Vianka Vanessa Castro Ordoñez #23201

Ricardo Arturo Godinez Sanchez #23247

Parte 4 – analisis

1. ¿Qué ventajas encontraron al encapsular lógica en funciones en lugar de repetir consultas SQL?

Poder encapsular lógica en funciones es muy útil para la reutilización del código, ya que es mas sencillo llamar a la función que se necesita y pasarle los parámetros a estar llamando creado querys para modificar distintos casos.

Esto también puede reducir la posibilidad de errores ya que nos evita reescribir el código manualmente para cada solicitud.

2. ¿Qué criterios usaron para decidir cuándo implementar una función y cuándo una vista?

Una función la aplicamos cuando era necesario retornar datos mas específicos o realizar verificaciones, como la función de retornar el total de los pagos hechos por el usuarios o la de verificar el estado de la membresia del usuario. Las vistas se hicieron para cuando se necesitaba varios conjuntos de datos, hacer varios joins para mostrar información completa o solicitudes que necesitan de una vista en forma de tabla, ya que así se pueden encapsular de mejor manera esos querys. Las vistas fueron útiles para separar la lógica de consulta del cliente y las funciones eran más cuando se requería un parámetro, condición o devolver un valor calculado.

3. ¿Qué limitaciones encontraron al trabajar con procedimientos almacenados en comparación con funciones?

En general creemos que las limitaciones si así lo podemos llamar es que los procedimientos almacenados son generalmente para hacer cosas mas complejas en la base de datos, esto se ve reflejado en que los procedimientos almacenados pueden usar transacciones, esto nos indica que están pensados para realizar acciones mas complejas y de alto riesgo que no se lleguen a consolidar. Esto lo que conlleva es que no puedan ser utilizados desde una consulta, limitando así su uso en vistas o subconsultas. Por otro lado, las funciones pueden utilizarse en la mayoría de casos como en consultas complejas.

4. ¿Creen que el trigger que implementaron garantiza la integridad de los datos en todos los escenarios posibles? Justifiquen su respuesta.

Si, los triggers que implementamos mantienen la integridad de los datos en todo momento y en todos los escenarios posibles. En nuestro Trigger de Before que nos sirve para verificar que existan cupos en una clase antes de generar una reserva mantiene la integridad ya que nos asegura que no se revase el limite de cupos en la clase y no se generen conflictos. En nuestro trigger de after nos ayuda a mantener el control de las maquinas, ya que si una maquina es muy solicitada se pondrá en estado de mantenimiento debido al uso excesivo que pudo haber tenido, manteniendo la integridad y las reglas del negocio.

5. ¿Cómo adaptarían su solución para que escale en una base de datos con millones de registros?

Para poder escalar la base de datos creemos que se podrían definir las funciones, triggers, vistas y procedimientos almacenados necesarios. Esto hablando con el cliente, nos ayudaría a eficientar de gran manera todas las acciones que los usuarios o administradores puedan tener, ya que al tener estos procedimientos en la base de datos hace que se agilise mucho la visualización de datos y el mantener las reglas de negocio.

6. ¿Qué escenarios podrían romper su lógica actual si no existiera el trigger?

Por ejemplo, se podría reservar una clase cuando ya no tiene cupos, esto podría llevar a que los usuarios se muestren inconformes y se rompan las reglas del negocio al llegar a la clase y ver que su cupo ha sido tomado. Por el lado del trigger que cambia el estado de la maquina en mantenimiento, si esto no se hace puede ser que una maquina se descomponga por uso excesivo y puede ocasionar que las personas puedan lesionarse, estas son situaciones que nos evitamos con el uso de triggers.

7. ¿Qué dificultades enfrentaron al definir funciones que devuelven conjuntos de resultados?

El principal desafío al definir funciones que devuelven conjuntos de resultados fue distinguir conceptualmente entre estas funciones y las vistas, ya que ambas retornan datos tabulares. En la función clases_reservadas(p_id_usuario INT), enfrentamos dificultades con la sintaxis específica de RETURNS TABLE, el mapeo preciso de tipos de datos y la integración de parámetros en las consultas. También tuvimos que evaluar el compromiso entre rendimiento y flexibilidad, pues las funciones parametrizadas ofrecen mayor adaptabilidad pero pueden ser menos eficientes para consultas frecuentes. Finalmente, optamos por implementar esta funcionalidad como función en lugar de vista debido a la necesidad de parametrización por

usuario, característica que una vista estática no podría proporcionar sin filtros adicionales en cada consulta.

8. ¿Consideran que su diseño sería compatible con una arquitectura de microservicios? ¿Por qué sí o por qué no?

Si, nuestro diseño si sería compatible porque tenemos separado la lógica de datos como las vistas funciones y triggers de un posible consumo de API o una interfaz. Cada módulo puede ser implementado como un Micro servicio independiente que puede in]interactuar con su propio esquema. Esto nos ayuda a cumplir con escalabilidad, mantenibilidad e independencia

9. ¿Cómo reutilizarían las vistas que definieron en reportes o en otros sistemas?

Las vistas las podríamos reutilizar en dashboards de mantenimiento, reportes o para importaciones a hojas de cálculo/CSV. Otra opción de uso seria el poder exponerlos como endpoints de Apls, para que otras aplicaciones como una aplicación móvil o sistemas de admins puedan consumir estos datos ya filtrados, sin necesidad de ir a la base de datos a realizar el query nuevamente para cada cliente intentando obtener los datos.

10. ¿Qué aprendieron sobre la separación entre la lógica de negocio y la lógica de persistencia al hacer este laboratorio?

Al separar correctamente la lógica para cada uso hace que el sistema sea más escalable y mantenible. La lógica de persistencia nos permite almacenar y relacionar los datos por medio de esquemas, restricciones y claves. Mientras que la lógica de negocio está más centrada en las funciones, procedimientos y triggers utilizados para controlar los datos. Cada uno tiene su propósito y trabajan de forma modular y simplifica más la colaboración entre equipos técnico y negocio.

DBDIAGRAM: dbdiagram.io - Database Relationship Diagrams Design Tool