

Banco de Dados NoSQL

Aula 1: Introdução a NoSQL

Apresentação

Nesta aula, iniciaremos o estudo dos bancos de dados NoSQL, que é uma das áreas mais recentes em banco de dados. Porém, será necessário antes compreendermos a evolução dos modelos de dados para identificarmos as razões da criação desses novos modelos. Após isso, compreenderemos as principais características dos modelos Not Only SQL. Efetuaremos a comparação com o banco de dados relacional e suas diferenças para os modelos NoSQL.

Objetivo

- Analisar a evolução dos modelos de bancos de dados;
- Reconhecer os tipos de dados: estruturados, semiestruturados e não estruturados;
- Reconhecer as diferenças do banco de dados relacional para os modelos NoSQL.

Introdução

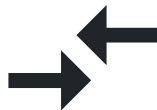
A área de banco de dados é uma das mais antigas na computação e sua evolução foi marcada pelos diferentes modelos de dados que foram sendo criados ao longo do tempo. Um modelo de dado é a forma como os dados são organizados no banco.

Primeiros modelos de dados

Os primeiros modelos de dados foram o modelo hierárquico e o modelo em rede.

Modelo hierárquico

Era estruturado de modo que era permitido o desenvolvimento de relacionamentos do tipo pai-filho entre as tabelas, apresentando a exigência que todo o projeto do banco de dados necessitava ser definido de forma hierárquica.



Modelo em rede

era organizado com o seu conjunto de registros interligados por meio de ponteiros. As dificuldades encontradas tanto em nível de projeto quanto em relação ao seu uso e suas limitações, provocaram a busca por outros modelos de dados.

Em 1970, surge então um modelo de dados que ainda é amplamente usado, representando aproximadamente 70% do mercado atual (https://db-engines.com/en/ranking_categories). Veja a seguir.

Modelo relacional

O modelo relacional surgiu com o objetivo principal de superar questões relacionadas com o suporte (manutenção facilitada em relação aos seus antecessores), a independência e a integridade dos dados nos Sistemas Gerenciadores de Banco de Dados (SGBD) (Elmasri, 2015). No período em que foi lançado, operações comuns envolvendo consultas apresentavam problemas por falta de integridade dos dados e, para solucionar isso era necessário o desenvolvimento de programas complexos.

Fatores de sucesso do modelo relacional

Houve uma boa aceitação dos usuários e da comunidade acadêmica ao modelo relacional devido a sua simplicidade e

1 uniformidade com um conjunto de tabelas utilizando atributos atômicos; seu formalismo matemático baseado na Álgebra e Cálculo Relacional, além da Teoria de Conjuntos e o uso de uma linguagem padronizada (SQL - Structured Query Language).

2 Além disso, o controle de transações disponíveis por meio das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) permitiu um maior controle de suas operações. Cabe destacar, também, a característica da primeira forma normal, a qual estabelece que os atributos armazenados em uma tabela devem ser simples, não podendo ser multivalorados ou compostos, sendo essa uma diferença marcante para os modelos NoSQL (Fowler, 2013).

3 Para completar a lista de principais fatores de sucesso do modelo relacional, o Modelo Entidade-Relacionamento criado em 1976 facilitou bastante o projeto desse modelo de dados, assim como o entendimento da estrutura do banco de dados relacional, exibindo o esquema (conjunto de tabelas relacionadas em uma aplicação) de forma gráfica e facilitando a construção de consultas envolvendo várias tabelas (*joins*).

Os SGBDs relacionais disputam espaço na indústria de banco de dados com outros SGBDs não-relacionais, tais como o SGBD Orientado a Objeto e os NOSQL (Not Only SQL). Diante do avanço da Orientação a Objetos (OO), os grandes fabricantes de SGBDs decidiram acrescentar recursos presentes na tecnologia OO em seus SGBDs relacionais, denominando-os como Relacionais-Objeto (SGBDRO), ou universais.

