Análisis de Datos

Nivel Innovador – Avanzado

Misión 1 - Tutorial

Técnicas Avanzadas de Actualización y Replicación

Introducción

En la gestión de bases de datos, las técnicas avanzadas de actualización y replicación son esenciales para mantener la integridad, disponibilidad y rendimiento de los datos. Este documento cubre las técnicas avanzadas de actualización utilizando las sentencias UPDATE y MERGE, así como conceptos y estrategias de replicación de datos. Se incluyen ejemplos prácticos para ilustrar cada técnica.

1. Técnicas Avanzadas de Actualización

1.1 UPDATE con Condiciones Complejas

La sentencia UPDATE se utiliza para modificar los datos existentes en una tabla. Las condiciones complejas permiten realizar actualizaciones más sofisticadas, basadas en múltiples criterios.

Ejemplo: **Actualización basada en JOIN** Este ejemplo muestra cómo actualizar los salarios de los empleados en el departamento de ventas (Sales), incrementándolos en un 10%.

```
UPDATE employees
SET salary = salary * 1.1
FROM employees e
JOIN departments d ON e.department_id = d.department_id
WHERE d.department_name = 'Sales';
```

Explicación:

- La cláusula FROM permite especificar una tabla adicional (departments) para realizar un JOIN.
- La condición WHERE asegura que solo se actualicen los empleados del departamento de ventas.

Ejemplo: **Actualización con Subconsultas** Este ejemplo muestra cómo actualizar los salarios de los empleados cuyo salario está por debajo del salario promedio de su departamento.

```
UPDATE employees
SET salary = salary + 1000
WHERE salary < (SELECT AVG(salary)
                FROM employees
                WHERE department_id = employees.department_id);
```

Explicación:

- La subconsulta en la cláusula WHERE calcula el salario promedio por departamento.
- Solo los empleados con salarios por debajo de este promedio reciben un aumento.

1.2 MERGE con Condiciones Complejas

La sentencia MERGE combina la funcionalidad de INSERT, UPDATE y DELETE en una sola operación, permitiendo la sincronización de datos entre tablas.

Ejemplo: MERGE para Sincronización de Datos Este ejemplo sincroniza los datos de una tabla de origen (source_table) con una tabla de destino (target_table).

```
MERGE INTO target_table AS target
USING source_table AS source
ON target.id = source.id
WHEN MATCHED THEN
   UPDATE SET target.name = source.name
WHEN NOT MATCHED BY TARGET THEN
   INSERT (id, name) VALUES (source.id, source.name)
WHEN NOT MATCHED BY SOURCE THEN
   DELETE;
```

Explicación:

- La cláusula USING especifica la tabla de origen.
- La condición ON define cómo se relacionan las filas de ambas tablas.
- WHEN MATCHED THEN UPDATE actualiza las filas que coinciden.
- WHEN NOT MATCHED BY TARGET THEN INSERT inserta nuevas filas que no existen en la tabla de destino.
- WHEN NOT MATCHED BY SOURCE THEN DELETE elimina filas que ya no existen en la tabla de origen.

---

2. Técnicas de Replicación de Datos

2.1 Conceptos de Replicación

La replicación de datos es el proceso de duplicar y mantener los datos en múltiples ubicaciones para mejorar la disponibilidad y el rendimiento.

- **Replicación Síncrona vs. Asíncrona:**
  - **Replicación Síncrona:** Los datos se copian a las réplicas en tiempo real. Ventaja: Garantiza consistencia. Desventaja: Puede introducir latencia.
  - **Replicación Asíncrona:** Los datos se copian a las réplicas con un retraso. Ventaja: Mejor rendimiento. Desventaja: Riesgo de inconsistencias temporales.

- **Replicación de Base de Datos Completa vs. Parcial:**
  - **Completa:** Toda la base de datos se replica.
  - **Parcial:** Solo una parte de la base de datos (por ejemplo, ciertas tablas o filas) se replica.

