

A woman with long brown hair, wearing a light blue and white striped lab coat over a white shirt and white trousers, stands in a modern laboratory. She is smiling and looking towards the camera, with her hands resting on a black server rack. The server rack has a mesh door and visible internal components. In the background, there are computer workstations with monitors and keyboards, and a blue wall with white chemical structures. A circular logo in the top right corner reads 'MISAE ACOSTA INSTITUTO UNIVERSITARIO' around a green stylized book icon.



## TERCER CICLO

# Base de Datos II.

**Autor:**

Ing. Myriam Aracely Caiza Analuisa  
Ing. Francisco Javier Lobato Merino

**Edición:**

Primera

**Editorial:**

Instituto Superior Tecnológico Dr. Misael Acosta Solís

**Créditos:**

- Ing. Doris Pamela Chapalbay Chungata
- Ing. Cristian Patricio Guayanlema Fajardo

**Revisión de contenidos (pares):**

- Ing. Edwin F. Tipan T. Mgs.
- Ing. Ignacio Chuiluisa

**Revisión y corrección externa**

- Dra. María Gabriela Oña Velástegui

**Diseño y diagramación:**

- Mgtr. Gabriela Puentes
- Lic. Bryan Daniel Ortega
- Ing. Alex Santillán Aguirre

**ISBN:**

978-9942-7240-5-2

**Riobamba - Ecuador**

|                                    |    |
|------------------------------------|----|
| Introducción a la asignatura       | 05 |
| Orientaciones generales de estudio | 06 |
| OBJETIVO                           | 06 |
| PLANIFICACIÓN DEL APRENDIZAJE      | 07 |

## 08 UNIDAD 1

|                                                              |           |
|--------------------------------------------------------------|-----------|
| <b>1. Introducción, Normalización, Comandos SQL</b>          | <b>09</b> |
| Introducción                                                 | 10        |
| <b>1.1. Introducción al modelo entidad- relación</b>         | <b>11</b> |
| 1.1.1. Modelo entidad relación                               | 11        |
| 1.1.2. Modelo lógico                                         | 14        |
| 1.1.3. Claves Primarias                                      | 16        |
| 1.1.4. Claves Foráneas Únicas                                | 18        |
| <b>1.2. Normalización</b>                                    | <b>19</b> |
| 1.2.1. Primera forma normal                                  | 20        |
| 1.2.2. Segunda forma normal                                  | 21        |
| 1.2.3. Tercera forma normal                                  | 21        |
| 1.2.4. Cuarta forma normal                                   | 22        |
| 1.2.5. Quinta forma normal                                   | 24        |
| <b>1.3. Comandos SQL en el Gestor de base de datos MySQL</b> | <b>24</b> |
| 1.3.1. Comandos DDL                                          | 24        |
| 1.3.1.1. Create                                              | 24        |
| 1.3.1.2. Drop                                                | 27        |
| 1.3.1.3. Alter                                               | 27        |
| 1.3.2. Comandos DML                                          | 27        |
| 1.3.2.1. Insert                                              | 28        |
| 1.3.2.2. Update                                              | 28        |
| 1.3.2.3. Delete                                              | 28        |
| 1.3.2.4. Select                                              | 29        |
| ACTIVIDADES RECOMENDADAS                                     | 30        |

## 31 UNIDAD 2

|                                                |           |
|------------------------------------------------|-----------|
| <b>2. Reglas lógicas, INNER JOIN y TRIGGER</b> | <b>32</b> |
| Introducción                                   | 32        |
| Objetivo                                       | 33        |
| <b>2.1. Reglas lógicas</b>                     | <b>34</b> |
| <b>2.2. Inner Join</b>                         | <b>35</b> |
| <b>2.3. Trigger o disparador</b>               | <b>38</b> |
| ACTIVIDADES RECOMENDADAS                       | 42        |

## 43 UNIDAD 3

|                                        |           |
|----------------------------------------|-----------|
| <b>3.1. Vistas</b>                     | <b>46</b> |
| <b>3.2. Procedimientos almacenados</b> | <b>49</b> |
| ACTIVIDADES RECOMENDADAS               | 53        |

## **54** UNIDAD 4

|                                                       |        |
|-------------------------------------------------------|--------|
| Objetivo.....                                         | 55     |
| 4.1. Administración.....                              | 56     |
| 4.2. Backup.....                                      | 62     |
| 4.3. Reingeniería de base de datos con Workbench..... | 68     |
| ACTIVIDADES RECOMENDADAS.....                         | 73     |
| <br>BIBLIOGRAFÍA BÁSICA.....                          | <br>74 |
| BIBLIOGRAFÍA DE CONSULTA.....                         | 74     |

## INTRODUCCION A LA ASIGNATURA

Base de datos II es un curso de informática y gestión de la información, proporciona a los estudiantes una base sólida en los conceptos y principios básicos relacionados con las bases de datos que son esenciales en la era digital en la que vivimos. En Base de datos II, los estudiantes exploran varios temas clave, que incluyen:

Conceptos básicos de bases de datos: los estudiantes aprenden qué son las bases de datos, por qué son importantes y cómo usarlas en una variedad de aplicaciones y entornos.

Modelos de datos: presenta uno de los modelos más utilizados en la industria, como el modelo relacional, y explora conceptos como tablas, registros y campos.

Lenguaje SQL: presenta los conceptos básicos de SQL

(lenguaje de consulta estructurado), que se utiliza para interactuar con bases de datos relacionales para que los datos se puedan crear, leer, actualizar y eliminar.

Integridad de los datos: los estudiantes comprenden la importancia de mantener la integridad y coherencia de los datos a través de restricciones y reglas de validación.

Consultas y recuperación de datos: le enseña cómo ejecutar consultas para recuperar información específica de una base de datos y cómo utilizar filtros y condiciones.

Normalización: permite la exploración de técnicas de normalización para reducir la duplicación de datos y mejorar la eficiencia y calidad de la base de datos.

**BIENVENIDOS/AS Y ÉXITOS EN SUS ESTUDIOS**

## ORIENTACIONES GENERALES PARA EL ESTUDIO

**Antes de adentrarse** en temas más complejos, es esencial adquirir conocimientos básicos sobre bases de datos, como tablas, registros, claves primarias y relaciones entre tablas.

**Se recomienda estudiar el modelo relacional**, ampliamente utilizado en la gestión de bases de datos, para comprender la estructura de las tablas y las relaciones establecidas entre ellas.

**Practicar la escritura de consultas SQL es importante**, ya que es el lenguaje utilizado para interactuar con las bases de datos. Realizar ejercicios y familiarizarse con las consultas SQL ayuda a obtener información específica de las bases de datos.

**Estudiar la normalización es esencial para el diseño de bases de datos**, ya que permite eliminar redundancias y mejorar la eficiencia. Es importante conocer los diferentes niveles de normalización y saber aplicarlos correctamente.

**Realizar proyectos prácticos** es una excelente manera de aplicar los conocimientos adquiridos. Desarrollar aplicaciones que utilicen bases de datos permite poner en práctica las habilidades y comprender cómo se integran en el desarrollo de software.

### OBJETIVO

Desarrollar en el estudiante la habilidad para utilizar el lenguaje de operaciones comerciales SQL y su capacidad para normalizar un modelo lógico y definirlo en un modelo físico, con ética y responsabilidad.

## PLANIFICACIÓN DEL APRENDIZAJE

| CONTENIDOS                                                   | ACTIVIDADES DE APRENDIZAJE EN HORAS |           |           |
|--------------------------------------------------------------|-------------------------------------|-----------|-----------|
|                                                              | DOCE                                | PRACT     | AUTO      |
| <b>UNIDAD 1</b><br>Introducción, Normalización, Comandos SQL | 06                                  | 14        | 18        |
| <b>UNIDAD 2</b><br>Reglas lógicas, INNER JOIN y TRIGGER      | 06                                  | 10        | 18        |
| <b>UNIDAD 3</b><br>Vistas y procedimientos almacenados       | 06                                  | 12        | 18        |
| <b>UNIDAD 4</b><br>Administración, backup y reingeniería     | 06                                  | 12        | 18        |
| <b>Total:</b>                                                | <b>24</b>                           | <b>48</b> | <b>72</b> |

# UNIDAD 1

INTRODUCCIÓN, NORMALIZACIÓN, COMANDOS SQL



# Introducción, Normalización, Comandos SQL

## Introducción

Las bases de datos desempeñan un papel vital en el vasto mundo de la informática y la gestión de la información. Están en el corazón de la mayoría de las aplicaciones y sistemas que utilizamos todos los días, desde redes sociales hasta aplicaciones comerciales y sitios web. La gestión eficiente de los datos es esencial para una organización, almacenamiento y acceso eficaces a la información. Esta introducción cubre tres conceptos principales: bases de datos, normalización y comandos SQL.

Una base de datos es un conjunto organizado de datos que se almacena sistemáticamente. Puede incluir información sobre personas, productos, transacciones, registros financieros, etc. Las bases de datos permiten una gestión y recuperación eficiente de datos, lo que facilita la toma de decisiones informadas y la automatización de procesos.

La normalización es un proceso esencial en el diseño de bases de datos relacionales. El objetivo es estructurar de forma óptima los datos para reducir la redundancia y mejorar la integridad de los datos. La aplicación de la normalización divide una tabla grande en tablas relacionadas más pequeñas. Esto no sólo ahorra espacio en disco, sino que también garantiza la precisión y coherencia de los datos.

Comandos SQL, o Structured Query Language, es el lenguaje estándar que permite gestionar bases de datos relacionales. Con SQL, los usuarios pueden realizar una variedad de operaciones, como insertar nuevos datos, recuperar información específica, actualizar registros existentes y eliminar datos. Los comandos SQL, como DDL y DML, son herramientas poderosas para interactuar con bases de datos y extraer información relevante.

## Objetivo

Utilizar comandos SQL para la construcción de esquemas de bases de datos garantizando la eficiencia, integridad y consistencia en el manejo de los datos almacenados.

## Introducción al modelo entidad- relación

El Modelo Entidad-Relación (ER) es una técnica gráfica que se utiliza para representar la estructura de una base de datos, identificando entidades, atributos y relaciones. Por otro lado, el Modelo Lógico de datos es la fase posterior en el diseño de bases de datos, donde se traducen los elementos del modelo ER en tablas, claves primarias, foráneas y restricciones, brindando un enfoque más detallado y específico para la implementación de la base de datos. El ER es una herramienta conceptual y visual, mientras que el Modelo Lógico se enfoca en la implementación práctica de la base de datos.

### Modelo entidad relación

El modelo de datos Entidad-Relación (E-R) se basa en una percepción del mundo real que consta de objetos básicos llamados entidades y las relaciones entre esos objetos. Fue desarrollado para el diseño de bases de datos sea más fácil al permitir la especificación de un esquema empresarial que represente la estructura lógica completa de la base de datos. Los modelos E-R son muy útiles para mapear significados e interacciones empresariales que existen en el mundo real con esquemas conceptuales. Por medio de esta utilidad, muchas herramientas de diseño de bases de datos se basan en el concepto del modelo E-R. (Silberschatz et al., 2002)

El modelo entidad-relación (ER) es una herramienta crucial en el diseño de bases de datos ya que ayuda a representar de manera efectiva la estructura de los datos, facilita la comunicación entre los equipos de desarrollo y las partes interesadas, y contribuye a la creación de bases de datos eficientes y mantenibles. A continuación, se describen algunos beneficios:

**Claridad y comprensión:** El modelo ER proporciona una representación gráfica que es fácil de entender para los diseñadores y las partes interesadas. Permite visualizar de manera clara las entidades, relaciones y atributos, lo que facilita la comunicación y comprensión de la estructura de los datos.

**Abstracción de datos:** Permite abstraer y simplificar la complejidad de los datos del mundo real en entidades, relaciones y atributos, lo que simplifica el proceso de diseño y la gestión de bases de datos.

**Diseño estructurado:** Ayuda a definir de manera sistemática cómo se organizan los datos y cómo se relacionan entre sí, lo que guía el proceso de diseño de bases de datos de manera estructurada.

**Identificación de requerimientos:** Facilita la identificación de los requisitos del sistema al modelar las entidades y relaciones, lo que ayuda a comprender mejor las necesidades del negocio.

**Mantenibilidad:** El modelo ER actúa como una hoja de ruta para el desarrollo y mantenimiento de la base de datos, lo que simplifica la tarea de realizar cambios o mejoras en la estructura de datos.

**Integración de datos:** Ayuda a integrar datos de diferentes fuentes o sistemas al definir cómo se relacionan las entidades, lo que es esencial en entornos empresariales complejos.

**Normalización:** Facilita la normalización de la base de datos al ayudar a eliminar la redundancia de datos y mejorar la integridad de los datos.

**Optimización del rendimiento:** Permite diseñar bases de datos de manera más eficiente, lo que puede mejorar el rendimiento de las consultas y operaciones de bases de datos.

Según (ESIC Business & Marketing School., 2018) hay tres elementos básicos del modelo entidad-relación:

**Entidades:** El modelo tiene una entidad para cada componente del proceso de negocio. Por ejemplo, una tienda de suscripción a revistas puede tener entidades como "Cliente", "Dirección", "Factura", "Producto" e "Incidente".

**Atributos:** Un componente fundamental del Modelo Entidad-Relación, los atributos le permiten describir las propiedades de una entidad. Nombre,

Nombre, Segundo Apellido, Fecha de Nacimiento, Sexo o Segmento de Valor son atributos de la entidad Cliente.

Relación: Una relación establece una conexión entre un par de entidades. Cada Cliente recibe una suscripción, se suscribe a uno o más Productos y recibe una dirección de envío desde la cual puede recibir Facturas con la frecuencia acordada.

### **Símbolos utilizados en el modelo entidad-relación**

GeeksforGeeks, explica que el modelo ER es utilizado para modelar la vista lógicadel sistema viéndolo desde la perspectiva de datos que consta de los siguientes símbolos:

Rectángulos: Los rectángulos representan a las entidades en el Modelo E-R.

Elipses: representan a los atributos en el modelo E-R.

Diamantes: representan relaciones que existen entre entidades.

Doble diamante: representa una relación débil, quiere decir que el hijo depende de la entidad padre.

Líneas: representan atributos a entidades y conjuntos de entidades con otros tipos derelación.

Elipse Doble: representan a los atributos multivaluados.

Doble Rectángulo: representa a una entidad débil.

