****

**Índice Primer Semestre**

**1. Introducción a las Bases de Datos**

**2. ¿Qué es una base de datos?**

**3. Datos y tipos de datos**

**4 Sistema Gestor de Bases de Datos (SGBD)**

**5. Definición de SGBD**

**6. Características de un SGBD**

**7. Tipos de SGBD**

**8. Relacionales**

**9. No Relacionales**

**10. Multidimensional**

**11. Distribuidas**

**12. Orientadas a objetos**

**13. Herramientas de Análisis de Datos Indispensables para el Data Management**

**14. Bases de Datos Relacionales**

**15. ¿Cuándo usar una base de datos relacional?**

**16. Modelo Entidad-Relación**

**17. Entidades y atributos**

**18. Cardinalidad**

**19. Acciones (Operaciones en relaciones)**

**20. Llave primaria y llave foránea**

**21. Normalización**

**22. 1FN**

**23. 2FN**

**24. 3FN**

**25. Introducción a SQL**

**26. Definición y propósito de SQL**

**27. Usos de SQL**

**28. Tipos de comandos SQL**

**29. Comandos DDL**

**30. CREATE**

**31. ALTER**

**32. DROP**

**33. TRUNCATE**

**34. RENAME**

**35. Comandos DML**

**36. SELECT**

**37. INSERT INTO**

**38. UPDATE**

**39. DELETE**

**40. Tablas, filas y columnas**

**41. Tipos de datos de las columnas**

**42. Consultas SQL Básicas**

**43. WHERE**

**44. GROUP BY**

**45. GROUP BY + HAVING**

**46. ORDER BY**

**47. Consultas SQL Avanzadas**

**48. INNER JOIN**

**49. LEFT JOIN**

**50. RIGHT JOIN**

**51. FULL OUTER JOIN**

**52. Subconsultas en SQL**

**53. Subconsulta escalar**

**54. Subconsulta de múltiples filas**

**55. Subconsulta correlacionada**

**56. Índices en SQL**

**57. Restricciones SQL**

**58. Transacciones en SQL**

**59. Bases de Datos Relacionales vs. Bases de Datos No Relacionales**

**60. Casos de Uso de las Bases Relacionales**

**61. Futuro de las Bases de Datos Relacionales**

**62. Que es la inteligencia de negocios?**

**63. Conceptos y definiciones**

**64. minería de datos**

**65. dashboards**

**66. patrones de comportamiento**

**67. anomalías**

**68. segmentación de mercado**

**69. predicción**

**70. recolección de datos**

**71. limpieza de datos**

**72. transformación de datos**

**73. selección de conjuntos de datos**

**74. generación de informes**

**75. métricas**

**76. valores de referencia**

**78. análisis descriptivo y estadístico**

**79. staging**

**80. data marts**

**81. Componentes principales de la BI**

**82. ETL**

**83. Fase de extracción**

**84. métodos de extraccion**

**85. Fase de transformación**

**86. Fase de carga**

**87. Que es data warehouse?**

**88. Sistema OLAP y OLTP**

**89. Esquemas de un almacén de datos**

**90. Reporting**

**91. Analystics**

[**https://github.com/sebastia9330**](https://github.com/sebastia9330)

**### Parte 1: Explicaciones**

#### Inteligencia de Negocios (BI)

\*\*1. ¿Qué es la inteligencia de negocios?\*\*

La inteligencia de negocios (BI) es un proceso que utiliza tecnologías, aplicaciones y prácticas para recopilar, integrar, analizar y presentar información empresarial. El objetivo de la BI es ayudar a las organizaciones a tomar decisiones basadas en datos mediante el análisis de la información recopilada de diversas fuentes.

#### Bases de Datos

\*\*1. Introducción a las Bases de Datos\*\*

\*\*1.1. ¿Qué es una base de datos?\*\*

Una base de datos es un conjunto organizado de datos que se almacenan y se acceden electrónicamente. Las bases de datos permiten la gestión eficiente de grandes cantidades de información, facilitando su acceso, manipulación y actualización.

\*\*1.2. Datos y tipos de datos\*\*

Los datos son hechos sin procesar, que pueden representar texto, números, imágenes, etc. Los tipos de datos son las categorías que definen el tipo de valor que puede ser almacenado y manipulado en una base de datos, como texto (string), números enteros (integer), fechas, etc.

**### Parte 2: Preguntas para Retroalimentación**

1. \*\*¿Cuál es el principal objetivo de la inteligencia de negocios?\*\*

2. \*\*¿Qué diferencia hay entre una base de datos relacional y una no relacional?\*\*

3. \*\*¿Qué es un sistema gestor de bases de datos (SGBD)?\*\*

4. \*\*¿Cómo describirías la relación entre datos y tipos de datos en una base de datos?\*\*

5. \*\*¿Qué rol juega la inteligencia de negocios en la toma de decisiones empresariales?\*\*  
  
respuestas:

1. el principal objetivo de la inteligencia de datos es ayudar a las compañías con la toma de decisiones, mediante el manejo de información  
2. Una base de datos relacional maneja una estructura de filas y tablas, además de eso maneja relaciones entre las tablas, las bases de datos relacionales trabajan en modelo de conjuntos, clave valor y no funcionan con relaciones

3 un sistema gestor de base de datos es el software que nos ayuda a desplegar nuestras bases de datos y nos otorga funcionalidades de crud, de respaldo, de uso, pues tiene por lo general plataformas amigables con el usuario, y de seguridad  
4. Describiría dicha relación viendo que tipos de datos voy a manejar antes de crear mis tablas y además analizando si son datos estructurados, semiestructurados o no estructurados para trabajar con ellos en la medida que se necesite  
5. El rol de ayudarnos en el análisis de información para tomar acciones con base en el pasado y asi tener una mejor planeación futura

Aquí tienes las correcciones y ajustes a tus respuestas:

1. \*\*El principal objetivo de la inteligencia de negocios\*\* es ayudar a las compañías en la toma de decisiones mediante el análisis de información recopilada de diversas fuentes.

2. \*\*Una base de datos relacional\*\* maneja una estructura de filas y tablas, permitiendo establecer relaciones entre diferentes tablas a través de claves primarias y foráneas. \*\*Una base de datos no relacional\*\*, en cambio, puede manejar modelos de datos como clave-valor, documentos, grafos, entre otros, y generalmente no depende de relaciones entre datos, siendo más flexible para ciertos tipos de aplicaciones.

3. \*\*Un sistema gestor de bases de datos (SGBD)\*\* es un software que facilita la creación, gestión y operación de bases de datos. Proporciona funcionalidades para operaciones CRUD (crear, leer, actualizar, eliminar), respaldo de datos, seguridad y, usualmente, una interfaz amigable para el usuario.

4. \*\*La relación entre datos y tipos de datos\*\* es esencial en el diseño de bases de datos. Al definir los tipos de datos, se determina cómo se almacenarán, procesarán y se accederá a los datos. Es fundamental analizar si los datos serán estructurados, semiestructurados o no estructurados para seleccionar el tipo de base de datos más adecuado.

5. \*\*El rol de la inteligencia de negocios\*\* es asistir en el análisis de información histórica para tomar decisiones informadas y planificar estratégicamente el futuro de la organización.

¡Buen trabajo en tus respuestas! Estas correcciones mejoran la precisión y claridad.

**### Parte 2: Explicaciones**

#### Inteligencia de Negocios (BI)

\*\*Conceptos y definiciones\*\*

La inteligencia de negocios (BI) abarca un conjunto de estrategias, tecnologías y aplicaciones utilizadas para recopilar, integrar, analizar y presentar datos empresariales. Su objetivo es mejorar la toma de decisiones dentro de la organización. Incluye conceptos como data warehousing, reporting, analítica avanzada, minería de datos, y dashboards.

\*\*Componentes principales de la BI\*\*

Los componentes principales de la inteligencia de negocios incluyen:

- \*\*Data Warehousing\*\*: Repositorio centralizado donde se almacenan grandes volúmenes de datos históricos.

- \*\*ETL (Extract, Transform, Load)\*\*: Procesos que extraen datos de diferentes fuentes, los transforman para su análisis y los cargan en un data warehouse.

- \*\*Reporting\*\*: Herramientas y procesos para generar informes basados en datos.

- \*\*Analytics\*\*: Técnicas avanzadas de análisis de datos, como minería de datos y machine learning.

\*\*ETL (Extract, Transform, Load)\*\*

El proceso ETL es fundamental en la inteligencia de negocios, permitiendo la integración y preparación de datos desde múltiples fuentes para el análisis en un data warehouse.

- \*\*Fase de extracción\*\*: Se recuperan los datos de diversas fuentes, como bases de datos, archivos y aplicaciones.

- \*\*Fase de transformación\*\*: Los datos se limpian, transforman y preparan para su análisis.

- \*\*Fase de carga\*\*: Los datos transformados se almacenan en el data warehouse para su posterior análisis.

\*\*¿Qué es data warehouse?\*\*

Un data warehouse es un sistema de almacenamiento centralizado diseñado para facilitar el análisis y la consulta de grandes volúmenes de datos históricos. A diferencia de las bases de datos operativas, que están optimizadas para transacciones diarias, un data warehouse está optimizado para consultas complejas y análisis de grandes conjuntos de datos.

\*\*Sistema OLAP y OLTP\*\*

- \*\*OLAP (Online Analytical Processing)\*\*: Sistema que permite realizar consultas complejas y análisis multidimensionales de grandes volúmenes de datos. Es ideal para el análisis de tendencias y el apoyo a la toma de decisiones.

- \*\*OLTP (Online Transaction Processing)\*\*: Sistema orientado a la gestión de transacciones en tiempo real. Es usado en aplicaciones que requieren operaciones rápidas y de alta disponibilidad.

\*\*Esquemas de un almacén de datos\*\*

Existen diferentes esquemas para organizar la estructura de un data warehouse, entre los más comunes están:

- \*\*Esquema en estrella\*\*: Consiste en una tabla de hechos central conectada a varias tablas de dimensiones.

- \*\*Esquema en copo de nieve\*\*: Similar al esquema en estrella, pero con dimensiones normalizadas, es decir, con tablas de dimensiones que se dividen en subtablas.

- \*\*Esquema de galaxia\*\*: Extensión del esquema en estrella que utiliza múltiples tablas de hechos.

\*\*Reporting\*\*

Reporting se refiere a la generación de informes que presentan los datos de manera estructurada para ayudar a las organizaciones a monitorear su rendimiento y tomar decisiones informadas. Los informes pueden ser estáticos o interactivos, y generalmente se presentan en formatos gráficos o tabulares.

\*\*Analytics\*\*

Analytics implica el uso de técnicas avanzadas de análisis de datos, como la minería de datos, el machine learning, y la estadística, para descubrir patrones, predecir tendencias y tomar decisiones basadas en datos. Es una de las áreas más poderosas de la inteligencia de negocios, ya que permite un análisis profundo de los datos para obtener información procesable.

#### Bases de Datos

\*\*Sistema Gestor de Bases de Datos (SGBD)\*\*

\*\*2.1. Definición de SGBD\*\*

Un Sistema Gestor de Bases de Datos (SGBD) es un software que permite la creación, administración y manipulación de bases de datos. Facilita la definición de esquemas, la gestión de datos y la ejecución de consultas, garantizando la integridad y seguridad de los datos almacenados.

\*\*2.2. Características de un SGBD\*\*

Las principales características de un SGBD incluyen:

- \*\*Independencia de datos\*\*: Separación entre los datos y las aplicaciones que los usan.

- \*\*Control de redundancia\*\*: Minimización de la duplicación de datos.

- \*\*Seguridad y control de acceso\*\*: Gestión de quién puede acceder y modificar los datos.

- \*\*Soporte para transacciones\*\*: Asegura la consistencia y confiabilidad de los datos durante las operaciones.

- \*\*Gestión de consultas\*\*: Optimización y ejecución de consultas de manera eficiente.

\*\*2.3. Tipos de SGBD\*\*

Los SGBD se clasifican en varios tipos según su modelo de datos:

- \*\*Relacionales\*\*: Basados en el modelo de tablas, donde los datos se organizan en filas y columnas.

- \*\*No Relacionales\*\*: Incluyen bases de datos clave-valor, documentos, grafos y otros, que no siguen un esquema tabular.

- \*\*Multidimensionales\*\*: Usados principalmente en sistemas OLAP, permiten analizar datos en múltiples dimensiones.

- \*\*Distribuidas\*\*: Almacenan y gestionan datos en múltiples servidores o ubicaciones físicas diferentes.

- \*\*Orientadas a objetos\*\*: Combina conceptos de bases de datos y programación orientada a objetos, permitiendo almacenar objetos complejos como datos.

### Parte 2: Preguntas para Retroalimentación

1. \*\*¿Cuáles son los componentes principales de la inteligencia de negocios (BI)?\*\*

2. \*\*¿Qué es un sistema OLAP y en qué se diferencia de un sistema OLTP?\*\*

3. \*\*¿Cuál es la función de la fase de transformación en el proceso ETL?\*\*

4. \*\*¿Qué es un data warehouse y para qué se utiliza?\*\*

5. \*\*Menciona y describe brevemente dos características de un SGBD.\*\*

Respuestas

1. Etl, data warehouse, reporting y analytics
2. Un sistema olap permite el análisis de gran cantidad de datos y el sistema oltp, va mas enfocado a la velocidad de procesamiento
3. Extrar los datos de las fuentes de información, tranformarlos en datos limpíos, sin errores, redundacias y mala clasificación, carga de datos al data warehouse
4. Es un alamacen de datos donde después de la carga quedan los datos disponibles para consultas, reportes y alaisis de los mismos
5. Transacciones ACID el se asegura de que cada transacción de la base de datos siga este modelo, seguridad, definiedo roles específicos para saber quien puede modular los datos y quien no

Aquí tienes las correcciones y ajustes a tus respuestas:

1. \*\*Componentes principales de la inteligencia de negocios (BI):\*\* Los componentes principales son ETL (Extracción, Transformación y Carga), Data Warehouse, Reporting y Analytics.

2. \*\*Diferencia entre OLAP y OLTP:\*\* Un sistema OLAP está diseñado para el análisis de grandes volúmenes de datos de forma multidimensional, mientras que OLTP se enfoca en la eficiencia y rapidez en la ejecución de transacciones en tiempo real.

3. \*\*Fase de transformación en el proceso ETL:\*\* En esta fase, los datos extraídos de las fuentes se limpian, eliminando errores, redundancias y malas clasificaciones, y se transforman para adecuarse al formato requerido antes de ser cargados en el Data Warehouse.

4. \*\*¿Qué es un Data Warehouse?\*\* Un Data Warehouse es un almacén centralizado donde se almacenan grandes volúmenes de datos que han sido procesados, quedando disponibles para consultas, reportes y análisis.

5. \*\*Características de un SGBD:\*\*

- \*\*Transacciones ACID:\*\* Asegura que cada transacción sea atómica, consistente, aislada y duradera.

- \*\*Seguridad:\*\* Permite definir roles y permisos para controlar quién puede acceder y modificar los datos.

¡Tus respuestas son sólidas! Estas correcciones mejoran la precisión y claridad en tus explicaciones.

**### Parte 3: Explicaciones**

#### Inteligencia de Negocios (BI)

\*\*ETL (Extract, Transform, Load)\*\*

- \*\*Fase de extracción:\*\* La extracción es el primer paso en el proceso ETL. Aquí, los datos se recopilan de diversas fuentes, como bases de datos, archivos, aplicaciones web, y sistemas ERP. Es crucial que la extracción sea eficiente y precisa, ya que los errores en esta fase pueden propagarse a lo largo del proceso.

- \*\*Fase de transformación:\*\* Una vez que los datos se han extraído, pasan a la fase de transformación. Aquí, los datos se limpian y se convierten en un formato adecuado para su análisis en el Data Warehouse. Las transformaciones incluyen operaciones como normalización, desduplicación, conversión de formatos, y la aplicación de reglas de negocio.

- \*\*Fase de carga:\*\* En esta fase final, los datos transformados se cargan en el Data Warehouse. Esta etapa puede realizarse de manera incremental o en cargas masivas, dependiendo de la frecuencia y el volumen de los datos. La carga debe ser manejada con cuidado para asegurar que los datos estén disponibles para análisis sin interrumpir las operaciones en curso.

\*\*Data Warehouse y sus Esquemas\*\*

- \*\*Esquemas de un almacén de datos:\*\* Los esquemas son las estructuras que definen cómo se organizan los datos dentro de un Data Warehouse. Los principales esquemas incluyen:

- \*\*Esquema en estrella:\*\* Tiene una tabla de hechos central conectada a varias tablas de dimensiones. Es simple y fácil de entender.

- \*\*Esquema en copo de nieve:\*\* Es una extensión del esquema en estrella donde las tablas de dimensiones están normalizadas en múltiples niveles, reduciendo la redundancia.

- \*\*Esquema de galaxia:\*\* Consiste en múltiples tablas de hechos que comparten dimensiones comunes, lo que permite análisis más complejos.

\*\*Reporting y Analytics\*\*

- \*\*Reporting:\*\* El reporting en la inteligencia de negocios se refiere a la creación de informes detallados basados en datos, permitiendo a las organizaciones rastrear su rendimiento y tomar decisiones basadas en datos. Los informes pueden ser generados de forma automática y presentarse en diversos formatos, como gráficos, tablas y dashboards interactivos.

- \*\*Analytics:\*\* Analytics se enfoca en el uso de técnicas avanzadas de análisis de datos, como la minería de datos, machine learning, y análisis predictivo. El objetivo es descubrir patrones ocultos, prever tendencias futuras y proporcionar insights que ayuden en la toma de decisiones estratégicas.

#### Bases de Datos

\*\*3. Herramientas de Análisis de Datos Indispensables para el Data Management\*\*

Existen varias herramientas que son fundamentales para el análisis de datos y la gestión de bases de datos. Algunas de las más comunes incluyen:

- \*\*SQL (Structured Query Language):\*\* Es el lenguaje estándar para consultar y manipular bases de datos relacionales.

- \*\*Herramientas de BI:\*\* Como Power BI, Tableau y Qlik, que permiten la visualización y análisis de datos a través de dashboards interactivos.

- \*\*Lenguajes de Programación:\*\* Python y R son ampliamente utilizados para análisis de datos, incluyendo el uso de librerías como Pandas, NumPy, y SciPy en Python, o dplyr y ggplot2 en R.

- \*\*Hadoop y Spark:\*\* Herramientas para el procesamiento y análisis de grandes volúmenes de datos distribuidos.

- \*\*Data Lakes:\*\* Repositorios centralizados que permiten almacenar grandes volúmenes de datos sin necesidad de estructurarlos de antemano.

\*\*4. Bases de Datos Relacionales\*\*

- \*\*¿Cuándo usar una base de datos relacional?\*\*

Las bases de datos relacionales son ideales cuando los datos están estructurados y se requieren relaciones claras entre diferentes conjuntos de datos. Por ejemplo, son útiles para gestionar transacciones en tiempo real, sistemas ERP, CRM, y aplicaciones que necesitan consistencia e integridad de los datos.

- \*\*Modelo Entidad-Relación (MER):\*\*

El Modelo Entidad-Relación es una técnica de modelado que se utiliza para diseñar bases de datos relacionales. Permite representar gráficamente las entidades (objetos o conceptos) y las relaciones entre ellas.

- \*\*Entidades y atributos:\*\*

Una entidad representa un objeto del mundo real que tiene una existencia independiente. Los atributos son las propiedades o características de las entidades. Por ejemplo, en una base de datos de una universidad, una entidad podría ser "Estudiante", y sus atributos podrían ser "Nombre", "Matrícula", "Correo electrónico", etc.

- \*\*Cardinalidad:\*\*

La cardinalidad define la relación entre dos entidades en términos de números. Por ejemplo, un profesor puede enseñar a muchos estudiantes (relación uno a muchos), y un estudiante puede estar inscrito en múltiples cursos (relación muchos a muchos).

- \*\*Acciones (Operaciones en relaciones):\*\*

Las acciones o operaciones en relaciones se refieren a las formas en que las entidades interactúan entre sí. Por ejemplo, una operación podría ser "asignar" un estudiante a un curso o "inscribir" un estudiante en una materia.

### Parte 3: Preguntas para Retroalimentación

1. \*\*¿Qué sucede durante la fase de extracción en el proceso ETL?\*\*

2. \*\*Describe brevemente la función de la fase de carga en el proceso ETL.\*\*

3. \*\*¿Qué es un esquema en copo de nieve en un Data Warehouse y cómo se diferencia de un esquema en estrella?\*\*

4. \*\*¿Cuáles son algunas de las herramientas más utilizadas en el análisis de datos y la gestión de bases de datos?\*\*

5. \*\*¿Qué son las entidades y atributos en el Modelo Entidad-Relación (MER)?\*\*

Respuestas

1. Es la fase fundamentas, es donde se toman todos los datos de las fuentes de información, pueden ser archivos planos, archivos Excel, bases de datos existentes pero no ordenadas, lo que hacemos es extrar los datos para pasar a la fase de transformación, esta fase es crucial pues un error que se presente en ella, perdurara durante todo el proceso
2. Es la fase donde tomamos los datos luego de la transformación y los cargamos al data wharehouse, se puede hacer de forma masiva o por segmentos
3. Es un esquema mas complejo donde las tablas dimensiones tienen sub dimensiones, eliminado las redundacias que pueden existir, y se diferencia en la cantidad de tablas que se manejan
4. Sql como lenguaje mas usado en bases de datos relacionales

* Powerbi como herramienta de análisis y reporting
* Python como lenguaje de programación con librerías como pandas y numpy para el análisis de datos

1. Las entidades son los objetos que conforman nuestra base de datos se representan como rectángulos en el mes, y los atributos son los datos que componen dichas entidades, ejemplo nombre,cedula, teléfono y demás

**### Clase 4: Bases de Datos Relacionales y Modelo Entidad-Relación (MER)**

#### 4.1 ¿Cuándo usar una base de datos relacional?

Las bases de datos relacionales son ideales cuando necesitas manejar datos estructurados que se organizan en tablas y existe la necesidad de establecer relaciones entre diferentes conjuntos de datos. Son útiles en escenarios donde los datos necesitan integridad referencial, es decir, las relaciones entre tablas deben mantenerse consistentes. Estas bases de datos son especialmente recomendadas para aplicaciones empresariales, donde las operaciones CRUD (Crear, Leer, Actualizar, Borrar) son comunes, y donde se requiere realizar consultas complejas que involucran múltiples tablas.

#### 4.2 Modelo Entidad-Relación (MER)

El Modelo Entidad-Relación (MER) es una metodología para modelar datos que utiliza diagramas para representar las entidades, los atributos y las relaciones entre entidades. Este modelo es fundamental para el diseño de bases de datos, ya que permite visualizar cómo los diferentes elementos de datos interactúan entre sí antes de implementar la base de datos física.

- \*\*4.2.1 Entidades y atributos\*\*:

- \*\*Entidades\*\*: Representan objetos del mundo real que tienen una existencia independiente. Por ejemplo, en un sistema de gestión de biblioteca, "Libro", "Autor" y "Cliente" son entidades.

- \*\*Atributos\*\*: Son las propiedades o características de una entidad. Siguiendo con el ejemplo anterior, un "Libro" puede tener atributos como "Título", "ISBN", y "Fecha de Publicación".

- \*\*4.2.2 Cardinalidad\*\*:

- La cardinalidad describe el número de instancias de una entidad que pueden estar asociadas con las instancias de otra entidad. Existen tres tipos principales:

- \*\*Uno a Uno (1:1)\*\*: Una instancia de una entidad se asocia con una única instancia de otra entidad.

- \*\*Uno a Muchos (1:N)\*\*: Una instancia de una entidad se asocia con muchas instancias de otra entidad.

- \*\*Muchos a Muchos (M:N)\*\*: Muchas instancias de una entidad se asocian con muchas instancias de otra entidad.

- \*\*4.2.3 Acciones (Operaciones en relaciones)\*\*:

- Las acciones o las operaciones en las relaciones son las interacciones que pueden ocurrir entre entidades. Estas operaciones suelen ser CRUD y se implementan en la base de datos para manipular los datos almacenados. Por ejemplo, una operación de "Prestamo" entre las entidades "Libro" y "Cliente" podría registrar cuándo un cliente toma prestado un libro.

#### 4.3 Llave primaria y llave foránea

- \*\*Llave primaria\*\*: Es un atributo (o un conjunto de atributos) que identifica de manera única a una fila en una tabla. Por ejemplo, el atributo "ISBN" en la tabla "Libro" podría ser la llave primaria, ya que cada libro tiene un ISBN único.

- \*\*Llave foránea\*\*: Es un atributo en una tabla que se refiere a la llave primaria de otra tabla, estableciendo una relación entre las dos tablas. Por ejemplo, en una tabla "Prestamo", podría haber una llave foránea que haga referencia al "ISBN" de la tabla "Libro", indicando qué libro se ha prestado.

#### 4.4 Normalización

La normalización es el proceso de organizar las columnas y tablas de una base de datos para reducir la redundancia de datos y mejorar la integridad. El objetivo es dividir una base de datos en tablas más pequeñas y enlazarlas mediante relaciones, lo que facilita la gestión y mejora el rendimiento.

- \*\*4.4.1 1FN (Primera Forma Normal)\*\*: Una tabla está en 1FN si todas las columnas contienen valores atómicos, es decir, no contienen conjuntos de valores o listas.

- \*\*4.4.2 2FN (Segunda Forma Normal)\*\*: Una tabla está en 2FN si está en 1FN y todos los atributos que no son clave dependen completamente de la clave primaria.

- \*\*4.4.3 3FN (Tercera Forma Normal)\*\*: Una tabla está en 3FN si está en 2FN y no hay dependencias transitivas entre los atributos que no son clave.

### Preguntas de Retroalimentación:

1. ¿Qué características hacen que una base de datos relacional sea adecuada para aplicaciones empresariales?

2. Explica con tus palabras qué es una entidad y qué es un atributo en el contexto del Modelo Entidad-Relación.

3. ¿Cuál es la diferencia entre una relación de cardinalidad 1:1 y una relación de cardinalidad 1:N?

4. ¿Qué función cumple la llave foránea en una base de datos relacional?

5. ¿Cómo se diferencia la 2FN de la 3FN en el proceso de normalización?

Respuestas

1. Las características de poder manejar los datos de manera mas sencilla, además que nos da la capacidad de realizar operaciones crud, lo cual puede mejorar el manejo de la información dentro de la empresa
2. Una entidad se relaciona con cualquier objeto del mundo real al que queramos hacer referencia en nuestra base de datos, pueden ser libros, productos, personas, animales etc. Y un atributo es todo lo que le pertenece a ese objeto como nombre, dirección, edad, cedula, o id
3. La diferencia consiste en que 1 a 1 solo puedo relacionas un objeto con otro, ejemplo una manzana solo puede ser fruta y uno a muchos puede decir que un objeto puede pertenecer a muchas tablas, ejemplo un libro puede tener copias en muchas bibliotecas
4. Es la encargada de generar las relaciones entre tablas desde la llave primaria de una tabla hasta la otra que necesite enlazar con esta
5. La diferencia es que en la 2fn almacenamos los datos que no son dependientes de la llave primara en una tabla y en el 3fn verifica la 2fn y además nos separa los valores dentro de esas tablas que dependen de la llave primaria

**### Profundización en las Normas de Normalización**

La \*\*normalización\*\* es un proceso crucial en el diseño de bases de datos relacionales que ayuda a organizar los datos de manera eficiente, evitando redundancias y asegurando la integridad de los datos. Las normas de normalización, o formas normales (FN), son reglas que aplicamos para estructurar las tablas en una base de datos.

#### Primera Forma Normal (1FN)

\*\*Objetivo:\*\* Asegurarse de que todos los valores en una columna sean atómicos, es decir, indivisibles.

\*\*Ejemplo:\*\*

Supongamos que tenemos una tabla de "Pedidos" con las siguientes columnas: `ID\_Pedido`, `Cliente`, `Producto`.

| ID\_Pedido | Cliente | Producto |

|-----------|---------|--------------------|

| 1 | Juan | Manzana, Plátano |

| 2 | Ana | Naranja |

Aquí, la columna `Producto` no está en 1FN porque contiene múltiples valores en una sola celda. Para normalizarla:

| ID\_Pedido | Cliente | Producto |

|-----------|---------|----------|

| 1 | Juan | Manzana |

| 1 | Juan | Plátano |

| 2 | Ana | Naranja |

Ahora, cada celda contiene un solo valor, cumpliendo con la 1FN.

#### Segunda Forma Normal (2FN)

\*\*Objetivo:\*\* Asegurarse de que cada atributo no clave esté completamente dependiente de la clave primaria completa.

\*\*Ejemplo:\*\*

Supongamos que tenemos la siguiente tabla:

| ID\_Pedido | Producto | Cantidad | Precio\_Total |

|-----------|-----------|----------|--------------|

| 1 | Manzana | 2 | 200 |

| 2 | Naranja | 3 | 150 |

En este caso, `Precio\_Total` depende de la cantidad y el precio unitario del producto, pero no de `ID\_Pedido` ni de `Producto`. Para normalizar, separamos la tabla:

\*\*Tabla Pedidos:\*\*

| ID\_Pedido | Producto | Cantidad |

|-----------|-----------|----------|

| 1 | Manzana | 2 |

| 2 | Naranja | 3 |

\*\*Tabla Precios:\*\*

| Producto | Precio |

|-----------|--------|

| Manzana | 100 |

| Naranja | 50 |

Ahora, la tabla cumple con la 2FN.

#### Tercera Forma Normal (3FN)

\*\*Objetivo:\*\* Asegurarse de que no existan dependencias transitivas, es decir, que los atributos no clave no dependan entre sí.

\*\*Ejemplo:\*\*

Supongamos que tenemos una tabla de empleados:

| ID\_Empleado | Nombre | Departamento | Jefe\_Departamento |

|-------------|--------|--------------|-------------------|

| 1 | Pedro | Ventas | Ana |

| 2 | Luis | Finanzas | Juan |

Aquí, `Jefe\_Departamento` depende del `Departamento`, no directamente del `ID\_Empleado`. Para normalizar:

\*\*Tabla Empleados:\*\*

| ID\_Empleado | Nombre | Departamento |

|-------------|--------|--------------|

| 1 | Pedro | Ventas |

| 2 | Luis | Finanzas |

\*\*Tabla Departamentos:\*\*

| Departamento | Jefe\_Departamento |

|--------------|-------------------|

| Ventas | Ana |

| Finanzas | Juan |

Con esto, hemos eliminado la dependencia transitiva y cumplimos con la 3FN.

### Preguntas de Retroalimentación

1. ¿Por qué es importante que las tablas estén en la Primera Forma Normal (1FN)? Da un ejemplo de una tabla que no cumple con la 1FN y cómo la normalizarías.

2. Explica cómo identificar si una tabla no cumple con la Segunda Forma Normal (2FN) y proporciona un ejemplo de cómo corregirla.

3. ¿Qué es una dependencia transitiva? Ilustra con un ejemplo cómo afecta a la Tercera Forma Normal (3FN) y cómo solucionarlo.

4. ¿Qué problemas puede generar una tabla que no sigue las normas de normalización en una base de datos relacional?

5. Considera una tabla de "Órdenes de Compras". Si cada orden contiene información del proveedor, como nombre y dirección, ¿cómo podrías normalizar esta tabla para cumplir con la 2FN y 3FN?

Respuestas

1. Es importante pues al manejar los datos es mucho mejor que cada columna tenga un solo dato  
   ejemplo  
   empleados /teléfono

Juan /3245864032,3106100064

Correcion de tabla emleados

Empleado /teléfono

Juan /3245864032

Juan /3106100064

1. Tenemos que evidencias que un dato no depende de la clave primaria

Ejemplo

Id\_empleado /Nombre /ciudad

1 /juan /bogota

2 /pedro /medellin

En este ejemplo el campo ciudad no depende del id empleado, lo que podríamos hacer es crear una tabla para almacenar las ciudades

1. Es una dependencia que va cambiando con el tiempo, ejemplo: en una tabla libros hay un campo de prestado, sabemos que ese estado va variando, entonces lo mejor seria crear una tabla con un id para el estado y asignar como llave foránea ese campo a la tabla libros
2. Puede general mal manejo en las colsultas, si no cumple la 1fn, o puede generar redundacia en los datos y sobre cargo en las tablas si tenemos muchas columnas en ellas
3. Podríamos crear una tabla dirección donde se almacenen las direcciones del proveedor asi cumplimos la 2fn y podemos asignar un id\_direccion a la tabla ´proveedor, asi si cambia de dirección, no tenemos que cambiar nada en la tabla proveedores sino solo en la tabla dirección y cumplimos la 3fn

**Claro, vamos a abordar la \*\*Clase 5: Introducción a SQL\*\*.**

### 5. Introducción a SQL

\*\*5.1. Definición y propósito de SQL\*\*

SQL (Structured Query Language) es un lenguaje de programación estándar para gestionar y manipular bases de datos relacionales. Su propósito principal es permitir a los usuarios interactuar con las bases de datos a través de la ejecución de consultas, la actualización de datos, y la administración de la estructura de la base de datos.

\*\*5.2. Usos de SQL\*\*

SQL se utiliza para realizar diversas operaciones en una base de datos, como:

- Consultar datos (seleccionar y filtrar información).

- Insertar nuevos datos en las tablas.

- Actualizar datos existentes.

- Eliminar datos que ya no son necesarios.

- Crear y modificar la estructura de las tablas y otras estructuras de base de datos.

\*\*5.3. Tipos de comandos SQL\*\*

SQL se divide en varias categorías de comandos, cada una con un propósito específico. Aquí abordamos los más fundamentales:

\*\*5.3.1. Comandos DDL (Data Definition Language)\*\*

Estos comandos se utilizan para definir y modificar la estructura de las bases de datos y sus objetos.

- \*\*CREATE\*\*: Crea nuevas tablas, vistas, índices, etc.

```sql

CREATE TABLE empleados (

id INT PRIMARY KEY,

nombre VARCHAR(50),

salario DECIMAL(10, 2)

);

```

- \*\*ALTER\*\*: Modifica la estructura de una tabla existente (agregar, eliminar o modificar columnas).

```sql

ALTER TABLE empleados

ADD fecha\_contratacion DATE;

```

- \*\*DROP\*\*: Elimina tablas, vistas, índices, etc., de la base de datos.

```sql

DROP TABLE empleados;

```

- \*\*TRUNCATE\*\*: Elimina todos los registros de una tabla pero mantiene la estructura de la tabla.

```sql

TRUNCATE TABLE empleados;

```

- \*\*RENAME\*\*: Cambia el nombre de una tabla o columna.

```sql

RENAME TABLE empleados TO trabajadores;

```

\*\*5.3.2. Comandos DML (Data Manipulation Language)\*\*

Estos comandos se utilizan para manipular los datos dentro de las tablas.

- \*\*SELECT\*\*: Recupera datos de una o más tablas.

```sql

SELECT nombre, salario

FROM empleados

WHERE salario > 50000;

```

- \*\*INSERT INTO\*\*: Inserta nuevos registros en una tabla.

```sql

INSERT INTO empleados (id, nombre, salario)

VALUES (1, 'Ana', 60000);

```

- \*\*UPDATE\*\*: Modifica los datos existentes en una tabla.

```sql

UPDATE empleados

SET salario = 65000

WHERE id = 1;

```

- \*\*DELETE\*\*: Elimina registros de una tabla.

```sql

DELETE FROM empleados

WHERE id = 1;

```

\*\*5.4. Tablas, filas y columnas\*\*

- \*\*Tablas\*\*: Son la estructura básica en una base de datos relacional donde se almacenan los datos.

- \*\*Filas\*\*: Cada fila en una tabla representa un registro único.

- \*\*Columnas\*\*: Cada columna en una tabla representa un atributo de los datos almacenados.

\*\*5.5. Tipos de datos de las columnas\*\*

Los tipos de datos especifican qué tipo de datos pueden ser almacenados en cada columna. Algunos tipos de datos comunes son:

- \*\*INT\*\*: Enteros.

- \*\*VARCHAR(n)\*\*: Cadenas de texto con longitud variable.

- \*\*DATE\*\*: Fechas.

- \*\*DECIMAL(p, s)\*\*: Números decimales con una precisión y escala específicas.

### Preguntas para Retroalimentar

1. \*\*¿Qué comando SQL se usa para eliminar una tabla completa de una base de datos?\*\*

\*Ejemplo:\* ¿Qué SQL usarías para eliminar la tabla `empleados`?

2. \*\*¿Cuál es la diferencia entre los comandos `INSERT INTO` y `UPDATE` en SQL?\*\*

\*Ejemplo:\* ¿Cómo se utilizarían estos comandos para agregar un nuevo empleado y para cambiar el salario de un empleado existente?

3. \*\*¿Qué tipo de datos usarías para almacenar una dirección de correo electrónico en una base de datos?\*\*

\*Ejemplo:\* ¿Cómo definirías una columna para almacenar direcciones de correo electrónico en una tabla de usuarios?

4. \*\*¿Cómo se crea una nueva columna en una tabla existente usando SQL?\*\*

\*Ejemplo:\* ¿Qué SQL usarías para agregar una columna de fecha de nacimiento a una tabla de empleados?

