

Banco de dados orientado a documentos

Licenciatura em Engenharia Informática

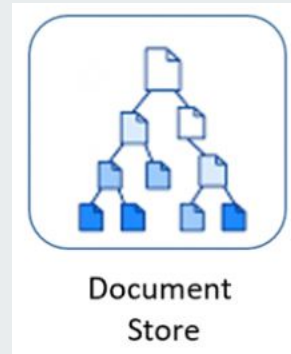
Sistemas Gestores de Bases de Dados

Professores: David Aveiro e Fábio Mendonça

Licenciatura em Engenharia Informática

Trabalho realizado por:

A- Francisco Afonseca
B- João Francisco
C- Leonardo Ferreira
D- Lucas Neves
E- Paulo Alves
F- Renato Pêssego



Funchal, 12 de Outubro de 2023

Introdução



Atualmente, o armazenamento de dados tornou-se uma parte fundamental de muitas empresas e organizações. A quantidade de dados gerados diariamente é enorme e, por isso, é preciso saber escolher a melhor forma de armazená-los para poderem ser acessados e utilizados de maneira eficiente. É nesse contexto que entram os bancos de dados, sistemas que permitem armazenar, organizar e gerenciar muitos dados.

Existem diversos tipos de bases de dados disponíveis e cada um possui suas próprias características e funcionalidades. [A1]

Nesta apresentação, falaremos de um dos mais populares, o NoSQL Documentos.

[A1] - <https://www.impacta.com.br/blog/diferencas-entre-sql-e-nosql-comparando-bancos-de-dados-relacionais-e-nao-relacionais/>

Base de dados não relacional “NoSQL”

As bases de dados NoSQL são alternadamente designadas por "não relacionais", "BDs NoSQL" ou "não SQL" para realçar o facto de que conseguem processar volumes enormes de dados não estruturados e em constante mudança de formas diferentes de uma base de dados relacional (SQL) com linhas e tabelas. [A2]

Estas bases de dados NoSQL são altamente escaláveis e são frequentemente utilizadas em aplicações web e móveis que precisam lidar com o excesso de dados . Elas são projetadas para serem distribuídas em múltiplas máquinas, o que as torna altamente disponíveis e tolerantes a falhas. [A1]

Resumidamente , este tipo de bases de dados é favorecido , sendo que é extremamente eficaz no processamento de dados imprevisíveis e tem uma velocidade de consulta rápida. [A2]

ORADOR:

As bases de dados do tipo não relacional (“NoSQL”) são BD que não tem a capacidade de processar grandes volumes de dados não estruturados e com diferenças significativas em comparação as bases de dados SQL com linhas e colunas . Apesar das suas restrições , este tipo de BD são favorecidas pelo facto de terem uma alta e eficiente velocidade na consulta de dados dentro das mesmas . Sendo assim usada em muitos dispositivos recentes , como dispositivos móveis , web e jogos . Proporcionando excelentes experiências aos utilizadores .

CONTEÚDO:

As tecnologias NoSQL já existem desde a década de 60 com vários nomes, mas estão a crescer em popularidade à medida que o panorama dos dados muda e os programadores precisam de se adaptar para processarem o grande volume e o vasto leque de

dados gerados na nuvem, em dispositivos móveis, redes sociais e macrodados. [A2]

Motivações para esta abordagem incluem: simplicidade de projeto, escalonamento "horizontal" mais simples para clusters de máquinas (o que é um problema para bancos de dados relacionais) e controle mais refinado sobre a disponibilidade. As estruturas de dados usadas pelos bancos de dados NoSQL (e.g., chave-valor, coluna larga, grafo ou documento) são diferentes daquelas usadas por padrão em bancos de dados relacionais, tornando algumas operações mais rápidas em NoSQL. A adequação particular de um determinado banco de dados NoSQL depende do problema que ele deve resolver. Algumas vezes as estruturas de dados usadas por bancos de dados NoSQL também são vistas como "mais flexíveis" que tabelas de bancos de dados relacionais.

Muitos armazenamentos NoSQL comprometem a consistência (no sentido do teorema CAP) em favor da disponibilidade, tolerância a partição e velocidade. Barreiras para a grande adoção de armazenamentos NoSQL incluem o uso de linguagens de consulta de baixo nível (em vez de SQL, por exemplo, a falta de capacidade de realizar junções ad-hoc entre tabelas), falta de interfaces padronizadas e grandes investimentos anteriores em bancos de dados relacionais existentes.^[2] A maioria dos armazenamentos NoSQL carece de transações ACID verdadeiras, embora alguns bancos de dados, como MarkLogic, Aerospike, FairCom c-treeACE [...] tornaram-as centrais em seus projetos.[A3]

Resumidamente , este tipo de bases de dados é favorecido , sendo que é extremamente eficaz no processamento de dados imprevisíveis e velocidade de consulta rápida . [A2]

Muitos armazenamentos NoSQL comprometem a consistência (no sentido do teorema CAP) a favor da disponibilidade, tolerância a

partição e velocidade. Barreiras para a grande adoção de armazenamentos NoSQL incluem o uso de linguagens de consulta de baixo nível (em vez de SQL, por exemplo, a falta de capacidade de realizar junções ad-hoc entre tabelas), falta de interfaces padronizadas e grandes investimentos anteriores em bases de dados relacionais existentes. A maioria dos armazenamentos NoSQL carece de transações ACID verdadeiras, embora algumas bases de dados, como MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (embora tecnicamente bases de dados NewSQL), Symas LMDB e OrientDB tornaram-as centrais nos seus projetos.[A3]

[A2]-

<https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-nosql-database>

[A3]- <https://pt.wikipedia.org/wiki/NoSQL>

Bases de dados orientadas a documentos



Uma base de dados orientado a documentos, ou armazenamento por documentos (document store), é um modelo de bases de dados projetada para armazenar, recuperar e gerir formações orientadas a documentos, também conhecidas como dados semi-estruturados. Bases de dados orientadas a documentos são uma das principais categorias de bases de dados NoSQL. [A4]

Principais vantagens em usar bases de dados orientadas a documentos [A5] :

- Permite que os desenvolvedores armazenem e consultem dados usando o mesmo formato de modelo de documento que usam no código do aplicativo;
- A natureza flexível, semi estruturada e hierárquica dos documentos e das bases de dados de documentos permite que elas evoluem conforme as necessidades dos aplicativos;

ORADOR:

Nomeadamente, nesta apresentação teremos em conta unicamente as bases de dados orientadas a documentos.

A particularidade deste tipo de bases de dados é o facto de serem projetadas para armazenar, recuperar e gerir informações orientadas a documentos, como o próprio nome indica, também conhecidos pelos dados semi-estruturados.

O modelo de documentos também funciona bem com catálogos, perfis de utilizadores e sistemas de gerenciamento de conteúdo, onde cada documento é único e evolui com o passar do tempo. Nos formatos O Amazon DocumentDB (com compatibilidade com o MongoDB) e o MongoDB são bases de dados de documentos populares que fornecem APIs eficientes e intuitivas para desenvolvimento flexível e interativo. [A5]

As principais vantagens deste tipo de bases de dados é permitirem que os desenvolvedores armazenem e consultem os

dados, usando o mesmo formato de documentos que usam no aplicativo e o facto de terem uma natureza flexível e hierárquica dos documentos, dois grandes fatores na construção de bases de dados, e também o facto de permitirem que este tipo de BD evoluem conforme necessário.

O conteúdo armazenado pode ser consultado e examinado.

CONTEÚDO:

Uma base de dados orientada a documentos, ou armazenamento por documentos (document store), é um modelo de bases de dados projetada para armazenar, recuperar e gerir informações orientadas a documentos, também conhecidos como dados semi-estruturados . Bases de dados orientadas a documentos são uma das principais categorias de bases de dados NoSQL e a popularidade do termo "Base de dados orientada a documentos" têm crescido^[1] com o uso do termo NoSQL propriamente dito.

O conceito central de uma base de dados orientada a documentos é a noção de um *documento*. Apesar de cada implementação de bases de dados orientada a documentos diferir-se nos detalhes desta definição, em geral, todas elas admitem que os documentos encapsulam e codifiquem dados (ou informação) em alguns formatos ou codificações padrões. Codificações em uso incluem XML, YAML, JSON e BSON, bem como formas binárias como PDF e documentos do Microsoft Office (MS Word, Excel e assim por diante). [A4]

No código do aplicativo, os dados costumam ser representados como um objeto ou um documento do tipo JSON porque esse é um modelo de dados eficiente e intuitivo para os desenvolvedores. Os bancos de dados de documentos facilitam para que os desenvolvedores armazenem e consultem dados usando o mesmo

formato de modelo de documento que usam no código do aplicativo. A natureza flexível, semi estruturada e hierárquica dos documentos e das bases de dados de documentos permite que eles evoluem conforme as necessidades dos aplicativos. O modelo de documentos funciona bem com catálogos, perfis de utilizadores e sistemas de gerenciamento de conteúdo, onde cada documento é único e evolui com o passar do tempo. O Amazon DocumentDB (com compatibilidade com o MongoDB) e o MongoDB são bases de dados de documentos populares que fornecem APIs eficientes e intuitivas para desenvolvimento flexível e interativo. [A5]

Os documentos armazenados são similares entre si, porém não necessariamente devem possuir a mesma estrutura . Podemos imaginar bases de dados como sendo BD do tipo “key-value”, pois possuem uma estrutura semelhante, porém o conteúdo do “valor” armazenado é pesquisável e examinável. [A33]

[A4]-

https://pt.wikipedia.org/wiki/Banco_de_dados_orientado_a_documentos

[A5]- <https://aws.amazon.com/pt/nosql/>

[A33]-

<https://pt.linkedin.com/pulse/bancos-de-dados-orientados-documentos-tiago-ferreira-fernandes>

Teorema CAP



Segundo Brewer [E1] este teorema consistente em três pilares: consistência, disponibilidade e tolerância na partição.

Consistência: Todos os nós retornam a informação mais recente. Uma mudança num nó reflete no nó que o conecta.

Disponibilidade: Cada pedido feito por um nó que não falhou deve retornar uma resposta.

Tolerância na partição: O sistema continua operacional mesmo que algumas das mensagens não tenham sido recebidas ou com algum atraso devido à rede que ligava os nós.

Será este modelo o mais atualizado? Sabendo que este teorema têm mais de 20 anos? [E2]

ORADOR:

Em vários serviços da sociedade é necessário garantir que as operações ocorram de maneira normal e expectável. Seja na sociedade, seja no funcionamento de base de dados. Para isso, Brewer, elaborou um teorema que serviu de orientação no desenvolvimento de sistemas durante algum tempo. Afirmou só era possível ter dois destes três atributos em simultâneo.

Por exemplo, imaginemos dois nós ligados que guardam a informação de quanto dinheiro o cliente tem no banco. Imaginemos que o cliente tira dez euros da sua conta. Quando o cliente fizer um pedido para o sistema a fim de saber quanto dinheiro têm, os dois nós têm de retornar a mesma informação a fim de manter a consistência.

A disponibilidade afirma que deve existir sempre uma

resposta depois de efetuado um pedido ao sistema, nem que seja uma resposta que não está atualizada.

A tolerância na partição significa se algum dos nós não estiver disponível o sistema deve continuar operacional.

O nosso grupo não concorda plenamente com este teorema. Apesar de algumas partes estarem corretas achamos que o modelo encontra-se desatualizado tendo surgido um teorema similar ao de Brewer que nos parece mais correto. O teorema "PACELC". Em quê difere do teorema de Brewer?

É uma adaptação do teorema de Brewer que mostra a importância da latência e perda da consistência.

A latência é o atraso, normalmente em ms, de um pacote que estamos a enviar ou a receber demora a chegar ao data-center de destino, está ligada diretamente à distância aos data-centers, sendo que podemos imaginar isto num jogo online. Se estivermos a jogar na nossa casa ligados a servidor do Peru a nossa latência vai ser muito mais alta do que se estivessemos conectados a um servidor de Espanha.

A perda de consistência na maior parte das vezes advém de alta latência ou de falhas na rede.

Por exemplo o sistema PNUTS do Yahoo incorre em inconsistência ao manter cópias remotas de forma assíncrona. No entanto, torna a cópia mestre local, o que diminui a latência. Esta estratégia funciona bem na prática porque os dados de um único usuário são naturalmente particionados de acordo com a localização do usuário. Idealmente, a versão

mestre dos dados de cada usuário está próximo. O Facebook usa a estratégia oposta: a cópia mestre está sempre em um local, então um usuário normalmente tem uma cópia mais próxima, mas potencialmente não atualizada.