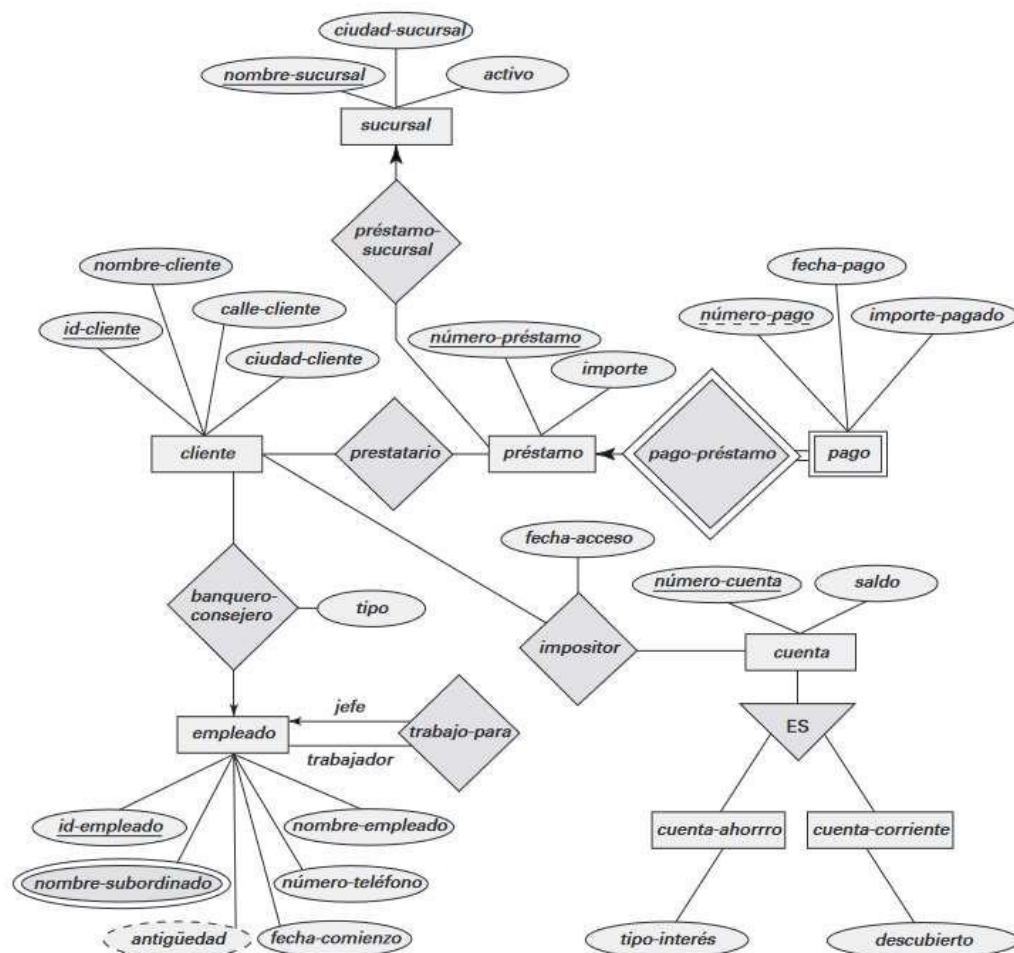


Figura 1. Diagrama E-R para un banco

(Silberschatz et al., 2002)

## Modelo lógico

Un modelo de datos lógico tiene por objetivo establecer la estructura de los elementos de datos y las relaciones existentes entre ellos. Es totalmente independiente de la base de datos física que detalla cómo se implementarán los datos, también es utilizado como modelo para los datos manejados. El modelo de datos lógicos carga los elementos del modelado de datos conceptuales un paso más allá al agregarles más información. (Cloud Software Group, Inc., 2023)

Los pasos para diseñar el modelo de datos lógicos son los siguientes:

Especificar las claves primarias para todas las entidades.

Encontrar las relaciones entre diferentes entidades.

Encontrar todos los atributos para cada entidad.

Resolver las relaciones de muchos a muchos.

Normalización.

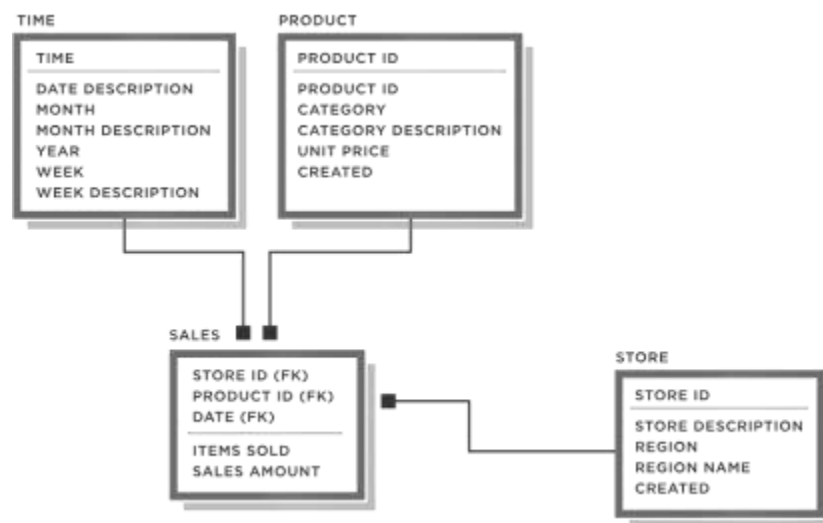


Figura 2. Ejemplo de modelo lógico (Cloud Software Group, Inc., 2023)

### Características de un modelo de datos lógico

Según (Cloud Software Group, Inc., 2023) estas son las características más importantes de un modelo de datos lógico:

Un modelo de datos lógico logra describir los requisitos de datos de un proyecto individual. Ha sido diseñado para fusionarse perfectamente con otros modelos de datos lógicos si algún proyecto lo requiere.

Se puede realizar el desarrollo y diseño de un modelo de datos lógico independiente del SGBD. El tipo de sistema de gestión de bases de datos influye poco en esto.

Los atributos de datos dominan tipos de datos con longitud y las precisiones son exactas.

En el modelado de datos lógico, no se definen llaves (primaria o foránea). A esta altura de modelado de datos, es prudente verificar y modificar los detalles del conector que se establecieron antes de definir las relaciones.

Es una representación gráfica de los requisitos de información de un área empresarial. No es considerado una base de datos ni un sistema de gestión de bases de datos en sí.

Actúa de forma independiente de cualquier dispositivo físico de almacenamiento de datos.

Debe ser diseñado para que sea independiente de la tecnología, de modo que no se vea afectado por los rápidos cambios de la tecnología.

### Claves Primarias

La clave primaria de una tabla es una columna cuyo valor difiere para cada fila. Como son diferentes, cada fila es única. Si no existe tal columna, la clave principal es una combinación de dos o más columnas cuyos valores combinados son diferentes en cada fila. Todas las tablas de su modelo deben tener una clave primaria. Esta regla se deriva automáticamente de la regla de que todas las filas deben ser únicas. Una clave principal se compone de todas las columnas, si se desea. No es recomendable usar cadenas largas como claves principales. (IBM Corporation, 2021b)

Las claves primarias desempeñan un papel vital en el diseño y la gestión de bases de datos. A continuación, se detallan algunos de los beneficios clave de usar claves primarias en su base de datos.

Unicidad: una clave principal garantiza que cada fila de una tabla tenga un valor único, lo que evita la duplicación de datos y garantiza la integridad de los datos. Identificación única: se utiliza una clave principal para identificar de forma única cada registro en una tabla, lo que facilita la recuperación y actualización de información específica.

Referencias cruzadas: las claves primarias pueden servir como puntos de referencia para establecer relaciones con otras tablas, lo que facilita el establecimiento de relaciones entre datos en una base de datos relacional.

Indexación: las bases de datos a menudo crean automáticamente índices en columnas de clave principal, lo que puede acelerar las consultas y las búsquedas, mejorando así el rendimiento.

Integridad de los datos: las claves primarias ayudan a mantener la integridad de los datos al evitar que se inserten entradas duplicadas o incorrectas en una tabla. Restricciones de integridad referencial: la clave principal sirve como base para definir restricciones de integridad referencial para garantizar que las relaciones entre tablas sean coherentes y se ajusten a las reglas de integridad.

Facilidad para actualizar y eliminar registros: las claves primarias le permiten identificar datos con precisión, simplificando la tarea de cambiar o eliminar registros específicos en una tabla. Optimización de consultas: el uso de claves primarias en las consultas proporciona un acceso más rápido a los datos, mejorando así el rendimiento de las operaciones de recuperación de datos.

Cumplimiento de estándares: el uso de claves primarias es una práctica recomendada en el diseño de bases de datos que facilita a los desarrolladores y administradores la comprensión y el mantenimiento de la base de datos

Para eficacia, la clave primaria debe ser de los siguientes tipos:

- Numérico (INT o SMALLINT)
- Serie (BIGSERIAL, SERIAL o SERIAL8)

Una serie de caracteres corta (como la que se utiliza para códigos).

Los valores NULL nunca se permiten en las columnas de clave primaria, tampoco se pueden comparar. Es decir, no se puede asegurar si son similares o diferentes. Por lo tanto, una fila no puede ser única en comparación con otras. Una columna no puede ser parte de una clave primaria si la misma permite valores nulos. Cuando define una restricción PRIMARY KEY, el

servidor de la base de datos también crea implícitamente una restricción NOT NULL en la misma columna o conjunto de columnas que componen la clave principal. (IBM Corporation, 2021b)

### Claves Foráneas Únicas

Una clave foránea se puede definir como una columna o grupo de columnas en una tabla que contiene valores que coinciden con la clave principal en una tabla distinta. Las claves foráneas son utilizadas para unir tablas. En la Figura 3 se puede observar las claves primaria y foránea de las tablas Cliente y Pedido en la base de datos de ejemplo. (IBM Corporation, 2021a)

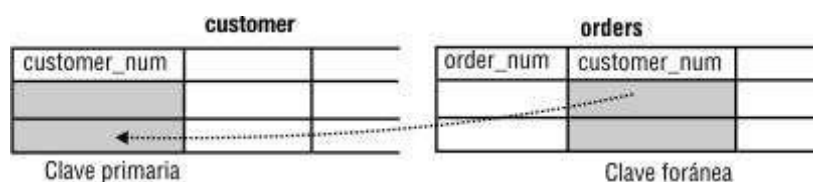


Figura 3. Claves primaria y foránea en las relaciones de cliente y pedido  
(IBM Corporation, 2021a)

Las claves foráneas se muestran cuando aparecen en el modelo porque su presencia podría limitar su capacidad para eliminar filas de tablas.

Las claves foráneas únicas son un componente importante en el diseño de bases de datos relacionales, ya que ayudan a establecer relaciones de datos significativas, garantizan la integridad referencial y facilitan la realización de consultas y operaciones de bases de datos que involucran múltiples tablas.

A continuación, se explican los aspectos clave de las claves foráneas únicas:

**Relaciones entre tablas:** Las claves foráneas únicas se utilizan para vincular una tabla con otra, creando una relación entre ellas. Esta relación permite que los datos se relacionen de manera significativa.

**Unicidad:** Al igual que una clave primaria, una clave foránea única garantiza que los valores en la columna correspondiente sean únicos en la tabla a la

que se hace referencia. Esto impide la duplicación de datos en la tabla relacionada.

**Integridad referencial:** La presencia de una clave foránea única garantiza la integridad referencial en la base de datos. Esto significa que los valores en la tabla que contiene la clave foránea deben coincidir con los valores en la tabla a la que se hace referencia. Si un valor no existe en la tabla referenciada, la base de datos no permitirá la inserción o actualización de datos que violen esta integridad.

**Utilidad en restricciones:** Las claves foráneas únicas son útiles para establecer restricciones de integridad referencial. Esto impide que se creen registros en la tabla que contiene la clave foránea si no se cumplen las condiciones de referencia a la tabla relacionada.

**Relaciones uno a uno:** Las claves foráneas únicas a menudo se utilizan en relaciones uno a uno, donde un registro en una tabla se relaciona con un único registro en otra tabla y viceversa.

**Utilización en consultas:** Las claves foráneas únicas permiten realizar consultas y operaciones que relacionan datos entre tablas, lo que facilita la recuperación de información relacionada.

## Normalización

Según (IBM Corporation, 2021c) normalización es una orientación formal que emplea un conjunto de normas para relacionar atributos a entidades y al normalizar un modelo de datos, se logra obtener los siguientes objetivos:

- Producir mayor flexibilidad en el diseño.
- Asegurarse que los atributos se ubican en las tablas correctas.
- Reducir la redundancia de datos.
- Aumentar la eficacia del programador.
- Reducir los costes de mantenimiento de la aplicación.
- Maximizar la estabilidad de la estructura de datos.

Para reducir las entidades a propiedades físicas más deseables la normalización consta de varios pasos, los cuales se denominan normas de normalización o formularios normales. (IBM Corporation, 2021c)

Según (GeeksforGeeks, 2015a) estos son los puntos más importantes con respecto a las formas normales en DBMS:

### **Primera forma normal**

Es el nivel más básico de la normalización, también llamada 1FN, en la tabla cada celda debe contener un valor exacto y cada columna debe tener un nombre que debe ser único. Este primer formulario normal ayuda a eliminar datos iguales y facilita las consultas.

Cumplir con la primera forma normal implica que una tabla de una base de datos satisface ciertas condiciones básicas para garantizar la organización y la integridad de los datos. Estas condiciones son las siguientes:

**Valores Atómicos:** Cada celda en una tabla debe contener un valor atómico, lo que significa que no debe haber valores múltiples o compuestos en una sola celda. Cada valor en una columna debe ser indivisible. Esto evita la redundancia y facilita la gestión de datos.

Por ejemplo, si tienes una tabla de "Pedidos" y una columna llamada "Productos" que contiene una lista de productos separados por comas, esto violaría la primera forma normal, ya que los valores no son atómicos. En cambio, cada producto debe tener su propia fila en la tabla.

Cumplir con la primera forma normal es un paso fundamental en el proceso de normalización de bases de datos, ya que sienta las bases para organizar los datos de manera eficiente y garantizar la integridad y la consistencia de estos

## Segunda forma normal

Conocida también como 2NF. Para que todos los atributos que no sean clave dependan de la clave principal es importante eliminar los datos redundantes. Esto quiere decir que cada columna tiene que estar relacionada de manera directa con la clave primaria y no con las otras columnas.

La segunda forma normal se basa en los principios de la primera forma normal y añade un requisito adicional:

Cumplimiento con la Primera Forma Normal: La tabla debe cumplir con los requisitos de la primera forma normal, lo que significa que cada celda debe contener un valor atómico.

Identificación de Claves Candidatas: Se deben identificar todas las claves candidatas en la tabla. Una clave candidata es un conjunto de uno o más atributos que, en combinación, pueden identificar de manera única una fila en la tabla.