5. \*\*¿Qué comando SQL se utilizaría para recuperar todos los registros de una tabla donde el salario es mayor a 50,000?\*\*

\*Ejemplo:\* ¿Qué SQL usarías para seleccionar todos los empleados con un salario mayor a 50,000?

Respuestas

1. Se usa el comando DROP  
   ejemplo DROP TABLE empleados
2. La diferencia es que inser into nos sirve para insertar registros y update modifica un registro existente  
   ejemplo:  
   insert into empleados (id,nombre,edad) values (1,’juan’,15);  
   update empleados  
   set nombre = ‘pedro’  
   where id = 1;
3. Usario tipo de dato varchar y lo definiría asi  
   email varchar(20),
4. Crearía la nueva columan con el comando con el comando alter  
   ejemplo  
   alter table empleados  
   add fecha\_nacimiento date;
5. Select \*  
   from empleados  
   where salario > 50000;

**Vamos a profundizar en la \*\*Clase 6: Consultas SQL Básicas\*\*.**

### 6. Consultas SQL Básicas

\*\*6.1. WHERE\*\*

El comando `WHERE` se usa para filtrar registros que cumplen con ciertas condiciones. Permite seleccionar solo aquellos registros que cumplen con una condición específica.

\*\*Ejemplo:\*\*

Supongamos que tenemos una tabla llamada `empleados` y queremos seleccionar solo aquellos empleados cuyo salario es mayor a 50,000.

```sql

SELECT nombre, salario

FROM empleados

WHERE salario > 50000;

```

\*\*6.2. GROUP BY\*\*

El comando `GROUP BY` se usa para agrupar filas que tienen valores idénticos en columnas especificadas. Es útil para realizar cálculos agregados en grupos de datos.

\*\*Ejemplo:\*\*

Si queremos contar cuántos empleados hay en cada departamento, usaríamos:

```sql

SELECT departamento, COUNT(\*)

FROM empleados

GROUP BY departamento;

```

\*\*6.3. GROUP BY + HAVING\*\*

El comando `HAVING` se utiliza para filtrar grupos de datos después de aplicar `GROUP BY`. Es similar al `WHERE`, pero se usa para condiciones en los resultados agregados.

\*\*Ejemplo:\*\*

Para encontrar los departamentos con más de 10 empleados, usaríamos:

```sql

SELECT departamento, COUNT(\*)

FROM empleados

GROUP BY departamento

HAVING COUNT(\*) > 10;

```

\*\*6.4. ORDER BY\*\*

El comando `ORDER BY` se usa para ordenar los resultados de una consulta en un orden específico. Puede ser en orden ascendente (`ASC`) o descendente (`DESC`).

\*\*Ejemplo:\*\*

Para ordenar los empleados por salario en orden descendente:

```sql

SELECT nombre, salario

FROM empleados

ORDER BY salario DESC;

```

### Preguntas para Retroalimentar

1. \*\*¿Cómo usarías el comando `WHERE` para seleccionar empleados con un salario menor a 30,000?\*\*

\*Ejemplo:\* ¿Qué SQL escribirías para encontrar empleados con salarios por debajo de 30,000?

2. \*\*¿Qué hace el comando `GROUP BY` en una consulta SQL y para qué se usa típicamente?\*\*

\*Ejemplo:\* ¿Cómo agruparías los empleados por su departamento?

3. \*\*¿Cómo filtrarías los resultados de una consulta que agrupa datos con `GROUP BY` para mostrar solo aquellos grupos con más de 5 registros?\*\*

\*Ejemplo:\* ¿Qué SQL usarías para encontrar departamentos con más de 5 empleados?

4. \*\*¿Cómo ordenarías los resultados de una consulta para mostrar los empleados con los salarios más altos primero?\*\*

\*Ejemplo:\* ¿Qué SQL usarías para ordenar empleados por salario en orden descendente?

5. \*\*¿Cuál es la diferencia entre `WHERE` y `HAVING` en SQL y cuándo usarías cada uno?\*\*

\*Ejemplo:\* ¿Cómo usarías `WHERE` para filtrar antes de agrupar y `HAVING` para filtrar después de agrupar?

Respuestas

1. SELECT \*  
   FORM empleados  
   WHERE salario < 30000
2. El comando GROUP BY agrupa los datos con similar contenido para su análisis  
   ejemplo  
   SELECT NOMBRE\_EMPLEADO  
   FROM EMPLEADOS  
   GROUP BY DEPARTAMENTOS
3. Usaría el método HAVING  
   EJEMPLO  
   SELECT NOMBRE\_LIBROS  
   FROM BIBLIOTECAS  
   GROUP BY ID\_LIBROS  
   HAVING COUNT(\*) > 5;
4. USARIA EL COMANDO ORDER BY Y DESC

SELECT \*  
FROM EMPLEADOS  
ORDER BY SALARIO DESC

1. LA DIFERENCIA ES QUE WHERE SE PUEDE USAR PARA FILTRAR CUALQUIER CONSULTA Y HAVING SOLO SE USA DESPUES DE AGRUPAR  
   EJEMPLOS WHERE  
   SELECT \*  
   FORM empleados  
   WHERE salario < 30000

EJEMPLO HAVING

SELECT NOMBRE\_LIBROS  
FROM BIBLIOTECAS  
GROUP BY ID\_LIBROS  
HAVING COUNT(\*) > 5;  
  
¡Muy bien! Aquí está la retroalimentación sobre tus respuestas:

1. \*\*Correcto\*\*. El comando `WHERE` filtra los registros de acuerdo con la condición establecida. Tu ejemplo está correcto, pero ten en cuenta que el comando `FROM` está escrito como `FORM` en tu respuesta. Debería ser:

```sql

SELECT \*

FROM empleados

WHERE salario < 30000;

```

2. \*\*Correcto en esencia\*\*. `GROUP BY` agrupa los datos con valores similares en una o más columnas para realizar cálculos agregados. Tu ejemplo de agrupación está en la dirección correcta, pero la consulta debería incluir una función de agregación para mostrar la utilidad completa del `GROUP BY`. Ejemplo corregido:

```sql

SELECT departamento, COUNT(\*)

FROM empleados

GROUP BY departamento;

```

3. \*\*Correcto\*\*. Usar `HAVING` para filtrar después de agrupar es el enfoque correcto. Sin embargo, el ejemplo debería tener sentido con `COUNT(\*)` en una consulta que agrupe por una columna. Ejemplo corregido:

```sql

SELECT departamento

FROM empleados

GROUP BY departamento

HAVING COUNT(\*) > 5;

```

4. \*\*Correcto\*\*. Usar `ORDER BY` en combinación con `DESC` es el método adecuado para ordenar los resultados. Tu consulta está correcta, pero asegúrate de que el formato esté bien escrito:

```sql

SELECT \*

FROM empleados

ORDER BY salario DESC;

```

5. \*\*Correcto\*\*. La diferencia entre `WHERE` y `HAVING` es clara. `WHERE` filtra antes de agrupar, mientras que `HAVING` filtra después de agrupar. Tus ejemplos son adecuados para ilustrar la diferencia:

- \*\*WHERE\*\*:

```sql

SELECT \*

FROM empleados

WHERE salario < 30000;

```

- \*\*HAVING\*\*:

```sql

SELECT departamento

FROM empleados

GROUP BY departamento

HAVING COUNT(\*) > 5;

```

¡Estás manejando bien los conceptos de consultas SQL básicas! Si necesitas más detalles o practicar otros temas, avísame.  
  
**### Clase 7: Consultas SQL Avanzadas**

Las consultas SQL avanzadas permiten realizar operaciones más complejas sobre los datos en una base de datos. Aquí se detallan los principales tipos de uniones y sus aplicaciones:

#### 7.1. INNER JOIN

El `INNER JOIN` se utiliza para combinar filas de dos o más tablas basadas en una condición de coincidencia. Solo se devuelven las filas que tienen una coincidencia en ambas tablas.

\*\*Ejemplo:\*\*

Supongamos que tenemos las siguientes dos tablas:

\*\*Empleados\*\*

| id\_empleado | nombre | id\_departamento |

|-------------|-----------|-----------------|

| 1 | Juan | 101 |

| 2 | María | 102 |

| 3 | Pedro | 101 |

\*\*Departamentos\*\*

| id\_departamento | nombre\_departamento |

|-----------------|----------------------|

| 101 | Ventas |

| 102 | Marketing |

| 103 | IT |

Para obtener los nombres de los empleados junto con el nombre del departamento al que pertenecen:

```sql

SELECT empleados.nombre, departamentos.nombre\_departamento

FROM empleados

INNER JOIN departamentos

ON empleados.id\_departamento = departamentos.id\_departamento;

```

#### 7.2. LEFT JOIN

El `LEFT JOIN` devuelve todas las filas de la tabla de la izquierda (la primera tabla mencionada) y las filas coincidentes de la tabla de la derecha (la segunda tabla mencionada). Si no hay coincidencia, el resultado es NULL para las columnas de la tabla de la derecha.

\*\*Ejemplo:\*\*

Para obtener todos los empleados y, si existen, los nombres de sus departamentos:

```sql

SELECT empleados.nombre, departamentos.nombre\_departamento

FROM empleados

LEFT JOIN departamentos

ON empleados.id\_departamento = departamentos.id\_departamento;

```

#### 7.3. RIGHT JOIN

El `RIGHT JOIN` devuelve todas las filas de la tabla de la derecha y las filas coincidentes de la tabla de la izquierda. Si no hay coincidencia, el resultado es NULL para las columnas de la tabla de la izquierda.

\*\*Ejemplo:\*\*

Para obtener todos los departamentos y, si existen, los nombres de los empleados en esos departamentos:

```sql

SELECT empleados.nombre, departamentos.nombre\_departamento

FROM empleados

RIGHT JOIN departamentos

ON empleados.id\_departamento = departamentos.id\_departamento;

```

#### 7.4. FULL OUTER JOIN

El `FULL OUTER JOIN` devuelve todas las filas cuando hay una coincidencia en una de las tablas. Devuelve NULL para las filas que no tienen coincidencia en la otra tabla.

\*\*Ejemplo:\*\*

Para obtener todos los empleados y todos los departamentos, mostrando NULL donde no haya coincidencia:

```sql

SELECT empleados.nombre, departamentos.nombre\_departamento

FROM empleados

FULL OUTER JOIN departamentos

ON empleados.id\_departamento = departamentos.id\_departamento;

```

### Preguntas para Retroalimentación

1. \*\*¿Qué es el `INNER JOIN` y qué tipo de datos devuelve en la consulta?\*\*

- \*\*Respuesta esperada:\*\* Devuelve solo las filas que tienen coincidencias en ambas tablas involucradas en la unión.

2. \*\*¿Cómo se usa `LEFT JOIN` y qué sucede si no hay coincidencias en la tabla de la derecha?\*\*

- \*\*Respuesta esperada:\*\* Devuelve todas las filas de la tabla de la izquierda y las filas coincidentes de la tabla de la derecha. Si no hay coincidencias, las columnas de la tabla de la derecha serán NULL.

3. \*\*¿Cuál es la diferencia principal entre `RIGHT JOIN` y `LEFT JOIN`?\*\*

- \*\*Respuesta esperada:\*\* `RIGHT JOIN` devuelve todas las filas de la tabla de la derecha y las coincidencias de la tabla de la izquierda, mientras que `LEFT JOIN` devuelve todas las filas de la tabla de la izquierda y las coincidencias de la tabla de la derecha.

4. \*\*¿Qué resultado se obtiene con un `FULL OUTER JOIN` y cuándo es útil usarlo?\*\*

- \*\*Respuesta esperada:\*\* Devuelve todas las filas cuando hay una coincidencia en una de las tablas. Es útil cuando se necesita obtener todos los datos de ambas tablas, incluso si no hay coincidencias.

5. \*\*Dado el siguiente conjunto de tablas y datos, ¿cómo escribirías una consulta para obtener todos los productos y las ventas realizadas de esos productos, incluyendo aquellos productos que no han sido vendidos?\*\*

- \*\*Tablas:\*\*

- \*\*Productos:\*\* id\_producto, nombre\_producto

- \*\*Ventas:\*\* id\_venta, id\_producto, cantidad

- \*\*Respuesta esperada:\*\* Usar un `LEFT JOIN` para incluir todos los productos y las ventas correspondientes, incluyendo productos sin ventas:

```sql

SELECT productos.nombre\_producto, ventas.cantidad

FROM productos

LEFT JOIN ventas

ON productos.id\_producto = ventas.id\_producto;

```

**### Clase 8: Subconsultas en SQL**

Las subconsultas, también conocidas como consultas anidadas, son consultas SQL que se utilizan dentro de otras consultas. Son útiles para realizar consultas más complejas que dependen de los resultados de otras consultas. Aquí se describen los tres tipos principales de subconsultas:

#### 8.1. Subconsulta Escalar

Una subconsulta escalar devuelve un solo valor (una sola fila y una sola columna). Se usa generalmente en una cláusula `WHERE` o en una expresión.

\*\*Ejemplo:\*\*

Supongamos que tenemos una tabla de empleados con salarios y una tabla de departamentos con un presupuesto máximo por departamento. Queremos encontrar los nombres de los empleados cuyo salario es menor que el presupuesto máximo del departamento al que pertenecen.

```sql

SELECT nombre

FROM empleados

WHERE salario < (

SELECT MAX(presupuesto)

FROM departamentos

);

```

#### 8.2. Subconsulta de Múltiples Filas

Una subconsulta de múltiples filas devuelve más de una fila de resultados. Se utiliza comúnmente con operadores como `IN`, `ANY`, o `ALL`.

\*\*Ejemplo:\*\*

Para encontrar todos los empleados que están en los departamentos con un presupuesto superior a 50000:

```sql

SELECT nombre

FROM empleados

WHERE id\_departamento IN (

SELECT id\_departamento

FROM departamentos

WHERE presupuesto > 50000

);

```

#### 8.3. Subconsulta Correlacionada

Una subconsulta correlacionada es una subconsulta que se refiere a columnas de la consulta externa. A diferencia de una subconsulta normal, una subconsulta correlacionada se ejecuta una vez por cada fila procesada por la consulta externa.

\*\*Ejemplo:\*\*

Para encontrar los empleados que tienen el salario mayor que el salario promedio en su propio departamento:

```sql

SELECT nombre

FROM empleados e1

WHERE salario > (

SELECT AVG(salario)

FROM empleados e2

WHERE e1.id\_departamento = e2.id\_departamento

);

```

### Preguntas para Retroalimentación

1. \*\*¿Qué es una subconsulta escalar y en qué situaciones podrías usarla?\*\*

2. \*\*¿Cómo se utiliza una subconsulta de múltiples filas y qué operadores se pueden emplear con ella?\*\*

3. \*\*¿Qué diferencia hay entre una subconsulta normal y una subconsulta correlacionada?\*\*

4. \*\*Dado el siguiente ejemplo de tabla de empleados y departamentos, ¿cómo escribirías una consulta para encontrar los empleados que ganan más que el salario promedio de su departamento utilizando una subconsulta correlacionada?\*\*

5. \*\*En una subconsulta escalar, ¿qué sucede si la subconsulta devuelve más de un valor? ¿Cómo se maneja esta situación en una consulta SQL?\*\*

RESPUESTAS

1. ES LA SUBCONSULTA QUE ME DA UN SOLO DATO COMO RESULTADO Y ES UTIL CUANDO QUIERO OBTENER UN DATO ESPECIFICO PARA COMPLEMENTAR MI PRIMERA CONSULTA
2. Se usa integrando los comandos in, any o all, para obtener multiples registros en mi consulta
3. La diferencia es que la consultacorrelacionada se refiera a las columnas de la consulta externa, mientas que la normal se refiera a las consultas de la misma
4. Select nombre\_empleados  
   from empleados e1  
   where salario>(  
    select avg>(salario)  
    from empleados e2  
    where e1.id\_departamento = e2.id\_departamentos

**### Clase 9: Índices en SQL**

#### 1. \*\*Introducción a los Índices\*\*

Un índice en SQL es una estructura que mejora la velocidad de las consultas al permitir un acceso más rápido a los datos. Los índices se crean en una o más columnas de una tabla y permiten que el sistema de gestión de bases de datos (SGBD) busque y ordene datos de manera más eficiente.

#### 2. \*\*Tipos de Índices\*\*

- \*\*Índice Único (Unique Index):\*\* Garantiza que todos los valores en la columna o columnas indexadas sean únicos. Se utiliza para mantener la integridad de los datos.

\*\*Ejemplo:\*\*

```sql

CREATE UNIQUE INDEX idx\_email ON usuarios (email);

```

- \*\*Índice Compuesto (Composite Index):\*\* Un índice creado en más de una columna. Es útil para consultas que filtran por múltiples columnas.

\*\*Ejemplo:\*\*

```sql

CREATE INDEX idx\_nombre\_fecha ON ventas (nombre\_cliente, fecha\_venta);

```

- \*\*Índice Clustered (Clustered Index):\*\* El índice determina el orden físico de los datos en la tabla. Solo puede haber un índice clustered por tabla.

\*\*Ejemplo:\*\*

```sql

CREATE CLUSTERED INDEX idx\_id ON productos (id\_producto);

```

- \*\*Índice No Clustered (Non-Clustered Index):\*\* Crea una estructura separada del almacenamiento de la tabla que apunta a las ubicaciones de los datos. Puede haber múltiples índices no clustered en una tabla.

\*\*Ejemplo:\*\*

```sql

CREATE NONCLUSTERED INDEX idx\_precio ON productos (precio);

```

- \*\*Índice de Texto Completo (Full-Text Index):\*\* Permite realizar búsquedas de texto completas y más avanzadas en columnas de texto.

\*\*Ejemplo:\*\*

```sql

CREATE FULLTEXT INDEX idx\_texto ON documentos (contenido);

```

#### 3. \*\*Beneficios de los Índices\*\*

- \*\*Rendimiento Mejorado:\*\* Aceleran la recuperación de datos y mejoran el rendimiento de las consultas SELECT.

- \*\*Eficiencia en Búsquedas y Ordenamientos:\*\* Facilitan búsquedas rápidas y ordenamientos en grandes volúmenes de datos.

- \*\*Reducción de Tiempo de Respuesta:\*\* Reducen el tiempo necesario para localizar registros específicos.

#### 4. \*\*Consideraciones al Usar Índices\*\*

- \*\*Impacto en el Rendimiento de Inserciones/Actualizaciones:\*\* La creación y el mantenimiento de índices pueden ralentizar las operaciones de inserción, actualización y eliminación.

- \*\*Espacio en Disco:\*\* Los índices ocupan espacio adicional en disco.

- \*\*Selección de Columnas:\*\* Es importante seleccionar las columnas correctas para indexar, basándose en las consultas más comunes y los patrones de acceso a los datos.

### Preguntas para Retroalimentación

1. \*\*¿Qué es un índice clustered y cómo afecta al orden físico de los datos en una tabla?\*\*

2. \*\*¿Cuándo deberías utilizar un índice compuesto en lugar de un índice simple? Proporciona un ejemplo.\*\*

3. \*\*¿Cuáles son las diferencias clave entre un índice clustered y un índice no clustered?\*\*

4. \*\*¿Cómo pueden los índices de texto completo mejorar la búsqueda en columnas de texto?\*\*

1. \*\*¿Qué consideraciones deben tenerse en cuenta al decidir crear un índice en una tabla?\*\*

**Preguntas Adicionales para Retroalimentación:**

1. **¿Cómo se puede combinar una subconsulta escalar con una subconsulta de múltiples filas en una sola consulta? Proporciona un ejemplo.**
2. **¿Qué sucede si una subconsulta de múltiples filas devuelve un conjunto de resultados vacío? ¿Cómo afectaría esto a la consulta externa que la utiliza?**
3. **¿Puedes dar un ejemplo de cómo usar la subconsulta correlacionada con un JOIN en lugar de una subconsulta correlacionada en la cláusula WHERE?**
4. **¿Cuál es la diferencia en términos de rendimiento entre una subconsulta normal y una subconsulta correlacionada?**
5. **En el caso de una subconsulta escalar, ¿qué errores comunes se pueden cometer si la subconsulta devuelve más de un valor?**

**Respuestas:**

1. **Es un índice que sirve para organizar la tabla como se indique en la sentencia**
2. **Debería usar un índice compuesto compuesto en vez de un simple cuando necesite organizar varias columnas  
   ejemplo  
   necesito organizar una base de datos por orden alfabetico en los nombres de las columnas y además ordenar si el libro esta prestado o no**
3. **La diferencia principal es que solo puede haber un índice clustered y muchos no clustered**
4. **Siendo mas específicos en el tipo de texto que quiero buscar**
5. **Que son un poco complicados de amtener  
   que ocupan espacio en el disco  
   y que se debe ser exacto en las columnas a los que se les van a aplicar**
6. SELECT e.nombre  
   FROM empleados e  
   WHERE e.salario > (  
    SELECT AVG(salario)  
    FROM empleados e2  
    WHERE e.id\_departamento = e2.id\_departamento  
   )  
   AND e.id\_departamento IN (  
    SELECT id\_departamento  
    FROM departamentos  
    WHERE AVG(  
    SELECT salario  
    FROM empleados e3  
    WHERE e3.id\_departamento = departamentos.id\_departamento  
    ) > (  
    SELECT AVG(salario)  
    FROM empleados  
    )  
   );
7. Devuelve un valor null, y como tal no se ejecuta la consulta general
8. Usando where  
   SELECT e.nombre

FROM empleados e

WHERE e.id\_departamento = (

SELECT id\_departamento

FROM empleados

WHERE id\_empleado = 10

);  
usando join  
SELECT e.nombre

FROM empleados e

JOIN empleados e2 ON e.id\_departamento = e2.id\_departamento

WHERE e2.id\_empleado = 10;

1. La subconsulta correlacionada al ejecutarse fila por fila es mucho mas pesada que la consulta normal que analisa todas las filas
2. Puede generar errores en el plantemaiento de la consulta ya que espera un solo resultado, por lo general esto se da por errores de sintaxis

### Clase 10: Restricciones SQL

Las restricciones en SQL son reglas que se aplican a las columnas de una tabla para asegurar la integridad y exactitud de los datos. Estas restricciones ayudan a mantener la calidad de los datos al limitar los tipos de datos que se pueden almacenar, así como las relaciones entre las tablas. Aquí se detallan algunas de las principales restricciones SQL:

#### 1. \*\*Restricción PRIMARY KEY\*\*

- \*\*Descripción:\*\* Asegura que cada fila en una tabla sea única y no tenga valores NULL en la columna (o columnas) que se define como clave primaria.

- \*\*Uso:\*\* Define la clave primaria de una tabla.

- \*\*Sintaxis:\*\*

```sql

CREATE TABLE empleados (

id\_empleado INT PRIMARY KEY,

nombre VARCHAR(100)

);

```

#### 2. \*\*Restricción FOREIGN KEY\*\*

- \*\*Descripción:\*\* Establece una relación entre dos tablas. La columna (o columnas) con la clave foránea en la tabla secundaria debe coincidir con la clave primaria en la tabla principal.

- \*\*Uso:\*\* Mantiene la integridad referencial entre tablas.

- \*\*Sintaxis:\*\*

```sql

CREATE TABLE departamentos (

id\_departamento INT PRIMARY KEY,

nombre VARCHAR(100)

);

CREATE TABLE empleados (

id\_empleado INT PRIMARY KEY,

nombre VARCHAR(100),

id\_departamento INT,

FOREIGN KEY (id\_departamento) REFERENCES departamentos(id\_departamento)

);

```

#### 3. \*\*Restricción UNIQUE\*\*

- \*\*Descripción:\*\* Garantiza que todos los valores en una columna (o combinación de columnas) sean únicos en toda la tabla. Permite valores NULL, pero solo uno por columna.

- \*\*Uso:\*\* Asegura que no haya duplicados en una columna o conjunto de columnas.

- \*\*Sintaxis:\*\*

```sql

CREATE TABLE productos (

id\_producto INT PRIMARY KEY,

codigo\_producto VARCHAR(50) UNIQUE

);

```

#### 4. \*\*Restricción NOT NULL\*\*

- \*\*Descripción:\*\* Asegura que una columna no contenga valores NULL.

- \*\*Uso:\*\* Garantiza que se ingresen datos en una columna específica.

- \*\*Sintaxis:\*\*

```sql

CREATE TABLE clientes (

id\_cliente INT PRIMARY KEY,

nombre VARCHAR(100) NOT NULL

);

```

#### 5. \*\*Restricción CHECK\*\*

- \*\*Descripción:\*\* Permite definir una condición que debe cumplirse para que los valores sean aceptados en una columna.

- \*\*Uso:\*\* Valida los datos ingresados para asegurarse de que cumplen con ciertos criterios.

- \*\*Sintaxis:\*\*

```sql

CREATE TABLE productos (

id\_producto INT PRIMARY KEY,

precio DECIMAL(10, 2),

CHECK (precio > 0)

);

```

#### 6. \*\*Restricción DEFAULT\*\*

- \*\*Descripción:\*\* Asigna un valor predeterminado a una columna si no se especifica ningún valor al insertar una nueva fila.

- \*\*Uso:\*\* Facilita la asignación automática de valores cuando no se proporciona uno explícitamente.

- \*\*Sintaxis:\*\*

```sql

CREATE TABLE empleados (

id\_empleado INT PRIMARY KEY,

nombre VARCHAR(100),

fecha\_ingreso DATE DEFAULT CURRENT\_DATE

);

```

### Preguntas para Retroalimentación

1. \*\*¿Cuál es la principal función de una restricción PRIMARY KEY y cómo se define en una tabla?\*\*

2. \*\*¿Cómo ayuda la restricción FOREIGN KEY a mantener la integridad referencial entre dos tablas?\*\*

3. \*\*¿Qué diferencia hay entre las restricciones UNIQUE y NOT NULL en términos de la capacidad de aceptar valores duplicados y NULL?\*\*

4. \*\*¿Cómo se utiliza la restricción CHECK para validar los datos en una columna y qué tipo de condición puede imponer?\*\*

5. \*\*¿Cuál es el propósito de la restricción DEFAULT y cómo puede ser útil al insertar nuevos registros en una tabla?\*\*

6.  **¿Qué impacto tiene un índice clustered en el rendimiento de las consultas en comparación con un índice no clustered?**

 **¿Por qué es importante considerar el uso de índices compuestos en consultas que involucran múltiples columnas?**

 **¿Cuándo se recomienda utilizar un índice de texto completo y qué ventajas ofrece sobre otros tipos de índices?**

 **¿Cuáles son las implicaciones del uso de índices en términos de espacio en disco y rendimiento de las operaciones de escritura?**

 **¿Cómo puede una subconsulta escalar afectar el rendimiento de una consulta general si no devuelve el valor esperado?**

**Respuestas**

1. **Que cada registro sea único y tenga su identificación que lo hace único**
2. **Obligando a que las llaves foráneas y la llave primaria con la que se elnaza sean iguales y no se generan cunficiones en caso de necesitar hacer consultas**
3. **Unique solo acepta un valos null por columna mientras que not null acepta infinidad**
4. **Se utiliza en la medida de que deseemos que un registro no supere cierto rango, y se usa con los operadores de comparación para asignar la condición**
5. **La función es facilitar el diligenciamiento de la base de datos y es útil para prever errores de diligenciamiento**
6. **El impacto en mi opinión seria que puede reducir el uso del modelo asendente o desendente en la tabla**
7. **Por que son mas efectivos para trabajar con multiples columnas**
8. **En caso de que quiera buscar palabras especificas, iniciales o letras, y es mejor que otros índices pues me da mas especificidad en mis búsquedas**
9. **Las implicaciones son el aumento en el espacio de disco y hace la base de datos mas pesada lo cual puede volver lentas las consultas**
10. **Afecta en el sentido de que no obtendría el dato esperado**

**Aquí tienes la revisión de tus respuestas a las preguntas sobre restricciones SQL:**

**1. \*\*¿Cuál es la principal función de una restricción PRIMARY KEY y cómo se define en una tabla?\*\***

**- \*\*Respuesta correcta:\*\* La restricción PRIMARY KEY asegura que cada registro en una tabla sea único y no contenga valores NULL en las columnas que la definen. Esto significa que cada fila puede ser identificada de manera única por la clave primaria. Se define utilizando la sintaxis `PRIMARY KEY` al crear una tabla.**

**2. \*\*¿Cómo ayuda la restricción FOREIGN KEY a mantener la integridad referencial entre dos tablas?\*\***

**- \*\*Respuesta correcta:\*\* La restricción FOREIGN KEY asegura que los valores en una columna (o combinación de columnas) en una tabla secundaria coincidan con los valores en la clave primaria de otra tabla (la tabla principal). Esto mantiene la integridad referencial y evita que se inserten datos en la tabla secundaria que no tengan correspondencia en la tabla principal.**

**3. \*\*¿Qué diferencia hay entre las restricciones UNIQUE y NOT NULL en términos de la capacidad de aceptar valores duplicados y NULL?\*\***

**- \*\*Respuesta correcta:\*\* La restricción UNIQUE garantiza que todos los valores en una columna (o combinación de columnas) sean únicos y permite solo un valor NULL por columna. La restricción NOT NULL asegura que una columna no pueda tener valores NULL en ninguna fila, forzando la presencia de un valor en esa columna.**

**4. \*\*¿Cómo se utiliza la restricción CHECK para validar los datos en una columna y qué tipo de condición puede imponer?\*\***

**- \*\*Respuesta correcta:\*\* La restricción CHECK se usa para imponer una condición que los datos deben cumplir para ser aceptados en una columna. Puedes usar operadores de comparación para definir reglas, como asegurarte de que los valores estén dentro de un rango específico o cumplan ciertos criterios.**

**5. \*\*¿Cuál es el propósito de la restricción DEFAULT y cómo puede ser útil al insertar nuevos registros en una tabla?\*\***

**- \*\*Respuesta correcta:\*\* La restricción DEFAULT asigna automáticamente un valor predeterminado a una columna si no se especifica un valor al insertar un nuevo registro. Esto ayuda a evitar errores al omitir valores y proporciona un valor por defecto que se usa cuando no se ingresa ningún dato explícito.**

**Tus respuestas son en su mayoría correctas, pero hay algunos matices y detalles adicionales que podrías considerar:**

**- En la respuesta 2, la idea de "no generar conflictos en caso de hacer consultas" no es precisa; la FOREIGN KEY evita la inserción de datos que no correspondan a las claves primarias existentes, garantizando la integridad referencial.**

**- En la respuesta 3, es importante destacar que UNIQUE permite un solo valor NULL por columna, mientras que NOT NULL no permite valores NULL en absoluto.**

**- En la respuesta 4, especificar que la restricción CHECK se usa para imponer restricciones en los valores y condiciones de datos sería útil.**

**- En la respuesta 5, la restricción DEFAULT efectivamente facilita el ingreso de datos al proporcionar un valor automático en caso de omisión, pero la explicación podría ser un poco más específica.**

**Si necesitas más detalles o aclaraciones sobre algún tema, ¡házmelo saber!**

**### Clase 11: Transacciones en SQL**

**Las transacciones en SQL son un conjunto de operaciones que se ejecutan como una única unidad de trabajo. El objetivo principal de las transacciones es garantizar la integridad y consistencia de la base de datos en situaciones donde múltiples operaciones deben ejecutarse juntas.**

**#### Conceptos Clave de las Transacciones en SQL**

**1. \*\*Definición de Transacción:\*\***

**Una transacción es una secuencia de operaciones SQL que se agrupan para ser ejecutadas de manera atómica. Esto significa que todas las operaciones deben completarse exitosamente para que la transacción se confirme; de lo contrario, todas las operaciones se deshacen.**

**2. \*\*Propiedades ACID:\*\***

**Las transacciones deben cumplir con las propiedades ACID para garantizar la integridad de los datos:**

**- \*\*Atomicidad:\*\* La transacción se realiza en su totalidad o no se realiza en absoluto. Si una parte de la transacción falla, todas las operaciones se revierten.**

**- \*\*Consistencia:\*\* La base de datos pasa de un estado consistente a otro estado consistente.**

**- \*\*Aislamiento:\*\* Las operaciones de una transacción son invisibles para otras transacciones hasta que se completan.**

**- \*\*Durabilidad:\*\* Una vez que una transacción se ha confirmado, sus efectos son permanentes, incluso en caso de fallos del sistema.**

**3. \*\*Comandos de Control de Transacciones:\*\***

**- \*\*BEGIN TRANSACTION (o START TRANSACTION):\*\* Inicia una nueva transacción.**

**- \*\*COMMIT:\*\* Guarda todas las operaciones realizadas durante la transacción de manera permanente en la base de datos.**

**- \*\*ROLLBACK:\*\* Reversa todas las operaciones realizadas durante la transacción en caso de error o si la transacción no puede completarse.**

**4. \*\*Ejemplo Básico:\*\***

**```sql**

**BEGIN TRANSACTION;**

**INSERT INTO cuentas (cuenta\_id, saldo) VALUES (1, 1000);**

**UPDATE cuentas SET saldo = saldo - 100 WHERE cuenta\_id = 1;**

**UPDATE cuentas SET saldo = saldo + 100 WHERE cuenta\_id = 2;**

**COMMIT;**

**```**

**En este ejemplo, se realiza una transferencia de dinero entre dos cuentas. Si todas las operaciones se ejecutan exitosamente, el cambio se confirma. Si ocurre algún error, el `ROLLBACK` revertirá todos los cambios.**

**5. \*\*Manejo de Errores y Rollbacks:\*\***

**Las transacciones permiten manejar errores de manera efectiva. Si una operación dentro de la transacción falla, se puede usar `ROLLBACK` para deshacer los cambios y restaurar el estado anterior.**

**6. \*\*Transacciones Anidadas:\*\***

**Algunas bases de datos soportan transacciones anidadas, donde una transacción puede contener otras transacciones. Sin embargo, la mayoría de las bases de datos tratan las transacciones anidadas como una única transacción.**

**### Preguntas para Retroalimentación**

**1. \*\*¿Qué es una transacción en SQL y por qué es importante en la gestión de bases de datos?\*\***

**2. \*\*¿Cuáles son las propiedades ACID que deben cumplir las transacciones? Explica cada una de ellas.\*\***

**3. \*\*¿Qué comando se usa para iniciar una transacción en SQL y qué comandos se utilizan para confirmar o revertir una transacción?\*\***

**4. \*\*¿Cómo se maneja un error dentro de una transacción y qué efecto tiene el comando ROLLBACK?\*\***

**5. \*\*¿Qué es una transacción anidada y cómo se diferencia de una transacción estándar?\*\***

Respuestas

1. es una secuancia de operaciones que se agrupan para ser tratadas de manera atómica, esto significa que todas las transacciones deben completarce para que la oeracion sea exitosa
2. atomicidad: todas las transacciones deben completarse para que la operación se ejecute completa **-** consistencia: la base de datos pasa de un estado consiente a otro estado cinsiente **-** aislamiento: cada operación es independiente de la otra  
   - durabilidad: luego de ejecutada la operación debe ser permanete asi el sitema se caiga
3. BEGIND PARA INICIAR  
   COMMIT PARA GUARDAR  
   ROLLBACK PARA REVERTIR
4. EL ERROR DEVUELVE TODAS LAS OPERACIONES A SU ESTADO INICIAR Y EL ROLLBACK SE EJECUTA
5. ES UNA TRANSACCION QUE VIAJA COMO UNA SOLA, PERO COMO TAL ES UN GRUPO DE TRANSACCIONES

**### Clase 12: Bases de Datos Relacionales vs. Bases de Datos No Relacionales**

#### 1. Introducción a las Bases de Datos Relacionales y No Relacionales

Las bases de datos relacionales y no relacionales representan dos enfoques fundamentales para el almacenamiento y la gestión de datos. Cada tipo tiene su arquitectura, casos de uso, y ventajas según el contexto en el que se utilice.

#### 2. Bases de Datos Relacionales

Las bases de datos relacionales organizan los datos en tablas que están interconectadas a través de relaciones. Estas bases de datos siguen un modelo estructurado y son gestionadas mediante SQL (Structured Query Language). Las características principales incluyen:

- \*\*Estructura Tabular\*\*: Los datos se almacenan en tablas con filas y columnas.

- \*\*Relaciones\*\*: Se pueden establecer relaciones entre diferentes tablas utilizando claves primarias y foráneas.

- \*\*Consistencia y ACID\*\*: Aseguran la integridad de los datos mediante transacciones que siguen las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).

##### Ventajas:

- Fuerte consistencia y transacciones ACID.

- Ideal para aplicaciones que requieren datos estructurados y relaciones complejas.

##### Desventajas:

- Escalabilidad vertical limitada.

- Rigidez en la estructura de datos.

#### 3. Bases de Datos No Relacionales

Las bases de datos no relacionales, también conocidas como NoSQL, no siguen un esquema tabular fijo. Son flexibles en cuanto al esquema y están diseñadas para manejar grandes volúmenes de datos distribuidos. Hay varios tipos de bases de datos NoSQL, incluyendo:

- \*\*Bases de Datos de Documentos\*\*: Almacenan datos en formato JSON o BSON (ej., MongoDB).

- \*\*Bases de Datos de Claves y Valores\*\*: Almacenan pares clave-valor (ej., Redis).