CONTEÚDO:

“O Teorema CAP, também chamado de Teorema de Brewer, diz que em um sistema como uma base de dados somente é possível ter duas das seguintes características”[E1][E2]: Consistência, Disponibilidade e Tolerância à falha[E17]

“Consistência é a variável que garante que se um dado foi alterado, iremos receber a versão mais recente desse dado, independentemente de qual for o nodo que vai responder ao pedido. Para que a consistência aconteça, deve existir uma sincronização dos nodos toda vez que algum valor for alterado. Podemos dizer que cada nodo deve manter uma cópia do estado do sistema ou de alguma maneira ter acesso ao estado mais atualizado do mesmo.”[E17]

“Disponibilidade é uma variável que garante que o sistema estará disponível sempre que for necessário.”[E17]

“A tolerância à falha é uma variável que garante que se algum dos nodos do sistema está em baixo, o sistema ainda será capaz de atender aos pedidos.”[E17]

Dizer que devemos escolher duas das três características(CAP) é uma alegação enganosa em diversas frentes. Primeiro, como as partições são raras, há poucos motivos para perder C ou A quando o sistema não está particionado. Segundo, a escolha entre C e A pode ocorrer muitas vezes dentro do mesmo sistema com granularidade muito fina(número de vezes que é dividida); Não apenas os subsistemas podem fazer escolhas diferentes, mas a escolha pode mudar de acordo com a operação ou mesmo com os dados específicos ou com o usuário envolvido. A disponibilidade é obviamente contínua de 0 a 100 por cento, mas também existem muitos níveis de consistência, e até mesmo as partições têm diferenças entre si, incluindo divergências dentro do sistema sobre a existência de uma partição.[E2]

“Na sua interpretação clássica, o teorema CAP ignora a latência, embora na prática a latência e as partições estejam profundamente relacionadas. Operacionalmente, a essência do teorema “CAP” ocorre durante um intervalo, um período em que o programa deve tomar uma decisão fundamental – a decisão de partição: cancelar a operação e assim diminuir a disponibilidade, ou prosseguir com a operação e, portanto, correr o risco de inconsistência. Tentar novamente a comunicação para obter consistência, por exemplo, via Paxos ou um commit em duas fases, apenas atrasa a decisão. Em algum momento o programa deverá tomar a decisão, tentar

novamente a comunicação indefinidamente é, em essência, escolher consistência em vez de disponibilidade. Assim, pragmaticamente, uma partição é um limite de tempo na comunicação. Não conseguir consistência dentro do limite de tempo implica uma partição e, portanto, uma escolha entre consistência e disponibilidade para esta operação. Esses conceitos capturam a questão central sobre que sistema escolher em relação à latência: os dois lados estão avançando sem comunicação? Esta visão pragmática dá origem a várias consequências. A primeira é que não existe uma noção global de partição, uma vez que alguns nós podem detectar uma partição e outros não. A segunda consequência é que os nós podem detectar uma partição e entrar em um modo de partição – uma parte central da otimização da consistência e da disponibilidade. Finalmente, esta visão significa que quem implementa os sistemas podem definir limites de tempo intencionalmente de acordo com os tempos de resposta alvo. Sistemas com limites mais rígidos provavelmente entrarão no modo de partição com mais frequência e em momentos em que a rede estiver meramente lenta e não estiver realmente particionada. Às vezes faz sentido abrir mão da consistência para evitar a alta latência de manter a consistência em uma área ampla. O sistema PNUTS do Yahoo incorre em inconsistência ao manter cópias remotas de forma assíncrona. No entanto, torna a cópia mestre local, o que diminui a latência. Esta estratégia funciona bem na prática porque os

dados de um único usuário são naturalmente particionados de acordo com a localização (normal) do usuário. Idealmente, o mestre de dados de cada usuário está próximo. O Facebook usa a estratégia oposta: a cópia mestre está sempre em um local, então um usuário remoto normalmente tem uma cópia mais próxima, mas potencialmente obsoleta. No entanto, quando os usuários atualizam suas páginas, a atualização vai diretamente para a cópia master, assim como todas as leituras do usuário por um curto período de tempo, apesar da maior latência. Após 20 segundos, o tráfego do usuário reverte para a cópia mais próxima, que nesse momento deverá refletir a atualização.”[E2]

“Daniel J. Abadi” publicou o artigo “Consistency Tradeoffs in Modern Distributed Database System Design”, sendo que pouco depois “Brewer” divulgou o seu artigo “CAP Twelve Years Later: How the “Rules” Have Changed”. O nosso grupo afirma e concorda com ambos, tanto “Brewer” como “Daniel J. Abadi” de que o teorema “CAP” encontra-se desatualizado. Brewer acha que para resolver esse problema consistência, disponibilidade e tolerância à falha não são suficientes, concordando assim, com o que foi dito por “Daniel J. Abadi” no teorema “PACELC” que é necessário ter atenção com a latência e a perda de consistência. Sendo que a latência pode ser resolvida com proximidade aos “datacenters” ou adotar a estratégia da Yahoo. Para aumentar a consistência poderíamos adotar a estratégia do Facebook, embora

comprometa a latência, aumentamos a consistência. [E19]

[E1] https://en.wikipedia.org/wiki/CAP_theorem

[E2]

<https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed/>

[E17] https://www.juanrivillas.com/blog/cap_theorem/

[E19]

<https://pronnus.com.br/blog/o-que-e-latencia-e-qual-a-sua-importancia/>

Formato JSON e BSON



O formato JSON é o formato geral de bases de dados document. JSON significa JavaScript Object Notation, e como o nome indica, é uma notação de um objeto em javascript, ou seja, deposita e carrega dados relativos a dito objeto. O formato é legível apenas olhando para ele, sendo isso uma das suas vantagens. [D3]

O formato BSON é o formato usado em alguns programas, como o MongoDB. BSON significa Binary javascript Object Notation, e a sua principal diferença é o facto de este ser binary-coded, logo, feito e escrito em binário e não legível apenas olhando para ele. [D1][D2]

Conteúdo:

[D3] O formato JSON é o formato geral de bases de dados document. JSON significa JavaScript Object Notation, e como o nome indica, é uma notação de um objeto em javascript, ou seja, deposita e carrega dados relativos a dito objeto. O formato é legível apenas olhando para ele, sendo isso uma das suas vantagens. [D2] O formato BSON é o formato usado em alguns programas, como o MongoDB. BSON significa Binary javascript Object Notation, e a sua principal diferença é o facto de este ser binary-coded, logo, feito e escrito em binário e não legível apenas olhando para ele. Objetos JSON são containers associativos, nos quais uma chave de string é mapeada para um valor (que pode ser um número, uma string, um booleano, um array, um valor vazio - nulo, ou até mesmo outro objeto). Quase qualquer linguagem de programação tem uma implementação para esta estrutura de dados abstrata - objetos em JavaScript, dicionários em Python, tabelas de hash em Java e C#, arrays associativos em C++, e assim por diante. Objetos

JSON são fáceis de entender para os humanos e de analisar e gerar para as máquinas. BSON é uma notação de objeto binária codificada (Binary JavaScript Object Notation) — uma notação de objeto textual amplamente usada para transmitir e armazenar dados em aplicativos baseados na web. [D1] O JSON é mais fácil de entender, pois é legível por humanos, mas em comparação com o BSON, ele suporta menos tipos de dados. O BSON também codifica informações sobre o tipo e o comprimento, tornando-o mais fácil para as máquinas analisarem. JSON e BSON são parentes próximos, como seus nomes quase idênticos sugerem, mas você não saberia disso ao olhá-los lado a lado. JSON, ou Notação de Objetos JavaScript, é o padrão amplamente popular para a troca de dados na web, no qual o BSON (Binary JSON) é baseado.

[D1]: <https://www.mongodb.com/json-and-bson#what-is-json->

[D2]: <https://www.mongodb.com/basics/bson>

[D3]: https://www.w3schools.com/whatis/whatis_json.asp

Comparação entre JSON BSON [D2]

	Tipo e Velocidade de Leitura	Uso	Espaço	Vantagens
JSON	Escrito em formato de texto, sendo mais rápido para ler mas mais lento para construir.	Usado para mandar dados pela network, principalmente através do uso de API's.	Em termos de espaço, o JSON ocupa menos espaço.	Ficheiros que podem ser lidos sem serem interpretados por máquinas, maior facilidade a ser mandado por networks.
BSON	Escrito em formato binário, sendo mais lento para ler mas mais rápido para construir e scanear.	Usado para guardar os dados de uma base de dados.	Em termos de espaço, o BSON ocupa mais espaço.	Acesso a mais tipos de dados, como bindata para dados binários, e devido ao seu formato, apesar de mais pesados que os JSON, são muito mais fáceis de scanear.

Conteúdo:

[D2] Como o BSON difere do JSON: Tipo e Velocidade: Arquivos JSON são escritos em formato de texto, sendo mais rápido para ler mas mais lento para construir, enquanto que arquivos BSON são Escritos em formato binário, sendo mais lento para ler mas mais rápido para construir e scanear. Espaço: Em termos de espaço, o JSON ocupa menos espaço, enquanto que, em termos de espaço, o BSON ocupa mais espaço. Codificar e Decodificar: Podemos enviar JSON através de APIs sem codificação e decodificação. Os arquivos BSON são codificados antes de serem armazenados e decodificados antes de serem exibidos. Analisar: JSON é um formato legível por humanos que não requer análise. O BSON precisa ser analisado, pois é gerado por máquinas e não é legível por humanos. Tipos de Dados: JSON tem um conjunto específico de tipos de dados - string, booleano, número para tipos de dados numéricos, array, objeto e nulo. Diferentemente do JSON, o BSON oferece tipos de dados adicionais, como bindata para dados binários e decimal128 para números. Uso: Usado para enviar dados pela rede (principalmente por meio de APIs). Bancos de dados usam BSON para armazenar dados. Vantagens do BSON As três características a seguir tornam o BSON vantajoso de se usar: Leve e Navegável: O BSON é leve: Isso torna possível armazenar uma grande

quantidade de dados no formato de arquivo BSON. Além disso, os arquivos BSON são facilmente armazenados e enviados pela rede, tornando-os ideais para armazenar e enviar dados. Eficiente: O BSON foi projetado para armazenar espaço de forma eficiente e permitir varreduras eficazes: Em um documento BSON, elementos grandes são prefixados com um campo de comprimento. Documentos em BSON ocupam mais espaço do que o JSON devido aos prefixos de comprimento e índices explícitos de arrays, mas graças a esses prefixos, é possível realizar varreduras e consultas muito mais rapidamente. Os prefixos facilitam a comparação e o cálculo direto nos dados, simplificando o consumo de código de aplicativos. Lida com Tipos de Dados Adicionais: Diferentemente do JSON, você pode encontrar tipos de dados como Bindata, Minkey, Maxkey, Dados Binários, ObjectID, Expressão Regular, JavaScript, Decimal128 e Data para data e hora em BSON. Esses tipos de dados são cruciais ao trabalhar com programas especializados. Por exemplo, ao lidar com dados de sistemas financeiros, você pode usar o Decimal128 do BSON para representações decimais de alta precisão de 128 bits. Muitos sistemas modernos usam aritmética de ponto flutuante baseada em binário para representar frações decimais exatas por meio de tipos de dados Float e double. Esses tipos de dados fornecem uma aproximação, mas não o valor preciso. Usar o BSON nesses casos oferece alta precisão.

[D2]: <https://www.mongodb.com/basics/bson>

Formato XML (Extensible Markup Language)

Extensible Markup Language (XML) é uma linguagem para intercâmbio de dados estruturado. Em vez de ser um formato de arquivo rígido, XML é uma linguagem para definir formatos definidos que grupos podem usar para trocar dados. Muitas pessoas, organizações e empresas usam XML para transferir informações sobre o produto, transações, inventário e outros dados de negócios. [A6]

Vantagens [A9] :

- XML usa linguagem humana, não de computador. XML é legível e compreensível, mesmo para iniciantes, e não é mais difícil de codificar do que HTML.
- XML é extensível. Crie as suas próprias tags ou use tags criadas por terceiros, que utilizem a linguagem natural do seu domínio, que tenham os atributos necessários e que façam sentido para si e os seus usuários.

```
<?xml version="1.0"?>
<aviso>
<para>Janice
data="01/04/2000"</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Amanha voce tem
prova de matematica</corpo>
</aviso>
Aviso.xml
```

ORADOR:

Uma das linguagens mais usadas é o XML (Extensible Markup Language), específica para intercâmbio de dados estruturados, deriva da linguagem SGML e é muito utilizada para compartilhamento fácil de informações por intermédio da internet, como dito antes, através da criação de documentos com dados organizados hierarquicamente para serem usados por diferentes sistemas informatizados. Contudo, também é possível criarmos a nossa própria linguagem de marcação que inclui um conjunto de regras e tags que descrevem a informação adequada às suas necessidades. As principais vantagens de XML é o facto de ser legível e compreensível, não é mais difícil de programar que HTML, é compatível com java, qualquer aplicação que processe XML pode usar as suas informações, independentemente da plataforma.

Do meu lado esquerdo, podemos observar um exemplo de código em XML.

CONTEÚDO:

XML (*Extensible Markup Language*) é um tipo de linguagem de marcação da W3C (World Wide Web), derivada da linguagem SGML, utilizada para compartilhamento fácil de informações por intermédio da internet, através da criação de documentos com dados

organizados hierarquicamente para ser usado por diferentes sistemas informatizados (portabilidade); ou seja, tem objetivo de garantir que documentos codificados de acordo com suas regras possam ser transportados de um ambiente de hardware e software para outro sem perda de informação, usando a potencialidade e flexibilidade da linguagem SGML de forma simplificada. [A7]

O XML fornece um padrão aos programas e, mais importante ainda, aos programadores. Esse padrão é um formato amplamente aceite para transmissão de dados entre sistemas diferentes. Desse modo, o XML tem mais em comum com o JSON do que com o HTML.