Eliminación de Dependencias Parciales: Cada atributo en la tabla debe depender completamente de la clave primaria, lo que significa que no debe existir ninguna dependencia parcial. En otras palabras, si tienes una tabla con una clave primaria compuesta por varios atributos, cada atributo no clave en la tabla debe depender de la totalidad de la clave primaria, no solo de una parte de ella.

La segunda forma normal se enfoca en eliminar la redundancia de datos y asegurarse de que los atributos estén relacionados de manera adecuada con la clave primaria. Esto permite una mejor organización de los datos y evita problemas de actualización y anomalías en la base de datos.

## Tercera forma normal

Se basa en 2NF al requerir que todos los atributos que no son clave sean independientes. Esto representa que cada columna tiene que estar relacionada directamente con la clave primaria y no con ninguna otra columna de la misma tabla.

Forma canónica de Boyce-Codd (BCNF): BCNF, una forma más precisa de 3FN que avala que cada determinante sea una clave candidata en una tabla. De otra manera, BCNF garantiza que cada atributo (que no sea clave) dependa únicamente de claves candidatas.

La tercera forma normal se basa en los principios de la primera y segunda forma normal y añade requisitos adicionales:

Cumplimiento con la Primera Forma Normal: La tabla debe cumplir con los requisitos de la primera forma normal, lo que significa que cada celda debe contener un valor atómico.

Cumplimiento con la Segunda Forma Normal: La tabla debe cumplir con los requisitos de la segunda forma normal, lo que incluye la eliminación de dependencias parciales.

Eliminación de Dependencias Transitivas: No debe haber dependencias transitivas en la tabla. Esto significa que, si un atributo no clave depende de otro atributo no clave a través de la clave primaria, esas dependencias deben ser eliminadas.

Para entender mejor la tercera forma normal, considera un ejemplo: supongamos que tienes una tabla de "Pedidos" con los siguientes atributos: ID del pedido (clave primaria), ID del cliente, Nombre del cliente y Ciudad del cliente. En este caso, la tercera forma normal requeriría que el atributo "Ciudad del cliente" se almacene en una tabla separada de "Clientes" y que se utilice una clave foránea para relacionar los datos.

### **Cuarta forma normal**

Es un refinamiento más de BCNF para garantizar que las tablas no contengan dependencias de múltiples valores.

La "cuarta forma normal" es un nivel más avanzado de normalización en el diseño de bases de datos relacionales, que se centra en reducir aún más la redundancia de datos y abordar las dependencias multivaluadas. Cumplir con la cuarta forma normal implica que una tabla de una base de datos ha sido

organizada de manera óptima y se han eliminado las dependencias multivaluadas. Para comprenderla, es necesario entender algunos conceptos clave:

**Dependencias Multivaluadas:** Una dependencia multivaluada es una relación en la que un conjunto de atributos (no clave) determina otro conjunto de atributos (no clave), y esto ocurre independientemente de la clave primaria. En otras palabras, si en una tabla tenemos dependencias en las que un conjunto de valores en una columna determina otro conjunto de valores en otra columna, independientemente de la clave primaria, estamos ante una dependencia multivaluada.

**Cumplimiento con las Formas Normales Anteriores:** Antes de aplicar la cuarta forma normal, la tabla debe cumplir con las formas normales anteriores, es decir, la primera, segunda y tercera forma normal.

**Eliminación de Dependencias Multivaluadas:** En la cuarta forma normal, se deben eliminar o descomponer las dependencias multivaluadas en tablas separadas. Cada conjunto de atributos relacionados debe convertirse en su propia tabla y utilizarse una clave foránea para establecer relaciones entre estas tablas.

La aplicación de la cuarta forma normal es especialmente relevante cuando se trabaja con bases de datos que contienen información multivaluada. Un ejemplo típico de dependencia multivaluada podría ser una tabla de "Estudiantes" en la que un estudiante puede estar inscrito en múltiples cursos y cada curso puede tener múltiples libros de texto asociados. Para cumplir con la cuarta forma normal, se descompondría esta estructura en tablas separadas para "Estudiantes", "Cursos" y "Libros de texto", y se utilizarían claves foráneas para establecer relaciones adecuadas.

La cuarta forma normal contribuye a una mayor eficiencia y evita la redundancia de datos en situaciones donde las dependencias multivaluadas pueden dificultar la gestión y la integridad de los datos.

## Quinta forma normal

Es el nivel más valioso de normalización e involucra dividir la tabla en tablas más pequeñas para descartar la redundancia de datos y mejora la integridad de los datos.

## Comandos SQL en el Gestor de base de datos MySQL

### Comandos DDL

También conocido como lenguaje de definición de datos, está constituido por comandos SQL que puede utilizar para definir el esquema de su base de datos. Solo se ocupa de describir el esquema de la base de datos y crear y modificar la estructura de los objetos de la base de datos dentro de la base de datos. DDL es un conjunto de comandos SQL que se utilizan para crear, modificar y eliminar estructuras de bases de datos, pero no para crear datos. Estos comandos no suelen ser utilizados por usuarios generales que necesitan acceder a bases de datos a través de aplicaciones. (GeeksforGeeks, 2017)

Según (Micho García & Jorge Arévalo, 2013) los comandos básicos son los siguientes:

### Create

Este comando en el gestor de base de datos permite crear un objeto internamente. Puede ser una tabla, base de datos, índice, procedimiento almacenado o vista.

Para la creación de una tabla es importante tener en cuenta que existen diferentes tipos de datos, a continuación, se detalla cada tipo.

| Tipo de dato                    | Descripción/ejemplo                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHARACTER                       | Especifica el número exacto de caracteres (que debe ser de un conjunto de caracteres) que se almacenará por cada valor. Por ejemplo, si se define el número de caracteres como 10, pero el valor contiene sólo seis caracteres, los cuatro caracteres restantes serán espacios. El tipo de dato puede abreviarse como CHAR.<br>Ejemplo: NOMBRE_ARTISTA CHAR(60)    |
| CHARACTER VARYING               | Especifica el mayor número de caracteres (que debe ser de un conjunto de caracteres) que se incluyen en un valor. El número de caracteres almacenados es exactamente el mismo número que el valor introducido; por lo tanto, no se agregan espacios al valor. El tipo de dato puede abreviarse como CHAR VARYING o VARCHAR.<br>Ejemplo: NOMBRE_ARTISTA VARCHAR(60) |
| CHARACTER LARGE OBJECT          | Almacena grandes grupos de caracteres, hasta la cantidad especificada. El número de caracteres almacenados es exactamente el mismo número que el valor introducido; por lo tanto, no se agregan espacios al valor. El tipo de dato puede abreviarse como CLOB.<br>Ejemplo: BIO_ARTISTA CLOB(200K)                                                                  |
| NATIONAL CHARACTER              | Funciona igual que el tipo de dato CHARACTER, excepto que se basa en una aplicación definida de un conjunto de caracteres. El tipo de dato puede abreviarse como NATIONAL CHAR y NCHAR.<br>Ejemplo: NOMBRE_ARTISTA NCHAR(60)                                                                                                                                       |
| NATIONAL CHARACTER VARYING      | Funciona igual que el tipo de dato CHARACTER VARYING, excepto que se basa en una aplicación definida de un conjunto de caracteres. El tipo de dato puede abreviarse como NATIONAL CHAR VARYING o NCHAR VARYING.<br>Ejemplo: NOMBRE_ARTISTA NCHAR VARYING(60)                                                                                                       |
| NATIONAL CHARACTER LARGE OBJECT | Funciona igual que el tipo de dato CHARACTER LARGE OBJECT, excepto que se basa en una aplicación definida de un conjunto de caracteres. El tipo de dato puede abreviarse como NCHAR LARGE OBJECT o NCLOB.<br>Ejemplo: BIO_ARTISTA NCLOB(200K)                                                                                                                      |
| BIT                             | Especifica el número exacto de bits que pueden almacenarse para cada carácter. Por ejemplo, si se define el número de bits como 2, pero el valor contiene sólo 1 bit, el bit restante será un espacio. Si el número de bits no se especifica, 1 bit se almacena.<br>Ejemplo: EN_EXISTENCIA BIT                                                                     |

| Tipo de dato        | Descripción/ejemplo                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BIT VARYING         | Especifica el mayor número de bits que pueden incluirse en un valor. El número de bits almacenados es exactamente el mismo número que el valor introducido; por lo tanto, no se agregan espacios al valor.<br>Ejemplo: EN_EXISTENCIA BIT VARYING(2)                                                                                                                                                                                             |
| BINARY LARGE OBJECT | Almacena grandes grupos de bytes hasta la cantidad especificada. El número de bytes almacenados es exactamente el mismo número que el valor introducido; por lo tanto, no se agregan espacios al valor. El tipo de dato también puede remitirse como BLOB.<br>Ejemplo: IMG_ARTISTA BLOB(1M)                                                                                                                                                     |
| XML                 | El lenguaje de marcado extensible (XML) es un lenguaje de marcado para fines generales utilizado para describir documentos en un formato que es conveniente para la visualización de páginas web y para intercambiar datos entre diferentes partes. Las especificaciones para almacenar datos XML en bases de datos SQL se añaden al estándar SQL en SQL:2003 y se abordan en el capítulo 18.<br>Ejemplo: BIO_ARTISTA<br>XML(DOCUMENT(UNTYPED)) |

Tabla 1. Tipos de datos en cadena con ejemplos de definiciones de columna (Oppel Andy & Sheldon Robert, 2010)

También existen los datos de tipo numérico, los cuales se describen a continuación:

| Tipo de dato     | Descripción/ejemplo                                                                                                                                                                                                                                                                                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NUMERIC          | Especifica la precisión y la escala de un valor numérico. Se puede especificar sólo la precisión y utilizar la escala definida (predeterminada) por la aplicación, o se puede especificar la precisión y la escala. Si no se especifica ni la precisión ni la escala, la aplicación proporcionará los valores predeterminados para ambas.<br>Ejemplo: TASA_ARTISTAS NUMERIC (5,2)       |
| DECIMAL          | Especifica valores similares al del tipo de datos NUMERIC. Sin embargo, si la precisión definida por la aplicación es superior a la precisión especificada, los valores con la precisión superior se aceptarán, pero la escala siempre será la que se especifique.<br>Ejemplo: REGALIAS_ARTISTAS DECIMAL (5,2)                                                                          |
| INTEGER          | Especifica un valor con una precisión definida por la aplicación y una escala de 0, lo que significa que sólo se aceptan enteros y no se especifica ningún parámetro con este tipo de datos. El tipo de dato puede abreviarse como INT.<br>Ejemplo: ID_ARTISTA INT                                                                                                                      |
| SMALLINT         | Especifica un valor similar al del tipo de datos INTEGER. Sin embargo, la precisión definida por la aplicación debe ser menor que la precisión de INTEGER.<br>Ejemplo: ID_ARTISTA SMALLINT                                                                                                                                                                                              |
| FLOAT            | Especifica la precisión de un valor numérico, pero no la escala.<br>Ejemplo: REGALIAS_ARTISTAS FLOAT (6)                                                                                                                                                                                                                                                                                |
| REAL             | Especifica un valor con una precisión definida por la aplicación, pero sin una escala. La precisión debe ser menor a la precisión definida por un tipo de datos DOUBLE PRECISION.<br>Ejemplo: REGALIAS_ARTISTAS REAL                                                                                                                                                                    |
| DOUBLE PRECISION | Especifica un valor con una precisión definida por la aplicación, pero sin una escala. La precisión debe ser superior a la precisión definida por el tipo de datos REAL. La implicación es que el valor de la precisión debe ser doble que en el tipo de datos REAL, pero cada implementación define <i>doble</i> de diferentes maneras.<br>Ejemplo: REGALIAS_ARTISTAS DOUBLE PRECISION |

Tabla 2. Tipos de datos numéricos con un ejemplo de definiciones de columna. (Oppel Andy & Sheldon Robert, 2010)

| Tipo de dato             | Descripción/ejemplo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DATE                     | Especifica el año, mes y día del valor de una fecha. El año tiene cuatro dígitos y respalda valores desde 0001 hasta 9999; el mes tiene dos dígitos y respalda valores desde 01 hasta 12, y el día tiene dos dígitos y respalda valores desde 01 hasta 31.<br>Ejemplo: FECHA_CONTRATACION DATE                                                                                                                                                                                                                                                                          |
| TIME                     | Especifica la hora, minuto y segundo del valor de una hora. La hora tiene dos dígitos y respalda valores desde 00 hasta 23; el minuto tiene dos dígitos y respalda valores desde 00 hasta 59, y el segundo tiene, al menos, dos dígitos y respalda valores desde 00 hasta 61.999 (para dar cabida al salto de segundos). El tipo de datos no incluye dígitos fraccionarios a menos que se especifiquen. Por ejemplo, TIME(3) daría tres dígitos fraccionarios. El tipo de dato también se puede denominar como TIME WITHOUT TIME ZONE.<br>Ejemplo: HORA_EVENTO TIME (2) |
| TIME WITH TIME ZONE      | Especifica la misma información que el tipo de datos TIME salvo que el valor también incluye información específica del UTC y las zonas horarias. Los valores agregados al tipo de datos tienen un rango desde -11:59 a +12:00.<br>Ejemplo: HORA_EVENTO TIME WITH TIME ZONE (2)                                                                                                                                                                                                                                                                                         |
| TIMESTAMP                | Combina los valores de TIME y DATE. La única diferencia es que con el tipo de datos TIME, el número predeterminado de dígitos fraccionarios es 0, pero con el tipo de datos TIMESTAMP, el número predeterminado es 6. Se puede especificar un número diferente de dígitos fraccionarios incluyendo un parámetro, tal como TIMESTAMP(4). El tipo de datos se puede también denominar como TIMESTAMP WITHOUT TIME ZONE.<br>Ejemplo: FECHA_COMPRA TIMESTAMP (3)                                                                                                            |
| TIMESTAMP WITH TIME ZONE | Especifica la misma información que el tipo de datos TIMESTAMP, salvo que el valor también incluye información específica del UTC y las zonas horarias. Los valores añadidos al tipo de datos tienen un rango desde -11:59 a +12:00.<br>Ejemplo: FECHA_COMPRA TIMESTAMP WITH TIME ZONE (2)                                                                                                                                                                                                                                                                              |

Tabla 3. Fuente: (Oppel Andy & Sheldon Robert, 2010)

Ejemplo (crear una tabla):

```
# CREATE TABLE Empleado (  
  
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  
Nombres VARCHAR (50), Apellidos VARCHAR (50), Direccion VARCHAR (255),  
Ciudad VARCHAR (60),  
  
Tlf VARCHAR (15),  
  
Edad (2), idCargo INT)
```

## Drop

Este comando permite eliminar un objeto de la base de datos. Puede ser una vista, tabla, índice, función, trigger, procedimiento o cualquier objeto diferente que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Ejemplo:

```
# DROP TABLE 'NOMBRE_TABLA';# DROP SCHEMA 'ESQUEMA';  
  
# DROP DATABASE 'BASEDATOS';
```

## Alter

Este comando modifica la estructura de un objeto. Se puede añadir/quitar campos a una tabla, modificar el tipo de un campo, añadir/quitar índices a una tabla, modificar un trigger, etc.

