vez descargada la herramienta debe descomprimirla y buscar un archivo sqldeveloper.exe este archivo ejecutara la aplicación con un simple doble click.

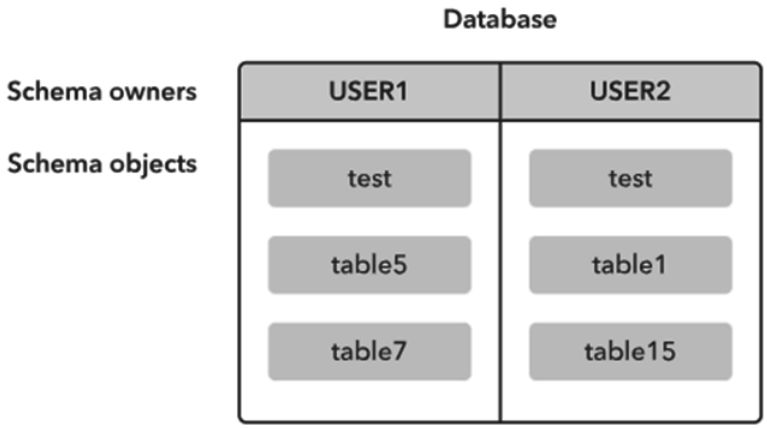


Nota

Es conveniente crear una shortcut o atajo rápido de la aplicación sobre el escritorio para acceder más rápidamente a la misma en los próximos usos.

ARQUITECTURA ORACLE

Un servidor de Base de Datos es la máquina física o virtual que ejecuta el software Oracle Database. Un servidor Oracle tiene dos elementos distintos, la instancia y la base de datos. La base de datos se compone de un conjunto de archivos físicos, que fundamentalmente contienen los datos. La instancia se compone de una estructura de memoria compartida y un conjunto de procesos que son necesarios para acceder a la información contenida en la base de datos. Estos dos elementos están íntimamente relacionados, pero se deben diferenciar correctamente. Los usuarios no pueden acceder directamente a la base de datos, sino que lo hacen a través de una instancia. Una instancia sólo podrá abrir y utilizar una base de datos a la vez, aunque una base de datos podría ser utilizada por varias instancias. Un servidor Oracle puede alojar una o más instancias de base de datos. Una base de datos tiene estructuras físicas y estructuras lógicas. Gracias a esta separación podemos administrar el almacenamiento físico sin alterar las estructuras lógicas de la base de datos. Una Base de Datos Oracle está almacenada físicamente en archivos, y la correspondencia entre los archivos y las tablas es posible gracias a las estructuras internas de la BD. Está división lógica se hace gracias a los espacios de tablas (tablespaces). Se considera lógico porque un tablespace no es visible en el sistema de archivos del sistema operativo. Los archivos que componen una base de datos de Oracle se pueden clasificar en tres tipos: el archivo de datos (data file), archivo de rehacer (redo log file) y el archivo de control (control file). El archivo de datos es donde residen todos los datos, puede haber cualquier número de archivos de datos en una base de datos de Oracle. Los archivos Rehacer registran cada cambio realizado a los datos y se utiliza para la recuperación del sistema. Los archivos de control son un tipo especial de archivo que contiene pequeñas piezas de información vital acerca de la base de datos. Oracle actualiza automáticamente el archivo de control durante cada modificación de la estructura de la base de datos (Por ejemplo, en el agregado o movimiento de un archivo). Si no se puede encontrar el archivo de control (o éste se encuentra dañado), la base de datos no se puede abrir. Cada instalación de Oracle tiene una sola base de datos y, dentro de esa base de datos, puede haber varios esquemas. Cada esquema está vinculado a un usuario y los esquemas pueden tener varios objetos como tablas, índices y vistas. Entonces, para crear un conjunto de tablas y objetos relacionados, debe crear un esquema en lugar de una base de datos.



TABLESPACES

Oracle almacena datos físicamente en los llamados archivos de datos (datafiles). Estos son archivos con extensión .DBF. Oracle crea dichos archivos y guarda en ellos las tablas de la base de datos, las vistas, los índices y otros objetos del esquema.
Los archivos de datos se unen y organizan en unidades lógicas de almacenamiento denominadas espacios de tablas (tablespace). Es decir, un tablespace es una unidad lógica de almacenamiento, y está compuesto por uno o varios archivos físicos (datafiles).
Puntos importantes a tener en cuenta sobre los espacios de tabla de Oracle:

- Un tablespace pertenece a una base de datos solamente. Un archivo de datos pertenece solo al tablespace específico. No puede compartir ni mover archivos de datos entre espacios de tablas, y es imposible mover/compartir espacios de tablas entre bases de datos.
- Un tablespace debe constar de al menos un archivo de datos. El número máximo de archivos de datos en un tablespace es 1022 (el número real de archivos de datos que puede contener un tablespace dependerá del sistema operativo).

La mayoría de las operaciones de administración relativas al almacenamiento se realiza a nivel del tablespace y no a nivel de los archivos de datos. Cuando se crea una base de datos se crea un tablespace por defecto denominado SYSTEM en el que se van a crear los usuarios SYS y SYSTEM automáticamente. Estos usuarios son los que tienen la información necesaria para que funcione nuestra base de datos y podamos hacer todo tipo de operaciones como, por ejemplo, crear nuevos usuarios o crear nuevos tablespaces y tablas en esos nuevos tablespaces.

EL DICCIONARIO DE DATOS

El diccionario de datos es un conjunto de tablas y vistas que dan información sobre el contenido de una base de datos. El diccionario de datos se almacena en el tablespace SYSTEM. Se crea durante la creación de la base de datos y Oracle lo actualiza automáticamente, cuando se ejecutan sentencias SQL DDL como CREATE, ALTER, DROP. Para utilizarlo, es suficiente con realizar consultas usando sentencias SELECT. Salvo excepciones, toda la información se almacena en mayúsculas en el diccionario de datos; téngalo en cuenta cuando escriba sus cláusulas WHERE. Por ejemplo, si escribimos este comando en SQL Plus podremos saber si la base de datos está abierta o no.

```
select status from v$instance;
```

El resultado sería en este caso:

```
STATUS
------------
OPEN
```

Y si por ejemplo escribimos esta instrucción SQL:

```
select version from v$instance;
```

Nos indica la versión del motor de nuestra base de datos. También podríamos consultar el nombre de la base de datos:

```
select name from v$database;
```

O la fecha de creación de la base de datos:

```
select created from v$database;
```

Tanto v$instance y v$version en Oracle se la conocen como vistas dinámicas.

TERMINOLOGIA DE SQL

Para comenzar a escribir en lenguaje SQL es importante conocer algunos términos, en principio veremos que son las constantes o literales, un literal es un valor fijo. Diferenciamos números (numbers) de constantes alfanuméricas (cadenas de caracteres o strings) según como se escriban. Las constantes de cadena de caracteres van entre comillas simples e incluyen caracteres alfanuméricos (a-z, A-Z y 0-9) y caracteres especiales, como el signo de exclamación (!), la arroba (@) y el signo de número (#). Otro tipo de constantes son las fechas o intervalos de tiempo que los veremos más adelante. En SQL como en la mayoría de los lenguajes existen los operadores. Los podemos dividir entre operadores aritméticos, operadores de comparación y operadores lógicos. Una expresión SQL es una cadena que contiene constantes, operadores o funciones (Por ejemplo 9+5 es una expresión que contiene dos constantes y un operador aritmético). Las funciones son una de las razones por la que se usa SQL. Puede reconocer las funciones de SQL por su firma: tienen un nombre único, seguido de unos paréntesis donde hay uno o más argumentos separados por comas. Por ejemplo la función avg() permite obtener el promedio de valores de una columna SQL. Como cualquier otro lenguaje, SQL tiene una lista de palabras reservadas. Estas son palabras solo se usan para la función que fueron implementadas.