2.2 Estrategias de Replicación

Replicación Maestra-Esclava (Master-Slave)

- **Descripción:** Un servidor maestro replica sus datos a uno o más servidores esclavos.
- **Ventajas:** Fácil configuración, adecuado para cargas de lectura pesada.
- **Desventajas:** El maestro es un único punto de fallo, retrasos en la replicación asíncrona.

Ejemplo de Configuración en MySQL:

- Configurar el servidor maestro

```
[mysqld]
log-bin=mysql-bin
server-id=1
```

- Configurar el servidor esclavo

```
[mysqld]
server-id=2
replicate-do-db=mydatabase
```

- En el maestro, crear un usuario de replicación

```
CREATE USER 'replicator'@'%' IDENTIFIED BY 'password';
GRANT REPLICATION SLAVE ON *.* TO 'replicator'@'%';
```

- En el esclavo, configurar la replicación

```
CHANGE MASTER TO
MASTER_HOST='master_host',
MASTER_USER='replicator',
MASTER_PASSWORD='password',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=4;
START SLAVE;
```

Replicación Maestra-Maestra (Master-Master)

- **Descripción:** Dos o más servidores maestros replican sus datos entre sí.
- **Ventajas:** Alta disponibilidad, balanceo de carga.
- **Desventajas:** Configuración más compleja, manejo de conflictos.

Ejemplo de Configuración en MySQL:

- Configurar ambos servidores como maestros

```
[mysqld]
log-bin=mysql-bin
server-id=1
```

- Configurar la replicación mutua

```
CHANGE MASTER TO
MASTER_HOST='master2_host',
MASTER_USER='replicator',
MASTER_PASSWORD='password',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=4;
START SLAVE;
```

3. Uso de Triggers y Procedimientos Almacenados

3.1 Triggers

Los triggers son procedimientos que se ejecutan automáticamente en respuesta a ciertos eventos en una tabla o vista.

Ejemplo de Trigger: **Actualización de Timestamp** Este trigger actualiza una columna de timestamp cada vez que se modifica una fila en la tabla employees.

```
CREATE TRIGGER update_timestamp
BEFORE UPDATE ON employees
FOR EACH ROW
EXECUTE PROCEDURE update_modified_column();
```

Procedimiento:

```
CREATE OR REPLACE FUNCTION update_modified_column()
RETURNS TRIGGER AS $$
BEGIN
   NEW.modified = NOW();
   RETURN NEW;
```

```
END;
$$ LANGUAGE plpgsql;
```

Explicación:

- El trigger BEFORE UPDATE se ejecuta antes de una operación de actualización.
- El procedimiento update_modified_column actualiza la columna modified con la fecha y hora actual.

3.2 Procedimientos Almacenados

Los procedimientos almacenados son conjuntos de instrucciones SQL que se almacenan en la base de datos y pueden ser ejecutados por aplicaciones.

Ejemplo de Procedimiento Almacenado: **Actualización de Salario** Este procedimiento actualiza el salario de un empleado dado su ID y el nuevo salario.

```
CREATE PROCEDURE update_salary(IN employee_id INT, IN new_salary DECIMAL)
BEGIN
   UPDATE employees
   SET salary = new_salary
   WHERE id = employee_id;
END;
```

Explicación:

- El procedimiento update_salary toma dos parámetros de entrada: employee_id y new_salary.
- Ejecuta una sentencia UPDATE para cambiar el salario del empleado con el ID proporcionado.

Conclusión

Las técnicas avanzadas de actualización y replicación de datos son cruciales para gestionar eficientemente las bases de datos en entornos modernos. El uso de

UPDATE y MERGE con condiciones complejas permite realizar actualizaciones precisas y efectivas. La replicación de datos, tanto en configuraciones Maestra-Esclava como Maestra-Maestra, mejora la disponibilidad y el rendimiento. Finalmente, los triggers y procedimientos almacenados proporcionan mecanismos poderosos para automatizar y asegurar la integridad de los datos.