Ejemplo (agregar columna a una tabla):

```
# ALTER TABLE 'NOMBRE_TABLA' ADD NUEVO_CAMPO INT;  
  
# ALTER TABLE 'NOMBRE_TABLA' DROP COLUMN NOMBRE_COLUMNNA;
```

## Comandos DML

El lenguaje de manipulación de datos (DML) es un lenguaje proporcionado por los sistemas de gestión de bases de datos que permite a los usuarios realizar

tareas de manipulación o consulta de datos organizadas por un modelo de datos adecuado. El lenguaje de manipulación de datos más popular en la actualidad es SQL, que se utiliza para recuperar y manipular datos en bases de datos relacionales. (Micho García & Jorge Arévalo, 2013)

## Insert

La sentencia INSERT de SQL añade uno o más registros a una (únicamente a una) tabla en una base de datos relacional.

Forma básica:

```
# INSERT INTO "tabla" ("columna1", ["columna2,... "]) VALUES ("valor1", ["valor2,... "])
```

La cantidad de columnas y valores tienen que ser iguales. Si una columna no es especificada, entonces se le asignará el valor por omisión, por otro lado, los valores especificados (o implícitos) por la sentencia INSERT tienen que cumplir con todas las restricciones aplicables. En caso de un error de sintaxis o si alguna de las restricciones es vulnerada, no se añade la fila y se muestra un error.

## Update

La sentencia UPDATE de SQL se utiliza para modificar los valores de un conjunto de registros que existen en una tabla.

Ejemplo:

```
# UPDATE mi_tabla SET campo1 = 'nuevo valor campo1' WHERE campo2 = 'N';
```

## Delete

La sentencia DELETE de SQL elimina uno o más registros que existen en una tabla. Forma básica:

```
# DELETE FROM 'tabla' WHERE 'columna1' = 'valor1'
```

## Select

Se utiliza para consultar registros que existen en la base de datos los cuales satisfagan un criterio determinado.

```
# SELECT Campos FROM Tabla;
```

## ACTIVIDADES RECOMENDADAS

Realizar los ejercicios prácticos del Capítulo III del libro Fundamentos de SQL.

# UNIDAD 2

## REGLAS LÓGICAS, INNER JOIN Y TRIGGER



## Reglas lógicas, INNER JOIN y TRIGGER

### Introducción

En el extenso mundo de las bases de datos, coexisten conceptos y herramientas avanzadas que permiten un manejo más eficiente y poderoso de los datos. Tres de estos elementos clave son las reglas lógicas, INNER JOIN y TRIGGER. En esta introducción, exploraremos cada uno de estos conceptos y cómo contribuyen a la gestión y manipulación de datos en bases de datos.

Las reglas lógicas, en el argumento de las bases de datos, son un conjunto de condiciones y acciones que se definen para automatizar ciertas tareas o mantener la consistencia de los datos. Estas reglas permiten establecer restricciones en la base de datos para garantizar que los datos cumplan ciertas condiciones específicas.

Es una operación utilizada en consultas SQL para combinar datos

de dos o más tablas relacionadas en una base de datos. Esto permite recuperar información de múltiples tablas que están conectadas mediante claves primarias y foráneas. INNER JOIN se utiliza para obtener resultados que coinciden en ambas tablas, es decir, solo se devolverán registros que tengan correspondencias en ambas tablas.

Un TRIGGER es una función o forma almacenada en una base de datos que se hace de manera automática como respuesta a ciertos eventos o acciones, como la inserción, actualización o eliminación de datos en una tabla. Los TRIGGERS permiten la automatización de tareas y aplicación de acciones predefinidas cuando se cumplan ciertas condiciones.

## Objetivo

Automatizar acciones en sistemas de bases de datos, para crear consultas eficientes mediante sentencias SQL, con pensamiento crítico y trabajo en equipo.

## Reglas lógicas

El autor Codd anuncia las 12 reglas que debe cumplir todo DBMS para ser apreciado como relacional. Estas reglas en la vida real las cumplen pocos sistemas relacionales y el autor (Sánchez Jorge, 2004) las describe de la siguiente manera:

### Regla 0: Fundación.

Un sistema para la gestión de base de datos debe encargarse de sus datos almacenados al utilizar sus capacidades relacionales.

### Regla 1: Información.

Toda la información de la base de datos debe representarse explícitamente en un esquema lógico. Entonces todos los datos están en la tabla.

### Regla 2: Acceso garantizado.

Puede acceder a los datos conociendo el valor de su clave y el nombre de la columna o atributo que contiene los datos.

### Regla 3: Manejo sistemático de valores NULL.

El DBMS debería poder manejar estos valores de forma adecuada.

### Regla 4: Catálogo online basado en modelo relacional.

Los metadatos deben ser accesibles a través de un esquema relacional.

### Regla 5: Sublenguaje de datos completos.

Como mínimo, necesitamos un lenguaje que permita la gestión completa de la base de datos. Por lo tanto, el lenguaje debería poder realizar cualquier operación.

### Regla 6: Actualizar su opinión.

Un DBMS debe garantizar que las vistas muestren la información más actualizada.

Regla 7: Inserción, modificación y eliminación de datos de nivel.

Las operaciones de modificación deben afectar el conjunto de filas, no registro por registro.

### **Regla 8: Independencia física.**

La accesibilidad a los datos es importante desde la lógica de la base de datos incluso si cambia el almacenamiento.

### **Regla 9: Independencia**

Lógica el programa no debe verse afectado por cambios en la tabla

### **Regla 10: Independencia de la integridad.**

Las restricciones deben ser almacenadas en la base de datos (diccionario de datos), no en el programa de aplicación.

### **Regla 11: Independencia de Distribución.**

El sublenguaje de datos debe permitir que sus declaraciones funcionen de la misma manera en bases de datos distribuidas que en bases de datos no distribuidas.

### **Regla 12: No Vandalismo.**

Si su DBMS tiene un lenguaje que permite el recorrido de registros, no puede usar ese lenguaje para violar las reglas relacionales.

## **Inner Join**

INNER JOIN es una de las condiciones más comunes perteneciente a JOIN, es utilizada para unir tablas y está especificada al usar las palabras clave INNER JOIN, sin embargo, la palabra INNER no es necesaria ya que al utilizar JOIN se asume que es una INNER JOIN. (Oppel Andy & Sheldon Robert, 2010)

INNER JOIN se diferencia de todas las demás declaraciones JOIN en que muestra un conjunto de resultados mínimo, ya que en el producto cruzado solo aparecen los registros que satisfacen los criterios de selección. Todos estos se muestran en una tabla de resultados llamada "Ver" que no contiene valores NULL. (IONOS Inc., 2019)

## INNER JOIN en la práctica

Iniciamos la explicación de INNER JOIN con un ejemplo, partiendo de dos tablas. Para este ejemplo utilizamos dos tablas, “Empleados” y “Departamentos”. En la tabla “Empleados” se tiene todos los empleados de una empresa, sus números de identificación (e\_id) y el número de departamento al que pertenecen (d\_id).

Tabla 4: Empleados

| e_id | Apellido        | Nombre    | d_id |
|------|-----------------|-----------|------|
| 1    | García Hurtado  | Macarena  | 3    |
| 2    | Oña Martínez    | Francisco | 1    |
| 3    | Guzmán Durán    | Elena     | 1    |
| 4    | Hernández Soria | Manuela   | 2    |
| 5    | Olivia Cansino  | Andrea    | NULL |

Fuente: (IONOS Inc., 2019)

En esta tabla tenemos un caso especial, los empleados Oña Martínez y Guzmán Durán trabajan en el mismo departamento. A la empleada Olivia Cansino todavía no se le ha asignado un departamento (NULL).

La tabla “Departamentos” cuenta todos los departamentos de la empresa, conteniendo el número identificador de cada departamento y su ubicación.

Tabla 5: Departamentos

| d_id | Denominación    | Localización |
|------|-----------------|--------------|
| 1    | García Hurtado  | Macarena     |
| 2    | Oña Martínez    | Francisco    |
| 3    | Guzmán Durán    | Elena        |
| 4    | Hernández Soria | Manuela      |
| 5    | Olivia Cansino  | Andrea       |

Fuente: (IONOS Inc., 2019)

Las dos tablas están vinculadas por una relación de una clave foránea. El ID de departamento, que en este caso actúa como clave primaria en la tabla “Departamentos”, ha sido integrado como clave foránea en la tabla “Empleados”. Esta conexión permite generar un INNER JOIN entre las dos tablas, que puede ayudar a establecer, por ejemplo, la ubicación del puesto de trabajo de un empleado. Al consultar una base de datos relacional, por lo general se define como condición de selección la correspondencia entre una clave primaria y una clave foránea. La condición se cumple si la clave foránea que ha sido seleccionada en una tabla coincide con la clave primaria de la otra tabla (=), entonces, solo se exponen los registros de datos que contienen valores en común. (IONOS Inc., 2019)

Un ejemplo de la sentencia SQL utilizando INNER JOIN sería la siguiente:

```
SELECT * FROM empleados INNER JOIN departamentos ON empleados.  
e_id = departamentos.d_id;
```

El comando SELECT pide al SGBD que recupere ciertos datos de la base de datos. Como opción, SQL brinda la posibilidad de introducir datos (INSERT INTO), modificarlos (UPDATE) o eliminarlos (DELETE FROM). El comando SELECT va seguido de los datos que se quieren recuperar. En este caso queremos recuperar el conjunto de datos completo, entonces elegimos un marcador de posición: el asterisco (\*). El comando SELECT siempre demanda la palabra clave FROM y la descripción de la tabla o grupo de tablas (JOIN) de las que queremos recuperar los datos. Por ahora, la fuente de datos es un INNER JOIN en las tablas “Departamentos” y “Empleados”. También utilizamos la palabra clave ON para especificar una condición para el enlace. Sugerimos enlazar los registros de datos y editarlos como una tabla de resultados, donde el d\_id de la tabla de “Empleados” se corresponde con el d\_id de la tabla “Departamentos”. (IONOS Inc., 2019)

El INNER JOIN combina las tablas con la condición empleados. d\_id = departamentos.d\_id muestra la siguiente tabla de resultados.

Tabla 6: SQL INNER JOIN de “Empleados” y “Departamentos”

| e_id | Apellido        | Nombre    | Empleados.d_id | departamentos.d_id | Denominación     | Localización |
|------|-----------------|-----------|----------------|--------------------|------------------|--------------|
| 1    | García Hurtado  | Macarena  | 3              | 3                  | Recursos Humanos | Marbella     |
| 2    | Oña Martínez    | Francisco | 1              | 1                  | Ventas           | Sevilla      |
| 3    | Guzmán Durán    | Elena     | 1              | 1                  | Ventas           | Sevilla      |
| 4    | Hernández Soria | Manuela   | 2              | 2                  | IT               | Málaga       |

Fuente: (IONOS Inc., 2019)

Al realizar la comparación, es indudable de que falta un registro de datos de cada tabla. Se trata de los registros cuyo valor en la columna d\_id no tiene equivalente en la otra tabla.

(5, Olivia Cansino, Andrea, NULL) y (4, Investigación, Málaga)

Olivia Cansino aún no ha sido asignada ningún departamento todavía y el departamento de investigación tampoco cuenta aún con personal, entonces estos conjuntos de datos están ocultos en INNER JOIN, ya que su función es identificar a cada empleado con su departamento.

### Trigger o disparador

Un disparador SQL es un objeto de base de datos agrupado con una tabla que ejecuta de manera automática un conjunto de declaraciones SQL cuando sucede un determinado acontecimiento en esa tabla. Los activadores son utilizados para las reglas comerciales se cumplan, mantener la integridad de los datos y automatizar algunas operaciones de bases de datos. Pueden liberar varios eventos, como añadir, actualizar o eliminar datos en una tabla, y permitirle realizar operaciones adicionales apoyadas en esos eventos. (Bilal Hungund, 2018)

Según el autor (Bilal Hungund, 2018) las diferencias clave entre desencadenadores y procedimientos almacenados son las siguientes:

Los desencadenadores no pueden invocar ni ejecutar manualmente.

No hay posibilidad de que los activadores reciban parámetros.

Una transacción no puede ser confirmada ni revertida dentro de un desencadenador.

Según (Oppel Andy & Sheldon Robert, 2010) la sintaxis básica para la creación de undisparador es la siguiente:

```
CREATE TRIGGER (nombre_disparador)[BEFORE / AFTER]
{INSERT / UPDATE / DELETE} (lista de la columna) ON [NOMBRE_TABLA]
[FOR EACH {ROW / STATEMENT}]
[WHEN] (condición de la búsqueda)
```

Explicación de la sintaxis:

*CREATE TRIGGER (nombre\_disparador): crea o reemplaza un disparador existente con el nombre\_disparador.*

*[BEFORE / AFTER]: Esto especifica cuándo se ejecutará el disparador.*

*{INSERT / UPDATE / DELETE}: esto especifica la operación DML.*

*En [ON]: esto especifica el nombre de la tabla asociada con el activador.*

*[FOR EACH]: especifica un disparador a nivel de fila, quiere decir que, el disparador se ejecutará para cada fila que ha sido afectada.*

*[WHEN]: permite proporcionar la operación que se va a realizar cuando se activa el disparador.*

En caso de querer eliminar un disparador la instrucción es muy sencilla:

```
DROP TRIGGER <nombre_disparador>
```

(Bilal Hungund, 2018) explica que los beneficios de utilizar desencadenadores SQL incluyen:

Integridad de los datos: los activadores permiten la aplicación de restricciones y reglas comerciales complejas a nivel de la base de datos, esto garantiza que los datos sean consistentes y precisos.

**Automatización:** los activadores pueden automatizar tareas que se repiten o son complejas estableciendo acciones predefinidas cada vez que ocurre un evento determinado. Esto permite reducir la necesidad de intervenir manualmente y mejorar la eficiencia.

**Seguimientos de auditoría:** los activadores se utilizan para averiguar los cambios realizados en los datos, como reconocer modificaciones en una tabla de auditoría separada. Esto ayuda a verificar y mantener un historial de cambios de datos.

**Validación de datos:** los activadores pueden comprobar la validación adicional de los datos antes de ser insertados, actualizarlos o eliminarlos, certificando que solo se acumulen en la BD datos válidos y conformes.