Com a XML, podemos fazer upgrade ou modificar convenientemente o design da nossa aplicação. Muitas tecnologias, especialmente as mais recentes, vêm com suporte XML integrado. Podendo ler e processar automaticamente arquivos de dados XML, para que possamos fazer alterações sem precisar reformatar toda a base de dados. [A34]

É possível usar XML para criar a sua própria linguagem de marcação que inclui um conjunto de regras e tags que descrevem informações adequadas às suas necessidades, por exemplo, nome, título, endereço e CEP. Define essa linguagem de marcação em uma definição de tipo de documento (DTD) ou em um arquivo de esquema XML que funciona como a maneira padrão de descrever as suas informações. Usar XML para compartilhar informações padronizadas significa que não precisa mais escrever programas para focar em software proprietário ou converter e traduzir diferentes formatos de dados. [A8]

As principais vantagens de XML são : XML usa linguagem humana, não de computador. XML é legível e compreensível, mesmo para iniciantes, e não é mais difícil de codificar do que HTML, XML é totalmente compatível com Java™ e 100% portátil. Qualquer aplicativo que possa processar XML pode usar as suas informações, independentemente da plataforma, XML é extensível. [A9]

Mesmo que os arquivos XML sejam muito densos em comparação com soluções modernas para a transmissão de dados, como o JSON, não faz mal saber como abri-los e editá-los. Felizmente, há muitas mais opções disponíveis. [A35]

https://help.claris.com/archive/help/16/fmp/pt/index.html#page/FMP_Help/xml-format.html

[A7]- <https://pt.wikipedia.org/wiki/XML>

[A8]- <https://www.ibm.com/docs/en/i/7.3?topic=toolkit-xml-introduction>

[A9]- <https://www.ibm.com/docs/en/i/7.3?topic=introduction-advantages-xml>

[A34]- <https://aws.amazon.com/pt/what-is/xml/>

[A35]-

<https://www.freecodecamp.org/portuguese/news/o-que-e-um-arquivo-xml-como-abrir-arquivos-xml-e-quais-sao-os-melhores-editores-de-xml/>

XQuery

XQuery (XML Query) é uma linguagem de programação funcional e de consulta que consulta e transforma coleções de dados estruturados e não estruturados, geralmente na forma de XML, texto e com extensões específicas do fornecedor para outros formatos de dados (JSON, binário, etc.). A linguagem é desenvolvida pelo grupo de trabalho XML Query do W3C. O trabalho é estreitamente coordenado com o desenvolvimento do XSLT pelo Grupo de Trabalho XSL; os dois grupos compartilham a responsabilidade pelo XPath, que é um subconjunto do XQuery. [A28]

Vantagens [A31] :

- Usando XQuery, dados hierárquicos e tabulares podem ser recuperados;
- XQuery pode ser usado para consultar estruturas gráficas e de árvore;
- XQuery pode ser usado diretamente para consultar páginas da web;
- XQuery pode ser usado para transformar documentos XML;

ORADOR:

As XQuerys é uma linguagem usada para consultar os tipos de dados XML , sejam estruturados ou semiestruturados . Está em constante desenvolvimento . Os documentos podem ser armazenados numa BD e depois consultados usando XQuerys . É muito usado nas BDs . É muito simples e fácil de aprender . Tem vindo a substituir linguagens de desenvolvimentos de aplicações Web . Substitui também programas em Java e C++ com apenas algumas linhas de código . É análogo ao SQL e é baseado no XPath (linguagem que usa uma sintaxe nao XML , que usa uma maneira flexível de endereçar diferentes partes de um documento XML. As XQuerys podem ser usadas para consultar estruturas gráficas de arvores , páginas Web e principalmente podem ser usadas para transformar documentos XML .

CONTEÚDO :

O Transact-SQL oferece suporte a um subconjunto da

linguagem XQuery usado para consultar o tipo de dados xml. Esta implementação do XQuery está alinhada com o rascunho de trabalho do XQuery de julho de 2004. A linguagem está em desenvolvimento pelo World Wide Web Consortium (W3C), com a participação de todos os principais fornecedores de bases de dados e também da Microsoft. Como as especificações do W3C podem sofrer revisões futuras antes de se tornarem uma recomendação do W3C, esta implementação pode ser diferente da recomendação final.

XQuery é uma linguagem que pode consultar dados XML estruturados ou semiestruturados. Com o suporte ao tipo de dados xml fornecido no Mecanismo de bases de Dados, os documentos podem ser armazenados numa base de dados e depois consultados usando XQuery.[A29]

XQuery é uma linguagem padronizada para combinar documentos, bases de dados, navegadores Web e tudo mais. É amplamente implementado. É poderoso e fácil de aprender. XQuery está substituindo linguagens de middleware proprietárias e linguagens de desenvolvimento de aplicações Web. XQuery está substituindo programas Java ou C++ complexos por algumas linhas de código. XQuery é mais simples de trabalhar e mais fácil de manter do que muitas outras alternativas.

É uma linguagem funcional (XQuery é uma linguagem para recuperar/consultar dados baseados em XML) análogo ao SQL (XQuery está para o XML assim como o SQL está para as bases de dados) baseado em XPath (XQuery usa expressões XPath para navegar por documentos XML) Universalmente aceito (XQuery é suportado por todas as principais bases de dados) Padrão W3C (XQuery é um padrão W3C). [A31]

XPath significa XML Path Language. Ele usa uma sintaxe não XML para fornecer uma maneira flexível de endereçar (apontar para) diferentes partes de um documento XML. [A32]

São muito usadas para pesquisar dados XML para objetos que estão em níveis desconhecidos da hierarquia, realizar transformações estruturais nos dados (por exemplo, você pode querer inverter uma hierarquia), retornar resultados que possuem tipos mistos e atualizar dados XML existentes. [A37]

[A28]- <https://en.wikipedia.org/wiki/XQuery>

[A29]-

<https://learn.microsoft.com/en-us/sql/xquery/xquery-language-reference-sql-server?view=sql-server-ver16>

[A30]- https://www.w3schools.com/xml/xquery_example.asp

[A31]- https://www.tutorialspoint.com/xquery/xquery_overview.htm

[A32]- <https://developer.mozilla.org/en-US/docs/Web/XPath>

[A37]- <https://www.ibm.com/docs/pt-br/db2/11.5?topic=data-introduction-xquery>

```

<?xml version="1.0" encoding="UTF-8"?>

<bookstore|

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J. K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>

```

XQuery (Exemplo)

A Partir do código XML inserido, neste exemplo selecionamos o documento que pretendemos nomeadamente "book.xml" , e dentro deste selecionamos a *bookstore* e de seguida todos os livros com preço menor que 30.

```
doc("books.xml")/bookstore/book[price<30]
```

Ao inserirmos esta XQuery conseguimos obter o seguinte resultado:

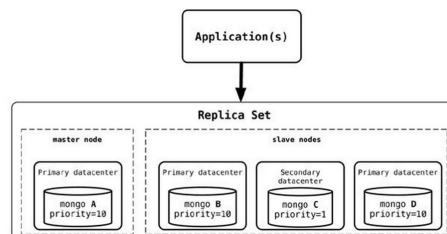
```

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J. K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

```

Escalabilidade em Document

Um sistema diz-se escalável quando este aumenta o seu desempenho proporcionalmente ao hardware nele contido. Esta característica pode ser avaliada de várias maneiras, entre elas pela sua carga de escalabilidade, que é a facilidade na expansão, pela sua escalabilidade geográfica, que é a sua capacidade de manter utilidade e desempenho com uma largura maior de dados, e pela sua escalabilidade administrativa, que é a sua facilidade em uso e administração, por mais complexo que sejam os seus dados. [D4] Para alcançar a escalabilidade, as bases de dados de documentos são construídas para escalar horizontalmente, isto é, são acrescentado novos nodos para o aumento exponencial dos seus sistemas. Com isto, bases de dados "NoSQL" conseguem evitar problemas provindos de outras maneiras de atingir escalabilidade, como o uso de joins, mantendo sempre o seu desempenho em alto nível. [D5] [D6]



Criação de um novo nodo em NoSQL

Conteúdo:

Em telecomunicações, infraestrutura de tecnologia da informação e na engenharia de software, escalabilidade é uma característica desejável em todo o sistema, rede ou processo, que indica a capacidade de manipular uma porção crescente de trabalho de forma uniforme, ou estar preparado para crescer. Por exemplo, isto pode-se referir à capacidade de um sistema suportar um aumento de carga total quando os recursos (normalmente do hardware) assim requisitam. Como característica de um sistema, a escalabilidade é normalmente difícil de se definir e, de forma particular, é necessário definir que requerimentos específicos de demanda deverão ser dimensionados para definir a importância da escalabilidade. A escalabilidade é um assunto extremamente importante em sistemas eletrônicos, bancos de dados, roteadores, redes de computadores, etc, e implica desempenho. Um sistema cujo desempenho aumenta com o acréscimo de hardware, proporcionalmente à capacidade acrescida, é chamado "sistema escalável". A escalabilidade pode ser medida de vários modos, tais como: Carga de escalabilidade – É a facilidade com que um sistema distribuído pode ser expandido usando todos os seus recursos para acomodar as demandas, sejam altas ou baixas. Escalabilidade geográfica - Um sistema é geograficamente escalável quando mantém sua utilidade e desempenho, independentemente dos pontos geográficos que compõem sua extensão ou como são usados os seus recursos. Escalabilidade administrativa - Não importa a variação de informação que diferentes organizações necessitem compartilhar em um único sistema distribuído,

este deve permanecer fácil de ser usado e gerenciado. Por exemplo, um sistema de processamento de transações on-line ou sistema de gerenciamento de banco de dados é escalável quando pode ser atualizado para aumentar o processamento de transações, mediante a adição de novos processadores, mecanismos e dispositivos de estocagem, que podem ser atualizados facilmente e de modo transparente, sem precisar de desligar o sistema. Escalonar horizontalmente (também chamado de scale out/in) significa adicionar mais nós a (ou remover nós de) um sistema, como adicionar um novo computador para uma aplicação de software distribuída. Um exemplo pode envolver escalonamento horizontal de um sistema de servidor Web para três. Como os preços de computadores caíram e o desempenho continua a crescer, aplicações de computação de alto desempenho, como análise sísmica e cargas de trabalho de biotecnologia adotaram sistemas "mercadoria" de baixo custo para tarefas que antes exigiam supercomputadores. Arquitetos de sistema podem configurar centenas de pequenos computadores em um cluster para obter poder de computação agregada que geralmente excede ao de computadores baseados em um único processador tradicional. O desenvolvimento de interconexões de alto desempenho como Ethernet Gigabit, InfiniBand e Myrinet alimentaram este modelo. Tal crescimento direcionou para demanda por software que possibilita o gerenciamento e manutenção eficientes de vários nós, bem como hardware como armazenamento de dados compartilhado com desempenho de E/S muito maior. Escalabilidade de tamanho é o número máximo de processadores que um sistema pode acomodar. Escalonar verticalmente (também chamado de scale up/down) significa adicionar recursos (ou remover recursos de) um único nó em um sistema, normalmente envolvendo a adição de UCPs ou memória para um único computador. Tal escalonamento vertical de sistemas existentes também permite-os usar tecnologia de virtualização de forma mais eficiente, uma vez que ele fornece mais recursos para o conjunto hospedado de sistema operacional e módulos de aplicação a compartilhar. Tomar vantagem de tais recursos também pode ser chamado de "aumento", como a expansão do número de processos daemon Apache executando no momento. Escalabilidade de aplicação é o desempenho melhorado de execução de aplicações em uma versão aumentada (scaled-up) do sistema.

É uma base de dados open-source,, do tipo "NOSQL" orientada a documentos. [E3] Usa o formato "BSON" que é uma variação do "JSON" com algumas características adicionais. [E4] Suporta documentos com mais de 16MB, para isso é necessário usar o "GridFS."

Principais vantagens e desvantagens em usar MongoDB[E5]:

- Compatibilidade com várias linguagens de programação
- Boa performance
 - Queries rápidas, com uso de indexes
 - Uso de joins e agregação, às vezes pode ser dispendioso
 - É possível usar "MapReduce"
- "Full text search", apenas disponível na versão Atlas
- Baixa redundância e grande disponibilidade dos dados
- Uso de "key shards" para resolver o problema de escalabilidade horizontal, várias partições
- É possível usar vários "engines" de armazenamento [E5]

ORADOR:

Esta base de dados "NOSQL" é das bases de dados mais usadas por empresas de topo como por exemplo "BOSCH","Wells Fargo","Vodafone","Verizon" entre outras empresas. Se estas empresas que são bastante respeitadas e usam a base de dados do tipo "MongoDB" deve ser por alguma razão, certo?

Compatibilidade com várias linguagens de programação. É suportado pela base de dados vários formatos de documentos que as linguagens de programação usam.

É permitido integrar informação que esteja relacionada num mesmo documento ou então em documentos separados.

Boa performance, ao usar indexes nas coleções.Sendo

collection o equivalente a uma tabela se estivessemos a trabalhar com "MySQL". Collection no "MongoDB" é um conjunto de documentos.

Imaginemos que a informação que queremos está apenas num index específico então realizamos uma query apenas a esse index em vez da coleção toda.

É possível criar,ler,atualizar informação e apagar através dos comandos:

db.collection.insertOne(); ou many

db.collection.find();

db.collection.updateOne(); ou many

db.collection.delete.One(); ou many

A baixa redundância e grande disponibilidade deve-se ao facto de que se a base de dados falhar, o "MongoDB" usa uma espécie de backup que guarda num nó secundário a informação necessária. Em caso de falha no nó principal recorre ao secundário.

O "MongoDB" usa "shard keys"" que significa chaves fragmentadas. As "shard keys" dividem as coleções em pedaços mais pequenos("chunks"), sendo mais fácil para balancear a informação por key shards, de forma a que todos os chunks tenham mais ao menos o mesmo tamanho.

Ao usarmos "sharding" resolvemos o problema de escalabilidade horizontal embora não resolva o de escalabilidade vertical, para melhorar a escalabilidade vertical

teríamos de adicionar um "CPU" mais poderoso ou mais "RAM".

