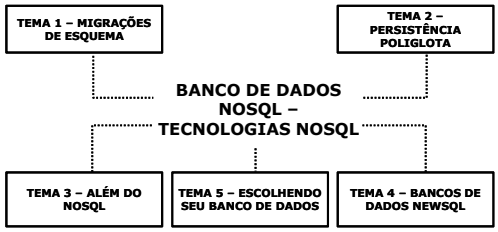
Aula 6

Banco de Dados NoSQL

Prof. Alex Mateus Porn

- Tecnologias NoSQL

Conversa Inicial



Migrações de Esquema

Migrações de esquema

- Características do NoSQL (Sadalage; Fowler, 2013):
  - Natureza livre de esquemas
  - Permite aos desenvolvedores concentrarem-se no projeto do domínio sem preocupação com alterações no esquema
  - Possibilitam a mesma proposta dos métodos ágeis, em que é importante atender as mudanças nos requisitos

Migrações de esquema

- Características do NoSQL:
  - Ao contrário dos bancos de dados relacionais, a abordagem sem esquema visa a flexibilidade nas alterações
  - Tem como objetivo atender as frequentes alterações no mercado e as inovações de produtos de *software*

Migrações de esquema

- Conforme Sadalage e Fowler (2013), mesmo diante da análise de que em alguns casos o esquema não precisa ser planejado antecipadamente em bancos de dados NoSQL e diante da flexibilidade da natureza livre de esquemas, ainda são necessários o planejamento e o projeto de alguns aspectos, tais como:

Migrações de esquema

- Os tipos de relacionamentos para bancos de dados orientados a grafos
- Os nomes das famílias de colunas, linhas e colunas ou a ordem das colunas para os bancos de dados orientados a colunas
- Como as chaves estão atribuídas e qual é a estrutura dos dados dentro do objeto de valores para os bancos de dados orientados a chave-valor

Migrações de esquema

- Conforme Sadalage e Fowler (2013):
  - Os bancos de dados NoSQL não são inteiramente desprovidos de esquema
  - O esquema tem de ser definido pelo aplicativo, pois o fluxo de dados tem de ser analisado por ele ao fazer a leitura dos dados
  - O aplicativo tem de criar os dados que seriam gravados no banco de dados
  - Se não é possível analisar os dados, há uma incompatibilidade de esquema

Migrações de esquema – exemplificando (Sadalage; Fowler, 2013)

- Banco de dados orientado a grafos
  - Ao alterar o tipo de uma aresta no aplicativo, não é mais possível percorrer o banco de dados, tornando-o inutilizável
  - Uma solução é percorrer todas as arestas do banco e alterá-las conforme necessário
  - Dependendo do tamanho do banco, essa operação é muito custosa e exige a criação de códigos para migrar todas as arestas necessárias

Migrações de esquema relacional para NoSQL (Zhao et al., 2014)

- Metodologia baseada em grafos:
  - Conversão do modelo relacional para qualquer modelo NoSQL
  - Resumidamente constrói-se um grafo em que o conjunto de vértices representa as tabelas e as arestas representam os relacionamentos do modelo relacional

Migrações de esquema relacional para NoSQL (Li; Ma; Chen, 2014)

- Metodologia baseada em consultas:
  - Consideram-se quais consultas serão realizadas no banco de dados a fim de aumentar o desempenho da busca, uma vez que operações de junção não são aconselháveis em bancos de dados NoSQL

Migrações de esquema relacional para NoSQL (Karnitis; Arnicans, 2015)

- Metodologia baseada nos níveis físico e lógico
  - Nível físico dos dados
    - As tabelas são identificadas e caracterizadas
  - Primeiro nível lógico dos dados
    - Os metadados são acrescidos de informações lógicas, em linguagem natural
  - Segundo nível lógico dos dados
    - As tabelas são utilizadas para gerar o esquema dos documentos e a semântica do negócio

Considerações para a migração de esquemas (Sadalage; Fowler, 2013)

- Esquemas fortes podem ser migrados gravando cada alteração do esquema, mais a migração de dados, em uma sequência controlada por versões
- Bancos de dados sem esquema precisam de uma migração cuidadosa devido ao esquema implícito nos códigos que acessam os dados

Considerações para a migração de esquemas (Sadalage; Fowler, 2013)

- Bancos de dados sem esquema podem utilizar as mesmas técnicas de migração dos bancos de dados com esquemas fortes
- Bancos de dados sem esquema também podem ler dados de forma tolerante às alterações no esquema implícito de dados e, além disso, podem utilizar migração incremental para atualizá-los

Persistência poliglota

Persistência poliglota

- De acordo com Sadalage e Fowler (2013):
  - Utilizar um único mecanismo de banco de dados para todas as necessidades resulta em soluções de baixo desempenho
  - Armazenar dados transacionais, guardar em cache as informações de sessão, percorrer grafos de clientes e produtos que seus amigos compraram são problemas essencialmente diferentes

Persistência poliglota

- Ainda conforme Sadalage e Fowler (2013):
  - Muitas instituições tendem a utilizar o mesmo mecanismo de banco de dados para armazenar:
    - Transações de negócio
    - Dados de gerenciamento de sessão
    - Relatórios, BI, *data warehouse* ou informações de registros

