



# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS



### BASES DE DATOS (TSDS)

ASIGNATURA:

Bases de Datos

PROFESOR:

Ing. Lorena Chulde

FECHA DE ENTREGA:

2025 - 02-05

PERÍODO ACADÉMICO:

2024-B

## PROYECTO FINAL

## TÍTULO

## AEROLINEAS



**Estudiantes**  
**Luis Oña**  
**Alisson Viracocha**

# INTRODUCCION

En el respectivo proyecto se hace con el fin de presentar una base de datos general (aerolíneas) donde la misma tendrá varias funciones como es la creación de usuario donde este tendrán un rol en específico donde demostrara que no todos los usuarios pueden hacer las realizar las mismas funciones para modificar la base. La implementación de búsquedas optimizadas como son las vistas y la modificación de las mismas a través de los triggers ya que este funcionara como un auditor donde tendrá una tabla que podrá generar o registrar los cambios hechos dentro de la base de datos

## Objetivo General

- Aplicar los conocimientos adquiridos como funciones, respaldos, ingreso y eliminación de datos así también tener una presentación funcional para su respectivo análisis.

## Objetivos Específicos

- Hacer el planteamiento de caso de estudio a realizar.
- Realizar los modelos (conceptual, lógico, físico).
- Aplicar las búsquedas y procedimientos (triggers, procedimientos almacenados y vistas)
- Implementar la encriptación de datos, creación de usuario(roles) .

## **Instrucciones Generales**

### **1. Modelado de Base de Datos y Diccionario de Datos**

**Objetivo:** Crear un diseño eficiente y bien documentado para la base de datos, utilizando el modelado ER y un diccionario de datos completo.

**Actividades:**

**Diseñar el modelo conceptual, lógico y físico.**

**Práctica:** modelo entidad-relación reflejando las entidades clave y sus relaciones.

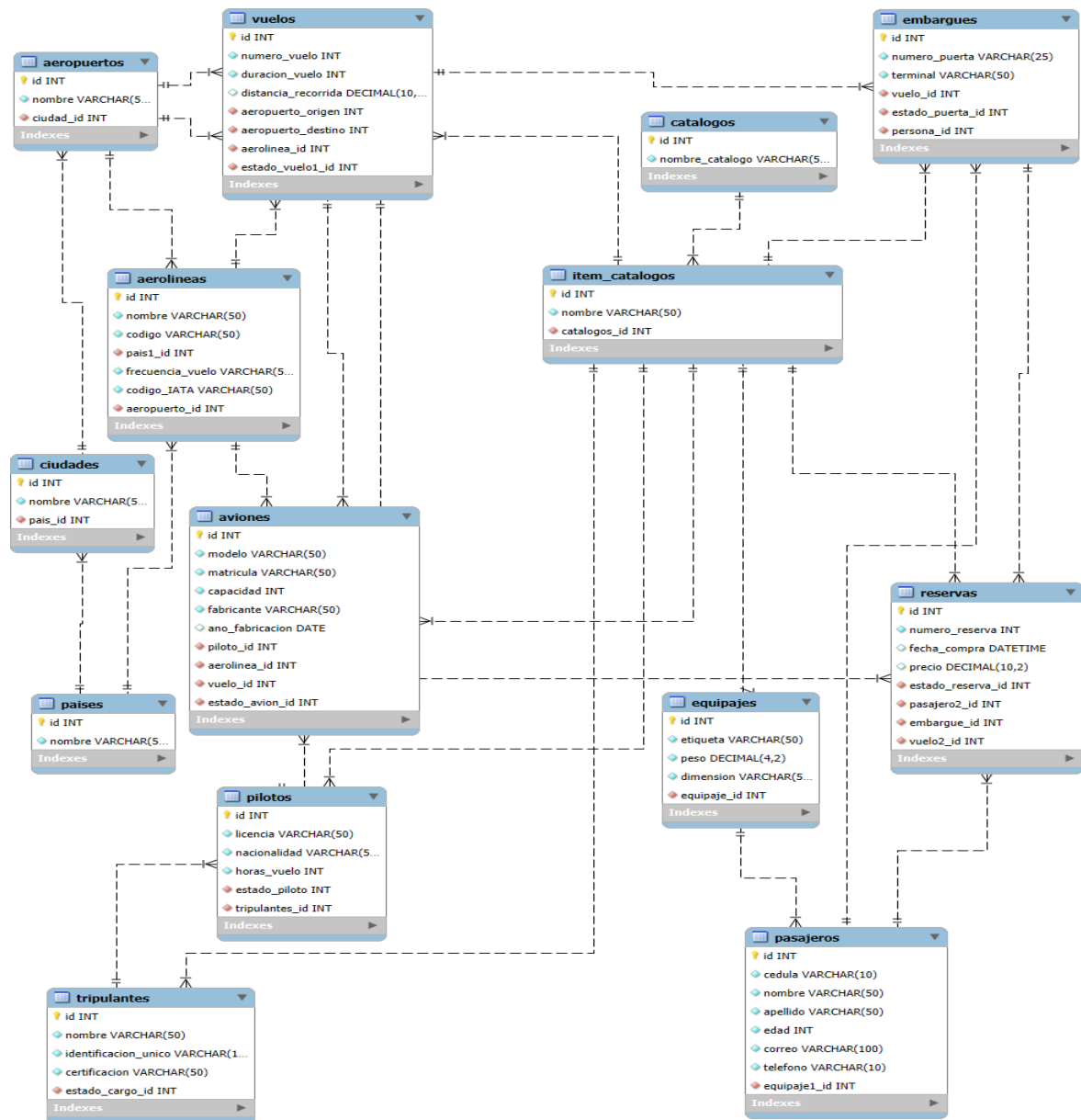


Imagen1.1 modelo lógico de aerolíneas  
Fuente:(propia)

```

create database aerolineas;
use aerolineas;

create table catalogos(
id int auto_increment primary key,
nombre_catalogo varchar(50)not null
);

create table item_catalogos(
id int auto_increment primary key,
nombre varchar(50) not null,
catalogos_id int not null,
constraint fk_catalogo foreign key(catalogos_id ) references catalogos(id) on delete
cascade
);
-- drop table item_catalogos
create table equipajes(
id int auto_increment primary key,
etiqueta varchar(50) not null unique,
peso decimal(4,2)not null,
dimension varchar(50) not null,
equipaje_id int not null,
constraint fk_item foreign key(equipaje_id) references item_catalogos(id) on delete
cascade

);

create table pasajeros(
id int auto_increment primary key,
cedula varchar(10) not null,
nombre varchar(50) not null,
apellido varchar(50) not null,
edad int not null,
correo varchar(100) not null unique,
telefono varchar(10) not null,
equipaje1_id int not null,
constraint fk_equipa foreign key(equipaje1_id) references equipajes(id) on delete
cascade

);

create table pilotos(
id int auto_increment primary key,
licencia varchar(50) not null unique,
nacionalidad varchar(50) not null,
horas_vuelo int not null,
estado_piloto int not null,
tripulantes_id int not null,
constraint fk_categoria foreign key(estado_piloto) references item_catalogos(id) on
delete cascade,
constraint fk_nombre foreign key(tripulantes_id) references tripulantes(id) on delete
cascade
);
-- drop table pilotos;

create table paises(
id int auto_increment primary key,
nombre varchar(50)not null
);
create table ciudades(
id int auto_increment primary key,
nombre varchar(50)not null,
pais_id int not null,
constraint fk_pais foreign key(pais_id) references paises(id) on delete cascade
);

create table aeropuertos(
id int auto_increment primary key,
nombre varchar(50)not null,
ciudad_id int not null,
constraint fk_ciudad foreign key(ciudad_id) references ciudades(id) on delete cascade
);

```

*Imagen 1.2 modelo físico de aerolíneas*

*Fuente:(propia)*



**Investigación:** cómo hacer escalables los modelos de bases de datos para sistemas de reservas en aerolíneas.

## **1. Elija el tipo de base de datos adecuado**

El primer paso para diseñar una base de datos escalable es elegir el tipo de base de datos adecuado para sus datos y casos de uso. Hay dos tipos principales de bases de datos: relacionales y no relacionales. Las bases de datos relacionales almacenan datos en tablas con esquemas y relaciones predefinidos, y utilizan un lenguaje de consulta estructurado (.SQL) para manipular y consultar datos. Las bases de datos no relacionales almacenan datos en varios formatos, como pares clave-valor, documentos, gráficos o columnas, y usan diferentes lenguajes de consulta o API para acceder a los datos. Las bases de datos relacionales son buenas para la coherencia, la integridad y la normalización de los datos, pero pueden tener limitaciones de escalabilidad debido a esquemas rígidos y combinaciones complejas. Las bases de datos no relacionales son buenas para la flexibilidad, la disponibilidad y la escalabilidad de los datos, pero pueden tener desafíos con la calidad, la seguridad y la complejidad de los datos.

## **2. Aplicar técnicas de modelado de datos**

El segundo paso en el diseño de una base de datos escalable es aplicar técnicas de modelado de datos para definir la estructura, las relaciones y las restricciones de los datos. El modelado de datos le ayuda a comprender los requisitos de datos, optimizar el rendimiento de la base de datos y evitar la redundancia y la incoherencia de los datos. Hay tres niveles de modelado de datos: conceptual, lógico y físico. El modelado conceptual de datos se centra en las entidades y relaciones de alto nivel de los datos, sin tener en cuenta los detalles técnicos de la base de datos. El modelado de datos lógicos especifica los atributos, las claves y las restricciones de cada entidad y relación, y los asigna al tipo de base de datos. El modelado de datos físicos implementa el modelo de datos lógicos en el sistema de base de datos y define los índices, las particiones y las opciones de almacenamiento.

## **3. Optimiza tus consultas**

El tercer paso en el diseño de una base de datos escalable es optimizar las consultas para reducir la carga en la base de datos y mejorar el tiempo de respuesta. Las consultas son comandos que se utilizan para recuperar, insertar, actualizar o eliminar datos de la base de datos. Para optimizar las consultas, debe usar índices para acelerar la búsqueda y la ordenación de los datos, pero tenga en cuenta la indización excesiva o insuficiente de las tablas. Las combinaciones se pueden usar para combinar datos de varias tablas, sin embargo, evite combinaciones innecesarias o complejas que puedan afectar al rendimiento. Los filtros y las condiciones se pueden usar para limitar la cantidad de datos devueltos por las consultas; sin embargo, se deben evitar funciones u operadores que puedan ralentizar la ejecución. Además, el almacenamiento en caché se puede usar para almacenar datos estáticos o a los que se accede con frecuencia en la memoria, pero evite el almacenamiento en caché de datos que cambien con frecuencia o que sean confidenciales.

#### **4.Implementación de la creación de particiones de datos**

El cuarto paso en el diseño de una base de datos escalable es implementar la partición de datos para distribuir los datos entre varios servidores o discos. La creación de particiones de datos le ayuda a escalar la base de datos horizontalmente, lo que significa agregar más nodos o recursos para controlar la carga de datos, en lugar de verticalmente, lo que significa actualizar los nodos o recursos existentes. Hay dos tipos principales de partición de datos: particionamiento y replicación. La fragmentación es el proceso de dividir los datos en fragmentos más pequeños e independientes, llamados fragmentos, y asignarlos a diferentes servidores o discos. La fragmentación mejora el rendimiento y la disponibilidad de la base de datos, pero también presenta desafíos con la coherencia y la complejidad de los datos. La replicación es el proceso de copiar los datos en varios servidores o discos, ya sea total o parcialmente. La replicación mejora la confiabilidad y la tolerancia a errores de la base de datos, pero también consume más almacenamiento y ancho de banda.

#### **5.Supervise y pruebe su base de datos**

El quinto paso en el diseño de una base de datos escalable es monitorearla y probarla constantemente para medir su rendimiento, identificar cualquier problema y realizar los ajustes necesarios. Esto ayuda a garantizar que la base de datos cumpla con sus expectativas, cumpla con los estándares y satisfaga a los usuarios. Para ello, debe utilizar métricas y paneles para realizar un seguimiento de indicadores clave como el rendimiento, la latencia, la tasa de error y la utilización de recursos. Además, los registros y las alertas pueden registrar eventos y actividades como consultas, transacciones, errores y excepciones. Además, los puntos de referencia y las pruebas de carga pueden simular y evaluar el comportamiento y la capacidad de la base de datos en diversos escenarios y cargas de trabajo. El diseño de una base de datos escalable es un proceso continuo que requiere una planificación, implementación y evaluación cuidadosas. Siguiendo estos pasos, puede diseñar una base de datos que pueda manejar las necesidades de datos con eficacia y eficiencia.

**Importancia del Conocimiento:** Conocer cómo diseñar bases de datos adecuadas asegura la eficiencia y fácil mantenimiento de la aplicación.

Para diseñar una base de datos primero debemos identificar los requerimientos del sistema y que tipos de atributos van a tener cada uno de estos, debemos tener en cuenta las relaciones que tendrán, reducir las redundancias de datos, definir las claves primarias y foráneas.



1. Desarrollar un diccionario de datos detallado.

Práctica: diccionario definiendo tablas, campos, relaciones y restricciones.

