Estrategias procedimentales para optimización de bases de datos

Procedural strategies for data bases optimization

Adriana Lucia Caicedo López[1]

Mario Luis Ávila Pérez[2]

Universidad Nacional Abierta y a Distancia, Colombia

Resumen

El lenguaje procedimental en los sistemas de gestión de bases de datos (DBMS) permite la creación de procedimientos y funciones para encapsular la lógica de negocio, facilitando el acceso y manipulación de datos de forma eficiente. A diferencia de los lenguajes declarativos, como SQL, los lenguajes procedimentales ofrecen control de flujo y estructuras avanzadas de programación, como en PL/SQL para Oracle y Transact-SQL (T-SQL) para SQL Server. Estos lenguajes permiten el desarrollo de procedimientos almacenados, funciones definidas por el usuario (UDF) y triggers, los cuales automatizan tareas repetitivas y complejas dentro de la base de datos. Las UDF proporcionan modularidad y reutilización de código, aunque pueden impactar el rendimiento en grandes volúmenes de datos. Además, los triggers responden automáticamente a eventos específicos, como inserciones o eliminaciones, y están clasificados en triggers DDL y DML. Estos componentes optimizan el rendimiento de la base de datos al reducir el tráfico de red y aumentar la eficiencia del procesamiento de datos.

Palabras clave: lenguaje procedimental, mantenimiento, seguridad, optimización.

Abstract

Procedural language in database management systems (DBMS) enables the creation of procedures and functions to encapsulate business logic, facilitating efficient data access and manipulation. Unlike declarative languages, such as SQL, procedural languages offer control flow and advanced programming structures, like those found in PL/SQL for Oracle and Transact-SQL (T-SQL) for SQL Server. These languages support the development of stored procedures, user-defined functions (UDF), and triggers, which automate repetitive and complex tasks within the

[1] Estudiante ingeniería de sistemas, UNAD. https://orcid.org/0009-0002-5685-2731 / adrianaluciacl21@gmail.com

[2] Ingeniero de sistemas, UNAD. https://orcid.org/0000-0002-7834-3578 / mavilap@gmail.com

database. UDF provide modularity and code reuse, though they may impact performance in large data volumes. Additionally, triggers respond automatically to specific events, such as inserts or deletions, and are classified into DDL and DML triggers. These components optimize database performance by reducing network traffic and increasing data processing efficiency.

Keywords: Procedural language, maintenance, security, optimization.

1. Introducción

El lenguaje procedimental en los sistemas de gestión de bases de datos (DBMS) es fundamental para crear procedimientos y funciones que encapsulan la lógica de negocio, facilitando un acceso y manipulación de datos eficientes. A diferencia de los lenguajes declarativos, como SQL, los lenguajes procedimentales proporcionan control de flujo y estructuras avanzadas de programación, como se observa en PL/SQL para Oracle y Transact-SQL (T-SQL) para SQL Server. Estos lenguajes permiten el desarrollo de procedimientos almacenados, funciones definidas por el usuario (UDF) y triggers, que automatizan tareas repetitivas y complejas dentro de la base de datos.

2. Desarrollo

El lenguaje procedimental se utiliza en la gestión de bases de datos para crear procedimientos y funciones que encapsulan la lógica del negocio, facilitando la manipulación y acceso a los datos. A diferencia de los lenguajes declarativos, los lenguajes procedimentales permiten control de flujo y estructuras de programación avanzadas (Zygiaris, 2018).

Los lenguajes procedimentales más populares son PL/SQL usado en Oracle, Transact-SQL (T-SQL) Usado en Microsoft SQL Server y PL/pgSQL utilizado en PostgreSQL, su sistema MVCC permite el acceso concurrente sin bloqueos, superando las limitaciones de otras bases de datos (Camuña Rodríguez, 2015).

3. Funciones definidas por el usuario UDF

Las UDF son bloques de construcción personalizados que podemos agregar a nuestro lenguaje SQL. Al igual que las piezas de Lego, estas funciones se pueden combinar de diversas maneras para crear estructuras más complejas. Podemos construir funciones sencillas que realicen cálculos básicos, o crear funciones más sofisticadas que implementen algoritmos complejos. La clave está en definir

correctamente el tipo de retorno de la función, ya sea un valor escalar, una fila o una tabla completa (IBM, s.f.).

```
CREATE FUNCTION CalcularPrecioTotal (@precio FLOAT, @impuesto
FLOAT)
RETURNS FLOAT
AS
BEGIN
   DECLARE @precio_total FLOAT
   SET @precio_total = @precio + (@precio * @impuesto)
   RETURN @precio_total
END;
```

Figura 1. comando SQL. Nota: muestra un ejemplo de un comando SQL que ejecuta la creación de una función.

Las funciones definidas por el usuario (UDF) ofrecen varias ventajas en la gestión de bases de datos. Promueven la modularidad y reutilización de código al permitir almacenar lógica compleja y reutilizar funciones.

Trigger (disparadores). Son objetos que se disparan cuando ocurre un evento específico. Los triggers se activan por cambios en las tablas a través de operaciones como Insert, Delete y Update. Existen dos tipos de triggers en SQL: Triggers DDL (Data Definition Language) los cuales se activan al modificar la estructura de la base de datos, como crear, modificar o eliminar tablas, o al realizar cambios en seguridad y eventos estáticos y los DML (Data Modification Language), siendo estos últimos los más comunes, se desencadenan por modificaciones de datos, como inserciones, actualizaciones o eliminaciones en tablas o vistas. En SQL Server, hay triggers especiales llamados Instead of ScholarHat (2024). que reemplazan las acciones estándar de insertar, actualizar o eliminar datos. En lugar de que la base de datos realice la acción directamente, el trigger se ejecuta de manera lógica, lo que permite un control más preciso y la posibilidad de manejar validaciones complejas o evitar cambios no deseados en tiempo real (Microsoft, s.f.).