Comentário

Tais sistemas têm ainda ampla utilização no mercado, como pode-se perceber pela lista dos principais fornecedores e produtos comerciais: Oracle, Microsoft SQL Server, SQLite, IBM (DB2 e Informix), Ingres MySQL e PostgreSQL. Esses dois últimos são exemplos de SGBDROs de código aberto.

Em relação às características dos SGBDs Relacionais-Objeto, podem ser listados a possibilidade de criação de outros tipos de dados ampliando a lista de tipos primitivos de dados pré-existentes nos SGBDs e suporte ao princípio da herança entre objetos no contexto da SQL, entre outros.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Tipos de dados

Com o avanço da computação e suas aplicações, surgiram outras necessidades de usos de bancos de dados quanto aos novos formatos dos dados. Tais aplicações foram caracterizadas pelo aumento da complexidade e necessidade de maior velocidade no processamento das consultas.

Exemplo

Como exemplos de aplicações, podem ser mencionados projetos na área de genética; análise de dados e padrões de consumo, redes sociais e Big Data. Vale destacar, também, o crescimento massivo dos dispositivos móveis, em particular os smartphones, que possibilitaram a muitos usuários produzir e consumir dados não-estruturados. Os tipos de dados serão discutidos nesta seção.

Outra justificativa para a criação de bancos de dados baseados em outros modelos de dados surgiu do fato de que novas aplicações passaram a necessitar de outros tipos de dados, além do formato pré-estabelecido e fixo que é usado nos bancos de dados relacionais. Um bom exercício mental para entender essa necessidade é o de imaginar como uma rede social seria projetada usando um conjunto de tabelas no modelo relacional. Em termos de classificação, podemos classificar os tipos de dados em estruturados, semiestruturados e não estruturados.

Dados estruturados

Nos **dados estruturados**, a estrutura é definida antes da carga de dados e segue essa estrutura de forma fixa, não importando se o atributo será ou não preenchido. O exemplo clássico de dados estruturados é a tabela no banco de dados relacional. Na tentativa de estabelecimento de padrões estruturais para pré-definir o conjunto de campos das tabelas, diversos metadados foram estabelecidos. No caso de documentos na web, um metadado foi criado com o objetivo de estabelecimento de um formato para os dados da Web, que é o padrão [Dublin Core](#) que visa descrever o formato de objetos digitais, tais como, vídeos, sons, imagens, textos e sites na web.

Para termos uma noção da dificuldade de estabelecimento de um padrão, o Dublin Core tem 15 elementos básicos que são:

1. **Title:** o título será o nome pelo qual o recurso é conhecido;
2. **Creator:** o autor que pode ser uma pessoa, uma organização ou um serviço;
3. **Subject:** o assunto que será expresso com o uso de palavras-chave;
4. **Description:** a descrição que será realizada incluindo tabelas, referências ou um texto livre;
5. **Publisher:** o publicador é uma pessoa, uma organização ou um serviço;
6. **Contributor:** o contribuidor é uma pessoa, uma organização ou serviço que auxiliou na publicação do documento;
7. **Date:** data de criação ou disponibilização do documento, sendo recomendado o uso do formato AAAA-MM-DD (ano-mês-dia);
8. **Type:** com o descritor tipo do recurso é possível descrever a categoria mais adequada do recurso;
9. **Format:** o formato pode incluir o tipo da mídia usada ou o tamanho ou as dimensões do recurso. É usado também para indicar o software, hardware ou outro equipamento necessário para apresentar ou manipular o recurso;
10. **Identifier:** recomenda-se utilizar uma chave única conforme um sistema de identificação formal, como, por exemplo, o Uniform Resource Identificador – URI, entre outros;
11. **Source:** indica a fonte do recurso que pode ser oriunda de apenas uma fonte ou de múltiplas fontes;
12. **Language:** esse descritor apresenta o código do idioma usado no recurso, com 2 letras;
13. **Relation:** a relação indica se o recurso tem algum tipo de relacionamento com outro recurso;

14. **Coverage:** esse importante descritor é usado para indicar a abrangência ou cobertura do recurso, incluindo a localização espacial, período temporal ou jurisdição aplicada;

15. **Rights:** é o descritor que indica regras relativas aos direitos autorais ou *Copyright*.