Por ejemplo, no podría usar la palabra select para por ejemplo usarla como nombre de campo. Finalmente pasemos a ver lo que es una clausula. Como una frase, una instrucción SQL tiene cláusulas. Cada cláusula realiza una función de la instrucción SQL. Por ejemplo, la cláusula from permite buscar en las tablas que se definan dentro de la instrucción select.

OPERADORES DE SQL

OPERADORES ARITMETICOS

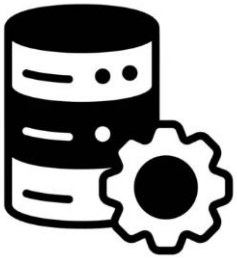
Los operadores aritméticos ejecutan operaciones matemáticas:

Operador	Significado
+ (Añadir)	Suma
- (Restar)	Sustracción
* (Multiplicar)	Multiplicación
/ (Dividir)	División

OPERADORES DE COMPARACION

El resultado de un operador de comparación es del tipo de datos Boolean. Tiene dos valores posibles verdadero o falso. Las expresiones que devuelven tipos de datos Boolean se conocen como expresiones booleanas.

Operador	Significado
= (Igual a)	Igual a
> (Mayor que)	Más que
< (menor que)	Menos que
>= (Mayor o igual a)	Mayor o igual a
<= (Menor o igual a)	Menor o igual a
<> (No es igual a)	No es igual a
!= (No es igual a)	No es igual a (no es estándar ISO)



Nota

SQL ofrece un solo operador alfanumérico que permite concatenar dos cadenas de caracteres. Ese operador se identifica con los símbolos || (doublé pipe)

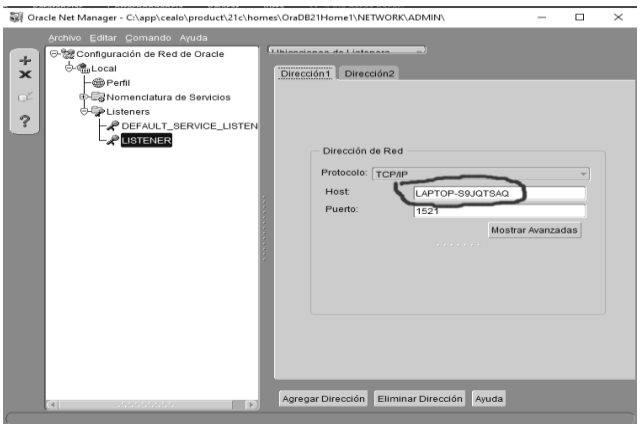
OPERADORES LOGICOS

Los operadores lógicos comprueban la veracidad de alguna condición. Al igual que los operadores de comparación, devuelven el tipo de datos Boolean con el valor verdadero o falso.

Operador	Significado
AND	VERDADERO si ambas expresiones booleanas son VERDADERAS.
OR	VERDADERO si alguna de las expresiones booleanas es VERDADERA.
NOT	Invierte el valor de cualquier otro operador booleano.

HERRAMIENTA NET MANAGER EN ORACLE

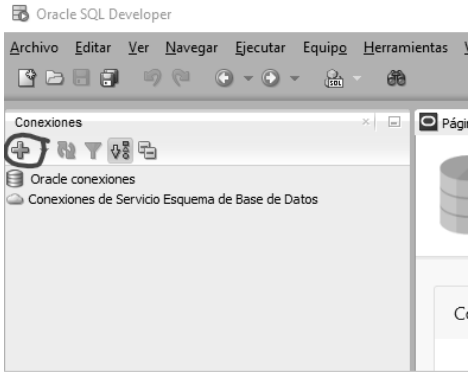
A través de a herramienta grafica Oracle Net Manager podemos tener acceso a la información del listener. La herramienta se puede ejecutar desde el Menú Inicio de Windows.



Si nos fijamos, vemos que a la izquierda tenemos un panel con una especie de árbol de directorios. En el primer nivel de carpetas tenemos el directorio ¨Local¨, que de cierta forma es el entorno donde nos encontramos. Vamos a proceder a desplegarlo y seleccionar la última opción de la carpeta Listeners. Como su nombre lo indica, aquí tendremos las configuraciones relacionadas con el listener actual. En el panel derecho sobre la pestaña Direccion1 podemos tener el nombre de nuestro host (servidor) y el puerto por defecto. Con esta información estaremos preparados para conectarnos a una base de datos.

CONECTAR A UNA BASE DE DATOS

Una vez ejecutada la aplicación deberá proceder a crear una nueva conexión a una base de datos. Para ello desde la página de bienvenida deberá realizar un click sobre el botón denominado Crear una conexión manualmente. Otra forma seria clickear en la pestaña (+) ubicado en la esquina superior izquierda de la ventana conexiones.



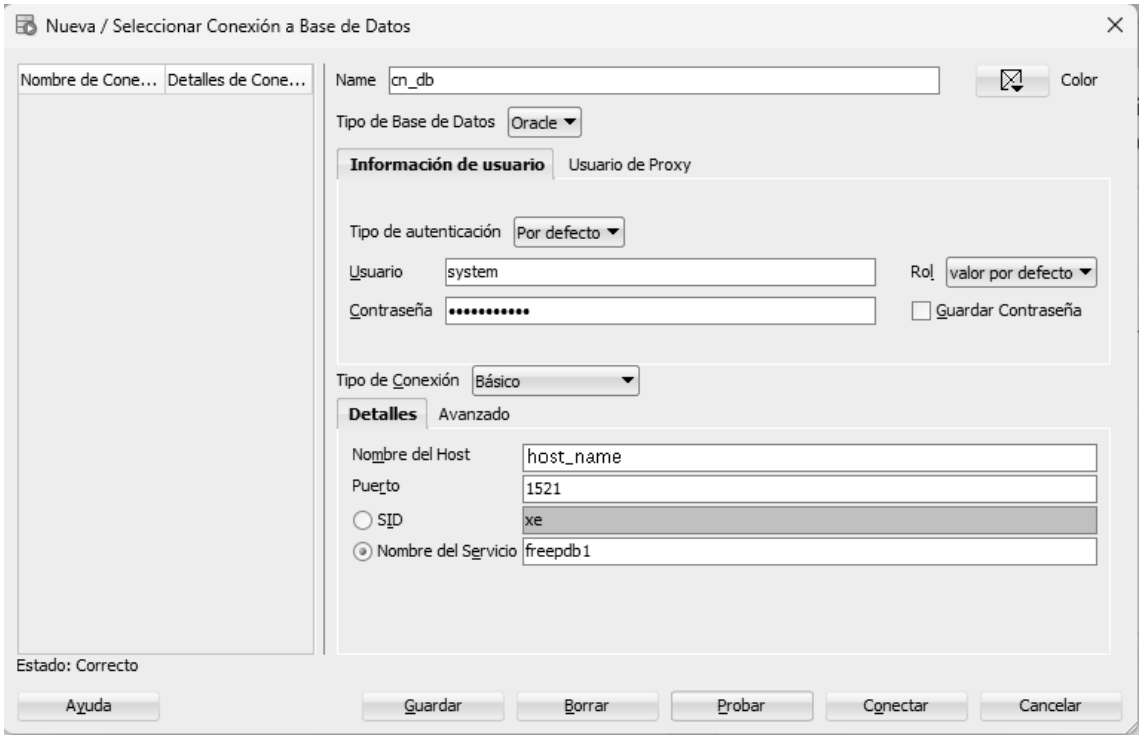
En la nueva ventana colocaremos un nombre a la nueva conexión en el cuadro Name (Por ejemplo el nombre cn_db) , el usuario con que nos conectaremos (system en nuestro caso) y la contraseña que colocamos en la instalación del producto. Cuando nos pida el nombre del host deberemos colocar una IP o el nombre del servidor que contiene la base de datos (en nuestro caso localhost). Luego dejaremos el puerto tal cual indica en el cuadro (1521) y el SID será el nombre de nuestra base de datos en este caso dejaremos el nombre propuesto que es free. Si no conoce el nombre de su base de datos entre por la línea de comando a sqlplus como lo hizo anteriormente y tipee el siguiente comando:

```
select name from v$database
```

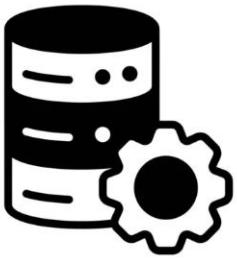
Realizado todo esto procederemos a clickear sobre el botón Guardar (este se encargará de guardar nuestra conexión para los próximos usos) y a clickear sobre conectar para comenzar a trabajar.

