DICCIONARIO DE OPERACIONES SQL

Miguel Ángel Tapiero Escobar, Deivison Vargas Mina, Jean Pier Garcia Garay.

ALMACENAMIENTO Y ESPACIO

1. **CREATE TABLESPACE**

    - **Definición**: Crea un espacio de almacenamiento donde se pueden guardar los datos de una base de datos.
    - **Para qué sirve**: Permite controlar dónde se almacenan los datos y gestionar mejor el espacio en disco.
    - **Ejemplo**:

```
CREATE TABLESPACE TblSpace_inventario ADD DATAFILE 'inventario.ibd' ENGINE=InnoDB;
```

Crea un espacio de almacenamiento llamado TblSpace_inventario y le asigna un archivo de datos.

2. **ALTER TABLESPACE**

    - **Definición**: Modifica un espacio de almacenamiento existente.
    - **Para qué sirve**: Se usa para cambiar la configuración de los espacios de almacenamiento, como renombrarlos o modificar sus rutas.
    - **Ejemplo**:

```
ALTER TABLESPACE TblSpace_inventario RENAME TO TblSpace_stock;
```

Renombra el espacio de almacenamiento TblSpace_inventario a TblSpace_stock.

3. **SHOW TABLES**

    - **Definición**: Muestra las tablas de una base de datos.
    - **Para qué sirve**: Permite ver todas las tablas presentes en una base de datos y verificar el espacio utilizado.

- **Ejemplo**:

SHOW TABLES IN Datacenter;

Muestra todas las tablas dentro de la base de datos Datacenter.

4. **SHOW DATABASES**

   - **Definición**: Muestra todas las bases de datos disponibles en el servidor.
   - **Para qué sirve**: Permite auditar la estructura de las bases de datos disponibles y sus respectivos tamaños.
   - **Ejemplo**:

SHOW DATABASES;

Muestra todas las bases de datos existentes en el servidor MySQL.

5. **CHECK TABLE**

   - **Definición**: Revisa la integridad de una tabla, buscando posibles corrupciones.
   - **Para qué sirve**: Es útil para detectar problemas en los datos o en la estructura de las tablas.
   - **Ejemplo**:

CHECK TABLE empleados;

Verifica la integridad de la tabla empleados.

6. **REPAIR TABLE**

   - **Definición**: Repara una tabla que esté dañada o corrupta.
   - **Para qué sirve**: Se usa para recuperar tablas que presentan errores de estructura o de datos.
   - **Ejemplo**:

REPAIR TABLE empleados;

Repara la tabla empleados si presenta daños.

//

MEMORIA Y RENDIMIENTO

1. **SHOW STATUS**
    - **Definición**: Muestra información sobre el estado actual del servidor, incluyendo métricas de rendimiento.
    - **Para qué sirve**: Es útil para monitorear la carga del servidor y detectar posibles cuellos de botella o problemas de rendimiento.
    - **Ejemplo**:

SHOW STATUS LIKE 'Threads_connected';

Muestra el número de conexiones activas al servidor MySQL.

2. **SHOW VARIABLES**
    - **Definición**: Muestra las variables de configuración del servidor, como las relacionadas con el tamaño de la memoria y el comportamiento de las conexiones.
    - **Para qué sirve**: Se utiliza para revisar los parámetros del servidor, lo que es útil para optimizar el uso de los recursos.
    - **Ejemplo**:

SHOW VARIABLES LIKE 'innodb_buffer_pool_size';

Muestra el tamaño de la memoria asignada a InnoDB, útil para ajustar el rendimiento de las consultas.

3. **SET GLOBAL**
    - **Definición**: Modifica una variable de configuración global del servidor.

- **Para qué sirve**: Se usa para ajustar parámetros del servidor como el uso de memoria, optimización de consultas, etc.
- **Ejemplo**:

SET GLOBAL innodb_buffer_pool_size = 268435456;

Esto ajusta el tamaño del pool de memoria de InnoDB a 256 MB.