A falta de flexibilidade do modelo relacional provoca a definição, em algumas situações, de tabelas com muitos atributos, considerando todas as possibilidades de dados a serem cadastrados no banco, como, por exemplo, um atributo para o cadastro do número do bloco de um endereço de um apartamento, mesmo com o conhecimento que nem todos os prédios possuem tal divisão.

Dados semiestruturados

Nos **dados semiestruturados**, os dados são irregulares, incompletos e não é necessário seguir um esquema pré-definido. Em relação aos dados irregulares, os livros podem ser exemplos, pois eles são descritos por uma estrutura usando capítulos e seções ou podem ser descritos somente por capítulos; outro exemplo seria a descrição de uma disciplina de uma universidade que pode variar em termos de seus atributos e tópicos, de um departamento para outro, onde em um departamento faltam atributos e em outro existem atributos a mais.

Com a variação da estrutura e o avanço da web, a linguagem de marcação HTML, que é de natureza estática, se mostrou inadequada para atender a essa variação. Na tentativa de solução, foi definida a linguagem de marcação extensível [XML](#) (eXtensible Markup Language), ampliando o uso dessa categoria de dados. Com a XML, foi possível realizar o intercâmbio de dados armazenados nos bancos de dados e diminuir o problema da predefinição da estrutura do arquivo, tendo em vista que permite que as aplicações apresentem os dados com mais flexibilidade, com o usuário definindo suas próprias tags para uso na estrutura do arquivo.

Quando os dados são incompletos, nem todos os atributos são preenchidos, apesar do campo ter sido criado na tabela. Um primeiro exemplo desse caso é que nem todos os livros têm apêndice ou prefácio. Outro bom exemplo é o arquivo no formato de referência bibliográfica [Bibtex](#), que tem uma estrutura para descrição de documentos, porém, como essa estrutura varia de acordo com a obra descrita, nem todos os campos são usados. Outro exemplo, que usa dados semiestruturados são os formatos permitidos para as amostras de experimentos do National Center for Biotechnology Information - [NCBI](#), que utiliza o formato .xml para disponibilizar arquivos de testes de RNA (ácido ribonucleico). As Figuras 1 e 2 apresentam os dados de dois experimentos de RNA com uma grande diferença na estrutura dos arquivos, mais especificamente na tag Attribute, ou seja, há uma variação na estrutura dos dados presentes na base de dados.

Figura 1 - Arquivo .xml com um experimento apresentando apenas 3 atributos na tag Attributes

```

<?xml version="1.0"?>
<BioSampleSet>
  <BioSample accession="SAMN00191573" id="191573" submission_date="2011-01-19T16:46:04.450" last_update="2014-02-24T17:54:39.395" publication_date="2011-11-07T08:24:33.127" access="public">
    <Ids>
      <Id is_primary="1" db="BioSample">SAMN00191573</Id>
      <Id db="GEO" db_label="Sample name">GSM652438</Id>
      <Id db="SRA">SRS153195</Id>
    </Ids>
    <Description>
      <Title>human_T0_RNA-seq</Title>
      <Organism taxonomy_name="Homo sapiens" taxonomy_id="9606">
    </Description>
    <Owner>
      <Name>Bioinformatics Lab, Department of Medicine, Mount Sinai School of Medicine</Name>
    </Owner>
    <Models>
      <Model>Generic</Model>
    </Models>
    <Package display_name="Generic">Generic.1.0</Package>
    <Attributes>
      <Attribute display_name="source name" harmonized_name="source_name" attribute_name="source_name">primary thyroid cell line, untreated</Attribute>
      <Attribute display_name="tissue" harmonized_name="tissue" attribute_name="tissue">thyroid</Attribute>
      <Attribute display_name="treatment" harmonized_name="treatment" attribute_name="treatment">none</Attribute>
    </Attributes>
    <Links>
      <Link label="GEO Sample GSM652438" type="url">http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM652438</Link>
      <Link label="PRJNA134511" type="entrez" target="bioproject">134511</Link>
    </Links>
    <Status when="2013-05-23T00:00:00" status="live"/>
  </BioSample>
</BioSampleSet>

```

Fonte: Autor.