BASES DE DATOS CDB y PDB EN ORACLE

En los últimos años, muchas empresas empezaron a consolidar gran número de bases de datos en servidores con infraestructuras bastante robustas. Cada instancia ocupa gran consumo de recursos. Oracle 12c introduce una nueva arquitectura llamada Oracle MULTITENANT en la que provee una base de datos CDB que es un gran contenedor donde pueden ser incluidas desde cero a mas bases de datos llamadas Pluggable Databases (PDB). Todos estos PDB comparten los mismos metadatos del CDB. Si un cliente quiere consolidar muchas bases de datos en un solo servidor puede hacer uso de esta arquitectura y tener una sola instancia con muchas bases de datos como Pluggable database. La idea es que esto permitiría desconectar fácilmente una PDB de un entorno y conectarla a otro (en la nube, por ejemplo). Esto también tiene que ver también con los usuarios que creamos. Un usuario local es especifico de cada PDB. Un usuario común se almacena en el CDB y se puede conectar en cada PDB. En el caso de Oracle 23ai ya existe una PDB creada llamada freepdb1.

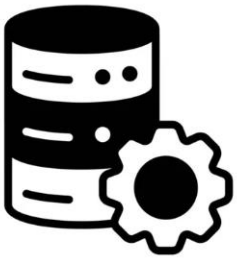


En la ventana anterior podemos ver como se selecciona la opción nombre del servicio en vez de SID (SID se usa si usamos base de datos contenedor) el nombre del servicio en este caso es freepdb1.



Nota

En Oracle, la cuenta system es una de las cuentas administrativas predefinidas que se generan automáticamente cuando se instala Oracle. Dicha cuenta es capaz de realizar la mayoría de las tareas administrativas.



Nota

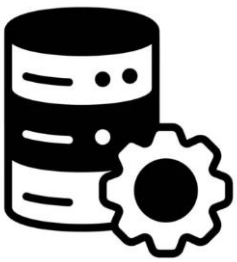
En Oracle, el identificador del sistema (o SID) es un identificador local de hasta ocho caracteres de longitud que se utiliza para identificar una base de datos en particular y diferenciarla de otras bases de datos del sistema.

USUARIOS

Una cuenta de usuario está relacionada con los objetos de la Base de Datos: los usuarios poseen los objetos de la Base de datos. Existen dos usuarios especiales: SYS y SYSTEM. El usuario SYS posee las tablas del diccionario de datos; que almacenan información sobre el resto de las estructuras de la Base de Datos.
Este usuario sys no se suele utilizar, ya que, si alguien viera y copiara la contraseña, tendría acceso total a la base de datos, pero si se usa para arrancar o parar la base de datos y hacer actualizaciones del Oracle. Luego tenemos el usuario system, este también tiene muchos derechos, pero no tantos como el sys. Es mucho más utilizado ya que es el usuario adecuado para hacer las operaciones más típicas en el mantenimiento de una base de datos. No hay que olvidar que estos dos usuarios se crean por defecto a la hora de crear la base de datos, pero en la vida real lo que se hace es crear un usuario que tenga todas los permisos y privilegios necesarios para gestionar la base de datos, y se entra con ese usuario. Todo objeto creado en la Base de Datos se crea por un usuario, en un espacio de tablas y en un archivo de datos determinado. Cada usuario puede acceder a los objetos que posea o a aquellos sobre los que tenga derecho de acceso. El conjunto de objetos de un usuario es conocido como esquema. Para crear un usuario primero debe conectarse con la conexión creada anteriormente y luego debemos escribir los siguientes comandos:

```
create user hr_admin identified by pwd123456hr;
grant create session to hr_admin with admin option;
grant connect, resource,dba to hr_admin;
grant unlimited tablespace to hr_admin;
```

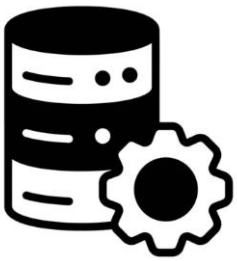
La primera línea crea el usuario hr_admin con el password indicado más arriba. Sin embargo, al crear un nuevo usuario no hará que dicho usuario acceda inmediatamente a la base de datos.
Hay roles y privilegios necesarios que deben asignarse al usuario. Con la cuenta creada, ahora podemos comenzar a agregar roles y permisos (privilegios) a la cuenta usando el comando GRANT. GRANT es un comando muy poderoso con muchas opciones. Recordemos que un usuario recién creado no puede hacer nada, ni siquiera conectarse a la base de datos. Por eso es necesario asignarle roles. Los roles permiten una mejor y más sencilla gestión de los privilegios.
Primero, debemos otorgar a nuestros usuarios el privilegio para iniciar sesión en la base de datos. Para ello usamos GRANT CREATE SESSION TO hr_admin WITH ADMIN OPTION.



Nota

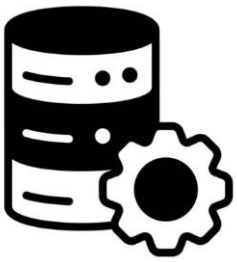
Un privilegio indica que acciones puede realizar un usuario. Un rol es una agrupación de permisos. Gracias a los roles, podemos juntar ciertos permisos en un solo rol y concederlo a un usuario o a un grupo de usuarios,

Si se le asignan los roles Connect y Resource ya tiene los permisos mínimos, podrá conectarse a la base de datos y realizar las operaciones más habituales de consulta, modificación y creación de objetos en su propio esquema. El rol DBA permite la administración total de la base de datos. La cláusula UNLIMITED permite que el usuario tenga un espacio ilimitado para crear objetos.



Nota

El comando show user puede mostrar en cualquier momento cual es el usuario actual.



Nota

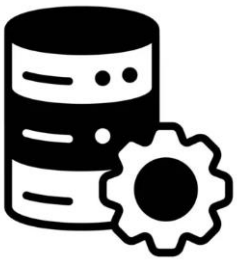
En el caso de que necesite eliminar a algún usuario por cualquier motivo, debe usar el comando DROP USER seguido del nombre del usuario.

Podemos adentrarnos un poco más con los comandos de SQL, por ejemplo, podríamos cambiar la contraseña con el siguiente comando:

```
alter user hr_admin identified by pwdOracle123456
```

ESQUEMAS DE UNA BASE DE DATOS

Un esquema es una colección lógica de todos los objetos de una base de datos que generalmente es propiedad del usuario del mismo nombre. Cuando se crea un usuario se creará el esquema y contendrá todos los objetos, como tablas, índices, vistas, procedimientos y funciones PL/SQL, y cualquier otra cosa que sea propiedad del usuario. La importancia de los esquemas de base de datos es que ayudan a los administradores y desarrolladores a comprender cómo se estructura una base de datos, y esto permite construir y administrar una base de datos de manera mucho más eficaz. Cada esquema de base de datos se crea para un propósito específico, como resolver una parte de un sistema de información empresarial (Por ejemplo: RRHH).