4. **ANALYZE TABLE**
    - **Definición**: Recalcula las estadísticas de una tabla, lo que ayuda a optimizar el plan de ejecución de las consultas.
    - **Para qué sirve**: Mejora el rendimiento de las consultas al actualizar las estadísticas internas de las tablas.
    - **Ejemplo**:

ANALYZE TABLE empleados;

Esto recalcula las estadísticas de la tabla empleados para mejorar el rendimiento en futuras consultas.

5. **OPTIMIZE TABLE**
    - **Definición**: Optimiza una tabla eliminando espacios de almacenamiento no utilizados y defragmentando los datos.
    - **Para qué sirve**: Mejora la velocidad de las consultas al reorganizar los datos en la tabla.
    - **Ejemplo**:

OPTIMIZE TABLE empleados;

Esto optimiza la tabla empleados para mejorar su rendimiento.

6. **PERFORMANCE_SCHEMA**

- **Definición**: Proporciona información detallada sobre el rendimiento de las consultas, incluyendo el tiempo que tardan y los recursos que consumen.
- **Para qué sirve**: Ayuda a identificar cuellos de botella en las consultas y optimizar el rendimiento de la base de datos.
- **Ejemplo**:

SELECT * FROM performance_schema.events_statements_summary_by_digest;

Muestra un resumen de las declaraciones SQL ejecutadas y sus tiempos de ejecución.

//

GESTIÓN DE USUARIOS Y PERMISOS

1. **CREATE USER**
   - **Definición**: Se usa para crear una nueva cuenta de usuario en MySQL, asignando un nombre de usuario, un host y una contraseña.
   - **Para qué sirve**: Se usa para dar acceso controlado a la base de datos (BD), permitiendo que usuarios específicos accedan y operen sobre ella.
   - **Ejemplo**:

CREATE USER 'Alfredo'@'%' IDENTIFIED BY 'contrasena321';

Esto crea un usuario llamado 'Alfredo' con acceso desde cualquier host (%), con la contraseña contrasena321.

2. **DROP USER**
   - **Definición**: Elimina una cuenta de usuario existente.
   - **Para qué sirve**: Se usa para borrar cuentas obsoletas o inseguras, eliminando acceso no autorizado.
   - **Ejemplo**:

```
DROP USER 'Alfredo'@'%';
```

Este comando elimina el usuario 'Alfredo' desde cualquier host.

3. **ALTER USER**
    - **Definición**: Modifica la configuración de un usuario, como bloquear la cuenta, cambiar la contraseña, entre otros.
    - **Para qué sirve**: Se utiliza para ajustar configuraciones de usuarios existentes, por ejemplo, bloquear cuentas inactivas.
    - **Ejemplo**:

```
ALTER USER 'Alfredo'@'%' ACCOUNT LOCK;
```

Esto bloquea la cuenta del usuario 'Alfredo', impidiendo su acceso.

4. **SET PASSWORD**
    - **Definición**: Cambia la contraseña de un usuario.
    - **Para qué sirve**: Se usa para mantener la seguridad de las contraseñas y realizar cambios de manera controlada.
    - **Ejemplo**:

```
SET PASSWORD FOR 'Alfredo'@'%' = 'NuevaContrasena';
```

Esto cambia la contraseña del usuario 'Alfredo' a NuevaContrasena.

5. **RENAME USER**
    - **Definición**: Renombra a un usuario en la base de datos.
    - **Para qué sirve**: Se utiliza para corregir errores en nombres de usuarios o para reorganizar cuentas de forma más clara.
    - **Ejemplo**:

RENAME USER 'Alfredo'@'%' TO 'Manuel'@'%';

Cambia el nombre del usuario 'Alfredo' a 'Manuel'.