Figura 2 - Arquivo .xml com um experimento apresentando 16 atributos na tag Attributes

```

<?xml version="1.0"?>
<BioSampleSet>
  <BioSample accession="SAMN06565291" id="6565291" submission_date="2017-03-13T16:06:13.857" last_update="2017-06-09T01:34:45.977" publication_date="2017-06-09T00:00:00.000" access="public">
    <Ids>
      <Id is_primary="1" db="BioSample">SAMN06565291</Id>
      <Id db="SRA">SRS2045242</Id>
      <Id db="GEO">GSM2535809</Id>
    </Ids>
    <Description>
      <Title>S25018_FluTmrposCD8pos_C56</Title>
      <Organism taxonomy_name="Homo sapiens" taxonomy_id="9606">
      <OrganismName>Homo sapiens</OrganismName>
    </Description>
    <Owner>
      <Name>Systems Immunology, Benaroya Research Institute</Name>
    </Owner>
    <Contacts>
      <Contact email="SPresnell@benaroyaresearch.org">
        <Name>
          <First>Scott</First>
          <Last>Presnell</Last>
        </Name>
      </Contact>
    </Contacts>
    </Owner>
    <Models>
      <Model>Generic</Model>
    </Models>
    <Package display_name="Generic">Generic.1.0</Package>
    <Attributes>
      <Attribute display_name="source name" harmonized_name="source_name" attribute_name="source_name">Tmr-positive CD8+ sorted T-cells</Attribute>
      <Attribute display_name="tissue" harmonized_name="tissue" attribute_name="tissue">Tmr-positive CD8+ sorted T-cells</Attribute>
      <Attribute attribute_name="samplelabel">lib4059</Attribute>
      <Attribute attribute_name="sampleID">S25018</Attribute>
      <Attribute display_name="sample name" harmonized_name="sample_name" attribute_name="sample_name">1_FluTmr+CD8+_C56</Attribute>
      <Attribute attribute_name="libraryid">lib4059</Attribute>
      <Attribute attribute_name="cdnasynthesis">C1+SMARTer v1</Attribute>
      <Attribute attribute_name="libraryprep">C1+NexteraXT</Attribute>
      <Attribute attribute_name="projectid">P85-3</Attribute>
      <Attribute attribute_name="seqsite">BRI</Attribute>
      <Attribute attribute_name="c1plateid">1771023180</Attribute>
      <Attribute attribute_name="c1capturesite">C56</Attribute>
      <Attribute attribute_name="c1captureannotation">single cell</Attribute>
      <Attribute attribute_name="flowcellid">C4WKJACXX</Attribute>
      <Attribute attribute_name="runchemistry">TruSeq SBS Kit v3</Attribute>
      <Attribute attribute_name="lane">8,7</Attribute>
    </Attributes>
    <Links>
      <Link label="GEO Sample GSM2535809" type="url">http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM2535809</Link>
      <Link label="PRJNA379015" type="entrez" target="bioproject">379015</Link>
    </Links>
    <Status when="2017-06-08T18:37:18.947" status="live"/>
  </BioSample>
</BioSampleSet>

```

Fonte: Autor.

Exemplo

Outro exemplo sobre os dados semiestruturados são as páginas web contendo dados sobre documentos em estruturas variadas. Essa característica facilitou a publicação de dados armazenados em bancos de dados na web, possibilitando a independência de armazenamento e permitindo a visualização de dados oriundos de fontes heterogêneas (diferentes SGBDs), alcançando assim uma independência de apresentação.


