**¿USAR PROCEDIMIENTOS ALMACENADOS O NO?**

Para presentar una idea general de lo que expresa el autor del artículo “Who Needs Stored Procedures, Anyways? (¿Quién necesita procedimientos almacenados, de todos modos?) puede decirse que el autor esta exponiendo las ventajas que tiene utilizar SQL en línea sobre los Procedimientos Almacenados. Expone la posibilidad de no usar esta última estructura mencionada dentro de la construcción y manejo de las bases de datos.

Expone varias desventajas de ello partiendo de que para nadie es un secreto que es muy grande la cantidad de lenguajes de programación que existen, así mismo estos leguajes evolucionan a través del tiempo y sufren modificaciones, en consecuencia, resulta en un dolor de cabeza realizar migraciones y lidiar con las diferentes versiones de estos.

Así mismo habla de que otra desventaja es que los Procedimientos Almacenados, al no poderse depurar dentro del mismo IDE en el que se escribe su interfaz de usuario, al presentarse una larga lista de pasos a seguir para poder conseguir depurarlo resulta poco productivo. A esta problemática se le suma el hecho de que si el Procedimiento quedo mal diseñado arrojará respuestas incoherentes o inclusive no devolverá ningún tipo de valor, lo cual no es conveniente si la información contenida dentro de la base de datos es especialmente utilizada para realizar informes estadísticos sobre los cuales se toman decisiones determinantes en la compañía.

El autor también señala que el hecho de que los Procedimientos Almacenados no puedan pasar parámetros deriva en una tarea bastante engorrosa, tratar de identificar la procedencia del error el pasar de manera incorrecta un parámetro por error de incompatibilidad de tipos de datos. Y además de todo esto que se ha mencionado, como los Procedimientos Almacenados ocultan la lógica de negocios no se puede confiar al 100% en la información que éstos devuelven. A diferencia de SQL en línea que muy a pesar de que su construcción es extensa se puede “ver en contexto, junto con la otra lógica de negocios.” Aunque francamente el rendimiento depende más del buen o mal diseño de la base de datos, los procedimientos almacenados son una “fachada” para ocultar los errores en el diseño de la DB.

Pero a pesar de todas estas desventajas seguimos usando los Procedimientos Almacenados porque así lo hemos hecho desde siempre. Porque a través de ellos sentimos que le damos más estructura al diseño de la base de datos, y por la experiencia adquirida hemos comprobado que a través de ellos se obtiene un rendimiento mejorado a través de la reutilización de códigos y además facilita el mantenimiento ya que como menciona el autor, es más sencillo modificar un procedimiento almacenado que cambiar una declaración SQL dentro de un componente implementado. Porque se pueden proteger de manera individual dentro de la base de datos a través de permisos otorgados a cada cliente para que pueda acceder a los procedimientos mas no directamente a las tablas.

Finalmente el autor expresa “Para las bases de datos modernas y los escenarios de uso del mundo real, creo que una arquitectura de Procedimiento almacenado tiene serias desventajas y pocos beneficios prácticos”.

Hacer uso de los Procedimientos Almacenados puede llegar a ser un arma de doble filo, por eso lo recomendable es tener una buena educación en base a la lógica y la estructura de tal manera que al implementarlos se puedan conseguir los resultados esperados.

Es un arduo trabajo que requiere una precisión casi milimétrica. A fin de hacer menos traumática la experiencia deberíamos procurar encontrar el equilibrio entre lo excesivo y escaso; encontrar un punto que se pueda considerar mesurado al aplicar los códigos con procedimientos Almacenados. Debemos procurar invertir un poco mas entrenando nuestro cerebro para conseguir implementar códigos que estén bien estructurados, bien diseñados. Códigos tan eficientes que disminuyan considerablemente el margen de error ante la cantidad de escenarios que hemos ilustrado

Como se mencionó anteriormente los Procedimientos Almacenados son una herramienta habitual, porque se puede mejorar el rendimiento de la base de datos, ayuda a la protección de los datos a través de permisos, facilitan las labores de mantenimiento y ayuda a reducir el tráfico de la red.

Es importante utilizar Procedimientos Almacenados en las bases de datos, ya que estos nos proporcionan una amplia gama de herramientas en temas como lo es por ejemplo la seguridad de los datos, pues no es lo mismo tener una pequeña base de datos con decenas de registros a tener una gran base de datos con cientos de registros en los cuales hay que cuidar hasta el último detalle, ya que un descuido puede derivar en ataques o perdida de información, y esto es muy delicado si hablamos por ejemplo de grandes empresas como bancos. Dado este panorama, estamos pensando en grande, no es un código sencillo, si se deben utilizar consultas complejas no se quiere tener perdida de datos.

