INFORME REPASO SQL

Presentado por:

YULY CUADROS

Presentado a:

ING.BRAYAN ARCOS

INSTITUTO TECNOLOGICO DEL PUTUMAYO

MOCOA-PUTUMAYO

2024

TABLA DE CONTENIDO

RESUMEN EJECUTIVO

Este informe presenta un análisis exhaustivo de los conceptos clave aplicados en la base de datos relacional gestion_empresarial en un entorno de desarrollo MySQL. Se abordan temas como el lenguaje de definición de datos (DDL), el lenguaje de manipulación de datos (DML), los tipos de datos SQL, funciones, operadores, consultas avanzadas y consideraciones de diseño, proporcionando un enfoque técnico detallado para la implementación de bases de datos eficientes.

INTRODUCCIÓN

El uso de bases de datos relacionales es fundamental para la gestión estructurada de información. En este informe, se examina la implementación de la base de datos gestion_empresarial en MySQL, describiendo cómo los conceptos fundamentales como DDL, DML, y consultas avanzadas son empleados para gestionar los datos y garantizar la integridad referencial.

METODOLOGÍA

La base de datos fue creada y gestionada mediante comandos SQL ejecutados en un entorno de desarrollo MySQL. Se utilizó una metodología basada en la manipulación y definición de datos, aplicando los conceptos teóricos de bases de datos relacionales en un entorno práctico, partiendo desde la creación de tablas hasta la consulta avanzada de los datos.

LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

El DML es el conjunto de instrucciones SQL utilizadas para manipular los datos dentro de las tablas. En esta base de datos, se ejecutaron las siguientes operaciones:

- **INSERT**: Se añadieron registros a las tablas, como los empleados asignados a proyectos.

```
INSERT INTO `asignaciones_proyectos`(`id_empleado`, `id_proyecto`, `fecha_asignacion`)
VALUES (1, 2, '2021-01-15');
```

- **UPDATE**: Modificación de registros.

```
UPDATE `empleados`
SET `nombre` = 'Juan Pérez'
WHERE `id_empleado` = 1;
```

- **DELETE**: Eliminación de registros.

```
DELETE FROM `asignaciones_proyectos`
WHERE `id_asignacion` = 5;
```

LENGUAJE DE DEFINICIÓN DE DATOS (DDL)

El DDL fue empleado para definir la estructura de las tablas, incluyendo las claves primarias y foráneas:

- **CREATE TABLE**: Se utilizó para crear las tablas asignaciones_proyectos, departamentos, y empleados.

```
CREATE TABLE `empleados` (
  `id_empleado` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(50) NOT NULL,
  PRIMARY KEY (`id_empleado`)
);
```

- **ALTER TABLE**: Modificación de la estructura de una tabla, por ejemplo, añadiendo una columna nueva.

```
ALTER TABLE `proyectos`
ADD `fecha_inicio` date;
```

- **DROP TABLE**: Eliminación de tablas.

```
DROP TABLE IF EXISTS `departamentos`;
```

TIPOS DE DATOS EN SQL

Los tipos de datos son cruciales para definir cómo se almacenan los datos en las tablas. Los tipos utilizados incluyen:

- **INT**: Para identificadores únicos como id_empleado, id_proyecto, etc.
- **VARCHAR**: Para cadenas de texto, como nombre_departamento.
- **DATE**: Para almacenar fechas, como fecha_asignacion.

Ejemplo:

```
CREATE TABLE `departamentos` (
  `id_departamento` int(11) NOT NULL,
```

```
  `nombre_departamento` varchar(50) NOT NULL,
  PRIMARY KEY (`id_departamento`)
);
```

FUNCIONES Y OPERADORES EN SQL

Las funciones y operadores permiten realizar cálculos y transformaciones sobre los datos almacenados. En la base de datos gestion_empresarial se utilizaron:

- **COUNT()**: Para contar el número de filas que coinciden con un criterio específico.

```
SELECT COUNT(*) FROM `empleados`;
```

- **SUM()**: Para sumar valores de una columna.

```
SELECT SUM(salario) FROM `empleados`;
```

- **Operadores lógicos**: AND, OR, NOT para combinar condiciones en consultas.

CONSULTAS SQL AVANZADAS

Las consultas avanzadas fueron utilizadas para obtener información más compleja:

- **JOINS**: Se utilizó para combinar datos de múltiples tablas relacionadas.

```
SELECT e.nombre, p.nombre_proyecto
FROM empleados e
JOIN asignaciones_proyectos ap ON e.id_empleado = ap.id_empleado
JOIN proyectos p ON ap.id_proyecto = p.id_proyecto;
```

- **Subconsultas**: Consultas anidadas dentro de otras.

```
SELECT nombre
FROM empleados
WHERE id_departamento = (SELECT id_departamento FROM departamentos WHERE nombre_departamento = 'Ventas');
```

CONSIDERACIÓN DE DISEÑO

La base de datos fue diseñada para optimizar la integridad referencial y la consistencia. Se emplearon claves primarias y foráneas para asegurar relaciones sólidas entre las tablas.