6. **GRANT**
    - **Definición**: Asigna permisos a un usuario o rol.
    - **Para qué sirve**: Se utiliza para autorizar a un usuario a realizar acciones específicas en una base de datos, como consultas o actualizaciones.
    - **Ejemplo**:

GRANT SELECT, INSERT ON Datacenter.* TO 'Alfredo'@'%';

Esto otorga a 'Alfredo' permisos para realizar consultas (SELECT) y hacer inserciones (INSERT) en todas las tablas de la base de datos Datacenter.

7. **REVOKE**
    - **Definición**: Revoca permisos previamente otorgados a un usuario o rol.
    - **Para qué sirve**: Se usa para restringir acciones que un usuario ya no debe poder realizar, como eliminar permisos de inserción en una tabla.
    - **Ejemplo**:

REVOKE INSERT ON Datacenter.empleados FROM 'Alfredo'@'%';

Revoca el permiso de inserción en la tabla empleados de la base de datos Datacenter para el usuario 'Alfredo'.

8. **CREATE ROLE**
    - **Definición**: Crea un rol que agrupe varios permisos.
    - **Para qué sirve**: Se usa para gestionar permisos de forma eficiente al asignar un conjunto común de permisos a varios usuarios.
    - **Ejemplo**:

CREATE ROLE 'lectura';

Crea un rol llamado lectura.

9. **GRANT ROLE**
    - **Definición**: Asigna un rol a uno o más usuarios.
    - **Para qué sirve**: Permite la asignación masiva de permisos, haciendo más fácil administrar accesos.
    - **Ejemplo**:

GRANT 'lectura' TO 'Andrea'@'%';

Asigna el rol lectura al usuario 'Andrea', dándole acceso solo de lectura.

10. **REVOKE ROLE**
    - **Definición**: Revoca un rol de un usuario.
    - **Para qué sirve**: Elimina los permisos asociados a un rol previamente asignado a un usuario.
    - **Ejemplo**:

REVOKE 'lectura' FROM 'Andrea'@'%';

Esto elimina el rol lectura del usuario 'Andrea'.

11. **SHOW GRANTS**
    - **Definición**: Muestra los permisos asignados a un usuario.
    - **Para qué sirve**: Se usa para auditar qué permisos tiene un usuario, asegurando que solo pueda realizar las acciones autorizadas.
    - **Ejemplo**:

SHOW GRANTS FOR 'Alfredo'@'%';

Muestra los permisos otorgados al usuario 'Alfredo'.

12. **FLUSH PRIVILEGES**

    - **Definición**: Recarga los privilegios para que los cambios en permisos tomen efecto.
    - **Para qué sirve**: Se usa para aplicar cambios en la gestión de usuarios y permisos sin reiniciar el servidor.
    - **Ejemplo**:

FLUSH PRIVILEGES;

Recarga los privilegios, aplicando inmediatamente los cambios realizados.

13. **SHOW USERS (manual)**

    - **Definición**: Muestra una lista de los usuarios existentes en el sistema.
    - **Para qué sirve**: Permite ver todas las cuentas de usuario registradas en el servidor.
    - **Ejemplo**:

SELECT user, host FROM mysql.user;

Muestra todos los usuarios y los host desde los que pueden acceder.

//

RESPALDO Y RECUPERACIÓN

1. **BACKUP (mysqldump)**

    - **Definición**: Crea una copia de seguridad de la base de datos utilizando mysqldump, una herramienta externa a MySQL.
    - **Para qué sirve**: Se usa para realizar respaldos completos o parciales de las bases de datos, para proteger la información ante posibles fallos.
    - **Ejemplo**:

mysqldump -u root -p inventario > inventario.sql

Crea una copia de seguridad de la base de datos inventario en un archivo .sql.

2. **RESTORE (mysql)**
    - **Definición**: Restaura una base de datos desde un archivo de respaldo previamente realizado.
    - **Para qué sirve**: Se utiliza para recuperar datos después de una pérdida o fallo del sistema.
    - **Ejemplo**:

mysql -u root -p inventario < inventario.sql

Restaura la base de datos inventario desde el archivo de respaldo inventario.sql.