Persistência poliglota (Sadalage; Fowler, 2013)

- Em 2006, Neal Ford propôs a expressão "programação poliglota" para expressar a ideia de que os aplicativos devem ser escritos em uma mistura de linguagens, de modo a aproveitar o fato de que diferentes linguagens são apropriadas para lidar com diferentes problemas

Persistência poliglota – exemplificando (Sadalage; Fowler, 2013)

- Sistema de comércio eletrônico:
  - Dados do carrinho de compras antes de o pedido ser confirmado e dados da sessão do usuário:
    - Banco de dados orientado a chave-valor
  - Confirmação de pagamento e compras realizadas:
    - Bancos de dados relacionais
  - Recomendação de produtos para clientes:
    - Banco de dados orientado a grafos

Persistência poliglota – exemplificando

- Analogia com uma caixa de ferramentas
  - Alicate, martelo, chave de fenda, serra, etc.
  - Não podemos afirmar qual é a melhor ferramenta na caixa, tudo dependerá do serviço a ser realizado
  - Se for necessário apertar um parafuso, pregar um prego ou serrar uma madeira, para cada atividade a ser implementada cabe analisarmos a melhor ferramenta a ser utilizada

Persistência poliglota

- Sadalage e Fowler (2013) destacam que:
  - À medida que múltiplos bancos de dados são implementados em um aplicativo, outros aplicativos na empresa poderão se beneficiar do uso desses bancos ou dos dados neles armazenados

Persistência poliglota

- Sadalage e Fowler (2013) reforçam que:
  - A persistência poliglota está relacionada ao uso de diferentes tecnologias de armazenamento de dados, de forma que possa lidar com necessidades variáveis de armazenamento de dados
  - A persistência poliglota pode ser aplicada em uma empresa ou dentro de um único aplicativo

Além do NoSQL

Além do NoSQL

- NoSQL é apenas um conjunto de tecnologias de armazenamento de dados. Como aumentam as facilidades com a persistência poliglota, devemos analisar outras tecnologias de armazenamento de dados, possuindo ou não o rótulo NoSQL (Sadalage; Fowler, 2013)

Bancos de Dados XML (Sadalage; Fowler, 2013)

- Similares aos bancos de dados orientados a documentos
- Os documentos são armazenados em um modelo de dados compatível com XML
- Alguns bancos de dados relacionais permitem inserir documentos XML como um tipo de coluna e possibilitam alguma forma de misturar as linguagens de consulta SQL e XML

Bancos de dados de objetos (Sadalage; Fowler, 2013)

- Começaram a surgir com a popularização da programação orientada a objetos
- Focam na complexidade do mapeamento de estruturas de dados na memória para tabelas relacionais
- Tem como objetivo evitar essa complexidade, de modo que o banco gerencie automaticamente o armazenamento das estruturas da memória para o disco

Elasticsearch (Paniz, 2016)

- Não é necessariamente um banco de dados, mas uma ferramenta para processamento de *queries* envolvendo textos
- Utiliza lógica difusa para fazer consultas aos dados. Cada palavra recebe uma nota baseada no termo buscado, assim retornando palavras semelhantes

openCypher (Paniz, 2016)

- Cypher é a linguagem padrão para manipulação e gerenciamento dos bancos de dados orientados a grafos
- Essa linguagem despertou a atenção de outras empresas que desenvolvem bancos orientados a grafos, que se uniram e criaram uma versão pública de código aberto denominada openCypher

Datomic (Paniz, 2016)

- Armazena todo o histórico de atualizações dos registros
- Ao alterar um registro, na verdade, é adicionando um novo fato a ele. Por isso, pode-se facilmente carregar apenas um conjunto de dados atual de um registro ou carregar todas as transações que modificaram o registro e verificar cada valor antes e depois de cada transação

Spark (Paniz, 2016)

- Não é um banco de dados, mas sim um motor para processamento de fluxo
- Permite montar e gerenciar um *cluster* de máquinas para executar processamentos em cima de grandes volumes de dados
- Possibilita a conexão de vários tipos de origens de dados, incluindo bancos de dados relacionais e NoSQL

PostgreSQL Document Store (Paniz, 2016)

- Oferece suporte para armazenar documentos JSON, tanto no formato JSON puro quanto no formato binário (jsonb), semelhante ao formato BSON usado pelo MongoDB
- Permite a realização de consultas por atributos, elementos aninhados e em *arrays*, da mesma forma como em um banco orientado a documentos

Redis e Memcached (Paniz, 2016)

- São bancos NoSQL orientados a chave-valor
- O Redis usa uma forma de replicação de dados baseada em *master*/*slave* e suporta tipos de dados especiais, como conjuntos (*set*) e listas (*list*)
- O Memcached segue uma filosofia mais simplista: parte da lógica deve ficar no cliente, pois o servidor não suporta nenhum tipo de replicação ou dados especiais

Bancos de dados NewSQL

NewSQL

- Surgiram a partir da necessidade de consistência dos dados e de poder escalonar mais facilmente o sistema
- Referem-se a uma classe de sistemas de gerenciamento de bancos de dados relacionais que procuram oferecer o mesmo desempenho escalável do modelo NoSQL

NewSQL
(Ryan, 2018; Grolinger et al., 2013)

- Banco de dados totalmente relacional
- Conformidade com as propriedades ACID
- Latência de milissegundos
- Tolerância a falhas
- Execução local ou na nuvem

(...)