Ejemplo:

Para este ejemplo se utilizará una base de datos en la cual se registran las calificaciones de los estudiantes, aquí se crea un activador para que el total y el porcentaje de marcas específicas sean insertadas de manera automática cada vez que se añada un registro, hoy aquí se invoca a un disparador antes de insertar el registro para que se pueda usar la etiqueta BEFORE.

Supongamos el esquema de la base de datos.

### Consulta

```
mysql>>desc Estudiante;
```

Figura 4. Base de datos de informes de estudiantes

| Field | Type        | Null | Key | Default | Extra          |
|-------|-------------|------|-----|---------|----------------|
| tid   | int(4)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30) | YES  |     | NULL    |                |
| subj1 | int(2)      | YES  |     | NULL    |                |
| subj2 | int(2)      | YES  |     | NULL    |                |
| subj3 | int(2)      | YES  |     | NULL    |                |
| total | int(3)      | YES  |     | NULL    |                |
| per   | int(3)      | YES  |     | NULL    |                |

Fuente: (Bilal Hungund, 2018)

### Activador SQL para el enunciado del problema.

Figura 5. Activador SQL

```
CREATE TRIGGER stud_marks
BEFORE INSERT ON Student
FOR EACH ROW
SET NEW.total = NEW.subj1 + NEW.subj2 + NEW.subj3,
    NEW.per = (NEW.subj1 + NEW.subj2 + NEW.subj3) * 60 / 100;
```

Fuente: (Bilal Hungund, 2018)

La declaración SQL vista anteriormente va a crear un activador en la base de datos de los estudiantes en la que cada vez que se ingrese las calificaciones de las materias, antes de ingresar estos datos el activador se encarga de calcular los 2 valores y los insertará con los valores ingresados.

### Producción

Figura 6. Producción TRIGGER

| Trigger    | Event  | Table   | Timing        | Created             | Status |
|------------|--------|---------|---------------|---------------------|--------|
| stud_marks | BEFORE | Student | BEFORE INSERT | 2023-05-02 00:00:00 | ACTIVE |

Fuente: (Bilal Hungund, 2018)

De este modo se logra crear y ejecutar disparadores en las bases de datos.

## ACTIVIDADES RECOMENDADAS

Realice ejercicios prácticos de INNER JOIN y TRIGGER del libro Fundamentos de SQL.

# UNIDAD 3

## VISTAS Y PROCEDIMIENTOS ALMACENADOS



## Vistas y procedimientos almacenados

### Introducción

Las vistas y los procedimientos almacenados son una parte importante de la gestión de bases de datos para mejorar la eficiencia y la organización de los datos y simplificar el acceso y la manipulación de la información almacenada. En esta introducción, veremos qué son las vistas y los procedimientos almacenados y su importancia en la administración de bases de datos.

**Vista de base de datos:** una vista es una representación virtual de una tabla de base de datos o un conjunto de tablas. A diferencia de las tablas físicas, las vistas no almacenan datos de forma independiente; en cambio, proporcionan una vista específica y personalizada de los datos almacenados en una o más tablas. Las vistas permiten a los usuarios acceder a partes relevantes de los datos sin exponer las estructuras de la base de datos subyacentes o los detalles de implementación. Controle el acceso con permisos

específicos, simplifique las consultas de datos y garantice la seguridad.

**Procedimiento almacenado de base de datos:** un procedimiento almacenado es un conjunto de declaraciones SQL predefinidas que se almacenan en la base de datos y se ejecutan repetidamente cuando se llama al procedimiento. Estos procedimientos pueden aceptar parámetros, realizar operaciones complejas, devolver resultados o modificar datos en la base de datos. Los procedimientos almacenados ayudan a encapsular la lógica de la aplicación en la base de datos, facilitan la reutilización del código, mejoran la seguridad y el rendimiento y simplifican la gestión de transacciones y concurrencia.

En resumen, las vistas y los procedimientos almacenados desempeñan un papel central en la gestión de bases de datos, proporcionando mayor flexibilidad, seguridad y eficiencia en el acceso

y manipulación de datos. Estas herramientas son esenciales para que los desarrolladores y administradores de bases de datos creen y mantengan sistemas de información sólidos y eficientes.

### **Objetivo**

Acceder a los datos y simplificar consultas recurrentes y establecer vistas y procedimientos almacenados, mediante un pensamiento crítico y proactividad.

## Vistas

Una vista de base de datos es un subconjunto de una BD apoyado en una consulta que se ejecuta en una o más tablas. Las vistas de BD se almacenan como consultas de base de datos con nombre y se pueden utilizar para almacenar consultas completas que han sido usadas con frecuencia.

Existen dos tipos de vistas:

Vistas dinámicas y estáticas. Las vistas dinámicas generalmente pueden contener datos de una o dos tablas e incluirlos de manera automática todas las columnas de la tabla o tablas especificadas. Las vistas dinámicas se pueden actualizar dinámicamente cuando se crean o modifican objetos relacionados u objetos extendidos. Las vistas dinámicas pueden actualizarse de forma manual cuando se crean o modifican objetos relacionados u objetos extendidos.

Las vistas estáticas pueden estar formadas por datos de varias tablas y las columnas pedidas para esas tablas deben especificarse en las cláusulas SELECT y WHERE de la vista estática. Las vistas dinámicas pueden actualizarse de forma manual cuando se crean o modifican objetos relacionados u objetos extendidos. (IMB Corporation, 2022)

Las vistas en bases de datos proporcionan varios beneficios significativos, lo que las convierte en una herramienta valiosa en el diseño y la gestión de bases de datos. Algunos de los beneficios clave de las vistas incluyen:

**Simplifican el acceso a datos:** Las vistas permiten a los usuarios y aplicaciones acceder a una porción específica de los datos sin necesidad de conocer la estructura subyacente de las tablas. Esto facilita el acceso a datos de manera más sencilla y reduce la complejidad.

**Seguridad:** Las vistas pueden utilizarse para aplicar restricciones de seguridad. Los administradores de bases de datos pueden definir vistas que muestren solo los datos necesarios para un usuario o grupo de usuarios, ocultando datos confidenciales o no autorizados.

**Abstracción de datos:** Las vistas proporcionan una capa de abstracción que oculta los detalles internos de la estructura de la base de datos. Esto facilita la gestión de cambios en la base de datos sin afectar a las aplicaciones que acceden a través de las vistas.

**Simplifican la estructura de datos:** Las vistas pueden utilizarse para combinar datos de múltiples tablas en una vista lógica. Esto simplifica las consultas y operaciones al ofrecer una vista más coherente de los datos.

**Rendimiento:** Las vistas materializadas, que almacenan los resultados de una vista en lugar de calcularlos en tiempo real, pueden mejorar el rendimiento de las consultas al minimizar el costo de los cálculos repetidos.

**Consistencia de datos:** Al proporcionar vistas específicas y predefinidas, se garantiza que los usuarios vean una representación coherente y consistente de los datos, lo que ayuda a evitar inconsistencias en las consultas.

**Reutilización de consultas:** Las vistas permiten definir consultas complejas una vez y reutilizarlas en múltiples ubicaciones y aplicaciones, lo que ahorra tiempo de desarrollo y evita errores en las consultas repetidas.

**Simplifican la administración de permisos:** La gestión de permisos en las vistas puede ser más granular y fácil de administrar que en las tablas subyacentes, lo que garantiza que los usuarios tengan acceso solo a los datos necesarios.

**Seguimiento de auditoría:** Las vistas pueden utilizarse para realizar un seguimiento de auditoría y generar registros de actividades en la base de datos. Esto facilita el control y la supervisión de quién accede y modifica los datos.

Una vista se considera como una tabla virtual y actúa como otra tabla cualquiera de la BD al consultar datos. Con el comando SELECT, se puede usar cláusulas WHERE para consultar la vista completa o una parte específica:

*CREATE VIEW (nombre de la vista) [(nombres de las columnas de la vista)] AS (expresión de consulta)*

[ WITH CHECK OPTION]

Explicación de la sintaxis:

*CREATE VIEW:* proporciona un nombre para la vista que vamos a crear, adicional se debe proporcionar los nombres para las columnas.

*AS:* es requerida, y el marcador de posición.

*WITH CHECK OPTION:* se aplica a vistas actualizables que incluyen la cláusula *WHERE* en la instrucción *SELECT*.

Ejemplo

Para el siguiente ejemplo se toma en cuenta las tablas que están en la figura 10.

Tabla 7. Inventario

| INVENTARIO                |                               |                   |                                  |                       |
|---------------------------|-------------------------------|-------------------|----------------------------------|-----------------------|
| ID_DISCO_COMPACTO:<br>INT | TITULO_CD:<br>VARCHAR(60)     | COPYRIGHT:<br>INT | PRECIO_MENUDEO:<br>NUMÉRICO(5,2) | EN_EXISTENCIA:<br>INT |
| 99301                     | Famous Blue Raincoat          | 1991              | 16.99                            | 6                     |
| 99302                     | Blue                          | 1971              | 14.99                            | 26                    |
| 99303                     | Court and Spark               | 1974              | 14.99                            | 18                    |
| 99304                     | Past Light                    | 1983              | 15.99                            | 2                     |
| 99305                     | Kojiki                        | 1990              | 15.99                            | 5                     |
| 99306                     | That Christmas Feeling        | 1993              | 10.99                            | 3                     |
| 99307                     | Patsy Cline: 12 Greatest Hits | 1988              | 16.99                            | 25                    |

Fuente: (Oppel Andy & Sheldon Robert, 2010)

Escribimos la instrucción:

*CREATE VIEW DESCUENTOS\_CD*

*(DISCO\_COMPACTO, PRECIO\_MENUDEO, PRECIO\_DESCUENTO) AS  
SELECT TITULO\_CD, PRECIO\_MENUDEO, PRECIO\_MENUDEO \* 0.9  
FROM INVENTARIO;*

Y obtenemos la vista creada

Tabla 8. Vista DESCUENTOS\_CD

DESCUENTOS\_CD      ↓                      ↓                      ↓

| DISCO_COMPACTO                | PRECIO_MENUDEO | PRECIO_DESCUENTO |
|-------------------------------|----------------|------------------|
| Famous Blue Raincoat          | 16.99          | 15.29            |
| Blue                          | 14.99          | 13.49            |
| Court and Spark               | 14.99          | 13.49            |
| Past Light                    | 15.99          | 14.39            |
| Kojiki                        | 15.99          | 14.39            |
| That Christmas Feeling        | 10.99          | 9.89             |
| Patsy Cline: 12 Greatest Hits | 16.99          | 15.29            |

Fuente: (Oppel Andy & Sheldon Robert, 2010)

### Procedimientos almacenados

Los procedimientos almacenados son considerados como un conjunto de una o varias instrucciones SQL almacenadas, similares a la forma en cómo una vista almacena una instrucción SELECT. Estos procedimientos se invocan desde las instrucciones SQL, más no desde un lenguaje de programación, también puede ser escrito en SQL o en cualquier otro lenguaje de programación, para invocarlo es necesario utilizar la instrucción CALL y por último soporta parámetros de entrada y salida, aunque ninguno de ellos sea obligatorio (Oppel Andy & Sheldon Robert, 2010)

Los procedimientos almacenados ofrecen varios beneficios en el contexto de las bases de datos y las aplicaciones que interactúan con ellas. A continuación, se destacan algunas de las ventajas y características de los procedimientos almacenados:

**Reutilización de código:** Los procedimientos almacenados permiten encapsular lógica de programación y consultas SQL en un solo lugar. Esto facilita la reutilización del código en múltiples partes de una aplicación o en diferentes aplicaciones.

**Mejora del rendimiento:** Los procedimientos almacenados a menudo se almacenan en caché en la base de datos, lo que puede mejorar el rendimiento

de las consultas frecuentes, ya que la ejecución de un procedimiento almacenado es más eficiente que el envío de múltiples instrucciones SQL individuales.

**Seguridad:** Los procedimientos almacenados pueden aplicar restricciones de seguridad a nivel de la base de datos, lo que garantiza que los usuarios solo tengan acceso a los datos y la funcionalidad que están autorizados a utilizar.

**Transacciones y control de concurrencia:** Los procedimientos almacenados pueden incluir transacciones, lo que permite asegurar la integridad de los datos y controlar la concurrencia en entornos de bases de datos multiusuario.

**Mantenimiento simplificado:** Los cambios en la lógica de la aplicación o las consultas SQL se pueden realizar en un solo lugar (el procedimiento almacenado) en lugar de actualizar múltiples ubicaciones en el código de la aplicación.

**Abstracción de datos:** Los procedimientos almacenados pueden ocultar detalles de la estructura de la base de datos, proporcionando una capa de abstracción para que las aplicaciones accedan a los datos de manera más sencilla.

**Documentación incorporada:** Los procedimientos almacenados pueden incluir comentarios y documentación dentro del código, lo que facilita la comprensión de su funcionalidad y uso.

**Programación modular:** Los procedimientos almacenados permiten dividir una aplicación en componentes más pequeños y manejables, lo que facilita el desarrollo y la depuración.

**Planificación y programación:** Los procedimientos almacenados se pueden programar para ejecutarse en momentos específicos o en respuesta a eventos, lo que es útil para tareas de mantenimiento o informes periódicos.

Sintaxis para crear un procedimiento:

*CREATE PROCEDURE (nombre del procedimiento)*

*((declaración de parámetro [ {,} declaración de par+ametrp} ...])) [ (características de la rutina)*

*(cuerpo de la rutina)*

Como se puede ver, debe proporcionarse un nombre para el procedimiento (en la cláusula CREATE PROCEDURE) seguido por cero o más declaraciones de parámetro, que van encerradas en paréntesis. Se deben proporcionar los paréntesis incluso si no se define ninguna declaración. Después de las declaraciones de parámetro se tiene la opción de definir una o más características de la rutina. (Oppel Andy & Sheldon Robert, 2010)

Una vez creado el procedimiento, se invoca utilizando la instrucción CALL, la instrucción básica es la siguiente:

*CALL (nombre del procedimiento) ([ (valor) [ {, (valor)} ...]])*

En la cláusula CALL se identifica el nombre del procedimiento y a continuación seguir con los valores que pasan por el procedimiento como parámetros.