CONTEÚDO:

O MongoDB é uma base de dados orientada a documentos tendo um formato parecido ao "JSON". Uma chave corresponde a um valor e os valores podem ser documentos, listas ou listas de documentos.[E5]

Para armazenar os documentos é armazenado numa coleção, analogamente para o modelo relacional seria uma tabela. É possível criar "views" ou seja escrevo o código da query e consigo guardar essa query numa "view".[E5][E14]

Existem pontos que destacam este tipo de base de dados que usa documentos. A performance é excelente devido ao suporte na integração de informação que está relacionada num mesmo documento[E7] e no uso de índices("index"), de vários tipos como um índice de valor único, composto, multivalorado, geoespacial (suporta coordenadas 2d), de texto(que suporta "strings") e de "hashed shards" o que facilita a realização de "queries".[E15]

A nosso ver o "MongoDB" suporta a maior parte das operações das típicas bases de dados "SQL" tal como o "MYSQL", sendo que apenas difere na linguagem, sendo o pensamento para as obter igual.[E16]

O sistema tem uma grande disponibilidade e baixa redundância pois existe um "daemon process", um processo

sempre presente em background que é capaz de copiar a informação do nó principal para os nós secundários. Em caso de falha do nó principal, os nós secundários é que irão tratar de devolver uma resposta.[E10]

Para resolver o problema da escalabilidade horizontal achamos que o "MongoDB" resolve de uma maneira lógica e eficiente ao dividir os dados em "chunks" e atribuir a cada "chunk" uma chave. Sendo que cada "chunk" tem um determinado tamanho, considerando um limite inferior e outro superior dependente da "shard key".[E11]

Em relação à escalabilidade vertical uma maneira eficiente de resolver esse aumento seria melhorar o "CPU" e a quantidade de RAM.[E11]

O "MongoDB" também suporta dois tipos de bases de dados como o "WiredTiger Storage Engine" e o "In-Memory Storage Engine".[E5]

[E3]<https://en.wikipedia.org/wiki/MongoDB>

[E4]<https://www.mongodb.com/json-and-bson>

[E5]<https://www.mongodb.com/docs/manual/introduction/>

[E6]<https://www.mongodb.com/who-uses-mongodb>

[E7]<https://www.mongodb.com/docs/manual/core/data-model-design/>

[E8]<https://www.mongodb.com/docs/manual/crud/#std-lib-abel-crud>

[E9]<https://www.mongodb.com/docs/manual/core/databases-and-collections/>

[E10]<https://www.mongodb.com/docs/manual/replication/>

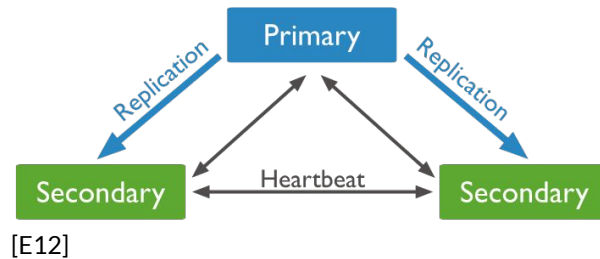
[E11]<https://www.mongodb.com/docs/manual/sharding/#std-label-sharding-introduction>

[E14]<https://www.mongodb.com/docs/manual/core/views/>

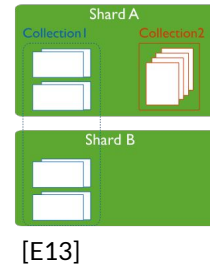
[E15]<https://www.mongodb.com/docs/manual/core/indexes/index-types/#std-label-index-types>

[E16]<https://kinsta.com/pt/blog/operadores-mongodb/>

Cópia entre as várias instâncias



Fragmentação da coleção (com e sem "key shards")

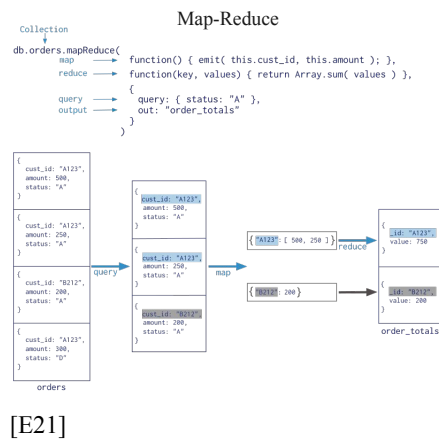


[E12]

<https://www.mongodb.com/docs/manual/images/replica-set-primary-with-two-secondaries.bakedsvg.svg>

[E13]

<https://www.mongodb.com/docs/manual/images/sharded-cluster-primary-shard.bakedsvg.svg>



[E21]

MongoDB, como criar uma base de dados e guardar ficheiros

Primeiros passos:

- Instalar o MongoDB
- Verificar se encontra-se na lista de autorizações da firewall
- Seguir os passos das figuras

Para sistemas Windows

Iniciar servidor MongoDB

```
net start MongoDB
```

A palavra use cria a base de dados.

No caso dessa base de dados existir, essa será selecionada

```
use Empresa
```

Para executar o servidor

```
mongo
```

Para inserir o que chamaria-se de tabelas no MySQL

```
db.Funcionario.insert(  
  {  
    "Funcionario_nome" : "Chris",  
    "Funcionario_departamento" : "Sales"  
  }  
)
```

[E40] <https://kinsta.com/pt/blog/banco-de-dados-mongodb/>

CouchBase Server



É uma base de dados open-source anteriormente conhecida como "Membase" desenvolvida pela "CouchBase, Inc". [E22]

Existe várias versões como CouchBase Capella, CouchBase Server e CouchBase Mobile. [E23]

Os seus documentos são guardados no formato "JSON" mas outros tipos também são permitidos. [E22]

Não é necessário fazer um esquema como nos modelos relacionais. [E22]

A linguagem para realizar queries é o "N1QL" é SQL92 compatível. É muita das vezes referido como SQL++. [E24]

Porquê usar CouchBase?

Orador:

Couchbase Server, é um software de banco de dados NoSQL multi modelo orientado a documentos. Foi projetado para funcionar como uma base de dados chave-valor ou documentos JSON, sendo permitido usar outros formatos. Tem uma fácil escalabilidade, com baixa latência e alto rendimento não comprometendo a latência.

Pode ser implementada para funcionar como uma base de dados de um único nó ou com vários nodos.

Não é necessário criar um esquema como nos modelos relacionais.

Para realizar pesquisas é muito simples, basta usar SQL.

Na nossa opinião usar CouchBase é uma mais valia pois permite realizar queries de uma maneira já conhecida, usando

SQL.A performance e a escalabilidade acompanham bem o aumento dos dados.Os desenvolvedores poderão escolher entre consistência e falha à partição ou disponibilidade e falha à partição. Por este conjunto de razões é uma base de dados a considerar.

Conteúdo:

Couchbase Server, originalmente conhecido como Membase, é um software de banco de dados NoSQL multi modelo orientado a documentos, otimizado para aplicações interativas. Estas aplicações podem servir muitos usuários simultâneos criando, armazenando, recuperando, agregando, manipulando e apresentando dados. Para dar suporte a esses tipos de necessidades de aplicações, o Couchbase Server foi projetado para fornecer acesso a valores-chave ou documentos JSON de fácil escalabilidade, com baixa latência e alto rendimento sem que a latência seja comprometida.Foi projetado para ser uma base de dados agrupado desde um único máquina/nó ou até implantações em larga escala abrangendo muitas máquinas/nós.[E22]

Um documento é a unidade mais básica de dados no Couchbase Server. Os documentos são armazenados no formato de documento "JSON" sem necessidade de um esquema como nos modelos relacionais. Documentos que não sejam JSON também podem ser armazenados no Couchbase Server, como por exemplo se estiverem no formato (binário,

XML, etc.).[E22]

É possível efetuar queries com SQL++ também conhecido como "N1QL". Compatível com o SQL92.[E24]

Existe várias versões tal como CouchBase Capella,CouchBase Server e CouchBase Mobile.[E23]

Fornece métodos de consistência à posterior e consistência imediata para garantir consistência num sistema.Não existe integridade referencial,portanto, não há chaves estrangeiras.Suporta vários métodos entre eles o Map-Reduce. No Couchbase, o particionamento pode ser feito por Sharding, o que leva a uma maior escalabilidade horizontal.Suporta os métodos de replicação "Master-Slave Replication" e "Master-Master Replication".

Por fim, possui funções em JavaScript para scripts por parte do servidor.

[E25]

Cada nó no Couchbase consiste em um serviço de dados, serviço de índice, serviço de consulta e componente gerenciador de cluster.Os três serviços, atualmente, podem ser distribuídos para execução em nós separados,se necessário. Segundo o teorema "CAP" de "Eric Brewer", o Couchbase é normalmente um sistema do tipo CP, o que significa que fornece consistência e tolerância à partição, ou pode ser configurado como um sistema AP com vários nós ligados.[E22]

Usar "CouchBase" é uma escolha consciente.É uma base

de dados poderosa com a grande vantagem de suas queries serem numa linguagem que muitos desenvolvedores conhecem “SQL” mas com algumas melhorias “SQL++”.

Podendo o desenvolvedor optar por um sistema CP ou AP.

[E22] https://en.wikipedia.org/wiki/Couchbase_Server

[E23] <https://www.couchbase.com/pricing/>

[E24]

https://docs.couchbase.com/server/6.6/analytics/3_query.html

[E25]

<https://www.geeksforgeeks.org/introduction-to-couchbase/>

CouchBase Server - Vantagens



- É rápido;
- Lida bem com escalabilidade;
- Realiza sharding automático;
- O administrador pode escolher onde quer manter a informação;
- Ao existir uma falha não compromete o sistema "NO SPOF";
- Ao realizar queries, muitas das vezes precisamos de realizar várias operações aí entra o "CBO" que é basicamente um otimizador para saber que operação é a mais eficiente; [E27]
- Analisa informação que muda ao longo do tempo. É útil por exemplo para saber as temperaturas de um país ao longo do tempo; [E28]
- Partilha de informação entre nodos de data-centers. Aumentando assim a consistência;
- Boa latência;
- Suporta Map-Reduce;
- Uso de scripts com Javascript;

Orador:

As vantagens do CouchBase são as seguintes: É rápido, lida bem com escalabilidade, realiza sharding automático, ao existir uma falha no sistema o sistema como um todo não é comprometido. Podemos guardar na base de dados informação que muda ao longo do tempo. É permitido a partilha de informação entre os data-centers. Normalmente o CouchBase oferece uma boa latência. Para desenvolvedores que queiram customizar ao máximo a sua base de dados podem usar javascript.

Conteúdo:

O Couchbase Server armazena dados como itens. Cada item consiste numa chave, pela qual o item é referenciado, e um valor associado, que deve ser binário ou um documento JSON.

As operações podem ser configuradas para serem

executados em vários nós do cluster, permitindo que as cargas de trabalho de alta prioridade sejam distribuídas e dimensionadas adequadamente.

Clusters e Disponibilidade: À medida que nós sucessivos são inicializados, cada um pode ser configurado para se juntar ao cluster existente. Entre os nós de cada cluster, os dados da base de dados são distribuídos de forma equitativa e replicados: os nós podem ser removidos e as falhas de nós tratadas sem perda de dados. [E29]

Buckets: Informação dividida em vários buckets usando uma função hash. Similar a partições [E26]

Cross Data Center Replication (XDCC) replica dados entre um bucket de origem e um bucket de destino. Os buckets podem estar localizados em diferentes clusters e em diferentes data centers: isso fornece proteção contra falhas no data center e também fornece acesso a dados de alto desempenho para aplicativos de missão crítica distribuídos globalmente. [E30]

Segurança: altamente seguro, a fim de preservar a privacidade e integridade dos dados e controlar tentativas de acesso. As instalações de segurança fornecidas cobrem áreas, incluindo autenticação, autorização e auditoria.