- \*\*Bases de Datos de Columnas\*\*: Almacenan datos en columnas en lugar de filas (ej., Cassandra).

- \*\*Bases de Datos de Grafos\*\*: Almacenan datos en nodos y aristas, que representan entidades y sus relaciones (ej., Neo4j).

##### Ventajas:

- Escalabilidad horizontal y flexibilidad en el manejo de datos no estructurados.

- Mejor rendimiento para ciertos tipos de consultas y volúmenes de datos muy grandes.

##### Desventajas:

- Falta de soporte completo para transacciones ACID en algunos sistemas.

- Complejidad en la gestión de relaciones entre datos.

#### 4. Comparación entre Bases de Datos Relacionales y No Relacionales

- \*\*Estructura\*\*: Relacionales usan tablas; No relacionales pueden usar documentos, pares clave-valor, grafos, etc.

- \*\*Consistencia\*\*: Las relacionales suelen ser consistentes y ACID; las no relacionales pueden priorizar la disponibilidad y la partición.

- \*\*Escalabilidad\*\*: Las relacionales suelen escalar verticalmente; las no relacionales escalan horizontalmente.

- \*\*Casos de Uso\*\*: Las relacionales son ideales para sistemas transaccionales; las no relacionales son mejores para big data, análisis en tiempo real, y aplicaciones que manejan datos no estructurados.

### Preguntas para Retroalimentación

1. ¿Qué es una base de datos no relacional y cómo se diferencia de una base de datos relacional?

2. Menciona un tipo de base de datos NoSQL y describe brevemente su uso principal.

3. ¿Qué ventajas ofrece una base de datos relacional en términos de integridad y transacciones?

4. ¿Cuándo sería más apropiado utilizar una base de datos no relacional en lugar de una relacional?

5. Explica una situación en la que la escalabilidad horizontal de una base de datos no relacional sería crucial.

RESPUESTAS

1. Es una base que no sigue los lineamientos de tablas y relaciones, además no usa sql, y su diferencia principal es que la no relacional es efectiva para crecer orizontalmente, y la relacional en manera vertical
2. Json, se usa principalmente para manejo de documentos
3. Es consistente en el manejo de los datos y mantiene las transacciones con el modelo acid
4. Cuando se quieran manejar archivos de diferente contexto sin relación alguna
5. La creación de paginas de datos

**### Clase 13: Casos de Uso de las Bases Relacionales**

Las bases de datos relacionales (RDBMS) son ampliamente utilizadas en diversas industrias debido a su capacidad para manejar grandes volúmenes de datos de manera estructurada y eficiente. A continuación, se presentan algunos de los casos de uso más comunes donde las bases de datos relacionales son preferidas:

1. \*\*Sistemas de Gestión Empresarial (ERP)\*\*

- Las empresas utilizan sistemas ERP (Enterprise Resource Planning) para integrar y gestionar todas las funciones de la organización, como finanzas, recursos humanos, ventas, producción, etc. Las bases de datos relacionales son esenciales para almacenar y recuperar información de manera coherente y estructurada en estos sistemas.

2. \*\*Sistemas de Gestión de Clientes (CRM)\*\*

- Las plataformas de CRM (Customer Relationship Management) dependen de las bases de datos relacionales para organizar y gestionar información detallada sobre los clientes, incluidas sus interacciones, historial de compras, preferencias, etc.

3. \*\*Aplicaciones Financieras\*\*

- En el sector financiero, las bases de datos relacionales se utilizan para manejar transacciones bancarias, inversiones, gestión de cuentas y otros servicios financieros. La integridad y la consistencia de los datos son críticas en este contexto.

4. \*\*Sistemas de Reservas y Ticketing\*\*

- Las aerolíneas, hoteles y otros servicios de reservas utilizan bases de datos relacionales para gestionar la disponibilidad de asientos, habitaciones y entradas, asegurando que la información esté actualizada en tiempo real.

5. \*\*Sistemas de Gestión de Inventarios\*\*

- Las empresas que manejan grandes volúmenes de productos utilizan bases de datos relacionales para rastrear y gestionar el inventario, asegurando que la información sobre existencias, pedidos y proveedores esté organizada y accesible.

### Preguntas de Retroalimentación

1. \*\*¿Por qué son las bases de datos relacionales preferidas en sistemas ERP?\*\*

2. \*\*¿Cómo se benefician las plataformas CRM del uso de bases de datos relacionales?\*\*

3. \*\*¿Qué papel juegan las bases de datos relacionales en aplicaciones financieras?\*\*

4. \*\*¿Cuál es la importancia de las bases de datos relacionales en sistemas de reservas y ticketing?\*\*

5. \*\*Explique cómo las bases de datos relacionales ayudan en la gestión de inventarios en grandes empresas.\*\*

Respuestas

1. Son preferidas en las empresas por que ayudan en la organización de la entidad además de ser útiles para la toma de decisiones basadas en datos
2. Se benefician al estudiar los patrones de consumo de sus clientes y mantener la seguridad de las transacciones
3. Son importantes para almacenar y ejecutar transacciones seguras
4. La impostancia es que son útiles para mantener los datos de reservas y vuelos actualizada y a la mano
5. Ayudan manteniendo organizada la información y volviéndola fácil de revizar

**### Clase 14: Futuro de las Bases de Datos Relacionales**

\*\*1. Evolución y Adaptación:\*\*

Las bases de datos relacionales han sido la piedra angular de la gestión de datos durante décadas, y aunque las bases de datos NoSQL y otras tecnologías emergentes han ganado popularidad, las bases de datos relacionales continúan evolucionando para adaptarse a nuevas demandas. Esta evolución incluye mejoras en la escalabilidad, integración con la nube, y soporte para grandes volúmenes de datos.

\*\*2. Integración con Tecnologías Emergentes:\*\*

Las bases de datos relacionales están integrándose con tecnologías emergentes como el procesamiento en la nube, el análisis de big data, y la inteligencia artificial. Los proveedores de bases de datos relacionales están añadiendo capacidades de análisis avanzado y soporte para aprendizaje automático dentro de sus sistemas.

\*\*3. Bases de Datos como Servicio (DBaaS):\*\*

El modelo de bases de datos como servicio está ganando terreno, donde las organizaciones pueden usar bases de datos relacionales en la nube sin preocuparse por la administración de la infraestructura subyacente. Esto facilita la escalabilidad y reduce la complejidad operativa.

\*\*4. Soporte para Datos No Relacionales:\*\*

Muchas bases de datos relacionales están ampliando su capacidad para manejar datos no relacionales, permitiendo que los usuarios combinen las fortalezas del modelo relacional con la flexibilidad de almacenar y consultar datos en formatos JSON, XML, y otros tipos de datos semi-estructurados.

\*\*5. Automatización y Autogestión:\*\*

Las bases de datos relacionales están adoptando la automatización y capacidades autogestionadas, incluyendo el ajuste automático de consultas, administración de recursos y mantenimiento proactivo. Esto se está logrando a través del uso de inteligencia artificial y aprendizaje automático.

### Preguntas de Retroalimentación

1. ¿Cómo están evolucionando las bases de datos relacionales para adaptarse a las nuevas demandas tecnológicas?

2. ¿Qué papel juega el modelo de bases de datos como servicio (DBaaS) en el futuro de las bases de datos relacionales?

3. ¿Por qué es importante que las bases de datos relacionales soporten datos no relacionales en el contexto actual?

4. ¿Cómo la automatización y autogestión están impactando el manejo de bases de datos relacionales?

5. ¿De qué manera las bases de datos relacionales se están integrando con tecnologías como la inteligencia artificial y el análisis de big data?

Respuestas

1. Están adaptándose para trabajar con mas manejo de datos, además de integraciones con la nube y adaptándose para manejo de datos semiestructurados
2. Estas haciendo mas accesible el trabajo de las bases de datos y el mantenimiento ya que al estar alojadas en la nube el ahorro de infraestructura es gigante
3. Es importante porque se están usando cada vez mas datos semi estructurados en la industria web, que coge cada vez mas campo
4. Las esta volviendo mas fuertes integrando el autoaprendizaje en las mismas
5. Integrándose con estas nuevas tecnologías para ser mas versátiles

**### Clase 15: ¿Qué es la Inteligencia de Negocios?**

#### 15.1. Conceptos y definiciones

\*\*Inteligencia de Negocios (BI)\*\* es un conjunto de procesos, tecnologías y herramientas que permiten a las organizaciones recopilar, almacenar, analizar y acceder a datos importantes para tomar decisiones informadas. La BI transforma datos en información útil y significativa, permitiendo a las empresas identificar tendencias, patrones, y obtener una visión integral de su funcionamiento.

\*\*Componentes Clave de la BI\*\*:

1. \*\*Recopilación de datos\*\*: Obtener datos de diferentes fuentes internas y externas.

2. \*\*Almacenamiento de datos\*\*: Centralizar los datos en almacenes de datos (data warehouses).

3. \*\*Análisis de datos\*\*: Aplicar técnicas de análisis para interpretar los datos.

4. \*\*Reporting y visualización\*\*: Presentar los datos de manera comprensible y visual.

\*\*Beneficios de la BI\*\*:

- Mejora la toma de decisiones.

- Optimiza procesos operativos.

- Incrementa la eficiencia y productividad.

- Identifica nuevas oportunidades de negocio.

- Facilita el análisis del comportamiento del cliente y la competencia.

#### Preguntas de retroalimentación:

1. ¿Qué papel juega la recopilación de datos en la Inteligencia de Negocios?

2. ¿Cómo puede la BI ayudar a las empresas a identificar nuevas oportunidades de negocio?

3. ¿Qué diferencia hay entre un reporte generado por BI y un simple informe de datos?

4. ¿Por qué es importante la visualización de datos en la Inteligencia de Negocios?

5. ¿Cómo contribuye la BI a la optimización de procesos operativos en una empresa?

Respuestas

1. Es un papel crucial al inicio del análisis de datos, pues es el inicio del proceso para todo lo que BI
2. Con la ayuda de el análisis de datos después de hacer todo el trabajo de transformación de los datos, podemos ver comportamientos y cifras mas claras
3. El reporte de BI es mas estructurado mas acertado, dinamico y fácil de entender
4. Porque es la que nos ayuda con la toma de deciciones tomando como base esos datos e informes proporcionados
5. De la manera en que plicando los datos podemos prever problemas futuros y saber donde estamos fallando, no estaremos a ciegas

**### Clase 16: Minería de Datos**

#### 16. Minería de Datos

La minería de datos es un proceso que se utiliza para descubrir patrones, tendencias y relaciones ocultas en grandes conjuntos de datos. Es una herramienta clave dentro de la inteligencia de negocios y la ciencia de datos, ya que permite a las organizaciones extraer información valiosa de los datos que recopilan y almacenan. Este proceso involucra varias técnicas de análisis y modelado que se aplican para convertir los datos en conocimiento útil.

##### 16.1. Conceptos Fundamentales

- \*\*Definición\*\*: La minería de datos es el proceso de analizar grandes volúmenes de datos para identificar patrones y relaciones que no son obvios a simple vista.

- \*\*Objetivo\*\*: El objetivo principal de la minería de datos es transformar datos en información significativa que pueda usarse para tomar decisiones estratégicas.

- \*\*Datos Masivos\*\*: Trabaja con grandes volúmenes de datos, a menudo almacenados en bases de datos relacionales, NoSQL, o en entornos de big data.

##### 16.2. Técnicas de Minería de Datos

- \*\*Clasificación\*\*: Es una técnica que organiza los datos en categorías predeterminadas. Por ejemplo, un algoritmo de clasificación podría usarse para categorizar clientes según su probabilidad de compra.

- \*\*Reglas de Asociación\*\*: Se utilizan para encontrar relaciones entre variables dentro de un conjunto de datos. Un ejemplo clásico es la recomendación de productos en función de las compras anteriores de un cliente.

- \*\*Clustering (Agrupamiento)\*\*: Agrupa los datos en clusters (grupos) basados en características similares. Esto es útil en la segmentación de mercado.

- \*\*Regresión\*\*: Se usa para predecir valores numéricos basados en otros atributos de los datos. Por ejemplo, predecir el precio de una casa en función de su tamaño, ubicación, etc.

- \*\*Detección de Anomalías\*\*: Identifica datos que no se ajustan a los patrones normales del conjunto de datos, lo cual es útil en la detección de fraudes.

##### 16.3. Herramientas de Minería de Datos

- \*\*R\*\*: Un entorno de software libre usado para computación estadística y gráficos. Es muy popular para la minería de datos y análisis estadístico.

- \*\*Python\*\*: Con librerías como Scikit-learn y Pandas, Python es muy usado para la minería de datos.

- \*\*RapidMiner\*\*: Una plataforma que ofrece un entorno de desarrollo integrado para la minería de datos.

- \*\*Weka\*\*: Un conjunto de herramientas para realizar minería de datos y aprendizaje automático.

##### 16.4. Aplicaciones de la Minería de Datos

- \*\*Marketing\*\*: Personalización de ofertas y promociones, análisis de la satisfacción del cliente.

- \*\*Salud\*\*: Identificación de patrones en datos clínicos, predicción de enfermedades.

- \*\*Finanzas\*\*: Detección de fraudes, análisis de riesgo crediticio.

- \*\*Retail\*\*: Optimización de inventario, análisis de comportamiento de compra.

### Preguntas de Retroalimentación

1. ¿Qué es la minería de datos y cuál es su principal objetivo dentro de la inteligencia de negocios?

2. Describe la técnica de clasificación y da un ejemplo de cómo podría utilizarse en un entorno empresarial.

3. ¿Qué diferencia hay entre las técnicas de clustering y las reglas de asociación en minería de datos?

4. Menciona al menos dos herramientas que se utilizan en minería de datos y sus aplicaciones principales.

5. ¿Cómo se aplica la detección de anomalías en la minería de datos, y por qué es importante en sectores como el financiero?

Respuestas

1. Es tomar los datos y analizarlos para sacar de ellos comportamientos, y datos mas profundos que al ser analizados nos dicen mucho de la información
2. Es un método que clasifica los datos de manera efectiva, y serviría en una empresa para saber intereses de los clientes o fallos que tengan las secciones
3. El clustering se usa para dividir la información en cajones del mismo tema, y la asociación relaciona la información como parecidos, la primera nos segmenta la información y la segunda nos asocia patrones
4. R, para generas graficos y analíticas y Python para análisis de datos con pandas
5. Reconoce datos que no pertencen al movimiento normal y asi previene los fraudes financieros

**### Clase 17: Dashboards**

\*\*Definición y Propósito\*\*:

Un dashboard, o panel de control, es una herramienta de visualización de datos que muestra información clave de manera concisa y accesible. Permite a los usuarios monitorear el estado de indicadores y métricas relevantes en tiempo real, facilitando la toma de decisiones informadas.

\*\*Componentes de un Dashboard\*\*:

1. \*\*Visualizaciones\*\*: Gráficos, tablas, mapas y otros elementos visuales que presentan datos de manera clara.

2. \*\*Métricas e Indicadores\*\*: Datos específicos y KPIs (Key Performance Indicators) que se rastrean para medir el rendimiento.

3. \*\*Filtros y Controles\*\*: Herramientas que permiten al usuario ajustar los datos mostrados, como filtros de fecha, categorías o niveles de detalle.

4. \*\*Actualización en Tiempo Real\*\*: Funcionalidad para actualizar datos en tiempo real o a intervalos regulares.

\*\*Tipos de Dashboards\*\*:

1. \*\*Dashboards Operativos\*\*: Enfocados en la gestión diaria y monitoreo de procesos en tiempo real.

2. \*\*Dashboards Analíticos\*\*: Diseñados para el análisis en profundidad de datos históricos y tendencias.

3. \*\*Dashboards Estratégicos\*\*: Orientados a la visualización de KPIs estratégicos y objetivos a largo plazo.

\*\*Beneficios de los Dashboards\*\*:

- \*\*Acceso Rápido a Información Clave\*\*: Facilitan la visualización rápida de datos críticos.

- \*\*Mejora en la Toma de Decisiones\*\*: Ayudan a identificar patrones y tendencias, permitiendo decisiones basadas en datos.

- \*\*Monitoreo Continuo\*\*: Permiten el seguimiento en tiempo real del rendimiento y de los indicadores clave.

\*\*Herramientas Comunes para Crear Dashboards\*\*:

- \*\*Tableau\*\*: Conocido por sus potentes capacidades de visualización y facilidad de uso.

- \*\*Power BI\*\*: Ofrece integración con diversas fuentes de datos y funcionalidades de análisis avanzado.

- \*\*Looker\*\*: Proporciona una plataforma flexible para la exploración y visualización de datos.

\*\*Mejores Prácticas\*\*:

1. \*\*Claridad en el Diseño\*\*: Utiliza visualizaciones claras y evita sobrecargar el dashboard con información innecesaria.

2. \*\*Relevancia de Datos\*\*: Asegúrate de que los datos mostrados sean relevantes para el usuario final.

3. \*\*Interactividad\*\*: Incorpora elementos interactivos para permitir a los usuarios explorar los datos en detalle.

4. \*\*Actualización Regular\*\*: Mantén los datos actualizados para asegurar que la información sea precisa y relevante.

### Preguntas de Retroalimentación

1. \*\*¿Cuál es el propósito principal de un dashboard en el contexto de la inteligencia de negocios?\*\*

2. \*\*¿Qué tipo de datos suelen mostrarse en un dashboard analítico y cómo se diferencian de los dashboards operativos?\*\*

3. \*\*Menciona dos herramientas comunes para crear dashboards y una característica destacada de cada una.\*\*

4. \*\*¿Por qué es importante la actualización en tiempo real en un dashboard y cómo puede afectar la toma de decisiones?\*\*

5. \*\*¿Qué aspectos deben considerarse para asegurar la efectividad de un dashboard en la visualización de datos?\*\*  
  
respuestas:

1. Sirve para ver los datos con mas claridad

2. Se muestran los datos que son utiles para la organizacion a la hora de tomar decisiones y se diferencia del operativo en el sentido que el operativo solo da datos de monitoreo en tiempo real

3. Tableau: es muy usada pues tiene potentes herramientas para la generacion de reporting

Power bi: cuenta con el beneficio de ser de microsoft y eso le presta conectividad con mas aplicaciones de analisis

4. al tener los datos en tiempo real, puedes tomar acciones mas rapido y actualizadas

5. que sea accesible, facil de entender y este actualizado  
  
**### Clase 18: Patrones de Comportamiento**

#### 1. \*\*Definición de Patrones de Comportamiento\*\*

Los patrones de comportamiento en el análisis de datos se refieren a las tendencias o regularidades que emergen de la evaluación de grandes volúmenes de datos. Estos patrones ayudan a identificar comportamientos recurrentes o predecibles que pueden ser utilizados para tomar decisiones informadas y realizar análisis predictivo.

#### 2. \*\*Tipos de Patrones de Comportamiento\*\*

- \*\*Patrones Estacionales\*\*: Variaciones en los datos que ocurren en intervalos regulares, como el aumento de las ventas durante las festividades.

- \*\*Patrones de Tendencia\*\*: Cambios en los datos a lo largo del tiempo, como un crecimiento constante en las ventas de un producto.

- \*\*Patrones Cíclicos\*\*: Fluctuaciones en los datos que no son estacionales pero que siguen ciclos más largos, como la fluctuación económica.

- \*\*Patrones de Serie Temporal\*\*: Datos secuenciales donde las observaciones están ordenadas en el tiempo, permitiendo la identificación de tendencias y estacionalidades.

#### 3. \*\*Uso de Patrones de Comportamiento\*\*

- \*\*Optimización de Estrategias de Marketing\*\*: Identificar cuándo los clientes están más propensos a comprar ciertos productos.

- \*\*Gestión de Inventarios\*\*: Ajustar el stock en función de patrones estacionales o de tendencia para evitar desabastecimientos o excesos.

- \*\*Prevención de Fraude\*\*: Detectar comportamientos inusuales o desviaciones significativas que podrían indicar fraude.

#### 4. \*\*Herramientas y Métodos para Identificar Patrones\*\*

- \*\*Análisis de Serie Temporal\*\*: Utilizado para analizar datos secuenciales y detectar patrones y tendencias.

- \*\*Algoritmos de Machine Learning\*\*: Métodos como clustering y regresión pueden identificar patrones complejos en grandes conjuntos de datos.

- \*\*Visualización de Datos\*\*: Herramientas como gráficos de líneas, histogramas y heatmaps pueden ayudar a identificar patrones visualmente.

#### 5. \*\*Ejemplos Prácticos\*\*

- \*\*Retail\*\*: Análisis de patrones de compra para predecir demanda y ajustar las estrategias de inventario.

- \*\*Salud\*\*: Identificación de patrones en los datos de pacientes para mejorar el diagnóstico y la prevención de enfermedades.

- \*\*Finanzas\*\*: Detección de patrones en transacciones financieras para identificar comportamientos sospechosos y prevenir fraudes.

### Preguntas para Retroalimentación

1. \*\*¿Qué tipos de patrones de comportamiento pueden identificarse en los datos de ventas estacionales, y cómo pueden ser utilizados para optimizar las estrategias de marketing?\*\*

2. \*\*¿Cómo diferenciarías entre un patrón cíclico y un patrón estacional en un análisis de serie temporal? Proporciona ejemplos de cada uno.\*\*

3. \*\*¿Qué herramientas o métodos pueden ser utilizados para identificar patrones de comportamiento en grandes volúmenes de datos, y cuáles son las ventajas de cada uno?\*\*

4. \*\*¿Cómo pueden los patrones de comportamiento ayudar en la gestión de inventarios, y qué impacto puede tener en la eficiencia operativa de una empresa?\*\*

5. \*\*Explica cómo los algoritmos de machine learning pueden ser utilizados para detectar patrones complejos en los datos y dar ejemplos de casos en los que esto podría ser beneficioso.\*\*

Respuestas:

1. Tipos de patrones como estacionales y de tendecia, nos ayudan a definir, en que momento los usuarios están mas propensos a adquirir ciertos productos, lo que permite optimizar los inventarios y las exibiciones
2. El patrón estacional puede presentarse en un solo mes tipo Halloween, el patrón cíclico, puede reportar temporalidades mas largas, tipo 6 meses
3. Análisis de datos que nos sirve para analizar como se comportan los consumidores o cualquier aspecto analizado  
   - usando métodos como clustering podemos analizar datos pasados  
   - herramientas de vizualizacion que nos permiten ver y analizar grandes volúmenes de datos
4. Ayudan en el sentido en que se analizan los patrones y con base en eso se realizan pedidos o se renueva la exhibición
5. Puede ser beneficioso, ya qye nos permite hacer toda una vista reversa a nuestros datos y sobre ellos obtener datos predictivos para tomar acciones futuras

¡Excelente! Has respondido correctamente a las preguntas, demostrando una buena comprensión de los conceptos relacionados con los patrones de comportamiento en el análisis de datos. Aquí te enseño la siguiente clase y te proporcionaré 5 nuevas preguntas de retroalimentación.

**### Clase 19: Anomalías**

#### 1. \*\*Definición de Anomalías\*\*

Las anomalías, también conocidas como outliers, son datos que se desvían significativamente del resto del conjunto de datos. Identificar estas desviaciones es crucial, ya que pueden señalar problemas, oportunidades o errores en el sistema.

#### 2. \*\*Tipos de Anomalías\*\*

- \*\*Anomalías Puntuadas (Point Anomalies)\*\*: Un solo punto de datos que es anómalo comparado con el resto del conjunto de datos.

- \*\*Anomalías Contextuales (Contextual Anomalies)\*\*: Un punto de datos que es anómalo solo en un contexto particular, como una temperatura inusualmente alta en invierno.

- \*\*Anomalías Colectivas (Collective Anomalies)\*\*: Un grupo de puntos de datos que, cuando se consideran juntos, son anómalos, aunque los puntos individuales puedan no serlo.

#### 3. \*\*Detección de Anomalías\*\*

- \*\*Métodos Estadísticos\*\*: Se utilizan para identificar valores atípicos basados en la desviación estándar, la media, o la mediana.

- \*\*Métodos Basados en Aprendizaje Automático\*\*: Algoritmos como Isolation Forest, Support Vector Machines (SVM) y redes neuronales se emplean para identificar patrones anómalos.

- \*\*Métodos Basados en Distancias\*\*: Utilizan la distancia entre puntos de datos para identificar aquellos que están aislados del resto.

#### 4. \*\*Aplicaciones de la Detección de Anomalías\*\*

- \*\*Detección de Fraude\*\*: Identificar transacciones financieras inusuales que puedan indicar fraude.

- \*\*Monitoreo de la Salud\*\*: Detección de señales anómalas en datos de pacientes que puedan indicar un problema de salud.

- \*\*Gestión de Calidad\*\*: Identificación de defectos en la producción que pueden llevar a fallos en productos.

#### 5. \*\*Desafíos en la Detección de Anomalías\*\*

- \*\*Falsos Positivos y Falsos Negativos\*\*: Identificar incorrectamente datos normales como anómalos, o no detectar anomalías reales.

- \*\*Volumen y Variedad de Datos\*\*: Grandes volúmenes de datos y diversidad pueden dificultar la identificación precisa de anomalías.

- \*\*Contexto Dependiente\*\*: En algunos casos, lo que es considerado anómalo puede depender del contexto específico, lo que añade complejidad al proceso.

### Preguntas para Retroalimentación

1. \*\*¿Qué es una anomalía y cómo puede impactar en el análisis de datos si no se detecta a tiempo?\*\*

2. \*\*Explica la diferencia entre una anomalía puntual y una anomalía contextual con ejemplos.\*\*

3. \*\*¿Qué métodos pueden ser utilizados para detectar anomalías en un conjunto de datos? Menciona al menos dos y describe sus principios básicos.\*\*

4. \*\*¿Cómo se pueden aplicar las técnicas de detección de anomalías en la prevención del fraude financiero?\*\*

5. \*\*Describe un desafío común en la detección de anomalías y cómo puede ser abordado en un sistema de análisis de datos.\*\*

Respuestas:

1. Es una variación en los datos que nos otorga una señal o un cotexto sobre un cambio en los datos
2. Animalia puntual es un punto de variación an varios datos, anomalía contextual es un punto en un contexto puntual, ejemplo una alza en la bolsa cuando la tendencia es 100% bajista
3. Método estadístico, basado en datos se pueden detectar anomalías  
   método de autoaprendizaje, con ayuda de maquinas podemos generas algoritmos que detecten anomalías
4. Detectando anomalías en transacciones
5. En las tendencias económicas, es difícil predecir su rumbo y puede ser abordado, perfeccionando los análisis para saber hasta el ultimo movimiento y detectar errores como en el 2008

**### Clase 20: Segmentación de Mercado**

#### 1. \*\*Definición de Segmentación de Mercado\*\*

La segmentación de mercado es el proceso de dividir un mercado en grupos más pequeños y homogéneos que comparten características similares, como necesidades, intereses, ubicación geográfica, demografía, comportamiento de compra, entre otros.

#### 2. \*\*Tipos de Segmentación de Mercado\*\*

- \*\*Segmentación Demográfica\*\*: Se basa en características demográficas como edad, género, ingresos, educación, y ocupación.

- \*\*Segmentación Geográfica\*\*: Divide el mercado según la ubicación, como país, región, ciudad, o clima.

- \*\*Segmentación Psicográfica\*\*: Se centra en características psicológicas, como estilo de vida, valores, intereses y opiniones.

- \*\*Segmentación Conductual\*\*: Se basa en el comportamiento del consumidor, como patrones de compra, lealtad a la marca, o respuesta a un producto.

#### 3. \*\*Beneficios de la Segmentación de Mercado\*\*

- \*\*Personalización\*\*: Permite a las empresas personalizar sus productos, servicios y mensajes de marketing para satisfacer las necesidades específicas de cada segmento.

- \*\*Eficiencia en el Marketing\*\*: Facilita la creación de campañas de marketing más efectivas al enfocarse en grupos específicos con un mayor potencial de conversión.

- \*\*Mejora en la Retención de Clientes\*\*: Al dirigirse a segmentos específicos, es más fácil mantener a los clientes satisfechos y leales.

#### 4. \*\*Proceso de Segmentación de Mercado\*\*

- \*\*Investigación de Mercado\*\*: Recolectar datos para identificar las diferentes características de los consumidores.

- \*\*Identificación de Segmentos\*\*: Analizar los datos para identificar grupos homogéneos dentro del mercado.

- \*\*Selección de Segmentos\*\*: Evaluar y seleccionar los segmentos que ofrecen el mayor potencial de mercado.

- \*\*Desarrollo de Estrategias\*\*: Crear estrategias de marketing específicas para cada segmento seleccionado.

#### 5. \*\*Aplicaciones de la Segmentación de Mercado\*\*

- \*\*Lanzamiento de Nuevos Productos\*\*: Identificar los segmentos de mercado más receptivos a un nuevo producto.

- \*\*Optimización de Campañas Publicitarias\*\*: Ajustar el mensaje publicitario para resonar mejor con diferentes segmentos.

- \*\*Expansión Internacional\*\*: Adaptar productos y campañas a diferentes regiones geográficas.

### Preguntas para Retroalimentación

1. \*\*¿Qué es la segmentación de mercado y por qué es importante para las empresas?\*\*

2. \*\*Describe las diferencias entre la segmentación demográfica y la segmentación psicográfica. ¿En qué escenarios podría ser más útil cada una?\*\*

3. \*\*Explica cómo la segmentación geográfica podría beneficiar a una empresa que desea expandirse a nivel internacional.\*\*

4. \*\*Menciona un ejemplo de cómo una empresa podría utilizar la segmentación conductual para mejorar la retención de clientes.\*\*

5. \*\*¿Cuál es el proceso general de segmentación de mercado y cómo ayuda a desarrollar estrategias de marketing más efectivas?\*\*

Respuestas:

1. Es organizar un grupo de personas por grupos homogeneos y es importante para definir estrategias de marketing o lanzamiento

2. La primera se centra en los rasgos fisicos y de entorno del grupo, y la otra mas en la cualidades y gustos, la primera puede ser usada para hacer publicidad para adultos, y la otra para hacer publicidad para personas con necesidades especiales como la sensacion de fiesta 3. La puede beneficiar en el sentido en que taniendo los datos de la poblacion a la que va a llegar puede perfeccionar su estrategia de mercadeo

4. Podria utilizarlo para saber que tan propensos estan los clientes a cambiar de marca, y para conocer sus patrones de compra sobre el producto a vender

5. Investigacion de mercado  
- identificacion de segmanto  
- seleccion de segmento  
- desarrollo de estrategias  
con estos pasos desarrollamos toda la estrategia de marketing al saber claramente a que publico llegar y que intereses tiene

**\*\*Clase 21: Predicción\*\***

La predicción en el análisis de datos se refiere al proceso de utilizar datos históricos y modelos matemáticos o algoritmos de machine learning para anticipar resultados futuros. Este es un componente crucial en muchos ámbitos, incluyendo negocios, ciencias, y tecnología, ya que permite tomar decisiones informadas y estratégicas basadas en las tendencias observadas.

### 21.1. Modelos de Predicción

Los modelos de predicción se construyen utilizando diferentes enfoques, dependiendo de la naturaleza de los datos y del objetivo. Los más comunes son:

- \*\*Regresión Lineal\*\*: Utilizado para predecir valores continuos, como ingresos o precios de la vivienda. Se basa en la relación entre una variable dependiente y una o más variables independientes.

- \*\*Series Temporales\*\*: Enfoque utilizado para predecir valores futuros basados en datos secuenciales a lo largo del tiempo, como las ventas mensuales de una tienda o las visitas a una página web.

- \*\*Modelos de Clasificación\*\*: Utilizados para predecir categorías o etiquetas, como si un cliente realizará una compra o no, basado en sus características y comportamiento.

- \*\*Redes Neuronales\*\*: Utilizadas en problemas más complejos, como el reconocimiento de imágenes o la predicción de texto. Las redes neuronales pueden captar patrones no lineales en los datos y realizar predicciones con alta precisión.

### 21.2. Aplicaciones de la Predicción

La predicción tiene aplicaciones en diversas áreas, como:

- \*\*Finanzas\*\*: Predicción de precios de acciones, tasas de interés, y riesgos financieros.

- \*\*Marketing\*\*: Anticipación de tendencias de consumo, efectividad de campañas publicitarias, y segmentación de clientes.

- \*\*Salud\*\*: Predicción de enfermedades, gestión de recursos hospitalarios, y personalización de tratamientos.

- \*\*Producción\*\*: Pronóstico de demanda, optimización de la cadena de suministro, y mantenimiento predictivo de maquinaria.

### 21.3. Retos en la Predicción

La predicción presenta varios retos, entre ellos:

- \*\*Calidad de los Datos\*\*: La precisión de las predicciones depende de la calidad y cantidad de los datos disponibles. Datos incompletos o sesgados pueden llevar a resultados incorrectos.

- \*\*Sobreajuste (Overfitting)\*\*: Ocurre cuando un modelo es demasiado complejo y se ajusta demasiado bien a los datos de entrenamiento, pero falla en generalizar en nuevos datos.

- \*\*Interpretabilidad\*\*: Algunos modelos de predicción, especialmente los basados en machine learning, pueden ser difíciles de interpretar y explicar.

### 21.4. Herramientas para la Predicción

Existen diversas herramientas y lenguajes de programación que facilitan la creación de modelos predictivos, como:

- \*\*Python\*\*: Con bibliotecas como Scikit-learn, TensorFlow, y Keras, es una opción popular para desarrollar modelos de predicción.

- \*\*R\*\*: Ofrece potentes paquetes para análisis estadístico y modelado predictivo.

- \*\*SPSS\*\*: Utilizado en entornos académicos y empresariales para análisis estadístico y predicción.

- \*\*Excel\*\*: Para predicciones simples, Excel ofrece funcionalidades básicas para análisis de regresión y series temporales.

---

### Preguntas de Retroalimentación

1. \*\*¿Qué es la predicción en el análisis de datos y por qué es importante?\*\*

2. \*\*Describe dos modelos de predicción y da un ejemplo de aplicación para cada uno.\*\*

3. \*\*¿Cuáles son algunos de los retos más comunes en la predicción y cómo pueden impactar los resultados?\*\*

4. \*\*Menciona tres áreas donde la predicción se utiliza comúnmente y explica cómo beneficia a cada una.\*\*

5. \*\*¿Qué herramientas conoces para desarrollar modelos predictivos y en qué situaciones las usarías?\*\*

Respuestas

1. Es tomar una agrupacion de datos y usarla para predcir acciones futuras o comportamientos futuros de los datos

2. Series temporales: se usa para predecir valores futuros tomando datos secuenciales

Redes neuronales: usados para predecir problemas mas complejos

3. datos sesgados o mal digitados, y dificultad de explicacion con algunos programas de machine lernig

4. salud, benefica en la prediccion de enfermedades

Economia, predecir indices y posibles fraudes

Marketing, predecir ventas y patrones de compras

5. Conozco python y excel, pero usaria mayoritariamente python

**### Clase 22: Recolección de Datos**

La recolección de datos es el primer paso en el proceso de análisis de datos. Este proceso consiste en obtener información que luego será utilizada para su análisis, ya sea para la toma de decisiones, creación de modelos predictivos o para cualquier otro propósito dentro de la inteligencia de negocios. A continuación, se presentan los conceptos clave de esta clase.

#### 22.1. \*\*Definición de Recolección de Datos\*\*

La recolección de datos es el proceso de reunir información a partir de diversas fuentes para su posterior análisis. Estos datos pueden ser cuantitativos o cualitativos y se obtienen utilizando diferentes métodos, dependiendo del tipo de información que se necesite y el propósito del análisis.

#### 22.2. \*\*Métodos de Recolección de Datos\*\*

Existen varios métodos para la recolección de datos, cada uno adecuado para diferentes tipos de estudios y análisis:

- \*\*Encuestas\*\*: Consisten en un conjunto de preguntas que se hacen a un grupo específico de personas para recolectar información sobre opiniones, comportamientos o características.

- \*\*Entrevistas\*\*: Método cualitativo donde se obtiene información detallada a través de preguntas directas a los participantes.

- \*\*Observación\*\*: Implica la recolección de datos mediante la observación directa de los sujetos o fenómenos de estudio.

- \*\*Experimentos\*\*: Método en el que se manipulan variables para observar los efectos en el sujeto de estudio.

- \*\*Datos secundarios\*\*: Recolección de información existente previamente, como informes, bases de datos públicas o registros históricos.

#### 22.3. \*\*Importancia de la Recolección de Datos\*\*

La recolección de datos es crucial porque:

- \*\*Proporciona la base del análisis\*\*: Sin datos precisos y relevantes, cualquier análisis posterior será defectuoso.

- \*\*Apoya la toma de decisiones informada\*\*: Los datos recolectados adecuadamente permiten a las empresas y organizaciones tomar decisiones basadas en evidencia, en lugar de suposiciones.

- \*\*Facilita el modelado predictivo\*\*: La calidad y cantidad de datos recolectados influyen directamente en la precisión de los modelos predictivos.

#### 22.4. \*\*Desafíos en la Recolección de Datos\*\*

- \*\*Calidad de los datos\*\*: Asegurarse de que los datos sean precisos, completos y libres de sesgo es fundamental.

