Wikicoa Project’s DataBase

ALMEIDA BENAVIDES ROMER ALEXANDER

DESARROLLO DE BASES DE DATOS

TECNOLOGÍA EN DESARROLLO DE SOFTWARE

QUINTO SEMESTRE

MOCOA PUTUMAYO

2024

Contenido

[Repositorio de GITHUB de Romerinjs: 3](#_Toc184403819)

[Introducción a las Bases de Datos Relacionales 4](#_Toc184403820)

[RESUMEN EJECUTIVO 4](#_Toc184403821)

[Introducción 5](#_Toc184403822)

[Metodología 6](#_Toc184403823)

[Herramientas utilizadas: 6](#_Toc184403824)

[Modelado de Datos Relacional 6](#_Toc184403825)

[Procedimientos 6](#_Toc184403826)

[1. Identificación de Requerimientos 6](#_Toc184403827)

[2. Diseño del Diagrama Entidad-Relación (ER) 6](#_Toc184403828)

[3. Proceso de Normalización 7](#_Toc184403829)

[4. Implementación de Tablas en SQL 7](#_Toc184403830)

[5. Inserción de Datos de Prueba 7](#_Toc184403831)

[6. Ejecución de Consultas SQL 7](#_Toc184403832)

[Operadores y Funciones en SQL\* 8](#_Toc184403833)

[Consultas SQL 8](#_Toc184403834)

[Descripción de la base de datos 15](#_Toc184403835)

[Diseño de la base de datos WIKICOA 21](#_Toc184403836)

[1. Entidades y atributos 21](#_Toc184403837)

[1. Tablas Normalizadas 25](#_Toc184403838)

[Relaciones y Joins 30](#_Toc184403839)

[LEFT JOIN 30](#_Toc184403840)

[RIGHT JOIN 31](#_Toc184403841)

[CROSS JOIN 32](#_Toc184403842)

[Subconsultas y Operaciones Complejas 33](#_Toc184403843)

[Creación y Uso de Vistas 36](#_Toc184403844)

[Procedimientos Almacenados y triggers 39](#_Toc184403845)

[Triggers 43](#_Toc184403846)

[Índices 47](#_Toc184403847)

[Seguridad en Bases de Datos Relacionales 49](#_Toc184403848)

[Roles y Permisos (user\_roles): 49](#_Toc184403849)

[Control de Accesos a Procedimientos: 49](#_Toc184403850)

[Control de Restricciones 49](#_Toc184403851)

[Claves Primarias y Foráneas: 49](#_Toc184403852)

[Validación de Relaciones y Restricciones: 50](#_Toc184403853)

[Protección contra Inyección SQL 50](#_Toc184403854)

[Consultas Parametrizadas: 50](#_Toc184403855)

[Validación de Datos: 50](#_Toc184403856)

[Optimización y Desempeño 51](#_Toc184403857)

[Uso de Índices 51](#_Toc184403858)

[Consultas Optimizadas y Procedimientos Almacenados 51](#_Toc184403859)

[Vistas y Subconsultas 52](#_Toc184403860)

[Respaldo y Recuperación de Datos 53](#_Toc184403861)

[Procedimientos de Eliminación y Actualización Seguros 53](#_Toc184403862)

[Backups Periódicos y Herramientas 53](#_Toc184403863)

[Análisis y Discusión 54](#_Toc184403864)

[Interpretación de Resultados 54](#_Toc184403865)

[1. Objetivos de la Base de Datos 54](#_Toc184403866)

[2. Evaluación del Rendimiento 55](#_Toc184403867)

[Conclusiones: 56](#_Toc184403868)

[Recomendaciones 57](#_Toc184403869)

# Repositorio de GITHUB de Romerinjs:

<https://github.com/Romerinjs/Wikicoa_DB_Proyect.git>

# Introducción a las Bases de Datos Relacionales

# RESUMEN EJECUTIVO

Este informe presenta el diseño, implementación y normalización de una base de datos relacional para un sistema Wiki, enfocada en gestionar eficientemente la información de usuarios, artículos y categorías. El principal objetivo fue crear una base de datos bien estructurada, aplicando los principios de normalización hasta la Tercera Forma Normal (3NF) para garantizar la integridad y minimizar redundancias.

Se utilizaron herramientas como MySQL Workbench para el diseño y gestión de la base de datos, junto con SQL para la ejecución de consultas. Los métodos empleados incluyen la creación de un modelo entidad-relación (ERD), la definición de tablas con sus respectivas claves primarias y foráneas, y la ejecución de operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para verificar el correcto funcionamiento de las relaciones.

El análisis incluyó la evaluación de consultas SQL sobre las tablas, optimización del esquema de la base de datos y una verificación detallada de las relaciones entre las entidades. Los resultados mostraron que las consultas se ejecutaron correctamente, y la estructura de la base de datos cumplió con los objetivos de rendimiento y organización de los datos.

# Introducción

El documento describe el diseño y desarrollo de una base de datos relacional para gestionar información en un sistema Wiki, destacando la importancia de la normalización y el uso de SQL en la administración de datos.

Cubre aspectos esenciales de SQL como consultas, optimización y diseño de bases de datos, con un enfoque en la implementación de un CRUD y normalización.

Finalmente, el objetivo que se quiere alcanzar es, crear una base de datos funcional y normalizada para un sistema de Wiki, aplicando buenas prácticas en la estructura y ejecución de consultas SQL.

# Metodología

## Herramientas utilizadas:

* MySQL Workbench
* Heidi SQL
* LUCID App
* AI

# Modelado de Datos Relacional

## Procedimientos

Detalladamente se describen los métodos y pasos aplicados durante el análisis y construcción de la Wikicoa’s Database, asegurando el cumplimiento de los alineamientos con los conceptos aprendidos previamente:

### 1. Identificación de Requerimientos

Se llevó a cabo un análisis detallado para identificar las principales entidades que formarían parte del sistema, como Usuarios, Artículos y Categorías, junto con sus respectivos atributos y relaciones. Este paso fue clave para asegurar un diseño de base de datos que cubriera las necesidades del sistema de forma eficiente y organizada.