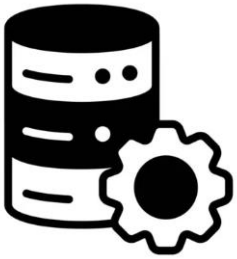
Nota

Un esquema es básicamente una descripción formal de cómo se forma una base de datos y dónde se encuentra todo A diferencia de SQL Server y PostgreSQL, no hay un objeto de esquema separado. Sin embargo, si un usuario se convierte en propietario de objetos como tablas, vistas, etc., se trata como un esquema.

Ahora crearemos una nueva conexión para el usuario recién creado. Para ello realizamos clickeamos en la pestaña (+) ubicado en la esquina superior izquierda de la ventana conexiones. En la nueva ventana colocaremos un nombre a la nueva conexión en el cuadro Name (Por ejemplo el nombre hr) , el usuario con que nos conectaremos (hr_admin en nuestro caso) y la contraseña recién modificada (pwdOracle123456). En nombre de host dejaremos nuestra ip o la palabra localhost. El puerto será el de por defecto: 1521. En nombre de servicio colocaremos freepdb1. Ahora testeemos la nueva conexión y si nos informa correcto la guardaremos y nos conectaremos. Al momento nos pedirá la contraseña del usuario recién creado (en este caso, pwdOracle123456). El paso siguiente será mostrar las tablas con que el usuario propietario puede trabajar. Para ello sobre la conexión con el botón derecho del Mouse seleccione la opción Explorador de Esquemas si selecciona en el menú superior los objetos tablas vera que aún no tiene ninguna tabla por lo tanto no tenemos datos para poder trabajar.

EJECUTAR SENTENCIAS SQL

Desde la ventana Hoja de Trabajo SQL se puede escribir y ejecutar sentencias de SQL y PL/SQL. Las sentencias se escriben como si se estuviera trabajando en un editor de texto. Si son varias cada sentencia deberá terminar con punto y coma. Tras introducir las sentencias, pueden ejecutarse mediante el icono de Sentencia de ejecución (ejecuta la sentencia sobre la que se encuentra el cursor) o mediante el icono Ejecutar Script que ejecuta todas las sentencias que existan. En la parte inferior se muestra la ventana Resultado de la consulta con los resultados de la última sentencia ejecutada. Después de escribir código se puede formatear utilizando la opción de formato (ctrl + F7) o haciendo clic con el botón derecho en la hoja de trabajo de SQL y seleccionando Formato. Esto permitirá definir un estándar común en todo el equipo de trabajo manteniendo el mismo formato de código y facilitaría el mantenimiento del mismo. La tecla F8 permite mostrar una ventana con el historial de los comandos recientes, si se realice un doble click sobre uno de los comandos aparecerá nuevamente sobre la Hoja de Trabajo SQL.



Nota

Si hay errores de sintaxis en los comandos el motor de base de datos arrojara el error ORA-00933: SQL en la ventana de resultados.

Si necesita guardar en discos los comandos arrojados deberá pulsar CTRL+S. Al momento aparecerá una nueva ventana para almacenar el archivo con un nombre y extensión SQL.

TABLAS EN ORACLE

Como mencionamos anteriormente una tabla en un SGBD Relacional se encarga de almacenar la información. Podemos listar las tablas que tenemos en nuestra base de datos tipeando el siguiente comando:

```
SELECT table_name FROM user_tables;
```

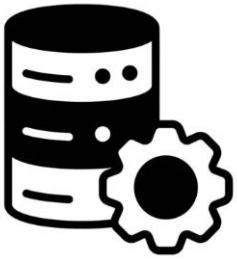
Este comando SQL devolverá una lista de todas las tablas en las que el usuario actual es propietario. Si está interesado en mostrar las tablas donde el usuario actual puede tener acceso debe tipear:

```
SELECT table_name FROM all_tables;
```

En la versión 23ai, podemos tipear el siguiente comando para realizar una prueba de un comando SQL:

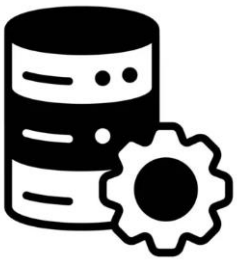
```
SELECT sysdate;
```

Este comando devolverá la fecha actual. Sin embargo, no hicimos uso de la cláusula FROM. En este caso permite excluir una tabla que era muy usada en versiones anteriores de Oracle Database, la tabla DUAL. Luego de escribir el comando en el editor deberá presionar el botón de Ejecutar (simbolizado con una flecha verde por encima del Editor), este botón ejecutará la consulta. Si la consulta fue bien interpretada vera en la ventana Resultado de la consulta la fecha de hoy.



Nota

Puede ejecutar un comando SQL con el atajo CTRL+ENTER



Nota

Puede comentar líneas individuales usando el doble guion (–). Para comentar varias líneas, colocamos / antes del comentario y terminamos con */.*

EJECUTAR UN SCRIPT SQL EN SQL DEVELOPER

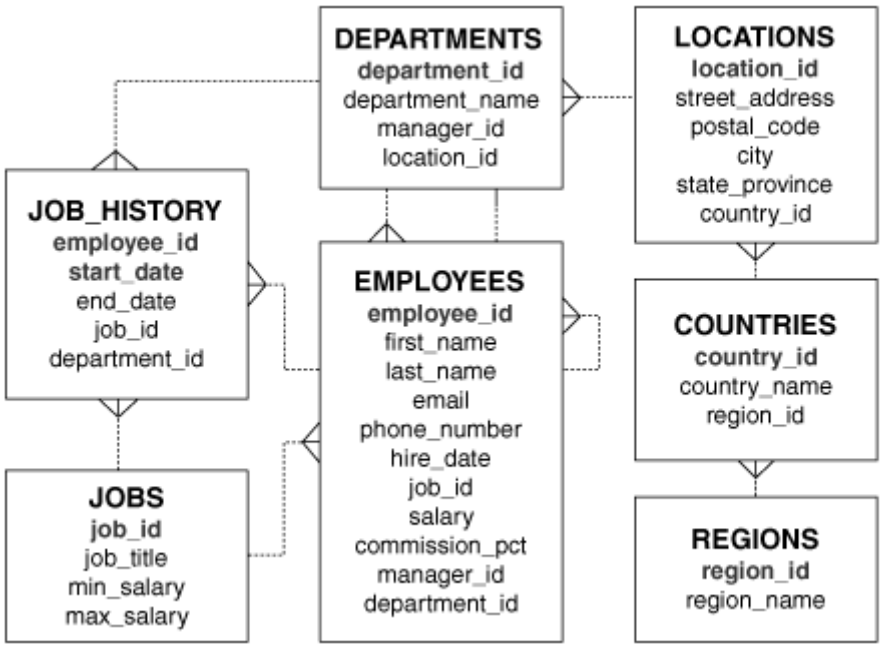
Vamos a suponer que tenemos una serie de comandos SQL almacenados en un script con extensión sql y queremos ejecutarlos en nuestra ventana de SQL Developer sin salir de la misma. Una forma sencilla es anteponer el símbolo @ y luego colocar la ruta completa para llegar al script:

```
@c:\oracle\script.sql
```

En este ejemplo hemos creado la carpeta oracle y dentro de ella un archivo con extensión sql. Finalmente, para ejecutar el script deberá presionar el botón de Sentencia de Ejecucion.