Ejemplo:

Tabla 9. INVENTARIO\_CD y TIPOS\_CD

| INVENTARIO_CD             |                        |                       | TIPOS_CD               |                             |
|---------------------------|------------------------|-----------------------|------------------------|-----------------------------|
| TITULO_CD:<br>VARCHAR(60) | ID_TIPO_CD:<br>CHAR(4) | EXISTENCIA_CD:<br>INT | ID_TIPO_CD:<br>CHAR(4) | NOMBRE_TIPO_CD:<br>CHAR(20) |
| Famous Blue Raincoat      | FROK                   | 19                    | FROK                   | Folk Rock                   |
| Blue                      | CPOP                   | 28                    | CPOP                   | Classic Pop                 |
| Past Light                | NEWA                   | 6                     | NEWA                   | New Age                     |
| Out of Africa             | STRK                   | 8                     | CTRY                   | Country                     |
| Fundamental               | NPOP                   | 10                    | STRK                   | Soundtrack                  |
| Blues on the Bayou        | BLUS                   | 11                    | BLUS                   | Blues                       |
| Kojiki                    | NEWA                   | 10                    | JAZZ                   | Jazz                        |

Fuente: (Oppel Andy & Sheldon Robert, 2010)

Instrucción para la creación de un procedimiento:

```
CREATE PROCEDURE NEW_AGE_CDS ()
SELECT TITULO_CD, EXISTENCIA_CD
FROM INVENTARIO_CD i, TIPOS_CD t
WHERE i.ID_TIPO_CD = t.ID_TIPO_CD
AND NOMBRE_TIPO_CD = 'New Age';
```

Instrucción CALL:

```
CALL CDS_NEW_AGE ();
```

Una vez que se ejecuta la instrucción se genera el siguiente procedimiento:

Figura 7. Procedimiento almacenado

| TITULO_CD  | EXISTENCIA_CD |
|------------|---------------|
| Past Light | 6             |
| Kojiki     | 10            |

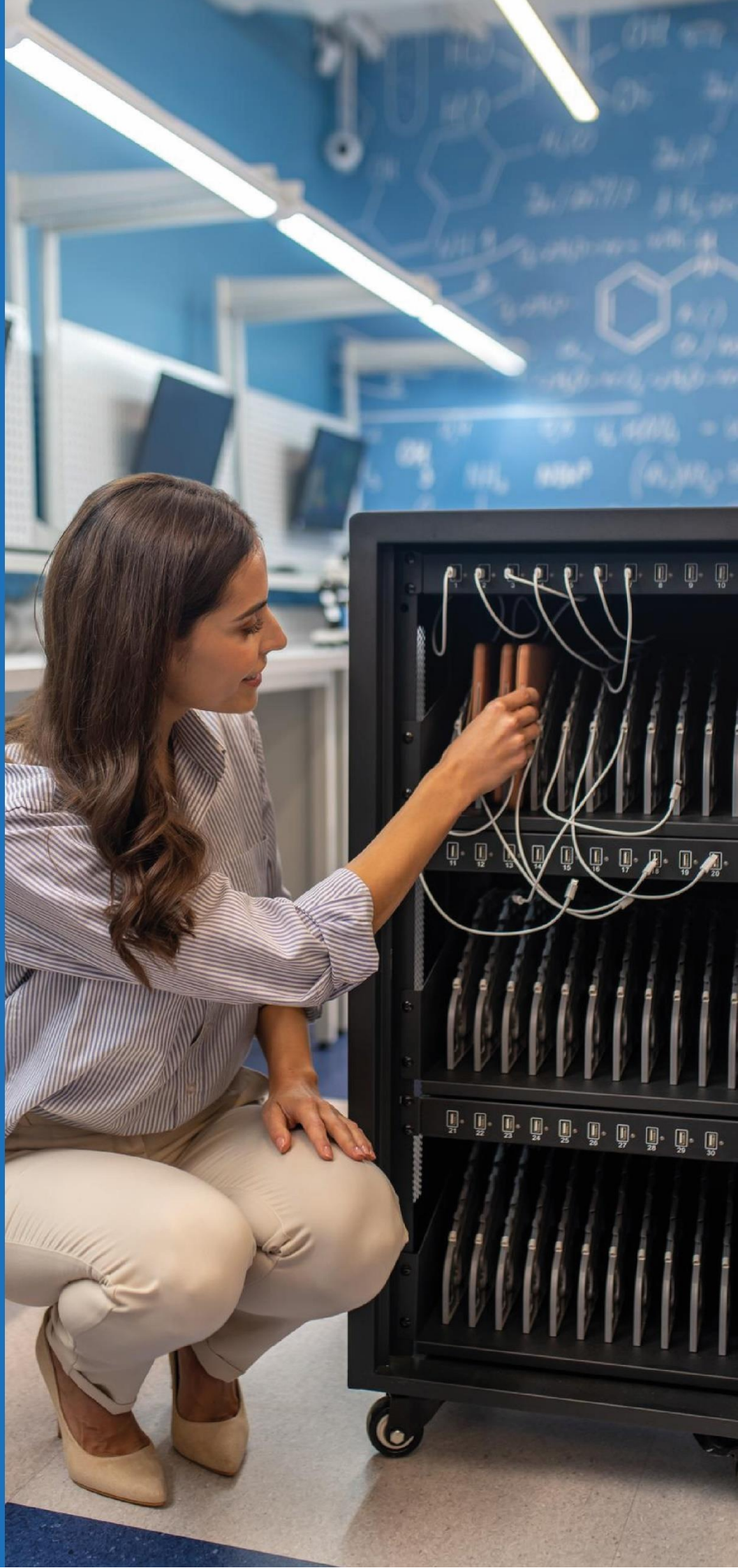
Fuente: (Oppel Andy & Sheldon Robert, 2010)

## ACTIVIDADES RECOMENDADAS

Realice ejercicios prácticos planteados en el libro Fundamentos de SQL sobre vistas y procedimientos almacenados.

# UNIDAD 4

## ADMINISTRACIÓN, BACKUP Y REINGENIERÍA



## Administración: Backup y reingeniería

### Introducción

La administración de bases de datos es una parte integral de la gestión de la información de cualquier organización. Incluye una serie de procesos y técnicas que garantizan la integridad, disponibilidad y seguridad de los datos almacenados en la base de datos. Dos conceptos clave en este contexto son la copia de seguridad y la reingeniería de la base de datos.

El Backup o copia de seguridad es una práctica importante de gestión de bases de datos. Consiste en realizar copias de seguridad de los datos almacenados en la base de datos en un momento específico.

Estas copias se utilizan para garantizar la recuperación de datos en caso de fallas, errores o pérdida de datos debido a diversos factores como errores humanos, problemas de hardware o software, ciberataques o desastres naturales.

La reingeniería de bases de datos es la revisión y modificación de la estructura y diseño de una base de datos existente para mejorar su eficiencia, eficacia, escalabilidad o funcionalidad. Esto puede ser necesario si la base de datos está obsoleta, no cumple con los requisitos actuales de su organización o tiene problemas de rendimiento.

### Objetivo

Optimizar su estructura y rendimiento, para ejecutar copias de seguridad y restauraciones de Backup, y procesos de reingeniería de bases de datos para, aplicando valores y principios personales.

## Administración

La administración de la base de datos está a cargo del administrador de la base de datos, quién es la persona que no sólo es responsable de crear la base de datos y realizar comprobaciones técnicas para garantizar que cumple con los requisitos establecidos, sino que también es quien garantiza que los sistemas funcionan correctamente. Cabe señalar que cuando hablamos de gestión de bases de datos, no se trata solo de un DBA, sino también de un grupo de programadores de sistemas u otros técnicos, es decir, un grupo de asistentes técnicos que se encargan de todo el funcionamiento de la base de datos las 24 horas del día, los 7 días de la semana, es decir el 100% del tiempo.

Según (Reinosa et al., 2012) las funciones principales de un DBA son:

**Diseño de Bases de Datos:** Los DBAs participan en el diseño inicial de las bases de datos, determinando la estructura, las tablas, las relaciones y los índices necesarios para satisfacer los requisitos del sistema.

**Creación y Mantenimiento de Bases de Datos:** Se encargan de crear, configurar y mantener bases de datos, lo que implica la creación de tablas, definición de esquemas, ajuste de parámetros de rendimiento y realización de copias de seguridad.

**Gestión de Seguridad:** Los DBAs administran la seguridad de la base de datos, estableciendo políticas de acceso y permisos para garantizar que solo las personas autorizadas tengan acceso a la información y los datos confidenciales estén protegidos.

**Optimización de Rendimiento:** Monitorizan y optimizan el rendimiento de las bases de datos para garantizar una respuesta rápida a las consultas y transacciones, lo que implica ajustar índices, configuraciones y resolver problemas de cuellos de botella.

**Realización de Copias de Seguridad y Recuperación:** Planifican y ejecutan estrategias de copia de seguridad y recuperación para garantizar la disponibilidad y la integridad de los datos en caso de fallos o desastres.

**Migración y Actualización:** Coordinan la migración de datos y aplicaciones a nuevas versiones de sistemas de gestión de bases de datos y realizan actualizaciones de software y hardware.

**Gestión de Espacio en Disco:** Supervisan y administran el uso eficiente del espacio en disco, lo que implica la asignación y liberación de espacio según sea necesario.

**Monitorización y Resolución de Problemas:** Utilizan herramientas de monitorización para seguir de cerca el rendimiento de la base de datos, identificar problemas y tomar medidas correctivas.

**Gestión de Transacciones:** Aseguran la integridad de las transacciones y gestionan la consistencia de los datos en sistemas que requieren transacciones complejas.

**Planificación de Capacidad:** Evalúan las necesidades futuras de almacenamiento y rendimiento de la base de datos y planifican la capacidad en consecuencia.

**Documentación:** Mantienen documentación detallada sobre la estructura de la base de datos, procedimientos y políticas de seguridad para uso interno y referencia.

**Apoyo a Desarrolladores y Usuarios:** Colaboran con desarrolladores de aplicaciones y usuarios finales para proporcionar orientación, resolver problemas y garantizar que las aplicaciones se integren adecuadamente con la base de datos.

**Cumplimiento Normativo:** Aseguran que las bases de datos cumplan con regulaciones y estándares de seguridad de datos, como GDPR o HIPAA, cuando sea aplicable.

**Actualización de Parches y Actualizaciones de Seguridad:** Aplican parches y actualizaciones de seguridad para proteger la base de datos contra vulnerabilidades conocidas.

Evaluación de Nuevas Tecnologías: Mantenerse al día con las tendencias y las nuevas tecnologías en el campo de la administración de bases de datos para evaluar su aplicabilidad en la organización.

### Creación de usuarios

Al iniciar un proceso para lograr acceder a una base de datos, a creación de usuarios es fundamental. Se realiza con el siguiente comando:

```
CREATE USER (nombre) IDENTIFIED BY (contraseña)
```

Nota: normalmente es completado con algunas cláusulas de almacenamiento y opciones de uso de la base de datos.

En caso de querer modificar la contraseña del usuario se ejecuta la siguiente sentencia:

```
ALTER USER (nombre) IDENTIFIED (contraseña)
```

### Asignación de permisos

Los permisos en el sistema permiten que el usuario que los recibe pueda crear, modificar y eliminar objetos de la base de datos, los cuales almacenan los datos de las aplicaciones. Para que el usuario realice operaciones en la base de datos necesita un permiso de sistema denominado CREATE SESSION.

Dentro del alcance de los permisos de sistema existen dos categorías:

El conjunto de permisos de sistema que se relacionen con el manejo de los objetos (tablas, índices, disparadores, secuencias, vistas, paquete, procedimientos almacenados), por medio de los cuales pueden ejecutar las siguientes acciones: crear, modificar su definición y eliminarlos. Para las tablas y programas los permisos son los siguientes: crear índices, hacer referencia a la clave primaria con una foránea y realizar un programa almacenado.

Los permisos que se refieren a la habilidad que tiene un usuario para ejecutar acciones especiales a lo largo del sistema. Estas acciones incluyen funciones como: generación de estadística para tolerar el funcionamiento del

optimizador establecido en costos, actividades de auditoría y permitir el acceso a la base de datos únicamente a los usuarios con un permiso de sistemas especial denominado “sesión restringida”. Estos permisos son asignados generalmente a usuarios que ejecutan tareas de administración de nivel alto.

En caso de que un usuario necesite crear objetos deberá recibir el permiso con la palabra ANY como se muestra en el siguiente ejemplo:

*GRANT CREATE ANY TABLE TO myuser*

Si se requiere anular un permiso otorgado esto se realiza con la sentencia REVOKE.

Existe una forma de capas intermedias llamados ROLES, con los que se distribuyen los permisos establecidos, esto facilita la administración de la base de datos, ya que con determinar el tipo de usuario que corresponde o la tarea que se le asigne se habilitará encontrar uno o dos ROLES. Así el usuario recibirá los permisos necesarios. A continuación, se ilustra la secuencia de tareas:

*CREATE ROLE developer*

*GRANT CREATE TABLE TO developer GRANT SELECT ANY TABLE TO developer GRANT DROP USER TO developer GRANT developer TO Luis*

*GRANT developer TO Juan*

### **Modificación de permisos**

Para la modificación de permisos de los roles y que estos soporten el requerimiento de contraseña se usa la siguiente sentencia:

*ALTER ROLE developer IDENTIFIED BY 12345*

Una vez que se ha ejecutado este comando el rol contará con una contraseña.

En caso de querer eliminar un rol se ejecuta la sentencia DROP ROLE. Para esta acción es necesario que el usuario tenga permisos como: CREATE ANY ROLE, ALTER ANY ROL Y DROP ANY ROL, o que sea el dueño del rol.

### **Protección de base de datos**

La protección de la base de datos es un tema de mucha importancia. En casi todas las organizaciones, es un conjunto de funciones que son manejadas por el DBA (Administrador de la base de datos) o también llamado administrador de seguridad. La persona que cumple este rol es el encargado de crear nuevos usuarios y brindar accesibilidad de los objetos de la base de datos. Como norma general mientras más grande sea la organización, la información es más sensible y existe mayor probabilidad de que la seguridad sea administrada por un Administrador de seguridad, pero es importante que los desarrolladores, los usuarios y los DBA entiendan todas las opciones que existen en el modelo de seguridad.

Se puede decir que la base de datos más segura es la que no tiene usuarios, pero esto sería ilógico. Por este mismo motivo lo que se debe lograr es que exista un balance entre el permiso de acceso a los usuarios y control al momento de realizar alguna actividad que requiera establecer un inicio de sesión a través de una conexión.

Según (Reinosa et al., 2012) el modelo de seguridad consta de varios elementos. Aquí se presentan algunos de los elementos fundamentales en los que se basa el modelo de seguridad en bases de datos:

**Autenticación y Autorización:** La autenticación verifica la identidad de los usuarios antes de permitirles acceder a la base de datos. La autorización define qué acciones y recursos puede llevar a cabo un usuario una vez autenticado. Esto se logra mediante roles, permisos y privilegios.

**Control de Acceso:** El control de acceso se refiere a la restricción del acceso a la base de datos, lo que incluye la autorización para realizar ciertas operaciones (lectura, escritura, modificación) en tablas y vistas específicas.