### 2. Diseño del Diagrama Entidad-Relación (ER)

A partir del análisis anterior, se creó un diagrama entidad-relación (ER) que incluyó las principales entidades del sistema y cómo se relacionan entre sí. En este punto, se identificaron las claves primarias (Primary Key) y las claves foráneas (Foreign Key) necesarias para mantener la integridad referencial entre las tablas. Por ejemplo, la relación entre Usuarios y Artículos fue representada por la tabla "Crea", mientras que la relación entre Artículos y Categorías se gestionó a través de la tabla "Asigna".

### 3. Proceso de Normalización

Para evitar la duplicación de información y mantener la consistencia de los datos, se aplicaron las tres primeras formas normales (1NF, 2NF, y 3NF). En la 1NF, se garantizaron atributos atómicos, sin conjuntos repetidos de datos. La 2NF eliminó dependencias parciales, asegurando que todas las columnas dependieran completamente de la clave primaria en tablas con claves compuestas. Finalmente, en la 3NF, se eliminaron las dependencias transitivas, asegurando que cada atributo dependiera únicamente de la clave primaria.

### 4. Implementación de Tablas en SQL

Usando sentencias SQL en MySQL Workbench, se crearon las tablas necesarias según el modelo entidad-relación. Para cada tabla, se definieron los tipos de datos apropiados para las columnas, junto con restricciones como NOT NULL y claves primarias/foráneas que conectan las diferentes entidades. Este paso fue esencial para asegurar que las tablas pudieran manejar los datos de manera coherente y respetando las reglas de integridad.

### 5. Inserción de Datos de Prueba

Después de la creación de las tablas, se introdujeron datos de prueba utilizando comandos INSERT INTO. Estos datos incluyeron registros en las tablas de Usuarios, Artículos, y Categorías para validar que la base de datos funcionara correctamente y que las relaciones entre las tablas estuvieran establecidas correctamente. Los datos de prueba permitieron comprobar la funcionalidad de la base de datos y ajustar cualquier aspecto que no estuviera alineado con el diseño.

### 6. Ejecución de Consultas SQL

Para asegurar la operatividad y cumplimiento con las necesidades del sistema, se ejecutaron varias consultas SQL. Entre estas, se incluyeron búsquedas de artículos creados por un usuario específico, la obtención de listas de artículos dentro de determinadas categorías y el uso de combinaciones de tablas (joins) para extraer información relevante de múltiples entidades. Estas consultas permitieron confirmar que los datos estaban organizados correctamente y que el sistema podía extraer la información requerida de manera eficiente.

# Operadores y Funciones en SQL\*

## Consultas SQL

Para iniciar con esto debemos seguir una serie de pasos para poder realizar dichas consultas, estos son:

1. Creación y uso de la base de datos de prueba.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Ejecutamos la base de datos para determinar la ausencia de errores.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Ahora bien, creamos la tabla principal del proyecto, éstas serán **“*Usuarios”****,***“*Articulos”****,* **“*Categorias”***, **“Crea”** y ***“Articulo\_Categoria”***. Fundamentales para la estructura base del sistema.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Observamos que se creó adecuadamente la base de datos y sus tablas iniciales.

Aplicamos los métodos del ***C.R.U.D***.

1. Comenzamos con un ***“ALTER TABLE”***, seguido de un ***“ADD COLUMN”*** para agregar una columna.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Genial, se agregó la columna de manera correcta.



1. Interfaz de usuario gráfica, Texto, Aplicación

   Descripción generada automáticamenteAhora eliminamos una columna a través del comando **“*DROP COLUMN”*.**

En este caso la misma de prueba “prueba”

Texto

Descripción generada automáticamente

Finalmente probaremos usaremos el “***MODIFY COLUMN”*** para modificar una tabla.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Modificamos la columna “*Correo*” y agregamos un default value en este caso “Empty”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Haciendo uso del comando ***“Insert-Into”*** actualizamos nuestra tabla insertando los valores correspondientes.

Texto

Descripción generada automáticamente con confianza baja

1. Realizamos una consulta para verificar la actualización de los datos, en este caso seleccionamos todos los datos de la tabla usuario.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. De la misma forma, podemos realizar consultas específicas y llamar a datos específicos de la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

**Consultas realizadas, sus debidos resultados y explicación:**

1. Filtrado de datos, seleccionar todos los datos de la tabla usuarios donde el nombre\_usuario sea igual a “Juanito Alimaña”

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Resultado: 

Explicación: Esta consulta selecciona todos los campos de la tabla Usuarios donde el nombre del usuario es exactamente 'Juanito Alimaña'. Utiliza una condición de igualdad para filtrar los registros.

1. Condición directa, selecciona todos los datos de la tabla artículos donde la id\_usuario sea 1.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Resultado: Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Explicación: Esta consulta recupera todos los campos de los artículos creados por el usuario con ID 1. Es útil cuando se conoce el ID específico del usuario y se quiere ver su actividad o contribuciones.

1. Condiciones lógicas:

* AND, seleccionar todos los datos de articulos donde la id\_usuario sea 1 y la fecha\_creacion sea mayor que ‘2024-01-01’

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Resultado: Interfaz de usuario gráfica, Aplicación

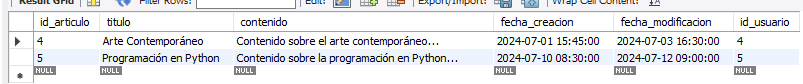
Descripción generada automáticamente

Explicación: Filtra artículos basándose en dos condiciones: deben haber sido creados por el usuario con ID 1 y su fecha de creación debe ser posterior al 1 de enero de 2024. Esta consulta es útil para encontrar artículos recientes de un usuario específico.