• [vuelos](#)

aerolíneas

Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		
nombre	VARCHAR(50)		✓							nombre de la aerolínea encargada
codigo	VARCHAR(50)		✓							numero unico de la aerolínea para ser identificada
pais1_id	INT		✓							hace referencia de la tabla pais
frecuencia_vuelo	VARCHAR(50)		✓							indicara si el vuelo es a diario o semanal
codigo_IATA	VARCHAR(50)		✓							El codigo IATA (Asociacion Internacional de Transporte Aereo) es un codigo de tres letras que identifica a cada aerolínea en el mundo
aeropuerto_id	INT		✓							hace referencia de la tabla aeropuerto

aeropuertos

Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		
nombre	VARCHAR(50)		✓							nombre del aeropuerto
ciudad_id	INT		✓							referencia de la tabla ciudades

aviones

Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		
modelo	VARCHAR(50)		✓							cada avion tiene un modelo en especifico
matricula	VARCHAR(50)		✓							todo avion debe tener una matricula para su respectiva identificacion
capacidad	INT		✓							dependera si el avion es de pasajeros, carga o mixta
fabricante	VARCHAR(50)		✓							cada avion esta fabricado por una empresa responsable
ano_fabricacion	DATE								NULL	indicara la fecha de fabricacion para un futuro tomar acciones correspondientes
piloto_id	INT		✓							hace referencia de la ta bla item_catalogos

Imagen1.4 Diccionario de aerolíneas

Fuente:(propia)

**Investigación:** mejores herramientas y métodos para generar diccionarios de datos y su uso en proyectos reales.

## **HERRAMINETAS**

**ERwin Data Modeler:** Una herramienta potente para el diseño y la documentación de bases de datos. Permite crear modelos lógicos y físicos, generar diccionarios de datos detallados y mantener la consistencia entre el modelo y la base de datos.

**PowerDesigner:** Herramienta para el modelado de datos que ofrece funcionalidades similares a ERwin, incluyendo la generación de diccionarios de datos y la sincronización con la base de datos.

**dbForge Studio:** Una suite de herramientas para diferentes bases de datos (MySQL, SQL Server, Oracle, entre otras) que incluye funcionalidades para el modelado, la documentación y la generación de diccionarios de datos.

## **MÉTODOS**

**Ingeniería inversa:** Los tres programas pueden conectarse a bases de datos existentes y analizar su estructura para generar automáticamente un modelo de datos. Este modelo incluye tablas, columnas, tipos de datos, claves primarias y foráneas, índices y otros objetos de la base de datos. A partir de este modelo, se puede generar un diccionario de datos que refleje la estructura actual de la base de datos.

**Diseño manual:** Estas herramientas permiten a los usuarios crear modelos de datos desde cero, definiendo tablas, columnas, tipos de datos y otros elementos. A medida que se construye el modelo, se puede ir documentando cada elemento con descripciones, comentarios y otra información relevante. Esta información se utiliza para generar el diccionario de datos.

**Metadatos del modelo:** Los modelos de datos creados en estas herramientas almacenan metadatos sobre cada elemento, como el nombre, el tipo de dato, la descripción, las restricciones, etc. Esta información se puede extraer y formatear para generar un diccionario de datos.

**Importancia del Conocimiento:** Un diccionario de datos permite mantener la consistencia y facilita la colaboración entre desarrolladores. Así también nos permite conocer las tablas introducidas en la base de datos dándonos un mejor perspectiva de como está estructurado como atributos, claves, etc.

## 2. Definir las restricciones de integridad referencial.

**Práctica:** claves primarias y foráneas entre las tablas, asegurando la coherencia de los datos



Imagen1.5 relación entre tablas

Fuente:(propia)

```
create table catalogos(  
  id int auto_increment primary key,  
  nombre_catalogo varchar(50) not null  
);  
  
create table item_catalogos(  
  id int auto_increment primary key,  
  nombre varchar(50) not null,  
  catalogos_id int not null,  
  constraint fk_catalogo foreign key(catalogos_id ) references catalogos(id) on delete cascade  
);
```

Imagen1.6 clave primaria y foránea

Fuente:(propia)

Como podemos en la imagen1.5 y 1.6 tenemos nuestra clave primaria en la tabla catálogos que hereda todos sus atributos a la tabla item\_catalogos pero a la vez convirtiéndose en una clave foránea ya que la tabla item\_catalogos dependerá de la tabla catálogos, y eso sucederá en todas las tablas que tengan relación

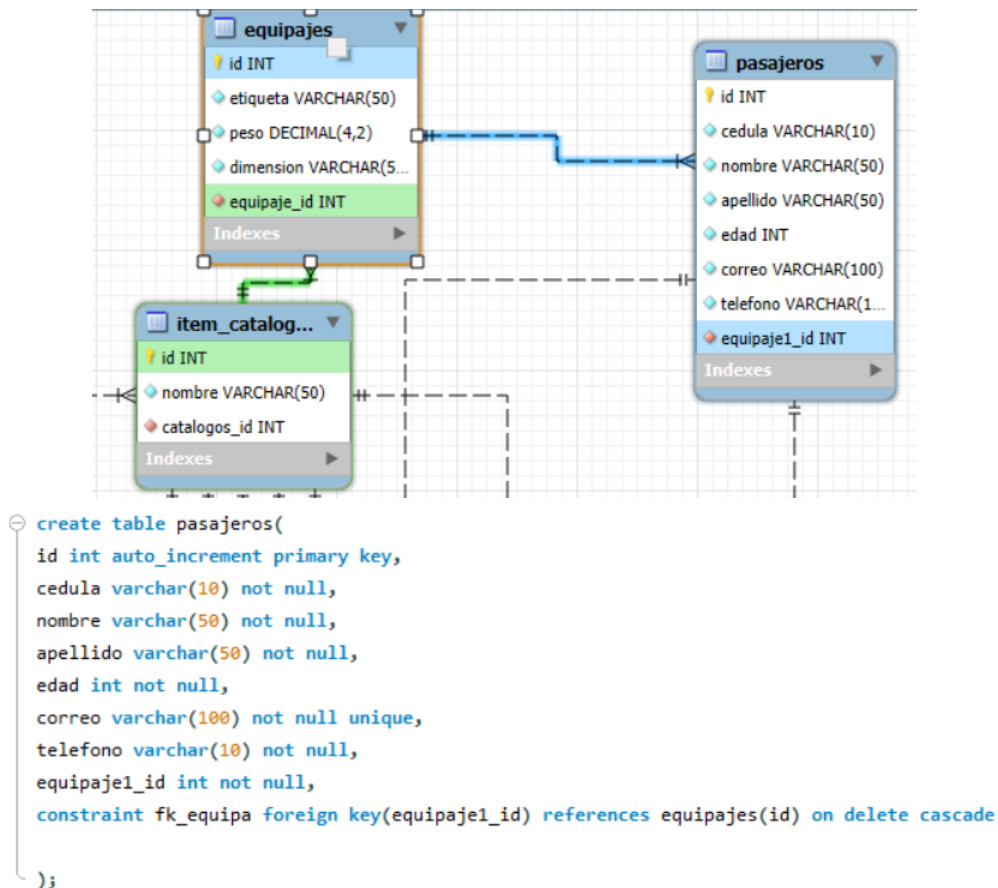


Imagen1.7 clave primaria, foránea y relación de tablas

Fuente:(propia)

Aquí tenemos otro ejemplo(imagen1.7) de como podemos relacionar con otra tabla (tabla pasajeros), podemos relacionar con n tablas dependiendo al necesidad y la estructura de como lo realicemos

**Investigación:** Investigar cómo la integridad referencial puede prevenir la pérdida de datos y mantener la consistencia.

## Importancia de la integridad en los datos

La integridad de los datos en una base de datos es esencial porque es un componente necesario de integración de datos. Si se mantiene la integridad de los datos, los valores de los datos almacenados en la base de datos son coherentes con el modelo y el tipo de datos. Por lo tanto, se pueden obtener conocimientos confiables del modelo de datos para que los usuarios puedan tomar decisiones comerciales informadas.

A continuación, se muestran algunos ejemplos de integridad de los datos en riesgo:

- Un intento de ingresar un número de teléfono en el formato incorrecto.
- Un desarrollador intenta accidentalmente insertar los datos en la tabla incorrecta mientras transfiere datos entre dos bases de datos.

- Un intento de eliminar un registro en una tabla, pero otra tabla hace referencia a ese registro como parte de una relación.
- Un usuario intenta ingresar accidentalmente un número de teléfono en un campo de fecha.

Estos son solo algunos ejemplos de la integridad de los datos en riesgo. Sin embargo, todo esto y más se puede evitar fácilmente. Por ejemplo, las columnas o celdas numéricas no deben incluir información textual para preservar la integridad de los datos. Además, para que los datos estén completos, sus características, como las reglas comerciales, las relaciones, las fechas, las definiciones y el linaje, deben ser precisas.

La integridad de los datos ayuda a garantizar que los datos almacenados en su base de datos se puedan encontrar y vincular a otros datos. Esto garantiza que todo su conjunto de datos se pueda recuperar y buscar cuando sea necesario. Fortalece la estabilidad de los datos, ofrece un rendimiento óptimo y los hace reutilizables y fáciles de mantener.

Ahora que conoce la importancia de la integridad de los datos y los dos métodos para garantizar la integridad de los datos, avancemos hacia los factores que obstaculizan la integridad.

## **Factores que afectan la integridad en una base de datos**

### **Errores humanos**

La entrada manual de datos aumenta las posibilidades de errores, duplicaciones o eliminación. A menudo, los datos ingresados no siguen el protocolo apt, o los errores en la entrada manual pueden extenderse a la ejecución de procesos y, por lo tanto, corromper los resultados. Todos estos problemas ponen en riesgo la integridad de los datos.

### **Errores de transferencia**

Se produce un error de transferencia si los datos no se transfieren correctamente de un sitio dentro de una base de datos a otro. Estos errores generalmente ocurren cuando existe un elemento de datos en la tabla de destino pero está ausente de la tabla de origen dentro de una base de datos relacional.

### **Errores y virus**

La integridad de sus datos también puede verse comprometida debido a spyware, malware y virus que invaden una computadora y alteran, eliminan o roban datos.

### **Cómo garantizar la integridad de los datos en una base de datos**

Estas son algunas de las mejores prácticas de integridad de datos que pueden minimizar o eliminar los riesgos de filtraciones de datos en una base de datos. Los métodos comunes utilizados para la verificación de la integridad de los datos incluyen:

1. Limite el acceso a los datos y cambie los permisos para restringir las modificaciones a los datos por parte de partes no aprobadas.
2. Concéntrese en la validación de datos para garantizar la precisión de los datos cuando se recopilan o integran.

3. Mantenga una copia de seguridad periódica de los datos.
4. Utilice registros para monitorear cuando se ingresan, alteran o borran datos.
5. Realice auditorías internas sistemáticas para garantizar que la información esté actualizada.

**Importancia del Conocimiento:** Las restricciones de integridad aseguran que los datos no se corrompan.

Los datos se mantienen ya que tienen relación una tabla con otra y viceversa, esto hace que la integridad se mantenga y con esto evitamos eliminar por accidentes tablas, si queremos hacer eso al usuario pedirá que elimine en cascada las tablas (es decir una a continuación de otra) ya que la configuración que se hizo fue de manera segura, adicional a esto se debe mantener en constante auditorías

## 2. Seguridad, Auditoría y Control de Acceso

**Objetivo:** Proteger los datos sensibles y controlar el acceso a la base de datos.

**Actividades:**

1. **Implementar políticas de acceso y seguridad.**

**Práctica:** Crear roles y permisos de usuario (por ejemplo, roles de Administrador, Usuario, Auditor) para controlar el acceso a las tablas y vistas.

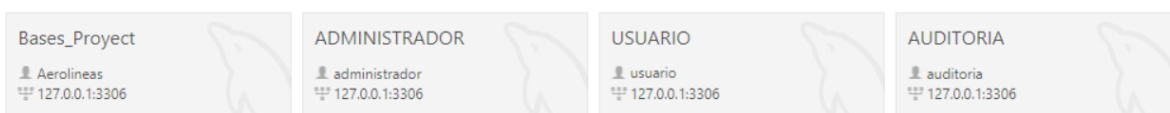
# Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

[Browse Documentation >](#)

[Read the Blog >](#)

[Discuss on the Forums >](#)



*Imagen 2.1 clave primaria, foránea y relación de tablas  
Fuente: (propia)*

```

1 • create user 'Aerolineas'@'localhost';
2 • GRANT ALL PRIVILEGES ON *.* TO 'Aerolineas'@'localhost';
3
4 • SELECT user, host FROM mysql.user;
5 • alter user 'Aerolineas'@'localhost' identified by '1234';
6 -----
7         -- CREACION DE USUARIOS--
8 -- se crean los usuarios pero no tienen permiso alguno
9 • create user 'administrador'@'localhost' identified by 'administrador';
10 • create user 'usuario'@'localhost' identified by 'usuario';
11 • create user 'auditoria'@'localhost' identified by 'auditoria';
12
13
14 -- verificar los usuarios creados
15 • SELECT*FROM mysql.user;
16
17 -----
18
19 -- dando privilegios a los usuarios creados
20         -- ADMINISTRADOR--
21 -- el administrador tendra todos los privilegio
22 • grant all privileges on aerolineas.* to 'administrador'@'localhost';
23 • flush privileges;
24

```

*Imagen2.2 clave primaria, foránea y relación de tablas*

*Fuente:(propia)*

**Investigación:** Investigar sobre los mejores enfoques para la seguridad en bases de datos en entornos de alta disponibilidad.

## Herramientas y plataformas de protección de datos

Hoy en día, una amplia gama de proveedores ofrecen herramientas y plataformas de protección de datos. Una solución a gran escala debe incluir todas las funciones siguientes:

- **Descubrimiento:** busque una herramienta que pueda explorar y clasificar las vulnerabilidades en todas las bases de datos, tanto si están alojadas en cloud o en local, y ofrezca recomendaciones para corregir cualquier vulnerabilidad identificada. Las prestaciones de descubrimiento a menudo son necesarias para cumplir los mandatos de conformidad con la normativa.
- **Supervisión de la actividad de datos:** la solución debe poder supervisar y auditar todas las actividades de datos en todas las bases de datos, independientemente de si el despliegue es local, en cloud o en un [contenedor](#). Debe poder alertarle de actividades sospechosas en tiempo real para que pueda responder a las amenazas más rápidamente. También querrá una solución que pueda aplicar reglas, políticas y separación de funciones y que ofrezca visibilidad sobre el estado de los datos a través de una interfaz de usuario completa y unificada. Compruebe que cualquier solución que elija pueda generar los informes que necesitará para cumplir los requisitos de conformidad.

