

Atividade 1

- Universidade Federal de Sergipe - CCET - Departamento de Computação
- COMP0455 - BANCO DE DADOS I (2023.1 - T01)
- Prof. Dr. André Britto de Carvalho
- Pedro Vinícius de Araújo Barreto
- João Filipe de Araújo Santos Rezende

Sistemas de Gerenciamento de Banco de Dados (SGBD) e modelos de dados

No começo do uso dos computadores, foi enxergado o potencial dessas máquinas de armazenarem e processarem dados. As primeiras abordagens de uso do computador como tecnologia fundamental no manuseio de dados consistia na utilização de arquivos e programas não-estruturados, porém logo percebeu-se que estas levavam à grande complexidade cognitiva, de tempo e de espaço, além de riscos de segurança, ou seja, preocupações de extrema prioridade para empresas de todo porte.

Era preciso então uma nova abordagem, que fosse eficiente, segura, simples. A partir dessa necessidade, nasceram os SGBDs.

Um SGBD consiste em um conjunto de dados armazenados, isto é, o banco de dados em si, e em um conjunto de programas úteis para lidar com esses dados. Um sistema capaz de encapsular os dados, trazendo as características citadas anteriormente e que funcionasse como uma plataforma de gerenciamento que possibilitasse tanto o armazenamento quanto a análise dos dados, esta última tendo mais importância com as ferramentas estatístico-computacionais do campo de aprendizagem de máquina, que trazem *insights* sobre os dados coletados para quem estiver interessado: indivíduos, empresas, universidades e governos.

Os programas para lidar com os dados podem cumprir as mais variadas funções, como:

- Implementação de baixo nível das estruturas de dados (Árvores B, Grafos, etc) no sistema de arquivos do Sistema Operacional do computador, buscando performance e confiabilidade.
- Definição de estruturas lógicas (tabelas, hierarquias, etc) sobre as quais os dados serão representados
- Manipulação e atualização das estruturas já definidas
- Garantia de restrições de segurança ou consistência
- Estabelecer uma interface básica de conexão entre os programas supracitados com as aplicações interessadas no consumo dos dados

SGBDs modernos buscam um fundamento, ou modelo, sob o qual o banco será abstraído e, a partir da abstração, modelado da maneira apropriada. Essas ferramentas de abstração são conhecidas como **modelos de dados**. Podemos citar alguns modelos, como:

Hierárquico e Rede

São modelos legados, que não passaram no teste do tempo, mas serviram de base e inspiração para modelos mais modernos.

Eram baseados em noções de relações primitivas para a atribuição de conexões entre o que os dados representavam no mundo real, ou “minimundo”. Enquanto o hierárquico era organizado ao redor de árvores onde os nós representavam dados estruturados, o de rede baseava-se numa estrutura de grafo generalizada.

Alguns exemplos de SGBDs baseados nesses modelos são:

- Rede: IMAGE, IDS, CODASYL
- Hierárquico: IMS e RDM Mobile

Objeto-relacional

Esse modelo é o resultado da evolução de dois modelos anteriores:

- Relacional, criado por Edgar Codd, baseado na teoria matemática de relações, explicitando a estrutura (ou esquema) de cada relação. Esse modelo teve um fortíssimo sucesso, eventualmente superados os modelos hierárquico e de rede. Junto com o advento do modelo relacional, uma linguagem de dados declarativa foi criada prover as utilidades de definição e manipulação dos SGBDs que seguiram esse modelo, a **SQL**.
- Orientado a objetos: Com o advento de linguagens de programação orientadas a objetos, que permitiam a criação de componentes de software que encapsulavam informações desnecessárias e expunham métodos de comunicação entre si, não demorou muito para que acadêmicos e profissionais buscassem trazer esse paradigma para o campo dos banco de dados, provendo suporte para classes, objetos e herança.

O modelo relacional passou a ser fortemente criticado devido ao seu baixo suporte pelo paradigma Orientado a Objetos, levando à uma fusão dos paradigmas existentes em um novo que trazia tanto a possibilidade de criação de relações e de seus esquemas, tendo o uso do SQL para sua definição e manipulação, mas também permitia a modelagem das relações como classes, de tal forma que os atributos dos objetos passaram a ser equivalente às colunas nas tabelas relacionais, e cada instância da classe é equivalente a uma tupla da relação. Hoje, podemos dizer que essa evolução do modelo Relacional virou o padrão da indústria, a ponto de passarem a surgir técnicas e ferramentas para fazer o mapeamento objeto-relacional, os *ORMs*.

Os principais SGBDs objeto-relacionais da atualidade são Microsoft Server, Oracle Database, MySQL, PostgreSQL e MariaDB.

NoSQL

Uma das principais críticas aos SGBDs relacionais é que eles são muito rígidos no formato dos dados o qual eles aceitam. Isso significa que sempre existiu um gasto

de recursos com a necessidade de garantir a compatibilidade com o banco. O advento da Web trouxe inovações no campo de armazenamento e representações de dados, como o JSON, levando a um confronto com a rigidez relacional e uma busca por alternativas.