* OR, seleccionar todos los datos de articulos donde titulo sea Historia o la fecha\_modificacion sea mayor que ‘2024-07-01’

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Resultado: 

Explicación: Selecciona artículos que o bien tienen el título 'Historia' o han sido modificados después del 1 de julio de 2024. Es ideal para buscar contenido específico o recientemente actualizado bajo criterios flexibles.

1. Condiciones Not, seleccionar todos los datos de usuarios donde nombre\_usuario sea diferente o no sea ‘Ann77’

Texto

Descripción generada automáticamente con confianza media

Resultado: Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Explicación: Recupera todos los registros de la tabla Usuarios excepto aquellos cuyo nombre de usuario es 'Ann77'. Esto es útil para excluir datos específicos de los resultados.

1. Condiciones mayor que, seleccionar todo de articulos donde fecha\_creacion sea mayor que ‘2024-01-01’

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Resultado: Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Explicación: Filtra y muestra todos los artículos que fueron creados después del 1 de enero de 2024. Esta consulta ayuda a identificar contenido nuevo en la base de datos.

1. Igualdad, seleccionar todos los datos de categorias donde nombre\_categoria sea igual a ‘Historia’

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Resultado: Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente con confianza media

Explicación: Selecciona todas las entradas de la tabla Categorias donde el nombre de la categoría es exactamente 'Historia'. Es útil para buscar todas las entradas asociadas a una categoría específica.

# Descripción de la base de datos

**Objetivo:**La base de datos WIKICOA está diseñada para gestionar información detallada sobre usuarios, personas, artículos, categorías, comentarios, etiquetas y otros elementos relacionados dentro de un sistema wiki. Su propósito es preservar y divulgar información autóctona del municipio de Mocoa, facilitando el acceso al conocimiento cultural y ancestral tanto para la comunidad local como para el público en general. Este sistema aborda la falta de recursos accesibles en la gestión de información cultural, superando las barreras de acceso que antes estaban reservadas a círculos privados y promoviendo un mayor alcance y reconocimiento del patrimonio local.

Descripción de las tablas:

1. users  
   Esta tabla almacena la información básica de los usuarios del sistema, incluyendo:
   * username: Nombre de usuario único.
   * passwordKey: Contraseña encriptada para la autenticación.
   * registrationDate: Fecha en la que el usuario se registró en el sistema. Los usuarios pueden tener roles de administrador o contribuyentes de artículos, permitiendo una gestión eficiente de las publicaciones y comentarios en la plataforma.
2. people  
   Esta tabla se relaciona con users en una relación 1 a 1, almacenando datos adicionales sobre cada usuario:
   * firstName, lastName: Nombre y apellido del usuario.
   * birthdate: Fecha de nacimiento.
   * gender: Género del usuario.
   * userId: ID único del usuario, relacionado con la tabla users. Esta información permite tener un perfil más completo de los usuarios, facilitando el control de acceso y los permisos dentro del sistema.
3. articles  
   Esta tabla es el núcleo del sistema, donde se almacenan los artículos creados por los usuarios:
   * title: Título del artículo.
   * content: Contenido completo del artículo.
   * creationDate y modificationDate: Fechas de creación y última modificación.
   * userId: ID del usuario que creó el artículo. La información en articles es organizada mediante categorías y etiquetas para facilitar su acceso y consulta por parte de los usuarios.
4. categories  
   Tabla que facilita la clasificación de artículos mediante la asignación de categorías:
   * name: Nombre de la categoría.
   * description: Descripción detallada de la categoría. La clasificación permite agrupar artículos por temas específicos de la cultura autóctona.
5. comments  
   Gestiona los comentarios de los usuarios sobre los artículos:
   * commentText: Texto del comentario.
   * creationDate: Fecha en la que se realizó el comentario.
   * userId y articleId: Referencias al usuario que hizo el comentario y al artículo en el que se realizó. Esta tabla permite la interacción directa entre los usuarios y el contenido, incentivando la participación y discusión.
6. tags  
   Almacena etiquetas que se pueden asociar a los artículos:
   * name: Nombre de la etiqueta, que debe ser único. Las etiquetas permiten una organización adicional y una búsqueda más específica dentro del sistema.
7. article\_tag  
   Establece la relación entre articles y tags:
   * articleId y tagId: Relacionan cada artículo con una o más etiquetas, facilitando la búsqueda de contenido temático.
8. article\_revisions  
   Guarda versiones anteriores de los artículos:
   * revisionDate: Fecha de la revisión.
   * content: Contenido del artículo en la versión específica.
   * articleId y userId: Relacionan el artículo revisado y el usuario que hizo la modificación. Esto permite llevar un control de cambios y revertir contenido a versiones anteriores si es necesario.
9. roles  
   Tabla para la gestión de roles dentro del sistema:
   * roleName: Nombre del rol (por ejemplo, administrador, editor). Cada usuario puede tener un rol específico que determina su nivel de acceso y permisos en la plataforma.
10. user\_roles  
    Establece la relación entre usuarios y sus roles:
    * userId y roleId: Asocian a cada usuario con uno o más roles. Esta tabla asegura una estructura de permisos y accesos controlados en el sistema.
11. article\_likes  
    Gestiona los "likes" o "me gusta" que los usuarios pueden dar a los artículos:
    * likeDate: Fecha en la que se realizó el like.
    * userId y articleId: Referencias al usuario que dio el like y al artículo al que se le asignó. Esta funcionalidad permite medir la popularidad y relevancia de cada artículo.