Dados temporais, são quaisquer dados que mudam ao longo do tempo. Geralmente é coletado com frequência, em intervalos regulares ou irregulares, de um dispositivo ou processo. [E28]

Os dados podem ser mantidos apenas na memória ou na memória e no armazenamento. [E29]

Ao gerar um plano para efetuar uma query, o otimizador "CBO" calculado os custos pode fazer o seguinte: Seleção de índice: o otimizador pode selecionar o índice ideal para a consulta. Método de junção: o otimizador pode escolher entre junções de loop aninhado, junções de hash e junções de transmissão de hash. Para junções hash, o otimizador pode

escolher qual lado da junção deve ser o lado da construção ou o lado da investigação. Enumeração de junção: o otimizador pode considerar diferentes ordens de junção e reescrever a consulta para usar a ordem de junção ideal.[E27]

[E26]<https://cloudxlab.com/blog/bucketing-clustered-by-and-cluster-by/>

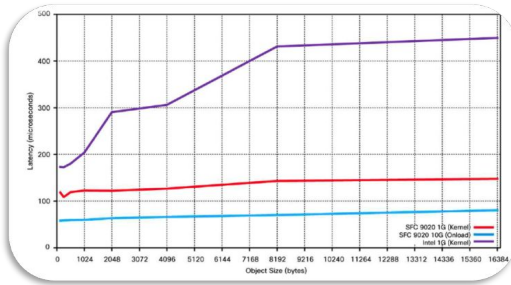
[E27]https://docs.couchbase.com/server/current/analytics/5b_cbo.html

[E28]<https://docs.couchbase.com/server/current/n1ql/n1ql-language-reference/time-series.html>

[E29]<https://docs.couchbase.com/server/current/learn/architecture-overview.html>

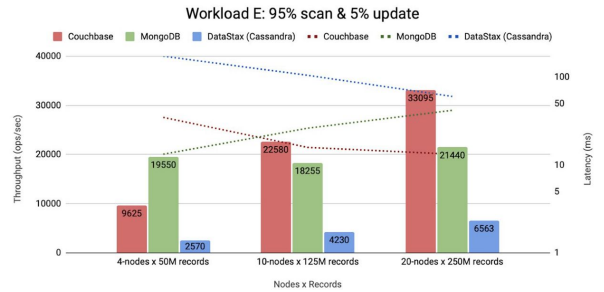
[E30]<https://docs.couchbase.com/server/current/learn/clusters-and-availability/xdcr-overview.html>

Latência em função do tamanho do documento-Teste realizado pela Cisco Systems



[E31]

Operações por segundo em função do número de nodos e documentos



ALTOROS

[E31]

<https://www.couchbase.com/blog/wp-content/uploads/2014/12/Benchmark-blog-Chart1.png>

[E35]

<https://www.altoros.com/blog/wp-content/uploads/2021/03/NoSQL-Couchbase-Server-MongoDB-DataStax-Enterprise-Short-range-Scan-Workload-E.png>

CouchBase, como criar a base de dados e guardar ficheiros



Primeiros passos:

- Fazer download do CouchBase e executar o ficheiro .msi no caso de tratar-se de um sistema Windows
- Seguir os passos das figuras

Configuração:

Acéder à console de gestão, através
<http://localhost:8091/>

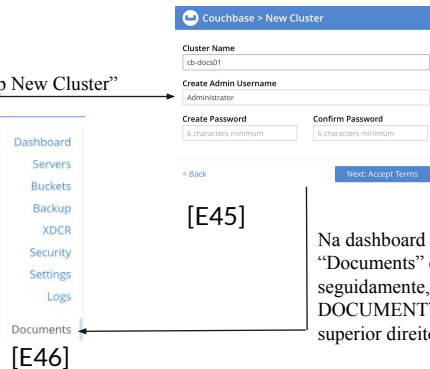
Escolha a opção "Setup New Cluster"

Depois de adicionar o item deverá aparecer
desta forma



```
{
  "id": "10",
  "type": "airline",
  "name": "40 Mile Air",
  "data": "Q5",
  "icao": "MLA",
  "callsign": "MILE-AIR",
  "country": "United States"
}
```

[E46]



Couchbase > New Cluster

Cluster Name
cb-docs01

Create Admin Username
Administrator

Create Password
6 characters minimum

Confirm Password
6 characters minimum

< Back

Next: Accept Terms

Dashboard
Servers
Buckets
Backup
XDCR
Security
Settings
Logs
Documents

[E45]

[E46]

Na dashboard selecione
"Documents" e
seguidamente, "ADD
DOCUMENT" no canto
superior direito

[E39] <https://docs.couchbase.com/server/current/manage/manage-ui/manage-ui.html>

[E45]

https://docs.couchbase.com/server/current/manage/_images/manage-nodes/setupNewCluster01.png

[E46]

https://docs.couchbase.com/server/current/manage/_images/manage-ui/documentsScreenWithDocuments.png

MongoDB vs CouchBase

Instalação e configuração

MongoDB: Muita das vezes a instalação pode ser demorada. Mais fácil de configurar do que outras bases de dados NOSQL. [E34]

Pode ser usado GUIs, para ser mais fácil gerir a base de dados e realizar queries. [E34]

CouchBase: Configurado tudo no mesmo sítio. Consola de gestão integrada. [E32]

Se for usado kubernetes, poderá ser usado o Couchbase Autonomous Operator que facilitará a criação de nodos e a sua junção ao cluster. [E33]

Linguagens suportadas

MongoDB: O MongoDB suporta uma ampla gama de linguagens de programação. Algumas delas incluem: C, C++, C#, Java, JavaScript, .NET, Erlang, Go,, PHP e Python, entre outras. [E32]

CouchBase: Menos linguagens de programação em comparação com o MongoDB, estas são: JavaScript, C, Go, .NET,, Java, PHP, Python, Scala e Ruby, entre outras. [E32]

Orador:

A maneira mais correta para um desenvolvedor escolher uma base de dados é saber que características gostaria de ter na sua de base de dados, seja consistência, disponibilidade, como o sistema resolve falhas e latência.

Nesta comparação entre MongoDB e CouchBase vamos analisar como é a instalação e as linguagens suportadas pelas duas bases de dados. No MongoDB é possível usar GUIs externas contudo o CouchBase já tem uma consola de gestão integrada.

O MongoDB suporta mais linguagens que o CouchBase o que permite que mais desenvolvedores usem o MongoDB em vez do CouchBase.

Conteúdo:

O Couchbase configura tudo num só lugar e oferece uma consola de administração integrada.

O MongoDB utiliza o paradigma "Master-Slave" e muitas das funcionalidades precisam ser configuradas

manualmente.[E32]

Existem várias GUIs (Interfaces Gráficas de Utilizador) para trabalhar com o servidor MongoDB, tais como o MongoDB Compass, o Studio 3T e outros. Estas ferramentas oferecem uma interface gráfica que lhe permite trabalhar facilmente com a sua base de dados e efetuar queries, em vez de utilizar uma linha de comandos e escrever queries manualmente.

O objetivo do "Couchbase Autonomous Operator" é garantir que não existem preocupações com as complexidades operacionais de executar o Couchbase. Além de administrar automaticamente o cluster do Couchbase, o Operador Autónomo do Couchbase também pode fazer a gestão do cluster de acordo com as melhores práticas do Couchbase.[E33]

O Operador Couchbase está desenhado para monitorizar constantemente o cluster Couchbase em busca de falhas. Quando é detectada uma falha num nó ou grupo de servidores, o "Autonomous Operator" está desenhado para criar automaticamente uma nova instância, preferencialmente na própria máquina ou, caso necessário, noutra máquina. Em seguida, reequilibra as instâncias com problemas, adiciona a nova instância e restaura o cluster. Se um cluster estiver configurado com volumes persistentes, o "Autonomous Operator" realiza o seguinte durante um evento de recuperação automática: Cria uma nova instância e associa-a ao mesmo volume persistente. Realiza operações complexas do Couchbase, como operações de aquecimento (mantém os dados que são mais acessados em cache, de maneira a que seja mais rápido o seu acesso), o que reduz o rebalanceamento de dados de todas as outras instâncias (uma operação demorada dependendo do tamanho dos dados). Remove a instância defeituosa do cluster e substitui-a por uma nova instância, garantindo que o cluster volte à configuração desejada sem perda de dados.

O Couchbase suporta menos linguagens de programação em comparação com o MongoDB. Algumas destas são: JavaScript, C, Go, .NET, ColdFusion, Java, PHP, Clojure, Erlang, Perl, Python, Scala e Ruby.[E32]

O MongoDB suporta uma ampla gama de linguagens de

programação. Algumas delas incluem: C, C++, Clojure, ColdFusion, D, Actionscript, C#, Dart, Delphi, Java, JavaScript, .NET, Erlang, Go, Groovy, Matlab, PHP, R e Python.[E32]

[E32] <https://intellipaat.com/blog/couchbase-vs-mongodb/>

[E33]

<https://cloud.redhat.com/blog/getting-started-with-the-couchbase-autonomous-operator-in-red-hat-openshift-3-11>

[E34]

<https://www.freecodecamp.org/news/learn-mongodb-a4ce205e7739/>

MongoDB vs CouchBase

Queries

MongoDB: Linguagem própria. Joins e agregações dispendiosas. [E36]

CouchBase: N1QL, suporta SQL. Util para quem estava acostumado a usar SQL. [E24]

Escalabilidade

MongoDB: Sharding Manual. [E11]

CouchBase: Sharding Automático pelos clusters. [E38]

Memória e latência

MongoDB: Tem de usar um engine subjacente de memória o que pode levar ao aumento da latência. [E39]

CouchBase: Usa uma "caching layer". Tornando o acesso aos dados mais rápido. [E37]

Orador:

O MongoDB tem uma linguagem própria, contudo o CouchBase permite que um desenvolvedor que saiba SQL tenha uma maior facilidade em realizar queries do que alguém a começar a realizar queries com o MongoDB.

No MongoDB é realizado sharding manual e no CouchBase é um sharding automático, cabendo a cada desenvolvedor o que melhor aplica-se à sua base de dados.

Em termos de Memória e latência, normalmente o CouchBase é superior pois guarda em memória cache os dados mais usados para fazer pesquisas, sendo o acesso a esta memória mais rápido e mais dispendioso para quem tem o servidor.

Conteúdo:

"Engine" de Armazenamento em Memória: Este engine de armazenamento funciona completamente na RAM, fornecendo o acesso mais rápido possível aos dados. No entanto, está limitado pela quantidade de memória livre disponível e todos os

dados são perdidos quando o servidor é desligado. O engine de armazenamento em memória, como o nome sugere, armazena dados na memória para obter um desempenho mais rápido e latências mais baixas. Este engine é útil para aplicações que exigem acesso rápido a registros individuais. No entanto, devido à sua dependência do subsistema de memória virtual, não é adequado para conjuntos de dados maiores.[E39]

O “engine” de Armazenamento WiredTiger é um engine de armazenamento poderoso para o MongoDB que substituiu o motor de armazenamento “MMAPv1” como a opção padrão para o banco de dados a partir da versão 3.2. Baseado em B-tree e conhecido pelo seu desempenho extremamente rápido, o WiredTiger oferece suporte ativo para compressão, encriptação e múltiplos controles de concorrência para garantir operações simplificadas e eficientes para todos os utilizadores, lidando facilmente com cargas de trabalho transacionais.[E39]

- Vantagens do Motor de Armazenamento WiredTiger:[E39]
 - gestão de conjuntos de dados maiores, aproveitando múltiplos núcleos da CPU
 - algoritmos de compressão melhorados, tornam-no uma escolha popular para empresas que exigem um bom desempenho
 - quando é detectado conflitos entre duas operações, uma delas irá incorrer num conflito de escrita, o que levará o MongoDB a repetir essa operação de escrita de forma transparente.

O WiredTiger é o motor de armazenamento recomendado nas versões mais recentes do MongoDB devido aos seus benefícios em termos de escalabilidade e desempenho. Apresenta bloqueio ao nível do documento, compressão por prefixo e opções de configuração ao nível da coleção. Tem um melhor suporte para armazenamento em conjuntos de dados maiores, graças à sua utilização de uma estrutura de dados bem ordenada.[E39]

O Couchbase Server disponibiliza uma camada de cache totalmente integrada que oferece um acesso rápido aos dados. O Couchbase Server gere automaticamente a camada de

cache, assegurando que há memória suficiente em relação ao espaço ocupado em disco para manter um bom desempenho.

O Couchbase utiliza "auto sharding" baseada em chaves para distribuir os dados de forma equitativa num cluster, e também oferece reequilíbrio automático e replicação automática. Estas automatizações podem simplificar processos críticos e libertar tempo valioso para a equipa de desenvolvimento.

Os "joins" no MongoDB são realizadas por meio do "Lookup". Realiza um "Left Outer Join" entre duas ou mais coleções. No entanto, o "Lookup" só é permitido em operações de agregação. Isso é semelhante a um "pipeline" que executa operações de consulta, filtro e agrupamento. O "Lookup" do MongoDB é muito útil e poderoso, mas requer consultas de agregação complexas.[E36]

Usar "CouchBase" é uma escolha consciente. É uma base de dados poderosa com a grande vantagem de suas queries serem numa linguagem que muitos desenvolvedores conhecem "SQL" mas com algumas melhorias "SQL++".[E24]

[E36]<https://hevodata.com/learn/mongodb-joins/>

[E24]

https://docs.couchbase.com/server/6.6/analytics/3_query.html

[E11]

<https://www.mongodb.com/docs/manual/sharding/#std-label-sharding-introduction>

[E39]<https://scalegrid.io/blog/mongodb-storage-engines/>

[E37]<https://docs.couchbase.com/server/current/learn/buckets-memory-and-storage/memory-and-storage.html>

[E38]

<https://www.couchbase.com/resources/concepts/what-is-database-sharding/>

É uma base de dados “open-source”, desenvolvida em “Java” que suporta documentos, grafos, chave-valor e objetos. [E41]

Para utilizar a base de dados é necessário usar o Java JDK. [E42]

- O utilizador poderá escolher entre não usar um esquema ou usá-lo. [E43]
- Os dados, nomeadamente os documentos são guardados na base de dados ligados através de apontadores, característica esta que pertence aos grafos. [E44]

Orador:

O OrientDB é uma base de dados open-source programada em Java. Para usá-la é necessário usar o Java JDK. É uma escolha para quem gostaria de guardar os dados em documentos, objetos ou como chave-valor, mas, maioritariamente de quem gosta de guardar os dados como grafos, pois o OrientDB para guardar a informação seja de que tipo for usa uma característica dos grafos de usar pointers entre eles. Como outras bases de dados NoSQL não é preciso criar esquema como nos modelos relacionais.

Conteúdo:

O OrientDB é o primeiro sistema de gestão de base de dados

NoSQL de código aberto multi modelo que combina o poder dos gráficos e a flexibilidade dos documentos numa única base de dados operacional escalável e de alto desempenho.[E42]

Embora as bases de dados de grafos tenham crescido em popularidade, a maioria dos produtos NoSQL ainda é usada para proporcionar escalabilidade a aplicações que se apoiam num sistema de gestão de base de dados relacional. O OrientDB, é o futuro: proporciona mais funcionalidade e flexibilidade, ao mesmo tempo que são suficientemente poderosos para substituir o seu sistema de gestão de base de dados.[E42]

Não existe junções: as relações são ligações através de pointers aos registos, permitindo uma velocidade de $O(1)$, ou seja a velocidade de pesquisa é elevado mesmo com um tamanho significativo de dados, suportando "full text search".[E44]

Os dados que são guardados dentro de documentos, são do tipo conjunto de pares chave/valor, onde a chave permite aceder ao seu valor. Os valores podem conter tipos de dados primitivos, documentos incorporados ou matrizes de outros valores. Normalmente, os documentos não são forçados a ter um esquema, o que pode ser vantajoso, pois permanecem flexíveis e fáceis de modificar. Os documentos são armazenados em coleções, permitindo aos programadores agrupar os dados como decidirem. O OrientDB utiliza os

conceitos de "classes" e "clusters" como a sua forma de "coleções" para agrupar documentos.