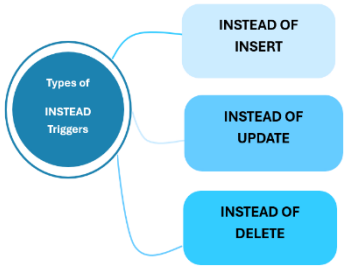

Figura 2. Estructura de un Instead. Nota: recurso gráfico para ilustrar la estructura de un instead.

4. Optimización con lenguaje procedimental

Un lenguaje procedimental brinda la posibilidad de construir stored procedures, funciones y triggers enfocados en la lógica del negocio en una base de datos; SQL es muy potente para realizar consultas puntuales, pero cuando se trata de automatizar tareas repetitivas o implementar lógica de negocio más compleja, los lenguajes procedimentales destacan. Estos lenguajes permiten crear funciones y procedimientos almacenados que encapsulan las operaciones, haciendo más eficiente el desarrollo y mantenimiento de las aplicaciones (Gabillaud, 2015).

Consideraciones de seguridad. Una base de datos debe tener un respaldo de seguridad, esta seguridad, se establece mediante herramientas, controles y medidas diseñadas para preservar la integridad, disponibilidad y confidencialidad de los datos integrados en la BD. Cuando se habla de seguridad en bases de datos se aborda y protege siguiendo la estructura de una jerarquía; esto permite crear un plan de mitigación de riesgos más orientada a la lógica del negocio:

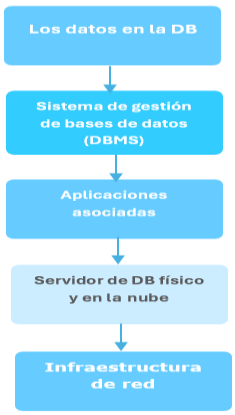

Figura 3. Estructura jerárquica de seguridad en una db. Nota: representa la jerarquía de la implementación de un servicio de seguridad aplicado a la base de datos.

La seguridad de una base de datos se relaciona con su accesibilidad: cuanto más accesible, más vulnerable. Las brechas de seguridad, a menudo por fallos de mantenimiento, pueden tener consecuencias graves, incluyendo sanciones por incumplimiento de normativas como SOX, PCI DSS y GDPR (ScholarHat, 2024).

5. Discusión

Si bien SQL es ideal para consultar y manipular datos de forma directa, los lenguajes procedimentales como Transact-SQL o PL/SQL añaden una capa de programación que permite automatizar tareas y crear lógica de negocio más compleja dentro de la base de datos. Esto se logra mediante procedimientos almacenados, funciones y triggers, que encapsulan una serie de instrucciones SQL. Al integrar SQL de manera nativa, los lenguajes procedimentales permiten combinar la potencia de las consultas con la flexibilidad de la programación, sin la necesidad de salir del entorno de la base de datos.

Los sistemas gestores de bases de datos (DBMS) ofrecen una amplia gama de funciones predefinidas, como MAX y MIN, para realizar cálculos básicos. Sin embargo, las funciones definidas por el usuario (UDF) permiten agregar funcionalidades que se adaptan a las necesidades específicas de nuestra aplicación. Al incorporar lógica de negocio dentro de estas funciones, se automatizan procesos complejos, mejorando la eficiencia y garantizando la integridad de los datos.

Las funciones personalizadas mejoran significativamente el rendimiento al ejecutarse dentro de la base de datos, facilitando el manejo de tipos de datos específicos. También reducen el tráfico de red al filtrar datos complejos mediante la cláusula WHERE, disminuyendo la cantidad de filas enviadas al cliente. A pesar de estos beneficios, es importante considerar que pueden afectar el rendimiento de las consultas en grandes conjuntos de datos, las funciones personalizadas optimizan varias tareas, pero deben ser gestionados cuidadosamente.

6. Conclusiones

El uso de un lenguaje procedimental es clave para realizar operaciones avanzadas en bases de datos y controlar la seguridad. Las funciones definidas por el usuario (UDF) mejoran la modularidad y reutilización del código, aunque su uso excesivo puede afectar el rendimiento y la seguridad. Los triggers automatizan acciones en respuesta a eventos específicos. La seguridad y el mantenimiento son esenciales, por lo que se deben implementar mecanismos de autenticación, control de acceso y copias de seguridad. Los sistemas de soporte de decisiones permiten un análisis profundo de datos. Además, es importante evaluar cuándo usar bases de datos NoSQL para superar las limitaciones de las bases de datos tradicionales.

Referencias

Camuña Rodríguez, J. F. (2015). *Lenguajes de definición y modificación dedatos SQL (UF1472).* IC Editorial.

Gabillaud, J. (2015). *SQL Server 2014: SQL, Transact SQL, diseño y creación de una base de datos (con ejercicios prácticos corregidos)*. Ediciones ENI.

IBM. (s.f.). *Functions user-defined*. https://www.ibm.com/docs/es/db2/11.5?topic=functions-user-defined

Microsoft. (s.f.). *User-defined functions*. https://learn.microsoft.com/en-us/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver16

ScholarHat (2024a). Estructura jerárquica de seguridad en una DB. *ScholarHat.* https://www.scholarhat.com/tutorial/sqlserver/different-types-of-sql-server-triggers

ScholarHat (2024b). Estructura de un Instead. *ScholarHat.* https://www.scholarhat.com/tutorial/sqlserver/different-types-of-sql-server-triggers

Zygiaris, S. (2018). *Database Management Systems: A Business-Oriented Approach Using ORACLE, MySQL and MS Access.* Emerald Publishing.