O NoSQL (*Not Only SQL*) não é um modelo de dados em si, mas uma categoria de modelos de dados que quebraram com a homogenia do modelo relacional e trazem (ainda são um fenômeno relativamente recente) inovações que permitem a criação de aplicações de maneira rápida e escalável, ao mesmo tempo que eles sacrificam consistência dos dados armazenados.

Os principais modelos NoSQL são aqueles baseados em:

- Documentos: Esse modelo assume que os dados são encapsulados por documentos, que podem estar em formatos como XML, YAML, JSON ou formatos binários como BSON, com cada documento associado à uma chave identificadora no banco. Diferentes implementações utilizam diferentes métodos de organizar conjuntos de documentos. Exemplos de bancos baseados em documentos são DynamoDB, MongoDB e Firestore.
- Grafos: Grafos são capazes de modelar relações do mundo real e alguns bancos de dados fazem uso dos nós, arestas e outras propriedades dos grafos para armazenar e manipular os dados de maneira escalável, fazendo uso de mais de 60 anos de algoritmos de grafos desenvolvidos na academia e na indústria. Os melhores exemplos de bancos baseados em grafos são o Neo4j e o Azure Cosmos DB (parte da plataforma cloud da Microsoft, Azure)
- Chave-valor: Bancos de dados baseados nessa abordagem utilizam um vetor associativo de chaves-valores para a representação dos dados, de tal maneira que as chaves são únicas. Os principais bancos baseados em chave-valor são Couchbase, Azure Cosmos DB, Dynamo DB e Redis. Este último faz uso de um sistema de cache eficiente.
- Tabular: É possível modelar um banco de chave-valor em duas dimensões. Isto é feito por bancos de coluna ampla (*wide-column*), ou tabular, que usam tabelas, linhas e colunas (assim como o relacional), mas os nomes e o formatos das colunas podem variar linha-a-linha numa mesma tabela. Exemplos de bancos tabulares são Cassandra, Bigtable e DynamoDB

SGBDs particulares

Apresentamos as principais características, modelos de dados e aplicações de alguns SGBDs particulares.

PostgreSQL

Oracle 23c

Cache Database

Neo4J

SQL Server

Firebase

O sucesso da computação em nuvem permitiu a oferta de vários serviços, sejam eles de computação, armazenamento ou monitoramento. Um dos serviços oferecidos é o BaaS (*Backend as a Service*), que permite delegar à plataforma nuvem muitas das atividades associadas com a lógica backend de serviços web.

Uma dessas plataformas, a GCP (*Google Cloud Platform*) oferece a Firebase, um conjunto de serviços BaaS. Um dos serviços cuida especificamente do banco de dados de uma aplicação Web.

MongoDB

Um dos melhores representantes da categoria NoSQL é o MongoDB, um dos bancos de dados baseados em documentos mais avançados da atualidade. O MongoDB

Redis

Uma das alternativas de armazenamento mais populares entre desenvolvedores, o banco Redis faz parte da categoria NoSQL, usando o modelo de dados Chave-Valor, porém com um sistema de cache eficiente.

Primeiramente, é importante entender que o “valor” no modelo chave-valor adotado por Redis não está limitado a tipos primitivos que programadores estão acostumados (inteiros, floats, chars), mas embarca estruturas de dados como listas, hashes, conjuntos, hyperlogslogs, entre outros. Os dados que essas estruturas armazenam estão sempre na memória RAM e no cache. Isso difere da maioria dos SGBDs tradicionais, que fazem uso de árvores-B representadas no disco rígido, que é mais lento que a memória principal em ordens de magnitude.

Redis foi construído usando uma arquitetura cliente-servidor. O servidor interno do Redis faz algumas operações para persistir os dados no disco, de tal maneira que a eficiência não seja muito prejudicada. Veja que ainda existe um pequeno risco de não-persistência dos dados. Aplicações onde tal característica seja crítica não são apropriadas ao banco Redis.

Veja que como Redis mapeia chaves para estruturas de dados, é completamente possível que esse mapeamento também ocorra para outras estruturas que servem de suporte para outros modelos NoSQL: Grafos (RedisGraph), documentos

(RedisJSON), séries temporais, etc. O conjunto dessas e outras extensões sobre Redis é chamado de *Redis Stack*.

Inicialmente, Redis foi muito utilizado em aplicações onde a escalabilidade devido ao seu sistema de cache podia ser aproveitada: Instagram, Twitter, entre outros.

Referências

- **Database System Concepts** (7th ed.). Abraham Silberschatz, Henry F. Korth, S. Sudarshan. McGraw-Hill Education, 2020.
- **NoSQL**. Wikipedia, acessado em junho de 2023. Disponível aqui
- **Redis Stack**. Redis.io, acessado em junho de 2023. Disponível aqui
- **The Infrastructure Behind Twitter: Scale**. Mazdak Hashemi. Twitter Engineering, 2017. Disponível aqui
- **Storing hundreds of millions of simple key-value pairs in Redis**. Mike Krieger. Instagram Engineering, Medium, 2011. Disponível aqui