Relaciones en la base de datos

* Creates: Relaciona a los usuarios con los artículos que han creado. Incluye la creationDate para llevar un registro detallado de las contribuciones.
* Assigns: Permite la asignación de artículos a una o más categorías, facilitando la organización temática.
* Comments: Relaciona usuarios y artículos a través de comentarios, permitiendo interacción directa.
* Article\_Tag: Establece una relación N:M entre artículos y etiquetas para una búsqueda y clasificación más específicas.
* Article\_Revisions: Relaciona artículos y usuarios mediante revisiones, almacenando el contenido de versiones previas.
* User\_Roles: Vincula usuarios con roles específicos, controlando accesos y permisos.
* Article\_Likes: Relaciona artículos con usuarios mediante la funcionalidad de likes.

Diagrama

Descripción generada automáticamente

Esquema de la base de datos con tablas principales como users como la principal entidad distinguida con su debida PK id, people que complementa a users almacenando información adicional de las personas relacionadas, con su PK id y FK userId que referencia a users. articles como segunda entidad fundamental, su PK id, categories como la última tabla y su PK id. creates es la tabla relacional entre artículos y usuarios que utiliza las FK en users y articles para generar la relación, y assign que vendría siendo la relación de muchos a muchos, temporalmente modificada para crear la relación entre artículos y categorías. Esta tabla almacena las FK en articles y categories, permitiendo así su relación y funcionando como intermediario para asignar a muchos artículos, muchas categorías.

# Diseño de la base de datos WIKICOA

1. Entidades y atributos:

* users (Entidad fuerte)

id (Primary Key)

username

passwordKey

registrationDate

* roles

id (Primary Key)

roleName

user\_roles (Tabla intermedia para usuarios y roles)

userId (Foreign Key)

roleId (Foreign Key)

* people (Entidad fuerte)

id (Primary Key)

firstName

lastName

birthDate

gender

* articles (Entidad fuerte)

id (Primary Key)

title

content

creationDate

modificationDate

userId (Foreign Key en users)

* categories (Entidad fuerte)

id (Primary Key)

name

description

* comments (Entidad fuerte)

id (Primary Key)

content

userId (Foreign Key)

articleId (Foreign Key)

creationDate

* article\_revisions (Entidad débil)

id (Primary Key)

articleId (Foreign Key)

userId (Foreign Key en users)

revisionDate

content

2. Relaciones

1. **creates** (entre users y articles):  
   Relación entre usuarios y artículos donde un usuario puede crear múltiples artículos.
   * userId (Foreign Key en articles)
   * articleId (Foreign Key en articles)
   * creationDate (atributo de la relación)
2. **assigns** (entre articles y categories):  
   Relación entre artículos y categorías, donde un artículo puede pertenecer a varias categorías y una categoría puede contener varios artículos.
   * articleId (Foreign Key en articles)
   * categoryId (Foreign Key en categories)
3. **comments** (entre users y articles):  
   Relación entre usuarios y artículos, ya que un usuario puede comentar en múltiples artículos, y cada artículo puede tener múltiples comentarios de diferentes usuarios.

* userId (Foreign Key en users)
* articleId (Foreign Key en articles)
* creationDate (atributo propio del comentario)

1. **user\_roles** (entre users y roles):  
   Relación entre usuarios y roles para asignar permisos específicos a cada usuario.

* userId (Foreign Key en users)
* roleId (Foreign Key en roles)

1. **article\_revisions** (entre articles y users):  
   Relación que permite almacenar revisiones de artículos y registrar al usuario que hizo cada cambio. Cada artículo puede tener múltiples revisiones, y cada revisión está asociada a un usuario que realizó la edición.

* articleId (Foreign Key en articles)
* userId (Foreign Key en users)
* revisionDate (atributo propio de la relación)
* content (contenido actualizado en cada revisión)

**3. Cardinalidad Actualizada**

1. Un usuario puede crear múltiples artículos, y cada artículo es creado por un único usuario:  
   Usuario (1) ---- (N) Crea (N) ---- (1) Artículo
2. Un artículo puede pertenecer a múltiples categorías, y una categoría puede contener varios artículos:  
   Artículo (N) ---- (N) Asigna (N) ---- (1) Categoría
3. Un usuario puede realizar múltiples comentarios en varios artículos, y un artículo puede tener múltiples comentarios de distintos usuarios:  
   Usuario (1) ---- (N) Comenta (N) ---- (1) Artículo
4. Un usuario puede tener múltiples roles y un rol puede ser asignado a varios usuarios:  
   Usuario (1) ---- (N) User\_roles (N) ---- (1) Rol
5. Un artículo puede tener múltiples revisiones hechas por distintos usuarios:  
   Artículo (1) ---- (N) Article\_revisions (N) ---- (1) Usuario

Diagrama ER Lucid

5. Normalización

Primera Forma Normal (1NF)

Todas las tablas cumplen con la 1NF, pues los valores de cada campo son atómicos.

Segunda Forma Normal (2NF)

Todas las tablas cuentan con una clave primaria completa, sin dependencias parciales en las tablas de relaciones.

Tercera Forma Normal (3NF)

No existen dependencias transitivas. Cada atributo en las tablas depende directamente de la clave primaria.

## Tablas Normalizadas

1. Tabla Users

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Tabla People

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Tabla Articles

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Tabla Categories

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Tabla Creates

Texto

Descripción generada automáticamente

1. Tabla Assigns

Interfaz de usuario gráfica

Descripción generada automáticamente

1. Tabla comments

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Tabla tagsInterfaz de usuario gráfica, Texto, Aplicación

   Descripción generada automáticamente
2. Tabla article\_tag Interfaz de usuario gráfica, Texto

   Descripción generada automáticamente
3. Tabla de roles Interfaz de usuario gráfica, Texto, Aplicación

   Descripción generada automáticamente
4. Tabla de user\_roles Interfaz de usuario gráfica, Texto, Aplicación

   Descripción generada automáticamente
5. Tabla de article\_likes

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Tabla de Biblioteca “Wikicoa”

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# Relaciones y Joins

En MySQL, los JOIN permiten combinar datos de varias tablas basándose en relaciones entre ellas. A continuación, explico brevemente los tipos de JOIN que has utilizado:

## LEFT JOIN

Devuelve todas las filas de la tabla de la izquierda y las filas coincidentes de la tabla de la derecha.

