# Slide 3

Muito se tem falado da utilização de NoSQL. Existem várias comparações de desempenho realizadas entre ele e o SQL, tendo como a velocidade o fator mais importante na decisão de qual banco utilizar. Há muitas opções de bancos de dados NoSQL com diferentes funcionalidades que podem ser muito úteis.

# Slide 4

O modelo relacional forneceu uma maneira padrão de representar e consultar dados que poderiam ser usados por qualquer aplicativo. Desde o início, os desenvolvedores reconheceram que a principal força do modelo de banco de dados relacional estava no uso de tabelas, que era uma maneira intuitiva, eficiente e flexível de armazenar e acessar informações estruturadas.

# Slide 5

## Consistência de Dados

O modelo relacional é o melhor em manter a consistência de dados entre aplicativos e cópias de banco de dados (chamadas de instâncias). Por exemplo, quando um cliente deposita dinheiro em um caixa eletrônico e analisa o saldo da conta em um celular, o cliente espera ver esse depósito refletido imediatamente em um saldo atualizado da conta. Os bancos de dados relacionais se destacam nesse tipo de consistência de dados, garantindo que várias instâncias de um banco de dados tenham os mesmos dados o tempo todo.

É difícil para outros tipos de bancos de dados manter esse nível de consistência oportunamente com grandes quantidades de dados. Alguns bancos de dados recentes, como o NoSQL, podem fornecer somente “consistência eventual.” Sob este princípio, quando o banco de dados é dimensionado ou quando vários usuários acessam os mesmos dados ao mesmo tempo, os dados precisam de algum tempo para “serem atualizados.” A consistência eventual é aceitável para alguns usos, como manter listagens em um catálogo de produtos, mas para operações comerciais essenciais, como transações de carrinho de compras, o banco de dados relacional ainda é o padrão ouro.

## Compromisso e Atomicidade

Os bancos de dados relacionais lidam com regras e políticas de negócios em um nível muito granular, com políticas rígidas sobre compromisso (isto é, fazer uma alteração no banco de dados permanente). Por exemplo, considere um banco de dados de inventário que rastreie três peças que são sempre usadas juntas. Quando uma peça é retirada do inventário, as outras duas também devem ser retiradas. Se uma das três peças não estiver disponível, nenhuma das peças deve ser retirada. Todas as três peças devem estar disponíveis antes que o banco de dados confirme. Um banco de dados relacional não assumirá o compromisso por uma peça até que saiba que pode confirmar todas as três. Essa capacidade de compromisso multifacetada é chamada atomicidade. A atomicidade é a chave para manter os dados precisos no banco de dados e garantir que eles estejam em conformidade com regras, regulamentos e políticas da empresa.

## Procedimentos Armazenados e Bancos de Dados Relacionais

O acesso a dados envolve muitas ações repetitivas. Por exemplo, uma consulta simples para obter informações de uma tabela de dados pode precisar ser repetida centenas ou milhares de vezes para produzir o resultado desejado. Essas funções de acesso a dados requerem algum tipo de código para acessar o banco de dados. Os desenvolvedores de aplicativos não querem criar um novo código para essas funções a cada aplicativo novo. Felizmente, os bancos de dados relacionais permitem procedimentos armazenados, que são blocos de código que podem ser acessados com uma simples chamada de aplicativo. Por exemplo, um único procedimento armazenado pode fornecer identificação de registro consistente para usuários de vários aplicativos. Os procedimentos armazenados também podem ajudar os desenvolvedores a garantir que determinadas funções de dados no aplicativo sejam implementadas de uma maneira específica.

## Bloqueio e Simultaneidade de Banco de Dados

Os conflitos podem surgir em um banco de dados quando vários usuários ou aplicativos tentam alterar os mesmos dados ao mesmo tempo. Técnicas de bloqueio e simultaneidade reduzem o potencial de conflitos, mantendo a integridade dos dados.

O bloqueio impede que outros usuários e aplicativos acessem dados enquanto estão sendo atualizados. Em alguns bancos de dados, o bloqueio se aplica à tabela inteira, o que cria um impacto negativo no desempenho do aplicativo. Outros bancos de dados, como bancos de dados relacionais da Oracle, aplicam bloqueios no nível de registro, deixando os outros registros dentro da tabela disponíveis, ajudando a garantir um melhor desempenho do aplicativo.

A simultaneidade gerencia a atividade quando vários usuários ou aplicativos fazem consultas ao mesmo tempo no mesmo banco de dados. Esse recurso fornece o acesso correto a usuários e aplicativos de acordo com as políticas definidas para o controle de dados.

# Slide 6

No slide

# Slide 7

No slide

# Slide 8

No slide

# Slide 9

Documento – Os dados são armazenados como documentos. Os documentos podem ser descritos como dados no formato de chave-valor, como por exemplo, o padrão JSON. Um exemplo de banco de dados neste formato é o MongoDB;

Colunas – Os dados são armazenados em linhas particulares de tabela no disco, podendo suportar várias linhas e colunas. Também permitem sub-colunas. Um banco de dados dessa família, por exemplo, é o Cassandra;

Grafos – Os dados são armazenados na forma de grafos (vértices e arestas). O Neo4j é um banco que utiliza grafos;

Chave-valor – Esta família de bancos NoSQL é a que aguenta mais carga de dados, pois o conceito dele é que um determinado valor seja acessado através de uma chave identificadora única. Um exemplo é o banco de dados Riak.

# Slide 10

Uma das diferenças nos bancos de dados NoSQL é que toda a informação é agrupada e guardada no mesmo registro. Já no SQL você precisa ter o relacionamento entre várias tabelas para ter a informação, informação esta disposta no modelo entidade e relacionamento.

O SQL tem certa dificuldade em conciliar a demanda por escalabilidade. Quanto a escalabilidade do NoSQL, deve se levar em consideração a modelagem do sistema. Do que adianta termos um sistema super simples e querermos utilizar o NoSQL apenas pela escalabilidade? Todo o ganho de performance seria perdido quando rodássemos a aplicação.

Um ponto forte do SQL é quanto à consistência das informações. Já o NoSQL garante o último valor atualizado, isso se nenhuma atualização for realizada até o momento da consulta.