Dados não estruturados

Quanto aos **dados não estruturados**, esses não podem ser descritos como em tabelas pré-definidas em termos de estrutura. Estão incluídos nesse grupo, dados como comentários em redes sociais, arquivos de vídeo e de áudio, imagens em seus diversos formatos, mensagens publicadas em redes sociais, arquivos de textos diversos, mensagens instantâneas de aplicativos, relatórios distintos dos exportados em consultas em bancos de dados, mensagens de correio eletrônico (e-mails), entre outros. Esse tipo de dado representa mais que 80% dos dados atuais usados no mundo e seu volume continua em franco crescimento, o que pode ser comprovado pela quantidade de empresas que foram criadas nos últimos anos.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Bancos de dados NoSQL

Tendo em vista as características dos tipos de dados apresentados anteriormente, houve a necessidade de pesquisa para o desenvolvimento de modelos de dados mais adequados para uso com as aplicações mais recentes, como é o caso das redes sociais, por exemplo. Para melhor entendermos as vantagens no uso desse modelo de dados, vamos compará-lo ao modelo relacional. A comparação será realizada em relação às seguintes características:

 Clique nos botões para ver as informações.

Linguagem de manipulação



A característica principal dos modelos NoSQL é que as operações de criação e gerenciamento dos dados não estão limitadas a serem executadas apenas com a linguagem de consulta estruturada (SQL). Nesses modelos, a SQL não é usada como a única linguagem de consulta. A tecnologia é na verdade um conjunto de tecnologias que facilitam a resolução dos problemas para uma ampla variedade de aplicações que não são indicadas para uso no modelo relacional. Em termos de linguagem, há inclusive outras linguagens que podem ser usadas, como a [UnQL](#) (*Unstructured Query Language*), ou simplesmente UnQL (pronunciada como "Uncle"), que é usada em SGBDs como o [Couchbase](#). A sintaxe da linguagem tem variação entre SGBDs e por modelos de dados NoSQL, com a existência de linguagens específicas como o caso da linguagem [Cypher](#) usada pelo [Neo4J](#), que é um SGBD baseado em grafos e que será ainda estudado na disciplina.

Quando comparamos os modelos de dados NoSQL e relacional observamos esta diferença: os bancos de dados relacionais usam um esquema pré-definido (fixo) para estruturar suas tabelas, enquanto os bancos de dados NoSQL possuem um esquema dinâmico. Esse tipo de esquema permite que o banco de dados armazene registros que não tenham o mesmo conjunto de atributos. Os bancos de dados relacionais têm em sua estrutura apenas tabelas ou relações, porém, no caso dos bancos de dados NoSQL, são usadas quatro estruturas que foram definidas para atender a diferentes demandas surgidas dos diversos formatos de dados mais recentes. Assim, podem ser usados modelos baseados em documentos; coleções de pares chave-valor; armazenamento orientado em colunas; e modelo baseado em grafos.

Poder de execução de consultas complexas



Ainda sobre as principais diferenças entre os modelos, o relacional é mais adequado para o ambiente em que são necessárias consultas complexas, enquanto os NoSQL não, pois normalmente contêm um conjunto de atributos para a publicação de uma base de dados. O relacional tem um poder maior na definição de consultas pois, normalmente, há a presença de muitas tabelas em uma aplicação relacional e com o cruzamento de dados pode-se aumentar as possibilidades advindas das junções, que podem ser criadas nas respostas às consultas.

Desempenho



Além disso, um banco de dados relacional pode apresentar dificuldades para a execução de consultas em grandes volumes de dados com alto desempenho, entre outras razões por conta da necessidade das junções que são realizadas. Por esse motivo, os modelos NoSQL também foram projetados para o armazenamento de dados em bancos distribuídos e, dessa forma, são muito indicados ao uso em aplicações com grandes volumes de dados, do tipo Big Data. Os bancos de dados relacionais não têm um ajuste para permitir o crescimento do processamento de forma a atender o aumento do volume de dados. Há a necessidade de ajustes na configuração do SGBD, o que nem sempre é trivial de ser realizado. Com o volume de dados aumentando consideravelmente devido ao crescimento do uso da web (redes sociais em particular), a necessidade de uma solução para permitir o controle do gerenciamento desse volume de dados se tornou urgente.