Com o OrientDB, pode decidir se deve incorporar documentos ou ligar-se a eles diretamente, por serem grafos. Quando recupera um documento, todas as ligações são automaticamente resolvidas pelo OrientDB. Isto é uma diferença importante em relação a outras bases de dados de documentos, como o MongoDB ou o CouchDB, onde o programador deve lidar com todas as relações entre os documentos por conta própria.[E47]

Estas características tornam o "OrientDB" uma boa escolha para aqueles que procuram uma solução versátil e de boa performance, nomeadamente nas queries.

[E41] <https://orientdb.org/docs/3.0.x/misc/Overview.html>

[E42] <https://db-engines.com/en/system/OrientDB>

[E43] <https://orientdb.org/docs/3.0.x/general/Schema.html>

[E44] <https://en.wikipedia.org/wiki/OrientDB>

[E47]

<https://orientdb.org/docs/3.0.x/datamodeling/Tutorial-Documents-and-graph-model.html>

OrientDB vs MongoDB

Indexes

MongoDB: Suporta sete tipos de indexação, sendo elas, campo único, composto, várias chaves, geoespacial, texto, hash e clustered. [E15]

OrientDB: Suporta quatro tipos de indexação sendo elas, uma árvore estritamente binária, Hash, sharding automatico, Lucene busca por palavras e Lucene Espacial.

Linguagens suportadas

MongoDB: Suporta uma ampla gama de linguagens de programação. Algumas delas incluem: C, C++, C#, Java, JavaScript, .NET, Erlang, Go,, PHP e Python, entre outras. [E32]

OrientDB: Suporta menos linguagens que o MongoDB, sendo estas: .Net, C, C#, C++, Clojure, Java, JavaScript (Node.js), PHP, Python, Ruby, Scala [E42]

Orador:

Aqui é possível ver uma comparação entre OrientDB e MongoDB. Foi comparado de que maneira as duas bases de dados usam indexes para guardar os dados. Tendo diversas formas de os guardar como por exemplo o MongoDB permite indexar de sete maneiras diferentes seja como campo único, composto, usando várias chaves, geoespacial, em texto, usando uma função de hash ou clustered. O OrientDB usa diferentes formas sendo estas, uma árvore estritamente binária, usar uma função de hash, realizar sharding automatico, fazer indexação por palavras ou por coordenadas espaciais.

Ambas as linguagens suportam várias linguagens sendo que o MongoDB suporta mais linguagens que o OrientDB, cabendo a cada desenvolvedor escolher a linguagem que mais se adequa aos seus conhecimentos.

Conteúdo:

O MongoDB para armazenar e organizar os dados recorre a vários tipos de indexação, sendo estes: Índices de campo único que recolhem e ordenam dados de um único campo em cada documento de uma coleção. Índices compostos recolhem

e ordenam dados de dois ou mais campos em cada documento de uma coleção. Os dados são agrupados pelo primeiro campo no índice e, em seguida, por cada campo subsequente. Índices multikey recolhem e ordenam dados armazenados em arrays. Os índices geoespaciais melhoram o desempenho de consultas em dados de coordenadas geoespaciais, suportando coordenadas planas ou esféricas. Os índices de texto suportam consultas de pesquisa de texto em campos que contêm conteúdo de string. Os índices hashed usam uma função de hash para atribuir índices. Os índices clusterizados especificam a ordem em que as coleções clusterizadas armazenam dados. Coleções criadas com um índice clusterizado são chamadas de coleções clusterizadas.

O OrientDB, tal como o MongoDB recorre a vários tipos de indexação, tal como:

Realização de uma árvore estritamente binária que fornece uma boa combinação de funcionalidades disponíveis em outros tipos de índice, é adequado para uso geral, o índice de hash fornece uma pesquisa rápida e consome muito pouco espaço em disco. Funciona como um hashmap, o que o torna mais rápido em pesquisas pontuais e consome menos recursos do que outros tipos de índice. O sharding automático, o Lucene busca por palavras fornece bons índices de texto completo, mas não pode ser usado para indexar outros tipos de índices. O índice espacial do Lucene fornece bons índices espaciais, mas não pode ser usado tal como o Lucene que busca por palavras para indexação de outros tipos.

O MongoDB destaca-se por sua ampla gama de linguagens suportadas, tornando-o acessível a uma variedade de desenvolvedores. Algumas das linguagens suportadas pelo MongoDB incluem C, C++, C#, Java, JavaScript, .NET, Erlang, Go, PHP e Python, entre outras [E32].

Por outro lado, o OrientDB suporta um conjunto mais restrito de linguagens em comparação com o MongoDB. As linguagens suportadas pelo OrientDB incluem .Net, C, C#, C++, Clojure, Java, JavaScript (Node.js), PHP, Python, Ruby e Scala [E42]. Embora o OrientDB ofereça suporte a linguagens comuns como

Java e PHP, ele tem uma cobertura mais limitada em comparação com o MongoDB.

[E32] <https://intellipaat.com/blog/couchbase-vs-mongodb/>

[E42]

<https://db-engines.com/en/system/MongoDB%3BOrientDB>

[E15]

<https://www.mongodb.com/docs/manual/core/indexes/index-types/#std-label-index-types>

Cosmos DB [F1]

O Azure Cosmos DB é um serviço de base de dados multi modelo globalmente distribuído, oferecido pela Microsoft, sendo uma base de dados NoSQL. O Cosmos DB armazena "itens" em "contentores", com esses dois conceitos sendo apresentados de forma diferente dependendo da API usada.

O Cosmos DB é compatível com diferentes API's.

Cada uma destas API's tem uma compatibilidade e mapeamento interno próprio, tornando o Cosmos DB mais flexível e compatível.



Notas de Orador:

O Azure Cosmos DB é um serviço de base de dados multimodelo distribuído pela Microsoft, sendo, tal como as anteriores, uma base de dados NoSQL. Este armazena os seus "itens" em "contentores", podendo estes dois conceitos ser apresentados de maneiras distintas dependendo da API usada. Este serviço foi imaginado para fornecer alta disponibilidade, escalabilidade e acesso de baixa latência aos dados para aplicativos de missão crítica. Sendo uma base de dados NoSQL, pode lidar com tipos de dados não estruturados e semiestruturados, ao contrário, das bases de dados relacionais tradicionais.

O Cosmos DB apesar de possuir uma API SQL proprietária, permite a compatibilidade com outras cinco diferentes API's compatíveis com o protocolo "wire", tal como o MongoDB, já antes falado, que é uma base de dados orientada a documentos. Estas API's de compatibilidade permite que qualquer aplicativo compatível se conecte e use o serviço.

Conteúdo:

O Azure Cosmos DB é um serviço de base de dados multimodelo globalmente distribuído oferecido pela Microsoft. Ele foi projetado para fornecer alta disponibilidade, escalabilidade e acesso de baixa latência aos dados para aplicativos de missão crítica. Ao contrário dos bases de dados relacionais tradicionais, o Cosmos DB é um base de dados NoSQL, o que significa que pode lidar com tipos de dados não estruturados e semiestruturados.

Internamente, o Cosmos DB armazena "itens" em "contentores", com esses dois conceitos sendo apresentados de forma diferente dependendo da API usada (esses seriam "documentos" em "coleções" ao usar a API compatível com MongoDB, por exemplo). Os contentores são agrupados em "bases de dados", que são análogos a namespaces acima dos contentores. Os contentores são independentes de esquema, o que significa que nenhum esquema é imposto ao adicionar itens. [F1] Os contentor também podem impor restrições de chave exclusiva para garantir a integridade dos dados. Um "Tempo de Vida" (ou TTL) pode ser especificado no nível do contentor para permitir que o Cosmos DB exclua automaticamente os itens após uma certa quantidade de tempo expressa em segundos. Essa contagem regressiva começa após a última atualização do item. Se necessário, o TTL também pode ser sobrecarregado no nível do item. [F1] O modelo de data interno, descrito anteriormente é exposto através de uma API SQL proprietária, cinco APIs de compatibilidade diferentes, expondo terminais que são parcialmente compatíveis com os protocolos "wire" de MongoDB, Gremlin, Cassandra, Armazenamento de tabelas do Azure e etcd. Essas APIs de compatibilidade possibilitam que qualquer aplicativo compatível se conecte e use o Cosmos DB por meio de drivers ou SDKs padrão, além de se beneficiar dos principais recursos do Cosmos DB, como particionamento e distribuição global. [F1] O Cosmos DB é compatível com diferentes API's, estas com diferentes objetivos, sendo a MongoDB, como já falada anteriormente, uma base de dados orientada aos documentos. Cada uma destas API's tem uma

compatibilidade e mapeamento interno próprio, tornando o Cosmos DB mais flexível e compatível.

[F1] : [Cosmos DB - Wikipedia](#)

[F7]: <https://nordcloud.com/blog/azure-cosmos-db/>



Benefícios principais do Cosmos DB [F2]

- Velocidade garantida em qualquer escala;
- Desenvolvimento de aplicação simplificado;
- Pronto para missões críticas;
- Totalmente gerenciado e econômico;
- Integração com o Azure Synapse Link.

Usos do Cosmos DB [F3]

- IoT (Internet of Things);
- Retalho e marketing;
- Jogos;
- Aplicações Web e móveis.

Notas de orador:

O CosmosDB apresenta alguns benefícios que o distinguem de outros serviços. Entre elas estão a sua velocidade garantida em qualquer escala, que permite o acesso em tempo real e com baixas latências de leitura e gravação, o desenvolvimento de aplicação simplificado, sendo este totalmente integrado com os principais serviços do Azure usados no desenvolvimento de aplicativos modernos.

O Cosmos DB está pronto para missões críticas, garantindo a continuidade dos negócios, disponibilidade quase garantida e segurança de nível empresarial para cada aplicativo. Este serviço é também totalmente gerenciado e econômico, tendo o seu serviço de bases de dados totalmente gerenciado com manutenção, atualizações e correções automáticas, o que permite economizar tempo e dinheiro aos desenvolvedores. O Cosmos DB possui ainda a integração com o Azure Synapse Link, que é uma capacidade de processamento transacional e analítico híbrido nativo de nuvem, permitindo análise em tempo real dos dados operacionais neste serviço. Isto causa a redução da complexidade das análises. Isto tudo permite ao

Cosmos DB ter um custo efetivo. O Cosmos DB é usado principalmente em IoT(internet of Things), retalho e marketing, jogos e aplicações Web e móveis, tais como, aplicações sociais e, de personalização. A própria Microsoft utiliza o seu serviço em diversas das suas aplicações como o Microsoft Office, Skype, Active Directory, Xbox e MSN.

Conteúdo:

Os benefícios principais do Cosmos DB são a sua velocidade garantida em qualquer escala, permitindo acesso em tempo real com baixas latências de leitura e gravação em todo o mundo, com a taxa de transferência e consistência respaldados por SLAs, gravações em várias regiões e distribuição de dados para qualquer região do Azure com apenas um botão, e ainda escalabilidade independente e elástica de armazenamento e taxa de transferência em qualquer região do Azure - mesmo durante picos de tráfego imprevisíveis - para escala ilimitada em todo o mundo. Este permite ainda o desenvolvimento de aplicação simplificado, totalmente integrado com os principais serviços do Azure usados no desenvolvimento de aplicativos modernos (nativos de nuvem), incluindo Azure Functions, IoT Hub, AKS (Azure Kubernetes Service), App Service e muito mais, permitindo, a escolha entre várias APIs de banco de dados, incluindo a API nativa para NoSQL, MongoDB, PostgreSQL, Apache Cassandra, Apache Gremlin e Table, a construção aplicativos na API para NoSQL usando as linguagens de sua escolha com SDKs para .NET, Java, Node.js e Python. Ou ainda a sua escolha de drivers para qualquer uma das outras APIs de banco de dados. O Change feed ainda facilita o acompanhamento e gerenciamento de alterações nos contentores do banco de dados e a criação de eventos acionados com o Azure Functions, para além do serviço sem esquema do Azure Cosmos DB que indexa automaticamente todos os seus dados, independentemente do modelo de dados, para fornecer consultas extremamente rápidas. O Cosmos DB está também pronto para missões críticas (Garantindo a continuidade dos negócios, disponibilidade de 99,999% e segurança de nível empresarial para cada aplicativo.). O Azure

Cosmos DB oferece um conjunto abrangente de SLAs, incluindo disponibilidade líder do setor em todo o mundo. Este permite ainda a distribuição fácil de dados para qualquer região do Azure com replicação automática de dados. Desfrutando de tempo de inatividade zero com gravações em várias regiões e de criptografia de nível empresarial com chaves auto-gerenciadas. O controle de acesso baseado em funções do Azure mantém os seus dados seguros e oferece controle refinado. Este é ainda totalmente gerenciado e econômico, tendo um serviço de base de dados totalmente gerenciado. Manutenção, atualizações e correções automáticas, economizando tempo e dinheiro dos desenvolvedores. Opções econômicas para cargas de trabalho imprevisíveis ou esporádicas de qualquer tamanho ou escala, permitindo que os desenvolvedores comecem facilmente sem precisar planejar ou gerenciar a capacidade. O modelo sem servidor oferece ainda um serviço automático e responsivo para gerenciar picos de tráfego sob demanda. E o provisionamento da taxa de transferência em escala automática dimensiona automaticamente a capacidade para cargas de trabalho imprevisíveis, mantendo os SLAs. Este produto permite ainda a integração com o Azure Synapse Link. O Azure Synapse Link para Azure Cosmos DB é uma capacidade de processamento transacional e analítico híbrido nativo de nuvem (HTAP) que permite análises quase em tempo real sobre dados operacionais no Azure Cosmos DB. O Azure Synapse Link cria uma integração perfeita e estreita entre o Azure Cosmos DB e o Azure Synapse Analytics. Causando a redução da complexidade das análises sem necessidade de trabalhos de ETL a serem gerenciados. Permite ainda visão quase em tempo real de seus dados operacionais. Não tem efeito sobre as cargas de trabalho operacionais. É ótimo para cargas de trabalho de análise em grande escala. Tem um custo efetivo. Permite análises para gravações disponíveis localmente, globalmente distribuídas, em várias regiões. Para além da integração nativa com o Azure Synapse Analytics.