3. **EXPORT TABLE**
    - **Definición**: Realiza una exportación de una tabla específica utilizando mysqldump.
    - **Para qué sirve**: Es útil cuando necesitas hacer respaldos o migraciones parciales, solo de una tabla específica.
    - **Ejemplo**:

mysqldump -u root -p Datacenter empleados > empleados.sql

Exporta solo la tabla empleados de la base de datos Datacenter a un archivo .sql.

4. **RESTAURAR TABLA**
    - **Definición**: Similar a la restauración de bases de datos, pero se realiza solo para una tabla específica.
    - **Para qué sirve**: Permite recuperar únicamente una tabla seleccionada de un respaldo, sin afectar el resto de la base de datos.
    - **Ejemplo**:

      Procedimiento manual utilizando mysqldump y luego importando la tabla:

mysqldump -u root -p Datacenter empleados > empleados_backup.sql

mysql -u root -p Datacenter < empleados_backup.sql

Esto realiza una restauración de la tabla empleados de la base de datos Datacenter.

//

MANTENIMIENTO Y OPTIMIZACIÓN

1. **ALTER TABLE**
    - **Definición**: Modifica la estructura de una tabla existente.
    - **Para qué sirve**: Se utiliza para agregar, eliminar o modificar columnas, índices o claves en una tabla sin tener que recrearla.
    - **Ejemplo**:

ALTER TABLE empleados ADD COLUMN edad INT;

Agrega una columna edad de tipo entero a la tabla empleados.

2. **CREATE INDEX**
    - **Definición**: Crea un índice en una tabla para mejorar la velocidad de las consultas.
    - **Para qué sirve**: Se usa para acelerar las búsquedas y consultas que involucren las columnas indexadas.
    - **Ejemplo**:

CREATE INDEX idx_name ON empleados(nombre);

Crea un índice llamado idx_name en la columna nombre de la tabla empleados.

3. **DROP INDEX**
    - **Definición**: Elimina un índice de una tabla.

- **Para qué sirve**: Se utiliza para optimizar el rendimiento del sistema cuando un índice ya no es necesario o es ineficiente.
- **Ejemplo**:

DROP INDEX idx_name ON empleados;

Elimina el índice idx_name de la tabla empleados.

4. **TRUNCATE TABLE**
    - **Definición**: Elimina todos los registros de una tabla de manera rápida y sin registrar la eliminación individual de cada fila.
    - **Para qué sirve**: Se usa para limpiar la tabla de manera eficiente, sin generar registros en el log de transacciones para cada fila.
    - **Ejemplo**:

TRUNCATE TABLE empleados;

Elimina todos los registros de la tabla empleados.

5. **DELETE FROM**
    - **Definición**: Elimina registros específicos de una tabla basándose en una condición.
    - **Para qué sirve**: Se usa para eliminar filas individuales que cumplan con un criterio específico, con la posibilidad de registrar cada eliminación en los logs.
    - **Ejemplo**:

DELETE FROM empleados WHERE edad > 60;

Elimina todos los registros de la tabla empleados donde la edad sea mayor a 60.

//

SEGURIDAD

1. **ENCRYPTION**
    - **Definición**: Activa el cifrado de datos para una tabla específica.
    - **Para qué sirve**: Se utiliza para proteger datos sensibles en la base de datos, asegurando que no puedan ser leídos fácilmente sin la clave adecuada.
    - **Ejemplo**:

ALTER TABLE empleados ENCRYPTION='Y';

Esto activa el cifrado en la tabla empleados, lo que asegura que los datos estén cifrados en disco.

2. **CREATE AUDIT LOG**
    - **Definición**: Instala un plugin de auditoría para registrar eventos importantes.
    - **Para qué sirve**: Permite auditar los eventos y acciones dentro del sistema MySQL para fines de seguridad, como accesos, modificaciones y errores.
    - **Ejemplo**:

INSTALL PLUGIN audit_log SONAME 'audit_log.so';

Esto instala el plugin audit_log que guarda un registro de las acciones realizadas.