- **Funciones de cifrado y tokenización:** en caso de infracción, el cifrado ofrece una línea final de defensa contra el compromiso. Cualquier herramienta que elija debe incluir funciones de cifrado flexibles que puedan proteger los datos en entornos locales, en cloud, híbridos o multicloud. Busque una herramienta con funciones de cifrado de archivos, volúmenes y aplicaciones que se ajusten a los requisitos de conformidad de su sector; para ello, pueden ser necesarias funciones de tokenización (enmascaramiento de datos) o funciones avanzadas de gestión de claves de seguridad.
- **Optimización de la seguridad de los datos y análisis de riesgos:** una herramienta que pueda generar información contextual combinando la información de seguridad de datos con analítica avanzada le permitirá realizar la optimización, el análisis de riesgos y la creación de informes con facilidad. Elija una solución capaz de retener y sintetizar grandes cantidades de datos históricos y recientes sobre el estado y la seguridad de las bases de datos, y busque una que ofrezca funciones de exploración, auditoría e informes de datos a través de un panel de control de autoservicio completo pero fácil de usar.

**Importancia del Conocimiento:** El control adecuado de acceso previene fugas de información y mejora la seguridad general

## 2. Cifrado de datos sensibles.

**Práctica:** Cifrar información sensible como contraseñas y detalles de pago (por ejemplo, usando AES\_ENCRYPT en MySQL).

```

96 -- agregamos una nueva tabla
97 • ALTER TABLE pasajeros ADD cedula_encryptado VARBINARY(255);
98 • ALTER TABLE pasajeros ADD cedula_desencryptada VARCHAR(255);
99
100 -- encriptacion de datos
101 • UPDATE pasajeros SET cedula_encryptado = AES_ENCRYPT(cedula, 'encriptado');
102 • UPDATE pasajeros SET cedula_desencryptada = CAST(AES_DECRYPT(cedula_encryptado, 'encriptado') AS CHAR);
103
104 -- verificamos la encriptacion
105 • select*from pasajeros;
106 -- recordadndo

```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	cedula	nombre	apellido	edad	correo	telefono	equipaje1_id	cedula_encryptado	cedula_desencryptada
▶	1	1234567890	Mateo	Torres	16	mateotorres@gmail.com	0952647894	1	BLOB	1234567890
	2	1736185107	Ithan	Camacho	28	ithancamacho@hotmail.com	0987654321	2	BLOB	1736185107
	3	1795195107	Richard	Soria	37	richardsoria@yahoo.es	0987654322	3	BLOB	1795195107
	4	1736635219	Carlos	Perez	65	carlosperez@hotmail.com	0987654323	1	BLOB	1736635219
	5	1755195317	Adrian	Ramos	29	adrianramos@gmail.com	0987654324	2	BLOB	1755195317
	6	1726395709	Adrian	Cadena	33	adrian123@yahoo.es	0987654325	3	BLOB	1726395709
	7	1736492203	Paul	Cabrera	26	paul_Cabrera@hotmail.com	0987654326	3	BLOB	1736492203
	8	1826152241	Mireva	Garcia	28	mireva_garcia@hotmail.com	0987654327	2	BLOB	1826152241

pasajeros 2

×

pasajeros 2 x

Imagen2.1 encriptación y desencriptación de datos

Fuente:(propia)



**Investigación:** Explorar algoritmos de cifrado y su impacto en el rendimiento de la base de datos.

La introducción del cifrado nativo de Db2® a una base de datos existente aumenta los recursos del sistema necesarios y afecta al rendimiento de las cargas de trabajo en ejecución.

El alcance de este impacto depende de dos factores principales:

- Si existe la aceleración de hardware de CPU que Db2 puede aprovechar
- Cómo se ha aislado la carga de trabajo de un aumento en la latencia de las solicitudes de E/S física

Db2 cifrado nativo se basa en el producto de software IBM Global Security Kit (GSKit) incorporado para reconocer y aprovechar la aceleración de hardware de CPU incorporada siempre que sea posible. Esta aceleración hace una diferencia significativa en el impacto tanto en el consumo de recursos del sistema como en el rendimiento de las aplicaciones. A partir de Db2 11.1, Db2 aprovecha las siguientes mejoras de CPU:

- Soporte de nuevas instrucciones estándar de cifrado avanzado de Intel (AES-NI)
- Soporte en núcleo de Power8 para AES
- Series CP Assist for Cryptographic Functions (CPACF)

Dado que el cifrado nativo de Db2 se implementa para cifrar y descifrar datos a medida que van y vienen del disco, el efecto del cifrado aparece en cualquier solicitud de E/S física de Db2. En términos prácticos, el efecto es que el ancho de banda de E/S del sistema se reduce de su nivel actual. La forma en que las cargas de trabajo reaccionan a este cambio determina el impacto en el rendimiento.

Puesto que este cambio en la latencia de E/S física puede negar la configuración ajustada de un sistema de base de datos existente, se recomienda que tenga previsto reajustar una base de datos recién cifrada. Volver a crear la base de datos garantiza que el impacto de cualquier nuevo tiempo de espera de E/S física introducido por el cifrado se soluciona correctamente.

**Importancia del Conocimiento:** El cifrado es esencial para la protección de la información confidencial de los usuarios.

Podemos asegurarnos de cualquier tipo de hacker no lo tendrá fácil para poder obtener cierta información ya que este cifrado tendrá varios caracteres dentro del mismo

### 3. Habilitar auditoría y registrar eventos de base de datos.

**Práctica:** Activar los logs de acceso y auditoría para monitorear las actividades de los usuarios (por ejemplo, registrar quién accedió a qué datos).

```
1 |
2 -- verificar si esta encendido
3 • SHOW GLOBAL VARIABLES LIKE 'general_log';
4
5
6 • set global general_log='on';
7
8 • select*from mysql.general_log
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

event_time	user_host	thread_id	server_id	command_type	argument
------------	-----------	-----------	-----------	--------------	----------

Imagen2.3 activacion del auditoria  
Fuente:(propia)

Aquí podemos observar quien ingreso a la base de datos y que tipo de acción hicieron

Bases_Project Client Connections												
Threads Connected: 12		Threads Running: 2		Threads Created: 13		Threads Cached: 1		Rejected (over limit): 0				
Total Connections: 110		Connection Limit: 151		Aborted Clients: 4		Aborted Connections: 23		Errors: 0				
Id	User	Host	DB	Command	Time	State	Threa...	Type	Name	Paren...	Instrumented	Info
5	event_sched...	localhost	None	Daemon	275230	Waiting on e...	43	FOREGROUND	thread/sql/e...	1	YES	NULL
7	None	None	None	Daemon	275230	Suspending	45	FOREGROUND	thread/sql/c...	1	YES	NULL
91	root	localhost	None	Sleep	46	None	131	FOREGROUND	thread/sql/o...	0	YES	NULL
92	root	localhost	None	Sleep	46	None	132	FOREGROUND	thread/sql/o...	47	YES	NULL
101	usuario	localhost	None	Sleep	246	None	141	FOREGROUND	thread/sql/o...	0	YES	NULL
102	usuario	localhost	aerolineas	Sleep	246	None	142	FOREGROUND	thread/sql/o...	0	YES	NULL
103	administrador	localhost	None	Sleep	196	None	143	FOREGROUND	thread/sql/o...	47	YES	NULL
104	administrador	localhost	aerolineas	Sleep	127	None	144	FOREGROUND	thread/sql/o...	47	YES	NULL
105	administrador	localhost	None	Sleep	43	None	145	FOREGROUND	thread/sql/o...	47	YES	NULL
106	administrador	localhost	None	Sleep	2	None	146	FOREGROUND	thread/sql/o...	47	YES	NULL
107	Aerolineas	localhost	None	Query	0	executing	147	FOREGROUND	thread/sql/o...	47	YES	SELECT t.PROC
108	Aerolineas	localhost	None	Sleep	3	None	148	FOREGROUND	thread/sql/o...	47	YES	NULL
109	Aerolineas	localhost	aerolineas	Sleep	22	None	149	FOREGROUND	thread/sql/o...	47	YES	NULL
110	Aerolineas	localhost	aerolineas	Sleep	22	None	150	FOREGROUND	thread/sql/o...	0	YES	NULL

Imagen2.4 auditoria dentro de Mysql  
Fuente:(propia)

**Investigación:** Buscar cómo configurar herramientas de auditoría en MySQL o PostgreSQL.

Para habilitar la auditoría de datos en MySQL, es necesario configurar las configuraciones necesarias y utilizar las herramientas disponibles. Un enfoque efectivo es aprovechar el poder del control de acceso basado en atributos ([ABAC](#)) en MySQL.

ABAC permite definir políticas de acceso detalladas basadas en atributos asociados con usuarios, recursos y el entorno. Al utilizar ABAC junto con la auditoría de datos, puede controlar de cerca quién accede a ciertos datos y monitorear sus acciones.

A continuación, se muestra un ejemplo de cómo configurar la auditoría de datos usando ABAC en MySQL:

```
-- Habilitar el registro de auditoría
SET GLOBAL audit_log_format = 'JSON';
SET GLOBAL audit_log_file = '/var/log/mysql/audit.log';
SET GLOBAL audit_log = 'ON';
-- Crear una política de auditoría
CREATE AUDIT POLICY policy_name
FOR SELECT, UPDATE, DELETE ON mydb.*
TO 'user1'@'%'
ATTRIBUTE '{"department":"finance","role":"manager"}'
WHEN ('hour(now())' BETWEEN '09' AND '17');
```

*Imagen 2.5 ejemplo para configurar una auditoría*  
*Fuente: (propia)*

En este ejemplo, habilitamos el registro de auditoría, especificando el formato del registro y la ubicación del archivo. Creamos una regla para que **user1** pueda ver, cambiar y eliminar datos en la base de datos **mydb**. La política incluye condiciones de atributo que restringen el acceso basado en el departamento y el rol del usuario. Además, la política incorpora una restricción basada en el tiempo, permitiendo el acceso solo durante el horario laboral (de 9 AM a 5 PM).

**Importancia del Conocimiento:** La auditoría permite rastrear cambios en los datos y detectar actividades sospechosas. Así también podemos observar a tipo de bases entraron que hicieron

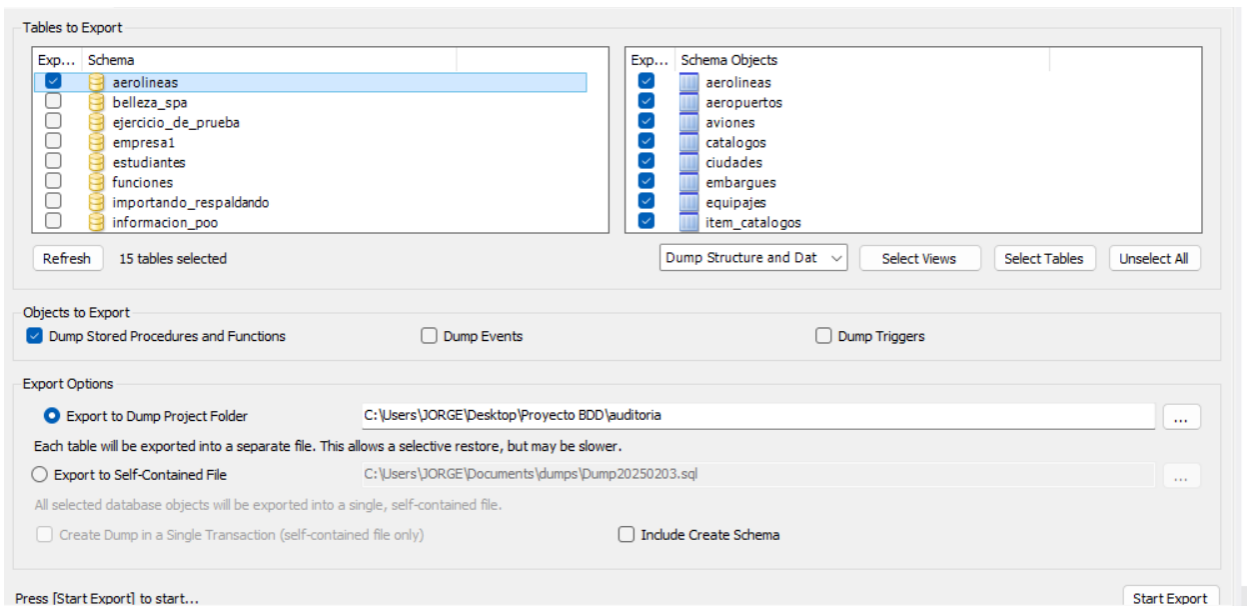
### 3. Respaldos y Recuperación de Datos

**Objetivo:** Asegurar la integridad y disponibilidad de los datos mediante técnicas de respaldo confiables.

**Actividades:**

1. **Crear respaldos completos (full backups).**

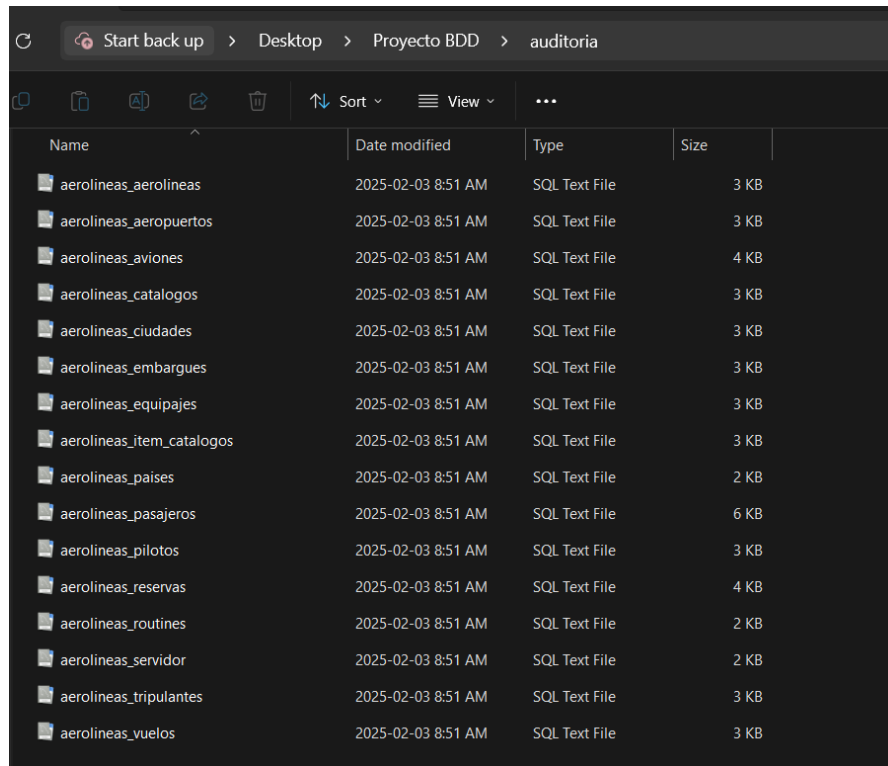
**Práctica:** Utilizar mysqldump o herramientas similares para hacer respaldos completos de la base de datos.



*Imagen3.1 creando respaldo de aerolineas*  
Fuente:(propia)



*Imagen3.2 exportando la base de datos al respaldo*  
Fuente:(propia)



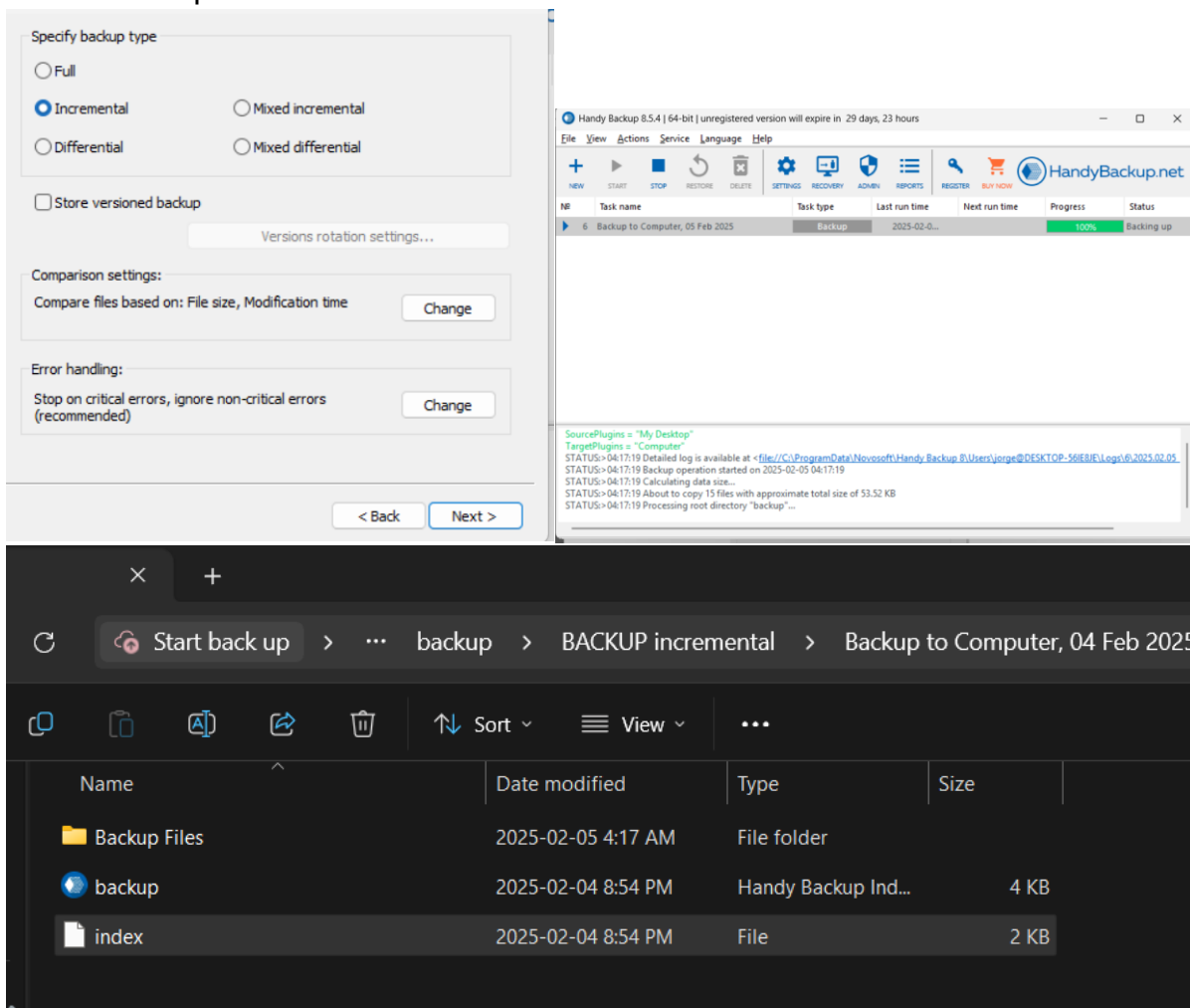
*Imagen3.3 verificando el respaldo de la base de datos*  
*Fuente:(propia)*

**Investigación:** Buscar estrategias de respaldo para bases de datos de gran tamaño y la mejor manera de gestionarlas.

**Importancia del Conocimiento:** Los respaldos completos permiten restaurar toda la base de datos ante una falla. También pueden ser vulnerados ante cualquier evento, con el respaldo le damos una mayor seguridad a la base de datos que se este trabajando

## 2. Configurar respaldos incrementales.

**Práctica:** Realizar respaldos incrementales para reducir el tiempo y espacio de almacenamiento.



**Investigación:** Investigar cómo realizar respaldos incrementales y cuándo es más conveniente utilizarlos.

### Ventajas del backup incremental

- **Optimización del espacio de almacenamiento:** El backup incremental permite ahorrar espacio en el almacenamiento, ya que solo se copian los cambios realizados en los archivos y no la totalidad de los datos.
- **Rapidez en la realización de copias de seguridad:** Al respaldar solo los cambios, el proceso de backup incremental es más rápido en comparación con otros tipos de copias de seguridad que requieren respaldar toda la información.
- **Ahorro de recursos:** Al utilizar menos espacio de almacenamiento y tener un proceso más ágil, el backup incremental optimiza los recursos de hardware y software de la organización.

## Proceso de realización de un backup incremental

El proceso para realizar un backup incremental consta de los siguientes pasos:

1. Realizar una copia completa inicial de todos los datos y archivos.
2. Identificar y respaldar únicamente los archivos que han sufrido modificaciones desde la última copia.
3. Actualizar y guardar los cambios en una ubicación segura destinada al almacenamiento de los backups.

Es importante tener en cuenta que el backup incremental se debe realizar de forma periódica para asegurar la protección de los datos más actualizados y minimizar la pérdida de información en caso de fallos o errores.

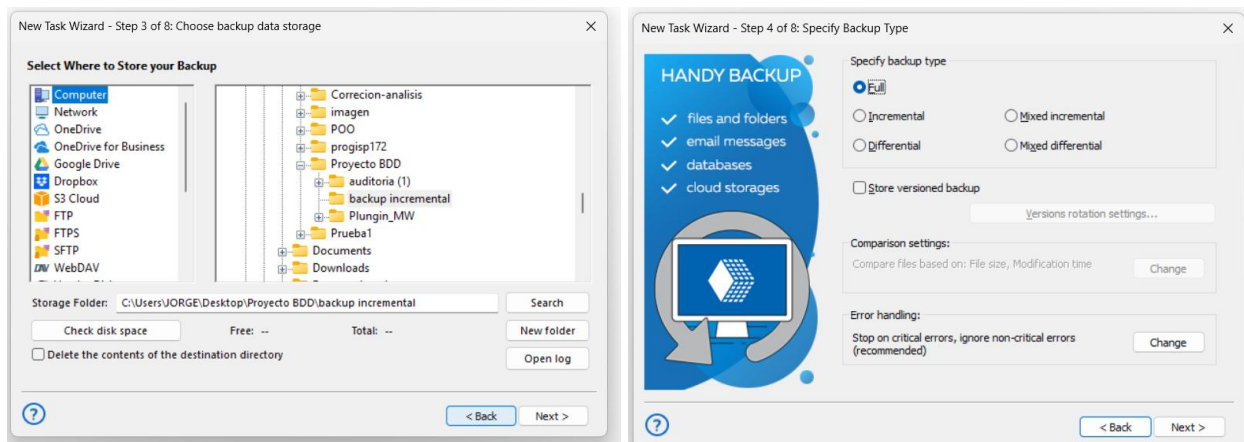
### ¿Cuándo se utilizan las copias de seguridad incrementales?

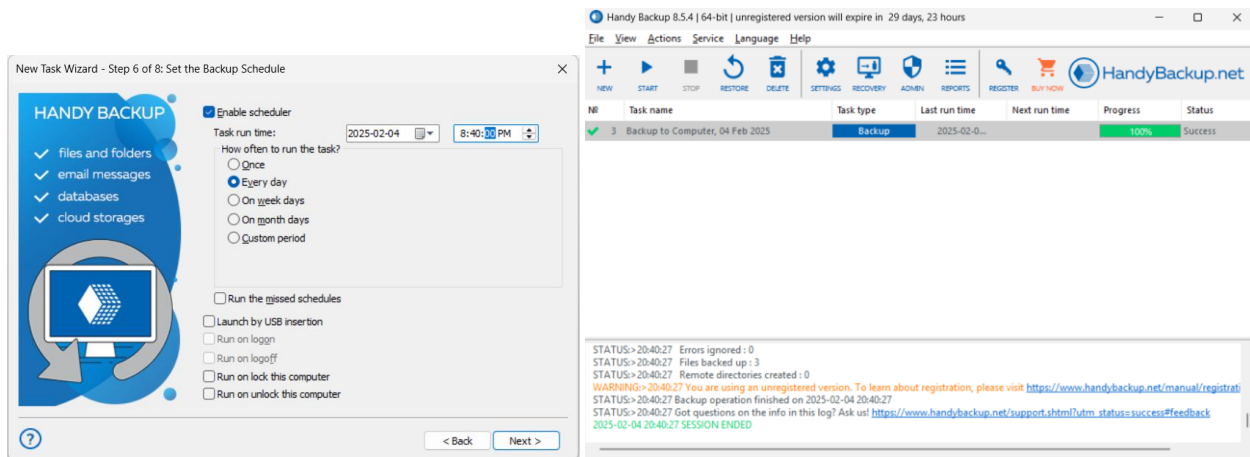
Las copias de seguridad incrementales tienen como objetivo principal minimizar los requisitos de almacenamiento y el tiempo necesario para crear una copia de seguridad. Por eso se utilizan siempre que la creación repetida de copias de seguridad completas no tiene sentido desde el punto de vista logístico. Veamos algunos escenarios en detalle.

**Importancia del Conocimiento:** Los respaldos incrementales permiten optimizar los recursos y acelerar los tiempos de recuperación. Dando una mejor confiabilidad al usuario además este respaldo lo que hace es guardar los últimos cambio hecho en la base de datos

### 3. Implementar respaldos en caliente (Hot Backups).

**Práctica:** Hacer respaldos sin interrumpir el servicio (por ejemplo, usando Percona XtraBackup).





**Investigación:** Investigar cómo hacer respaldos sin detener la base de datos.

Con la herramienta de Handy Backup es un programa de copia de seguridad e imagen de disco desarrollado por Novosoft LLC. Handy Backup hace backup de datos almacenados en ordenadores que funcionan bajo Microsoft Windows OS o tales sistemas Unix-like como Linux. Con este software un usuario puede crear tareas de copia de seguridad, restauración y sincronización a través de una interfaz gráfica de usuario. Esto es una solución unificada de copia de seguridad que permite hacer backup en un modo completamente automático de archivos, carpetas, bibliotecas, imágenes de disco, bases de datos populares, máquinas virtuales, FTP y contenido de nube, y otros datos existentes.

**Importancia del Conocimiento:** Los respaldos en caliente son esenciales para bases de datos de producción que no pueden permitirse inactividad. Este tipo de respaldos son muy importantes ya que están en constante actividad sin que el usuario este preocupado por la respaldacion de la información



## 4. Optimización y Rendimiento de Consultas

**Objetivo:** Mejorar la eficiencia en la recuperación de datos mediante la optimización de consultas y el uso adecuado de índices.

### Actividades:

#### 1. Crear y gestionar índices.

**Práctica:** Implementar índices en las columnas más consultadas, como VueloID, ClienteID, etc.