**CUANDO ROMPER EL PARADIGMA DE LA NORMALIZACION**

Es una constante preocupación buscar y analizar las diferentes alternativas que nos permitan mejorar los niveles de rendimiento y seguridad en las bases de datos a medida que aumenta la cantidad de información contenida en ella. Para este artículo, el autor cuestiona la eficacia de utilizar la normalización como un complemento de las buenas practicas al diseñar bases de datos.

Si bien, el objetivo de la normalización es organizar los datos, minimizar la cantidad de datos duplicados y perfeccionar la organización de la base de datos; no obstante, todo ese trabajo que implica normalizar la base de datos trae como consecuencia que las tablas se van a subdividir mas de la cuenta… y todo para recuperar la información de un solo usuario. Construir una consulta como un inner join a partir de tablas normalizadas se convertirá fácilmente en una compleja consulta, un verdadero dolor de cabeza sin que exista una real necesidad. Partiendo de allí podemos ultimar que, debido a su naturaleza, esto no mejorará el rendimiento del sistema.

La solución parece casi obvia que para el caso sería desnormalizar, lo cual convertiría automáticamente las consultas muchísimo mas sencillas. Pero hay un precio a pagar, esto no garantiza la integridad de los datos y por lo tanto, habrá un montón de huecos vacíos dentro de las filas que contienen la información. Una vez más, la normalización tiene tanto ventajas como desventajas, pero para determinar la mejor opción hay que tener en cuenta muchas cosas; si el escenario es una pequeña base de datos con unos cuantos registros, todo se torna muy diferente si el caso es de una gran base de datos con cientos o miles de registros. La clave es no ser pretenciosos y asumir con humildad los alcances de nuestro proyecto.

Como lo menciono Ted Dziuba (co-fundador de la compañía de compras en línea Milo, adquirida por eBay) al diseñar la base de datos es mejor optar por un esquema el cual uno considere mas fácil de entender y con el cual se trabaje usualmente, y normalizar parcialmente lo que se considere necesario, mas no todo.

Normalizar toda la base de datos en absoluto es una practica que en cierto punto pierde su sentido, pero que sigue siendo muy usada al ser considerada una técnica habitual ligada a “las buenas practicas de programación” como un paso necesario para garantizar un diseño exitoso. Es por ello por lo que se debería confiar sin orgullo un poco en la intuición y aplicar la normalización cuando se considere que tenga sentido hacerlo, sin predisposiciones ni sesgos y no solo porque es “el deber ser”.

Adicional a esto también debemos tener en cuenta que la normalización de bases de datos va mucho más allá de extender el código, detrás de ella hay organización en la cual esta fundamentada, la razón por la cual se considera que ayuda a darle mas estructura a la base de datos. Análisis de dependencias, relaciones, claves y hasta 5 formas normales, hacen parte de la edificación de la normalización así como varias reglas que hay que seguir para garantizar que sea correctamente aplicada. Emplear la técnica adecuadamente con todas sus normas es un gran desafío, que al mirarlo de cierto modo nos hace sentir la confianza de que nada podría salir mal.

Teniendo en cuenta todo lo anteriormente mencionado la desnormalización puede funcionar realmente bien cuando se tienen grandes volúmenes de información. Realizar un buen diseño de la base de datos normalizada e ir desnormalizando a medida que se vaya viendo la necesidad de hacerlo, después de todo desnormalizar es una manera de optimizar el rendimiento de la base de datos, nos falta romper los paradigmas ya que no siempre lo que siempre se hace es lo correcto o lo más conveniente, deberíamos explorar otras alternativas que nos permitan encontrar otras soluciones, quizás más eficientes o más dinámicas.

**LA REVOLUCION DE LA BASE DE DATOS MULTI-MODELO**

En el mundo de la programación la constante y variante es la creciente lista de lenguajes y técnicas de programación, lo que supone un problema de adaptación entre aplicaciones y bases de datos. [Nikhil Palekar](http://blog.foundationdb.com/author/nikhil-palekar), el autor del artículo presenta la idea de un motor de bases de datos multi-modelo que ha nacido recién y que puede solucionar las necesidades de muchas aplicaciones evitando a las empresas tener que preocuparse por “mantener sistemas, licencias de software, desarrolladores y administradores separados.” Estas bases de datos ofrecen un backend el cual contiene múltiples modelos compatibles con aplicaciones diferentes. Pero por qué es tan importante un Backend? Porque es el responsable de hacer que todo lo que está detrás de una aplicación funcione correctamente para posteriormente enviarle la información al usuario.