- \*\*Volumen de datos\*\*: La gestión y almacenamiento de grandes volúmenes de datos puede ser un desafío técnico.

- \*\*Privacidad y ética\*\*: Recolectar datos personales o sensibles requiere cumplir con regulaciones de privacidad y mantener altos estándares éticos.

### 5 Preguntas de Retroalimentación

1. ¿Qué es la recolección de datos y por qué es un paso crucial en el proceso de análisis de datos?

2. Menciona y describe dos métodos de recolección de datos que se utilizan en estudios cualitativos.

3. ¿Por qué es importante garantizar la calidad de los datos durante el proceso de recolección?

4. ¿Qué desafíos éticos y legales pueden surgir durante la recolección de datos?

5. ¿Cómo afecta la cantidad de datos recolectados a la precisión de los modelos predictivos?

Respuestas

1. Es importante, por que trabajamos sobre los datos, para analizarlos y poder crear modelos con base en ellos

2. La encuesta y la entrevista

3. Porque cuanto mas certeros sean los datos mejor sera el analisis y la prediccion

4. Que al manejar datos personales, se viole la privacidad y esos datos sean compartidos de mala manera

5. Entre mayor sea la cantidad de datos que se recolecta, mejor sera el trabajo sobre ellos

Vamos a continuar con la \*\*Clase 23: Limpieza de Datos\*\*. Este es un proceso esencial en el manejo de datos, ya que asegura que la información que utilizas para análisis o modelado sea precisa, completa y relevante.

**Clase 23: Limpieza de Datos**

#### 23.1. \*\*¿Qué es la Limpieza de Datos?\*\*

La limpieza de datos es el proceso de detectar y corregir (o eliminar) datos corruptos, inexactos, incompletos, duplicados o mal formateados dentro de un conjunto de datos. Este proceso es crucial para asegurar que los datos utilizados en el análisis o en la creación de modelos sean confiables.

#### 23.2. \*\*Pasos Comunes en la Limpieza de Datos\*\*

1. \*\*Eliminación de Duplicados\*\*: Los datos duplicados pueden sesgar los resultados de un análisis. Identificar y eliminar duplicados es un paso esencial.

2. \*\*Manejo de Datos Faltantes\*\*:

- \*\*Imputación\*\*: Rellenar los valores faltantes con la media, mediana, o una estimación basada en otros datos.

- \*\*Eliminación\*\*: Si un dato faltante es crítico y no puede ser imputado, es posible eliminar esa entrada del análisis.

3. \*\*Corrección de Inconsistencias\*\*: Asegurarse de que los datos sigan un formato consistente. Por ejemplo, si una columna de fechas contiene diferentes formatos, es necesario estandarizarlos.

4. \*\*Identificación y Corrección de Valores Atípicos\*\*: Los valores atípicos pueden ser errores o simplemente representaciones raras de la realidad. Es importante analizarlos y decidir si se corrigen, eliminan o se manejan de otra forma.

5. \*\*Transformación de Datos\*\*: A veces es necesario transformar los datos para que sean más fáciles de analizar. Esto puede incluir la normalización, estandarización o la conversión de categorías.

6. \*\*Validación de Datos\*\*: Verificar que los datos sean válidos y consistentes. Por ejemplo, que una columna de edades no contenga valores negativos.

#### 23.3. \*\*Herramientas y Técnicas para la Limpieza de Datos\*\*

- \*\*Software de Hojas de Cálculo\*\*: Herramientas como Excel o Google Sheets permiten realizar tareas básicas de limpieza de datos, como la eliminación de duplicados o el filtrado de datos.

- \*\*Lenguajes de Programación\*\*: Python y R son ampliamente utilizados en la limpieza de datos. Librerías como `pandas` en Python o `dplyr` en R facilitan el manejo y limpieza de grandes conjuntos de datos.

- \*\*Herramientas de ETL (Extract, Transform, Load)\*\*: Estas herramientas están diseñadas para mover y limpiar grandes volúmenes de datos. Ejemplos incluyen Talend, Apache Nifi, y herramientas de BI como Power BI o Tableau Prep.

#### 23.4. \*\*Importancia de la Limpieza de Datos\*\*

La limpieza de datos es un proceso fundamental porque asegura que el análisis sea preciso y que las decisiones basadas en esos datos sean informadas y correctas. Datos limpios conducen a mejores modelos predictivos, análisis más precisos y resultados más confiables.

---

### Preguntas de Retroalimentación

1. \*\*¿Qué es la limpieza de datos y por qué es esencial en el análisis de datos?\*\*

2. \*\*Menciona dos métodos para manejar datos faltantes y explica brevemente cuándo usarías cada uno.\*\*

3. \*\*¿Cómo puede afectar la presencia de duplicados en un conjunto de datos al análisis final?\*\*

4. \*\*¿Por qué es importante la identificación y corrección de valores atípicos en la limpieza de datos?\*\*

5. \*\*Nombra dos herramientas o técnicas que se pueden utilizar para la limpieza de datos y explica su uso.\*\*

Respuestas

1. Es importante pues para el analisi de datos la calidad de los datso con los que trabajamos es fundamental

2. Imputacion, llenar campos con datos tomados en relacion con datos pasados, caso de faltar algun dato relacionado con las ventas

- eliminacion: caso de que los datos no se puedan rellenar, ejemplo transacciones faltante

3. Puede presentar un analisi sobrecargado de informacion y variacion en los resultados al tener dos datos iguales

4. Para tener claridad en el analisis, que los datos son coherentes con los resultados

5. Herramientas de etl como odi de oracle y python con su biblioteca pandas

**Clase 24: Transformación de Datos**

La transformación de datos es una fase crucial en el proceso de preparación de datos para su análisis. Esta fase implica modificar, agregar o convertir datos para que sean adecuados para el análisis o la integración en un sistema de destino. La transformación de datos puede incluir procesos como normalización, agregación, filtrado, y derivación de nuevos atributos.

#### Subtemas:

1. \*\*Normalización y Estandarización:\*\*

- \*\*Normalización:\*\* Es el proceso de ajustar los valores de las variables numéricas a una escala común sin distorsionar las diferencias en los rangos de valores. Por ejemplo, se pueden escalar los valores de una columna entre 0 y 1.

- \*\*Estandarización:\*\* Consiste en centrar las variables y escalarlas de tal manera que el valor medio sea cero y la desviación estándar sea uno. Esto es útil en métodos de análisis estadísticos y de machine learning.

2. \*\*Agregación:\*\*

- Implica resumir datos detallados en un formato más sencillo y compacto. Por ejemplo, en lugar de analizar datos de ventas diarias, se puede agregar la información para obtener ventas mensuales o anuales.

3. \*\*Filtrado:\*\*

- Es el proceso de seleccionar un subconjunto de datos que cumple con ciertos criterios. Esto puede incluir la eliminación de valores atípicos, la selección de columnas relevantes, o la eliminación de registros duplicados.

4. \*\*Derivación de Nuevos Atributos:\*\*

- Consiste en crear nuevas variables a partir de las existentes. Por ejemplo, a partir de una columna con fechas de nacimiento, se puede derivar una nueva columna con la edad de las personas.

5. \*\*Conversión de Tipos de Datos:\*\*

- Implica transformar los datos de un tipo a otro, como convertir datos numéricos a texto o viceversa. Esto es necesario cuando los datos provienen de múltiples fuentes con diferentes formatos.

### Preguntas de Retroalimentación:

1. ¿Por qué es importante la normalización de datos en el análisis de datos?

2. ¿Qué diferencia hay entre normalización y estandarización?

3. ¿En qué casos es útil la agregación de datos?

4. ¿Por qué podría ser necesario filtrar datos durante la fase de transformación?

5. ¿Cómo podría derivarse un nuevo atributo a partir de los datos existentes?

Respuestas

1. Para tener un valor mas certero de los datos

2. La diferencia esta en la aproximacion, la normalizacion es mas exacta que la estandarizacion

3. en el caso de tener muchos datos para calcular periodos de tiempo largo

4. para eliminar datos atipicos, redundancias o errores en los datos

5. podriamos tomar la fecha de ingreso y restar la fecha actual para saber cuanto tiempo a permanecido en la compañia

¡Perfecto! Aquí está la siguiente clase.

**Clase 25: Selección de Conjuntos de Datos**

La selección de conjuntos de datos es un paso crítico en el análisis de datos y en la construcción de modelos de machine learning. Este proceso implica identificar y elegir los datos más relevantes y adecuados para el análisis o el entrenamiento del modelo. La calidad y la relevancia de los datos seleccionados influyen directamente en la precisión y efectividad de los resultados.

#### Subtemas:

1. \*\*Importancia de la Selección de Conjuntos de Datos:\*\*

- Seleccionar conjuntos de datos adecuados es crucial para obtener resultados precisos y relevantes. La selección incorrecta de datos puede llevar a conclusiones erróneas y modelos de baja calidad.

2. \*\*Criterios para la Selección de Datos:\*\*

- \*\*Relevancia:\*\* Los datos seleccionados deben ser directamente relevantes para el problema que se intenta resolver.

- \*\*Calidad:\*\* Los datos deben ser precisos, completos y libres de errores o valores atípicos que puedan distorsionar los resultados.

- \*\*Disponibilidad:\*\* Es importante considerar la disponibilidad y accesibilidad de los datos, especialmente en términos de tiempo y costo.

3. \*\*Fuentes de Datos:\*\*

- Los conjuntos de datos pueden provenir de diversas fuentes, como bases de datos internas, datos abiertos, APIs, o datos generados por sensores. Es esencial evaluar la confiabilidad de la fuente antes de utilizar los datos.

4. \*\*Tamaño del Conjunto de Datos:\*\*

- El tamaño del conjunto de datos puede influir en la capacidad del modelo para generalizar. Los conjuntos de datos grandes pueden ofrecer una mayor diversidad, mientras que los conjuntos de datos más pequeños pueden requerir técnicas de balanceo o aumento.

5. \*\*Preprocesamiento de los Datos Seleccionados:\*\*

- Después de seleccionar un conjunto de datos, es necesario preprocesarlo, lo que puede incluir la limpieza, transformación, y normalización de los datos para que estén listos para el análisis o modelado.

### Preguntas de Retroalimentación:

1. ¿Por qué es crucial la selección de conjuntos de datos en el análisis de datos?

2. ¿Cuáles son algunos criterios clave para seleccionar un conjunto de datos adecuado?

3. ¿Qué importancia tiene la fuente de los datos en el proceso de selección?

4. ¿Cómo puede influir el tamaño de un conjunto de datos en el modelado?

5. ¿Qué pasos de preprocesamiento son necesarios después de seleccionar un conjunto de datos?

Respuestas:

1. porque es mas eficaspara el analisis de datos y el uso de ellos

2. relevancia, calidad y disponibilidad

3. porque debe ser una fuente fiable y que nos brinde datos claros

4. dependiendo el tamaño elejimos el metodo de trabajo, si es grande tendremos que agrupar la gran variedad y si es pequeño, debemos centrarnos en la claridad

5. limpieza, extraccion, normalizacion, analisis y modelado

**Clase 26: Generación de Informes**

La generación de informes es una parte esencial del análisis de datos y del proceso de inteligencia de negocios. Los informes permiten a los usuarios visualizar, interpretar y tomar decisiones basadas en los datos analizados. Estos informes pueden ser periódicos, como informes mensuales, o pueden ser generados ad hoc, según las necesidades específicas del negocio.

**Subtemas:**

**Importancia de los Informes en el Análisis de Datos:**

Los informes son fundamentales para comunicar hallazgos y recomendaciones a las partes interesadas. Un buen informe facilita la toma de decisiones informadas y puede tener un impacto significativo en la dirección estratégica de la empresa.

Tipos de Informes:

Informes Operativos: Proveen detalles sobre el rendimiento diario de operaciones.

Informes Analíticos: Ofrecen un análisis profundo de los datos para identificar tendencias, patrones y correlaciones.

Dashboards: Son informes visuales interactivos que permiten a los usuarios explorar los datos en tiempo real.

Herramientas para la Generación de Informes:

Existen diversas herramientas que facilitan la generación de informes, como Power BI, Tableau, Google Data Studio, entre otras. Estas herramientas permiten crear informes dinámicos y personalizables.

Componentes de un Buen Informe:

Claridad: El informe debe ser claro y fácil de entender.

Relevancia: Debe centrarse en los aspectos más importantes y relevantes para el lector.

Visualización: El uso de gráficos, tablas y otros elementos visuales ayuda a comunicar la información de manera efectiva.

Accionabilidad: Los informes deben proporcionar información que permita a las partes interesadas tomar decisiones concretas.

Proceso de Generación de Informes:

Recolección de Datos: Compilación de los datos necesarios.

Análisis: Procesamiento y análisis de los datos para obtener insights.

Diseño del Informe: Selección de la estructura y visualización adecuada.

Revisión y Distribución: Validación del informe y su distribución a las partes interesadas.

Preguntas de Retroalimentación:

1. ¿Por qué es importante la generación de informes en la inteligencia de negocios?
2. ¿Cuál es la diferencia entre un informe operativo y un informe analítico?
3. ¿Qué ventajas ofrecen los dashboards frente a los informes tradicionales?
4. ¿Qué componentes son clave para asegurar la calidad de un informe?
5. ¿Cuáles son los pasos involucrados en el proceso de generación de informes?

Respuestas

1. Para trasmitir los resultados obtenidos del trabajo realizado con los datos
2. El informe analitico es mas profundo y detallado
3. Son graficos, claros y mucho mas comodos de explicar
4. Debe ser claro, relevantes, directos, faciles de vizualizar, y deben dar oportunidad a la accionabilidad
5. Recoleccion de datos  
   analisis  
   diseños  
   revision y distribucion

**Clase 27: Métricas**

Las métricas son valores numéricos que permiten medir el rendimiento, éxito o fracaso de un proceso, proyecto o acción en cualquier área del negocio. Son esenciales para evaluar resultados y tomar decisiones informadas basadas en datos. Las métricas bien definidas permiten a las empresas monitorear su progreso y hacer ajustes cuando sea necesario.

#### Subtemas:

1. \*\*Definición de Métricas:\*\*

- Las métricas son indicadores clave que miden el desempeño de un área específica. Ayudan a cuantificar el progreso hacia objetivos y metas.

2. \*\*Importancia de las Métricas en la Inteligencia de Negocios:\*\*

- Las métricas permiten tomar decisiones basadas en hechos y datos objetivos. Proporcionan una visión clara del rendimiento de la empresa y permiten detectar áreas que necesitan mejora.

3. \*\*Tipos de Métricas:\*\*

- \*\*Métricas Financieras:\*\* Miden el rendimiento financiero de una empresa (ej. ROI, ingresos, ganancias).

- \*\*Métricas de Rendimiento Operativo:\*\* Miden la eficiencia de procesos y operaciones internas (ej. tasa de defectos, productividad).

- \*\*Métricas de Cliente:\*\* Miden la satisfacción y retención de clientes (ej. NPS, tasa de retención).

- \*\*Métricas de Marketing:\*\* Miden la efectividad de las estrategias de marketing (ej. tasa de conversión, costo por adquisición).

4. \*\*Cómo Definir Métricas Relevantes:\*\*

- Las métricas deben estar alineadas con los objetivos estratégicos de la empresa. Es importante definir métricas que sean accionables y que permitan identificar áreas de mejora.

5. \*\*Herramientas para Medir Métricas:\*\*

- Hay muchas herramientas para medir métricas, como Google Analytics para marketing digital, Power BI para inteligencia de negocios, y CRMs como Salesforce para métricas de cliente.

6. \*\*KPIs (Key Performance Indicators):\*\*

- Los KPIs son un tipo de métrica que mide el éxito de una empresa en áreas críticas. Son los indicadores más relevantes y estratégicos para el crecimiento y evolución del negocio.

### Preguntas de Retroalimentación:

1. ¿Por qué son importantes las métricas en el análisis de datos?

2. Menciona un ejemplo de métrica financiera y uno de métrica de cliente.

3. ¿Cuál es la principal diferencia entre una métrica y un KPI?

4. ¿Cómo se pueden definir métricas que sean relevantes para una empresa?

5. ¿Qué tipo de métricas crees que sería más importante para un área de marketing?

Respuestas:

1. Son importantes para medir el desempeño de los datos obtenidos
2. El valor de mercado de una empresa, el costo por adquisision
3. El kpi es solo centrado en los datos mas relevantes para el negocio
4. Analizando el negocio y viendo que sectores necesitan mediciones para ver su evolucion
5. La metrica de impacto en la publicidad y la de costo por obtencion de cliente

**Clase 28: Valores de Referencia**

Los valores de referencia, también conocidos como "benchmarks", son puntos de comparación utilizados para evaluar el rendimiento de un proceso, empresa o sistema en relación con estándares predefinidos o con los resultados de otras entidades similares. Estos valores sirven para determinar si los resultados obtenidos están dentro de los parámetros esperados, son superiores o, por el contrario, requieren mejoras.

#### Subtemas:

1. \*\*Definición de Valores de Referencia:\*\*

- Un valor de referencia es un estándar o un punto de comparación utilizado para medir el rendimiento de procesos, productos o sistemas en función de un criterio o objetivo establecido.

2. \*\*Importancia de los Valores de Referencia:\*\*

- Proporcionan un marco de evaluación que permite identificar oportunidades de mejora y áreas que están por debajo del estándar.

- Ayudan a medir el éxito en relación con competidores, normas industriales o expectativas internas.

3. \*\*Tipos de Valores de Referencia:\*\*

- \*\*Valores Internos:\*\* Establecidos por la misma empresa en función de su propio rendimiento anterior.

- \*\*Valores Externos:\*\* Comparan el rendimiento con el de otras empresas o con estándares de la industria.

- \*\*Valores Históricos:\*\* Comparan el desempeño actual con resultados pasados para evaluar el progreso a lo largo del tiempo.

4. \*\*Cómo Establecer Valores de Referencia:\*\*

- Analizar datos históricos.

- Comparar con competidores o con el promedio de la industria.

- Definir metas claras y medibles que alineen con los objetivos estratégicos de la empresa.

5. \*\*Uso de Valores de Referencia en Inteligencia de Negocios:\*\*

- Permiten detectar qué áreas están funcionando bien y cuáles necesitan intervención.

- Facilitan la comparación de desempeño a lo largo del tiempo o frente a la competencia.

6. \*\*Herramientas para Utilizar Valores de Referencia:\*\*

- Herramientas de análisis como Power BI, Tableau o Google Analytics pueden ayudar a establecer y monitorear valores de referencia.

### Preguntas de Retroalimentación:

1. ¿Qué es un valor de referencia y por qué es importante?

2. Menciona dos tipos de valores de referencia y sus diferencias.

3. ¿Cómo puedes establecer un valor de referencia para un área de marketing?

4. ¿Qué herramienta recomendarías para monitorear valores de referencia y por qué?

5. ¿Cómo ayuda el uso de valores de referencia a mejorar el desempeño de una empresa?

Respuestas

1. Es un valor que se usa para tomar como base en el analisis el cual es importante para comparar el desempeño del negocio
2. Externo e interno, la diferencia es el punto de vista, el interno sirve para analizar el desempeño del negfocio internamente, y el externo para comparar con otras compañias
3. Comparando el desempeño obtenido por el area de marketing ya sea con otra area, con otra empresa o con la misma area atravez del tiempo
4. Recomendaria power bi, no la conozco mucho, pero tioene muy buenas referencias
5. Analizando los datos historicos podemos ver cdonde hay mejoria y donde no, esos puntos donde no se han mejorado deben ser analisados

**Clase 29: Análisis Descriptivo y Estadístico**

El análisis descriptivo y estadístico es una técnica fundamental en el campo del análisis de datos. Permite resumir y describir las características de un conjunto de datos de manera clara y concisa, facilitando su interpretación y proporcionando una visión general del comportamiento de los datos.

#### Subtemas:

1. \*\*Análisis Descriptivo:\*\*

- \*\*Definición:\*\* Se centra en la descripción de los datos mediante el uso de métricas como medias, medianas, modas, rangos y desviaciones estándar.

- \*\*Propósito:\*\* Proporcionar una comprensión básica del comportamiento de los datos sin hacer inferencias más profundas.

- \*\*Métricas comunes:\*\*

- \*\*Media:\*\* El promedio de todos los valores.

- \*\*Mediana:\*\* El valor que divide el conjunto de datos en dos mitades iguales.

- \*\*Moda:\*\* El valor más frecuente en el conjunto de datos.

- \*\*Desviación Estándar:\*\* Medida de la dispersión de los datos en torno a la media.

2. \*\*Análisis Estadístico:\*\*

- \*\*Definición:\*\* Técnica que emplea métodos estadísticos para identificar patrones, relaciones y tendencias en los datos.

- \*\*Propósito:\*\* Hacer inferencias y predicciones basadas en los datos analizados.

- \*\*Métodos comunes:\*\*

- \*\*Distribución Normal:\*\* Distribución de probabilidad que se utiliza para modelar muchos fenómenos naturales.

- \*\*Pruebas de Hipótesis:\*\* Permiten validar suposiciones sobre los datos mediante un proceso estadístico.

- \*\*Correlación:\*\* Evalúa la relación entre dos variables.

3. \*\*Importancia del Análisis Descriptivo y Estadístico:\*\*

- \*\*Análisis Descriptivo:\*\* Ayuda a entender los datos de forma rápida y fácil, sin necesidad de realizar análisis complejos.

- \*\*Análisis Estadístico:\*\* Proporciona una base para tomar decisiones fundamentadas y basadas en datos.

4. \*\*Herramientas Comunes para el Análisis Descriptivo y Estadístico:\*\*

- Excel, Python (bibliotecas como `pandas`, `numpy` y `scipy`), R, SPSS y herramientas de Business Intelligence como Power BI o Tableau.

5. \*\*Aplicación en Inteligencia de Negocios:\*\*

- El análisis descriptivo permite tener una visión clara del rendimiento pasado, mientras que el análisis estadístico facilita la predicción de futuros comportamientos.

### Preguntas de Retroalimentación:

1. ¿Cuál es la principal diferencia entre el análisis descriptivo y el análisis estadístico?

2. ¿Qué es la desviación estándar y por qué es importante en el análisis descriptivo?

3. Menciona dos métodos de análisis estadístico y su propósito.

4. ¿Por qué es útil el análisis descriptivo en la inteligencia de negocios?

5. ¿Qué herramientas recomendarías para realizar un análisis estadístico completo?

Respuestas

1. Analisis estadistico se centra en analisar los numeros  
   analisis descriptivo se sentra en tomar los datos y analizarlos de manera mas profunda y real
2. Es la diferencia que van tomando los datos con respecto a la media
3. Hipotesis: con base en los datos y con un analisi profundo tratar de hacer predicciones  
   definicion tomar los datos
4. Poruqe con el analisi a fondo que hace podemos tomar mas deciciones sobre los datos
5. Recomendaria python y power bi

**Clase 30: Staging**

El término \*staging\* se refiere a una fase clave dentro del proceso de extracción, transformación y carga (ETL) de datos. Se trata de un área temporal donde se almacenan los datos antes de ser transformados y cargados en un sistema de destino, como un Data Warehouse. El \*staging\* permite realizar la limpieza, validación y transformación de los datos sin afectar el sistema de destino.

#### Subtemas:

1. \*\*Definición de Staging:\*\*

- \*\*Descripción:\*\* Es un área de almacenamiento temporal utilizada para preparar los datos antes de su integración en el Data Warehouse.

- \*\*Propósito:\*\* Garantizar que los datos estén limpios, válidos y listos para ser procesados, minimizando los errores en el sistema final.

2. \*\*Características del Staging:\*\*

- \*\*Temporalidad:\*\* Los datos en el \*staging\* suelen ser volátiles, lo que significa que permanecen solo durante el proceso de transformación.

- \*\*Transformación:\*\* Los datos pueden ser transformados en este espacio antes de ser cargados.

- \*\*Seguridad:\*\* A menudo, el \*staging\* también proporciona una capa de seguridad para evitar que datos incorrectos o incompletos se carguen en el sistema principal.

3. \*\*Ventajas del Staging:\*\*

- \*\*Optimización del proceso ETL:\*\* Ayuda a evitar errores y asegurarse de que solo los datos correctos y limpios sean cargados en el Data Warehouse.

- \*\*Flexibilidad:\*\* Permite realizar múltiples operaciones de transformación antes de cargar los datos.

- \*\*Mejor rendimiento:\*\* Reduce el impacto en los sistemas de producción, ya que la mayoría de las transformaciones y validaciones se realizan fuera del sistema principal.

4. \*\*Desventajas del Staging:\*\*

- \*\*Requiere almacenamiento adicional:\*\* Se necesita espacio adicional para almacenar los datos temporalmente.

- \*\*Esfuerzo en gestión:\*\* El proceso de \*staging\* requiere una correcta planificación y gestión para evitar problemas de seguridad y rendimiento.

5. \*\*Ejemplo de Uso de Staging en Inteligencia de Negocios:\*\*

- Imagina una empresa que extrae datos de varias fuentes como hojas de cálculo, bases de datos y sistemas externos. Estos datos se almacenan temporalmente en una capa de \*staging\*, donde se validan y transforman antes de cargarlos en el sistema de destino.

### Preguntas de Retroalimentación:

1. ¿Qué es el \*staging\* y cuál es su principal función en el proceso ETL?

2. Menciona una ventaja clave del uso de \*staging\* en el manejo de datos.

3. ¿Qué diferencia existe entre los datos en el \*staging\* y los datos en un Data Warehouse?

4. ¿Cuáles son algunos desafíos o desventajas del \*staging\*?

5. ¿Cómo contribuye el \*staging\* a mejorar la calidad de los datos en una empresa?

Respuestas

1. El staging es un almacen temporal donde se mantienen los datos durante el proceso de limpieza
2. La fexibilidad que se tiene al poder mantener los datos en un alamacen aparte del alamcen de datos y asi evitar errores
3. Los datos en el estagin estan siendo trabajados en su limpieza y apropiacion de uso, los datos en el data warehouse, estan listos para su uso
4. Mayo espacio de almacenamieto y mayor gestion
5. Al tener un espacio idepediente de trabajo para los datos, prepara elespacio de uso de los datos de una manera diferente

**Clase 31: Data Marts**

#### \*\*1. Introducción a los Data Marts\*\*

Un \*\*Data Mart\*\* es una versión simplificada y más específica de un almacén de datos (\*\*Data Warehouse\*\*), centrada en un área funcional o un equipo de una organización. Los data marts se construyen para satisfacer necesidades particulares de análisis de datos de un grupo específico de usuarios.

#### \*\*2. Diferencias entre Data Marts y Data Warehouse\*\*

- \*\*Alcance\*\*: Los \*\*Data Marts\*\* están enfocados en un tema específico (por ejemplo, ventas, marketing, finanzas), mientras que los \*\*Data Warehouse\*\* abarcan toda la organización.

- \*\*Tamaño\*\*: Un \*\*Data Mart\*\* suele ser más pequeño que un \*\*Data Warehouse\*\*, ya que contiene solo los datos relevantes para el área específica.

- \*\*Propósito\*\*: Los \*\*Data Marts\*\* permiten análisis más rápidos y específicos, mientras que los \*\*Data Warehouse\*\* son más globales y contienen información integral.

- \*\*Tiempo de implementación\*\*: Los \*\*Data Marts\*\* suelen ser más rápidos de implementar debido a su menor tamaño y alcance.

#### \*\*3. Tipos de Data Marts\*\*

Existen dos tipos principales de \*\*Data Marts\*\*:

- \*\*Data Mart Dependiente\*\*: Se extrae de un \*\*Data Warehouse\*\* ya existente. Los datos se filtran y seleccionan del \*\*Data Warehouse\*\* para crear un subconjunto relevante.

- \*\*Data Mart Independiente\*\*: Es un sistema autónomo que no depende de un \*\*Data Warehouse\*\* más grande. Estos se construyen desde cero y suelen ser utilizados por pequeñas empresas o proyectos con necesidades específicas.

#### \*\*4. Ventajas de los Data Marts\*\*

- \*\*Menor complejidad\*\*: Dado que los \*\*Data Marts\*\* tienen un alcance limitado, suelen ser más simples de gestionar y de implementar.

- \*\*Costos reducidos\*\*: Al ser más pequeños, son menos costosos en términos de almacenamiento y recursos computacionales.

- \*\*Rápido acceso\*\*: Los \*\*Data Marts\*\* permiten obtener información específica y relevante para análisis rápidamente.

- \*\*Adaptabilidad\*\*: Se pueden personalizar para necesidades concretas de un departamento o área de negocio, como ventas o marketing.

#### \*\*5. Desventajas de los Data Marts\*\*

- \*\*Fragmentación de datos\*\*: Al tener múltiples \*\*Data Marts\*\* para diferentes áreas, puede surgir inconsistencia en los datos si no se gestionan adecuadamente.

- \*\*Duplicación\*\*: A veces, se duplican datos entre \*\*Data Marts\*\*, lo que puede generar problemas de sincronización y almacenamiento innecesario.

- \*\*Poca flexibilidad\*\*: Al estar diseñados para un propósito específico, pueden no ser lo suficientemente flexibles para adaptarse a otros tipos de análisis.

#### \*\*6. Implementación de un Data Mart\*\*

- \*\*Identificación de necesidades\*\*: Se identifican las áreas que requieren análisis específicos, como ventas o marketing.

- \*\*Definición de fuentes de datos\*\*: Se seleccionan los datos relevantes de las bases de datos o del \*\*Data Warehouse\*\*.

- \*\*Diseño y construcción\*\*: Se construye el \*\*Data Mart\*\* seleccionando solo los datos necesarios para el análisis requerido.

- \*\*Carga de datos\*\*: Se extraen, transforman y cargan los datos en el \*\*Data Mart\*\*.

- \*\*Mantenimiento\*\*: Es fundamental mantener el \*\*Data Mart\*\* actualizado con los datos más recientes y asegurarse de que los usuarios tengan acceso a la información correcta.

---

### \*\*Preguntas para Retroalimentación\*\*

1. ¿Cuál es la principal diferencia entre un Data Mart y un Data Warehouse?

2. Menciona dos tipos de Data Marts y explica brevemente en qué se diferencian.

3. ¿Cuáles son algunas ventajas de utilizar Data Marts en una organización?

4. ¿Por qué el uso de múltiples Data Marts puede causar fragmentación de datos?

5. ¿Cuál es el primer paso en la implementación de un Data Mart y por qué es crucial?

---

Una vez que revises esta clase, ¿te gustaría avanzar a la siguiente o realizar ejercicios prácticos sobre Data Marts?

Respuestas

1. la principaldiferencia es el tamaño y sus dependencias, algunos data marts dependen del data warehouse y su tamaño siempre será menor

2. dependiente e independiente, el dependiente, depende del data warehouse y el independiente no necesariamente

3. tener mas especificidad de uso y un enfoque mas dedicado a la sección de la organización en la que se quiere implemntar

4. puede generar redundacia en algunos casos y repetición de datos

5. identficacion de necesidades y es importantes puesto que con el se hace el diseño del mismo datamart para ver que aspectos abarcara

¡Excelentes respuestas! Sigamos avanzando con la siguiente clase:

---

**Clase 32: OLAP y OLTP**

#### \*\*1. Introducción a OLAP y OLTP\*\*

- \*\*OLAP\*\* (Online Analytical Processing) y \*\*OLTP\*\* (Online Transaction Processing) son dos enfoques distintos para gestionar y procesar datos en bases de datos.

#### \*\*2. ¿Qué es OLTP?\*\*

\*\*OLTP\*\* es un sistema diseñado para gestionar transacciones de forma rápida y eficiente. Es utilizado principalmente en sistemas donde las transacciones diarias son frecuentes, como en las bases de datos de ventas, sistemas bancarios y aplicaciones de reserva.

- \*\*Características\*\*:

  - Procesamiento rápido de transacciones.

  - Mantiene la consistencia de los datos.

  - Se utiliza para actividades cotidianas como inserciones, actualizaciones y eliminaciones.

  - La base de datos está altamente normalizada para evitar redundancias.

#### \*\*3. ¿Qué es OLAP?\*\*

\*\*OLAP\*\* es un sistema orientado al análisis de grandes volúmenes de datos. Se utiliza principalmente para hacer consultas complejas, análisis y generación de informes basados en datos históricos.

- \*\*Características\*\*:

  - Está orientado al análisis de datos.

  - Permite realizar consultas complejas sobre grandes volúmenes de datos.

  - Los datos suelen estar desnormalizados para mejorar el rendimiento de las consultas.

  - Se utiliza para la toma de decisiones estratégicas.

#### \*\*4. Diferencias clave entre OLAP y OLTP\*\*

- \*\*Finalidad\*\*:

  - OLTP: Orientado a la gestión de transacciones diarias.

  - OLAP: Orientado al análisis de datos históricos y a la toma de decisiones.

- \*\*Velocidad\*\*:

  - OLTP: Enfocado en el procesamiento rápido de muchas transacciones simples.

  - OLAP: Enfocado en consultas complejas que pueden requerir más tiempo.

- \*\*Estructura de la base de datos\*\*:

  - OLTP: Altamente normalizada para minimizar la redundancia y optimizar las transacciones.

  - OLAP: Desnormalizada para optimizar el rendimiento de las consultas.

#### \*\*5. Uso de OLAP y OLTP en una organización\*\*

- \*\*OLTP\*\* se utiliza en operaciones diarias, como procesar ventas, registrar transacciones bancarias o manejar inventarios.

- \*\*OLAP\*\* se utiliza para generar informes de ventas, analizar tendencias de clientes, y realizar predicciones basadas en datos históricos.

#### \*\*6. Ejemplo de sistemas OLTP y OLAP\*\*

- \*\*OLTP\*\*: Un sistema de punto de venta (POS) en un supermercado, que procesa transacciones de ventas.

- \*\*OLAP\*\*: Un sistema de análisis que genera informes mensuales sobre las ventas en todas las sucursales del supermercado, y que permite analizar las tendencias.

---

### \*\*Preguntas para Retroalimentación\*\*

1. ¿Cuál es la función principal de un sistema OLTP?

2. ¿En qué tipo de sistemas es más común encontrar OLAP?

3. Menciona dos diferencias clave entre OLTP y OLAP.

4. ¿Por qué se dice que las bases de datos OLTP suelen estar altamente normalizadas?

5. ¿En qué áreas de una organización sería más útil implementar un sistema OLAP?

 Respuestas

1. Gestionar transacciones diarias
2. En sistemas de análisis y que usan consultas muy complejas
3. Oltp se usa para tranaacciones, tiene mayor normalización, olap, se usa para análisis y tiene baja normalización
4. Por que se usan mayormente para el registri transaccional
5. En el área de ventas, marketing y finanzas

**Clase 33: ¿Qué es un Data Warehouse?**

Un \*\*Data Warehouse\*\* (o almacén de datos) es un sistema de almacenamiento centralizado que permite organizar, gestionar, y analizar grandes cantidades de datos que provienen de diversas fuentes. Este almacén de datos está diseñado para ayudar a las organizaciones a obtener una visión general e histórica de sus operaciones, lo que facilita la toma de decisiones informadas. Los datos en un Data Warehouse son almacenados de manera estructurada, organizados en tablas y pueden ser consultados de manera eficiente para realizar análisis avanzados.

#### Características principales de un Data Warehouse:

1. \*\*Integración de datos\*\*: Reúne datos de diferentes fuentes (bases de datos, sistemas operacionales, hojas de cálculo, etc.) en un solo lugar, proporcionando una visión unificada.

2. \*\*Orientado a temas\*\*: Está estructurado en torno a áreas temáticas como ventas, finanzas, marketing, etc., lo que facilita el análisis específico de cada área.

3. \*\*No volátil\*\*: Los datos en un Data Warehouse no cambian constantemente. Una vez almacenados, no se modifican, lo que garantiza su estabilidad para análisis históricos.

4. \*\*Histórico\*\*: Almacena grandes cantidades de datos a lo largo del tiempo, permitiendo realizar análisis comparativos y estudiar tendencias en periodos prolongados.

5. \*\*Optimizado para consultas y análisis\*\*: El diseño del Data Warehouse está enfocado en el rendimiento eficiente de consultas complejas, que pueden implicar agregaciones y cálculos sobre grandes volúmenes de datos.

#### Componentes de un Data Warehouse:

1. \*\*ETL (Extract, Transform, Load)\*\*: Es el proceso encargado de extraer los datos de las fuentes, transformarlos (limpieza, normalización, etc.) y cargarlos en el Data Warehouse. Las fases del proceso ETL son:

- \*\*Extracción\*\*: Se obtienen los datos de diversas fuentes (bases de datos relacionales, sistemas OLTP, hojas de cálculo, etc.).

- \*\*Transformación\*\*: Se limpian y formatean los datos para asegurar que sean consistentes y adecuados para el análisis.

- \*\*Carga\*\*: Se almacenan los datos en el Data Warehouse.

2. \*\*Metadatos\*\*: Son los datos sobre los datos, que describen el contenido, estructura, y origen de los datos almacenados en el Data Warehouse. Ayudan a los usuarios a comprender y utilizar eficientemente el almacén de datos.