[F2]. O Cosmos DB é usado maioritariamente em IoT (Internet of Things), retalho e marketing, jogos, e aplicações Web e

móveis, tais como, aplicações sociais e, de personalização[F3]. O cosmos DB é usado em muitas das aplicações da Microsoft como o Microsoft Office, Skype, Active Directory, Xbox e MSN [F1].

[F1] : [Cosmos DB - Wikipedia](#)

[F2]: [Introduction - Azure Cosmos DB | Microsoft Learn](#)

[F3]: [Common use cases and scenarios for Azure Cosmos DB | Microsoft Learn](#)

[F7]: <https://nordcloud.com/blog/azure-cosmos-db/>

Diferentes API's compatíveis com o Cosmos DB [F1]



API	Mapeamento Interno		Estado da compatibilidade e observações
	Contentores	Artigos	
MongoDB	Coleções	Documentos	Compatível com wire protocol versão 6 e server Versão 3.6 do MongoDB.
Gremlin	Gráficos	Nós e arestas	Compatível com a versão 3.2 da especificação Gremlin.
Cassandra	Quadro	Linha	Compatível com a versão 4 do Protocolo de ligação Cassandra Query Language (CQL).
Armazenamento de tabelas do Azure	Quadro	Artigo	
etcd	Chave	Valor	Compatível com a versão 3 do etcd.

[F1] : [Cosmos DB - Wikipedia](#)

[F7]: <https://nordcloud.com/blog/azure-cosmos-db/>

Exemplo de uso do Cosmos DB [F16]

```
C#  
using MongoDB.Driver;
```

```
C#  
  
// Create new object and upsert (create or replace) to container  
_products.InsertOne(new Product(  
    Guid.NewGuid().ToString(),  
    "gear-surf-surfboards",  
    "Yamba Surfboard",  
    12,  
    false  
));
```

```
C#  
  
// Read multiple items from container  
_products.InsertOne(new Product(  
    Guid.NewGuid().ToString(),  
    "gear-surf-surfboards",  
    "Sand Surfboard",  
    4,  
    false  
));  
  
var products = _products.AsQueryable().Where(p => p.Category == "gear-surf-surfboards");  
  
Console.WriteLine("Multiple products:");  
foreach (var prod in products)  
{  
    Console.WriteLine(prod.Name);  
}
```

[F16]: [Quickstart - Azure Cosmos DB for MongoDB for .NET with MongoDB driver | Microsoft Learn](#)

HCL Notes e HCL Domino [F4]



HCL Notes e o HCL Domino são, respectivamente, o cliente e o servidor de uma plataforma de software colaborativo cliente-servidor anteriormente possuído pela IBM e agora pela HCLTech.

O HCL Domino é um ambiente de tempo de execução de aplicações cliente-servidor multiplataforma. Este disponibiliza email, calendários, mensagens instantâneas, discussões/fóruns, blogs e um diretório pessoal.

Este é também dividido em diversas componentes:

- Aplicação cliente HCL Notes;
- Cliente HCL Verse;
- Servidor HCL Domino;
- Cliente de administração HCL Domino;
- Designer HCL Domino.

Notas de Orador:

O HCL Domino é um ambiente de tempo de execução de aplicações cliente-servidor multiplataforma. O Domino disponibiliza email, calendários, mensagens instantâneas, discussões/fóruns, blogs e um diretório de pessoal/ utilizador integrado. A base de dados no Domino pode ser replicada entre servidores e entre o servidor e o cliente, permitindo assim aos clientes funcionalidades offline. O Domino disponibiliza aplicações que podem ser usadas para aceder, armazenar e apresentar informações através de uma interface de utilizador, aplicar medidas de segurança, e permitir que vários servidores contenham a mesma informação e que muitos utilizadores trabalhem com esses dados.

Conteúdo:

O HCL Domino é um ambiente de tempo de execução de aplicações cliente-servidor multiplataforma. O Domino disponibiliza email, calendários, mensagens instantâneas (com

recursos adicionais de voz e vídeo conferência e colaboração na web pela HCL Software), discussões/fóruns, blogs e um diretório de pessoal/ utilizador integrado. Além destas aplicações padrão, uma organização pode utilizar o ambiente de desenvolvimento Domino Designer e outras ferramentas para criar aplicações integradas adicionais, como aprovação de pedidos, fluxos de trabalho e gestão de documentos. [F4] O produto Domino é composto por vários componentes, sendo estes a aplicação cliente HCL Notes (desde a versão 8, baseada no Eclipse), o Cliente HCL Notes, composto por, um cliente rico, um cliente web, o HCL iNotes, e um cliente de email móvel, HCL Notes Traveler, o Cliente HCL Verse, sendo este composto por, um cliente de email web, o Verse on Premises (VOP), e um cliente de email móvel, Verse Mobile (para iOS e Android), o Servidor HCL Domino, o Cliente de Administração HCL Domino, e também pelo Designer HCL Domino (ambiente de desenvolvimento integrado baseado no Eclipse) para criar aplicações cliente-servidor que funcionam dentro da estrutura do Notes) [F4]. O Domino é a plataforma de colaboração mais utilizada por empresas que pregam segurança como ponto estratégico. Houve épocas em que o Lotus Domino era proibido de sair dos Estados Unidos devido a sua criptografia. [F4] O Domino compete com produtos de outras empresas, como a Microsoft, o Google, o Zimbra e outros. Devido às suas capacidades de desenvolvimento de aplicações, o HCL Domino é frequentemente comparado a produtos como o Microsoft Sharepoint. A base de dados no Domino pode ser replicada entre servidores e entre o servidor e o cliente, permitindo assim aos clientes funcionalidades offline.[F4] O Domino, uma aplicação empresarial e servidor de mensagens, é compatível tanto com o Notes quanto com os navegadores da web. O Notes (e desde o IBM Domino 9, o HCAA) pode ser usado para aceder a qualquer aplicação Domino, como fóruns de discussão, bibliotecas de documentos e inúmeras outras aplicações. O Notes assemelha-se a um navegador da web, pois pode executar qualquer aplicação compatível para a qual o utilizador tenha permissão. O Domino disponibiliza aplicações que podem ser usadas para aceder, armazenar e apresentar informações através de uma interface de utilizador, aplicar

medidas de segurança, e replicar, ou seja, permitir que vários servidores contenham a mesma informação e que muitos utilizadores trabalhem com esses dados. [F4]

[F4]:https://en.wikipedia.org/wiki/HCL_Domino

[F6]:<https://www.teamstudio.com/hcl-products>

Mecanismo de armazenamento e programação[F4]



O mecanismo de armazenamento padrão no Domino é um formato de base de dados de documentos. O ficheiro normalmente irá conter tanto o design de uma aplicação como os respetivos dados associados. Enquanto o HCL Notes inclui um sistema de gestão de bases de dados onde os seus ficheiros são também centrados em documentos.

Os programadores podem desenvolver aplicações para o Domino em várias linguagens de programação, incluindo :

- a linguagem de programação Java, diretamente ou através de XPages;
- LotusScript, uma linguagem semelhante ao Visual Basic;
- a linguagem de programação JavaScript via o Domino AppDev Pack.

Notas de Orador:

O mecanismo de armazenamento padrão no Domino é o formato de base de dados de documentos, o "Notes Storage Facility" (.nsf). O ficheiro .nsf normalmente contém o design de uma aplicação e os dados associados. Enquanto o HCL Notes inclui um sistema de gestão de bases de dados onde os seus ficheiros são também centrados em documentos. Os programadores podem desenvolver aplicações para o Domino em várias linguagens de programação, incluindo a linguagem de programação Java, diretamente ou através de XPages, a LotusScript, e ainda a linguagem de programação JavaScript via o Domino AppDev Pack. O Notes é uma base de dados orientada para documentos (baseada em documentos, sem esquema, estruturada de forma flexível) com suporte para conteúdo rico e potentes funcionalidades de indexação.

O mecanismo de armazenamento padrão no Domino é o formato de base de dados de documentos, o "Notes Storage Facility" (.nsf). O ficheiro .nsf normalmente contém o design de uma aplicação e os dados associados. O Domino também pode aceder a bases de dados relacionais, seja através de um servidor adicional chamado HCL Enterprise Integrator for Domino, através de chamadas ODBC ou através do uso de XPages.[F4] Como o Domino é um ambiente de tempo de execução de aplicações, o email e os calendários funcionam como aplicações dentro do Notes, que a HCL fornece com o produto. Um desenvolvedor de aplicações Domino pode modificar ou substituir completamente essa aplicação. A HCL também disponibilizou os modelos base como open source.[F4] Os programadores podem desenvolver aplicações para o Domino em várias linguagens de programação, incluindo a linguagem de programação Java, diretamente ou através de XPages, a LotusScript, uma linguagem semelhante ao Visual Basic, a linguagem de programação JavaScript via o Domino AppDev Pack.[F4] As notas de documentos podem ter relações de pai-filho, mas o Notes não deve ser considerado uma base de dados hierárquica no sentido clássico dos sistemas de gestão de informação. As bases de dados do Notes também não são relacionais, embora exista um controlador SQL que pode ser usado com o Notes e que tem algumas funcionalidades que podem ser usadas para desenvolver aplicações que imitam funcionalidades relacionais. O Notes não suporta transações atômicas, e o seu bloqueio de ficheiros é rudimentar. O Notes é uma base de dados orientada para documentos (baseada em documentos, sem esquema, estruturada de forma flexível) com suporte para conteúdo rico e potentes funcionalidades de indexação. Esta estrutura imita de perto os fluxos de trabalho baseados em papel que o Notes é normalmente utilizado para automatizar.[F4] O Java foi integrado no IBM Notes a partir da versão 4.5. Com a versão 5,

o suporte para Java foi amplamente aprimorado e expandido, e o JavaScript foi adicionado. Embora o LotusScript continue a ser uma ferramenta principal no desenvolvimento de aplicações para o cliente Lotus Notes, o Java e o JavaScript são as principais ferramentas para o processamento baseado em servidor, o desenvolvimento de aplicações para acesso via navegador e permitem que os navegadores emulem a funcionalidade do cliente IBM Notes. Com as XPages, o cliente IBM Notes agora pode processar nativamente código Java e JavaScript, embora o desenvolvimento de aplicações normalmente exija algum código específico apenas para o IBM Notes ou apenas para um navegador.[F4]

[F4]: https://en.wikipedia.org/wiki/HCL_Domino

[F6]: <https://www.teamstudio.com/hcl-products>

Exemplo de uso do HCL Domino [F15]

```
Sub Initialize
  Dim session As New NotesSession
  Dim db As NotesDatabase
  Dim dc As NotesDocumentCollection
  Dim doc As NotesDocument
  Set db = session.CurrentDatabase
  Set dc = db.AllDocuments
  Set doc = dc.GetFirstDocument
  While Not(doc Is Nothing)
    category = doc.Category
    totalSales = doc.TotalSales
    Select Case totalSales(0)
      Case Is >= 200000 : category(0) = "Above Quota"
      Case Is >= 100000 : category(0) = "OK"
      Case Else : category(0) = "Below Quota"
    End Select
    doc.Category = category
    Call doc.Save(True, False)
    Set doc = dc.GetNextDocument(doc)
  Wend
End Sub
```

[F15]: [Examples: Agents \(hcltechsw.com\)](http://hcltechsw.com)

Caraterísticas principais do HCL Domino [F5]



Caraterísticas Principais	Vantagens
Libertar os trabalhadores do cliente Domino Notes.	Utilizar as aplicações do HCL Notes através do browser.
Possibilidade de utilizar as aplicações sem conexão à internet.	Parar com as interrupções de trabalho e aumentar a eficiência do mesmo.
Criar aplicações móveis mais rapidamente e aproveitar a capacidade específica de cada dispositivo.	Melhora a experiência do utilizador.
Modernizar a aparência e experiência do utilizador das aplicações.	Facilidade em atualizar aplicações já existentes.