Escalonamento



Escalonamento Existem dois tipos de escalonamento de servidores: o vertical, no qual há um aumento (*upgrade*) no servidor (*scale up*) para a busca de uma maior capacidade de processamento; e o escalonamento horizontal, em que ocorre um aumento do número de servidores (*scale out*), buscando paralelismo. A primeira ideia de solução para aumentar o poder de processamento dos SGBDs relacionais foi o escalonamento vertical. Porém, mesmo assim, a conclusão foi que o volume de dados continuaria a ser um fator limitador, com impacto direto no desempenho do sistema. Considerou-se o uso da distribuição dos dados em vários nós (servidores de bancos de dados), ideia que tem como consequência a implementação de ajustes para a utilização no processamento de consultas, assim como no projeto de banco de dados distribuídos.

Os bancos de dados NoSQL possuem escalonamento horizontal e, com isso, podem ter um aumento na capacidade de processamento. A distribuição do banco de dados em várias máquinas é feita pelo particionamento dos dados. O processo é também conhecido como *sharding*. O processo de *sharding* objetiva trabalhar com o escalonamento horizontal, paralelizando seus dados em vários servidores. O próprio volume dos dados por máquina é em muito minimizado como consequência do processo de distribuição. Conjuntos de dados de tamanho menor são mais fáceis de serem acessados, atualizados e gerenciados;

Em aplicações que necessitam de alta capacidade de operações transacionais, os bancos de dados relacionais são os mais adequados, pois têm um controle mais rígido sobre essas operações. Isso se deve ao fato de seguirem as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Uma transação consiste em uma unidade de trabalho ou bloco de comandos que tem um significado lógico no sistema de banco de dados, como uma operação bancária, por exemplo. Já os bancos de dados NoSQL são baseados no Teorema CAP (Consistência, Disponibilidade e tolerância à Partição).

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Atividade

1. Assinale a afirmativa incorreta em relação aos modelos de dados hierárquico, em rede, relacional, relacional-objeto e NoSQL:

- a) ONOSQL tem os modelos chave-valor, orientado a documentos e de modo colunar e em grafos.
- b) No modelo de dados hierárquico todo o projeto do banco necessitava ser definido de forma hierárquica.
- c) Na organização do modelo em rede utilizava-se ponteiros para o acesso aos registros.
- d) O modelo relacional-objeto permite criar tipos de dados e usar herança.
- e) O relacional tem fundamentação matemática, usa a SQL e segue o Teorema CAP.

2. Assinale a opção que indica a relação incorreta quanto ao tipo de dado e o arquivo usado:

- a) E-mail – semiestruturado.
- b) Vídeos – não estruturado.
- c) Tabela – relacional.
- d) Arquivo .xml – semiestruturado.
- e) Rede social – não estruturado.

3. Marque a opção que assinala a opção falsa quanto às diferenças dos modelos NoSQL para o modelo de dados relacional:

- a) Os modelos NoSQL podem responder consultas em SQL, porém existem linguagens próprias para explorar todas as suas potencialidades.
- b) Uma das maiores vantagens dos modelos NoSQL está na facilidade de responder consultas complexas.
- c) O relacional necessita de uma pré-definição da estrutura de suas tabelas.
- d) Os NoSQL têm escalonamento vertical e o relacional, horizontal.
- e) O relacional tem o controle de transações baseado nas propriedades ACID.

Notas

Referências

Próxima aula

Na próxima aula, daremos continuidade aos estudos vistos aqui, abordando os seguintes conteúdos:

- Características das diferentes categorias de bancos de dados NoSQL: orientado a documentos, chave-valor, colunas e grafos;
- Diferenças entre as categorias e suas vantagens;
- Tipos de aplicações adequadas para cada modeloNoSQL.

Explore mais

Para aprofundar seus conhecimentos fica a sugestão de leitura dos seguintes trechos do livro *Sistema de banco de dados* de Elmasri, R.; Navathe, S.B.

Cap. 5 – O modelo de dados relacional e as restrições em bancos de dados relacionais

Seção 24.1.1 – Surgimento de sistemas NOSQL.