3. \*\*Motor de consulta y análisis\*\*: Permite a los usuarios ejecutar consultas y realizar análisis sobre los datos almacenados, aprovechando herramientas como SQL, herramientas OLAP, y más.

4. \*\*Dashboards e informes\*\*: Las herramientas de inteligencia de negocios suelen estar conectadas a los Data Warehouses, proporcionando visualizaciones y reportes que permiten interpretar los resultados del análisis de datos.

### Importancia del Data Warehouse:

Un Data Warehouse facilita la \*\*toma de decisiones estratégicas\*\* mediante el análisis profundo de los datos históricos de una organización. Es fundamental en procesos como la \*\*inteligencia de negocios\*\* y el \*\*análisis de grandes volúmenes de datos\*\*, ya que permite realizar consultas complejas y obtener información valiosa de manera eficiente.

---

### Preguntas para retroalimentación:

1. ¿Qué diferencia principal existe entre un Data Warehouse y una base de datos transaccional como un sistema OLTP?

2. ¿Cuáles son las tres fases del proceso ETL en un Data Warehouse?

3. ¿Por qué se dice que los datos en un Data Warehouse son no volátiles?

4. ¿Cómo ayuda un Data Warehouse en la toma de decisiones estratégicas dentro de una organización?

5. Menciona dos ventajas de utilizar un Data Warehouse en lugar de sistemas operacionales tradicionales para el análisis de datos.

Respuestas

1. el sistema transaccional no es muy útil para análisis históricos, además que sus datos suelen ser redundantes pues no han tenido un tratamiento de limpieza

2. extraemos los datos de las diversas fuentes  
limpiamos los datos para que sean trabajables  
y luego los cargamos

3. suelen ser históricos, por lo que no varian demasiado

4. con la creación de dashboard con base en el data warehouse, se pueden tomar decicioones estratégicas

5. la limpieza de los datos y la disposición eficaz, el histórico de los datos y su poca volatilidad

Vamos a continuar con el siguiente tema que es \*\*Esquemas de un Almacén de Datos\*\*.

**Clase 35: Esquemas de un Almacén de Datos**

Cuando hablamos de un Data Warehouse (almacén de datos), los \*\*esquemas\*\* son las estructuras que definen cómo se organizan y almacenan los datos en el sistema. Estos esquemas son esenciales para estructurar la información de una manera que facilite la consulta y el análisis.

#### 1. ¿Qué es un esquema en un Data Warehouse?

Un \*\*esquema\*\* en un Data Warehouse es una representación visual y estructural que define la forma en que los datos se organizan y cómo se relacionan entre las diferentes tablas. Los esquemas son fundamentales para optimizar el rendimiento de las consultas y organizar los datos de manera lógica.

#### 2. Tipos de Esquemas

Existen diferentes tipos de esquemas que se pueden utilizar en un Data Warehouse, cada uno con sus características específicas. Los más comunes son:

##### 2.1. Esquema Estrella (\*\*Star Schema\*\*)

El \*\*esquema estrella\*\* es uno de los más sencillos y utilizados en los Data Warehouses. Su estructura consiste en:

- \*\*Tabla de hechos (fact table)\*\*: Contiene los datos transaccionales o métricas clave que se quieren analizar (por ejemplo, ventas, ingresos, cantidades).

- \*\*Tablas de dimensión (dimension tables)\*\*: Contienen descripciones detalladas de los datos transaccionales, como productos, fechas, clientes, ubicaciones, etc.

En este esquema, todas las tablas de dimensión están conectadas directamente a la tabla de hechos, formando una estructura visual que se asemeja a una estrella.

##### 2.2. Esquema Copo de Nieve (\*\*Snowflake Schema\*\*)

El \*\*esquema copo de nieve\*\* es una versión normalizada del esquema estrella. En lugar de tener tablas de dimensión directamente conectadas a la tabla de hechos, las tablas de dimensión se normalizan en varias subtablas. Esto significa que se eliminan redundancias dentro de las tablas de dimensión.

Este esquema es más complejo, pero puede ser más eficiente en términos de espacio de almacenamiento.

##### 2.3. Esquema Galaxia o Constelación (\*\*Galaxy Schema\*\*)

El \*\*esquema galaxia\*\* (también llamado \*\*constelación de hechos\*\*) es una versión más avanzada del esquema estrella. En lugar de tener una única tabla de hechos, puede haber varias tablas de hechos que comparten dimensiones comunes. Esto es útil cuando se quieren analizar datos de diferentes áreas del negocio, pero que tienen dimensiones en común, como ventas y marketing.

#### 3. ¿Cuándo utilizar cada tipo de esquema?

- \*\*Esquema Estrella\*\*: Se utiliza cuando se necesita simplicidad en el diseño y las consultas deben ejecutarse rápidamente. Es ideal para organizaciones que prefieren una estructura fácil de entender y optimizada para consultas de datos.

- \*\*Esquema Copo de Nieve\*\*: Se utiliza cuando se desea optimizar el almacenamiento y reducir la redundancia. Este esquema es útil cuando el tamaño del Data Warehouse es considerable y se necesita eficiencia en el uso del espacio.

- \*\*Esquema Galaxia\*\*: Se usa cuando se requieren múltiples análisis complejos a partir de diferentes tablas de hechos, pero con dimensiones compartidas. Es útil para grandes organizaciones con diversas áreas de análisis.

#### 4. Comparación de los Esquemas

- \*\*Esquema Estrella\*\*: Fácil de usar, rápido para consultas, pero puede tener redundancia en las tablas de dimensión.

- \*\*Esquema Copo de Nieve\*\*: Reduce la redundancia, más complicado de entender y mantener, pero optimiza el espacio.

- \*\*Esquema Galaxia\*\*: Flexible y poderoso, pero puede ser complejo en su diseño y gestión.

---

### 5 Preguntas de Retroalimentación:

1. ¿Cuál es la principal diferencia entre el esquema estrella y el esquema copo de nieve?

2. ¿Qué ventajas ofrece el esquema galaxia en comparación con el esquema estrella?

3. ¿Por qué las tablas de dimensión en un esquema estrella pueden tener redundancias?

4. ¿En qué situaciones se recomienda utilizar un esquema copo de nieve en lugar de un esquema estrella?

5. ¿Cómo se relacionan las tablas de hechos y las tablas de dimensión en el esquema estrella?

Déjame saber cuando tengas las respuestas o si deseas aclaraciones adicionales.

Respuestas

1. la principal diferencia consiste en la normalización de las dimensiones

2. es útil para análisis mas sofisticados y para organizaciones mas robustas

3. al ser menos tablas, puede injerir en necesitar mas datos dentro de las dimensiones existentes para dar mas contexto a los hechos

4. en el caso de necesitar normalizar mas los datos y optimizar el espacio de disco

5. con las llaves subrrugagadas, dan relación entre la llave primaria y la llave foránea

¡Sigamos adelante! El próximo tema es \*\*Reporting\*\* en el contexto de \*\*Business Intelligence (BI)\*\*. A continuación te enseño los puntos clave del tema.

**36. Reporting**

#### ¿Qué es el Reporting?

El reporting en Business Intelligence se refiere a la recopilación, organización y presentación de datos, con el objetivo de proporcionar información útil para la toma de decisiones en una organización. Los reportes permiten a los usuarios visualizar los datos de manera clara y concisa, facilitando el análisis y la interpretación de la información.

#### Tipos de Reportes:

1. \*\*Reportes Operacionales\*\*: Utilizados para el seguimiento de las actividades diarias de una empresa. Estos reportes suelen enfocarse en aspectos como las ventas diarias, inventarios, o ingresos por sucursales.

2. \*\*Reportes Analíticos\*\*: Ayudan a comprender las tendencias a largo plazo o el desempeño global de una empresa. Estos reportes analizan los datos históricos para identificar patrones o realizar proyecciones futuras.

3. \*\*Dashboards\*\*: Herramientas gráficas que muestran KPIs (indicadores clave de desempeño) en tiempo real, permitiendo una interpretación rápida y directa de la información. Son altamente personalizables y muestran métricas como gráficos de barras, gráficos circulares, tablas, entre otros.

4. \*\*Informes Estándar vs. Informes Ad-hoc\*\*:

- \*\*Informes Estándar\*\*: Son predefinidos y distribuidos periódicamente a los usuarios. Están diseñados para cubrir necesidades de información comunes y recurrentes.

- \*\*Informes Ad-hoc\*\*: Son generados bajo demanda, permitiendo al usuario realizar consultas específicas según sus necesidades particulares en un momento dado.

#### Herramientas de Reporting:

1. \*\*Power BI\*\*: Ofrece la posibilidad de crear dashboards interactivos y reportes detallados mediante gráficos, filtros y visualizaciones dinámicas.

2. \*\*Tableau\*\*: Es popular por sus capacidades de visualización de datos, permitiendo crear reportes interactivos con facilidad.

3. \*\*Google Data Studio\*\*: Es una herramienta gratuita que permite crear dashboards e informes interactivos.

4. \*\*Crystal Reports\*\*: Herramienta tradicional que permite la creación de informes avanzados con múltiples opciones de personalización.

5. \*\*QlikView/Qlik Sense\*\*: Herramientas de BI que permiten la creación de reportes y dashboards para la toma de decisiones basada en datos.

#### Características Clave de un Buen Reporte:

- \*\*Claridad\*\*: Debe ser comprensible por todos los usuarios, independientemente de su nivel técnico.

- \*\*Relevancia\*\*: Debe proporcionar información útil que esté alineada con los objetivos del negocio.

- \*\*Precisión\*\*: Los datos presentados deben ser precisos y actualizados.

- \*\*Acción\*\*: Los reportes deben ser fácilmente interpretable para permitir a los usuarios tomar decisiones basadas en los datos.

#### Importancia del Reporting:

El reporting permite a las organizaciones monitorear el rendimiento, evaluar estrategias y tomar decisiones informadas. Ayuda a detectar problemas, medir el éxito de las iniciativas, y optimizar los procesos operacionales y estratégicos.

### Preguntas de retroalimentación:

1. ¿Qué diferencia hay entre un reporte operacional y un reporte analítico?

2. ¿Qué es un dashboard y para qué se utiliza en Business Intelligence?

3. ¿Cuál es la ventaja de utilizar herramientas como Power BI o Tableau para el reporting?

4. ¿Qué características debe tener un buen reporte para ser efectivo?

5. ¿En qué casos sería útil un informe ad-hoc en lugar de un informe estándar?

Déjame saber si quieres profundizar en algo más o si prefieres continuar con el siguiente tema.

Respuestas

1. el reporte operacional se utiliza para ver el desarrollo diario de la operación, y el reporte analítico nos ayuda a ver un histórico de los datos

2. el dashboard es un informe grafico que nos muestra métricas de manera mas interactiva y sirve para que los datos sean comprensibles por mayores personas

3. son fáciles de utilizar y tiene muy buenas herramientas para crear reportes

4. debe ser claro y fácil de entender. Debe ser relevante, precico y poderce aplicar

5. en caso de que se quiera un informe mas personalizado y a petición del usuario  
  
¡Respuestas correctas! Has mostrado una sólida comprensión de los conceptos clave de \*\*Reporting\*\* en Business Intelligence. Ahora que tienes claro cómo funcionan los reportes y su importancia, sigamos adelante con el próximo tema.

**37. Analytics**

#### ¿Qué es Analytics?

Analytics, o análisis de datos, se refiere al proceso de examinar datos con el objetivo de obtener información que pueda ser utilizada para tomar decisiones informadas. Utiliza una combinación de estadísticas, modelos matemáticos y técnicas computacionales para identificar patrones, tendencias y relaciones en los datos.

#### Tipos de Analytics:

1. \*\*Descriptivo\*\*: Este tipo de análisis se enfoca en describir y resumir los datos históricos para obtener una visión clara de lo que ha ocurrido. Responde a preguntas como "¿Qué pasó?" o "¿Cuál fue el desempeño el mes pasado?".

2. \*\*Diagnóstico\*\*: Va un paso más allá del análisis descriptivo, ya que busca comprender por qué sucedieron ciertos eventos. Responde a preguntas como "¿Por qué ocurrió este resultado?".

3. \*\*Predictivo\*\*: Utiliza técnicas estadísticas y de machine learning para hacer proyecciones sobre lo que podría ocurrir en el futuro. Responde a preguntas como "¿Qué pasará si las condiciones actuales continúan?".

4. \*\*Prescriptivo\*\*: Va más allá de la predicción, ya que proporciona recomendaciones sobre qué decisiones tomar para lograr ciertos resultados. Responde a preguntas como "¿Qué debo hacer para optimizar el resultado?".

5. \*\*Cognitivo\*\*: Es una forma más avanzada de análisis que implica la inteligencia artificial (IA) y el procesamiento de lenguaje natural para simular el proceso de toma de decisiones humanas.

#### Herramientas para Analytics:

- \*\*Python y R\*\*: Ambos lenguajes de programación son ampliamente utilizados en el análisis de datos por sus poderosas bibliotecas y funcionalidades para análisis estadístico y machine learning.

- \*\*Power BI y Tableau\*\*: No solo para reporting, también son útiles para realizar análisis más profundos al incluir filtros, métricas personalizadas y gráficos interactivos.

- \*\*Google Analytics\*\*: Específicamente enfocado en analizar el comportamiento de usuarios en sitios web, permitiendo entender el tráfico y la conversión.

- \*\*Excel\*\*: Aunque básico, sigue siendo una herramienta poderosa para realizar análisis simples de datos, visualización y manipulación de conjuntos de datos.

#### Importancia del Analytics:

El análisis de datos permite a las empresas mejorar su toma de decisiones, identificar nuevas oportunidades de negocio, optimizar procesos y entender mejor las preferencias de sus clientes. Además, permite prever posibles problemas y evaluar el impacto de estrategias o políticas empresariales antes de implementarlas.

#### Aplicaciones del Analytics:

- \*\*Ventas y Marketing\*\*: Analizar el comportamiento de los consumidores, predecir las tendencias de compra y optimizar campañas publicitarias.

- \*\*Finanzas\*\*: Previsión financiera, análisis de riesgo y detección de fraudes.

- \*\*Operaciones\*\*: Optimización de la cadena de suministro y mejora en la eficiencia operativa.

- \*\*Recursos Humanos\*\*: Evaluar el rendimiento de los empleados y mejorar los procesos de contratación.

### Preguntas de retroalimentación:

1. ¿Qué tipo de análisis de datos respondería a la pregunta "¿Qué pasará si las condiciones actuales continúan?"?

2. ¿Qué diferencia hay entre el análisis descriptivo y el análisis predictivo?

3. ¿Cuál es la principal función de un análisis prescriptivo?

4. ¿Qué herramientas recomendarías para realizar análisis de datos más avanzados?

5. ¿Cómo puede beneficiar el análisis de datos al área de marketing de una empresa?

Respuestas

1. Analisis predictivo, nos permite tomar los datos y con base en el hacer predicciones futuras
2. El descriptivo, nos describe la situacion actual y el predictivo nos ayuda a determinar condiciones futuras
3. Su funcion se centra en ayudarnos a implementar los datos futuros, a tomar analisis y consejo sobre ellos de manera mas apta
4. Python, tiene librerias muy robustas, power bi, tiene una interfaz sencilla y amigable
5. Su potencial nos sirve para analizar comportamientos de los clientes, describir la curva de ventas y la efectividad de la publicidad

**Clase 34: \*\*Sistemas OLAP y OLTP\*\***

En el ámbito de las bases de datos y la gestión de datos empresariales, los sistemas OLAP (Online Analytical Processing) y OLTP (Online Transaction Processing) juegan roles fundamentales, pero muy diferentes. Mientras que OLTP está diseñado para gestionar operaciones transaccionales en tiempo real, OLAP se centra en el análisis de datos históricos y permite a las empresas tomar decisiones estratégicas basadas en esos análisis.

#### \*\*1. Definición de OLAP y OLTP\*\*

##### \*\*1.1 OLTP (Online Transaction Processing)\*\*

El procesamiento de transacciones en línea (OLTP) se refiere a los sistemas que gestionan las transacciones diarias de una empresa. Estas transacciones pueden incluir desde ventas de productos en una tienda en línea, hasta registros de movimientos en una cuenta bancaria. El objetivo principal de OLTP es garantizar que las operaciones se realicen de manera rápida, precisa y consistente. Este sistema está diseñado para:

- \*\*Realizar múltiples operaciones de escritura\*\*: Como insertar, actualizar o eliminar datos en la base de datos, con una respuesta rápida.

- \*\*Garantizar la integridad de las transacciones\*\*: OLTP sigue las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), que aseguran que cada transacción se complete correctamente o se revierta si hay algún error.

- \*\*Ejemplos comunes\*\*: Sistemas de reservas de vuelos, cajeros automáticos (ATM), ventas en línea, gestión de inventarios en supermercados.

##### \*\*1.2 OLAP (Online Analytical Processing)\*\*

El procesamiento analítico en línea (OLAP) se enfoca en la \*\*analítica de datos históricos\*\* para apoyar la toma de decisiones. En lugar de realizar operaciones transaccionales, OLAP se encarga de ejecutar consultas complejas sobre grandes volúmenes de datos, permitiendo el análisis de información para detectar patrones, realizar predicciones y tomar decisiones informadas. OLAP es ideal para:

- \*\*Consultas intensivas y complejas\*\*: Estas incluyen agrupaciones, cálculos agregados, comparaciones entre periodos y análisis de tendencias.

- \*\*Soportar decisiones estratégicas\*\*: Los datos son analizados para determinar cómo optimizar procesos, predecir demandas futuras o identificar oportunidades de mercado.

- \*\*Estructuras multidimensionales\*\*: OLAP organiza los datos en "cubos" que permiten consultas desde múltiples dimensiones, como tiempo, ubicación y producto.

#### \*\*2. Diferencias clave entre OLAP y OLTP\*\*

|  |  |  |
| --- | --- | --- |
| Característica | OLTP | OLAP |
| Propósito | Gestión de transacciones cotidianas | Análisis de datos históricos |
| Consultas | Simples, de lectura y escritura rápidas | Complejas, enfocadas en análisis y agregación |
| Volumen de datos | Pequeño, enfocado en datos actuales | Gran volumen de datos históricos |
| Estructura de datos | Relacional (tablas normales) | Multidimensional (cubos de datos) |
| Usuarios | Usuarios operativos y sistemas de transacción | Analistas y personal de toma de decisiones |
| Tiempo de respuesta | Rápido (milisegundos) | Puede ser más lento, debido a la complejidad de las consultas |
| Integridad transaccional | Alta (Propiedades ACID) | No es un enfoque principal |
| Ejemplo | Registro de una compra en un supermercado | Análisis de ventas para un informe trimestral |

#### \*\*3. Ventajas y Desventajas de OLAP y OLTP\*\*

##### \*\*3.1 OLTP\*\*

- \*\*Ventajas\*\*:

- \*\*Velocidad\*\*: OLTP está optimizado para manejar grandes volúmenes de transacciones en tiempo real con tiempos de respuesta rápidos.

- \*\*Consistencia\*\*: Gracias a las propiedades ACID, OLTP asegura que las transacciones sean consistentes y confiables.

- \*\*Escalabilidad\*\*: Los sistemas OLTP pueden crecer fácilmente para manejar más usuarios y transacciones simultáneamente.

- \*\*Desventajas\*\*:

- \*\*Capacidad de análisis limitada\*\*: OLTP no está diseñado para manejar consultas complejas ni para el análisis histórico, lo que limita su uso para la toma de decisiones estratégicas.

- \*\*No apto para grandes volúmenes históricos\*\*: OLTP trabaja con datos actuales y transaccionales, pero no es adecuado para almacenar y consultar grandes volúmenes de datos históricos.

##### \*\*3.2 OLAP\*\*

- \*\*Ventajas\*\*:

- \*\*Capacidad de análisis avanzado\*\*: OLAP está optimizado para consultas complejas que implican grandes volúmenes de datos y múltiples dimensiones.

- \*\*Soporte para decisiones estratégicas\*\*: Al ofrecer capacidades de análisis en profundidad, OLAP ayuda a los gerentes y analistas a tomar decisiones informadas basadas en datos históricos.

- \*\*Estructura de datos multidimensional\*\*: Los cubos OLAP permiten visualizar datos desde varias perspectivas simultáneamente, lo que facilita análisis más detallados y complejos.

- \*\*Desventajas\*\*:

- \*\*Consultas más lentas\*\*: Las consultas OLAP suelen ser más lentas que las transacciones OLTP debido a la complejidad del análisis.

- \*\*Costos de almacenamiento y procesamiento\*\*: Dado que OLAP maneja grandes volúmenes de datos históricos, requiere más almacenamiento y poder de procesamiento, lo que puede ser costoso.

#### \*\*4. Casos de Uso Combinados de OLAP y OLTP\*\*

Muchas organizaciones utilizan ambos sistemas de manera complementaria. Mientras OLTP maneja las operaciones cotidianas en tiempo real, OLAP se encarga del análisis histórico y estratégico. Un \*\*caso típico\*\* es el de un \*\*sistema de comercio electrónico\*\*:

- \*\*OLTP\*\*: Cada vez que un cliente realiza una compra en línea, el sistema OLTP se encarga de registrar la transacción, actualizar el inventario y procesar el pago.

- \*\*OLAP\*\*: Posteriormente, los analistas pueden utilizar un sistema OLAP para analizar las tendencias de ventas, identificar los productos más vendidos y realizar predicciones sobre la demanda futura.

#### \*\*5. Ejemplo de Uso de OLAP y OLTP\*\*

- \*\*OLTP en una cadena de supermercados\*\*: Cada vez que un cliente pasa un producto por la caja registradora, el sistema OLTP registra la venta y actualiza el inventario. Esto ocurre en tiempo real para reflejar las existencias de productos con precisión.

- \*\*OLAP para análisis financiero\*\*: Al final de cada trimestre, un analista financiero puede usar OLAP para analizar las ventas de todas las tiendas de la cadena durante los últimos 5 años. El sistema OLAP permite generar informes detallados sobre el rendimiento de ventas, identificar tendencias y oportunidades de crecimiento.

---

### Preguntas para Retroalimentación

1. \*\*¿Cuál es el principal objetivo de un sistema OLTP y por qué es esencial en las operaciones diarias de una empresa?\*\*

2. \*\*¿Por qué OLAP es más adecuado para análisis históricos y consultas complejas en comparación con OLTP?\*\*

3. \*\*¿Cómo se diferencian las estructuras de datos utilizadas en OLAP y OLTP? Explica qué tipo de estructura utiliza cada uno y por qué es importante.\*\*

4. \*\*Menciona un caso práctico en el que una organización utilizaría OLTP y otro en el que usaría OLAP. ¿Cómo podrían complementarse ambos sistemas en la misma organización?\*\*

5. \*\*¿Cuáles son las ventajas y desventajas más notables de OLAP cuando se compara con OLTP? ¿En qué situaciones preferirías usar uno sobre el otro?\*\*

Respuestas

1. Su principal funcion es llevar el manejo de transacciones, al ser un enfoque diseñado para el tratamiento de datos rapidamente, es ideal para este caso
2. Porque el enfoque olap esta diseñado para el analisi de datos historicos y el manejo de gran cantidad de datos
3. La estructura en oltp es de tablas relacionales, no necesita consultas complejas solo un buen almacenamiento y actualizacion rapida, la estructura en olap es de dimensiones y cubos, para ofrecer mayor soporte en el manejo de datos historicos y analisis
4. En el caso de un banco, necesita llevar el sistema oltp para tratar sus trasacciones diarias y usaria el sistema olap para almacenar los datos de clientes y almacenar historicos de las transacciones
5. La ventaja mas notable es su propio enfoque, uno es solo para transacciones rapidas y otro para analisis, usaria oltp para almacenar transacciones que deban ser rapidas y verificar su efectividad, y usaria olap para almacenar y analizar los datos

**Clase 35: Esquemas de un Almacén de Datos**

\*\*Introducción\*\*

En un almacén de datos, los esquemas son una parte fundamental para estructurar cómo se organizan los datos. Los esquemas son utilizados para modelar las relaciones entre los datos y ayudar a optimizar las consultas. Los dos esquemas más comunes en los almacenes de datos son el \*\*esquema en estrella\*\* y el \*\*esquema en copo de nieve\*\*.

#### 1. \*\*Esquema en Estrella (Star Schema)\*\*

El esquema en estrella es el más sencillo y común en los almacenes de datos. Este esquema consiste en una tabla de hechos central que se conecta directamente con varias tablas de dimensiones. Las tablas de hechos contienen datos cuantitativos que representan transacciones o eventos del negocio, como las ventas o los ingresos. Las tablas de dimensiones, por otro lado, contienen atributos que describen el contexto de estos hechos (por ejemplo, productos, clientes, tiempo).

\*\*Características:\*\*

- \*\*Tabla de hechos\*\*: Contiene las medidas numéricas que representan un evento o transacción, como el total de ventas.

- \*\*Tablas de dimensiones\*\*: Proveen el contexto necesario para las consultas sobre los hechos. Ejemplos de dimensiones son el producto, el cliente, el tiempo, etc.

- \*\*Simplicidad\*\*: Los datos se organizan de manera clara, lo que facilita las consultas y la comprensión para los usuarios finales.

- \*\*Consultas rápidas\*\*: Dado que las tablas de dimensiones están denormalizadas (es decir, todos los datos contextuales se encuentran en una sola tabla), las consultas son más rápidas porque no requieren múltiples uniones (joins).

\*\*Ventajas del Esquema en Estrella:\*\*

- \*\*Simplicidad\*\*: Es más fácil de entender y usar, especialmente para usuarios finales.

- \*\*Eficiencia en consultas\*\*: Al tener las dimensiones denormalizadas, las consultas son más rápidas ya que se evitan múltiples joins.

\*\*Desventajas del Esquema en Estrella:\*\*

- \*\*Redundancia\*\*: Los datos en las tablas de dimensiones pueden ser redundantes, ocupando más espacio.

- \*\*Mantenimiento\*\*: Las tablas de dimensiones grandes pueden ser difíciles de mantener.

#### 2. \*\*Esquema en Copo de Nieve (Snowflake Schema)\*\*

El esquema en copo de nieve es una variación del esquema en estrella, donde las tablas de dimensiones están normalizadas. En lugar de tener una sola tabla de dimensión, esta puede dividirse en varias tablas relacionadas entre sí. Por ejemplo, una tabla de dimensión de productos podría dividirse en una tabla para productos y otra para categorías.

\*\*Características:\*\*

- \*\*Normalización\*\*: Las tablas de dimensiones están normalizadas, es decir, se eliminan redundancias dividiendo los datos en múltiples tablas más pequeñas.

- \*\*Estructura más compleja\*\*: A medida que las tablas de dimensiones se dividen en más tablas, el número de joins necesarios en las consultas aumenta, lo que puede hacerlas más lentas.

\*\*Ventajas del Esquema en Copo de Nieve:\*\*

- \*\*Ahorro de espacio\*\*: Al estar normalizadas, las dimensiones ocupan menos espacio en el almacenamiento.

- \*\*Facilidad de mantenimiento\*\*: Es más fácil actualizar los datos y evitar inconsistencias, ya que la información no se repite en las tablas de dimensiones.

\*\*Desventajas del Esquema en Copo de Nieve:\*\*

- \*\*Consultas más lentas\*\*: Al requerir más joins, las consultas pueden ser más complejas y lentas.

- \*\*Complejidad para el usuario final\*\*: Dado que el esquema es más complicado, los usuarios finales pueden tener dificultades para entenderlo.

#### 3. \*\*Comparación entre Esquema en Estrella y Copo de Nieve\*\*

- \*\*Consultas\*\*: El esquema en estrella suele ser más rápido en consultas debido a la falta de joins. En contraste, el esquema en copo de nieve puede ser más lento debido a las múltiples tablas que necesitan ser unidas.

- \*\*Uso de Espacio\*\*: El esquema en copo de nieve ahorra espacio, ya que reduce la redundancia de los datos en las dimensiones.

- \*\*Simplicidad\*\*: El esquema en estrella es más simple de entender y utilizar para usuarios finales, mientras que el esquema en copo de nieve es más complejo y adecuado para situaciones donde se prioriza la eficiencia en almacenamiento.

#### 4. \*\*Implementación de los Esquemas\*\*

En la práctica, la elección entre un esquema en estrella y un esquema en copo de nieve depende del caso de uso:

- \*\*Esquema en Estrella\*\*: Se utiliza cuando se necesitan consultas rápidas y simples, donde la redundancia de datos no es un problema crítico.

- \*\*Esquema en Copo de Nieve\*\*: Se prefiere en situaciones donde el almacenamiento es un factor importante y la eficiencia en el uso de espacio tiene prioridad sobre la velocidad de las consultas.

### 5. \*\*Optimización de Consultas en Esquemas\*\*

La optimización de consultas en almacenes de datos se puede mejorar utilizando índices, vistas materializadas, y técnicas de particionamiento. Los índices pueden hacer que las consultas sean más rápidas al facilitar el acceso a los datos, mientras que las vistas materializadas pueden precomputar los resultados de consultas complejas, ahorrando tiempo en futuras ejecuciones.

---

### Preguntas de retroalimentación

1. \*\*¿Qué es una tabla de hechos y qué tipo de datos almacena?\*\*

2. \*\*¿Cuáles son las principales ventajas del esquema en estrella en comparación con el esquema en copo de nieve?\*\*

3. \*\*¿Por qué el esquema en copo de nieve puede ser más difícil de consultar que el esquema en estrella?\*\*

4. \*\*¿Cuándo sería preferible utilizar un esquema en copo de nieve en lugar de uno en estrella?\*\*

5. \*\*¿Qué técnicas pueden aplicarse para optimizar las consultas en un almacén de datos?\*\*

### Clase 36: \*\*Comando SELECT en SQL\*\*

El comando \*\*`SELECT`\*\* es uno de los más importantes en SQL, ya que se utiliza para realizar consultas y recuperar datos de una base de datos. Permite seleccionar columnas específicas de una o varias tablas, filtrarlas y ordenarlas según sea necesario.

#### \*\*Sintaxis básica de SELECT\*\*:

```sql

SELECT columna1, columna2, ...

FROM tabla

[WHERE condición]

[GROUP BY columna]

[HAVING condición]

[ORDER BY columna];

```

#### \*\*Componentes del comando SELECT\*\*:

1. \*\*`SELECT`\*\*: Es la palabra clave que indica que se va a realizar una consulta.

2. \*\*Columnas\*\*: Se especifican las columnas que se quieren recuperar de la tabla.

3. \*\*`FROM`\*\*: Indica de qué tabla o tablas se obtendrán los datos.

4. \*\*`WHERE`\*\* (opcional): Permite filtrar los resultados según una condición específica.

5. \*\*`GROUP BY`\*\* (opcional): Agrupa filas que tienen los mismos valores en columnas específicas.

6. \*\*`HAVING`\*\* (opcional): Filtra los grupos creados por el `GROUP BY`.

7. \*\*`ORDER BY`\*\* (opcional): Ordena los resultados según una columna o varias.

### Ejemplos de uso:

1. \*\*Consulta básica\*\*:

```sql

SELECT nombre, apellido

FROM empleados;

```

Esto selecciona las columnas `nombre` y `apellido` de la tabla `empleados`.

2. \*\*Filtrado con WHERE\*\*:

```sql

SELECT nombre, salario

FROM empleados

WHERE salario > 3000;

```

Esta consulta muestra el nombre y salario de los empleados cuyo salario es mayor a 3000.

3. \*\*Ordenación con ORDER BY\*\*:

```sql

SELECT nombre, salario

FROM empleados

ORDER BY salario DESC;

```

Aquí se ordenan los empleados según su salario, de mayor a menor.

4. \*\*Agrupación con GROUP BY\*\*:

```sql

SELECT departamento, AVG(salario)

FROM empleados

GROUP BY departamento;

```

Agrupa a los empleados por departamento y calcula el salario promedio en cada grupo.

5. \*\*Uso de HAVING\*\*:

```sql

SELECT departamento, COUNT(\*)

FROM empleados

GROUP BY departamento

HAVING COUNT(\*) > 5;

```

Esta consulta muestra los departamentos que tienen más de 5 empleados.

### \*\*Consultas con múltiples tablas\*\*:

El comando `SELECT` también se usa en combinación con otros operadores como `JOIN` para recuperar datos de múltiples tablas:

```sql

SELECT empleados.nombre, departamentos.nombre

FROM empleados

JOIN departamentos ON empleados.departamento\_id = departamentos.id;

```

Esto une las tablas `empleados` y `departamentos` para mostrar el nombre del empleado y el nombre de su departamento.

### Ventajas del comando SELECT:

- \*\*Flexibilidad\*\*: Permite realizar consultas sencillas o complejas con facilidad.

- \*\*Filtrado y agrupación\*\*: Posibilita obtener solo los datos relevantes mediante condiciones y agrupaciones.

- \*\*Ordenación\*\*: Facilita ordenar los resultados según criterios específicos.

### Buenas prácticas:

- Usar solo las columnas necesarias en el `SELECT` para optimizar el rendimiento.

- Filtrar los resultados con `WHERE` para reducir la cantidad de datos procesados.

- Usar índices en las columnas que se filtran frecuentemente para mejorar la velocidad de las consultas.

---

### 5 Preguntas de retroalimentación:

1. \*\*¿Cuál es la función del comando `SELECT` en SQL?\*\*

2. \*\*¿Qué rol juega la cláusula `WHERE` en una consulta SQL?\*\*

3. \*\*¿Cómo se utiliza la cláusula `ORDER BY` en una consulta, y qué opciones de ordenación ofrece?\*\*

4. \*\*¿Qué diferencia hay entre `GROUP BY` y `HAVING`, y en qué casos se usa cada una?\*\*

5. \*\*¿Cómo se realiza una consulta que une dos tablas y qué es necesario para hacer esta operación?\*\*

Con esta clase, tienes una visión profunda de cómo funciona el comando \*\*`SELECT`\*\* en SQL y cómo se puede aprovechar para realizar consultas eficientes. ¡No dudes en practicar estas consultas!

Respuestas 1. se usa para seleccionar los campos que queremos visualizar en la consulta 2. juega el rol de filtrar por la condición que usemos sobre el campo consultado 3. se utiliza después del from y trata de ordenar el campo en la manera que le indiquemos, ya sea desendente o asendente 4. la diferencia es: group by agrupa los registros por alguna cohicidencia y having los filtra, having solo se usa con la consulta group by 5. una consulta para unir dos tablas se realiza con la clausula join, y es estrictamente necesario tener campos en común entre las tablas que queremos consultar juntas

**Clase 37: \*\*`INSERT INTO` en SQL\*\***

En esta clase, aprenderemos a fondo sobre el comando `INSERT INTO` en SQL, que se utiliza para añadir nuevos registros (filas) a una tabla. Es uno de los comandos más básicos y comunes en SQL y forma parte de los comandos \*\*DML (Data Manipulation Language)\*\*.

#### 1. \*\*Sintaxis básica de `INSERT INTO`\*\*:

La sintaxis básica para insertar datos en una tabla es:

```sql

INSERT INTO nombre\_de\_la\_tabla (columna1, columna2, columna3, ...)

VALUES (valor1, valor2, valor3, ...);

```

\*\*Ejemplo\*\*:

Supongamos que tienes una tabla llamada `clientes` con las columnas `nombre`, `edad` y `email`. Para insertar un nuevo cliente, harías lo siguiente:

```sql

INSERT INTO clientes (nombre, edad, email)

VALUES ('Carlos', 30, 'carlos@email.com');

```

Esto añadirá una nueva fila con los valores proporcionados en cada columna.

#### 2. \*\*Insertar múltiples registros\*\*:

SQL también permite insertar varias filas en una sola consulta `INSERT INTO`. Esto es eficiente si necesitas agregar múltiples datos de una vez.

```sql

INSERT INTO clientes (nombre, edad, email)

VALUES

('Ana', 25, 'ana@email.com'),

('Luis', 32, 'luis@email.com'),

('Marta', 28, 'marta@email.com');

```

#### 3. \*\*Insertar datos en todas las columnas\*\*:

Si deseas insertar datos en todas las columnas de una tabla, puedes omitir la lista de columnas en la consulta. Sin embargo, los valores proporcionados deben coincidir con el orden de las columnas en la tabla.

```sql

INSERT INTO clientes

VALUES ('Pedro', 35, 'pedro@email.com');

```

Este formato funciona solo si estás proporcionando un valor para cada columna, en el mismo orden en que las columnas fueron creadas.

#### 4. \*\*Insertar datos seleccionados de otra tabla\*\*:

También es posible insertar registros desde una tabla a otra, utilizando una consulta `SELECT`. Esta técnica es útil cuando quieres mover o duplicar datos entre tablas.

```sql

INSERT INTO clientes\_backup (nombre, edad, email)

SELECT nombre, edad, email

FROM clientes

WHERE edad > 30;

```

Esto insertará en la tabla `clientes\_backup` los registros de la tabla `clientes` donde la edad sea mayor a 30.

#### 5. \*\*Consejos y mejores prácticas\*\*:

- \*\*Especificar columnas\*\*: Siempre es recomendable especificar las columnas en las que vas a insertar datos para evitar problemas si la estructura de la tabla cambia en el futuro.