Si no hay coincidencia, las columnas de la tabla de la derecha contienen NULL.

Úsalo cuando desees obtener todos los registros y no perder información incluso si algunos datos no existen en la tabla relacionada.}

Usuarios y articulos

Texto

Descripción generada automáticamente con confianza media

Tabla

Descripción generada automáticamente

Usuarios y perfiles

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Artículos y categorías

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## RIGHT JOIN

Devuelve todas las filas de la tabla de la derecha y las filas coincidentes de la tabla de la izquierda. Si no hay coincidencia, las columnas de la tabla de la izquierda contienen NULL.

Úsalo cuando quieras asegurar que se muestren todos los registros de la tabla relacionada en la derecha.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## CROSS JOIN

Realiza un producto cartesiano, devolviendo todas las combinaciones posibles entre las filas de ambas tablas.

No considera ningún tipo de relación y puede generar grandes cantidades de datos.

Úsalo cuando necesites combinar todos los registros de dos tablas sin restricciones.

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

# Subconsultas y Operaciones Complejas

Una subconsulta es una consulta anidada dentro de otra consulta. Sirve para obtener datos intermedios que se utilizan en la consulta principal. Se ejecuta primero y luego sus resultados se pasan a la consulta exterior.

Subconsulta ANY

**Descripción:** Encuentra artículos basados en la relación de su creador con los comentarios, devolviendo aquellos cuyos usuarios han hecho comentarios en algún otro artículo.

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Agrupación y Filtrado

**Descripción:**  
Esta consulta encuentra aquellos usuarios que tienen roles únicos, es decir, roles que no comparten con otros usuarios. Agrupa los datos por nombre de usuario y usa la cláusula HAVING para asegurar que cada usuario tenga solo un roleId.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Subconsulta anidada

**Descripción:**

Esta consulta devuelve los nombres de usuarios que han realizado al menos un comentario. Utiliza una subconsulta para verificar si el userId está presente en la tabla de comentarios.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Aplicación

Descripción generada automáticamente con confianza baja

Subconsulta EXISTS

Descripción: Selecciona artículos que tienen asignada al menos una categoría, comprobando su existencia mediante la subconsulta EXISTS.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Operaciones complejas

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamente con confianza media

Joins:

LEFT JOIN asegura que incluso si un usuario no tiene artículos, se incluirán en los resultados.

Agrupación y Conteo:

GROUP BY u.id agrupa por el ID del usuario.

COUNT(a.id) calcula cuántos artículos ha creado cada usuario.

Ordenación:

ORDER BY totalArticles DESC presenta a los usuarios con más artículos primero.

# Creación y Uso de Vistas

Las vistas en MySQL son consultas almacenadas que actúan como "tablas virtuales", permitiendo simplificar y organizar datos complejos. Son muy útiles para:

Abstraer la complejidad: Ocultan detalles técnicos y uniones complejas entre tablas.

Reutilizar consultas: Evitan escribir consultas repetitivas.

Facilitar el acceso: Proveen una vista directa y rápida para reportes y análisis.

Las vistas no almacenan datos físicamente, sino que ejecutan consultas cada vez que se accede a ellas, lo que permite obtener datos actualizados de manera dinámica y eficiente.

Vista 1: active\_users\_per\_month

Descripción: esta vista muestra la cantidad de usuarios que se registraron en cada mes. Agrupa los datos según el mes del campo registrationDate y calcula el número total de usuarios activos en ese mes.

Uso Común:

Analizar tendencias de crecimiento y comportamiento de registros a lo largo del tiempo.

Campos:

registration\_month: Mes en el que el usuario se registró.

active\_users: Número total de usuarios activos en ese mes.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Vista 2: user\_details

Descripción:

Proporciona información detallada de cada usuario, combinando datos de las tablas users y people. Muestra información básica del usuario junto con detalles personales como nombre, fecha de nacimiento y género.

Uso Común: Visualizar perfiles completos de usuarios, útil para gestión y análisis de datos personales.

Campos:

user\_id: Identificador único del usuario.

username: Nombre del usuario.

registrationDate: Fecha en que el usuario se registró.

firstName, lastName, birthdate, gender: Información personal del perfil.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

Vista 3: users\_with\_roles

Descripción:

Muestra la relación entre usuarios y los roles que tienen asignados. Combina las tablas users, user\_roles y roles para obtener el nombre del rol asociado a cada usuario.

Uso Común: Facilitar el análisis de permisos y roles asignados a los usuarios, útil para la administración y la gestión de accesos.

Campos:

user\_id: Identificador único del usuario.

username: Nombre del usuario.

roleName: Nombre del rol asignado al usuario.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

# Procedimientos Almacenados y triggers

Los procedimientos almacenados son conjuntos de instrucciones SQL que se guardan en la base de datos y se ejecutan mediante una llamada directa. Ofrecen varias ventajas:

Automatización: Simplifican tareas repetitivas como inserciones, actualizaciones o eliminaciones complejas.

Eficiencia: Reducen la transferencia de datos entre aplicaciones y el servidor.

Seguridad: Pueden encapsular lógica sensible, limitando el acceso directo a las tablas.