**Auditoría y Registro:** La auditoría implica el seguimiento y registro de actividades en la base de datos para detectar y registrar eventos importantes, como intentos de acceso no autorizado o modificaciones de datos sensibles.

**Encriptación:** La encriptación se utiliza para proteger los datos almacenados y transmitidos, garantizando que solo las partes autorizadas puedan acceder a la información en un formato legible.

**Gestión de Contraseñas:** La gestión de contraseñas incluye políticas de seguridad que rigen la creación y el almacenamiento seguro de contraseñas, así como la gestión de contraseñas caducadas o comprometidas.

**Control de Transacciones:** El control de transacciones garantiza que las transacciones se completen con éxito y que se mantenga la consistencia de los datos, incluso en caso de fallos o interrupciones.

**Restricciones de Integridad:** Se establecen reglas y restricciones en la base de datos para garantizar la integridad de los datos, como claves primarias, claves foráneas y validaciones de integridad.

**Segmentación de Red:** La segmentación de red se utiliza para aislar la base de datos y limitar el acceso desde redes no confiables o no autorizadas.

**Actualizaciones y Parches:** Se aplican regularmente actualizaciones y parches de seguridad para corregir vulnerabilidades conocidas en el sistema de gestión de bases de datos (DBMS).

**Cumplimiento Normativo:** La base de datos debe cumplir con regulaciones y estándares de seguridad de datos, como GDPR, HIPAA o PCI DSS, cuando corresponda.

**Evaluación de Amenazas y Vulnerabilidades:** Se realiza una evaluación continua de amenazas y vulnerabilidades para identificar y abordar posibles riesgos de seguridad en la base de datos.

Políticas de Seguridad: Se desarrollan políticas de seguridad que describen las medidas de seguridad, las responsabilidades de los usuarios y los procedimientos a seguir en caso de incidentes de seguridad.

## **Backup**

La copia de seguridad de la base de datos es la creación de una copia (copia de seguridad) de los datos estructurados de una organización utilizados por bases de datos populares como SAP, HANA y Microsoft SQL Server. El objetivo de crear una copia es restaurar los datos originales en caso de que sea necesario restaurar los datos y las cargas de trabajo de la base de datos contra una eliminación accidental, corrupción o ataque. En las implementaciones de nube híbrida, los datos estructurados pueden residir localmente en cargas de trabajo de bases de datos que se ejecutan en servidores físicos, máquinas virtuales o contenedores, o en máquinas virtuales o contenedores en la nube. (A Commvault Venture, 2023)

## **¿Qué es una copia de seguridad de una base de datos?**

El Backup de las bases de datos es una solución de protección para datos estructurados en cargas de trabajo de bases de datos y debería cubrir de manera óptima entornos híbridos, ya que es una necesidad de las empresas que sus datos sean respaldados y restaurados en ambos espacios, como la nube. Los datos estructurados almacenados en bases de datos empresariales son importantes para las empresas que los utilizan para almacenar datos en un formato relacional como backend. A medida que aumenta el costo del tiempo de inactividad, las empresas deben garantizar copias de seguridad confiables de sus datos para una recuperación rápida.

La copia de seguridad de la base de datos resguarda todos los datos estructurados asociados con la carga de bases de datos críticas para el negocio.

Los pasos principales en el proceso de copia de seguridad de la base de datos son los siguientes:

Identificar fuentes de datos estructurados de las que es necesario realizar una copia de seguridad y proteger, como tablas, instancias y registros.

Especificar el tipo de copia de seguridad de la base de datos: completa, diferencial, registro de eventos, solo copia, parcial o archivo y grupo de archivos.

Especificar el destino de los datos de la copia de seguridad, ya sea local, en la nube o ambos.

Ejecutar la copia de seguridad o prográmela para un momento más conveniente. Restaure periódicamente en otra ubicación de la base de datos para verificar los datos.

### **Beneficios de una Backup**

Tener o contar con una Backup o copia de seguridad presentan algunos beneficios, los cuales se describen a continuación:

**Recuperación de Datos:** En caso de pérdida de datos debido a fallos de hardware, errores humanos, ataques cibernéticos, o desastres naturales, las copias de seguridad te permiten recuperar tus datos, evitando la pérdida permanente de información valiosa.

**Continuidad del Negocio:** Si experimentas una interrupción en tus operaciones debido a la pérdida de datos, las copias de seguridad te permiten reanudar tus actividades comerciales rápidamente, minimizando el tiempo de inactividad y sus impactos financieros.

**Protección contra Ransomware y Malware:** En el caso de ataques de ransomware y malware que cifran tus datos y exigen un rescate para su recuperación, puedes evitar el pago de rescates y restaurar tus datos desde las copias de seguridad.

**Seguridad de Datos Sensibles:** Las copias de seguridad protegen los datos confidenciales y sensibles, asegurando que estén resguardados de accesos no autorizados y cumpliendo con regulaciones de privacidad.

**Integridad de Datos:** Las copias de seguridad permiten recuperar versiones previas de los datos, lo que facilita la corrección de errores y la restauración de datos a un estado anterior si se produce corrupción o pérdida de información.

**Pruebas y Desarrollo:** Las copias de seguridad se utilizan para pruebas y desarrollo, lo que permite realizar cambios, probar nuevas configuraciones o realizar actualizaciones sin afectar los datos en producción.

**Historial de Datos:** Las copias de seguridad mantienen un historial de datos a lo largo del tiempo, lo que facilita la recuperación de versiones anteriores o la revisión de cambios en los datos.

**Confiabilidad y Redundancia:** Tener copias de seguridad distribuidas en diferentes ubicaciones o medios garantiza la redundancia y la disponibilidad continua de los datos en caso de fallas.

**Recuperación de Desastres:** Las copias de seguridad son esenciales para la recuperación de desastres, lo que permite a las organizaciones reconstruir su infraestructura de TI y restaurar los servicios en caso de eventos catastróficos.

**Cumplimiento Normativo:** Algunas regulaciones y estándares de la industria exigen la implementación de copias de seguridad como parte de las prácticas de seguridad y cumplimiento de datos.

**Paz Mental:** Saber que tus datos están respaldados te brinda tranquilidad y confianza en la capacidad de recuperar información crítica en situaciones de pérdida de datos.

### **Desventajas de una Backup**

Así como las copias de seguridad son esenciales y beneficiosas para proteger tus datos, también pueden presentar algunas desventajas y desafíos. Aquí hay algunas desventajas comunes de las copias de seguridad:

**Costo de Almacenamiento:** El almacenamiento de copias de seguridad puede ser costoso, especialmente cuando se trata de grandes volúmenes de datos. Cuanto más dato necesites respaldar, más espacio de almacenamiento requerirás, lo que puede aumentar los costos.

**Complejidad:** Configurar y administrar un sistema de copias de seguridad puede ser complejo y requerir conocimientos técnicos. Esto puede ser un desafío para usuarios y organizaciones que no tienen experiencia en administración de datos.

**Tiempo de Respuesta:** La recuperación de datos desde una copia de seguridad puede llevar tiempo, especialmente si los datos se almacenan fuera del sitio o en medios no inmediatamente accesibles. Esto puede causar retrasos en la recuperación de datos críticos.

**Consumo de Recursos:** La creación de copias de seguridad puede consumir recursos del sistema, como capacidad de CPU y ancho de banda de red, lo que puede afectar el rendimiento de otras aplicaciones y procesos.

**Incompatibilidad de Versiones:** Las copias de seguridad de una versión anterior de un sistema o aplicación pueden no ser directamente compatibles con versiones más nuevas. Esto puede requerir conversiones o ajustes al restaurar los datos.

**Posibilidad de Errores Humanos:** Las copias de seguridad pueden estar sujetas a errores humanos, como olvidar programar copias de seguridad regulares o realizar copias de seguridad incorrectas.

**Requisitos de Mantenimiento:** Las copias de seguridad requieren mantenimiento constante, como la gestión de cadenas de retención, la supervisión y las pruebas regulares. Esto puede ser una carga adicional para los administradores de bases de datos.

**Confidencialidad de Datos:** La información contenida en las copias de seguridad también debe ser protegida, ya que puede ser un objetivo para el acceso no autorizado o filtración de datos.

**Complejidad de Restauración:** La restauración de datos desde una copia de seguridad puede ser complicada si no se lleva a cabo de manera adecuada. Debe ser un proceso bien planificado y probado para garantizar una recuperación exitosa.

**Cumplimiento Normativo:** En algunos casos, las copias de seguridad pueden ser objeto de regulaciones y normativas de seguridad de datos, lo que requiere controles y medidas adicionales.

### **Importancia de hacer un Backup de manera segura**

La seguridad un Backup es fundamental para proteger los datos críticos. Una estrategia sólida de copias de seguridad, que incluye prácticas de cifrado, control de acceso y almacenamiento seguro, es esencial para mantener la integridad y confidencialidad de tus datos de respaldo. A continuación, algunas prácticas clave para asegurar que tus copias de seguridad sean seguras:

**Cifrado de Datos:** Encripta los datos de respaldo para que solo puedan ser leídos por personas autorizadas. Asegúrate de utilizar algoritmos de cifrado fuertes y estándares.

**Almacenamiento Seguro:** Guarda las copias de seguridad en ubicaciones seguras y controladas, como un servidor de respaldo en un centro de datos o en dispositivos de almacenamiento encriptados. Evita almacenar copias no cifradas en ubicaciones públicas o no seguras.

**Gestión de Acceso:** Limita el acceso a las copias de seguridad solo a personal autorizado. Utiliza autenticación y autorización para controlar quién puede acceder y modificar los respaldos.

**Control de Versiones:** Mantén un registro de versiones de tus copias de seguridad para poder rastrear cambios y recuperar versiones anteriores si es necesario.

**Protección de Contraseñas:** Asegúrate de que las contraseñas utilizadas para acceder a las copias de seguridad sean fuertes y se almacenen de manera segura. No almacenes contraseñas junto con los respaldos.

**Auditoría y Registro:** Establece un sistema de auditoría y registro que registre quién accede a las copias de seguridad y qué acciones se realizan.

**Conexiones Seguras:** Si se realizan copias de seguridad a través de una red, utiliza conexiones seguras, como SSH o VPN, para proteger la transferencia de datos.

**Almacenamiento en Medios Seguros:** Si se utilizan dispositivos de almacenamiento físico (como discos duros externos o cintas), almacénalos de manera segura y fuera del alcance de terceros.

**Recuperación de Datos en Casos de Pérdida:** Planifica la recuperación de datos en caso de pérdida o daño de las copias de seguridad. Esto puede incluir la distribución de copias en múltiples ubicaciones.

**Eliminación Segura:** Cuando se retiren medios o copias de seguridad obsoletos, asegúrate de eliminar los datos de manera segura para evitar fugas de información.

**Seguridad Física:** Protege las ubicaciones físicas donde se almacenan las copias de seguridad para evitar el robo o daño físico.

**Pruebas de Recuperación:** Realiza pruebas regulares de recuperación de datos para garantizar que tus copias de seguridad sean efectivas y puedas recuperar la información en caso de necesidad.

**Mantenimiento de Software:** Mantén actualizado el software de respaldo y las herramientas relacionadas para asegurarte de que estén protegidos contra vulnerabilidades conocidas.

**Cumplimiento Normativo:** Asegúrate de que tus prácticas de copias de seguridad cumplan con las regulaciones y normativas de seguridad de datos aplicables a tu industria.

## Reingeniería de base de datos con Workbench

La reingeniería o rediseño de la base de datos implica cambiar la estructura y el diseño existentes para mejorar la eficiencia, la escalabilidad y la eficacia, y mantenerlo actualizado con las necesidades cambiantes de la organización. Para realizar reingeniería (ingeniería inversa) en la base de datos de Workbench es necesario tener en cuenta las indicaciones que se encuentran en la página oficial de MySQL.

Esta actividad puede ser necesaria por varias razones y sirve para alcanzar una serie de objetivos clave:

**Optimización del Rendimiento:** La reingeniería de bases de datos se utiliza para mejorar el rendimiento de una base de datos que puede haberse vuelto lenta debido al crecimiento de datos, cambios en el volumen de datos o ineficiencias en el diseño inicial. Esto puede incluir la reorganización de tablas, la adición de índices, la eliminación de redundancias y la optimización de consultas.

**Adaptación a Cambios en Requisitos:** A medida que las necesidades del negocio evolucionan, es posible que sea necesario adaptar la estructura de la base de datos. La reingeniería permite hacer cambios para cumplir con nuevos requisitos, como agregar campos, tablas o relaciones.

**Normalización o Denormalización:** Dependiendo de los requisitos específicos, la reingeniería de bases de datos puede implicar la normalización o denormalización de la estructura para garantizar una gestión eficiente de los datos.

**Mejora de la Seguridad:** Si se identifican vulnerabilidades de seguridad en la base de datos existente, la reingeniería puede utilizarse para fortalecer las medidas de seguridad, como la aplicación de restricciones de acceso y la incorporación de cifrado.

**Eliminación de Datos Obsoletos:** A lo largo del tiempo, una base de datos puede acumular datos obsoletos o inútiles. La reingeniería permite limpiar y purgar datos que ya no son necesarios.

**Mejora de la Eficiencia de Almacenamiento:** A medida que los datos crecen, la optimización del almacenamiento se vuelve importante. La reingeniería puede incluir la implementación de compresión de datos y otras técnicas para reducir el espacio de almacenamiento necesario.

**Facilitación de la Integración:** Cuando se deben integrar bases de datos existentes con sistemas o aplicaciones nuevos, la reingeniería es útil para alinear las estructuras y garantizar la compatibilidad.

**Estándares de Datos:** La reingeniería puede utilizarse para estandarizar la estructura de la base de datos de acuerdo con estándares y mejores prácticas reconocidos.

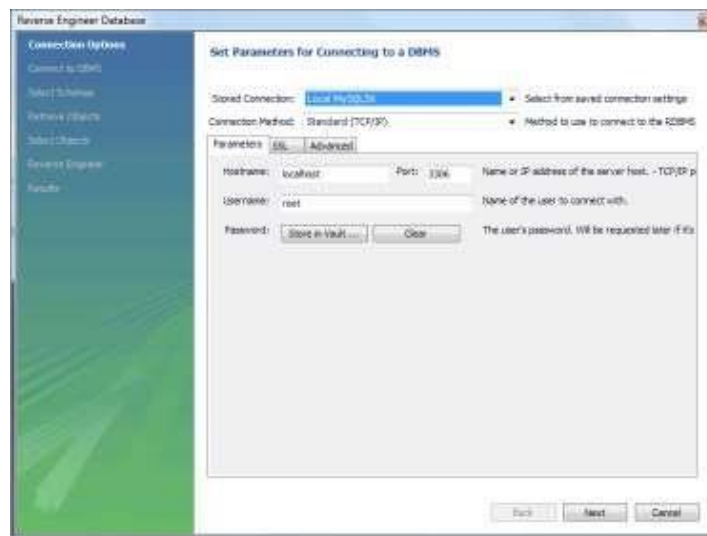
**Documentación y Comprender la Base de Datos:** A menudo, las bases de datos carecen de documentación adecuada. La reingeniería es una oportunidad para documentar la estructura y el diseño de la base de datos, lo que facilita su comprensión y mantenimiento.