```
290  -----
291      -- CREANDO INDEX
292  •   use aerolineas;
293  •   CREATE INDEX index_nombre ON pasajeros (nombre);
294
295      -- verificando
296  •   SHOW INDEX FROM pasajeros WHERE Key_name = 'index_nombre';
297
```

Result Grid   Filter Rows:   Export:   Wrap Cell Content:											
	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
▶	pasajeros	1	index_nombre	1	nombre	A	15	NULL	NULL		BTREE

*Imagen4.1 creación del index en la tabla nombre*

*Fuente:(propia)*

```
297
298  •   show index from pasajeros
299
```

Result Grid   Filter Rows:   Export:   Wrap Cell Content:												
	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comm
▶	pasajeros	0	PRIMARY	1	id	A	23	NULL	NULL		BTREE	
	pasajeros	0	correo	1	correo	A	23	NULL	NULL		BTREE	
	pasajeros	1	fk_equipa	1	equipaje1_id	A	3	NULL	NULL		BTREE	
	pasajeros	1	index_nombre	1	nombre	A	15	NULL	NULL		BTREE	

*Imagen4.2 verificación de la creación del index*

*Fuente:(propia)*

```

299
300    -- consultando
301 •   SELECT * FROM pasajeros WHERE nombre = 'Adrian';
302

```

Result Grid								
Filter Rows:								
Edit: Export/Import: Wrap Cell Co								
	id	cedula	nombre	apellido	edad	correo	telefono	equipaje1_id
▶	5	1755195317	Adrian	Ramos	29	adrianramos@gmail.com	0987654324	2
	6	1726395709	Adrian	Cadena	33	adrian123@yahoo.es	0987654325	3
	19	6000263486	Adrian	Cadena	20	adriancadena963@yahoo.es	0988654318	3
★	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Imagen4.3 consulta con el index  
Fuente:(propia)

**Investigación:** Investigar sobre los tipos de índices más adecuados para bases de datos transaccionales y cómo afectan el rendimiento.

## 1. Índice único

Un índice único es un índice que garantiza que cada valor en una columna (o conjunto de columnas) sea único. Este tipo de índice es útil para mejorar la integridad de los datos, ya que evita que se inserten registros duplicados en la base de datos. Los índices únicos también pueden mejorar el rendimiento de las consultas al buscar información específica, puesto que el motor de la base de datos sabe que solo existe un registro que coincide con el valor buscado.

## 2. Índice compuesto

Un índice compuesto es un índice que incluye múltiples columnas de una tabla. Este tipo de índice es especialmente útil cuando se realizan consultas que implican condiciones en varias columnas. Al utilizar un índice compuesto, el motor de la base de datos puede encontrar registros que coinciden con todas las condiciones de búsqueda de manera más eficiente, lo que reduce el tiempo necesario para recuperar la información.

## 3. Índice de texto completo

Un índice de texto completo es un índice diseñado para mejorar el rendimiento de las consultas de búsqueda de texto. Este tipo de índice es especialmente útil en aplicaciones que requieren búsquedas de palabras clave o frases en grandes cantidades de texto, como motores de búsqueda o sistemas de gestión de documentos. Los índices de texto completo permiten que las consultas de búsqueda se ejecuten de manera rápida y eficiente, lo que mejora la experiencia del usuario y reduce la carga en el servidor de la base de datos.

## 4. Índice espacial

Un índice espacial es un tipo de índice diseñado para mejorar el rendimiento de las consultas que involucran datos geográficos o espaciales. Este tipo de índice es común en

aplicaciones de sistemas de información geográfica (GIS) y en aplicaciones que requieren búsquedas basadas en la ubicación. Los índices espaciales permiten que las consultas espaciales, como la búsqueda de puntos dentro de un área determinada o la búsqueda de la distancia entre dos puntos, se realicen de manera rápida y eficiente.

## 5. Índice de mapa de bits

Un índice de mapa de bits es un tipo de índice que utiliza una estructura de datos compacta, llamada mapa de bits, para representar la información sobre la presencia o ausencia de valores específicos en una columna. Los índices de mapa de bits son especialmente útiles en columnas con un número limitado de valores distintos, como columnas booleanas o de categoría. Este tipo de índice puede mejorar significativamente el rendimiento de las consultas que involucran filtros o agregaciones en columnas con baja cardinalidad.

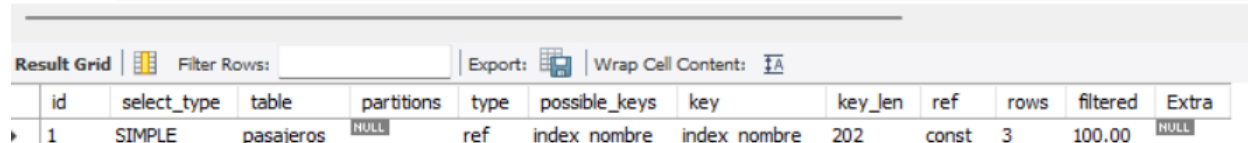
Los índices son una herramienta esencial para optimizar el rendimiento de las aplicaciones de bases de datos y garantizar una rápida recuperación de la información. Al elegir el tipo de índice adecuado para cada caso de uso, los desarrolladores pueden reducir el tiempo de las consultas y mejorar la experiencia del usuario. Además, el uso de índices también puede mejorar la integridad y consistencia de los datos en la base de datos, lo que es crucial para el éxito de cualquier proyecto de desarrollo de software.

**Importancia del Conocimiento:** Los índices son cruciales para acelerar las consultas y mejorar el rendimiento general de la base de datos. Dándonos una respuesta casi instantánea por lo cual debemos tener definidos los índices

## 2. Optimizar consultas SQL.

**Práctica:** Utilizar herramientas como EXPLAIN para identificar cuellos de botella en las consultas y optimizarlas.

```
14  -- EXPLAIN
15  • explain select*from pasajeros where nombre='Adrian';
```



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	pasajeros	NULL	ref	index_nombre	index_nombre	202	const	3	100.00	NULL

Imagen4.4 utilizando el explain

Fuente:(propia)

```
16 • explain analyze select * from pasajeros where apellido='Garcia';
17
18
```

Result Grid | Filter Rows: | Exports | Wrap Cell Content: |

EXPLAIN

► -> Filter: (pasajeros.apellido = 'Garcia') (cost=2.55 rows=2.3) (actual time=0.402..0.424 rows=1 loops=1) -> Table scan on pasajeros (cost=2.55 rows=23) (actual time=0.37..0.412 rows=22 loo...

Imagen4.5 utilizando el explain analyze  
Fuente:(propia)

**Investigación:** Investigar cómo hacer uso eficiente de las uniones (JOIN), subconsultas, y optimizar las consultas complejas.

**JOIN:** es una operación que permite combinar datos de dos o más tablas basándose en una relación entre columnas comunes

Existen diferentes tipos de JOIN, cada uno con un comportamiento específico:

- **INNER JOIN:** Devuelve solo las filas para las cuales se cumple la condición de unión en ambas tablas. En el ejemplo anterior, solo mostraría los clientes que tienen pedidos y los pedidos que fueron realizados por clientes.
- **LEFT JOIN:** Devuelve todas las filas de la tabla izquierda (la primera mencionada en el JOIN) y las filas coincidentes de la tabla derecha. Si un cliente no tiene pedidos, se mostrará igualmente en el resultado, pero las columnas correspondientes a los pedidos tendrán valores nulos.
- **RIGHT JOIN:** Similar al anterior, pero devuelve todas las filas de la tabla derecha y las coincidentes de la izquierda.
- **FULL OUTER JOIN:** Devuelve todas las filas de ambas tablas. Si no hay coincidencia en alguna de ellas, las columnas de la tabla sin coincidencia tendrán valores nulos.

**Subconsultas:** es una consulta dentro de otra consulta. Se utilizan para obtener un conjunto de resultados que se utiliza en la consulta principal.

- Tipos de subconsultas:\*
- Escalares: Devuelven un solo valor.
- De fila: Devuelven una o varias columnas, pero solo una fila.
- De tabla: Devuelven una o varias filas y columnas.

**Importancia del Conocimiento:** Las consultas optimizadas aseguran un sistema rápido y eficiente, especialmente en sistemas con alta demanda. Con las unión y las subconsultas podemos mejorar la respuesta de una búsqueda

### 3. Utilizar particionamiento de tablas.

**Práctica:** Dividir tablas grandes, como Reservas, en particiones según una clave (por ejemplo, por fecha).

```
alter table pasajeros partition by range (id) (  
    partition particion_1 values less than (5),  
    partition particion_2 values less than (10),  
    partition particion_3 values less than (15),  
    partition particion_4 values less than (21)  
);
```

Imagen4.6 creando la partición en la tabla pasajeros

Fuente:(propia)

Result Grid

Filter Rows:

</

Imagen.4.7. verificación de la partición\_1

Fuente:(propia)

**Investigación:** Investigar sobre los beneficios del particionamiento y cómo implementarlo en sistemas de bases de datos grandes.

El particionamiento de tablas es un esquema de organización de datos en el que los datos de una tabla se dividen entre varios objetos de almacenamiento denominados *particiones de datos* en función de los valores de una o varias columnas de la tabla. Cada partición de datos se almacena por separado. Estos objetos de almacenamiento pueden estar en varios espacios de tablas, en el mismo espacio de tablas o en una combinación de ambos.

Los objetos de almacenamiento se comportan de forma muy parecida a las tablas individuales, lo que facilita la realización de un roll-in rápido mediante la incorporación de una tabla existente en una tabla particionada utilizando el comando ALTER TABLE ... SENTENCIA ATTACH. Del mismo modo, se consigue un despliegue sencillo con la función ALTER TABLE ... DETACH. El procesamiento de consultas también puede aprovechar la separación de los datos para evitar el escaneo de datos irrelevantes, lo que se traduce en un mejor rendimiento de las consultas para muchas consultas de estilo almacén de datos.

Los datos de la tabla se particionan tal y como se especifica en la cláusula PARTITION BY de la sentencia CREATE TABLE. Las columnas utilizadas en esta definición se denominan columnas clave de partición de la tabla.

Este esquema de organización puede utilizarse de forma aislada o en combinación con otros esquemas de organización. Combinando las cláusulas DISTRIBUTE BY y PARTITION BY de la sentencia CREATE TABLE, los datos se pueden distribuir en particiones de bases de datos que abarcan varios espacios de tablas. Los esquemas de organización incluyen:

- DISTRIBUIR POR HASH
- PARTITION BY RANGE
- ORGANIZE BY DIMENSIONS

**Importancia del Conocimiento:** El particionamiento de tablas mejora la escalabilidad y el rendimiento en bases de datos con gran volumen de datos.

## 5. Procedimientos Almacenados, Vistas y Triggers

**Objetivo:** Mejorar la eficiencia y automatizar tareas mediante el uso de procedimientos almacenados, vistas y triggers.

**Actividades:**

### 1. Crear procedimientos almacenados.

**Práctica:** Crear un procedimiento para calcular el precio total de una reserva, aplicando descuentos y cargos adicionales.

Procedimiento: actualizaremos el precio con una un iva al 65,15%

**ANTES**

Result Grid								
Filter Rows:								
Edit: Export/Import: Wrap Cell Content:								
	id	numero_reserva	fecha_compra	precio	estado_reserva_id	pasajero2_id	embargue_id	vuelo2_id
▶	1	3	2025-01-30 22:04:13	420.75	15	1	1	7
	2	4	2025-01-30 22:04:13	400.75	15	20	2	6
	3	2	2025-01-30 22:04:13	420.75	15	19	3	5
	4	1	2025-01-30 22:04:13	200.03	16	18	4	4
	5	6	2025-01-30 22:04:13	850.00	17	17	5	3
	6	5	2025-01-30 22:04:13	300.89	16	16	6	2
	7	3	2025-01-30 22:04:13	978.35	15	15	7	1
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Imagen.5.1. tabla reservas antes de actualizar precio

Fuente:(propia)

```

31
32 delimiter //
33 • create procedure act_precio(in vuelos_id int, in porcentaje decimal(10,2))
34 begin
35     update reservas
36     set precio=round(precio*(1 + porcentaje/100),2)
37     where vuelo2_id=vuelos_id;
38 end //
39 delimiter ;
40 • -- llamando al procedimiento
41 call act_precio(1,65.15);
42

```

Output :

#	Time	Action	Message
✓ 22	10:11:56	drop table servidor	0 row(s) affected
✓ 23	12:21:43	select * from reservas LIMIT 0, 1000	7 row(s) returned
✓ 24	12:33:07	create procedure act_precio(in vuelos_id int, in porcentaje decimal(10,2)) begin update reservas set p...	0 row(s) affected
✓ 25	12:34:01	call act_precio(1,65.15)	1 row(s) affected

Imagen.5.2. creacion y ejecución del procedimiento  
Fuente:(propia)

## DESPUES

32 delimiter //

Result Grid

	id	numero_reserva	fecha_compra	precio	estado_reserva_id	pasajero2_id	embargue_id	vuelo2_id
	1	3	2025-01-30 22:04:13	420.75	15	1	1	7
	2	4	2025-01-30 22:04:13	400.75	15	20	2	6
	3	2	2025-01-30 22:04:13	420.75	15	19	3	5
	4	1	2025-01-30 22:04:13	200.03	16	18	4	4
	5	6	2025-01-30 22:04:13	850.00	17	17	5	3
	6	5	2025-01-30 22:04:13	300.89	16	16	6	2
▶	7	3	2025-01-30 22:04:13	1615.75	15	15	7	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Imagen.5.3. verificación del procedimiento realizado  
Fuente:(propia)

**Investigación:** Explorar cómo los procedimientos almacenados pueden mejorar la reutilización de código y la eficiencia.

## ¿Qué son los Procedimientos Almacenados?

Un procedimiento almacenado es un conjunto de una o más sentencias SQL que se almacenan en la base de datos. Una vez creado, se puede ejecutar repetidamente con diferentes parámetros, lo que lo convierte en una herramienta poderosa para realizar tareas complejas de manera eficiente.

## Reutilización de Código

- **Encapsulación de lógica:** Los procedimientos almacenados permiten encapsular lógica de negocio compleja en un solo lugar. En lugar de escribir el mismo código SQL repetidamente en diferentes partes de una aplicación, puedes llamar al procedimiento almacenado, lo que reduce la redundancia y facilita el mantenimiento del código.
- **Modularidad:** Los procedimientos almacenados fomentan la modularidad del código. Puedes crear procedimientos más pequeños y específicos que realizan tareas individuales y luego combinarlos para crear procedimientos más grandes y complejos. Esto mejora la organización y legibilidad del código.
- **Abstracción:** Los procedimientos almacenados actúan como una capa de abstracción entre la aplicación y la base de datos. Los desarrolladores de aplicaciones no necesitan conocer los detalles de implementación de las consultas SQL; simplemente llaman al procedimiento almacenado con los parámetros adecuados.

## **Eficiencia**

- **Reducción del tráfico de red:** Cuando se llama a un procedimiento almacenado, solo se envía la llamada y los parámetros a la base de datos, en lugar de enviar todo el código SQL. Esto reduce la cantidad de datos que se transmiten a través de la red, lo que mejora el rendimiento, especialmente en aplicaciones con mucho tráfico.
- **Precompilación y optimización:** Los procedimientos almacenados se compilan y optimizan en la base de datos la primera vez que se ejecutan. Luego, se almacenan en caché para su reutilización, lo que acelera las ejecuciones posteriores.
- **Reducción de la carga del servidor de aplicaciones:** Al ejecutar la lógica de negocio en la base de datos, se reduce la carga del servidor de aplicaciones, lo que permite que este se centre en otras tareas.

**Importancia del Conocimiento:** Los procedimientos almacenados centralizan la lógica y pueden mejorar el rendimiento al ejecutarse directamente en el servidor.



## 2. Crear vistas para simplificar consultas complejas.

**Práctica:** Crear vistas que presenten información de varias tablas de manera unificada (por ejemplo, una vista que combine datos de Vuelos, Clientes y Reservas).