3. **SHOW MASTER STATUS**
    - **Definición**: Muestra el estado actual de los registros binarios, útiles para la replicación de MySQL.
    - **Para qué sirve**: Se usa para monitorear el estado de la replicación maestro-esclavo, verificando la sincronización de los datos.
    - **Ejemplo**:

SHOW MASTER STATUS;

Muestra la información sobre el archivo de registro binario y el punto de replicación actual.

4. **SHOW VARIABLES (auth)**

    - **Definición**: Muestra las variables de configuración relacionadas con la autenticación.
    - **Para qué sirve**: Ayuda a revisar las políticas de seguridad de contraseñas y otros aspectos relacionados con la autenticación de usuarios.
    - **Ejemplo**:

SHOW VARIABLES LIKE 'validate_password%';

Muestra las configuraciones de la política de contraseñas (por ejemplo, requisitos de longitud mínima o complejidad).

5. **SET GLOBAL validate_password_policy**

    - **Definición**: Ajusta la política de contraseñas global en MySQL.
    - **Para qué sirve**: Permite configurar las políticas de seguridad de contraseñas, como la complejidad o longitud mínima.
    - **Ejemplo**:

SET GLOBAL validate_password_policy = 'STRONG';

Esto establece una política de contraseñas fuerte, exigiendo que las contraseñas cumplan con un conjunto más estricto de reglas.

//

REPLICACIÓN Y ALTA DISPONIBILIDAD

1. **CHANGE MASTER TO**

    - **Definición**: Configura la replicación entre un servidor maestro y un servidor esclavo en MySQL.
    - **Para qué sirve**: Se utiliza para sincronizar los datos entre servidores, asegurando la alta disponibilidad y la redundancia.

- **Ejemplo**:

CHANGE MASTER TO MASTER_HOST='192.168.0.10', MASTER_USER='repl',

MASTER_PASSWORD='pass';

//

MONITOREO Y AUDITORÍA

1. **SHOW PROCESSLIST**

    - **Definición**: Muestra los procesos activos en el servidor MySQL, como las consultas que están en ejecución.
    - **Para qué sirve**: Permite identificar procesos que están ocupando muchos recursos o bloqueando otras consultas, ayudando a detectar cuellos de botella.
    - **Ejemplo**:

SHOW PROCESSLIST;

Muestra todos los procesos activos en el servidor MySQL.

2. **KILL**

    - **Definición**: Termina un proceso específico en el servidor MySQL utilizando su ID.
    - **Para qué sirve**: Se usa para liberar recursos de procesos que están bloqueados o son inactivos.
    - **Ejemplo**:

KILL 1109;

Termina el proceso con ID 1109 en el servidor MySQL.

3. **SHOW EVENTS**

    - **Definición**: Muestra los eventos programados en MySQL.

- **Para qué sirve**: Permite auditar las tareas programadas, como eventos de mantenimiento o automatización.
- **Ejemplo**:

SHOW EVENTS;

Muestra los eventos programados en la base de datos.

4. **INFORMATION_SCHEMA**
    - **Definición**: Proporciona acceso a una base de datos virtual que contiene información sobre las tablas, columnas, índices y otros objetos de la base de datos.
    - **Para qué sirve**: Es útil para auditar la configuración de la base de datos y obtener detalles sobre su estructura.
    - **Ejemplo**:

SELECT * FROM information_schema.tables WHERE table_schema='Datacenter';

Muestra todas las tablas de la base de datos Datacenter.

//

AUTOMATIZACIÓN Y PROGRAMACIÓN

1. **CREATE EVENT**
    - **Definición**: Crea un evento programado que se ejecutará en un momento específico o de forma repetitiva.
    - **Para qué sirve**: Permite automatizar tareas de mantenimiento o acciones repetitivas en la base de datos, como limpiar registros inactivos.
    - **Ejemplo**:

CREATE EVENT Clear_Inactivos ON SCHEDULE EVERY 1 DAY DO DELETE FROM usuarios WHERE activo=0;

Crea un evento llamado Clear_Inactivos que se ejecuta todos los días para eliminar usuarios inactivos.