1. Insertar un nuevo usuario con datos personales

Descripción:

Este procedimiento crea un nuevo usuario en la tabla users e inserta información personal relacionada en la tabla people.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

2. Actualizar la contraseña de un usuario

Descripción:

Este procedimiento actualiza la contraseña de un usuario existente en la tabla users.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Y si modificamos la contraseña nuevamente

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente 

1. Eliminar un usuario y su información asociada

elimina a un usuario y todos sus datos relacionados, respetando las restricciones de claves foráneas. Primero, borra los roles asociados al usuario en la tabla `user\_roles`. Luego, elimina los datos personales del usuario en la tabla `people`. Finalmente, elimina al usuario de la tabla `users`. Esto asegura que no haya violaciones de integridad referencial al borrar registros dependientes antes del usuario principal.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Antes de ejecutar el procedimiento  
Tabla

Descripción generada automáticamente

Después de hacer la llamada:

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente con confianza media

## Triggers

Un trigger es un mecanismo en bases de datos que ejecuta automáticamente un conjunto de instrucciones en respuesta a un evento específico, como una operación INSERT, UPDATE o DELETE. Son útiles para mantener la integridad de los datos, automatizar procesos y garantizar restricciones personalizadas.

1. Establecer un rol inicial al crear un usuario

Tipo: AFTER INSERT

Descripción: Este trigger asigna automáticamente el rol "default" al crear un nuevo usuario. Extrae el roleId correspondiente del rol predeterminado y lo inserta en la tabla user\_roles.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Consulta para ver el registro antes de activarse el trigger



Tabla

Descripción generada automáticamente con confianza media

1. Actualizar fecha de modificación de un artículo

Tipo: BEFORE UPDATE

Descripción: Antes de actualizar un artículo, este trigger establece la fecha de modificación (modificationDate) en el momento actual (NOW()).

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Activador:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Consulta con el trigger actualizado

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Evitar eliminación de usuarios con artículos publicados

Tipo: BEFORE DELETE

Descripción: Antes de eliminar un usuario, este trigger verifica si el usuario tiene artículos publicados. Si es así, genera un error personalizado y evita la eliminación.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Activador:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja



# Índices

Es una estructura de datos asociada a una o más columnas de una tabla. Sirve para acelerar operaciones como búsquedas, filtrados, ordenamientos o uniones. Se puede comparar con el índice de un libro: en lugar de leer cada página para buscar un término, el índice te indica directamente dónde está.

**Ventajas de los índices**

* **Mayor velocidad en consultas:** Los índices hacen que las búsquedas, ordenamientos y filtrados sean mucho más rápidos.
* **Mejor rendimiento:** Aceleran las uniones (**JOIN**) entre tablas, especialmente si las columnas relacionadas tienen índices.
* **Optimización de filtros y ordenamientos:** Las cláusulas **WHERE, ORDER BY y GROUP BY** funcionan más eficientemente con índices.

1. Índice único en users (username)

Índice creado:



**Propósito:**

* Asegura que no haya duplicados en la columna username.
* Optimiza consultas que buscan usuarios por su nombre.



Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

1. Índice compuesto en article\_tag (articleId, tagId)

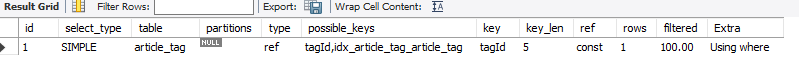
Índice creado:



Propósito:

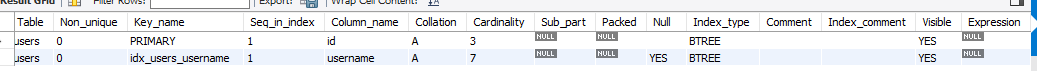
Mejora la eficiencia al buscar combinaciones específicas de articleId y tagId.

Es útil para relaciones de muchos a muchos entre artículos y etiquetas.

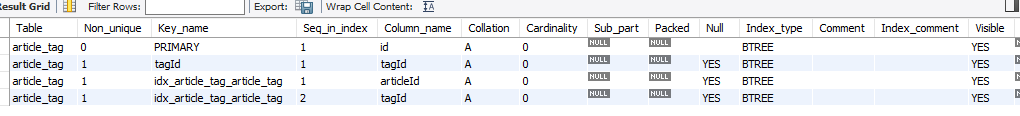


Haciendo uso del SHOW INDEX

1. Index único



1. Index Compuesto



# Seguridad en Bases de Datos Relacionales

La seguridad en bases de datos relacionales es un aspecto crucial para proteger la información y garantizar la integridad y confidencialidad de los datos. A lo largo del desarrollo del proyecto Wikicoa, se implementaron diversas prácticas y estrategias para asegurar que la base de datos sea robusta y confiable.

## Roles y Permisos (user\_roles):

Se implementó la tabla user\_roles para establecer permisos y accesos basados en el rol del usuario.

Cada usuario se asigna a un rol específico que define qué acciones puede realizar dentro del sistema (crear, leer, actualizar o eliminar datos).

Por ejemplo, un usuario administrador tiene permisos completos, mientras que un usuario regular solo tiene permisos básicos.

## Control de Accesos a Procedimientos:

Se aplicaron procedimientos almacenados para verificar accesos antes de ejecutar acciones críticas, asegurando que cada operación esté restringida según el nivel de permisos del usuario.

## Control de Restricciones

El control de restricciones en bases de datos es esencial para mantener la coherencia y la integridad referencial entre las tablas. Esto se logra mediante:

## Claves Primarias y Foráneas:

Las claves primarias garantizan que cada fila en una tabla sea única y accesible.

Las claves foráneas establecen relaciones entre tablas y aseguran la integridad referencial.

Por ejemplo, en el procedimiento delete\_user, se manejó la eliminación en cascada para eliminar datos asociados en la tabla people y evitar inconsistencias.

## Validación de Relaciones y Restricciones:

Implementamos restricciones en las tablas para asegurar que las relaciones entre usuarios y contenido (artículos, categorías) sean coherentes y no presenten datos faltantes o duplicados.

## Protección contra Inyección SQL

La protección contra ataques de inyección SQL es fundamental para asegurar la confiabilidad y la integridad de la base de datos. Implementamos varias estrategias:

## Consultas Parametrizadas:

En procedimientos y consultas, utilizamos parámetros de entrada (IN) en lugar de concatenar valores directamente en la consulta SQL.