- \*\*Validación de datos\*\*: Antes de hacer una inserción, asegúrate de que los tipos de datos de los valores coincidan con los tipos de datos de las columnas.

- \*\*Respaldo de datos\*\*: Si realizas una inserción masiva, es importante hacer un respaldo de los datos existentes.

### Preguntas de retroalimentación

1. ¿Cuál es la diferencia entre insertar un solo registro y múltiples registros con el comando `INSERT INTO`?

2. ¿Por qué es recomendable especificar las columnas en las que se van a insertar los datos al usar `INSERT INTO`?

3. ¿Cómo se podría utilizar una consulta `SELECT` dentro de un `INSERT INTO` para copiar registros de una tabla a otra?

4. ¿Qué sucede si los tipos de datos de los valores a insertar no coinciden con los tipos de las columnas de la tabla?

5. ¿Es posible omitir la lista de columnas al usar `INSERT INTO`? ¿En qué casos se recomienda hacerlo?

Respuestas

1. la diferencia es la diligencia al ingresar la cantidades de datos

2. para tener una mejor claridad en los datos que vamos a ingresar en cada sentencia

3. se usa especificanco los campos que queremos pasar de la tabla a otra

4. genera un error al ejecutar la consulta

5. si es posible y se recomienda en el caso de tener total claridad de los campos a ingresar

**Clase 38: \*\*UPDATE en SQL\*\***

El comando `UPDATE` es una de las operaciones más importantes en SQL, ya que permite modificar datos en las filas de una tabla. A diferencia de los comandos `INSERT` (que añade datos) y `DELETE` (que elimina datos), `UPDATE` modifica los datos existentes. A continuación, se explicará cómo funciona este comando, sus componentes principales y las buenas prácticas para evitar errores o modificaciones no deseadas.

#### 1. \*\*Sintaxis del comando `UPDATE`\*\*

La sintaxis básica para el comando `UPDATE` es la siguiente:

```sql

UPDATE nombre\_tabla

SET columna1 = valor1, columna2 = valor2, ...

WHERE condición;

```

- `nombre\_tabla`: Es la tabla en la que se quiere realizar la actualización.

- `SET`: Define las columnas que deseas modificar junto con los nuevos valores.

- `WHERE`: Especifica las filas que serán afectadas por el comando. Si no incluyes una cláusula `WHERE`, todas las filas de la tabla serán actualizadas (¡ten cuidado con esto!).

#### 2. \*\*Ejemplo simple de `UPDATE`\*\*

Imagina que tienes una tabla llamada `empleados` con las columnas `id`, `nombre`, y `salario`. Si deseas aumentar el salario de un empleado con `id = 101`, el comando sería:

```sql

UPDATE empleados

SET salario = salario + 50000

WHERE id = 101;

```

Este comando selecciona la fila en la que `id = 101` y aumenta el salario en 50,000 unidades.

#### 3. \*\*Actualizaciones masivas\*\*

El comando `UPDATE` también se puede usar para realizar cambios en múltiples filas simultáneamente. Si se quiere actualizar el salario de todos los empleados de un departamento específico, la consulta sería:

```sql

UPDATE empleados

SET salario = salario + 10000

WHERE departamento = 'Ventas';

```

En este ejemplo, todos los empleados del departamento "Ventas" recibirán un aumento de 10,000 en su salario.

#### 4. \*\*Uso de subconsultas en `UPDATE`\*\*

Es posible usar subconsultas dentro del comando `UPDATE` para modificar datos basados en otra tabla. Por ejemplo, si tienes una tabla llamada `departamentos` y deseas actualizar los salarios de los empleados dependiendo del presupuesto de su departamento, puedes hacerlo así:

```sql

UPDATE empleados

SET salario = salario \* 1.1

WHERE departamento\_id IN (

SELECT id

FROM departamentos

WHERE presupuesto > 500000

);

```

Este comando aumentará el salario de los empleados cuyos departamentos tengan un presupuesto superior a 500,000.

#### 5. \*\*Buenas prácticas con `UPDATE`\*\*

- \*\*Siempre usa `WHERE`\*\*: Si omites la cláusula `WHERE`, todas las filas de la tabla se actualizarán, lo que puede llevar a errores serios.

- \*\*Pruebas en entornos de desarrollo\*\*: Antes de ejecutar un `UPDATE` en una base de datos de producción, realiza pruebas en un entorno controlado.

- \*\*Realiza respaldos\*\*: Si se trata de actualizaciones masivas o críticas, es importante hacer un respaldo de la tabla o base de datos antes de realizar la operación.

- \*\*Usa la cláusula `RETURNING`\*\*: En algunas bases de datos (como PostgreSQL), puedes usar la cláusula `RETURNING` para devolver las filas modificadas. Esto es útil para verificar qué filas fueron afectadas sin hacer otra consulta:

```sql

UPDATE empleados

SET salario = salario + 1000

WHERE departamento = 'Ventas'

RETURNING \*;

```

#### 6. \*\*Errores comunes en `UPDATE`\*\*

- \*\*Actualización de todas las filas accidentalmente\*\*: No usar la cláusula `WHERE` puede generar que se actualicen todas las filas de la tabla.

- \*\*Inconsistencia en los tipos de datos\*\*: Es fundamental que los tipos de datos entre las columnas y los valores coincidan. Por ejemplo, intentar asignar un valor de texto a una columna numérica generará un error.

- \*\*Condiciones `WHERE` incorrectas\*\*: Si la condición en el `WHERE` no coincide con ninguna fila, no se actualizarán registros, lo que puede ser un problema si esperabas un resultado.

### Preguntas de Retroalimentación

1. ¿Qué diferencia principal existe entre los comandos `INSERT INTO` y `UPDATE` en SQL?

2. ¿Qué sucedería si no incluyes una cláusula `WHERE` en un comando `UPDATE`?

3. ¿Cómo se puede usar una subconsulta en un comando `UPDATE`? Da un ejemplo práctico.

4. ¿Cuáles son las buenas prácticas recomendadas antes de realizar un `UPDATE` masivo en una base de datos?

5. ¿Qué problemas podrías encontrar si los tipos de datos entre las columnas y los valores no coinciden en un `UPDATE`?

Respuestas

1. insert into, ingresa datos y update los actualiza

2. actualiza todas las filas, es importante poner where

3. update productos

Set precio = precio \* 1.9

Where marca\_id(

Select nombre

From marca

Where nombre = sony);

4. siempre usar la clausula where, realizar pruebas y realizar respaldos

5. errores en la ejecución de la consulta

**Clase 39: \*\*DELETE en SQL\*\***

\*\*Introducción:\*\*

El comando `DELETE` en SQL es fundamental para eliminar filas o registros en una tabla dentro de una base de datos. A diferencia de `TRUNCATE`, que elimina todos los registros de una tabla sin condiciones, `DELETE` permite eliminar filas específicas utilizando una cláusula `WHERE`. Es importante usar este comando con cuidado para evitar eliminar accidentalmente todos los registros de una tabla.

\*\*Uso del comando DELETE:\*\*

La sintaxis básica de `DELETE` es:

```sql

DELETE FROM nombre\_tabla

WHERE condición;

```

- \*\*`nombre\_tabla`\*\*: Es el nombre de la tabla de la que deseas eliminar uno o más registros.

- \*\*`WHERE`\*\*: Es opcional, pero generalmente se utiliza para especificar la condición que las filas deben cumplir para ser eliminadas. Si no se usa `WHERE`, todas las filas de la tabla serán eliminadas.

\*\*Ejemplo de DELETE:\*\*

```sql

DELETE FROM empleados

WHERE departamento = 'Ventas';

```

En este ejemplo, se eliminan todos los empleados que pertenecen al departamento de ventas.

\*\*Consideraciones Importantes:\*\*

1. \*\*Cláusula WHERE\*\*: Siempre se recomienda usar `WHERE` para evitar eliminar accidentalmente todos los registros.

2. \*\*Respaldo de Datos\*\*: Antes de ejecutar un `DELETE`, especialmente en tablas críticas, es buena práctica realizar un respaldo de la información.

3. \*\*Restricciones\*\*: Si existen restricciones de integridad referencial (como claves foráneas), puede que no se permita eliminar una fila si otras tablas dependen de esa fila. En tales casos, puede ser necesario utilizar `CASCADE` para eliminar todas las filas relacionadas.

4. \*\*Transacciones\*\*: Es posible ejecutar un `DELETE` dentro de una transacción. Esto permite revertir el borrado si ocurre algún error, utilizando `ROLLBACK`.

\*\*Diferencia entre DELETE y TRUNCATE:\*\*

- `DELETE`: Elimina filas individuales basadas en una condición. Es más lento porque las operaciones se registran fila por fila y pueden disparar triggers.

- `TRUNCATE`: Elimina todas las filas de una tabla de forma rápida, sin registrar la eliminación fila por fila. No se puede utilizar `WHERE` con `TRUNCATE`.

\*\*Consecuencias de un DELETE sin WHERE:\*\*

Si se omite la cláusula `WHERE`, el comando eliminará todas las filas de la tabla especificada:

```sql

DELETE FROM empleados;

```

Este comando eliminaría todos los empleados de la tabla sin excepción. Asegúrate de usar `WHERE` adecuadamente para evitar pérdidas no deseadas de datos.

---

### Preguntas de Retroalimentación:

1. \*\*¿Cuál es la diferencia clave entre los comandos `DELETE` y `TRUNCATE` en SQL?\*\*

2. \*\*¿Por qué es importante utilizar una cláusula `WHERE` en un comando `DELETE`?\*\*

3. \*\*Si quiero eliminar todos los registros de una tabla sin disparar triggers, ¿qué comando es más adecuado?\*\*

4. \*\*¿Qué sucede si se intenta eliminar una fila con `DELETE` que tiene dependencias en otras tablas con restricciones de claves foráneas?\*\*

5. \*\*¿Cómo podría revertir un `DELETE` si se ha cometido un error y se está utilizando una transacción en SQL?\*\*

Respuestas

1. delete, elimina las filas un poco mas lento al tener una condición, va fila por fila buscando dichos registros, truncate, elimina todos los registros, no puedes usar where con truncate

2. para envitar borrar todos los registros

3. truncate

4. te lanza un error, necesitaríamos usar cascade, para evitar dicho error

5. usando el comando rollback

**Clase 40: \*\*Tablas, Filas y Columnas\*\***

En esta clase, exploraremos los conceptos fundamentales de las tablas, filas y columnas en bases de datos relacionales, ya que son la estructura básica para almacenar y organizar datos.

#### \*\*Tablas\*\*

Una tabla es una colección de datos organizados en filas y columnas. Cada tabla está destinada a representar una entidad específica, por ejemplo, empleados, productos, o ventas. Las tablas en una base de datos son similares a una hoja de cálculo, pero con un propósito estructurado, organizado en una base de datos.

- \*\*Nombre de la tabla\*\*: Cada tabla tiene un nombre único dentro de la base de datos para poder referenciarla.

- \*\*Esquema\*\*: Un esquema define el nombre, los tipos de datos, las restricciones, y otras propiedades de las columnas que componen la tabla.

#### \*\*Filas\*\*

Una fila (o registro) representa una instancia única de la entidad que la tabla está modelando. Cada fila contiene valores que corresponden a las columnas de la tabla.

- \*\*Filas únicas\*\*: Las tablas pueden tener claves primarias que garantizan que cada fila sea única.

- \*\*Contenido de las filas\*\*: Cada fila puede contener datos de diferentes tipos, según la estructura de las columnas de la tabla.

#### \*\*Columnas\*\*

Las columnas son campos que definen los atributos de la entidad que representa la tabla. Cada columna tiene un nombre y un tipo de dato específico, como números, cadenas de texto o fechas.

- \*\*Nombre de la columna\*\*: Define qué tipo de información contiene cada columna.

- \*\*Tipo de dato\*\*: Cada columna tiene un tipo de dato asociado (por ejemplo, `INTEGER`, `VARCHAR`, `DATE`), lo que permite definir qué tipo de valores pueden almacenarse en esa columna.

- \*\*Restricciones\*\*: Las columnas también pueden tener restricciones como `NOT NULL`, que garantiza que una columna no puede contener valores vacíos.

#### \*\*Interacción entre Tablas, Filas y Columnas\*\*

- Las \*\*filas\*\* y \*\*columnas\*\* forman una cuadrícula en la que cada fila contiene información sobre un único objeto o entidad, y cada columna proporciona detalles específicos sobre ese objeto.

- En SQL, los comandos como `INSERT INTO`, `SELECT`, `UPDATE`, y `DELETE` permiten manipular los datos almacenados en las tablas.

Las tablas son esenciales para la organización de datos en bases de datos relacionales. Permiten estructurar y gestionar grandes volúmenes de datos de manera eficiente y segura.

---

### \*\*Preguntas para Retroalimentación:\*\*

1. \*\*¿Qué es una tabla en una base de datos relacional y cuál es su función principal?\*\*

2. \*\*¿Cuál es la diferencia entre una fila y una columna en una tabla?\*\*

3. \*\*¿Qué papel juegan los tipos de datos en las columnas de una tabla?\*\*

4. \*\*¿Por qué es importante definir una clave primaria en una tabla?\*\*

5. \*\*¿Cómo se relacionan las tablas entre sí en una base de datos relacional?\*\*

Respuestas

1. la tabla es la parte fundamental del modelo relacional, su función es albergar los datos de manera organizada y representar alguna entidad dentro del modelo relacional

2. la principal consiste en que las filas albergan los datos y las columnas dan el nombre al campo y el tipo de dato que se almacena

3. definen el tipo de dato que almacenara esa columna, ya sea texto numérico, fecha o boleano

4. para tener una identificación en cada registro

5. con las llaves primarias y las llaves foráneas, estos campos sirven como conectores para establecer una relación entre tablas

**Clase 41: Tipos de datos de las columnas**

\*\*Descripción:\*\*

En SQL, los tipos de datos definen el tipo de información que se puede almacenar en cada columna de una tabla. Es esencial elegir el tipo de dato adecuado para cada columna, ya que esto influye en la eficiencia del almacenamiento, la velocidad de las consultas y la integridad de los datos. A continuación, se explican los tipos de datos más comunes y su uso:

---

\*\*1. Tipos de Datos Numéricos:\*\*

- \*\*INTEGER/INT:\*\* Almacena números enteros sin decimales. Es útil para almacenar identificadores únicos, cantidades, y otras cifras enteras. Ejemplo: `1, 2, 150`.

- \*\*FLOAT/DOUBLE:\*\* Utilizado para almacenar números con decimales, proporcionando más precisión que `INTEGER`. Se usa comúnmente para cálculos científicos o cuando se necesita precisión con decimales. Ejemplo: `1.23, 3.14159`.

- \*\*DECIMAL/NUMERIC:\*\* Almacenan números exactos con un número fijo de decimales. Son útiles para cálculos financieros donde la precisión de los decimales es importante. Ejemplo: `100.50, 2500.75`.

---

\*\*2. Tipos de Datos de Cadena de Caracteres:\*\*

- \*\*CHAR(n):\*\* Almacena cadenas de caracteres con una longitud fija de `n`. Si la cadena tiene menos caracteres que `n`, se rellena con espacios. Es adecuado para códigos o identificadores con una longitud fija. Ejemplo: `CHAR(5)` almacenaría la cadena `abc` como `abc--`.

- \*\*VARCHAR(n):\*\* Almacena cadenas de caracteres con una longitud variable de hasta `n` caracteres. Es más flexible que `CHAR` y es útil para almacenar nombres, descripciones y otros textos de longitud variable. Ejemplo: `VARCHAR(50)`.

- \*\*TEXT:\*\* Utilizado para almacenar grandes cantidades de texto. No suele tener un límite definido como `VARCHAR`, por lo que se usa para almacenar descripciones, comentarios, entre otros. Ejemplo: `TEXT` puede almacenar un párrafo o incluso varios.

---

\*\*3. Tipos de Datos de Fecha y Hora:\*\*

- \*\*DATE:\*\* Almacena solo la fecha (año, mes, día). Es útil para registrar fechas de nacimiento, fechas de contratación, entre otros. Ejemplo: `2024-10-03`.

- \*\*TIME:\*\* Almacena solo la hora (hora, minuto, segundo). Se utiliza cuando solo se necesita la hora sin la fecha. Ejemplo: `13:45:30`.

- \*\*DATETIME:\*\* Almacena tanto la fecha como la hora. Es ideal para registrar eventos que requieren precisión en la fecha y hora, como transacciones. Ejemplo: `2024-10-03 13:45:30`.

- \*\*TIMESTAMP:\*\* Similar a `DATETIME`, pero es más adecuado cuando se necesita registrar el momento exacto de un evento y realizar operaciones de comparación. Ejemplo: `2024-10-03 13:45:30`.

---

\*\*4. Tipos de Datos Booleanos:\*\*

- \*\*BOOLEAN:\*\* Almacena valores de `TRUE` o `FALSE`. Se utiliza para columnas que solo necesitan representar dos estados, como `activo`/`inactivo`, `sí`/`no`, o `verdadero`/`falso`.

---

\*\*5. Otros Tipos de Datos Especiales:\*\*

- \*\*BLOB (Binary Large Object):\*\* Almacena grandes cantidades de datos binarios, como imágenes, archivos, y otros tipos de datos binarios. Es útil cuando se desea almacenar archivos directamente en la base de datos.

- \*\*ENUM:\*\* Permite definir un conjunto de valores posibles para una columna. Es útil cuando una columna debe contener un valor predefinido de una lista, como `masculino`/`femenino`, `básico`/`avanzado`.

\*\*Consideraciones para la elección de tipos de datos:\*\*

- \*\*Espacio de Almacenamiento:\*\* Algunos tipos de datos, como `TEXT` o `BLOB`, ocupan más espacio de almacenamiento que otros. Se debe elegir un tipo de dato que optimice el uso del espacio.

- \*\*Integridad de Datos:\*\* Utilizar tipos de datos correctos garantiza que solo se almacenen valores válidos en las columnas (por ejemplo, una columna `DATE` solo almacenará fechas válidas).

- \*\*Rendimiento de las Consultas:\*\* El tipo de dato puede afectar la rapidez con la que se realizan las consultas. Tipos de datos numéricos suelen ser más rápidos de consultar que los de texto.

---

### Preguntas de Retroalimentación:

1. \*\*¿Qué tipo de dato utilizarías para almacenar el número de identificación de un usuario y por qué?\*\*

2. \*\*¿Cuál es la diferencia entre los tipos de datos `CHAR` y `VARCHAR`? ¿Cuándo preferirías usar uno sobre el otro?\*\*

3. \*\*¿Qué tipo de dato elegirías para almacenar un campo de descripción extensa de un producto? Justifica tu respuesta.\*\*

4. \*\*¿Cómo afectan los tipos de datos a la integridad de una base de datos? Proporciona un ejemplo.\*\*

5. \*\*¿Cuál es la diferencia entre `DATETIME` y `TIMESTAMP` en SQL? Proporciona un caso de uso para cada uno.\*\*

Respuestas

1. usaría el tipo de dato int, al ser un numero no tan extenso y de uso único

2. el tipo char es de cantidas estatica y varchar variable, usaría char cuando me soliciten que la cantidad de caracteres debe ser la misma en cada string y varchar para ahorrar espacio en almacenamiento de strings

3. text, es ideal para almacenar grandes cantidades de caracteres

4. afectan en diferente forma, tanto su peso en disco como su mal almacenamiento, ejemplo al colocar un dato numerico que necesite realizar cálculos matemáticos como tipo string

5. datatime proporciona fecha y hora ideal para almacenar transacciones, timestamp ideal para comparaciones y registro mas exacto en cuanto al tiempo de ejecución de la transacción

**Clase 42: Consultas SQL Básicas**

Las consultas SQL básicas son fundamentales para interactuar con una base de datos. Estas permiten realizar la recuperación de datos, aplicar filtros y organizar los resultados según los requerimientos. A continuación, se describen algunos de los conceptos más importantes en el ámbito de las consultas básicas:

#### 1. \*\*SELECT: La Base de las Consultas\*\*

- El comando `SELECT` es la instrucción central para recuperar datos de una base de datos.

- Se utiliza para especificar las columnas que deseamos ver en los resultados de una consulta.

- Su estructura básica es:

```sql

SELECT column1, column2, ...

FROM table\_name;

```

- Si se desean obtener todos los datos de una tabla, se puede usar `SELECT \*`, que devuelve todas las columnas.

#### 2. \*\*FROM: Especificar la Fuente de los Datos\*\*

- El comando `FROM` se utiliza para indicar la tabla desde la cual se van a extraer los datos.

- En una consulta simple, se puede extraer la información de una única tabla, pero también es posible realizar uniones (`JOIN`) para combinar datos de múltiples tablas.

#### 3. \*\*LIMIT: Restringir la Cantidad de Resultados\*\*

- `LIMIT` es útil cuando se quieren ver solo una cantidad específica de registros.

- Esto puede ser muy útil al trabajar con grandes volúmenes de datos, ya que permite revisar un subconjunto de la información sin sobrecargar la consulta.

- Ejemplo:

```sql

SELECT \*

FROM employees

LIMIT 10;

```

- Este ejemplo retorna los primeros 10 registros de la tabla `employees`.

#### 4. \*\*DISTINCT: Eliminar Duplicados en los Resultados\*\*

- `DISTINCT` se utiliza para seleccionar valores únicos, eliminando registros duplicados en los resultados.

- Esto es útil cuando se necesita obtener una lista de valores únicos de una columna específica.

- Ejemplo:

```sql

SELECT DISTINCT department

FROM employees;

```

- En este caso, se obtendría una lista de todos los departamentos únicos de la tabla `employees`.

#### 5. \*\*ORDER BY: Ordenar los Resultados\*\*

- `ORDER BY` permite organizar los resultados de una consulta en un orden específico, ya sea ascendente (`ASC`) o descendente (`DESC`).

- Es útil para ordenar datos por una o varias columnas, como nombres, fechas o cantidades.

- Ejemplo:

```sql

SELECT name, salary

FROM employees

ORDER BY salary DESC;

```

- Aquí, se obtendría una lista de empleados ordenada por el salario de mayor a menor.

#### 6. \*\*WHERE: Filtrar los Resultados\*\*

- El comando `WHERE` se usa para filtrar los registros que cumplen con una condición específica.

- Esto es fundamental para obtener solo los datos relevantes según un criterio, como un rango de fechas, un valor específico, entre otros.

- Ejemplo:

```sql

SELECT name, department

FROM employees

WHERE department = 'Sales';

```

- En este caso, se seleccionan solo aquellos empleados que pertenecen al departamento de ventas.

#### 7. \*\*Funciones de Agregación (SUM, COUNT, AVG, MIN, MAX)\*\*

- Las funciones de agregación permiten realizar cálculos en conjunto de datos.

- Ejemplos de estas funciones incluyen:

- `SUM()`: Suma de valores de una columna.

- `COUNT()`: Número de registros.

- `AVG()`: Promedio de valores de una columna.

- `MIN()` y `MAX()`: Valor mínimo y máximo.

- Ejemplo:

```sql

SELECT COUNT(\*) AS total\_employees

FROM employees;

```

- Aquí, se cuenta el total de empleados en la tabla `employees`.

Estas consultas básicas forman la base para manipular y analizar datos en SQL. A medida que se domina su uso, es posible combinar estos conceptos para crear consultas más complejas y poderosas.

---

### 5 Preguntas para Retroalimentación:

1. ¿Cuál es la función del comando `SELECT` en una consulta SQL?

2. ¿En qué casos utilizarías el modificador `DISTINCT` en una consulta?

3. ¿Cuál es la diferencia entre `ORDER BY ASC` y `ORDER BY DESC`?

4. ¿Para qué sirve el comando `LIMIT` y por qué podría ser útil al trabajar con grandes volúmenes de datos?

5. ¿Cómo podrías contar cuántos empleados hay en el departamento de marketing usando `WHERE` y `COUNT`?

Respuestas

1. es donde asignamos que campos queremos visualizar

2. en caso de que quiera ver solo una lista de campos únicos

3. el primero ordena de forma asendente de mayor a menos y el segundo desendente de mayor a menos

4. limita la cantidad de datos que veremos, lo cual evita sobre cargas en la consulta

5. realizaría la siguiente consulta:

Select count(\*) total\_empleados

From empleados

Where departamento = “marketing”;

**Clase 43: \*\*WHERE\*\***

La cláusula `WHERE` es fundamental en SQL, ya que permite filtrar los registros de una tabla para obtener únicamente aquellos que cumplen con ciertas condiciones. Se utiliza junto con los comandos `SELECT`, `UPDATE`, `DELETE`, entre otros, para especificar qué filas deben ser consideradas al realizar la operación.

#### Estructura básica de la cláusula `WHERE`

La sintaxis general de la cláusula `WHERE` es la siguiente:

```sql

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

```

En este caso:

- `SELECT`: Especifica las columnas que queremos visualizar.

- `FROM`: Indica la tabla de donde se tomarán los datos.

- `WHERE`: Define una condición que los registros deben cumplir para ser incluidos en el resultado.

#### Uso de operadores en `WHERE`

La cláusula `WHERE` permite el uso de varios operadores para definir condiciones más complejas. Entre los más comunes se encuentran:

1. \*\*Operadores de comparación\*\*:

- `=` (igual a)

- `!=` o `<>` (diferente de)

- `>` (mayor que)

- `<` (menor que)

- `>=` (mayor o igual que)

- `<=` (menor o igual que)

Ejemplo:

```sql

SELECT \* FROM empleados WHERE salario > 50000;

```

Esta consulta devolverá todos los empleados cuyo salario es mayor a 50,000.

2. \*\*Operadores lógicos\*\*:

- `AND`: Permite combinar múltiples condiciones, y ambas deben ser verdaderas.

- `OR`: Permite combinar condiciones, y al menos una de ellas debe ser verdadera.

- `NOT`: Invierte el resultado de una condición.

Ejemplo:

```sql

SELECT \* FROM empleados WHERE departamento = 'ventas' AND salario > 40000;

```

Esta consulta devolverá todos los empleados que pertenezcan al departamento de ventas y cuyo salario sea superior a 40,000.

3. \*\*Operadores de rango y conjuntos\*\*:

- `BETWEEN`: Se utiliza para especificar un rango de valores.

- `IN`: Se utiliza para verificar si un valor está dentro de una lista de valores.

- `LIKE`: Se utiliza para buscar un patrón dentro de una columna de texto.

Ejemplos:

- `BETWEEN`:

```sql

SELECT \* FROM empleados WHERE salario BETWEEN 30000 AND 50000;

```

Esto devolverá todos los empleados con un salario entre 30,000 y 50,000 (inclusive).

- `IN`:

```sql

SELECT \* FROM empleados WHERE departamento IN ('ventas', 'marketing', 'soporte');

```

Esto devolverá todos los empleados que pertenezcan a cualquiera de los departamentos especificados.

- `LIKE`:

```sql

SELECT \* FROM clientes WHERE nombre LIKE 'A%';

```

Esto devolverá todos los clientes cuyo nombre comience con la letra "A". `%` es un comodín que representa cualquier cadena de caracteres.

4. \*\*Operador `IS NULL`\*\*:

Sirve para verificar si una columna tiene un valor nulo, es decir, si no tiene ningún valor almacenado.

Ejemplo:

```sql

SELECT \* FROM empleados WHERE fecha\_nacimiento IS NULL;

```

Esta consulta devolverá todos los empleados que no tienen una fecha de nacimiento registrada.

#### Importancia de la cláusula `WHERE`

- \*\*Filtrar datos\*\*: Permite enfocarse solo en los registros que son relevantes para el análisis o la operación que se está realizando.

- \*\*Mejorar el rendimiento\*\*: Al reducir la cantidad de datos que se procesan, las consultas pueden ser más rápidas y eficientes.

- \*\*Realizar análisis específicos\*\*: La cláusula `WHERE` es fundamental para realizar cálculos y análisis que dependan de criterios específicos, como sumar ventas de un periodo específico o contar registros que cumplen con una condición.

#### Ejemplo completo:

```sql

SELECT nombre, salario

FROM empleados

WHERE salario > 40000 AND departamento = 'Recursos Humanos';

```

Esta consulta selecciona los nombres y salarios de los empleados que trabajan en el departamento de Recursos Humanos y que ganan más de 40,000.

---

### Preguntas de retroalimentación:

1. \*\*¿Para qué se utiliza la cláusula `WHERE` en SQL y con qué comandos se puede combinar?\*\*

2. \*\*¿Cuál es la diferencia entre `AND` y `OR` al utilizarse en la cláusula `WHERE`?\*\*

3. \*\*¿Qué resultado esperas al utilizar la condición `WHERE salario BETWEEN 30000 AND 50000`?\*\*

4. \*\*¿Cómo podrías buscar registros que no tienen un valor en una columna específica usando `WHERE`?\*\*

5. \*\*Escribe una consulta SQL que seleccione todos los clientes cuyo nombre comience con la letra "M" y que pertenezcan a la ciudad "Madrid".\*\*

Respuestas

1. la clausula where se usa para el filtrado de una columna, se pueden mezclar con update, delect y select, también se puede mezclar con operadores lógicos, de rango y de comparación

2. and se vuelve true si ambas condiciones son verdaderas, or se vuelve true si alguna de las dos es verdadera

3. los registros con un saldo entre 30000 y 50000

4. con el comando is null

5. select nombre\_cliente

From clientes

Where ciudad = ‘madrid’ and nombre like ‘M%’;

**Clase 44: `GROUP BY` en SQL**

#### ¿Qué es `GROUP BY`?

`GROUP BY` es una cláusula en SQL que se utiliza para agrupar filas que tienen valores idénticos en columnas específicas. Su principal propósito es aplicar funciones de agregación (como `SUM`, `COUNT`, `AVG`, `MIN`, `MAX`, etc.) sobre grupos de datos. En lugar de calcular los valores agregados para toda la tabla, `GROUP BY` los calcula para cada grupo de datos definido.

#### ¿Cómo funciona `GROUP BY`?

La cláusula `GROUP BY` se coloca después de la cláusula `WHERE` y antes de `ORDER BY`. Toma los resultados de una consulta y los agrupa en subconjuntos con base en uno o más campos de la tabla.

Por ejemplo, si se tiene una tabla de `ventas` y se desea obtener el total de ventas por cada `categoria`, se utilizaría `GROUP BY` de la siguiente manera:

```sql

SELECT categoria, SUM(monto\_venta) AS total\_ventas

FROM ventas

GROUP BY categoria;

```

En este ejemplo, se agruparán todas las filas de la tabla `ventas` que pertenezcan a la misma `categoria` y se sumarán los valores de `monto\_venta` para cada grupo.

#### ¿Cuándo usar `GROUP BY`?

`GROUP BY` es ideal cuando necesitas realizar análisis o reportes que impliquen agrupar datos para obtener métricas resumidas. Algunos casos comunes de uso son:

- Calcular el total de ventas por cada cliente.

- Contar la cantidad de productos vendidos por cada categoría.

- Obtener el promedio de calificaciones por curso o clase.

- Determinar el salario promedio por departamento en una empresa.

#### Funciones de agregación compatibles con `GROUP BY`

Las funciones de agregación más comunes que se utilizan con `GROUP BY` son:

- \*\*`SUM()`\*\*: Suma los valores de una columna numérica.

- \*\*`COUNT()`\*\*: Cuenta el número de filas en cada grupo.

- \*\*`AVG()`\*\*: Calcula el promedio de los valores de una columna numérica.

- \*\*`MIN()`\*\*: Encuentra el valor mínimo en cada grupo.

- \*\*`MAX()`\*\*: Encuentra el valor máximo en cada grupo.

Cada una de estas funciones opera sobre los grupos creados por `GROUP BY`.

#### Ejemplo práctico

Imagina que tienes una tabla llamada `empleados` con las columnas `departamento` y `salario`. Si deseas calcular el salario promedio de los empleados por cada departamento, podrías escribir:

```sql

SELECT departamento, AVG(salario) AS salario\_promedio

FROM empleados

GROUP BY departamento;

```

En este caso, los datos se agrupan por `departamento` y la función `AVG` calcula el promedio de `salario` dentro de cada grupo.

### Consideraciones importantes al usar `GROUP BY`

1. \*\*Compatibilidad con `SELECT`\*\*: Toda columna que esté en el `SELECT` y no sea utilizada en una función de agregación, debe estar en la cláusula `GROUP BY`. De lo contrario, se generará un error.

- Ejemplo incorrecto:

```sql

SELECT nombre\_empleado, AVG(salario)

FROM empleados

GROUP BY departamento;

```

Esto dará un error porque `nombre\_empleado` no está en `GROUP BY` ni en una función de agregación.

- Ejemplo correcto:

```sql

SELECT departamento, AVG(salario)

FROM empleados

GROUP BY departamento;

```

2. \*\*Orden de ejecución\*\*: La cláusula `GROUP BY` se ejecuta después de que los datos son filtrados por `WHERE` y antes de ser ordenados por `ORDER BY`.

3. \*\*Niveles de agrupamiento\*\*: Se pueden agrupar los datos en varios niveles si se especifican múltiples columnas en `GROUP BY`. Esto permitirá agrupar por varias características, como en este ejemplo:

```sql

SELECT departamento, ciudad, COUNT(\*) AS total\_empleados

FROM empleados

GROUP BY departamento, ciudad;

```

Esto agrupa los datos primero por `departamento` y luego por `ciudad` dentro de cada `departamento`.

### Preguntas para retroalimentar

1. ¿Qué función cumple la cláusula `GROUP BY` en una consulta SQL?

2. Menciona al menos tres funciones de agregación que se pueden usar junto con `GROUP BY` y su propósito.

3. ¿Por qué es necesario que las columnas en `SELECT` también estén en `GROUP BY` si no se están utilizando funciones de agregación?

4. ¿Cuál es la diferencia entre `GROUP BY` y `ORDER BY` en SQL?

5. Escribe una consulta que cuente la cantidad de productos vendidos por cada categoría en una tabla llamada `productos\_vendidos`.

Respuestas

1. agrupa los registros por un campo especifico

2. sum, count, avg

3. para no generar un error en la consulta

4. group by agrupa, order by ordena es su diferencia principal

5. select categoría, count(cantidad\_vendidod) as total vendidos

From productos\_vendidos

Group by categoría;

**Clase 45: GROUP BY + HAVING**

En esta clase, nos adentraremos en la cláusula `HAVING`, que se utiliza junto con `GROUP BY` para filtrar resultados después de que los datos han sido agrupados. La principal diferencia entre `WHERE` y `HAVING` es que `WHERE` filtra filas antes de la agrupación, mientras que `HAVING` se utiliza para filtrar los resultados de las funciones de agregación después de que los datos han sido agrupados.

#### ¿Qué es la cláusula HAVING?

La cláusula `HAVING` es especialmente útil cuando se quiere realizar un filtrado sobre los resultados de una agrupación que ha sido generada por `GROUP BY`. Por ejemplo, si deseas encontrar categorías de productos que tienen un total de ventas superior a una cantidad específica, usarías `HAVING` después de agrupar los productos por categoría.

\*\*Sintaxis Básica:\*\*

```sql

SELECT columna1, columna2, FUNCION\_AGREGADA(columna3)

FROM tabla

GROUP BY columna1, columna2

HAVING FUNCION\_AGREGADA(columna3) condición valor;

```

- `columna1` y `columna2`: las columnas que serán agrupadas.

- `FUNCION\_AGREGADA`: como `SUM`, `AVG`, `COUNT`, `MIN` o `MAX`.

- `condición`: un operador de comparación como `=`, `>`, `<`, `>=`, `<=`.

- `valor`: el valor contra el cual se está comparando el resultado de la función de agregación.

#### Ejemplos de Uso de HAVING

1. \*\*Filtrar con SUM:\*\*

Supongamos que tenemos una tabla de ventas y queremos encontrar los productos que han vendido más de 100 unidades:

```sql

SELECT producto\_id, SUM(cantidad) AS total\_vendido

FROM ventas

GROUP BY producto\_id

HAVING SUM(cantidad) > 100;

```

- Primero, se agrupan las ventas por `producto\_id`.

- Después, se calcula el total de unidades vendidas para cada producto con `SUM(cantidad)`.

- Finalmente, `HAVING` filtra para mostrar solo aquellos productos que superan las 100 unidades vendidas.

2. \*\*Filtrar con COUNT:\*\*

Si deseamos listar categorías de productos que tienen más de 5 productos distintos:

```sql

SELECT categoria, COUNT(\*) AS total\_productos

FROM productos

GROUP BY categoria

HAVING COUNT(\*) > 5;

```

- Agrupa los productos por `categoria`.

- Luego, cuenta la cantidad de productos en cada categoría.

- `HAVING` limita los resultados para mostrar solo las categorías que tienen más de 5 productos.

3. \*\*Filtrar con AVG:\*\*

Si necesitamos encontrar empleados cuyo salario promedio por departamento supere los 3000:

```sql

SELECT departamento\_id, AVG(salario) AS salario\_promedio

FROM empleados

GROUP BY departamento\_id

HAVING AVG(salario) > 3000;

```

- Agrupa los salarios de los empleados por `departamento\_id`.

- Calcula el promedio con `AVG(salario)`.

- Filtra con `HAVING` para mostrar solo los departamentos con un salario promedio superior a 3000.