2. **CREATE TRIGGER**

    - **Definición**: Crea un disparador (trigger) que se ejecuta automáticamente antes o después de una operación (INSERT, UPDATE, DELETE) sobre una tabla.
    - **Para qué sirve**: Permite monitorear cambios en la base de datos y realizar acciones automáticas, como auditorías o ajustes de datos.
    - **Ejemplo**:

CREATE TRIGGER trg_borrar BEFORE DELETE ON empleados FOR EACH ROW INSERT INTO auditoria VALUES (OLD.id, NOW());

Crea un disparador que inserta un registro en la tabla auditoria antes de eliminar un registro de la tabla empleados.

3. **CREATE PROCEDURE**

    - **Definición**: Crea un procedimiento almacenado, que es un conjunto de instrucciones SQL que pueden ejecutarse de forma repetida.
    - **Para qué sirve**: Se usa para automatizar lógicas complejas y evitar la repetición de código SQL.
    - **Ejemplo**:

CREATE PROCEDURE Actualizar_Salarios() BEGIN UPDATE empleados SET salario = salario * 1.03; END;

Crea un procedimiento que aumenta el salario de todos los empleados en un 3%.

4. **CREATE VIEW**

    - **Definición**: Crea una vista, que es una consulta almacenada que presenta datos de una o varias tablas como si fuera una tabla.

- **Para qué sirve**: Simplifica consultas frecuentes y presenta una interfaz más sencilla para los usuarios.
- **Ejemplo**:

CREATE VIEW Empleados_activos AS SELECT * FROM empleados WHERE activo=1;

Crea una vista llamada Empleados_activos que solo muestra los empleados activos de la tabla empleados.

//

PREGUNTAS DATA-CENTER

DEIVISON

1) ¿Cómo abordan los desafíos de latencia en la replicación de bases de datos distribuidas geográficamente y qué mecanismos implementan para garantizar la consistencia eventual o fuerte según el caso?

R Para minimizar el impacto de la latencia en réplicas distribuidas, suelen optar por consistencia eventual en aquellas aplicaciones donde la sincronización inmediata no es indispensable, de modo que el sistema mantiene un alto rendimiento sin esperar a que todos los nodos estén alineados al instante. En cambio, en entornos de misión crítica, como ejemplo: las transacciones bancarias, se prefiere una consistencia fuerte, aceptando un mayor tiempo de respuesta para garantizar que todos los nodos compartan exactamente el mismo estado antes de continuar. Adicionalmente, aplican la conocida “regla 3-2-1-1-0” para asegurar recuperación y consistencia: conservan al menos tres copias de los datos, almacenadas en dos tipos de soportes distintos, y mantienen una de ellas fuera del sitio principal. Así, ante cualquier fallo, pueden restaurar rápidamente una versión íntegra y actualizada de la información.

2) ¿Cuáles son los principales controles de seguridad implementados en el datacenter Netgroup para proteger las bases de datos contra accesos no autorizados?

R Para proteger las bases de datos, Netgroup aplica un enfoque multicapa. Primero, implementan mecanismos rigurosos de autenticación y autorización, de modo que solo usuarios verificados y con

permisos explícitos puedan acceder a los datos. A esto suman monitoreo y auditoría continua: todas las acciones quedan registradas y analizadas en tiempo real para detectar patrones sospechosos y responder de inmediato. Asimismo, mantienen la seguridad física del centro de datos mediante controles de acceso biométricos y cámaras de vigilancia que impiden intrusiones al recinto. Por último, y para garantizar tanto integridad como disponibilidad, siguen la estrategia 3-2-1-1-0 de copias de respaldo —tres réplicas en dos tipos de medios distintos, con al menos una copia fuera de las instalaciones—, lo que asegura que cualquier incidente no comprometa la información ni su recuperación.

3) ¿Qué tipo de racks o gabinetes utilizan y cómo se gestionan los espacios físicos y el cableado estructurado?