NewSQL
(Ryan, 2018; Grolinger et al., 2013)

- Processamento de milhões de transações por segundo
- Escalonamento vertical
- Podem usar diferentes formas de armazenamento

NewSQL (Grolinger et al., 2013)

- Adequados para as seguintes situações:
  - Cenários em que o tradicional banco de dados relacional é utilizado, mas que têm requisitos adicionais de escalabilidade e desempenho
  - Aplicativos no mercado financeiro, nos quais operações como transferências de dinheiro precisam atualizar duas contas automaticamente e todos os aplicativos precisam ter a mesma visão do banco de dados

Escolhendo seu banco de dados

Escolhendo seu banco de dados

- Conforme apresentam Sadalage e Fowler (2013), duas importantes situações podem ser consideradas no momento de escolher um banco de dados NoSQL:
  1. Produtividade do programador
  2. Desempenho no acesso aos dados

Produtividade do programador

- O primeiro passo na avaliação, conforme destacam Sadalage e Fowler (2013), é:
  - Examinar o que o *software* precisará fazer
  - Observar os recursos atuais
  - Verificar como o uso dos dados é mais apropriado
- Com essas três premissas, é possível que um modelo de dados apropriado comece a se formar

Produtividade do programador

- Sadalage e Fowler (2013) destacam que não há como medir apropriadamente quão produtivos são diferentes projetos, sendo uma solução para isso:
  - Escolher alguns recursos iniciais do projeto e desenvolvê-los, ao mesmo tempo em que se presta atenção se é realmente fácil utilizar a tecnologia em questão

(...)

Produtividade do programador

- Considerar criar os mesmos recursos com alguns bancos de dados diferentes, para ver qual se adequa melhor ao problema sendo solucionado

Desempenho no acesso aos dados

- Conforme Sadalage e Fowler (2013), o mais importante é testar o desempenho em cenários adequados às necessidades do desenvolvedor
- A melhor forma de avaliar o desempenho apropriadamente é criando uma solução para um problema específico, executando e medindo-o

Desempenho no acesso aos dados (Sadalage; Fowler, 2013)

- Não é possível testar todas as formas como o aplicativo será utilizado
- É necessário criar um conjunto representativo de testes, selecionando:
  - Cenários que sejam os mais comuns
  - Cenários mais dependentes de desempenho
  - Cenários que não pareçam se adaptar bem ao modelo de banco de dados proposto

Principais motivos para usar NoSQL

- Conforme Sadalage e Fowler(2013):
  - Melhorar a produtividade do programador, utilizando um banco de dados que se adapte melhor às necessidades de um aplicativo
  - Melhorar o desempenho no acesso aos dados por meio de alguma combinação na manipulação de volumes maiores de dados, reduzindo a latência e melhorando o rendimento

(...)

Principais motivos para usar NoSQL

- É essencial testar as expectativas sobre a produtividade do programador e/ou desempenho antes de decidir utilizar uma tecnologia NoSQL

Referências

- SADALAGE, P. J.; FOWLER, M. NoSQL Essencial: um guia conciso para o mundo emergente da persistência poliglota. São Paulo: Novatec, 2013.
- ZHAO, G. et al. Schema conversion model of SQL database to NoSQL. In: *Proceedings of 9th International Conference of P2P, Parallel, Grid, Cloud Internet Comput,* pg. 355–362. 2014.
- LI, X.; MA, Z.; CHEN, H. QODM: A query-oriented data modeling approach for NoSQL databases. Proceedings of IEEE Workshop on Advanced Research and Technology in Industry Applications, pg. 338–345, 2014.
- KARNITIS, G.; ARNICANS, G. Migration of Relational database to document-oriented database: structure denormalization and data transformation. In: Proceedings of 7th Computational Intelligence, Communication Systems and Networks, pg. 113–118, 2015.
- PANIZ, D. NoSQL: como armazenar os dados de uma aplicação moderna. São Paulo: Casa do Código, 2016.
- GROLINGER, K. et al. Data management in cloud environments: NoSQL and NewSQL data stores. Journal Of Cloud Computing: Advances, Systems And Applications, 2013. Disponível em: https://link.springer.com/content/pdf/10.1186%2F2192-113X-2-22.pdf. Acesso em: 8 dez. 2020.
- RYAN, J. Oracle vs. NoSQL vs. NewSQL Comparing Database Technology. VoltDB, 2018. Disponível em: https://www.voltdb.com/wp-content/uploads/2018/01/VoltDB_Oracle_vs_NoSQL_vs_NewSQL_eBook_7March2019.pdf . Acesso em: 8 dez. 2020.