Palekar presenta lo que él considera los 10 beneficios más importantes que ofrecen los multi-modelos:

1. Consolidación: una base de datos multi-modelo admite diferentes tipos de datos para diferentes casos de uso y los consolida en una sola plataforma. Lo cual es muy útil ya que tiene un motor de almacenamiento común y es flexible en el lenguaje de consulta y el modelo de datos.
2. Escala de rendimiento: Varia de acuerdo al uso que se le dé a la aplicación y la mejor opción no siempre es usar una maquina con mayor capacidad. Los sistemas multi-modelos permiten diferentes componentes para ser escalados de forma independiente a medida que cambian las necesidades.
3. Complejidad Operacional: busca dominar múltiples lenguajes para permitirle utilizar el mas adecuado, sin embargo en la practica puede significar terminar con varias bases de datos.
4. Flexibilidad: involucra la asignación de varios modelos de datos en un único motor de almacenamiento para admitir diferentes aplicaciones.
5. Fiabilidad: Cuando se ejecutan múltiples bases de datos la seguridad cada base de datos puede ser un punto único de falla.
6. Consistencia de datos: Un sistema backend soporta multiples modelos de datos tiene la capacidad de ayudar a que los datos se adapten entre el sistema y la base de datos.
7. Tolerancia a fallos: para poder garantizarla el equipo debe tener experiencia con cada componente para que el sistema en general siga funcionando bien.
8. Costo: cada componente del sistema multimodelo requiere mantenimiento, actualizaciones, parches, correcciones entre otros y puede ir aumentando el costo con el paso del tiempo.
9. Transacciones: Un sistema multi-modelo requiere transacciones para garantizar que los datos se almacenan constantemente y todas sus aplicaciones lo heredan.
10. Mejores aplicaciones: una aplicación compatible con una base de datos multi-modelo obtiene los beneficios de escalabilidad, tolerancia a fallos y alto rendimiento.

Como muchas otras cosas en el mundo del desarrollo, los multi-modelos también tienen su estructura y normas a través de los cuales se busca alcanzar el objetivo, que para este caso es desarrollar un sistema en un entorno en el cual tiene una gama más amplia donde adaptarse, de tal forma que el usuario podrá cumplir con diferentes requisitos de la aplicación sin necesidad de implementar diferentes sistemas de bases de datos.

Mientras que las bases de datos relaciones están basadas en tablas para finalmente conectar con modelos orientados a objetos y que en ciertos puntos del camino han probado poca eficiencia, las bases de datos multi-modelo podría decirse que maneja una sola tabla (todo lo contrario a SQL), no almacena la información de forma uniforme como lo hacen las bases de datos relacionales, se basa en dimensiones o niveles a través de las cuales se puede navegar entre ellos y analizar los datos de una manera más profundizada.

Naturalmente, para implementar un desarrollo multi-modelo se requiere estudiar y empaparse bien en el tema, saber lo que se esta haciendo, ya que si está bien implementado puede traer multiples beneficios, al momento de realizar complejas consultas puede ser más eficiente que un modelo de base de datos relacional. No obstante cabe mencionar que también tiene ciertas desventajas y solo por mencionar una es que no puede ser actualizado en uno solo de sus componentes, se tiene que actualizar todas las áreas del sistema.

No en vano este tipo de sistemas multi-modelos no es muy común, apenas está surgiendo como una alternativa que puede brindar soluciones de manera diferente ante problemas comunes cuando se usan las bases de datos relacionales. Considero que al ser un ámbito poco explorado y tan poco popular no se debería utilizar mas que para experimentar y aprender “por cultura general” en el mundo de los desarrolladores como es que funciona esta técnica, ya que llevarla al área comercial puede llegar a ser traumático teniendo en cuenta que el mercado esta muy sesgado hacia ciertos productos comercializados (sistemas operativos) y entornos de desarrollo.

Este profundo mar ya es bastante diverso, el aprendizaje debe ser dinámico como lo es la creación de las nuevas tecnologías, pero no debería utilizarse (para experimentar) una práctica tan reciente que compromete no solo todo el software de una empresa sino que inclusive compromete el hardware para manipular el activo mas valioso dentro de una compañía que es su información, dado que esta es la pieza clave para la toma de decisiones e implementación de nuevas estrategias. Probablemente sea una tecnología que en el futuro tendrá mucho éxito pero en la actualidad aun le falta mucho camino por recorrer y mejoras a implementar.