#### Consideraciones al Usar HAVING

- \*\*Orden de Ejecución\*\*: `HAVING` se evalúa después de `GROUP BY`, por lo que las funciones de agregación ya han sido calculadas para cuando `HAVING` aplica el filtro.

- \*\*Diferencia con WHERE\*\*: `WHERE` no puede filtrar los resultados de una función de agregación; solo `HAVING` tiene esa capacidad.

- \*\*Uso con múltiples condiciones\*\*: Se pueden usar operadores como `AND` y `OR` dentro de `HAVING` para combinar múltiples condiciones, al igual que con `WHERE`.

---

### Preguntas de Retroalimentación

1. ¿Cuál es la principal diferencia entre `WHERE` y `HAVING` en una consulta SQL?

2. ¿Por qué es importante usar `HAVING` cuando se trabajan con funciones de agregación como `SUM` o `COUNT`?

3. ¿Cómo se filtrarían los productos que tienen un `COUNT` mayor a 10 en un `GROUP BY`?

4. Si una consulta incluye `GROUP BY` y `HAVING`, ¿en qué orden se procesan?

5. Escribe una consulta SQL que muestre las ciudades con un número promedio de habitantes mayor a 500,000 utilizando `GROUP BY` y `HAVING`.

Respuestas

1. primero que having se usa sobre agrupaciones es decir que si no esta el comando group by no se pude usar y segunda que se usa sobre funciones agregadas

2. para dar mayor filtrado o selección a los campos agrupados

3. con having > 10

4. having va después del comando grouo by

5. select ciudad, avg(numero\_habitantes) as promedio\_habitantes

From habitantes

Group by ciudad

Having avg(numero\_habitantes) > 500000;

**Clase 46: `ORDER BY` en SQL**

La cláusula `ORDER BY` en SQL se utiliza para ordenar los resultados de una consulta en función de una o más columnas. Esto es útil cuando se desea presentar los datos en un orden específico, como alfabéticamente, numéricamente, o por fecha. La ordenación se puede realizar en orden ascendente (por defecto) o descendente, lo que ayuda a organizar la salida de datos según las necesidades de análisis o presentación.

#### Sintaxis de `ORDER BY`

```sql

SELECT columna1, columna2, ...

FROM tabla

ORDER BY columna1 [ASC|DESC], columna2 [ASC|DESC], ...;

```

- \*\*`SELECT`\*\*: Se usa para seleccionar las columnas deseadas.

- \*\*`FROM`\*\*: Especifica la tabla desde la cual se extraerán los datos.

- \*\*`ORDER BY`\*\*: Indica que se debe ordenar la salida según las columnas indicadas.

- \*\*`ASC`\*\*: Ordena de forma ascendente (de menor a mayor o de A a Z). Es la opción por defecto, por lo que si no se especifica, el orden será ascendente.

- \*\*`DESC`\*\*: Ordena de forma descendente (de mayor a menor o de Z a A).

#### Ejemplos de `ORDER BY`

1. \*\*Ordenar alfabéticamente por nombre de cliente:\*\*

```sql

SELECT nombre\_cliente, ciudad

FROM clientes

ORDER BY nombre\_cliente;

```

- Este ejemplo ordena la lista de clientes alfabéticamente por el nombre del cliente, de la A a la Z.

2. \*\*Ordenar numéricamente por el precio de un producto de mayor a menor:\*\*

```sql

SELECT nombre\_producto, precio

FROM productos

ORDER BY precio DESC;

```

- Aquí, los productos se ordenan por su precio de forma descendente, de modo que los productos más caros aparecen primero.

3. \*\*Ordenar por múltiples columnas:\*\*

```sql

SELECT nombre, apellido, edad

FROM personas

ORDER BY apellido ASC, edad DESC;

```

- En este caso, primero se ordenan las personas alfabéticamente por apellido (de A a Z) y, en caso de que haya varios con el mismo apellido, se ordenan por edad de forma descendente.

#### Consideraciones al Usar `ORDER BY`

- \*\*Rendimiento\*\*: Ordenar grandes cantidades de datos puede ser costoso en términos de rendimiento, especialmente si no hay índices sobre las columnas por las que se ordena. Por eso, es importante diseñar bien la estructura de índices cuando se espera realizar muchas consultas con `ORDER BY`.

- \*\*Columnas calculadas\*\*: Se puede utilizar `ORDER BY` con columnas calculadas en la consulta, como en el caso de sumar o restar valores.

```sql

SELECT nombre, salario, (salario \* 0.1) AS bono

FROM empleados

ORDER BY bono DESC;

```

- Este ejemplo ordena a los empleados según el valor del bono calculado, de mayor a menor.

### Preguntas de Retroalimentación

1. \*\*¿Cuál es la diferencia entre `ASC` y `DESC` en una consulta `ORDER BY`?\*\*

2. \*\*¿Qué sucede si se omite la palabra clave `ASC` o `DESC` al usar `ORDER BY` en una consulta?\*\*

3. \*\*Si deseas ordenar una lista de productos por su nombre de la A a la Z y por precio de mayor a menor, ¿cómo estructurarías la consulta?\*\*

4. \*\*¿Qué impacto tiene el uso de `ORDER BY` en el rendimiento de una consulta y cómo se puede mitigar este impacto?\*\*

5. \*\*Escribe una consulta que muestre los nombres y salarios de empleados ordenados primero por el nombre de forma ascendente y luego por el salario de forma descendente.\*\*

Respuestas

1. asc organiza de forma menor a mayot, es la forma predeterminada, desc organiza de mayor a menor

2. organiza por la forma asc que es la predeterminada

3. select nombre\_product, precio

From productos

Order by nombre\_product asc, precio desc;

4. puede aumentar el uso de disco de manera significativa y se puede mitigar aplicando el uso de índices

5. select nombre\_emple, salario

From empleados

Order by nombre\_emple asc, salario desc;

**Clase 47: Consultas SQL Avanzadas**

En esta clase, vamos a profundizar en \*\*consultas SQL avanzadas\*\*. Las consultas avanzadas son aquellas que permiten realizar operaciones más complejas sobre los datos, como unir múltiples tablas, realizar subconsultas, aplicar funciones de ventana y operaciones de conjuntos. Dominar estas consultas te permitirá trabajar con bases de datos de manera más eficiente y extraer información más detallada y precisa.

#### 1. \*\*Unión de Tablas (Joins)\*\*

- Las consultas avanzadas suelen involucrar el uso de `JOINs`, que son combinaciones de dos o más tablas a través de un campo en común. Esto permite relacionar la información de diferentes tablas, generando vistas más completas de los datos.

- Existen varios tipos de `JOINs` que ya veremos en detalle, como `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, y `FULL OUTER JOIN`.

- Ejemplo de `INNER JOIN`:

```sql

SELECT empleados.nombre, departamentos.nombre

FROM empleados

INNER JOIN departamentos ON empleados.departamento\_id = departamentos.id;

```

En este caso, estamos combinando la tabla `empleados` y `departamentos` para obtener el nombre del empleado junto con el nombre del departamento al que pertenece.

#### 2. \*\*Subconsultas\*\*

- Una subconsulta es una consulta dentro de otra consulta. Se utiliza para devolver datos que serán usados por la consulta principal, proporcionando una forma de realizar búsquedas anidadas.

- Las subconsultas se pueden usar en cláusulas `SELECT`, `WHERE` o `FROM`.

- Ejemplo de subconsulta:

```sql

SELECT nombre, salario

FROM empleados

WHERE salario > (SELECT AVG(salario) FROM empleados);

```

Esta consulta devuelve los empleados cuyo salario es superior al salario promedio de todos los empleados.

#### 3. \*\*Funciones de Ventana (Window Functions)\*\*

- Las funciones de ventana permiten realizar cálculos sobre un conjunto de filas relacionadas con la fila actual. A diferencia de las funciones agregadas, las funciones de ventana no agrupan los resultados en una sola fila, sino que mantienen el detalle.

- Ejemplo de `ROW\_NUMBER`:

```sql

SELECT nombre, salario, ROW\_NUMBER() OVER (ORDER BY salario DESC) AS ranking

FROM empleados;

```

Esta consulta asigna un número de fila a cada empleado basado en el orden de su salario, desde el más alto hasta el más bajo.

#### 4. \*\*Operaciones de Conjuntos\*\*

- Las operaciones de conjuntos en SQL permiten combinar resultados de dos o más consultas utilizando operadores como `UNION`, `INTERSECT` y `EXCEPT`.

- `UNION` combina el resultado de dos consultas y elimina los duplicados, mientras que `UNION ALL` conserva los duplicados.

- Ejemplo de `UNION`:

```sql

SELECT nombre FROM clientes\_2023

UNION

SELECT nombre FROM clientes\_2022;

```

En este ejemplo, se obtienen los nombres de los clientes que han realizado transacciones en 2022 o 2023, sin duplicados.

#### 5. \*\*Consultas Recursivas (CTE - Common Table Expressions)\*\*

- Las CTEs permiten crear una consulta temporal que puede ser referenciada dentro de una consulta más grande. Son útiles para consultas recursivas o cuando necesitas organizar la lógica en pasos.

- Ejemplo de CTE:

```sql

WITH Empleados\_Con\_Salario AS (

SELECT nombre, salario

FROM empleados

WHERE salario > 5000

)

SELECT nombre FROM Empleados\_Con\_Salario;

```

En este ejemplo, primero se define la CTE `Empleados\_Con\_Salario` y luego se consulta a partir de ella para obtener los nombres de los empleados con salario superior a 5000.

#### Importancia de las Consultas Avanzadas

- Las consultas SQL avanzadas permiten manipular y analizar grandes volúmenes de datos de forma eficiente.

- Facilitan la resolución de problemas complejos, como la identificación de patrones en los datos, la generación de reportes detallados, y la integración de datos de diferentes fuentes.

- Son esenciales para roles de \*\*Data Analyst\*\*, \*\*Data Engineer\*\*, y \*\*Data Scientist\*\*, así como para cualquier profesional que trabaje con grandes bases de datos.

### 5 Preguntas para Retroalimentar

1. \*\*¿Cuál es la diferencia principal entre `INNER JOIN` y `LEFT JOIN`?\*\*

2. \*\*¿En qué casos es útil utilizar una subconsulta en lugar de un `JOIN`?\*\*

3. \*\*Explica el uso de la cláusula `WITH` en una CTE. ¿Por qué sería preferible usarla en lugar de subconsultas anidadas?\*\*

4. \*\*Describe una situación en la que sería más útil utilizar una función de ventana en lugar de una función agregada estándar (como `SUM`).\*\*

5. \*\*Dado el siguiente fragmento de código, ¿qué hace la consulta y cuál es el resultado esperado?\*\*

```sql

SELECT nombre, salario

FROM empleados

WHERE salario > (SELECT AVG(salario) FROM empleados WHERE departamento\_id = 3);

Respuestas

1. inner join nos muestra todas los registros que tengan cohicidencia entre las dos tablas, left join nos muestra los registros que tengan cohicidencia en una sola tabla

2. en el caso de querer hacer búsquedas anidadas sobre una misma tabla

3. la clausula with guarda una consulta para luego ser llamada, y seria útil en el caso de que la consulta guardada se use varias veces

4. en el caso de que queramos generas una función sobre cada registro sin agruparlos

5. la culsulta nos mostrara los nombres y los salarios de los empleados cuyo salario sea mayor al promedio y sean pertenecientes al departamento con id = 3  
  
excelentes preguntas

**Clase 48: `INNER JOIN` en SQL**

El `INNER JOIN` es un tipo de unión en SQL que permite combinar registros de dos tablas basándose en una condición de coincidencia. Solo devuelve las filas que tienen coincidencias en ambas tablas involucradas en la unión. Es útil cuando solo nos interesa trabajar con los datos que tienen una relación directa en ambas tablas.

#### Conceptos Clave de `INNER JOIN`:

- \*\*Sintaxis básica\*\*:

```sql

SELECT columnas

FROM tabla1

INNER JOIN tabla2

ON tabla1.columna = tabla2.columna;

```

- `SELECT`: especifica las columnas que deseamos mostrar.

- `INNER JOIN`: indica la combinación de dos tablas.

- `ON`: establece la condición que define cómo se relacionan las tablas.

- \*\*Ejemplo práctico\*\*:

Supongamos que tenemos dos tablas:

- \*\*empleados\*\*:

|  |  |  |
| --- | --- | --- |
| id\_empleado | nombre\_empleado | id\_departamento |
| 1 | Juan | 101 |
| 2 | Maria | 102 |
| 3 | Pedro | 103 |

- \*\*departamentos\*\*:

|  |  |
| --- | --- |
| id\_departamento | nombre\_departamento |
| 101 | Ventas |
| 102 | Recursos Humanos |
| 104 | IT |

Si queremos obtener una lista de empleados junto con el nombre del departamento en el que trabajan, podríamos usar un `INNER JOIN`:

```sql

SELECT empleados.nombre\_empleado, departamentos.nombre\_departamento

FROM empleados

INNER JOIN departamentos

ON empleados.id\_departamento = departamentos.id\_departamento;

\*\*Resultado\*\*:

|  |  |
| --- | --- |
| nombre\_empleado | nombre\_departamento |
| Juan | Ventas |
| Maria | Recursos Humanos |

- En este caso, el `INNER JOIN` solo devuelve a Juan y María, ya que ambos tienen un departamento asociado en la tabla `departamentos`. Pedro no aparece porque su `id\_departamento` (103) no tiene una coincidencia en la tabla `departamentos`.

#### Ventajas del `INNER JOIN`:

- \*\*Precisión\*\*: Solo devuelve las filas donde hay una relación definida en ambas tablas, lo que asegura que la información sea relevante y esté directamente relacionada.

- \*\*Optimización\*\*: Al reducir la cantidad de datos que se devuelven (solo las coincidencias), puede ser más eficiente en ciertos casos que un `LEFT JOIN` o `RIGHT JOIN` cuando no se necesita la información adicional de una de las tablas.

- \*\*Control de Datos\*\*: Es ideal para reportes y análisis donde solo se requieren los datos que cumplen ciertas condiciones entre tablas.

#### Desventajas del `INNER JOIN`:

- \*\*Datos excluidos\*\*: Los registros que no tienen coincidencias en ambas tablas no se muestran, lo que puede resultar en pérdida de información si no se manejan adecuadamente las uniones.

### 5 Preguntas de Retroalimentación:

1. \*\*¿Cuál es la principal diferencia entre un `INNER JOIN` y un `LEFT JOIN`?\*\*

2. \*\*¿Qué sucede si en un `INNER JOIN` no existen coincidencias entre las tablas?\*\*

3. \*\*¿Por qué podríamos preferir un `INNER JOIN` sobre un `FULL OUTER JOIN` en ciertos análisis de datos?\*\*

4. \*\*En una consulta con `INNER JOIN`, ¿qué función cumple la cláusula `ON`?\*\*

5. \*\*¿Qué se mostraría si en un `INNER JOIN` un empleado tiene un `id\_departamento` que no existe en la tabla `departamentos`?\*\*

Respuestas:

1. inner join trae las cohicidencias existententes entre ambas tablas, left join trae solo las cohicidencias de una tabla y los datos no encontrados los muestra null

2. no muestra ningún registro

3. porque necesitamos mostras registros relacionados de ambas tablas para generar análisis mas completos

4. establece la relación que une las tablas ejemplo: ON C.cliente\_id = P.cliente\_id

5. no mostrara dicho empleado

**Clase 49: LEFT JOIN**

#### Explicación detallada

Un `LEFT JOIN` (también conocido como `LEFT OUTER JOIN`) es una operación de unión en SQL que devuelve todos los registros de la tabla izquierda (la que aparece primero en la cláusula `FROM`) y las coincidencias de la tabla derecha (la que se menciona después del `JOIN`). Si no se encuentra una coincidencia en la tabla derecha, los campos correspondientes a esta tabla mostrarán `NULL`.

La estructura básica de un `LEFT JOIN` es la siguiente:

```sql

SELECT columnas

FROM tabla1

LEFT JOIN tabla2 ON tabla1.columna = tabla2.columna;

```

#### Ejemplo práctico

Supongamos que tenemos dos tablas:

- `clientes`:

- `cliente\_id` (ID del cliente)

- `nombre` (Nombre del cliente)

- `pedidos`:

- `pedido\_id` (ID del pedido)

- `cliente\_id` (ID del cliente que hizo el pedido)

- `monto` (Monto del pedido)

Queremos obtener una lista de todos los clientes y los pedidos que hayan realizado. Sin embargo, algunos clientes pueden no haber realizado ningún pedido. Para esto, usamos un `LEFT JOIN`:

```sql

SELECT c.cliente\_id, c.nombre, p.pedido\_id, p.monto

FROM clientes c

LEFT JOIN pedidos p ON c.cliente\_id = p.cliente\_id;

```

Este `LEFT JOIN` devolverá:

- Todos los clientes, incluso aquellos que no tengan pedidos (en cuyo caso, `pedido\_id` y `monto` serán `NULL`).

- Si un cliente tiene pedidos, se mostrarán los detalles de esos pedidos junto a la información del cliente.

Esto es útil en situaciones donde se necesita mantener todos los registros de la tabla principal (la izquierda) y solo agregar información adicional de la tabla relacionada cuando exista.

#### Uso común de `LEFT JOIN`

Un `LEFT JOIN` se utiliza cuando es importante que la tabla de la izquierda muestre todos sus registros, independientemente de si hay datos relacionados en la tabla de la derecha. Algunos casos prácticos incluyen:

- Mostrar todos los productos de un inventario, incluso aquellos que no tengan ventas.

- Obtener la lista de empleados de una empresa y las evaluaciones de desempeño (incluso si algunos no tienen evaluaciones registradas).

- Crear reportes de clientes y sus transacciones, aunque algunos clientes no hayan hecho ninguna compra.

### Preguntas de Retroalimentación

1. \*\*¿Cuál es la principal diferencia entre un `INNER JOIN` y un `LEFT JOIN`?\*\*

2. \*\*Si realizas un `LEFT JOIN` entre dos tablas y no hay coincidencia para un registro de la tabla izquierda, ¿qué valor mostrará en las columnas de la tabla derecha?\*\*

3. \*\*¿En qué casos sería mejor utilizar un `LEFT JOIN` en lugar de un `INNER JOIN`? Da un ejemplo.\*\*

4. \*\*¿Qué sucede si un cliente no tiene un pedido registrado en el ejemplo anterior? ¿Cómo se mostrarán los campos de `pedido\_id` y `monto`?\*\*

5. \*\*¿Cuál es la estructura básica de un `LEFT JOIN` en una consulta SQL?\*\*

### Pregunta de Repaso (Clase 1)

6. \*\*¿Por qué es importante aprender sobre bases de datos y qué beneficios trae su conocimiento en la gestión de información?\*\*

Esta clase y las preguntas de retroalimentación deberían proporcionarte una comprensión profunda del `LEFT JOIN` y su importancia en la manipulación de datos en SQL, así como una oportunidad para repasar conceptos fundamentales sobre bases de datos. ¡Espero que te sea útil para continuar desarrollando tus habilidades en SQL!

**Clase 50: `RIGHT JOIN` en SQL**

#### 1. ¿Qué es un `RIGHT JOIN`?

El `RIGHT JOIN` es una instrucción SQL que se utiliza para combinar registros de dos tablas basándose en una condición de coincidencia entre ellas. A diferencia del `LEFT JOIN`, el `RIGHT JOIN` devuelve todos los registros de la tabla a la \*\*derecha\*\* de la instrucción y solo aquellos registros coincidentes de la tabla a la \*\*izquierda\*\*. Si no hay coincidencias, los valores de las columnas de la tabla a la izquierda se mostrarán como `NULL`.

#### 2. ¿Cuándo usar un `RIGHT JOIN`?

El uso de `RIGHT JOIN` es adecuado cuando se necesita obtener todos los datos de una tabla específica (la tabla que está a la derecha en la consulta) y ver si hay registros coincidentes en la otra tabla. Es especialmente útil si la tabla derecha es la principal y la tabla izquierda contiene datos opcionales que no necesariamente están presentes para todos los registros de la tabla derecha.

#### 3. Sintaxis del `RIGHT JOIN`

La estructura básica de un `RIGHT JOIN` es la siguiente:

```sql

SELECT column\_names

FROM table1

RIGHT JOIN table2

ON table1.common\_column = table2.common\_column;

```

- `SELECT`: Define qué columnas mostrar en los resultados.

- `FROM table1`: Indica la tabla desde la que se inicia la combinación.

- `RIGHT JOIN table2`: Indica que se desea obtener todos los registros de `table2`, combinándolos con los que coincidan de `table1`.

- `ON`: Establece la condición de coincidencia entre las dos tablas.

#### 4. Ejemplo práctico de `RIGHT JOIN`

Supongamos que tenemos dos tablas: `clientes` y `pedidos`.

- Tabla `clientes`:

|  |  |
| --- | --- |
| cliente\_id | nombre |
| 1 | Juan |
| 2 | Maria |
| 3 | Pedro |

- Tabla `pedidos`:

|  |  |  |
| --- | --- | --- |
| pedido\_id | cliente\_id | producto |
| 101 | 1 | Laptop |
| 102 | 2 | Smartphone |
| 103 | 4 | Teclado |

Queremos obtener una lista de todos los pedidos y los nombres de los clientes que realizaron esos pedidos. Si un pedido no tiene un cliente asociado, queremos que aún aparezca en los resultados. La consulta sería:

```sql

SELECT p.pedido\_id, p.producto, c.nombre

FROM clientes c

RIGHT JOIN pedidos p

ON c.cliente\_id = p.cliente\_id;

```

- \*\*Resultado de la consulta\*\*:

|  |  |  |
| --- | --- | --- |
| pedido\_id | producto | nombre |
| 101 | Laptop | Juan |
| 102 | Smartphone | María |
| 193 | Teclado | NULL |

En este ejemplo, el `RIGHT JOIN` muestra todos los registros de la tabla `pedidos` y busca coincidencias con la tabla `clientes` basada en `cliente\_id`. Como el pedido 103 no tiene un `cliente\_id` coincidente en la tabla `clientes`, su campo `nombre` aparece como `NULL`.

#### 5. Importancia de `RIGHT JOIN`

El `RIGHT JOIN` es útil para analizar todos los registros de una tabla que siempre se deben mostrar, pero que pueden no tener un valor asociado en otra tabla. Esto permite identificar, por ejemplo, pedidos sin clientes o registros que aún no han sido completados. Es particularmente útil para informes donde queremos mostrar datos completos de una tabla y verificar su relación con otra.

---

### 5 Preguntas de Retroalimentación

1. \*\*¿Cuál es la diferencia principal entre un `LEFT JOIN` y un `RIGHT JOIN`?\*\*

2. \*\*Si no hay coincidencias en una columna de la tabla de la izquierda durante un `RIGHT JOIN`, ¿qué valor se mostrará en la consulta para esos registros?\*\*

3. \*\*Proporciona un ejemplo de cuándo sería útil usar un `RIGHT JOIN` en lugar de un `INNER JOIN`.\*\*

4. \*\*En la consulta de un `RIGHT JOIN`, ¿qué tabla siempre mostrará todos sus registros, independientemente de las coincidencias?\*\*

5. \*\*Escribe una consulta `RIGHT JOIN` para combinar las tablas `empleados` y `proyectos` donde se muestren todos los proyectos, incluso si no tienen empleados asignados.\*\*

---

### Pregunta de Repaso de la Clase 2: ¿Qué es una base de datos?

\*\*Pregunta\*\*: \*\*¿Cuáles son las principales ventajas de utilizar una base de datos para almacenar información en lugar de mantenerla en archivos de texto plano?\*\*

Estas preguntas y explicaciones te ayudarán a profundizar en la comprensión de los conceptos y usos del `RIGHT JOIN` en SQL y a repasar la importancia de las bases de datos en general. ¡Espero que esta clase sea útil para tu aprendizaje!

**Clase 51: FULL OUTER JOIN**

El `FULL OUTER JOIN` es una técnica de combinación en SQL que permite recuperar todos los registros de ambas tablas involucradas en la consulta, incluso si no hay coincidencias entre ellas. A diferencia de los `INNER JOIN`, que solo traen los registros que coinciden en ambas tablas, y de los `LEFT JOIN` o `RIGHT JOIN`, que priorizan una de las tablas, el `FULL OUTER JOIN` une las ventajas de ambos tipos de `JOIN`.

#### ¿Cómo funciona el FULL OUTER JOIN?

Cuando se realiza un `FULL OUTER JOIN`, SQL combina los registros de la primera tabla con los de la segunda. Si existen coincidencias en ambas tablas según la condición especificada en el `ON`, estas se presentan como una sola fila. Si no hay coincidencias, el `FULL OUTER JOIN` incluye igualmente el registro de la tabla sin coincidencias, mostrando `NULL` en las columnas de la otra tabla.

Esto puede ser especialmente útil en situaciones donde se desea obtener una visión completa de los registros de dos tablas, sin perder datos por la falta de coincidencias.

#### Ejemplo de FULL OUTER JOIN

Imagina que tienes dos tablas: `clientes` y `pedidos`. La tabla `clientes` almacena información sobre los clientes, mientras que la tabla `pedidos` contiene los pedidos realizados por cada cliente. No todos los clientes han hecho un pedido, y algunos pedidos no tienen un cliente registrado.

```sql

SELECT c.nombre AS nombre\_cliente, p.numero\_pedido

FROM clientes c

FULL OUTER JOIN pedidos p

ON c.id\_cliente = p.id\_cliente;

```

Con este `FULL OUTER JOIN`:

- Obtienes una lista de todos los clientes y todos los pedidos.

- Si un cliente no ha realizado ningún pedido, sus datos aparecerán con `NULL` en la columna `numero\_pedido`.

- Si hay un pedido que no está asociado a un cliente, aparecerá con `NULL` en la columna `nombre\_cliente`.

Esto garantiza que no se pierda información de ninguna de las dos tablas, proporcionando una perspectiva más completa para el análisis.

#### Ventajas del FULL OUTER JOIN

- \*\*Visión Integral\*\*: Permite ver todos los registros de ambas tablas, incluso si no hay coincidencias.

- \*\*Ideal para Reportes Completos\*\*: Es útil cuando se deben presentar todos los datos de dos tablas, especialmente si se quiere detectar qué elementos de una tabla no tienen coincidencias en la otra.

- \*\*Facilita la Detección de Inconsistencias\*\*: Al mostrar registros que no tienen coincidencias, es fácil identificar dónde podrían faltar datos o existir problemas en la integridad de la información.

#### Desventajas del FULL OUTER JOIN

- \*\*Rendimiento\*\*: Puede ser más lento que otros `JOIN` cuando se trabaja con grandes volúmenes de datos, ya que recupera todos los registros de ambas tablas.

- \*\*Datos Nulos\*\*: La presencia de muchos valores `NULL` en el resultado puede complicar el análisis si no se manejan adecuadamente en la consulta o el análisis posterior.

### Pregunta de Repaso de la Clase 3:

- \*\*¿Cuáles son algunos de los tipos de datos más comunes en SQL y para qué se utilizan?\*\*

---

### Preguntas de Retroalimentación sobre el FULL OUTER JOIN:

1. \*\*¿Cuál es la principal diferencia entre un `FULL OUTER JOIN` y un `INNER JOIN`?\*\*

2. \*\*Si realizas un `FULL OUTER JOIN` entre dos tablas, y uno de los registros de la primera tabla no tiene coincidencia en la segunda, ¿qué valor mostraría en las columnas de la segunda tabla?\*\*

3. \*\*¿En qué situaciones sería más adecuado usar un `FULL OUTER JOIN` en lugar de un `LEFT JOIN` o `RIGHT JOIN`?\*\*

4. \*\*¿Qué ocurre con los registros de la segunda tabla en un `FULL OUTER JOIN` cuando no tienen coincidencias en la primera tabla?\*\*

5. \*\*Escribe una consulta SQL utilizando `FULL OUTER JOIN` para combinar las tablas `productos` y `ventas`, donde la columna de referencia sea `id\_producto`.\*\*

**Clase 52: Subconsultas en SQL**

#### ¿Qué son las subconsultas en SQL?

Las subconsultas, también conocidas como consultas anidadas, son consultas SQL que se incluyen dentro de otra consulta principal. Su objetivo principal es recuperar datos que luego se utilizan como parte de la consulta principal. Son una herramienta poderosa en SQL porque permiten ejecutar operaciones complejas y obtener resultados de múltiples tablas sin necesidad de realizar varios pasos por separado.

#### Características de las Subconsultas

1. \*\*Ubicación\*\*: Las subconsultas pueden colocarse en diferentes partes de una consulta SQL:

- En la cláusula `SELECT`.

- En la cláusula `WHERE`.

- En la cláusula `FROM`.

2. \*\*Tipo de Resultado\*\*: Las subconsultas pueden devolver un único valor (escalar) o múltiples valores. Dependiendo de esto, pueden utilizarse con operadores como `=`, `IN`, `ANY`, o `EXISTS`.

3. \*\*Encapsulamiento\*\*: Las subconsultas están entre paréntesis, lo que permite que se ejecuten de manera independiente antes de ser utilizadas por la consulta principal. Esto ayuda a mantener la legibilidad del código y permite que el motor de base de datos procese cada parte de la consulta por separado.

#### Tipos de Subconsultas

1. \*\*Subconsultas Escalares\*\*: Devuelven un único valor. Se utilizan cuando se espera un resultado singular, como el promedio de una columna.

- Ejemplo:

```sql

SELECT nombre, salario

FROM empleados

WHERE salario > (SELECT AVG(salario) FROM empleados);

```

En este ejemplo, la subconsulta calcula el salario promedio y la consulta principal selecciona a los empleados cuyo salario es mayor a ese promedio.

2. \*\*Subconsultas de Múltiples Filas\*\*: Devuelven múltiples valores. Se utilizan con operadores como `IN` o `ANY` para comparar los resultados con varias filas.

- Ejemplo:

```sql

SELECT nombre

FROM productos

WHERE categoria\_id IN (SELECT id FROM categorias WHERE tipo = 'Electrónica');

```

Aquí, la subconsulta obtiene todos los `id` de categorías de tipo 'Electrónica' y la consulta principal selecciona los nombres de productos que pertenecen a esas categorías.

3. \*\*Subconsultas Correlacionadas\*\*: Dependiendo de los datos de la consulta principal para su ejecución, se ejecutan fila por fila.

- Ejemplo:

```sql

SELECT nombre

FROM empleados e

WHERE salario > (SELECT AVG(salario) FROM empleados WHERE departamento\_id = e.departamento\_id);

```

En este caso, la subconsulta se ejecuta para cada empleado, obteniendo el salario promedio del departamento del empleado actual.

#### Ventajas de Usar Subconsultas

- \*\*Modularidad\*\*: Permiten descomponer problemas complejos en consultas más pequeñas, facilitando la comprensión y el mantenimiento del código.

- \*\*Reutilización de Datos\*\*: Se pueden utilizar los resultados de una consulta en varias partes de una consulta principal sin necesidad de repetir la misma lógica.

- \*\*Flexibilidad\*\*: Permiten hacer cálculos complejos dentro de una sola consulta sin necesidad de crear tablas temporales o múltiples pasos.

#### Consideraciones para el Uso de Subconsultas

- \*\*Rendimiento\*\*: En algunas situaciones, las subconsultas pueden ser menos eficientes que `JOINs`, ya que el motor de la base de datos puede tener que ejecutar la subconsulta varias veces.

- \*\*Legibilidad\*\*: Aunque las subconsultas hacen que las consultas sean más compactas, pueden ser difíciles de leer si no se estructuran correctamente. Es importante utilizar nombres de alias y una correcta indentación.

---

### Preguntas de Retroalimentación

1. \*\*¿Cuál es la diferencia principal entre una subconsulta escalar y una subconsulta de múltiples filas?\*\*

2. \*\*¿En qué cláusulas de una consulta SQL se puede incluir una subconsulta?\*\*

3. \*\*¿Por qué se encapsulan las subconsultas entre paréntesis?\*\*

4. \*\*¿En qué situaciones es más ventajoso usar una subconsulta en lugar de un `JOIN`?\*\*

5. \*\*¿Qué resultado esperaría si una subconsulta utilizada en una cláusula `WHERE` no devuelve ningún valor?\*\*

### Pregunta de Repaso (Clase 4: Sistema Gestor de Bases de Datos)

\*\*¿Por qué es importante utilizar un Sistema Gestor de Bases de Datos (SGBD) en lugar de almacenar datos en archivos planos?\*\*

**Clase 53: \*\*Subconsulta Escalar\*\***

Una \*\*subconsulta escalar\*\* es una consulta anidada dentro de otra consulta que devuelve un solo valor. Este valor puede ser una fila y una columna específica o una función agregada como `SUM()`, `AVG()`, `MIN()`, etc., que da como resultado un solo dato.

El término "escalar" se refiere a la naturaleza del resultado: un único valor que puede ser tratado como una constante dentro de la consulta principal. Las subconsultas escalares se utilizan en situaciones en las que necesitamos obtener un valor específico, que luego puede usarse en la cláusula `SELECT`, `WHERE` o incluso como parte de un cálculo.

#### Sintaxis básica:

```sql

SELECT columna1,

(SELECT valor\_unico FROM tabla2 WHERE condición) AS valor\_calculado

FROM tabla1;

```

En este ejemplo, la subconsulta devuelve un solo valor que será parte del conjunto de resultados principal. Si la subconsulta no devuelve un valor o hay un error en la coincidencia de los datos, devolverá `NULL`.

#### Características clave de las subconsultas escalares:

1. \*\*Un solo valor\*\*: Devuelven exactamente un valor. Si devuelve más de uno, se produce un error.

2. \*\*Ubicación en la consulta\*\*: Se pueden colocar en la lista de selección (`SELECT`), en las cláusulas de filtrado (`WHERE`) o en cláusulas de agrupación (`GROUP BY`).

3. \*\*Relación con la consulta principal\*\*: Las subconsultas escalares son independientes de la consulta principal; su resultado se obtiene primero y luego se utiliza en la consulta externa.

#### Ejemplo:

```sql

SELECT nombre,

(SELECT MAX(sueldo) FROM empleados WHERE departamento\_id = d.departamento\_id) AS sueldo\_max

FROM departamentos d;

```

En este caso, la subconsulta devuelve el sueldo máximo por departamento, lo que permite mostrar el nombre del departamento junto con el sueldo más alto asociado a ese departamento.

### Ventajas de usar Subconsultas Escalares:

- \*\*Simplicidad\*\*: Permiten resolver ciertos problemas sin necesidad de realizar múltiples consultas.

- \*\*Reutilización de lógica\*\*: Podemos calcular valores en una subconsulta y luego usarlos directamente en la consulta principal.

### Desventajas:

- \*\*Rendimiento\*\*: En grandes conjuntos de datos, una subconsulta escalar puede ser menos eficiente que otras técnicas como los `JOINs`, ya que debe ejecutarse una vez por cada fila de la consulta principal.

- \*\*Restricciones\*\*: Si la subconsulta devuelve más de un valor, puede causar errores.

---

### Preguntas de Retroalimentación

1. ¿Qué valor devuelve una subconsulta escalar si no se encuentran registros que cumplan con la condición?

2. ¿En qué casos es más ventajoso usar una subconsulta escalar en lugar de un `JOIN`?

3. ¿Qué ocurre si una subconsulta escalar devuelve más de un valor? ¿Por qué podría suceder esto?

4. ¿Cómo impacta el uso de subconsultas escalares en el rendimiento de una consulta?

5. Escribe un ejemplo de cómo usarías una subconsulta escalar para calcular el precio promedio de los productos vendidos en una tienda.

---

### Pregunta de Repaso sobre la Clase 5

\*\*¿Cuáles son las principales funciones de un Sistema Gestor de Bases de Datos (SGBD) y por qué son esenciales para el manejo de datos?\*\*

¡Muy bien! Tus respuestas son correctas y demuestran una buena comprensión de los temas. Has captado la idea central de las subconsultas escalares y su impacto en el rendimiento, además de la importancia de los SGBD en la gestión eficiente de datos.

### A continuación, te enseñaré la \*\*Clase 54: Subconsulta de Múltiples Filas\*\*:

---

**Clase 54: \*\*Subconsulta de Múltiples Filas\*\***

A diferencia de la \*\*subconsulta escalar\*\*, que devuelve un único valor, la \*\*subconsulta de múltiples filas\*\* devuelve varios valores o registros. Se utiliza cuando se necesita comparar varios resultados con la consulta principal, y generalmente se usa con operadores que permiten trabajar con más de un resultado, como `IN`, `ANY`, `ALL`, etc.

#### Sintaxis básica:

```sql

SELECT columna1, columna2

FROM tabla1

WHERE columnaX IN (SELECT columnaY FROM tabla2 WHERE condición);

```

En este caso, la subconsulta devuelve varios registros de la columna `columnaY` de `tabla2`, y la consulta externa selecciona filas de `tabla1` que tengan coincidencias en la subconsulta.

#### Operadores para subconsultas de múltiples filas:

1. \*\*IN\*\*: Compara si un valor específico está dentro del conjunto de valores devueltos por la subconsulta.

```sql

SELECT nombre

FROM empleados

WHERE departamento\_id IN (SELECT departamento\_id FROM departamentos WHERE localización = 'España');

```

En este ejemplo, la subconsulta devuelve todos los `departamento\_id` donde la localización es "España", y la consulta principal selecciona a los empleados que pertenecen a esos departamentos.