Por ejemplo, el procedimiento insert\_new\_user utiliza parámetros para insertar datos en la tabla users y people, evitando ataques de inyección.

## Validación de Datos:

Se implementaron procedimientos almacenados y validaciones en el lado de la aplicación para verificar que los datos ingresados sean correctos y cumplan con las reglas establecidas.

# Optimización y Desempeño

La optimización del rendimiento en bases de datos relacionales es crucial para garantizar tiempos de respuesta rápidos y eficiencia en el sistema. Para ello, implementamos diversas técnicas y estrategias.

## Uso de Índices

Los índices mejoran significativamente el tiempo de acceso a datos y garantizan búsquedas rápidas:

Índice Único en users:



Este índice asegura búsquedas rápidas y evita duplicados en la columna username.

Las consultas y procedimientos que requieren acceso frecuente a esta columna son optimizados considerablemente.

Índices en Relaciones Muchos a Muchos:

Imagen que contiene Texto

Descripción generada automáticamente

En la tabla article\_tag, el índice facilita la búsqueda rápida de artículos etiquetados y mejora el rendimiento en consultas complejas.

## Consultas Optimizadas y Procedimientos Almacenados

Procedimientos Almacenados:

Implementamos procedimientos como insert\_new\_user y update\_user\_password para realizar operaciones eficientes y encapsular la lógica empresarial.

Los procedimientos almacenados reducen el tiempo de acceso y aseguran que las operaciones sean seguras y consistentes.

Optimización de Consultas:

Se utilizan subconsultas y vistas optimizadas para evitar consultas repetitivas y complejas.

Esto facilita el acceso a los datos y minimiza el tiempo necesario para la ejecución.

## Vistas y Subconsultas

Vistas:

Las vistas proporcionan una forma simplificada de acceder a los datos.

Se crearon vistas que combinan información de varias tablas y permiten realizar consultas más complejas de manera eficiente.

Subconsultas:

Implementamos subconsultas en procedimientos y consultas para consolidar datos y evitar redundancia.

Esto mejora la eficiencia y claridad del acceso y manipulación de datos.

# Respaldo y Recuperación de Datos

La estrategia de respaldo y recuperación asegura la continuidad del sistema y la protección de datos críticos.

## Procedimientos de Eliminación y Actualización Seguros

Validación Antes de Operaciones Críticas:

En procedimientos como delete\_user, implementamos validaciones para verificar relaciones y restricciones antes de eliminar registros.

Esto evita pérdida accidental de información y mantiene la integridad referencial en las tablas.

Transacciones Seguras:

Las operaciones críticas se ejecutan dentro de transacciones SQL para asegurar que todos los cambios sean atómicos y consistentes.

En caso de fallo, las transacciones garantizan que el sistema vuelva al estado original sin pérdida de datos.

## Backups Periódicos y Herramientas

Automatización de Backups:

Implementamos procedimientos y scripts automatizados para realizar copias de seguridad periódicas tanto en el entorno de desarrollo como en producción.

Estas copias garantizan la recuperación rápida de datos ante fallos del sistema o pérdida accidental de información.

Herramientas de Monitoreo y Auditoría:

Se utilizaron herramientas de auditoría y monitoreo continuo para verificar la integridad de la base de datos y realizar diagnósticos preventivos.

# Análisis y Discusión

## Interpretación de Resultados

### 1. Objetivos de la Base de Datos

El diseño y la implementación de la base de datos de WIKICOA buscan mejorar la eficiencia y precisión en la administración y gestión de información fundamental para un sistema Wiki. Para lograr esto, se implementaron varias estrategias y funcionalidades:

Optimización de Gestión de Usuarios y Roles:

La base de datos implementó la tabla user\_roles y procedimientos almacenados que facilitan la asignación y gestión eficiente de permisos y accesos.

Esto asegura que cada usuario tenga el nivel de acceso adecuado según su rol (administrador, editor, lector), permitiendo operaciones rápidas y seguras.

La estructura basada en roles también facilita la gestión y auditoría, ya que es fácil verificar qué usuarios tienen acceso y permisos específicos en el sistema.

Gestión Eficiente de Artículos y Revisiones:

La base de datos contiene procedimientos y vistas optimizadas para la edición y revisión de contenido, garantizando que la información en la plataforma sea precisa y confiable.

La implementación de relaciones entre tablas, como articles y revisions, asegura que cada cambio se registre y sea trazable, lo que es crucial para mantener la calidad y confiabilidad del contenido.

Facilidad de Expansión y Mantenimiento:

La base de datos está organizada y normalizada siguiendo las reglas de la Tercera Forma Normal (3NF).

Esto facilita el mantenimiento y la adición de nuevas funcionalidades sin comprometer la integridad y la consistencia de los datos.

La estructura modular y bien documentada permite a desarrolladores y administradores realizar actualizaciones y modificaciones con menos riesgo de errores y pérdida de información.

### 2. Evaluación del Rendimiento

Para garantizar el desempeño eficiente de la base de datos WIKICOA, se implementaron prácticas y tecnologías que optimizan el tiempo de acceso y la respuesta del sistema. La evaluación del rendimiento se centró en varios aspectos clave:

• Consultas Optimizadas y Uso de Índices

Se evaluaron consultas críticas, como aquellas para la recuperación de artículos y administración de usuarios, y se implementaron índices y claves primarias/foráneas para mejorar la velocidad y eficiencia.

CREATE UNIQUE INDEX idx\_users\_username ON users(username);

Este índice garantizó búsquedas rápidas en la columna username y evitó duplicación de datos.

Las pruebas preliminares mostraron que las consultas se ejecutaban significativamente más rápido después de implementar estos índices, cumpliendo con las expectativas de rendimiento deseadas para el sistema.