CASO PRACTICO: BASES DE DATOS DE RRHH

Para este ejemplo trabajaremos con una base de datos de Recursos Humanos denominada HR, la misma puede descargarse en forma gratuita del repositorio (https://github.com/oracle-samples/db-sample-schemas/releases/tag/v23.3). El siguiente diagrama de base de datos ilustra la base de datos de ejemplo de HR. El diagrama lógico muestra siete entidades con sus relaciones.



La base de datos de muestra de recursos humanos tiene siete tablas:

La tabla **employees** almacena los datos de los empleados.

La tabla **jobs** almacena los distintos puestos incluido los salarios.

La tabla **departments** almacena los departamentos de la empresa

La tabla **jobs_history** almacena el historial de anteriores trabajos de cada empleado.

La tabla **locations** almacena la ubicación de los departamentos de la empresa.

La tabla **countries** almacena los datos de los países en los que la empresa hace negocios.

La tabla **regions** almacena los datos de regiones como Asia, Europa, América y Oriente Medio y África. Los países se agrupan en regiones.

La tabla Empleado muestra una relación recursiva, es decir una relación de entidad consigo misma, que implementa la jerarquía dentro de la empresa. Para poder trabajar con todo este esquema es necesario crear una carpeta que contenga dicho esquema (nosotros creamos una carpeta oracle y dentro de ella copiaremos la carpeta human_resources del repositorio recién descargado). Ahora ejecutaremos los scripts hr_create.sql y hr_populate.sql de la siguiente forma:

```
@c:\oracle\human_resources\hr_create.sql
@c:\oracle\human_resources\ hr_populate.sql
```

Estos dos scripts construyen el modelo y los datos de ejemplos asociado a ese modelo. Luego en el explorador de esquemas podrá ver el listado de tablas importados. Si realiza doble click en una tabla podrá ver su estructura.

DICCCIONARIO DE DATOS

El diccionario de datos es un conjunto de tablas y vistas que dan información sobre el contenido de una base de datos. Gracias al diccionario de datos podemos saber que tablas están presentes dentro de una base de datos, que columnas almacenan, cuales son los índices, que usuarios existen y que permisos tienen. Este diccionario se crea durante la creación de la base de datos y Oracle lo actualiza automáticamente, cuando se ejecutan sentencias DDL. Oracle mantiene automáticamente el diccionario de datos; por lo tanto, el diccionario de datos siempre está actualizado. La gran ventaja de esto es que podemos usar SQL para consultar información del diccionario de datos. Existe un comando muy usado que es describe, por ejemplo:

```
describe regions;
```

Lo que realiza este comando es mostrar información sobre la tabla regions. Por ejemplo, mostrara los nombres de las columnas y el tipo de dato asociado a cada columna.

HERRAMIENTA DATA MODELLER

El modelado de la base de datos o ingeniería inversa de la base de datos es una de las herramientas que permite documentar en un diagrama el diseño de una base de datos. Data Modeller es una herramienta creada por Oracle gratuita y que está incluida en SQL Developer. Para realizar el diagrama de diseño a partir de las tablas existentes de nuestra base de datos debe primero seleccionar la conexión y luego dirigirse al menú Archivo, luego debe seleccionar la opción Importar de la opción Data Modeler y por ultimo seleccionar la opción diccionario de datos. En la nueva ventana seleccionamos la conexión que queremos diagramar (en nuestro caso hr) y luego presionamos el botón siguiente. Ahora seleccione el esquema que quiere modelar (en nuestro caso hr_admin) y luego con el botón siguiente seleccionamos las tablas que aparecerán en el diagrama con el botón siguiente obtendrá un resumen de la realizado y procederá a realizar un click en el botón Terminar. Al momento tendrá un diagrama como el presentado anteriormente.

CONSULTAS DE SELECCIÓN: INSTRUCCION SELECT

Ahora que tiene datos introducidos en la base es el momento de aprender a realizar consultas para extraer información desde la base de datos. La instrucción select es la instrucción que me permitirá extraer información de una o más tablas. La sintaxis básica de la instrucción es la siguiente:

```
select * from employees;
```

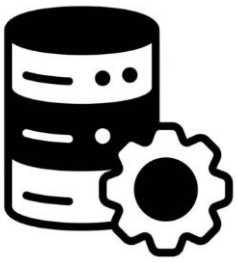
El símbolo * (todos) devuelve todos los campos de la tabla. Las cláusulas son palabras clave en SQL que le permiten solicitar una determinada acción. La cláusula from sirve para especificar la tabla o tablas a consultar.
Esta cláusula es obligatoria y sigue a cualquier comando select. El resultado final de esta consulta es devolver todos los campos con todas las filas de la tabla employees.
Si deseáramos mostrar solo las columnas last_name y email debemos escribir lo siguiente:

```
select last_name,email from employees;
```

La cláusula where permite devolver información basada en una condición. Por ejemplo, en este select solo se devolverán la información del empleado cuyo apellido es Fripp:

```
select * from employees where last_name='Fripp';
```

La cláusula where permite especificar condiciones. Esta pregunta es verdadera o falsa. Llamamos a esta pregunta (condición) un predicado. El tipo de pregunta que hacemos es una expresión. En nuestro ejemplo, en el predicado usamos el operador de comparacion igualdad porque estamos preguntando si el apellido es igual a Fripp.



Nota

Todas las consultas SQL deben terminar en ; (punto y coma)

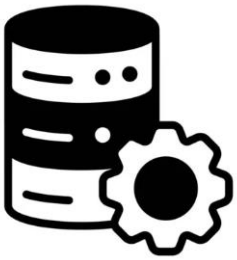
Podemos omitir registros que contienen datos duplicados usando distinct. Por ejemplo,podemos enumerar los distintos jefes que ´posee la empresa:

```
select distinct(manager_id) from employees
```

Si tenemos un mismo jefe para distintos empleados solo se lo visualizara una sola vez.

TIPO DE DATOS BASICOS EN SQL

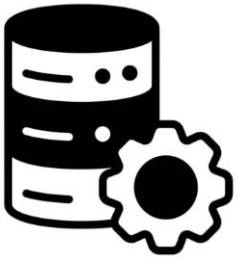
Oracle tiene muchos tipos de datos a medida que vayamos avanzando en el curso los iremos viendo. Las cadenas de caracteres son un tipo de dato básico y sirven para almacenar distintos tipos de caracteres. Esos tipos en Oracle son el char y los varchar2. El CHAR almacena una cadena de longitud fija, esto significa que si se almacenan menos de los que deberían almacenarse ese espacio se desperdicia. El tipo de datos VARCHAR2 se utiliza para almacenar cadenas de longitud variable. Esto significa que, si usted define un tamaño máximo y almacena una cadena, esta se almacenará tal como fue proporcionada. No se agregarán espacios en blanco, como el tipo de datos CHAR.



Nota

En Oracle el tipo VARCHAR y VARCHAR2 son iguales. Pero Oracle recomienda usar el tipo VARCHAR2.

CHAR tiene un tamaño máximo de 2000 bytes y VARCHAR/VARCHAR2 tiene un tamaño máximo de 32.767 en las últimas versiones de Oracle. Cuando se define un literal deben empezar y terminar con una comilla simple.



Nota

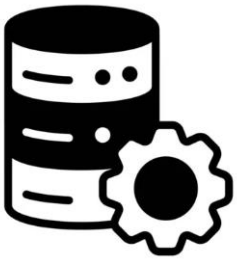
Piense en la última consulta realizada (cuando buscamos los empleados con el apellido Fripp).. La cadena a buscar se colocó entre comillas simples y se respetó las letras en mayúsculas y en minúsculas.