2. \*\*ANY\*\*: Compara el valor con cualquier resultado de la subconsulta.

```sql

SELECT nombre

FROM empleados

WHERE salario > ANY (SELECT salario FROM empleados WHERE departamento\_id = 10);

```

Esto selecciona empleados cuyo salario es mayor que cualquier salario de los empleados en el departamento 10.

3. \*\*ALL\*\*: Compara el valor con todos los resultados de la subconsulta.

```sql

SELECT nombre

FROM empleados

WHERE salario > ALL (SELECT salario FROM empleados WHERE departamento\_id = 10);

```

Aquí, la consulta selecciona empleados cuyo salario es mayor que todos los salarios de los empleados en el departamento 10.

#### Características clave:

- \*\*Devuelve múltiples filas\*\*: Puede devolver varios resultados, que luego son comparados en la consulta principal usando operadores como `IN`, `ANY`, `ALL`.

- \*\*Filtrado basado en múltiples valores\*\*: Te permite filtrar datos en función de una lista de resultados.

#### Ejemplo:

```sql

SELECT nombre

FROM productos

WHERE categoría\_id IN (SELECT categoría\_id FROM ventas WHERE cantidad > 100);

```

En este caso, la subconsulta devuelve todos los `categoría\_id` de las ventas donde la cantidad es mayor que 100, y la consulta principal selecciona los nombres de los productos que pertenecen a esas categorías.

### Ventajas de las subconsultas de múltiples filas:

- \*\*Flexibilidad\*\*: Permiten comparar conjuntos de valores complejos.

- \*\*Simplicidad en algunos casos\*\*: A veces es más fácil escribir una subconsulta que hacer una combinación compleja con `JOINs`.

### Desventajas:

- \*\*Rendimiento\*\*: Como las subconsultas escalares, las subconsultas de múltiples filas pueden ser ineficientes con grandes conjuntos de datos.

- \*\*Errores si no se gestionan adecuadamente\*\*: Si se devuelve un conjunto vacío o más resultados de los necesarios, pueden producirse errores.

---

### Preguntas de Retroalimentación:

1. ¿Qué operador utilizamos para comparar un valor con cualquier resultado de una subconsulta de múltiples filas?

2. ¿Cómo se diferencia una subconsulta de múltiples filas de una subconsulta escalar?

3. ¿Qué sucede si una subconsulta de múltiples filas no devuelve ningún valor? ¿Cómo impacta esto en la consulta principal?

4. ¿En qué situaciones es más recomendable usar una subconsulta de múltiples filas en lugar de un `JOIN`?

5. Escribe una consulta que use `ALL` para comparar salarios entre empleados de dos departamentos diferentes.

---

### Pregunta de Repaso sobre la Clase 5:

\*\*¿Cuáles son las principales características que definen a un Sistema Gestor de Bases de Datos (SGBD)?\*\*

Continúa practicando con consultas SQL y subconsultas, ya que es una habilidad clave en el manejo de bases de datos. ¡Estás progresando muy bien!

¡Excelente que estés comprometido con ser un profesional en ingeniería de datos! Vamos a profundizar en la \*\*Clase 55: Subconsulta Correlacionada\*\*. Este es un concepto clave en SQL, y dominarlo te permitirá crear consultas más complejas y eficientes, lo cual es esencial para la ingeniería de datos.

**\*\*Clase 55: Subconsulta Correlacionada\*\***

#### \*\*¿Qué es una Subconsulta Correlacionada?\*\*

Una subconsulta correlacionada es un tipo de subconsulta que depende de una o más columnas de la consulta externa (principal). A diferencia de las subconsultas independientes, la subconsulta correlacionada se ejecuta fila por fila, utilizando los valores de cada fila de la consulta principal para evaluar los resultados. Esto significa que la subconsulta se ejecuta repetidamente para cada fila procesada por la consulta externa.

#### \*\*¿Cómo funciona?\*\*

La clave de las subconsultas correlacionadas es la referencia a columnas de la consulta principal dentro de la subconsulta. Cada vez que la consulta principal procesa una fila, la subconsulta se ejecuta, utilizando los datos de esa fila. Este tipo de consulta suele ser útil cuando necesitas comparar un valor de la fila actual con datos de otras filas en la misma tabla o en tablas diferentes.

#### \*\*Ejemplo básico:\*\*

Imaginemos que tienes una tabla `empleados` con las columnas `id`, `nombre`, `salario`, y quieres obtener el nombre y el salario de los empleados cuyo salario es mayor al salario promedio de su propio departamento.

La consulta sería algo así:

```sql

SELECT nombre, salario

FROM empleados e1

WHERE salario > (

SELECT AVG(salario)

FROM empleados e2

WHERE e2.departamento\_id = e1.departamento\_id

);

```

En este caso:

- La subconsulta selecciona el salario promedio de los empleados en el mismo departamento que el empleado de la fila actual de la consulta externa (`e1`).

- La subconsulta se ejecuta para cada fila de la tabla `empleados`, comparando el salario de la fila actual con el salario promedio de su departamento.

#### \*\*Ventajas y desventajas:\*\*

- \*\*Ventajas:\*\*

- Ofrecen una gran flexibilidad para consultas complejas.

- Permiten realizar operaciones que comparan registros dentro de la misma tabla sin necesidad de `JOINs` complicados.

- \*\*Desventajas:\*\*

- Pueden ser menos eficientes que las subconsultas simples o los `JOINs` debido a que la subconsulta se ejecuta múltiples veces, una por cada fila de la consulta principal.

- En grandes conjuntos de datos, las subconsultas correlacionadas pueden volverse costosas en términos de rendimiento.

#### \*\*Cuándo utilizar una subconsulta correlacionada:\*\*

Este tipo de consulta es útil cuando necesitas realizar comparaciones complejas fila por fila. Sin embargo, en situaciones donde el rendimiento es crítico, es importante analizar si un `JOIN` o alguna otra optimización podría ser más eficiente.

### \*\*5 Preguntas para Retroalimentar:\*\*

1. \*\*¿Cómo difiere una subconsulta correlacionada de una subconsulta simple?\*\*

2. \*\*En una subconsulta correlacionada, ¿cómo se referencia una columna de la consulta externa?\*\*

3. \*\*¿Qué sucede con el rendimiento de una consulta cuando se utiliza una subconsulta correlacionada en grandes conjuntos de datos?\*\*

4. \*\*¿Cuál es el propósito de una subconsulta correlacionada en el siguiente ejemplo?\*\*

```sql

SELECT id, nombre

FROM productos p1

WHERE precio > (

SELECT AVG(precio)

FROM productos p2

WHERE p2.categoria\_id = p1.categoria\_id

);

```

5. \*\*¿Cuáles son las ventajas y desventajas de las subconsultas correlacionadas?\*\*

### \*\*Pregunta de Repaso sobre la Clase 5 (Definición de SGBD):\*\*

\*\*¿Qué características clave debe tener un Sistema Gestor de Bases de Datos (SGBD) para garantizar la integridad y la disponibilidad de los datos en sistemas de misión crítica?\*\*

¡Continúa practicando y no dudes en preguntar si algo no queda claro! Convertirte en un ingeniero de datos experto requiere entender no solo cómo funcionan las consultas, sino también cómo optimizar el rendimiento y escalar los sistemas para manejar grandes volúmenes de datos. ¡Sigue adelante!

**\*\*Clase 55: Subconsulta Correlacionada\*\***

#### \*\*¿Qué es una Subconsulta Correlacionada?\*\*

Una subconsulta correlacionada es un tipo de subconsulta que depende de una o más columnas de la consulta externa (principal). A diferencia de las subconsultas independientes, la subconsulta correlacionada se ejecuta fila por fila, utilizando los valores de cada fila de la consulta principal para evaluar los resultados. Esto significa que la subconsulta se ejecuta repetidamente para cada fila procesada por la consulta externa.

#### \*\*¿Cómo funciona?\*\*

La clave de las subconsultas correlacionadas es la referencia a columnas de la consulta principal dentro de la subconsulta. Cada vez que la consulta principal procesa una fila, la subconsulta se ejecuta, utilizando los datos de esa fila. Este tipo de consulta suele ser útil cuando necesitas comparar un valor de la fila actual con datos de otras filas en la misma tabla o en tablas diferentes.

#### \*\*Ejemplo básico:\*\*

Imaginemos que tienes una tabla `empleados` con las columnas `id`, `nombre`, `salario`, y quieres obtener el nombre y el salario de los empleados cuyo salario es mayor al salario promedio de su propio departamento.

La consulta sería algo así:

```sql

SELECT nombre, salario

FROM empleados e1

WHERE salario > (

SELECT AVG(salario)

FROM empleados e2

WHERE e2.departamento\_id = e1.departamento\_id

);

```

En este caso:

- La subconsulta selecciona el salario promedio de los empleados en el mismo departamento que el empleado de la fila actual de la consulta externa (`e1`).

- La subconsulta se ejecuta para cada fila de la tabla `empleados`, comparando el salario de la fila actual con el salario promedio de su departamento.

#### \*\*Ventajas y desventajas:\*\*

- \*\*Ventajas:\*\*

- Ofrecen una gran flexibilidad para consultas complejas.

- Permiten realizar operaciones que comparan registros dentro de la misma tabla sin necesidad de `JOINs` complicados.

- \*\*Desventajas:\*\*

- Pueden ser menos eficientes que las subconsultas simples o los `JOINs` debido a que la subconsulta se ejecuta múltiples veces, una por cada fila de la consulta principal.

- En grandes conjuntos de datos, las subconsultas correlacionadas pueden volverse costosas en términos de rendimiento.

#### \*\*Cuándo utilizar una subconsulta correlacionada:\*\*

Este tipo de consulta es útil cuando necesitas realizar comparaciones complejas fila por fila. Sin embargo, en situaciones donde el rendimiento es crítico, es importante analizar si un `JOIN` o alguna otra optimización podría ser más eficiente.

### \*\*5 Preguntas para Retroalimentar:\*\*

1. \*\*¿Cómo difiere una subconsulta correlacionada de una subconsulta simple?\*\*

2. \*\*En una subconsulta correlacionada, ¿cómo se referencia una columna de la consulta externa?\*\*

3. \*\*¿Qué sucede con el rendimiento de una consulta cuando se utiliza una subconsulta correlacionada en grandes conjuntos de datos?\*\*

4. \*\*¿Cuál es el propósito de una subconsulta correlacionada en el siguiente ejemplo?\*\*

```sql

SELECT id, nombre

FROM productos p1

WHERE precio > (

SELECT AVG(precio)

FROM productos p2

WHERE p2.categoria\_id = p1.categoria\_id

);

```

5. \*\*¿Cuáles son las ventajas y desventajas de las subconsultas correlacionadas?\*\*

### \*\*Pregunta de Repaso sobre la Clase 5 (Definición de SGBD):\*\*

\*\*¿Qué características clave debe tener un Sistema Gestor de Bases de Datos (SGBD) para garantizar la integridad y la disponibilidad de los datos en sistemas de misión crítica?\*\*

¡Bienvenido a la clase 56 de Ingeniería de Datos! Hoy vamos a explorar un tema fundamental para optimizar las consultas en bases de datos: \*\*Índices en SQL\*\*. Los índices permiten a las bases de datos realizar consultas y búsquedas de datos de manera mucho más rápida y eficiente. Al comprender los conceptos detrás de los índices, cómo funcionan y cuándo aplicarlos, estarás más cerca de convertirte en un profesional en Ingeniería de Datos.

**Clase 56: Índices en SQL**

#### 1. ¿Qué es un índice en SQL?

Un índice en SQL es una estructura de datos que optimiza el acceso a los registros de una tabla, similar a cómo un índice en un libro facilita la búsqueda de temas específicos. Al crear un índice sobre una o varias columnas, estamos indicando al sistema que prepare esa columna para búsquedas y consultas más rápidas. Los índices son esenciales cuando trabajamos con grandes volúmenes de datos, ya que pueden reducir significativamente el tiempo de respuesta en las consultas.

#### 2. Tipos de Índices en SQL

Existen varios tipos de índices, cada uno diseñado para diferentes casos de uso:

- \*\*Índice Único (Unique Index)\*\*: Garantiza que los valores en una columna o grupo de columnas sean únicos.

- \*\*Índice Clustered\*\*: Ordena físicamente las filas en la tabla según el orden del índice. Una tabla solo puede tener un índice \*clustered\*.

- \*\*Índice No Clustered\*\*: No afecta el orden físico de las filas en la tabla, pero mantiene un puntero que se dirige a la ubicación real de los datos.

- \*\*Índice Compuesto (Composite Index)\*\*: Se crea sobre varias columnas, útil cuando es necesario optimizar consultas que involucran múltiples columnas.

- \*\*Índices de Texto Completo (Full-Text Index)\*\*: Permiten realizar búsquedas en columnas de texto, ideal para encontrar palabras o frases en campos extensos.

#### 3. Ventajas y Desventajas de los Índices

\*\*Ventajas\*\*:

- \*\*Velocidad en las consultas\*\*: Los índices optimizan las búsquedas en las bases de datos y mejoran el rendimiento en consultas complejas.

- \*\*Optimización de JOINs\*\*: Facilitan las combinaciones de tablas mediante claves de referencia.

- \*\*Eficiencia en el ordenamiento\*\*: Los índices permiten ordenar los datos de manera más rápida y eficaz.

\*\*Desventajas\*\*:

- \*\*Mayor consumo de almacenamiento\*\*: Cada índice ocupa espacio adicional en la base de datos.

- \*\*Impacto en las operaciones de escritura\*\*: Las inserciones, actualizaciones y eliminaciones son más lentas cuando hay índices en las columnas afectadas, ya que estos deben actualizarse con cada cambio.

#### 4. Creación de un Índice en SQL

Aquí te muestro cómo crear un índice en SQL:

```sql

CREATE INDEX nombre\_indice ON nombre\_tabla (columna1, columna2);

```

Para crear un índice único:

```sql

CREATE UNIQUE INDEX nombre\_indice ON nombre\_tabla (columna);

```

#### 5. Uso de Índices en Consultas

Los índices funcionan mejor en columnas que se utilizan frecuentemente en las condiciones de la cláusula `WHERE`, `JOIN`, `ORDER BY`, y `GROUP BY`. Sin embargo, no es recomendable crear índices en todas las columnas, ya que cada índice consume espacio y puede ralentizar las operaciones de escritura.

#### 6. Estrategias de Uso de Índices en Ingeniería de Datos

Como ingeniero de datos, una parte importante de tu trabajo será determinar qué columnas deben indexarse y en qué momentos hacerlo. Algunos puntos importantes a tener en cuenta son:

- \*\*Frecuencia de consulta\*\*: Si una columna se usa mucho en las consultas de búsqueda o filtrado, es un buen candidato para un índice.

- \*\*Volumen de datos\*\*: Los índices muestran sus ventajas especialmente en tablas grandes.

- \*\*Combinaciones de columnas\*\*: Si las consultas suelen filtrar por varias columnas, los índices compuestos pueden ser una solución eficiente.

#### 5 Preguntas de Retroalimentación sobre la Clase 56

1. ¿Cuál es la principal ventaja de usar índices en una base de datos?

2. Explica la diferencia entre un índice \*clustered\* y un índice \*no clustered\*.

3. ¿Por qué un índice compuesto es útil en columnas que se filtran conjuntamente?

4. ¿Cuáles son dos desventajas de los índices?

5. ¿En qué tipo de columnas no es recomendable usar índices? Explica por qué.

#### 1 Pregunta sobre la Clase 6 para Practicar

1. Explica brevemente tres características clave que debe cumplir un Sistema Gestor de Bases de Datos (SGBD) para ser eficiente y confiable.

**Clase 57: Restricciones SQL**

En la \*\*Clase 57: Restricciones SQL\*\*, exploraremos cómo implementar reglas dentro de las tablas de nuestras bases de datos para asegurar que los datos almacenados cumplan con los criterios de integridad y coherencia necesarios. Las restricciones son esenciales en la ingeniería de datos, pues garantizan que la base de datos respete sus propias reglas y minimiza errores en el ingreso o manipulación de los datos.

### Conceptos clave de la clase

1. \*\*Restricción\*\*: Una regla que se aplica a los datos de una tabla para garantizar su validez e integridad. Las restricciones controlan aspectos como la unicidad, la relación entre tablas y la obligación de ciertos campos.

2. \*\*Tipos principales de restricciones SQL\*\*:

- \*\*NOT NULL\*\*: Impide que una columna contenga valores nulos, es decir, asegura que siempre haya un valor en dicha columna.

- \*\*UNIQUE\*\*: Garantiza que todos los valores en una columna sean únicos, evitando duplicados en dicha columna.

- \*\*PRIMARY KEY\*\*: Define una columna o combinación de columnas que identifican de forma única cada fila de una tabla. La clave primaria es siempre única y no puede contener valores nulos.

- \*\*FOREIGN KEY\*\*: Establece una relación entre dos tablas al vincular una columna con la clave primaria de otra tabla. Ayuda a mantener la integridad referencial entre las tablas.

- \*\*CHECK\*\*: Permite aplicar condiciones específicas para los datos en una columna; por ejemplo, limitar un valor a un rango.

- \*\*DEFAULT\*\*: Asigna un valor predeterminado a una columna si no se proporciona un valor durante la inserción de datos.

3. \*\*Aplicación de restricciones\*\*:

- \*\*Creación de restricciones\*\*: Se pueden aplicar al crear una tabla con `CREATE TABLE` o se pueden agregar posteriormente usando `ALTER TABLE`.

- \*\*Eliminación de restricciones\*\*: Para remover una restricción, se utiliza `ALTER TABLE DROP CONSTRAINT`, seguido del nombre de la restricción.

4. \*\*Beneficios de las restricciones\*\*:

- Garantizan la integridad de los datos.

- Mejoran la fiabilidad de la base de datos.

- Facilitan el mantenimiento y la organización de los datos.

### Ejemplos básicos

- \*\*NOT NULL\*\*: Para asegurarse de que siempre haya un nombre:

```sql

CREATE TABLE EMPLEADOS (

ID INT PRIMARY KEY,

NOMBRE VARCHAR(50) NOT NULL

);

```

- \*\*UNIQUE\*\*: Para que no haya dos empleados con el mismo correo electrónico:

```sql

ALTER TABLE EMPLEADOS

ADD CONSTRAINT UNICO\_CORREO UNIQUE (CORREO);

```

- \*\*CHECK\*\*: Para asegurarse de que el salario sea positivo:

```sql

ALTER TABLE EMPLEADOS

ADD CONSTRAINT CHEQUEO\_SALARIO CHECK (SALARIO > 0);

```

### 5 Preguntas de retroalimentación de la Clase 57

1. ¿Cuál es la diferencia entre las restricciones `UNIQUE` y `PRIMARY KEY`?

2. ¿Qué tipos de restricciones garantizarían que no se puedan ingresar datos nulos en una columna?

3. ¿Para qué sirve la restricción `CHECK` y cómo ayuda en la calidad de los datos?

4. ¿Cómo se crea una clave foránea y cuál es su propósito principal en la relación entre tablas?

5. ¿Qué sucede si intentamos insertar datos que violan una restricción `UNIQUE`?

### Pregunta de práctica de la Clase 7 (Tipos de SGBD)

\*\*¿En qué situaciones se usaría un SGBD No Relacional en lugar de uno Relacional? Explica por qué.\*\*

**Clase 58 las transacciones en SQL**

En esta clase, vamos a explorar \*\*las transacciones en SQL\*\*, un tema clave para los sistemas de bases de datos, especialmente en el contexto de la ingeniería de datos. Las transacciones son bloques de operaciones que se ejecutan como una unidad de trabajo para asegurar la \*\*integridad de los datos\*\* y son esenciales para evitar inconsistencias en situaciones complejas de actualización y modificación de datos.

### 1. ¿Qué es una transacción en SQL?

Una transacción en SQL es un conjunto de operaciones que se ejecutan de manera atómica, es decir, o se completan todas o no se completa ninguna. Las transacciones ayudan a que las operaciones de modificación de datos (como `INSERT`, `UPDATE` o `DELETE`) mantengan los datos coherentes, incluso en situaciones de fallos o errores.

### 2. Propiedades ACID

Las transacciones en bases de datos deben cumplir con las propiedades \*\*ACID\*\*, que garantizan la \*\*confiabilidad y la coherencia de los datos\*\*:

- \*\*Atomicidad\*\*: asegura que todas las operaciones dentro de una transacción se completen en su totalidad. Si alguna falla, ninguna se aplica.

- \*\*Consistencia\*\*: garantiza que la transacción mantenga la base de datos en un estado válido.

- \*\*Aislamiento\*\*: asegura que las transacciones se ejecuten de manera independiente, evitando interferencias entre ellas.

- \*\*Durabilidad\*\*: una vez completada la transacción, los cambios son permanentes.

### 3. Ciclo de una transacción

Las transacciones generalmente siguen estos pasos:

1. \*\*BEGIN TRANSACTION\*\*: inicia la transacción.

2. \*\*Operaciones de modificación de datos\*\* (`INSERT`, `UPDATE`, `DELETE`, etc.)

3. \*\*COMMIT\*\*: confirma y guarda permanentemente los cambios.

4. \*\*ROLLBACK\*\*: revierte los cambios realizados en la transacción si ocurre un error.

#### Ejemplo de transacción básica

```sql

BEGIN TRANSACTION;

UPDATE cuentas

SET balance = balance - 1000

WHERE id = 1;

UPDATE cuentas

SET balance = balance + 1000

WHERE id = 2;

COMMIT;

```

En este ejemplo, si una de las operaciones falla, la transacción completa se revierte para evitar inconsistencias, y ningún cambio es aplicado.

### 4. Control de concurrencia en transacciones

Cuando varias transacciones se ejecutan simultáneamente, es esencial garantizar que el acceso concurrente no genere inconsistencias. Para esto, SQL usa \*\*niveles de aislamiento\*\* que definen cómo y cuándo los cambios realizados por una transacción son visibles para otras transacciones:

- \*\*Read Uncommitted\*\*: permite leer datos no confirmados, con el riesgo de inconsistencias.

- \*\*Read Committed\*\*: solo permite leer datos confirmados.

- \*\*Repeatable Read\*\*: asegura que si se lee un dato varias veces, su valor no cambia durante la transacción.

- \*\*Serializable\*\*: el nivel más restrictivo, garantiza la máxima consistencia.

### 5. Bloqueo y desbloqueo de recursos

Los \*\*bloqueos\*\* evitan que varias transacciones modifiquen simultáneamente los mismos datos, evitando problemas como la lectura sucia y la pérdida de actualizaciones. Existen diferentes tipos de bloqueos:

- \*\*Bloqueo compartido\*\*: permite que varias transacciones lean un dato, pero ninguna lo modifica.

- \*\*Bloqueo exclusivo\*\*: impide que otras transacciones lean o modifiquen un dato.

---

### Preguntas para retroalimentar

1. ¿Cuál es la función de una transacción en SQL y qué operaciones incluye?

2. ¿Qué significa la propiedad de atomicidad en las transacciones y por qué es importante?

3. Explica qué es un `ROLLBACK` y cuándo se utilizaría.

4. ¿Cuáles son los niveles de aislamiento y qué problema específico soluciona el nivel \*\*Serializable\*\*?

5. ¿Qué diferencia hay entre un bloqueo compartido y un bloqueo exclusivo en una transacción?

### Pregunta sobre la clase 8 para reforzar

1. ¿En qué contexto preferirías usar una base de datos relacional en vez de una no relacional y por qué?

**\*\*Clase 59: Bases de Datos Relacionales vs. Bases de Datos No Relacionales\*\***

En esta clase, vamos a analizar en profundidad las \*\*Bases de Datos Relacionales (SQL)\*\* y las \*\*Bases de Datos No Relacionales (NoSQL)\*\*, comprendiendo en qué se diferencian, sus principales usos y cómo seleccionar el tipo correcto de base de datos para distintos escenarios de trabajo en ingeniería de datos. Cada tipo tiene ventajas que le permiten desempeñarse de manera óptima según las necesidades de almacenamiento, procesamiento, y escalabilidad de los datos.

### 1. ¿Qué es una Base de Datos Relacional?

Una base de datos relacional organiza los datos en \*\*tablas\*\* que están relacionadas entre sí mediante claves primarias y foráneas, permitiendo la consistencia y la integridad en los datos almacenados. Este modelo, basado en \*\*SQL\*\* (Structured Query Language), asegura que cada tabla pueda ser consultada, modificada y manipulada con un lenguaje de consulta estandarizado y potente.

#### Ventajas de las Bases de Datos Relacionales:

- \*\*Consistencia\*\*: Aseguran la integridad referencial a través de claves y restricciones.

- \*\*Consultas Complejas\*\*: Soportan un alto nivel de complejidad en consultas.

- \*\*ACID\*\*: Cumplen con las propiedades de Atomicidad, Consistencia, Aislamiento y Durabilidad, lo que asegura la fiabilidad en transacciones.

#### Limitaciones de las Bases de Datos Relacionales:

- \*\*Escalabilidad Limitada\*\*: Escalarlas horizontalmente (agregar más servidores) es más complejo.

- \*\*Rigidez\*\*: Su estructura de datos es rígida; cada tabla debe cumplir un esquema predeterminado.

### 2. ¿Qué es una Base de Datos No Relacional?

Una base de datos no relacional, o \*\*NoSQL\*\*, se diseña para manejar grandes volúmenes de datos no estructurados o semiestructurados de manera flexible. En lugar de tablas, utilizan otros tipos de estructuras, como \*\*documentos, grafos, columnas\*\* o \*\*claves-valor\*\*, que permiten una mayor elasticidad en su escalabilidad y manejo.

#### Ventajas de las Bases de Datos No Relacionales:

- \*\*Escalabilidad Horizontal\*\*: Pueden crecer de forma horizontal fácilmente, distribuyendo la carga en varios servidores.

- \*\*Flexibilidad en el Esquema\*\*: No están limitadas por esquemas rígidos, lo que permite manejar datos de diferentes tipos en un mismo contenedor.

- \*\*Velocidad\*\*: Adecuadas para aplicaciones en tiempo real, gracias a su rapidez en la escritura y lectura de datos.

#### Limitaciones de las Bases de Datos No Relacionales:

- \*\*Consistencia Eventual\*\*: Aunque son rápidas, la consistencia total puede no ser inmediata.

- \*\*No ACID\*\*: En general, no cumplen con todas las propiedades ACID, sino que optan por una consistencia eventual.

### 3. Principales Tipos de Bases de Datos No Relacionales

1. \*\*Bases de Datos de Documentos\*\*: Utilizan estructuras tipo JSON o BSON, y son ideales para aplicaciones que manejan documentos no estructurados, como MongoDB.

2. \*\*Bases de Datos Clave-Valor\*\*: Simplifican el almacenamiento con pares clave-valor, óptimas para almacenar datos de configuración o sesiones de usuario (ej., Redis).

3. \*\*Bases de Datos de Grafos\*\*: Diseñadas para gestionar redes complejas de relaciones, como redes sociales o recomendaciones, con ejemplos como Neo4j.

4. \*\*Bases de Datos de Columnas Anchas\*\*: Diseñadas para manejar grandes volúmenes de datos distribuidos en columnas, muy utilizadas en Big Data (ej., Cassandra).

### 4. Cuándo Elegir SQL vs NoSQL

|  |  |  |
| --- | --- | --- |
| **Criterio** | **SQL** | **NoSQL** |
| **Datos estructurados** | Ideal | No es la mejor opción |
| **Relaciones complejas** | Altamente recomendable | Limitado, depende del tipo de NoSQL |
| **Consultas complejas** | Excelente soporte | Consultas complejas suelen ser limitadas |
| **Escalabilidad horizontal** | Más complejo | Altamente escalable |
| **Flexibilidad del esquema** | Rígido | Altamente flexible |
| **Propiedades ACID** | Cumple | Variable |
| **Volumen de datos** | Datos medianos o grandes | Grandes volúmenes de datos |

### Ejemplos de Casos de Uso

- \*\*SQL\*\*: Sistemas bancarios, ERP, CRM.

- \*\*NoSQL\*\*: Redes sociales, big data, análisis de datos en tiempo real, Internet de las Cosas (IoT).

---

### Preguntas de Retroalimentación

1. ¿Cuáles son las principales ventajas de las bases de datos relacionales frente a las no relacionales en términos de consistencia y transacciones?

2. Explica cómo la escalabilidad horizontal beneficia a las bases de datos NoSQL en aplicaciones de alto tráfico.

3. ¿Qué diferencia clave hay entre los tipos de bases de datos no relacionales de grafos y de columnas anchas?

4. ¿En qué casos resulta más recomendable el uso de una base de datos relacional?

5. ¿Qué tipo de base de datos NoSQL utilizarías para una aplicación de red social y por qué?

### Pregunta de Repaso de la Clase 9 (No Relacionales)

1. Menciona y explica dos características de las bases de datos no relacionales que las hacen diferentes de las bases de datos relacionales.

**Clase 60: \*\*Casos de Uso de las Bases de Datos Relacionales\*\***

Hoy nos adentraremos en el uso práctico de las bases de datos relacionales en distintos contextos, enfatizando las razones por las que son una opción popular en diversas industrias. Las bases de datos relacionales (RDBMS) son excelentes para gestionar información estructurada y compleja debido a su modelo de organización en tablas y su capacidad de mantener integridad y consistencia. Su eficacia en el manejo de grandes volúmenes de transacciones y su compatibilidad con SQL las hace idóneas en múltiples escenarios.

### Casos de Uso Principales

1. \*\*Finanzas y Banca\*\*: En este sector, las bases de datos relacionales son la columna vertebral de operaciones como las transferencias bancarias, pagos de facturas, y seguimiento de cuentas. Su habilidad para mantener integridad transaccional y garantizar la consistencia es crucial para realizar cálculos financieros con exactitud, gestionar el historial de transacciones y asegurar el cumplimiento de regulaciones.

2. \*\*Sistemas de Reservas\*\*: Aerolíneas, hoteles y empresas de transporte confían en bases de datos relacionales para gestionar reservas en tiempo real, permitiendo múltiples usuarios y garantizando la consistencia de los datos. Por ejemplo, si se reserva un asiento en un avión, el sistema debe reflejar inmediatamente la disponibilidad actualizada.

3. \*\*Comercio Electrónico y Retail\*\*: En las plataformas de venta en línea, los RDBMS permiten gestionar datos de inventario, clientes, órdenes de compra y detalles de productos. Además, facilitan el seguimiento de pedidos, devoluciones y la creación de un historial de compras, que ayuda a brindar una experiencia de usuario personalizada y mantener el control de inventarios en tiempo real.

4. \*\*Atención Médica\*\*: En hospitales y clínicas, las bases de datos relacionales almacenan historiales clínicos, registros de pacientes y citas. Su estructura garantiza que el acceso a la información médica esté correctamente organizado, y es vital para el manejo seguro y conforme a regulaciones de la información médica sensible.

5. \*\*Gobierno y Administración Pública\*\*: Para gestionar registros de ciudadanos, impuestos, y datos censales, los gobiernos dependen de bases de datos relacionales. La capacidad de manejar grandes volúmenes de datos de manera estructurada permite realizar un seguimiento preciso de estos datos y facilita la elaboración de informes y análisis.

### ¿Por Qué Elegir una Base de Datos Relacional?

- \*\*Integridad de los Datos\*\*: Las bases de datos relacionales mantienen la consistencia de los datos a través de restricciones de integridad, lo cual es esencial en sectores donde se necesita información precisa y confiable.

- \*\*Escalabilidad Vertical\*\*: Aunque las RDBMS no son tan escalables horizontalmente como algunas NoSQL, pueden manejar volúmenes de datos crecientes mediante mejoras en el hardware.

- \*\*Uso Extensivo de SQL\*\*: SQL es el estándar en las RDBMS, lo que facilita la interoperabilidad entre sistemas y permite el desarrollo de consultas complejas.

- \*\*Seguridad y Cumplimiento Normativo\*\*: Las bases de datos relacionales suelen tener integradas funciones de seguridad robustas y cumplen con normativas internacionales, lo que las hace seguras para el manejo de información sensible.

---

### Preguntas para Retroalimentación

1. ¿Por qué las bases de datos relacionales son esenciales en el sector bancario?

2. Describe cómo se utilizan las bases de datos relacionales en el comercio electrónico para mejorar la experiencia del usuario.

3. ¿Qué beneficios ofrece una base de datos relacional en términos de integridad y consistencia?

4. ¿Cómo contribuyen las RDBMS a la seguridad y cumplimiento en la administración pública?

5. Explica por qué SQL es un recurso importante en el manejo de bases de datos relacionales.

### Pregunta de Repaso (Clase 10)

- ¿En qué casos es útil implementar una base de datos multidimensional y qué ventajas ofrece frente a otros tipos de bases de datos?

**Clase 61: \*\*El Futuro de las Bases de Datos Relacionales\*\***

Las bases de datos relacionales (RDBMS) han sido la columna vertebral de la mayoría de los sistemas de almacenamiento de datos durante décadas. Con la creciente demanda de datos y la evolución de las aplicaciones, los RDBMS están adoptando innovaciones para mantenerse relevantes. Veamos algunas de las tendencias y tecnologías clave que marcarán su futuro:

### 1. \*\*Escalabilidad Híbrida y Distribuida\*\*

- Tradicionalmente, los RDBMS escalan verticalmente (añadiendo más recursos a un solo servidor), pero cada vez más se integran con arquitecturas de escalabilidad horizontal (distribuidas). Ejemplos incluyen bases de datos como Google Spanner y CockroachDB, que distribuyen datos de forma global manteniendo la consistencia transaccional.

### 2. \*\*Integración de Características NoSQL\*\*

- Para adaptarse a la demanda de alta velocidad y la flexibilidad de tipos de datos no estructurados, las bases de datos relacionales están comenzando a integrar características NoSQL. Así, combinan los beneficios de consistencia y transacciones con el almacenamiento flexible de documentos o valores clave, como lo hace PostgreSQL con soporte para JSON y MongoDB.

### 3. \*\*Soporte de Machine Learning Integrado\*\*

- Los RDBMS modernos están comenzando a incluir capacidades de Machine Learning directamente dentro de sus sistemas. Esto permite a los equipos entrenar y aplicar modelos directamente en la base de datos, eliminando la necesidad de transferir datos a sistemas externos. Ejemplos incluyen Oracle con AutoML y SQL Server con integración de R y Python.

### 4. \*\*Automatización e Inteligencia Artificial para la Gestión de Datos\*\*

- Se está trabajando en la implementación de IA para optimizar el rendimiento de las bases de datos, ajustando índices y configuraciones automáticamente. Esta tecnología, a menudo llamada "Autonomous Database", permite que las bases de datos se gestionen, mantengan y aseguren de manera autónoma, como las bases de datos autónomas de Oracle.

### 5. \*\*Compatibilidad con Cloud y Multicloud\*\*

- Los proveedores de bases de datos están diseñando sus sistemas para ser compatibles con entornos de múltiples nubes, lo que permite mover datos entre proveedores de nube con facilidad. Además, el soporte para arquitecturas "cloud-native" significa que los RDBMS pueden aprovechar la infraestructura de la nube y sus beneficios en flexibilidad y costos.

### 6. \*\*Aumento de la Seguridad y Cumplimiento de Normativas\*\*

- Con regulaciones como GDPR, las bases de datos relacionales están implementando tecnologías avanzadas de cifrado, auditoría y control de acceso. Esto garantiza que los datos estén protegidos en múltiples niveles, desde el almacenamiento hasta el acceso de los usuarios finales.

### 7. \*\*Compatibilidad con Blockchain para Auditoría Inmutable\*\*

- Se está explorando la integración con blockchain para crear registros inmutables y auditables de transacciones en las bases de datos relacionales. Esto es útil en industrias donde la trazabilidad y la inmutabilidad de los registros son cruciales, como en finanzas y cadenas de suministro.

### 8. \*\*Bases de Datos Cuánticas (Quantum Databases)\*\*

- Aunque aún en fases iniciales, las bases de datos cuánticas representan una posibilidad futura donde los RDBMS aprovechan la computación cuántica para procesar y almacenar datos a velocidades sin precedentes. Actualmente, la investigación se centra en crear algoritmos y estructuras de datos que puedan utilizarse en hardware cuántico.

---

### Preguntas de Retroalimentación

1. ¿Qué características de las bases de datos relacionales les permiten adoptar una arquitectura híbrida de escalabilidad?

2. ¿Por qué se está integrando Machine Learning directamente en algunos sistemas de bases de datos relacionales?

3. ¿Cuál es la importancia de la "Autonomous Database" y cómo ayuda en la gestión de datos?

4. Explica cómo el soporte para JSON en una base de datos relacional permite flexibilidad en el almacenamiento de datos.

5. ¿Qué beneficios aporta la compatibilidad multicloud a los sistemas de bases de datos relacionales?

### Pregunta de Repaso sobre la Clase 11 (Bases de Datos Distribuidas)

¿Cuáles son las ventajas de una base de datos distribuida frente a una base de datos centralizada en términos de escalabilidad y redundancia de datos?