**Ensayo de la normalización no se tal vez normal**

la escalabilidad no es un problema, hacer que la gente se preocupe por eso si . Por lo tanto, cuando se trata del diseño de bases de datos, mida el rendimiento, pero intente errar fuertemente en el lado del diseño sano y simple . Elija el esquema de base de datos que crea que es más fácil de entender y con el que trabaje a diario. No tiene que ser todo o nada como lo he visto arriba; puede desnormalizar parcialmente donde tiene sentido hacerlo, y mantenerse completamente normalizado en otras áreas donde no lo tiene.

Dare Obasanjo tuvo una excelente publicación Cuándo no normalizar su base de datos SQL en la que proporciona un esquema de base de datos de muestra para un sitio de redes sociales genérico.

La normalización sin duda cumple en términos de limitar la duplicación. Cada entidad está representada una vez, y solo una vez, por lo que casi no hay riesgo de inconsistencias en los datos. Pero ese diseño también requería de seis combinaciones para recuperar la información de un solo usuario.

Esas seis uniones tampoco están haciendo nada para ayudar al rendimiento de el sistema. La normalización en toda regla no es simplemente difícil de entender y difícil de trabajar, también puede ser bastante lenta.Como señala Dare, la solución obvia es desnormalizar : colapsar una gran cantidad de datos en una sola tabla de Usuarios.

Esto funciona: las consultas ahora son increíblemente simples probablemente también muy rápidas. Pero tendrá un montón de huecos vacíos en sus datos, junto con una gran cantidad de matrices de campos con nombres incómodos.Y todos esos problemas molestos de integridad de datos que la base de datos usó para usted

Ambas soluciones tienen sus pros y sus contras. Entonces: ¿ cuál es mejor: una base de datos normalizada o una base de datos desnormalizada

La respuesta es que no importa! Hasta que tengas millones y millones de filas de datos, eso es. Todo es rápido para los pequeños . Incluso una PC modesta para los estándares actuales, digamos una caja de doble núcleo con 4 gigabytes de memoria, le dará un rendimiento casi idéntico en cualquier caso para cualquier cosa que no sea la base de datos más grande.

**Ensayo de procedimientos almacenados**

Es gratificante hablar que los procedimientos almacenados dan como resultado un mejor rendimiento y seguridad pero son mitos ya que se pueden presentar diversos escenarios donde se puede ver afectado el rendimiento la seguridad y la escalabilidad de nuestras bases de datos incluso se pueden presentar problemas entre versiones diferentes.

Para las bases de datos modernas y los escenarios de uso del mundo real,una arquitectura de Procedimiento almacenado tiene serias desventajas y pocos beneficios prácticos.

Los procedimientos almacenados se deben considerar como lenguaje de ensamblaje de la base de datos: para usar solo en las situaciones más críticas.

Ensayo de bases de datos múltiples

Estas bases de datos de múltiples modelos pueden proporcionar un único back-end que expone múltiples modelos de datos a las aplicaciones que admite. De esa manera, las bases de datos de modelos múltiples eliminan la fragmentación y proporcionan un backend consistente y bien comprendido que admite muchos productos y aplicaciones diferentes. Los beneficios para la organización son extensos, pero algunos de los beneficios más importantes incluyen:

**Consolidación**

Una base de datos de múltiples modelos admite diferentes tipos de datos para diferentes casos de uso y los consolida en una plataforma. Por lo tanto, obtiene flexibilidad en el lenguaje de consulta y el modelo de datos, pero simultáneamente se beneficia de una tecnología de motor de almacenamiento común.

**Escala De Rendimiento**

Los sistemas multimodelo que desacoplan el lenguaje de consulta y el modelo de datos del almacén de datos subyacente permiten escalar diferentes componentes dentro de la arquitectura de manera independiente a medida que cambian las necesidades.

**Complejidad Operacional**

Los entornos fragmentados causados ​​por la ejecución de diferentes bases de datos aumentan la complejidad tanto de las operaciones como del desarrollo

**Flexibilidad**

Un enfoque de modelos múltiples implica la asignación de múltiples modelos de datos en un único motor de almacenamiento subyacente que puede admitir diferentes casos de uso y aplicaciones

**Fiabilidad**

La confiabilidad de la base de datos también es un problema cuando se ejecutan múltiples bases de datos, ya que cada sistema de base de datos podría ser un punto único de falla para el sistema y la aplicación más grandes.

**Consistencia De Los Datos.**

una funcionalidad de transacción de nivel superior integrada en su aplicación, no hay soporte para transacciones en diferentes sistemas de bases de datos. En consecuencia, no hay una buena manera de mantener la coherencia entre los diferentes modelos.

**Tolerancia A Fallos**

la integración de múltiples sistemas que fueron diseñados para ejecutarse de manera independiente para que brinden tolerancia a fallas en todo el sistema impone costos de ingeniería y operativos significativos.

**Costo**

El uso de más sistemas de bases de datos distintos aumenta los costos según el hardware, el software y las necesidades operativas asociadas con cada sistema.

**Transacciones**

esun desafío proporcionar transacciones a través de múltiples máquinas, y casi todas las bases de datos NoSQL no proporcionan garantías transaccionales debido a sus diseños arquitectónicos. Debido a que un verdadero sistema de múltiples modelos requiere transacciones para garantizar que los datos se almacenen de manera consistente en la base de datos, todas sus aplicaciones heredan este contrato sólido de cómo se almacenan los datos**.**

**Mejores Aplicaciones**

En contraste, una aplicación que es compatible con una base de datos de múltiples modelos obtiene los beneficios de escalabilidad, tolerancia a fallas y, en un sistema bien diseñado, alto rendimiento integrado en el producto. Con menos lógica adicional necesaria en el nivel de la aplicación para manejar las interacciones de la base de datos y las posibles condiciones de falla, los desarrolladores pueden concentrarse en crear mejores aplicaciones. Debido a estos beneficios, los sistemas multimodelo son a los que se dirige el mercado de bases de datos: transacciones compatibles con ACID, API multimodelo y potentes motores de almacenamiento compartidos