```
46 • alter view informacion as
47 select ae.nombre as aerolinea,ae.codigo,
48 p.nombre as pais_origen, ae.frecuencia_vuelo,
49 a.nombre as aeropuerto,av.modelo,av.capacidad,
50 ic.nombre as estado
51 from aerolineas as ae
52 inner join paises as p on p.id=ae.pais1_id
53 inner join aeropuertos a on a.id=ae.aeropuerto_id
54 inner join aviones as av on ae.id=av.id
55 inner join item_catalogos as ic on ic.id=estado_avion_id;
56 • select*from informacion;
```

aerolinea	codigo	pais_origen	frecuencia_vuelo	aeropuerto	modelo	capacidad	estado
Tame	ECU789E-0	Ecuador	diario	aeropuerto internacional Mariscal Sucre	Airbus A320	220	mantenimiento
VivaAerobus	MEX789X-0	Mexico	diario	aeropuerto internacional Ciudad de Mexico	Boeing 737	600	fuera de servicio
Air France	LHR233R-0	Francia	semanal	Aeropuerto de Heathrow	Boeing 777	550	activo
AirAsia	PKX569X-0	China	semanal	Aeropuerto Internacional de Pekín-Daxing	Airbus A350	350	activo
Asiana Airlines	ICN000N-0	Corea del Sur	semanal	Aeropuerto Internacional de Incheon	Boeing 787 Dreamliner	400	mantenimiento

Imagen5.4. creación de vistas-aviones

Fuente:( propia)

```
61 • alter view informacion_persona as
62 select concat(p.nombre,' ',p.apellido) as identificacion,p.correo,p.telefono,e.numero_puerta,e.terminal,
63 ic.nombre as tipo_asiento,r.precio,r.fecha_compra,
64 v.distancia_recorrida,
65 aerol.nombre as aerolinea,ae.
66 nombre as aeropuerto
67 from reservas as r
68 inner join pasajeros as p on p.id=r.pasajero2_id
69 inner join embargues as e on e.id=r.embargue_id
70 inner join item_catalogos as ic on ic.id= estado_reserva_id
71 inner join vuelos as v on v.id=vuelo2_id
72 inner join aeropuertos as ae on ae.id=v.aeropuerto_destino
73 inner join aerolineas as aerol on ae.id=aerol.aeropuerto_id;
74
```

identificacion	correo	telefono	numero_puerta	terminal	tipo_asiento	precio	fecha_compra	distancia_recorrida	aerolinea	aero
Miguel Astudillo	miguel-astudillo@yahoo.es	0987754315	17	salida	business	300.89	2025-01-30 22:04:13	18789.03	Tame	aero
Sofia Cabrera	ithancamacho123@gmail.com	0988654319	18	llegada	primera clase	400.75	2025-01-30 22:04:13	74136.56	Tame	aero
Adrian Lopez	adrianlopez963@hotmail.com.com	0987654317	18	salida	business	200.03	2025-01-30 22:04:13	21649.60	VivaAerobus	aero
Adrian Cadena	adriancadena963@yahoo.es	0988654318	17	salida	primera clase	420.75	2025-01-30 22:04:13	19517.02	Air France	Aero
Mateo Torres	mateotorres@gmail.com	0952647894	17	salida	primera clase	420.75	2025-01-30 22:04:13	35795.35795.15	AirAsia	Aero
Minuel Lopez	minuel-lopez123@gmail.com	0987654316	17	llegada	economica	850.00	2025-01-30 22:04:13	26648.35	Asiana Airlines	Aero

Imagen5.5. creación de vistas-personas

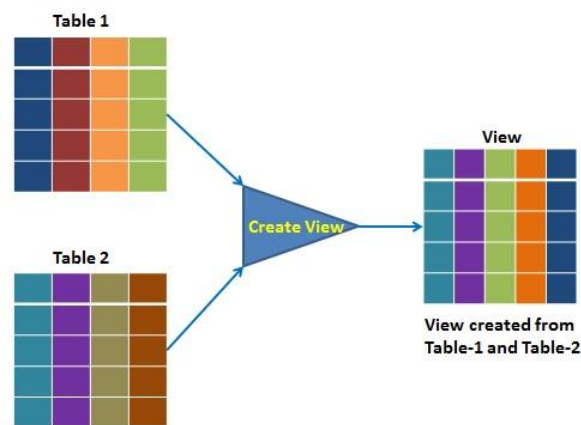
Fuente:( propia)

**Investigación:** Investigar las ventajas de usar vistas en lugar de consultas complejas repetitivas.

La vista es una consulta almacenada en el diccionario de datos, sobre la que el usuario puede consultar igual que sobre las tablas. No utiliza la memoria física, sólo se almacena la consulta en el diccionario de datos. Se calcula dinámicamente, cada vez que el usuario realiza alguna consulta sobre él. Los cambios realizados en cualquier punto de la vista se reflejan en la tabla base real.

La vista tiene principalmente dos finalidades:

- Simplifica las consultas SQL complejas.
- Restringe el acceso de los usuarios a datos sensibles.



*Imagen5.6 vista de tablas*

Fuente:( <https://www.datacamp.com/es/tutorial/views-in-sql>)

Las vistas en MySQL son una herramienta poderosa que ofrece una serie de ventajas para la gestión y manipulación de datos. Aquí te presento algunas de las principales ventajas de utilizarlas:

### **1. Simplificación de consultas complejas:**

- Las vistas permiten encapsular consultas SQL complejas en una sola entidad, lo que facilita la escritura y comprensión de consultas posteriores. En lugar de tener que repetir una consulta larga y complicada cada vez que necesitas acceder a los datos, puedes simplemente consultar la vista.

### **2. Abstracción de la estructura de la base de datos:**

- Las vistas pueden ocultar la complejidad de la estructura de la base de datos a los usuarios. Esto es especialmente útil cuando se trabaja con bases de datos grandes y complejas, ya que permite a los usuarios ver los datos de una manera más sencilla y organizada.

### **3. Control de acceso a los datos:**

- Las vistas se pueden utilizar para restringir el acceso a ciertas columnas o filas de una tabla. Esto permite a los administradores de la base de datos controlar qué información pueden ver los diferentes usuarios, lo que mejora la seguridad y la privacidad de los datos.

#### 4. Mejora del rendimiento:

- En algunos casos, las vistas pueden mejorar el rendimiento de las consultas. Esto se debe a que la base de datos puede optimizar la ejecución de la consulta almacenada en la vista.

#### 5. Reutilización de código:

- Las vistas se pueden reutilizar en diferentes consultas y aplicaciones, lo que reduce la necesidad de escribir el mismo código SQL varias veces. Esto ahorra tiempo y esfuerzo, y también facilita el mantenimiento del código.

#### 6. Mantenimiento de la integridad de los datos:

- Al utilizar vistas, se puede asegurar que las consultas siempre accedan a los datos correctos, incluso si la estructura de la base de datos cambia. Esto ayuda a mantener la integridad de los datos y evita errores en las consultas.

**Importancia del Conocimiento:** Las vistas ayudan a simplificar el acceso a datos complejos y pueden mejorar la seguridad al limitar el acceso directo a las tablas.

### 3. Implementar triggers para auditoría y control de cambios.

**Práctica:** Crear triggers que registren cambios en las tablas de Reservas y Pagos cada vez que un registro se actualiza o elimina.

108 • `select*from pasajeros;`  
 109 • `select*from tripulantes;`

	id	cedula	nombre	apellido	edad	correo	telefono	equipaje1_id
	19	6000263486	Adrian	Cadena	20	adriancadena963@yahoo.es	0988654318	3
	20	8840211992	Sofia	Cabrera	19	ithancamacho123@gmail.com	0988654319	2
	21	7716619596	Miguel	Cadena	23	miguelcadena123@hotmail...	0988654320	1
	22	7415178323	Paul	Sanchez	44	paulSanchez123@yahoo.es	0987654393	2
	23	6196022548	Mateo	Torres	16	mateo123@gmail.com	0987654321	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

pasajeros 105 x

Imagen5.7. tabla pasajeros antes del trigger

Fuente:( propia)

```

• create table auditoria_pasajeros(
  id int auto_increment primary key,
  cedula varchar(10) ,
  nombre varchar(50) ,
  apellido varchar(50) ,
  edad int ,
  correo varchar(100) ,
  telefono varchar(10) ,
  equipaje1_id int,
  fecha_eliminacion timestamp default current_timestamp
);

delimiter $$
• create trigger before_delete_pasajeros
  before delete on pasajeros
  for each row
  begin
    insert into auditoria_pasajeros(cedula,nombre,apellido,edad,correo,telefono,equipaje1_id)
    values(old.cedula,old.nombre,old.apellido,old.edad,old.correo,old.telefono,old.equipaje1_id);
  end $$
delimiter ;

```

Imagen5.8. creación de la tabla auditoria\_pasajeros y trigger

Fuente:( propia)

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell C

	id	cedula	nombre	apellido	edad	correo	telefono	equipaje1_id
	14	3342166849	Juan	Cabrera	27	juan123@gmail.com	0987754313	3
	15	1715353271	Lucia	Gonzalez	54	Lucia123@hotmail.com	0987754314	1
	16	0052717064	Miguel	Astudillo	18	miguel-astudillo@yahoo.es	0987754315	2
	17	5447931325	Miguel	Lopez	23	miguel-lopez123@gmail.com	0987654316	1
	18	1372137650	Adrian	Lopez	54	adrianlopez2963@hotmail.c...	0987654317	3
	19	6000263486	Adrian	Cadena	20	adriancadena963@yahoo.es	0988654318	3
	20	8840211992	Sofia	Cabrera	19	ithancamacho123@gmail.com	0988654319	2
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Imagen5.9. tabla pasajeros después del trigger

Fuente:( propia)