R En nuestro centro de datos empleamos armarios rack cerrados de 42 U, con puertas frontal y trasera perforadas para optimizar la seguridad física y el flujo de aire. Cada fila de racks se organiza en pasillos fríos y calientes, de modo que el aire frío entra por la parte frontal de los equipos y el aire caliente se expulsa por la trasera, facilitando tanto la refrigeración como el acceso para tareas de mantenimiento sin obstáculos.

El cableado estructurado se distribuye a través de bandejas horizontales y verticales, con canaletas y guías que mantienen los cables separados por función (datos, voz, fibra) y los agrupan por colores siguiendo la norma EIA/TIA-568. Cada extremo está etiquetado en patch panels y en los puertos de los switches, lo que permite localizar rápidamente cualquier conexión. Además, se utilizan bridas de Velcro y organizadores de cables fáciles de abrir, lo que reduce el riesgo de dañar los conductores al mover o añadir equipos y agiliza cualquier intervención.

JEAN PIER

1) ¿Qué mecanismos de monitoreo utilizan para detectar fallos o caídas en las bases de datos?

R Se utilizan diferentes mecanismos de monitoreo, pero estos varían entre cada cliente, debido a que son los propios clientes los que deciden que mecanismos utilizar, aunque sin duda, el más utilizado por Netgroup es Centreon.

2) ¿Qué consideraciones toman respecto al ciclo de vida de la información (creación, uso, almacenamiento y eliminación) bajo la norma ISO 27001?

R Netgroup hasta el momento no cuenta con una certificación de la ISO 27001, sin embargo, al contar con una certificación de la PCI, si cumple la mayoría de reglamentos y normativas para el ciclo de vida de la información.

3) ¿Qué mecanismos utilizan para la redundancia física de los servidores y del almacenamiento, en caso de fallos de hardware?

R Se tienen muchas contramedidas en casos del hardware, pero la mayor y mas importante es la replicación en todas sus sedes, incluyendo sedes fuera del país, lo que proporciona una red de seguridad inmensa ya que, si por alguna razón o circunstancia catastrófica una de sus sedes se volviera inoperativa, la replicación permitiría conservar los datos de todos sus clientes en las otras sedes.

MIGUEL ANGEL

1) ¿Qué estrategias aplican para evitar la fragmentación en bases de datos de alto volumen transaccional, y cómo identifican cuándo es necesario realizar reestructuración física?

R Se utilizan bases de datos como Oracle y SQL Server. Durante las ventanas de mantenimiento se ejecutan procesos de desfragmentación, y luego se reconstruyen los índices ya que suelen corromperse. También se utilizan estadísticas internas que permiten identificar el comportamiento de las consultas por tabla, lo que ayuda a determinar cuándo es necesaria una reestructuración física. En casos de fragmentación avanzada, se hacen respaldos y se rehace la base desde cero para optimizar su rendimiento.

2) 2. ¿Qué políticas existen para la gestión de cuentas de usuarios con privilegios elevados (DBA, administradores de red, etc.)?

R Se aplica control de acceso lógico mediante la gestión de roles, siguiendo normas de seguridad como PCI o en gran medida depende de los lineamientos del cliente. Cada perfil tiene permisos específicos según sus funciones. Esto permite controlar el acceso y asegurar la trazabilidad de las acciones realizadas por usuarios con privilegios elevados.

3) ¿Cómo se manejan las labores de mantenimiento físico preventivo en servidores, dispositivos de red y sistemas de enfriamiento?

R Para los servidores, se utilizan servidores de migración donde se despliegan máquinas virtuales a partir de respaldos hechos con QuickBackup, lo que permite hacer mantenimiento sin afectar la operación. En el caso de los sistemas de enfriamiento, se cuenta con equipos redundantes que entran en funcionamiento mientras se da mantenimiento a los otros. Esta misma lógica se aplica a dispositivos de red para garantizar la disponibilidad del servicio.