Oracle tiene varios tipos de datos numéricos. Estos tipos de datos son útiles para almacenar valores numéricos, tanto números enteros como valores decimales. El tipo de datos NUMBER se utiliza para almacenar números fijos y de punto flotante. Puede almacenar números positivos y números negativos, y puede almacenar hasta 38 dígitos. Al definir un NUMBER debe especificar la longitud y la cantidad de decimales. Ejemplo: number (8,1) almacena como máximo el número 9.999.999,9.

Ejemplo:

```
select * from employees where salary=9000
```

En este ejemplo seleccionamos los empleados cuyo salario es 9000 dólares.



Nota

La última consulta realizada no necesito colocar el valor a buscar entre comillas dobles ya que es un tipo de dato number.

USO DE ALIAS EN SQL

De forma predeterminada, los nombres de las columnas de la tabla se muestran encima del resultado de la consulta. Si, por ejemplo, cree que esos nombres no describen la información que muestra puede especificar un nombre diferente en las columnas que se muestran en los resultados. Para realizar esto debe usar un alias con la palabra AS.

Ejemplo:

```
select first_name || ',' || last_name AS nombre_completo from employees;
```

El resultado final es una columna con la denominación nombre_completo donde se muestra el nombre y el apellido separado por una coma.

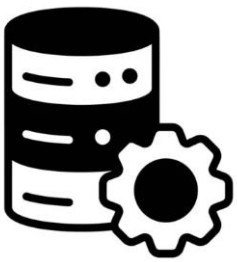
USO DE OPERADORES EN LAS CONSULTAS SELECT

El valor NULL representa a un valor desconocido o ausente. Para recuperar los registros que contengan el valor NULL en algún campo debe usar el operador IS NULL. Por ejemplo, si usamos la siguiente sentencia:

```
select * from employees where manager_id IS NULL;
```

Devolverá todas las columnas cuyo campo manager_id tenga el valor NULL. En cambio, el operador IS NOT NULL devolverá aquellos registros que no poseen el valor nulo en el campo determinado.

```
select * from employees where manager_id IS NOT NULL;
```



Nota

El estándar SQL no define NULL como un valor, sino más bien como un marcador con un valor ausente o desconocido. Por consiguiente, ningún valor puede ser NULL. Por dicho motivo no puede usar el operador = (igual) para comparar con el valor NULL sino que debe usar el operador IS NULL.

El operador like hace coincidir los registros con un patrón especificado. Con frecuencia, este patrón es un carácter comodín, como los signos % o _. El carácter % realiza una coincidencia con cualquier cantidad de caracteres (incluso con cero caracteres), en cambio el comodín _ solo realiza la coincidencia con un solo carácter.
Ejemplo:

```
select * from locations where country_id LIKE 'U_';
```

En este ejemplo la consulta devuelve los códigos de ubicaciones cuya letra comienza con la letra U seguida de cualquier carácter.

```
select * from locations where city like 'S%';
```

En este segundo ejemplo, devolverá las ubicaciones cuya ciudad comience con la letra S seguido de cualquier número de caracteres.
Los operadores AND y OR sirven para unir diferentes condiciones dentro de una cláusula WHERE. Mediante AND se mostrarán sólo aquellos registros que cumplan todas las condiciones, mientras que con OR se mostrarán los registros que cumplan al menos una de ellas. Por ejemplo:

```
select first_name,last_name,manager_id from employees where manager_id=101 or
manager_id=103;
```

En este ejemplo, tenemos todos los empleados cuyo jefe tiene como id el 101 o el 103. Fíjese que si hubiera colocado el operador and el resultado hubiera sido de cero filas porque es imposible que el empleado tenga como jefe inmediato el id 101 y el id 103 al mismo tiempo. Esta consulta se podría haber resuelto más eficientemente usando el operador IN:

```
select last_name,manager_id from employees where manager_id in (101,103);
```

El operador between permite definir un rango de valores numéricos. Si el valor se encuentra en dicho rango el registro aparecerá como resultado:

```
select first_name,last_name,manager_id from employees where manager_id between
100 and 103;
```

El resultado de la consulta es la misma si se hubiera utilizado el operador AND de esta forma:

```
select first_name,last_name,manager_id from employees where manager_id>=100
and manager_id<=103;
```

El operador NOT lógico niega el operando. Por ejemplo:

```
select first_name,last_name,manager_id from employees where not
manager_id=100;
```

En este ejemplo se está buscando los registros de empleados cuyo jefe inmediato no sea el código 100. Otra forma de hacer lo mismo seria de esta manera:

```
select first_name,last_name,manager_id from employees where manager_id<>100;
```

La cláusula ORDER BY permite ordenar los registros resultantes de una consulta por un campo o varios campos especificados en forma Ascendente (ASC) o Descendente (DESC).

```
select last_name,manager_id from employees order by last_name;
```

En este ejemplo ordenamos la salida de los registros ordenándolos por el apellido en forma ascendente. Si especifica varios campos en la ordenación, ordenará por el primero, y en caso de coincidir varios resultados, los ordenará por el segundo. Ejemplo:

```
select last_name,manager_id from employees order by manager_id asc, last_name
asc;
```

En este ejemplo se ordenó la salida primero por el id de su superior inmediato y al haber coincidencia en ese campo se lo reordena por apellido del empleado.

FUNCIONES EN ORACLE

Oracle admite una gran cantidad de funciones. Además de las diversas funciones estándar de SQL, se han agregado muchas funciones específicas de Oracle a la implementación de SQL de Oracle a lo largo de los años. Básicamente, existen dos tipos de funciones: de agregación y de transformación. Las funciones de agregación operan sobre un conjunto de valores, y devuelven un único valor. Por ejemplo, la función SUM (suma) es una función de agregación. Las funciones de transformación operan sobre un único valor, y devuelven también un único valor. Por ejemplo, el valor absoluto es una función de transformación.
Por ejemplo con la función COUNT() podremos saber cuántos empleados tenemos en nuestra empresa:

```
select count(*) from employees;
```

Puede usar un alias (Total de Empleados) que aparecerá como nombre de la columna:

```
select count(*) as "Total de Empleados" from employees;
```

Podemos hacer la última consulta un poco más compleja:

```
select count(*) as "Total de Empleados" from employees where manager_id=101;
```

Para este ejemplo contamos la cantidad de empleados cuyo jefe inmediato tiene el código 101. La función MAX() devuelve el valor máximo de la columna especificada. Por ejemplo, la próxima consulta devuelve el sueldo más alto de un empleado.

```
select max(salary) from employees;
```

La función de ORACLE NVL() nos permite obtener un valor concreto en vez de NULL como resultado.

```
select last_name,nvl(manager_id,-1) from employees;
```

En el ejemplo anterior la función evalua la columna manager_id, en el caso de que el valor devuelto sea NULL pondrá el valor -1. En el caso de que sea distinto a NULL devolverá el valor que ocupa en dicha columna. Sin embargo, para ser más explícito podemos devolver un texto que indique que no posee jefe. La función de conversión TO_CHAR() permite convertir un numero en un texto:

```
select last_name,nvl(to_char(manager_id),'Sin Jefe') from employees;
```

En este caso cuando el valor encontrado es NULL devuelve el valor Sin Jefe. Otro tipo de funciones muy útiles son las que trabajan con fechas. Por ejemplo, la función SYSDATE() devuelve la fecha actual del sistema. Si usamos esta función en combinación con la función MONTHS_BETWEEN() podemos obtener cuantos años han pasado entre dos fechas.

```
select hire_date, floor(months_between(sysdate, hire_date) /12) as antiguedad
from employees;
```

La función MONTH_BETWEEN() devuelve el número de meses entre dos fechas. Como el valor devuelto puede estar en decimales usamos la función FLOOR() que redondea hacia abajo. Podemos incluso obtener cual es el empleado más antiguo de la empresa usando la función MAX():

```
select max(floor(months_between(sysdate, hire_date) /12)) from employees;
```