```

125 • select*from auditoria_pasajeros;
126

```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	cedula	nombre	apellido	edad	correo	telefono	equipaje1_id	fecha_eliminacion
	1	7716619596	Miguel	Cadena	23	miguelcadena123@hotmail.com	0988654320	1	2025-02-04 18:08:25
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Imagen5.10. tabla auditoria\_pasajeros y la acción realizada

Fuente:( propia)

**Investigación:** Investigar cómo utilizar triggers para mantener un historial de cambios en la base de datos.

Los **triggers** o **disparadores** son herramientas poderosas en bases de datos que permiten automatizar tareas y mantener la integridad de los datos. Se ejecutan automáticamente en respuesta a eventos específicos, como inserciones, actualizaciones o eliminaciones en una tabla.

### ¿Cómo funcionan los triggers?

1. **Definición del trigger:** Se define un bloque de código SQL que se ejecutará cuando ocurra un evento específico en una tabla.
2. **Evento desencadenante:** Se especifica el evento que activará el trigger, como una inserción (INSERT), actualización (UPDATE) o eliminación (DELETE) en una tabla.
3. **Acción del trigger:** Se define la acción que se realizará cuando se active el trigger, como insertar un registro en una tabla de auditoría, actualizar otra tabla o enviar una notificación.

### Tipos de triggers

- **Según el momento de ejecución:**
  - BEFORE: Se ejecutan antes de que se realice la operación que desencadena el trigger.
  - AFTER: Se ejecutan después de que se haya completado la operación.
- **Según el evento que los activa:**
  - INSERT: Se activan al insertar un nuevo registro.
  - UPDATE: Se activan al actualizar datos.
  - DELETE: Se activan al eliminar registros.

Para que todos estos datos esten cambiados se deberá crear un tabla donde se guardara la información o los datos que se hizo en dicha tabla. Despues se ejecutara el trigger haciendo relación con la tabla y posterior a eso se verificara la nueva tabla con la tabla original.

**Importancia del Conocimiento:** Los triggers permiten automatizar tareas como la auditoría y validación de datos.

## 6. Monitoreo y Optimización de Recursos

**Objetivo:** Controlar el rendimiento de la base de datos, identificando y solucionando problemas de recursos.

### Actividades:

#### 1. Monitorear el rendimiento de consultas.

**Práctica:** Usar herramientas como SHOW PROCESSLIST para detectar consultas lentas y optimizarlas.

130 • SHOW PROCESSLIST;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Id	User	Host	db	Command	Time	State	Info
▶	5	event_scheduler	localhost	NULL	Daemon	411926	Waiting on empty queue	NULL
	185	root	localhost:60049	NULL	Sleep	56		NULL
	186	root	localhost:60050	aerolineas	Sleep	56		NULL
	188	Aerolineas	localhost:60064	NULL	Sleep	29		NULL
	189	Aerolineas	localhost:60065	aerolineas	Query	0	init	SHOW PROCESSLIST

Imagen6.1. ejecución del show processlist

Fuente:( propia)

131 • SHOW FULL PROCESSLIST;

132

Result Grid

Filter Rows:

Export:

Wrap Cell Content: [FA](#)

	Id	User	Host	db	Command	Time	State	Info
▶	5	event_scheduler	localhost	NULL	Daemon	412305	Waiting on empty queue	NULL
	185	root	localhost:60049	NULL	Sleep	435		NULL
	186	root	localhost:60050	aerolineas	Sleep	435		NULL
	188	Aerolineas	localhost:60064	NULL	Sleep	136		NULL
	189	Aerolineas	localhost:60065	aerolineas	Query	0	init	SHOW FULL PROCESSLIST

Imagen6.2. ejecución del show full processlist

Fuente:( propia)

**Investigación:** Investigar las mejores prácticas para monitorear el rendimiento de las consultas en producción.

- **Herramientas de monitoreo:** Implementa herramientas de monitoreo de bases de datos (como SolarWinds DPA, ManageEngine Applications Manager, o Performance Insights de AWS) que te permitan visualizar el rendimiento de las consultas en tiempo real, identificar cuellos de botella y recibir alertas sobre posibles problemas.
- **Métricas clave:** Define métricas clave de rendimiento (KPIs) para tus consultas, como tiempo de ejecución promedio, tiempo de ejecución máximo, uso de recursos (CPU, memoria, E/S), y frecuencia de ejecución.

- **Alertas:** Configura alertas para recibir notificaciones cuando las métricas clave excedan los umbrales aceptables. Esto te permitirá tomar medidas preventivas antes de que los problemas afecten a los usuarios.

## 2. Análisis de consultas:

- **Identificación de consultas lentas:** Utiliza las herramientas de monitoreo para identificar las consultas que consumen más recursos o tienen tiempos de ejecución elevados.
- **Planes de ejecución:** Examina los planes de ejecución de las consultas lentas para entender cómo el optimizador de la base de datos está ejecutando la consulta. Esto te ayudará a identificar posibles problemas, como índices faltantes o uniones ineficientes.
- **Profiler:** Utiliza herramientas de profiling (como el Profiler de SQL Server o el Explain Plan de MySQL) para obtener información detallada sobre el rendimiento de las consultas, incluyendo el tiempo dedicado a cada operación y los recursos consumidos.

## 3. Optimización de consultas:

- **Índices:** Asegúrate de tener los índices adecuados en las columnas que se utilizan en las cláusulas WHERE, JOIN y ORDER BY de tus consultas. Los índices pueden acelerar significativamente la búsqueda de datos.
- **Reescritura de consultas:** A veces, reescribir una consulta de manera más eficiente puede mejorar su rendimiento. Por ejemplo, evita usar funciones en las cláusulas WHERE, ya que esto puede impedir que se utilicen los índices.
- **Optimización de tipos de datos:** Utiliza los tipos de datos más pequeños posibles para tus columnas. Esto puede reducir el espacio de almacenamiento y mejorar el rendimiento de las consultas.
- **Particionamiento de tablas:** Si tienes tablas muy grandes, considera particionarlas en tablas más pequeñas. Esto puede mejorar el rendimiento de las consultas que acceden solo a una parte de los datos.

**Importancia del Conocimiento:** El monitoreo proactivo puede identificar cuellos de botella antes de que afecten el rendimiento del sistema.



## 2. Realizar pruebas de carga.

**Práctica:** Simular múltiples usuarios concurrentes usando herramientas como Apache JMeter para ver cómo responde la base de datos bajo alta carga.

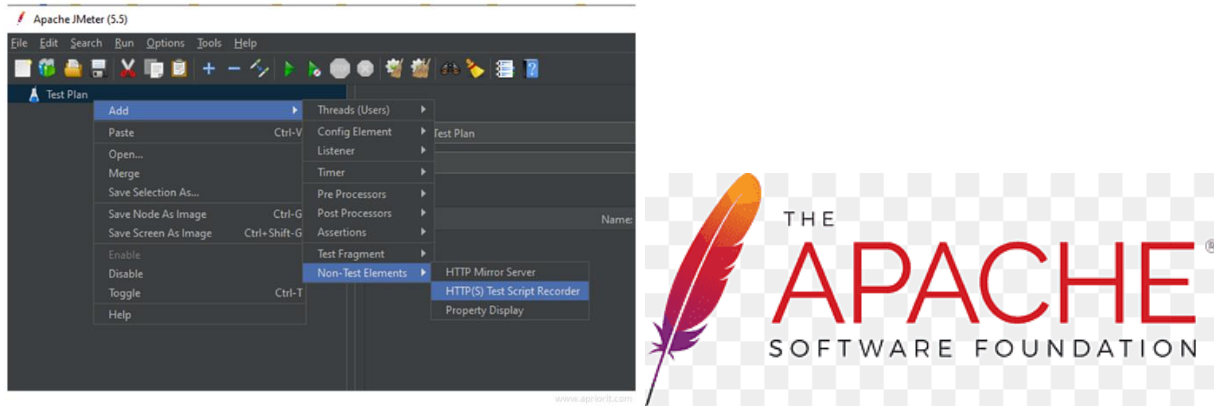


Imagen6.2. cargas con Apache JMeter  
Fuente:( propia)

**Investigación:** Investigar cómo realizar pruebas de estrés y carga en bases de datos de alto rendimiento.  
cómo realizar pruebas de estrés y carga en bases de datos de alto rendimiento.

Realizar pruebas de estrés y carga en bases de datos de alto rendimiento es crucial para asegurar que tu sistema pueda manejar grandes volúmenes de datos y tráfico sin comprometer el rendimiento. Aquí te presento una guía detallada sobre cómo llevar a cabo estas pruebas:

### 1. Define tus objetivos:

- Antes de comenzar, es fundamental que tengas claros tus objetivos. ¿Qué quieres lograr con estas pruebas? ¿Quieres identificar cuellos de botella, determinar el punto de quiebre de tu base de datos, o simplemente asegurarte de que cumple con los requisitos de rendimiento?

### 2. Elige las herramientas adecuadas:

- Existen diversas herramientas que te pueden ayudar a realizar pruebas de estrés y carga en bases de datos. Algunas opciones populares incluyen:
  - **JMeter:** Una herramienta de código abierto muy utilizada para pruebas de carga y estrés.
  - **LoadRunner:** Una herramienta comercial con una amplia gama de funcionalidades para pruebas de rendimiento.
  - **HammerDB:** Una herramienta de código abierto diseñada específicamente para pruebas de bases de datos.



- **SQL Server Performance Studio:** Una herramienta integrada en SQL Server para monitorear y analizar el rendimiento.
- **Oracle Performance Monitoring:** Una herramienta similar para bases de datos Oracle.

### 3. Configura tu entorno de pruebas:

- Es importante que tu entorno de pruebas sea lo más similar posible al entorno de producción. Esto incluye la configuración del hardware, el software y la red.

### 4. Diseña tus escenarios de prueba:

- Define los escenarios de prueba que simulen el uso real de tu base de datos. Esto incluye la creación de usuarios virtuales, la definición de las consultas que ejecutarán y la configuración de la carga que generarán.

### 5. Ejecuta las pruebas:

- Una vez que hayas configurado tu entorno y diseñado tus escenarios, es hora de ejecutar las pruebas. Monitorea de cerca el rendimiento de tu base de datos durante las pruebas para identificar posibles problemas.

### 6. Analiza los resultados:

- Después de ejecutar las pruebas, analiza los resultados para identificar cuellos de botella, problemas de rendimiento y áreas de mejora. Utiliza las herramientas de monitoreo y análisis para obtener información detallada sobre el rendimiento de tu base de datos.

### 7. Realiza ajustes y optimizaciones:

- Basándote en los resultados de las pruebas, realiza ajustes y optimizaciones en tu base de datos para mejorar su rendimiento. Esto puede incluir la optimización de consultas, la configuración de índices, el ajuste de parámetros de configuración, etc.

**Importancia del Conocimiento:** Las pruebas de carga aseguran que el sistema sea capaz de manejar tráfico alto y crecimiento de datos.

### 3. Optimizar el uso de recursos y gestionar índices.

**Práctica:** Identificar índices no utilizados y eliminarlos para liberar recursos y mejorar la velocidad de las operaciones de escritura.

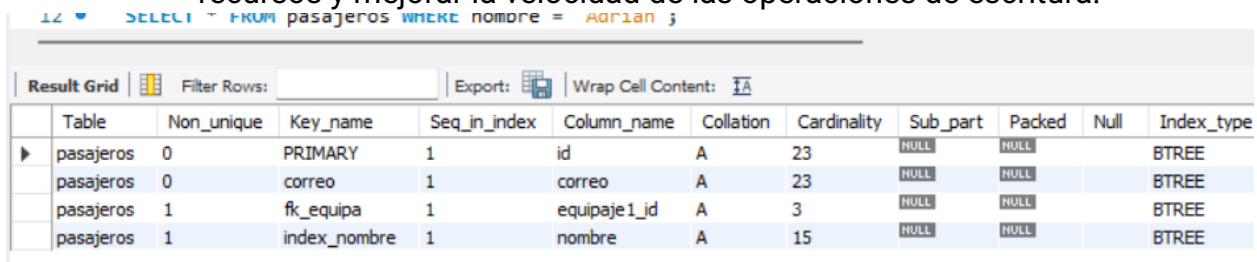


	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
▶	pasajeros	0	PRIMARY	1	id	A	23	NULL	NULL		BTREE
	pasajeros	0	correo	1	correo	A	23	NULL	NULL		BTREE
	pasajeros	1	fk_equipa	1	equipaje1_id	A	3	NULL	NULL		BTREE
	pasajeros	1	index_nombre	1	nombre	A	15	NULL	NULL		BTREE

Imagen6.2. visualización de los index

Fuente:( propia)

```
16 -- para eliminar los indices
17 DROP INDEX index_nombre ON pasajeros;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
pasajeros	0	PRIMARY	1	id	A	23	NULL	NULL		BTREE
pasajeros	0	correo	1	correo	A	23	NULL	NULL		BTREE
pasajeros	1	fk_equipa	1	equipaje1_id	A	3	NULL	NULL		BTREE

*Imagen6.3. tabla después de haber ejecutado el drop*  
*Fuente:( propia)*

**Investigación:** Investigar cómo ajustar el número de índices según el tipo de consulta (lectura/escritura).  
 cómo ajustar el número de índices según el tipo de consulta (lectura/escritura).

Las consultas de índice secundario (SI) utilizan índices secundarios para acelerar las consultas, en comparación con las consultas de índice primario (PI). Si bien se trata de una compensación entre la memoria y el rendimiento, una consulta SI generalmente funciona mucho mejor que una consulta PI en un conjunto grande. Cuando tiene dos índices en los mismos datos (conjunto o espacio de nombres), el índice de cardinalidad más alto brinda mejores resultados.

Antes de Aerospike Database 6.0, las consultas de índice primario (PI) se denominaban "scans" y las consultas de índice secundario (SI) se denominaban "consultas". Consulte la guía de funciones de Consultas .

A partir de la base de datos 6.0, las consultas PI y SI son atendidas por el mismo subsistema y comparten las mismas políticas y funciones. La consulta SI es la misma que la consulta PI , con un filtro de índice adicional: un predicado simple dirigido al índice secundario. La consulta SI puede entonces aplicar una expresión de filtro a los registros que coincidan primero con el predicado de filtro de índice.

**Importancia del Conocimiento:** La optimización de los recursos asegura un uso eficiente del hardware y mejora la escalabilidad.

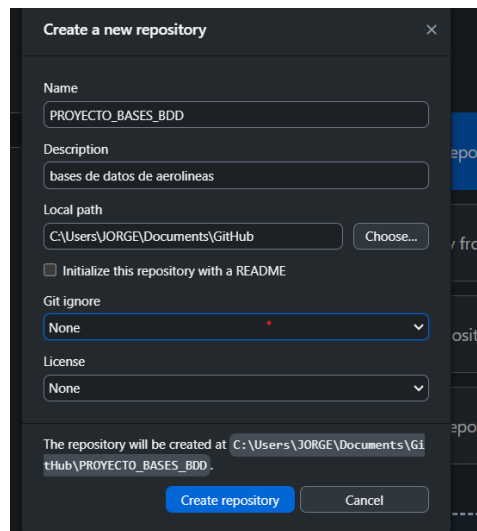
## 7. Git y Control de Versiones

**Objetivo:** Asegurar que el código relacionado con la base de datos esté versionado y que el equipo pueda colaborar de manera eficiente.

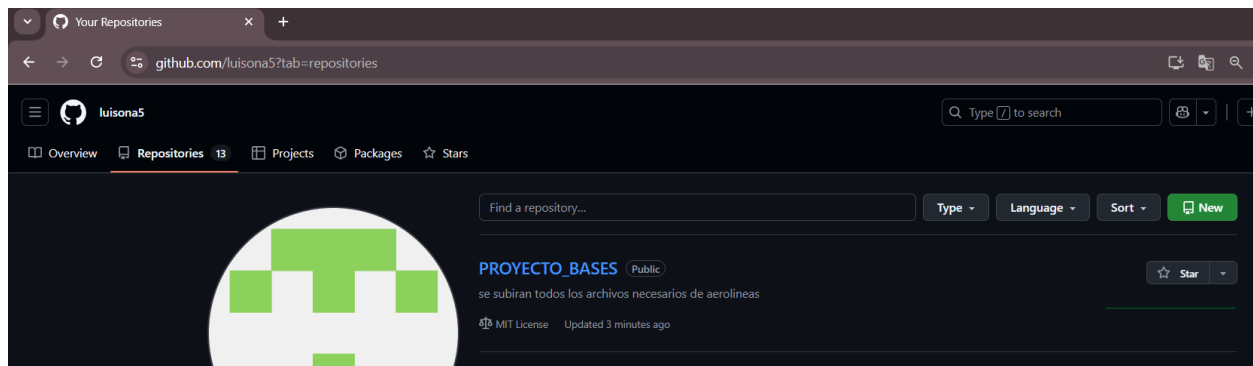
**Actividades:**

1. **Configurar un repositorio de Git para el proyecto.**

**Práctica:** Inicializar un repositorio en Git y subir los archivos de definición de la base de datos, scripts de SQL y procedimientos almacenados.