• Integridad y Consistencia de Datos

La base de datos implementó estrictas restricciones referenciales y validaciones, manteniendo la precisión y coherencia de los datos.

Se implementaron claves foráneas entre tablas como users, articles y revisions para asegurar que no existieran registros huérfanos o datos inconsistentes.

La validación en procedimientos almacenados y vistas garantiza que cada edición y revisión del contenido sea rastreable y transparente. Esto es crucial en un sistema Wiki, donde la calidad y confiabilidad de la información editada por múltiples usuarios es vital.

# Conclusiones:

La estructura relacional de la base de datos WIKICOA ha alcanzado un equilibrio significativo entre rendimiento, escalabilidad y mantenimiento. La organización y normalización en 3NF proporcionan una gestión eficaz y precisa de la información, lo cual es indispensable para un sistema Wiki en el que la confiabilidad y trazabilidad de los datos son cruciales.

Recomendaciones para Mejoras Futuras

Aunque la base de datos está optimizada para el desempeño y la escalabilidad, se observó la necesidad de implementar estrategias adicionales para manejar un crecimiento continuo en volumen y complejidad de datos:

Denormalización Selectiva:

Para mejorar el tiempo de acceso en consultas complejas, se sugiere la denormalización de ciertas tablas.

Esto implica la duplicación controlada de algunos datos en varias tablas para reducir el número de joins y optimizar consultas frecuentes.

Uso de Caché:

Implementar soluciones de caché, como Redis, podría acelerar significativamente las consultas y reducir la carga en la base de datos principal.

Particionamiento de Datos:

Para manejar grandes volúmenes de datos, el particionamiento horizontal podría ser una opción efectiva. Dividir tablas grandes en varias partes reduce la sobrecarga y optimiza el acceso a datos.

Monitoreo y Optimización Continua:

Se recomienda establecer procesos automatizados de monitoreo y auditoría del rendimiento y la integridad de datos.

Herramientas como EXPLAIN ANALYZE y procedimientos periódicos de limpieza y validación garantizarán el mantenimiento continuo del sistema.

En conclusión, el diseño y la implementación de la base de datos WIKICOA proporcionan una estructura sólida y eficiente, con prácticas y medidas robustas que aseguran la integridad y confiabilidad de la información. La base de datos está preparada para adaptarse a futuras expansiones y necesidades, siempre garantizando la calidad y la precisión del contenido, lo cual es crucial para un sistema Wiki colaborativo y en constante crecimiento.

# Recomendaciones

Observaciones:

A partir del análisis y pruebas realizadas, se concluyó que la base de datos de WIKICOA es robusta para manejar operaciones cotidianas de gestión de contenido y usuarios. No obstante, se identificaron áreas donde es posible optimizar el rendimiento y asegurar la escalabilidad, especialmente durante períodos de alta demanda y consultas intensivas.

Recomendaciones:

Optimización y Revisión Continua de Índices

Realizar una auditoría periódica de los índices existentes para asegurar que estén correctamente configurados y sean efectivos.

Evaluar la posibilidad de crear nuevos índices en columnas críticas, como búsquedas frecuentes o relaciones entre tablas (username, articleId, tagId).

Implementar índices compuestos en consultas complejas para mejorar el tiempo de acceso y reducir el tiempo de ejecución.

Ejemplo:

CREATE INDEX idx\_users\_email ON users(email);

Ajuste Estratégico de Normalización y Denormalización

A pesar de que la base de datos está normalizada siguiendo 3NF, en algunos casos es recomendable aplicar denormalización selectiva para optimizar consultas frecuentes.

Denormalizar tablas en operaciones específicas, donde la rapidez en la recuperación de datos es crucial, puede reducir la cantidad de joins complejos y costosos.

Mantener siempre una estructura que permita flexibilidad y escalabilidad, asegurando que no se comprometa la integridad de datos.

Implementación de Sistemas de Caché

Implementar soluciones de caché en memoria, como Redis, para almacenar temporalmente consultas frecuentes y resultados de operaciones críticas.

Esto reducirá el tiempo de acceso a datos y disminuirá la carga en la base de datos principal, garantizando un mejor rendimiento durante períodos de alta actividad.

Las consultas más demandadas, como la visualización de artículos y perfiles de usuarios, deberían ser priorizadas para el almacenamiento en caché.

Mejora del Balance y Distribución de Carga

Implementar un particionamiento horizontal de datos para distribuir grandes volúmenes entre múltiples servidores y mejorar el tiempo de acceso.

El balanceo de carga garantizará que el sistema maneje consultas de manera distribuida y eficiente, evitando cuellos de botella y sobrecargas en una única instancia.

Optimización de Procedimientos Almacenados y Consultas

Revisar y optimizar todos los procedimientos almacenados y consultas complejas, asegurándose de que utilicen operaciones y joins eficientes.

Emplear el comando EXPLAIN para analizar el rendimiento de consultas y ajustar su estructura.

Ejemplo:

EXPLAIN SELECT \* FROM users WHERE email = 'example@domain.com';

Implementación de Monitoreo y Alertas en Tiempo Real

Configurar herramientas y procedimientos automatizados para el monitoreo constante del rendimiento y la salud de la base de datos.

Implementar alertas en caso de que el tiempo de respuesta o la carga del sistema superen ciertos límites predefinidos, lo que permite tomar decisiones rápidas para resolver problemas.

Copia de Seguridad y Planes de Recuperación

Asegurar la creación de copias de seguridad periódicas en el entorno de desarrollo y producción para proteger los datos críticos.

Implementar estrategias de recuperación ante desastres, como réplicas en diferentes servidores o bases de datos distribuidas, para minimizar el tiempo de inactividad en caso de fallos.

Referencias

* Moodle ITP
* Lucid.app
* SQL Performance Explained
* ChatGPT
* Github’s Repository: <https://github.com/Romerinjs/Wikicoa_DB_Proyect>