Si usamos la función MIN() obtenemos lo contrario, es decir, el empleado con menos antigüedad en la empresa:

```
select min(floor(months_between(sysdate, hire_date) /12)) from employees;
```

También podemos mostrar la fecha en otro formato usando la función de conversión TO_CHAR():

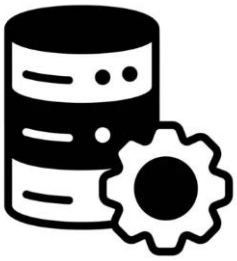
```
select first_name, last_name, to_char(hire_date, 'DD-MM-YYYY') from employees
order by hire_date desc;
```

Otra función muy utilizada es CONCAT(), esta función se encarga de unir dos cadenas y retornar la unión de dichas cadenas. Para ello la función necesita de las dos cadenas a pasarse. Por ejemplo, en este ejercicio a través de un SELECT agregaremos a cada correo electrónico el dominio MYCOMPANY.COM:

```
select concat(email,'@MYCOMPANY.COM') from employees;
```

La función UPPER() convierte una cadena a mayúsculas. La función LOWER() convierte una cadena a minúsculas (lo opuesto a la función UPPER). La función INITCAP() imprime en mayúscula la primera letra de cada palabra. Probaremos el uso de estas funciones listando los apellidos de los empleados en mayúsculas, los nombres en minúsculas y sus correos electrónicos con la primera letra en mayúsculas.

```
select upper(last_name)|| ',' || initcap(first_name) || ' ' || lower(email) ||
'@mycompany' from employees
```



Nota

Muchas veces los datos almacenados son un problema ya que pueden almacenarse en mayúsculas, minúsculas o una combinación entre ellos. Las consultas que usan las funciones UPPER(), LOWER(), INITCAP() no solución el problema de raíz de como fueron almacenados pero posibilita una recuperación inmediata de los mismos.

Las funciones LPAD() y RPAD() permiten rellenar los resultados repitiendo un carácter a la izquierda o a la derecha de cualquier cadena. LPAD rellena a la izquierda de una cadena mientras que RPAD a la derecha. Ambas funciones toman tres parámetros de entrada, el primero es la cadena que desea rellenar; el segundo parámetro la longitud a la que se debe rellenar la cadena; y el tercer parámetro el carácter con el que se rellenará. Por ejemplo:

```
select rpad(department_name, 25, '.') departmento, lpad(location_id, 15, '.')
zona  from departments;
```

Esta consulta coloca el nombre del departamento con un relleno a la derecha hasta una longitud total de 25 con el carácter de relleno "." Si algún nombre de departamento tiene exactamente 25 caracteres o más, no se agregará ningún carácter de relleno. La función LPAD agrega quince caracteres de relleno a la izquierda del código de zona.
Cuando necesite buscar una subcadena dentro de una cadena puede hacerlo con la función INSTR(). INSTR() devuelve la posición de la subcadena encontrada dentro de la cadena. En este nuevo ejemplo usaremos la función INSTR() sobre la columna last_name de la tabla employees para localizar todas las ocurrencias de la subcadena "an".

```
select last_name, instr(last_name, 'an') posicion from employees order by
last_name;
```

Puede ver por ejemplo como aparece 2 en la segunda columna en la fila número cinco. Esto sucede porque aparece encuentra la subcadena “an” a partir de la posicion 2 en el apellido BANDA. A veces es necesario extraer una parte de una cadena para obtener el resultado deseado. La función SUBSTR() puede ayudarle con esa tarea. La función SUBSTR() acepta dos parámetros obligatorios y uno opcional. El primer parámetro es el literal o valor de columna sobre el que desea que opere la función SUBSTR. El segundo parámetro es la posición del carácter inicial de la subcadena y el tercer parámetro opcional es el número de caracteres que se incluirán en la subcadena. Si no se especifica el tercer parámetro, la función SUBSTR devolverá el resto de la cadena. Por ejemplo:

```
select last_name, substr(last_name, 1, 3) from employees order by last_name;
```

En este caso la función substr() extrae los tres primeros caracteres de cada apellido. Podemos combinar las dos funciones anteriores SUBSTR() e INSTR() para mostrar la parte de cada columna last_name de la tabla employees que contiene la subcadena "an". En este ejemplo, la salida de la función INSTR() proporciona el valor del parámetro de entrada que especifica la posición del carácter inicial de la función SUBSTR(). En los valores del campo last_name en los que no se encuentra la subcadena "an", se devuelve el valor de la columna last_name completo por dos razones: SUBSTR() trata la posición inicial como si fuera 1 (es decir, la primera posición de la cadena) y, dado que la consulta omite el parámetro de longitud opcional, se devuelve el resto completo de la cadena.

```
select last_name, instr(last_name, 'an') posicion,
substr(last_name,instr(last_name, 'an')) from employees order by last_name;
```

FUNCIONES DE AGREGACION

Las funciones de agregación en SQL nos permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos. Las instrucciones SELECT que utilizan funciones de agregación pueden incluir dos cláusulas opcionales: GROUP BY y HAVING. La cláusula GROUP BY agrega y agrupa resultados por valores únicos en columnas específicas.

La cláusula HAVING restringe los resultados de retorno en filas donde una condición de agregación en particular es true. La cláusula HAVING se usa de la misma manera que WHERE: para restringir filas según el valor de una columna.
Las funciones de agregación básicas son:

- **COUNT**: Devuelve el número total de filas seleccionadas por la consulta.
- **MIN**: Devuelve el valor mínimo del campo que especifiquemos.
- **MAX**: Devuelve el valor máximo del campo que especifiquemos.
- **SUM**: Suma los valores del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.
- **AVG**: Devuelve el valor promedio del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

En este ejemplo desarrollaremos una consulta que agrupara los distintos puestos laborales y sumara por cada uno de ellos los salarios que hay en cada uno.

```
select job_id,sum(salary) from employees group by job_id;
```

La cláusula HAVING es otra cláusula que es relevante para funciones agregadas. Le permite limitar los resultados que se devuelven después de haberlos agrupado.

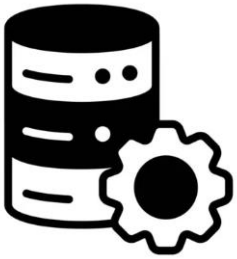
```
select job_id,sum(salary) from employees group by job_id having
job_id='IT_PROG';
```

Esta última consulta muestra únicamente la suma de los salarios, pero limitando el resultado al puesto laboral cuyo id es 14. Es importante recalcar que HAVING es utilizado para realizar restricciones después de haber agrupado registros mientras que la cláusula WHERE restringe antes de agrupar.

Ahora mostraremos otro ejemplo:

```
select department_id, max(salary)from employees group by department_id having
max(salary) > 10000;
```

Devuelve la suma de los salarios mayores a 1000 de cada departamento de la tabla empleados.



Nota

La cláusula WHERE funciona con los datos antes de que se hayan agrupado y la cláusula HAVING funciona con los datos después de que se hayan agrupado.