*Imagen.7.1. creación del repositorio  
Fuente:(propia)*



*Imagen.7.2. verificación en github del repositorio  
Fuente:(propia)*

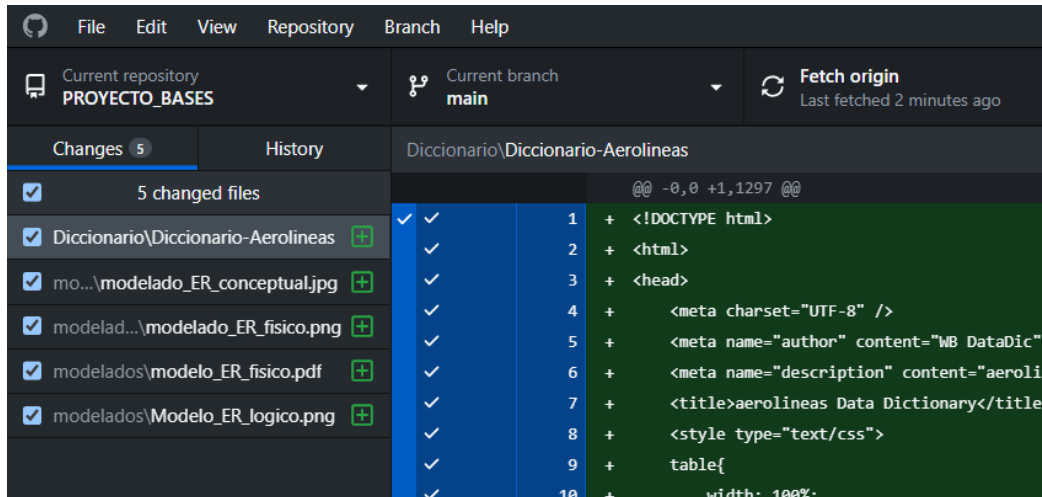


Imagen.7.3. introduciendo archivos al git  
Fuente:(propia)



Imagen.7.4. verificación en github  
Fuente:(propia)

**Investigación:** Investigar buenas prácticas de flujo de trabajo en Git (por ejemplo, uso de ramas, git merge).  
buenas prácticas de flujo de trabajo en Git (por ejemplo, uso de ramas, git merge).

Dominar el flujo de trabajo en Git es esencial para cualquier desarrollador que busque colaborar de manera efectiva y mantener un código base limpio y organizado. Aquí te presento algunas de las mejores prácticas, centrándonos en el uso de ramas y git merge:

## 1. Ramas para cada tarea

- Crea una rama para cada nueva funcionalidad, corrección de errores o tarea. Esto aísla los cambios y evita que afecten al código principal hasta que estén listos.

- Utiliza nombres descriptivos para las ramas. Esto facilita la comprensión del propósito de cada rama. Por ejemplo: feature/nueva-funcionalidad, bugfix/error-de-login, task/actualizacion-de-documentacion.

## 2. Flujo de trabajo con ramas de características

- Crea una rama a partir de la rama principal (main o master). Esta rama principal debe contener siempre el código estable y listo para producción.
- Desarrolla y realiza commits en la rama de características. Mantén los commits pequeños y descriptivos.
- Realiza pruebas exhaustivas en la rama de características. Asegúrate de que la nueva funcionalidad o corrección funciona correctamente y no introduce nuevos errores.

## 3. git merge para integrar cambios

- Cuando la rama de características esté lista, utiliza git merge para integrarla en la rama principal.
- Asegúrate de estar en la rama principal antes de ejecutar el merge.
- Resuelve los conflictos de merge si es necesario. Git te indicará si hay conflictos y te ayudará a resolverlos manualmente.

## 4. Pull Requests (PRs) para revisión de código

- Antes de realizar el merge, crea un Pull Request (PR) en tu plataforma Git (GitHub, GitLab, Bitbucket). Esto permite a otros miembros del equipo revisar el código y dar su aprobación antes de que se integre en la rama principal.
- Utiliza los PRs para discutir cambios, hacer preguntas y recibir feedback.

**Importancia del Conocimiento:** Git permite la colaboración y el manejo eficiente de cambios en el código, especialmente cuando se trabaja en equipo.

## 2. Realizar commits frecuentes y con mensajes claros.

**Práctica:** Hacer commits regularmente, describiendo claramente los cambios realizados en los scripts SQL y la estructura de la base de datos.

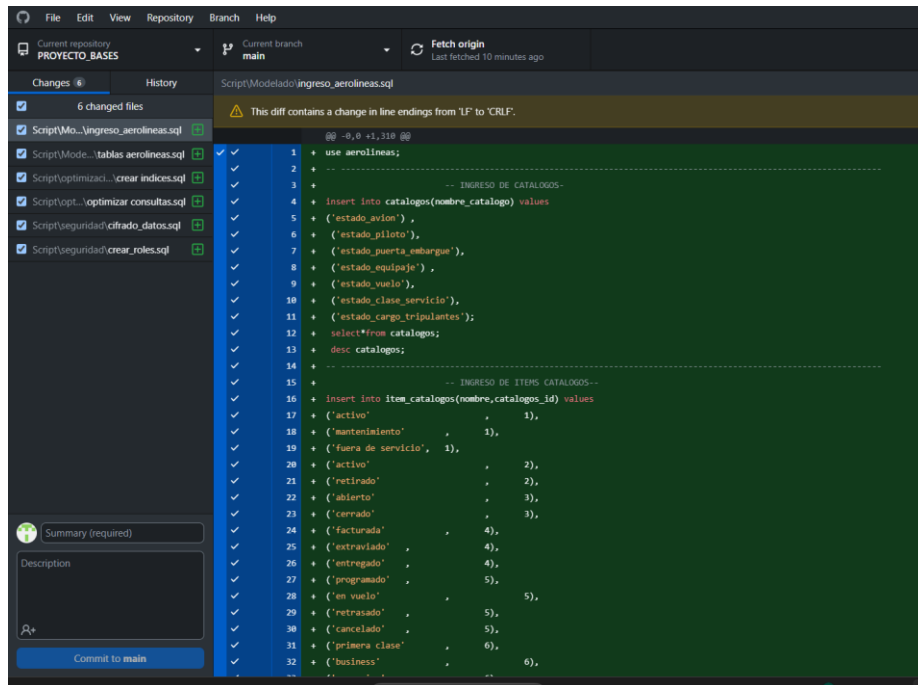


Imagen.7.5. haciendo commit and push  
Fuente:(propia)

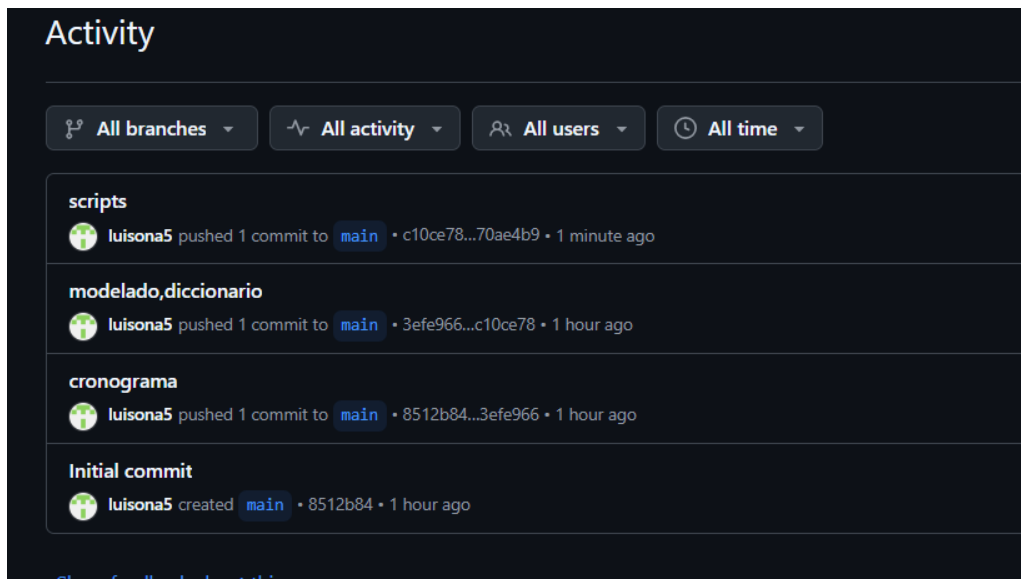


Imagen.7.6. verificación en github  
Fuente:(propia)

**Investigación:** Investigar cómo utilizar git rebase y git pull para evitar conflictos.

Antes de examinar las diferencias, definamos algunas operaciones clave:

- **git fetch** : este comando descarga confirmaciones, archivos y referencias de un repositorio remoto a su repositorio local. La obtención de cambios es una forma segura de revisar los cambios antes de integrarlos en su repositorio local.

- **git pull** : una git pull operación es básicamente una git fetch seguida de una git merge. Obtiene los cambios del repositorio remoto (la rama especificada) y luego los fusiona con la rama actual.
- **git rebase** : Rebase es una forma de mover o combinar una secuencia de confirmaciones en una nueva confirmación base. Git rebase a menudo se utiliza para mantener un historial de proyecto más limpio y lineal al mover toda la rama de características para que se ubique encima de la rama principal.

## Manejo de conflictos

- **Git pull** : si hay conflictos durante una extracción, Git pausa la fusión y te pide que los resuelvas manualmente. Después de resolver los conflictos, debes confirmar los cambios para completar la fusión.
- **Git rebase** : durante una rebase, si surgen conflictos, también es necesario resolverlos manualmente. Sin embargo, el proceso de rebase requiere que utilices `Git git rebase --continue` para continuar después de solucionar los conflictos. Este método garantiza que tu rama de funciones se pueda probar como si los cambios se hubieran realizado sobre la rama base ( main ).

**Importancia del Conocimiento:** Un flujo de trabajo claro en Git mejora la colaboración y la gestión de versiones.

### 3. Automatizar pruebas con GitHub Actions.

**Práctica:** Crear flujos de trabajo de CI/CD que automaticen las pruebas de las consultas SQL y otros scripts relacionados con la base de datos.

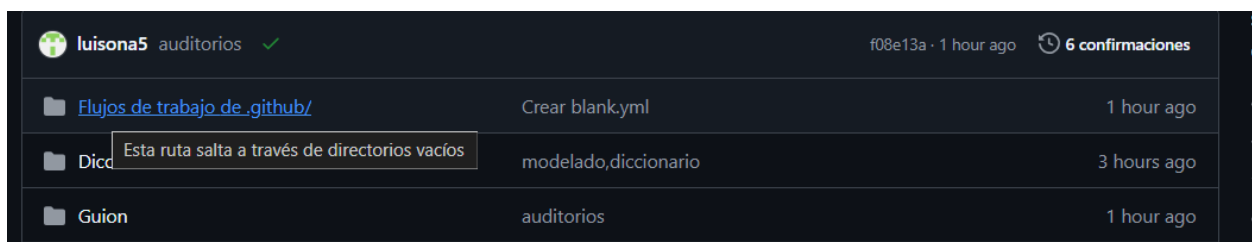


Imagen.7.6. verificación en github  
Fuente:(propia)

**Investigación:** Investigar sobre integración continua y cómo aplicarla en bases de datos con GitHub Actions.

## **Acerca de la integración continua**

La integración continua (CI) es una práctica de software que requiere la confirmación de código de forma periódica en un repositorio compartido. La confirmación de código con mayor frecuencia detecta errores más rápido y reduce la cantidad de código que un desarrollador necesita depurar al encontrar la fuente de un error. Las actualizaciones frecuentes de código facilitan también la fusión de cambios de diferentes miembros de un equipo de desarrollo de software. Esto es excelente para los desarrolladores, que pueden dedicar más tiempo a escribir el código y menos tiempo a depurar errores o resolver conflictos de fusión.

Al confirmar el código en tu repositorio, puedes crear y probar el código continuamente para asegurarte de que la confirmación no introduzca errores. Tus pruebas pueden incluir limpiadores de código (que verifican el formato de estilo), verificaciones de seguridad, cobertura de código, pruebas funcionales y otras verificaciones personalizadas.

Para crear y probar tu código es necesario un servidor. Puedes crear y probar las actualizaciones localmente antes de subir un código a un repositorio o puedes usar un servidor CI que verifique las nuevas confirmaciones de código en un repositorio.

### **Acerca de la integración continua utilizando GitHub Actions**

Tener una IC utilizando GitHub Actions ofrece flujos de trabajo que pueden compilar el código de tu repositorio y ejecutar tus pruebas. Los flujos de trabajo pueden ejecutarse en máquinas virtuales hospedadas en GitHub, o en máquinas que hospedes tú mismo. Para más información

Puede configurar el flujo de trabajo de CI para que se ejecute cuando se produzca un evento de GitHub (por ejemplo, cuando se inserta código nuevo en el repositorio), en una programación establecida o cuando se produce un evento externo mediante el webhook de envío de un repositorio.

GitHub ejecuta tus pruebas de CI y proporciona los resultados de cada prueba en la solicitud de cambios, de modo que puedas ver si el cambio en tu rama genera un error. Cuando se superan todas las pruebas de CI en un flujo de trabajo, los cambios que subiste están listos para su revisión por parte de un miembro del equipo o para su fusión. Cuando una prueba falla, es posible que uno de tus cambios haya causado la falla.

Al configurar la CI en tu repositorio, GitHub analiza el código en tu repositorio y recomienda flujos de trabajo de CI en función del lenguaje y el marco en tu repositorio. Por ejemplo, si usas NODE.JS, GitHub sugerirá una plantilla de flujo de trabajo que instala los paquetes de Node.js y ejecuta las pruebas. Puedes utilizar la plantilla de flujo de trabajo de CI que sugiere GitHub, personalizarla, o bien crear un archivo de flujo de trabajo personalizado propio para ejecutar las pruebas de CI.

Además de ayudarte a configurar flujos de trabajo de CI para tu proyecto, puedes usar GitHub Actions para crear flujos de trabajo durante todo el ciclo de vida de desarrollo de



software. Por ejemplo, puedes usar acciones para implementar, empaquetar o lanzar tu proyecto. Para más información,

**Importancia del Conocimiento:** Las pruebas automáticas aseguran que las bases de datos se mantengan consistentes y funcionales a lo largo del tiempo.

En este caso se debe tener en cuenta yml ya que es parte de de git y con ello podemos configurar para realizar