En cuanto al diseño de las relaciones, se han implementado **claves foráneas** en las tablas asignaciones_proyectos para vincular las asignaciones de empleados a proyectos específicos. Esta práctica no solo garantiza la integridad referencial, sino que también facilita la ejecución de consultas complejas mediante la combinación de múltiples tablas. Al aplicar estas claves foráneas, evitamos la creación de datos huérfanos y aseguramos que las asignaciones siempre correspondan a un empleado y un proyecto válidos.

Además, se ha considerado la escalabilidad de la base de datos. El uso de índices en las columnas más consultadas (como id_empleado y id_proyecto) permite optimizar el rendimiento de las consultas, lo que es crucial cuando el volumen de datos aumenta con el tiempo. El diseño también garantiza que la estructura sea lo suficientemente flexible para adaptarse a nuevas necesidades, como la inclusión de nuevas tablas o la modificación de las existentes sin afectar la integridad de los datos.

ELECCIÓN DE CLAVES PRIMARIAS

La elección de claves primarias es fundamental para garantizar la unicidad de los registros en cada tabla. En esta base de datos, se han elegido claves primarias numéricas (tipo INT) con la propiedad AUTO_INCREMENT. Esto permite asignar automáticamente un valor único a cada nuevo registro sin intervención manual, lo que reduce el riesgo de errores y garantiza que cada fila en la tabla sea identificable de manera exclusiva.

Por ejemplo, en la tabla empleados, la clave primaria es id_empleado, un campo entero que se incrementa automáticamente con cada nuevo empleado añadido. Este enfoque es adecuado porque los identificadores numéricos son rápidos de comparar y ocupan menos espacio en comparación con otros tipos de datos, como las cadenas de texto.

En las tablas de relaciones, como asignaciones_proyectos, la clave primaria es id_asignacion, que también sigue el mismo principio de auto-incremento, asegurando que cada asignación tenga un identificador único. Este diseño facilita la gestión y consulta de las asignaciones entre empleados y proyectos.

Además, las claves primarias han sido diseñadas para evitar la colisión de valores y, al mismo tiempo, mantener la integridad referencial cuando estas son usadas como claves foráneas en otras tablas. Esto simplifica la realización.

ANÁLISIS Y DISCUSIÓN

La implementación de la base de datos relacional gestion_empresarial ilustra cómo los conceptos de bases de datos relacionales se pueden aplicar eficazmente en un entorno de desarrollo real. A través del uso de DDL y DML, se ha logrado crear una estructura de datos bien definida, en la que las relaciones entre empleados, proyectos y asignaciones están claramente representadas.

Un aspecto clave es el uso de integridad referencial mediante claves foráneas. Esto asegura que las tablas estén correctamente relacionadas y evita errores como la eliminación de un empleado que esté asignado a un proyecto. El uso de consultas avanzadas, como las uniones (JOIN), permite extraer datos útiles de múltiples tablas de manera eficiente.

En cuanto a la optimización del rendimiento, se puede observar que la base de datos se beneficia del uso de índices en campos críticos, como las claves primarias y las claves foráneas. Estos índices permiten mejorar el tiempo de respuesta de las consultas, lo cual es esencial en bases de datos con un gran número de registros.

Sin embargo, se podrían considerar algunas mejoras, como la inclusión de índices compuestos en columnas que se consultan con frecuencia en conjunto, lo que podría mejorar aún más el rendimiento. Además, la utilización de triggers o disparadores podría automatizar ciertas acciones, como el registro automático de fechas de creación o modificación de registros.

En cuanto a la elección de los tipos de datos, se han utilizado tipos de datos eficientes, como INT para las claves primarias y VARCHAR para los textos, garantizando que se utilice el espacio adecuado en la base de datos. Esto es particularmente importante para mantener el rendimiento a medida que los datos crecen.

CONCLUSIÓN

La implementación de la base de datos relacional gestion_empresarial en MySQL muestra cómo los conceptos de DDL, DML, tipos de datos y consultas avanzadas son esenciales para gestionar eficazmente la información en un entorno empresarial. A través de un diseño adecuado, se ha logrado una estructura flexible, escalable y eficiente que permite la correcta administración de los datos de empleados, proyectos y asignaciones.

El uso de claves primarias y foráneas asegura la integridad referencial, mientras que las consultas avanzadas facilitan la obtención de información crítica para la toma de decisiones. La elección de los tipos de datos garantiza que la base de datos sea eficiente en términos de espacio y rendimiento.

La base de datos gestion_empresarial no solo cumple con los requisitos básicos de integridad y consistencia, sino que también está preparada para crecer y adaptarse a las necesidades futuras de la organización. No obstante, se sugiere implementar mejoras adicionales, como el uso de índices compuestos y triggers, para optimizar aún más el rendimiento y la automatización de procesos.

En definitiva, este proyecto es un claro ejemplo de cómo un enfoque metodológico en el diseño y gestión de bases de datos relacionales puede tener un impacto positivo en la eficiencia operativa de una empresa. Los principios aplicados, como la normalización, la integridad referencial y las buenas prácticas de diseño, son esenciales para asegurar que la base de datos continúe proporcionando valor a largo plazo.

REFERENCIAS

- Documentación oficial de MySQL.
- Tutoriales y guías de bases de datos relacionales
- https://github.com/yulycuadros6/repositorio-sql.git