En este ejercicio puede verse la diferencia:

```
select job_id, sum(salary) from employees where job_id in (2,4,5) group by
job_id having sum(salary) > 10000;
```

En este ejemplo filtramos primero por los job_id cuyo código sean 2, 4 y 5, luego lo agrupamos por el mismo campo y luego devolvemos la suma de salarios de los empleados que sean mayores a 1000.
La función AVG() devuelve el promedio de los valores en una columna. Por ejemplo:

```
select job_id,avg(salary) from employees group by job_id having
job_id='SH_CLERK';
```

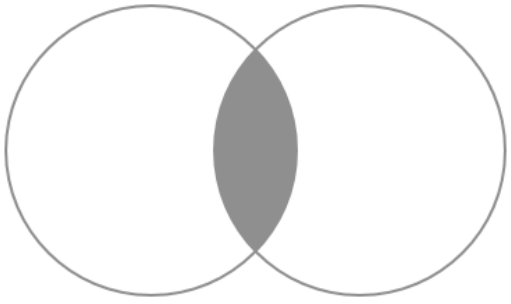
En esta consulta devuelve el promedio de los salarios cuyo job_id es 9. Al utilizar la cláusula GROUP BY no garantiza que los datos se devuelvan ordenados. Suele ser una práctica recomendable incluir una cláusula ORDER BY por las mismas columnas que utilicemos en GROUP BY, especificando el orden que nos interese. Por ejemplo:

```
select country_id,count(country_id) from locations group by country_id order
by country_id;
```

Esta consulta cuenta la cantidad de departamentos por países que tiene la empresa, ordenado por el código de país.

UNIONES SQL (JOINS)

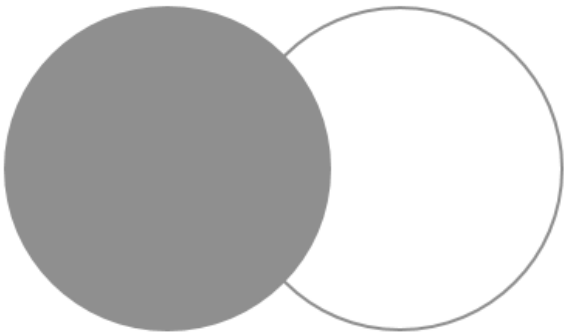
La sentencia JOIN en SQL permite combinar registros de dos o más tablas en una base de datos relacional. Si bien existen numerosos tipos de uniones que se pueden realizar, las más comunes son la INNER JOIN y la OUTER JOIN. Cuando se realiza un INNER JOIN (combinación interna) se devuelve un nuevo conjunto de datos con todas las instancias de la combinación donde se cumplió la condición. Si la condición no se cumplió entre las tablas, las filas se ignoran. Este tipo de combinación dará como resultado el menor número de resultados.
El siguiente diagrama de Venn ilustra una combinación interna cuando se combinan dos conjuntos



INNER JOIN

```
select emps.employee_id,
       emps.first_name,
       jh.department_id,
       jh.start_date,
       jh.end_date
from
       employees emps
inner join
       job_history jh
on  ( emps.employee_id = jh.employee_id );
```

El resultado es una combinación de las tablas empleados e historial de trabajos donde las filas se han unido por los id de empleados. Fíjese que no hay datos por ejemplo del id de empleado número 120 ya que no posee ningún historial de trabajos en otras áreas. La solución a esto puede ser una combinación externa izquierda, eso significa que todos los resultados de la tabla izquierda (o primera tabla) se mostrarán en el resultado, ya sea que coincidan o no con la tabla unida en la condición. Si no coinciden con ninguna fila de la tabla unida, se adjuntarán un valor nulo a la columna.
El siguiente diagrama de Venn ilustra la combinación izquierda:



LEFT OUTER JOIN

```
select emps.employee_id,
       emps.first_name,
       jh.department_id,
       jh.start_date,
       jh.end_date
from   employees emps
left outer join job_history jh
on ( emps.employee_id = jh.employee_id );
```

En este caso podemos ver en el resultado como aparecen los empleados que no tienen historial en otras áreas de trabajo.
Otra alternativa al uso de JOIN es SELF JOIN. Mientras que la mayoría de los JOIN enlazan dos o más tablas entre sí para presentar sus datos de forma conjunta, un SELF JOIN enlaza una tabla consigo misma. Para utilizar este tipo de unión, la tabla debe contener

una columna (llámese X) que actúe como clave primaria, y otra columna (llámese Y) que almacene valores que puedan coincidir con los valores de la columna X. En nuestro caso tenemos la tabla EMPLOYEES que tiene un ID de empleado y un ID de un gerente. Como un gerente es un empleado la información del gerente se encuentra también en la tabla empleados. Podemos obtener un listado de empleados con todos los datos de sus jefes.

```
select
    e.employee_id,
    e.first_name,
    e.last_name,
    e.manager_id,
    e1.first_name,
    e1.last_name
from employees e
inner join employees e1
on e.manager_id=e1.employee_id
```

Si queremos listar todos los empleados, tengan o no tengan jefes podemos usar un LEFT OUTER JOIN en su lugar. La consulta seria la siguiente:

```
select
    e.employee_id,
    e.first_name,
    e.last_name,
    e.manager_id,
    e1.first_name,
    e1.last_name
from employees e left outer join employees e1
on e.manager_id=e1.employee_id
```

SUBCONSULTAS

Una subconsulta es una instrucción SELECT incrustada en otra instrucción SQL. Las subconsultas se utilizan ampliamente para responder una pregunta dentro de otra pregunta. Por ejemplo, que empleados ganan más que el empleado número 110. Para recuperar esta información, debe responder dos preguntas, cada una en una consulta separada. Primero, cual es el salario del empleado número 110.

```
select salary from employees where employee_id=110;
```

Y segundo, en base al resultado obtenido en la consulta anterior podemos obtener un listado de los empleados que cumplen esa condición:

```
select employee_id, salary from employees where salary>8200;
```

En lugar de ejecutar cada consulta por separado, puede combinar las dos consultas, colocando una consulta dentro de la otra.

```
select employee_id, salary from employees where salary>(select salary from
employees where employee_id=110);
```

Hay que tener en cuenta varias cosas. Primero, la subconsulta se ejecuta antes de la consulta principal, luego el resultado de la subconsulta se envía a la consulta principal. Segundo, siempre la subconsulta debe encerrase entre paréntesis. Tercero, la subonsulta va en el lado derecho de la expresión que usa un operador de comparación.
Puede usar funciones de agregación en una subconsulta. Por ejemplo, podríamos listar los empleados que ganan más que el promedio:

```
select employee_id, salary from employees where salary>(select avg(salary)
from employees);
```

PAGINAR LOS RESULTADOS

Si trabaja con consultas todo el tiempo sabrá que la cantidad de filas que se devuelven en una consulta a veces puede resultar abrumadora, muchas aplicaciones necesitan paginar las filas extraídas de la base de datos, o al menos recuperar las primeras N filas. Si utiliza Oracle 12c o una versión posterior puede utilizar la cláusula FETCH FIRST para limitar la cantidad de filas devueltas. La cláusula FETCH FIRST funciona de manera similar a la clausula LIMIT usada en otros motores relacionales. A continuación, se muestra un ejemplo de cómo utilizar la cláusula FETCH FIRST para devolver solo las primeras 5 filas de la tabla employees:

```
select * from employees fetch first 5 rows only;
```

Otra opción para limitar la cantidad de filas devueltas en una consulta de base de datos Oracle es utilizar la cláusula FETCH NEXT en combinación con la cláusula OFFSET. Este método le permite omitir una cierta cantidad de filas y devolver una cantidad específica de filas después de las filas omitidas. Al utilizar la cláusula FETCH NEXT con OFFSET, puede paginar fácilmente los resultados de su consulta y recuperar un rango específico de filas de un conjunto de resultados más grande. Esto puede resultar especialmente útil cuando se trabaja con tablas grandes o cuando se presentan los resultados de la consulta a los usuarios en una interfaz de usuario.

```
select * from employees offset 5 rows fetch next 4 rows only;
```

En este ejemplo, la cláusula OFFSET omite las primeras 5 filas devueltas por la consulta, la cláusula FETCH NEXT devuelve las 4 filas siguientes a las omitidas.