**Cumplimiento Normativo:** Si se producen cambios en las regulaciones de seguridad de datos o de privacidad, la reingeniería puede ser necesaria para cumplir con los requisitos normativos actualizados.

A continuación, se detallan las instrucciones:

Primero hacemos clic en “Base de datos” y luego en “Ingeniería inversa” en el menú. A continuación, se podrá observar lo siguiente:

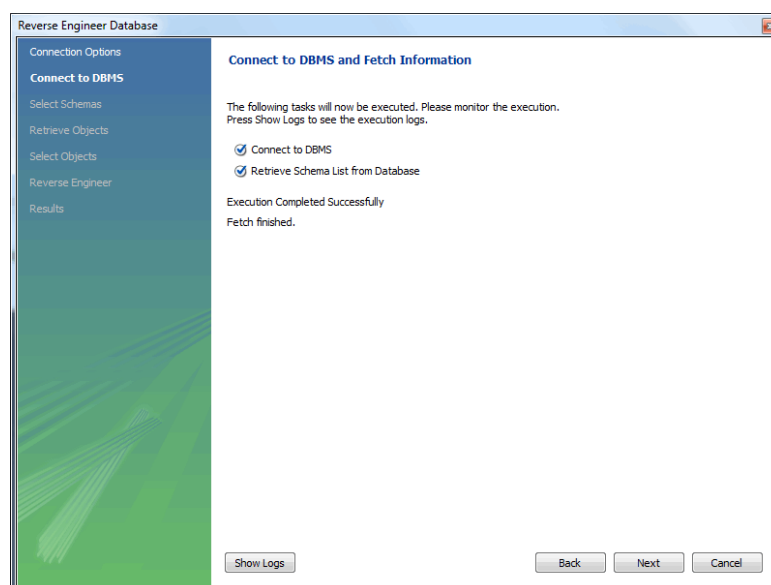
Figura 8. Asistente de BD de ingeniería inversa



Fuente: (Oracle, 2023)

En esta ventana, se puede especificar la conexión a la base de datos activa a la que vamos a aplicar la ingeniería inversa. Es posible crear una nueva conexión o seleccionar una conexión guardada creada previamente. La información típica requerida para establecer una conexión es el nombre de host, el nombre de usuario y la contraseña. Una vez ingresada esta información o seleccionar una conexión guardada, hacer clic en el botón “siguiente”.

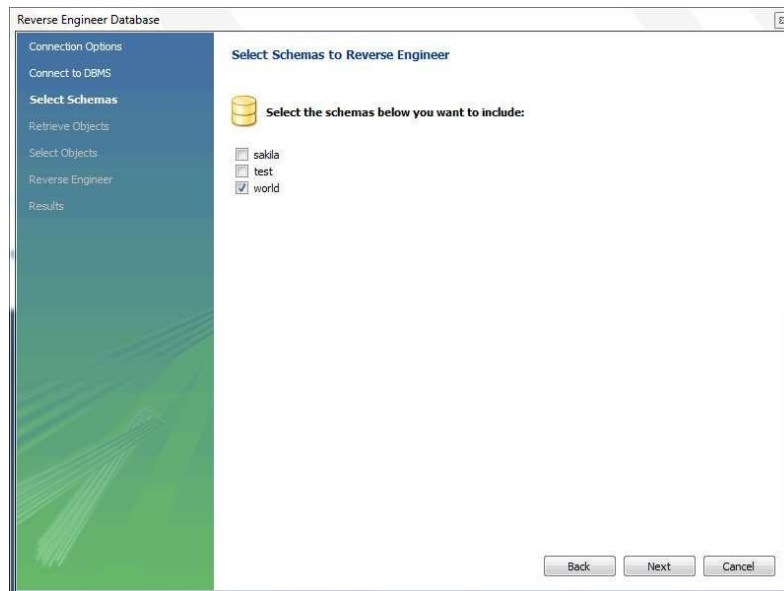
Figura 9. Conexión a DBMS



Fuente: (Oracle, 2023)

Una vez revisada la información que se muestra para verificarde que no se haya perdido la conexión y hacer clic en “Siguiente”. Luego observaremos como se muestran los esquemas disponibles en el servidor. Marcar la casilla de cada esquema a procesar. Luego se selecciona el esquema.

Figura 10. Seleccionar esquemas



Fuente: (Oracle, 2023)

Una vez seleccionados los esquemas hacemos clic en “Siguiente” y se mostrará la siguiente ventana.

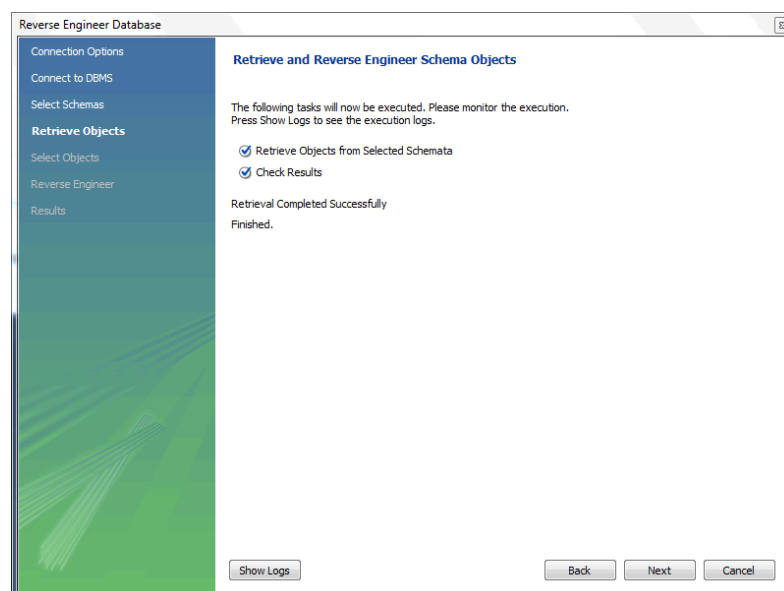
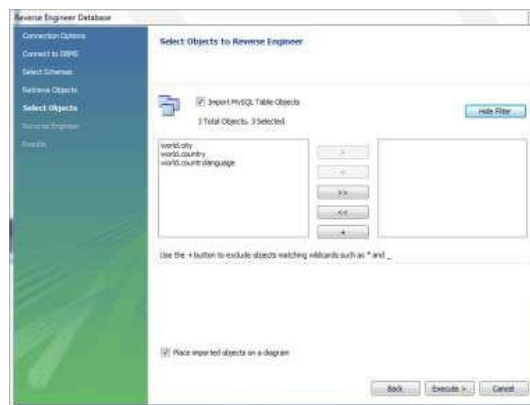


Figura 11. Recuperar objetos

Fuente: (Oracle, 2023)

Aquí es importante revisar los resultados antes de hacer clic en “Siguiente”. Luego se abre la página Seleccionar objetos para ingeniería inversa. Esta ventana tiene una sección para cada tipo de objeto de esquema que se puede importar (tablas, vistas, rutinas, etc.). Todos los tipos de objetos están seleccionados de forma predeterminada. Colocar objetos importados en el gráfico también está seleccionado de forma predeterminada.

Figura 12. Progreso de la ingeniería inversa

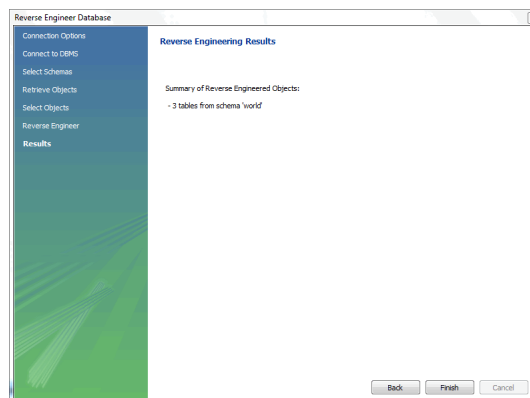


Fuente: (Oracle, 2023)

Hacer clic en “Siguiente” para continuar.

El último paso del asistente presenta un resumen de los objetos sometidos a ingeniería inversa.

Figura 13. Resultados de la ingeniería inversa



Fuente: (Oracle, 2023)

Hacemos clic en finalizar para finalizar el proceso y antes de cerrar MySQL Workbench guardamos el archivo generado.

## ACTIVIDADES RECOMENDADAS

Ejercicio práctico de reingeniería de base de datos en Workbench.

## BIBLIOGRAFÍA BÁSICA

Nevado Cabello, V. (2017). INTRODUCCIÓN A LAS BASES DE DATOS RELACIONALES. Vision Libros. Obtenido de <https://books.google.es/books?hl=es&lr=&id=0lUpB1lNUdIC&oi=fnd&pg=PA11&dq=modelo+de+entidad+relacion,+modelo+logico+de++una+abse+de+datos&ots=sLXYG0sTPK&sig=KnGLglHB59X4MeBZEeZGwucLLGE#v=onepage&q&f=false>

Osorio Rivera, F. (2008). Base de datos relacionales. Medellin-Colombia: Fondo Editorial ITM. Obtenido de <https://books.google.es/books?hl=es&lr=&id=buM5rlZME-cC&oi=fnd&pg=PA9&dq=base+de+datos+&ots=6M5JnsEJQN&sig=3reXLGqOafLeuowtmGe0DrGiTAo#v=onepage&q=base%20de%20datos&f=false>

## BIBLIOGRAFÍA DE CONSULTA

A Commvault Venture. (2023). Database Backup. *Metallic.Io*. <https://metallic.io/knowledge-center/glossary/database-backup>

Bilal Hungund. (2018, septiembre 24). SQL Trigger | Student Database. *GeeksforGeeks*. <https://www.geeksforgeeks.org/sql-trigger-student-database/>

Cloud Software Group, Inc. (2023). *What is a Logical Data Model?* TIBCO Software. <https://www.tibco.com/reference-center/what-is-a-logical-data-model>

ESIC Business & Marketing School. (2018, enero). *Modelo entidad relación: Descripción y aplicaciones*. <https://www.esic.edu/rethink/tecnologia/modelo-entidad-relacion-descripcion-aplicaciones>

GeeksforGeeks. (2015a, julio 8). Normal Forms in DBMS - Database Normalization. *GeeksforGeeks*. <https://www.geeksforgeeks.org/normal-forms-in-dbms/>

GeeksforGeeks. (2015b, octubre 13). Introduction for ER Model. *GeeksforGeeks*. <https://www.geeksforgeeks.org/introduction-of-er-model/>

GeeksforGeeks. (2017, noviembre 6). SQL | DDL, DQL, DML, DCL and TCL Commands.

*GeeksforGeeks*. <https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/>

IBM Corporation. (2021a, marzo 3). *Claves foráneas*. <https://www.ibm.com/docs/es/informix-servers/12.10?topic=s8gu8g-12-1-0-com-ibm-ddi-doc-ids-ddi-182-htm>

IBM Corporation. (2021b, marzo 3). *Claves primarias*. <https://www.ibm.com/docs/es/informix-servers/12.10?topic=s8gu8g-12-1-0-com-ibm-ddi-doc-ids-ddi-182-htm>

IBM Corporation. (2021c, marzo 3). *Normalización*. <https://www.ibm.com/docs/es/informix-servers/12.10?topic=s8gu8g-12-1-0-com-ibm-ddi-doc-ids-ddi-182-htm>

IBM Corporation. (2022, mayo 11). *Vistas de base de datos*. <https://www.ibm.com/docs/es/mfci/7.6.2?topic=structure-views>

IONOS Inc. (2019, julio 31). *INNER JOIN: Definition and application*. IONOS Digital Guide. <https://www.ionos.com/digitalguide/hosting/technical-matters/sql-inner-join/>

MichoGarcía, & Jorge Arévalo. (2013). *Conceptos básicos de SQL*. [https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos\\_sql.html](https://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html)

Oppel Andy, C. F. J., & Sheldon Robert. (2010). *Fundamentos de SQL. III*. [https://d1wqtxts1xzle7.cloudfront.net/54477680/Fundamentos\\_de\\_SQL\\_3ra\\_Edicion\\_-](https://d1wqtxts1xzle7.cloudfront.net/54477680/Fundamentos_de_SQL_3ra_Edicion_-)

\_Andy\_Oppel-FREELIBROS.ORG-libre.pdf?1505859961=&response-content-disposition=inline%3B+filename%3DFundamentos\_de\_SQL\_3ra\_Edicio\_n\_Andy\_Opp.pdf&Expires=1693957914&Signature=acztb5er1s6k0i6JR9gR83dFnCMfjHv4w~Dd6bPb7lwEaFE3vBHqMqhxeqdHjoggUwYbBny3lII2742R6rjHEYRwLiJwSq5NXT5PvQkndaQOJrg-9A0bDpzD43F4fVrUp8wAS5RlI1xZD78~aLlc9DblRblDYfmZ5fxmS1yFFbN16n~MywLoCST3NeaKdFV Vn3EtYRbLLk9L5QStJl0A5dy-p2CUflstixzaQJeevN6CqfThVteiGomtOXM9Y1Z~s2BK7yBF5anuf4m945QTm~uGOQ9EhF-qYvi2rBxxS8uNDaaVNw30Gc-Trw5cLAjQUT3o9TSL5CA1R5hpK1YUA\_\_\_\_\_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Oracle. (2023). *MySQL: MySQL Workbench Manual: 9.4.2.2 Ingeniería inversa de una base de datos activa*. <https://dev.mysql.com/doc/workbench/en/wb-reverse-engineer-live.html>

Reinosa, E., Maldonado, C., Muñoz, R., Damiano, L., & Abrutsky, M. (2012). *Bases de datos*. AlphaEditorial.

Sánchez Jorge. (2004). *Principios sobre bases de datos relacionales*. <https://cursa.ihmc.us/rid=1H73QYLH3-6LFRCX-JT6/bdrelacional.pdf>

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2002). *Database system concepts* (4th ed). McGraw-Hill.



**MISAEI ACOSTA**

CARRERA DE  
DESARROLLO DE SOFTWARE

Dirección: Loja 31 - 15 y Buenos Aires  
Teléfono fijo: 032 961 680  
Teléfono móvil: +593 98 818 3081

**RIOBAMBA - ECUADOR**