Notas de orador:

Tal como os outros serviços, o HCL Domino apresenta algumas caraterísticas que o separa dos demais. Primeiramente, este permite libertar os trabalhadores de um cliente, podendo executar as suas aplicações num navegador padrão, sem a necessidade de instalar ou atualizar nenhum software. Este permite também a utilização de aplicações sem conexão à internet, acabando com as interrupções no trabalho e melhorando a sua eficiência quando não há conexão à internet. O serviço permite também criar aplicações móveis mais rápido, aproveitando a capacidade específica de cada dispositivo, adaptando a experiência de utilizador com base no dispositivo do mesmo, permitindo a construção de aplicações com a acesso a funcionalidades específicas como a câmara, ficheiros, etc. Este permite ainda modernizar a aparência e experiência de utilizador das aplicações já existentes, permitindo uma maior facilidade em atualizar aplicações já existentes. Este serviço é uma base de dados distribuída orientada a documentos NoSQL e uma estrutura de mensagens multiplataforma, bem como um ambiente de desenvolvimento

de aplicações rápidas que inclui aplicações pré-construídas como email e calendário, entre outras. Isto distingue-os dos seus principais concorrentes comerciais, como o Microsoft Exchange ou o Novell GroupWise, aplicações específicas para correio eletrónico e calendário, oferecendo APIs para extensibilidade.

Conteúdo:

As principais Características do HCL Domino são que este permite libertar os trabalhador do cliente Domino Notes, podendo executar as suas aplicações do Notes num navegador padrão, sem a necessidade de instalar ou atualizar qualquer software, possibilita a utilização das aplicações sem conexão à internet, eliminando as interrupções no trabalho e melhorando a sua eficiência ao trabalhar offline quando não há ligação à internet disponível. Este permite ainda criar aplicações móveis mais rapidamente e aproveitar a capacidade específica de cada dispositivo, adaptando a experiência do utilizador com base no dispositivo, permitindo que o cliente construa aplicações que acessem à câmara, ficheiros, localização e gestos móveis nativos. Para além do mais, permite modernizar a aparência e a experiência do utilizador para aplicações existentes, permitindo que se restilizem aplicações existentes em minutos, atualizando facilmente uma aplicação ou várias ao mesmo tempo. O HCL Domino ainda alivia o stress de gerir o HCL Notes e permite capacitar a sua força de trabalho, reduzindo significativamente os pedidos de assistência e a supervisão administrativa com o MarvelClient, que permite a gestão e monitorização com funcionalidades de autorreparação. Permite ainda customizar a experiência do utilizador, permitindo o interruptor de aplicações para poupar tempo e fornecer atalhos de menu para aplicações específicas [F5]. O Notes e o Domino são uma base de dados distribuída orientada a documentos NoSQL e uma estrutura de mensagens multiplataforma, bem como um ambiente de desenvolvimento de aplicações rápidas que inclui aplicações pré-construídas como email e calendário, entre outras. Isto distingue-os dos seus principais concorrentes

comerciais, como o Microsoft Exchange ou o Novell GroupWise, que são aplicações específicas para correio eletrónico e calendário, oferecendo APIs para extensibilidade.[F4] Uma característica fundamental do Notes é que múltiplas réplicas da mesma base de dados podem existir ao mesmo tempo em servidores e clientes diferentes, em plataformas diferentes; a mesma arquitetura de armazenamento é usada tanto para réplicas de clientes como de servidores. Inicialmente, a replicação no Notes ocorria ao nível do documento (ou seja, do registo). Com o lançamento do Notes 4 em 1996, a replicação foi alterada de forma a que agora ocorre ao nível do campo.[F4]

[F4]: https://en.wikipedia.org/wiki/HCL_Domino

[F5]: <https://www.hcl-software.com/domino>

[F6]: <https://www.teamstudio.com/hcl-products>

Clusterpoint



Clusterpoint é uma base de dados de documentos sem esquema; [F9]

O Clusterpoint suporta transações multi-documento compatível com ACID e um modelo computacional avançado baseado em JavaScript; [F10]

Este incorpora as funcionalidades de um motor de pesquisa de texto completo e oferece encriptação de dados em repouso; [F11]

Alguns especialistas vêem isto como uma oportunidade para utilizar uma única base de dados em vez de três. [F11]

Notas de orador:

Clusterpoint é uma base de dados de documentos sem esquema. Para ser considerado uma verdadeira substituição para bases de dados relacionais, o Clusterpoint suporta transações multi-documento compatíveis com ACID e um modelo computacional avançado baseado em JavaScript. O Clusterpoint oferece transações multi-documento compatíveis com ACID hoje, uma funcionalidade que, por exemplo, o MongoDB só tem nos seus planos futuros. Além disso, o Clusterpoint incorpora as funcionalidades de um motor de pesquisa de texto completo. Alguns especialistas, como Rasscevsckis, viam isto como uma oportunidade para utilizar uma única base de dados em vez de três (um motor de pesquisa, um SGBD relacional e uma base de dados de documentos). A API do Clusterpoint baseia-se no conceito de que uma base de dados deve permitir a realização de cálculos arbitrários nos dados diretamente na base de dados. Assim, permitimos que várias fases de recuperação usem código

JavaScript arbitrário para transformar os dados. Além disso, o código JavaScript acede a índices em vez de objetos brutos, proporcionando assim uma elevada eficiência.

Conteúdo:

Clusterpoint é uma base de dados de documentos sem esquema. Exemplos são quando os dados de um SGBD SQL são utilizados em combinação com um motor de pesquisa empresarial para abordar as necessidades de desempenho e escalabilidade de aplicações web e móveis, ou quando ferramentas de big data e análise, como o Hadoop, podem ser necessárias devido ao grande volume de dados ou às grandes cargas de trabalho de computação.[F9]

Para ser considerado uma verdadeira substituição para bases de dados relacionais, o Clusterpoint 4 suporta transações multi-documento compatíveis com ACID e um modelo computacional avançado baseado em JavaScript, duas funcionalidades frequentemente ignoradas por outras bases de dados NoSQL no mercado atual. Para muitos casos de uso, é importante trabalhar com vários objetos de dados dentro de uma única transação e garantir consistência imediata. Outro aspeto importante da nossa base de dados são as suas capacidades computacionais. A API do Clusterpoint 4 baseia-se no conceito de que uma base de dados deve permitir a realização de cálculos arbitrários nos dados diretamente na base de dados. Assim, permitimos que várias fases de recuperação usem código JavaScript arbitrário para transformar os dados. Além disso, o código JavaScript acede a índices em vez de objetos brutos, proporcionando assim uma elevada eficiência. [F10]

O Clusterpoint oferece transações multi-documento compatíveis com ACID hoje, uma funcionalidade que o MongoDB só tem nos seus planos futuros. Além disso, o Clusterpoint incorpora as funcionalidades de um motor de pesquisa de texto completo. Rasscevskis vê isto como uma oportunidade para utilizar uma única base de dados em vez de três (um motor de pesquisa, um SGBD relacional e uma base

de dados de documentos). Também pode executar o Clusterpoint nas suas instalações se tiver criado uma conta. Claramente, isto é útil quando lida com volumes elevados de dados ou precisa de reduzir a latência, ou se considera que os seus dados são demasiado sensíveis para serem colocados na nuvem, apesar de o Clusterpoint oferecer encriptação de dados em repouso.[F11]

[F9]: <https://en.wikipedia.org/wiki/Clusterpoint>

[F10]:

<https://insidebigdata.com/2015/10/06/clusterpoint-4-computing-engine-combines-instantly-scalable-database-and-computational-power/>

[F11]:

<https://www.infoworld.com/article/2940140/clusterpoint-launches-document-database-as-a-service.html>

[F12]: <https://www.crunchbase.com/organization/clusterpoint>

Caraterísticas principais do Clusterpoint



- Os dados são geridos em formato aberto e padrão da indústria, em XML ou JSON, através de API's abertas como a Clusterpoint server Python API[F13] ou a JavaScript Node.js API [F14];
- Base de dados agnóstica em relação à estrutura dos dados e rica em tipos; [F9]
- Suporte de plataforma cruzada. O software da base de dados Clusterpoint pode ser compilado noutros sistemas operativos;
- Arquitetura de software de cluster multi-mestre;
- Escalabilidade horizontal da base de dados.

Notas de orador:

Apesar do Clusterpoint ter sido descontinuado, este apresentava caraterísticas únicas que o distinguia dos demais. Sendo alguns dos seus principais benefícios o facto de que os seus dados eram geridos em formato aberto e padrão da indústria, em XML ou JSON, através de API's abertas tais como a Clusterpoint server Python API e a JavaScript Node.js API. Para além disso era uma base de dados agnóstica em relação à estrutura dos dados, sendo lida com documentos XML ou JSON de estrutura variável numa única base de dados. Para além de tudo, este serviço suportava dados textuais não estruturados, datas, números e metadados. Ainda suportava plataforma cruzada, permitindo o software da base de dados Clusterpoint ser compilado em diferentes sistemas operativos. Esta base de dados possuía arquitetura de software de cluster multi-mestre, onde qualquer nó de cluster pode servir como mestre e executar a aplicação de gestão. Por fim, possui ainda uma escalabilidade horizontal da base de dados, onde a partir de um único servidor dimensiona-se até alguns milhares de servidores interligados num infraestrutura de cluster. O

clusterpoint possuía diversas aplicações complementares que melhoram a experiência da sua utilização, tornando numa base de dados mais completa e útil.

Conteúdo:

Apesar do Clusterpoint ter sido descontinuado este apresentava algumas características únicas que o distinguia dos demais. Os principais benefícios eram os dados que eram geridos em formato aberto e padrão da indústria, em XML ou JSON, através de API's abertas como a Clusterpoint server Python API[F13] ou a JavaScript Node.js API [F14]. A Base de dados agnóstica em relação à estrutura dos dados e rica em tipos, lida com documentos em XML ou JSON de estrutura variável numa única base de dados. Suporta dados textuais não estruturados, datas, números e metadados (todos os tipos de XML e JSON). Para além disso tem o suporte de plataforma cruzada: os binários estão disponíveis para Linux, FreeBSD, Mac OS X e Windows. O software da base de dados Clusterpoint pode ser compilado noutros sistemas operativos. Ainda possui arquitetura de software de cluster multi-mestre: qualquer nó de cluster pode servir como mestre e executar a aplicação de gestão. Por fim possui ainda uma escalabilidade horizontal da base de dados: dimensiona-se a partir de um único servidor até alguns milhares de servidores interligados numa infraestrutura de cluster. O Clusterpoint possuía diversas aplicações complementares tais como o clusterpoint DBMS, uma Base de dados NoSQL em cluster, que utiliza a abordagem de vários servidores para distribuir a carga e aumentar o desempenho. A base de dados Clusterpoint facilita uma elevada paralelização de computação e distribuição de dados. GOL, ferramenta de Análise de Big Data SIEM da Clusterpark - Pesquisa e Análise de Registos de Logs, Eventos e Registos de Segurança. Desnormalização rápida de SQL para base de dados

NoSQL - importa uma base de dados SQL multi-tabela para uma única base de dados Clusterpoint utilizando a desnormalização automática. E ainda o NTSS, Sistema de Vigilância de Tráfego de Rede para Intercepção Legal - Captura de alta velocidade, armazenamento, pesquisa e análise de todo o tráfego da Internet na rede empresarial.[F9]

[F9]: <https://en.wikipedia.org/wiki/Clusterpoint>

[F12]: <https://www.crunchbase.com/organization/clusterpoint>

[F13]: <https://pypi.org/project/clusterpoint-api/>

[F14]: <https://www.npmjs.com/package/cps-api>

Exemplo de uso da Api JavaScript Node.js [F17]

```
'use strict';
var Clusterpoint = require('clusterpoint');
var co = require('co');
var config = {
  host: 'api-eu.clusterpoint.com',
  account_id: 'ACCOUNT_ID',
  username: 'USERNAME',
  debug: true,
  port: 443
};
var cp = new Clusterpoint(config);
var bookshelfDB = cp.database('bookshelf');
var authorsCollection = bookshelfDB.collection('authors');
var booksCollection = cp.database('bookshelf').collection('books');
return co(function* () {
  console.log('select and delete all books');
  var response = yield booksCollection.limit(10000).get();
  var idsArr = [];
  for (let doc of response) {
    idsArr.push(doc._id);
  }
  if (idsArr.length > 0) {
    var response = yield booksCollection.deleteMany(idsArr);
  }
  console.log('select and delete all authors');
  var response = yield authorsCollection.limit(10000).get();
  var idsArr = [];
  for (let doc of response) {
    idsArr.push(doc._id);
  }
  if (idsArr.length > 0) {
    var response = yield authorsCollection.deleteMany(idsArr);
  }
  console.log('insert authors');
  var documents = [
    {
      '_id': 1,
      'name': 'John'
```

```
    {
      '_id': 2,
      'name': 'Fred'
    }
  ];
  var response = yield authorsCollection.insertMany(documents);
  console.log('insert books');
  var documents = [
    {
      '_id': 1,
      'title': 'Book 1',
      'category': 'Science',
      'color': 'red',
      'availability': true,
      'author_id': 1,
      'price': 1,
      'deeper': {
        'foo': 1,
        'bar': 2
      }
    },
    {
      '_id': 2,
      'title': 'Book 2',
      'category': 'Fiction',
      'author_id': 2,
      'price': 2,
      'color': 'red',
      'availability': true
    }
  ];
  var response = yield booksCollection.insertMany(documents);
  var response = yield booksCollection.raw('SELECT *, author.name ' +
    'FROM books ' +
    'LEFT JOIN authors AS author ON author._id == books.author_id');
```

[F17]: [GitHub - clusterpoint/nodejs-client-api-v4](https://github.com/clusterpoint/nodejs-client-api-v4): Clusterpoint Node.js API for Clusterpoint v4.* database.

MarkLogic



O MarkLogic é uma base de dados NoSQL multi modelo que evoluiu a partir das suas raízes de base de dados XML para armazenar nativamente documentos **JSON** e triplas **RDF** para o seu modelo de dados semânticos.

Os dados devem ser recuperados de forma **rápida** e **fácil** e apresentados de uma forma que faça sentido para diferentes tipos de utilizadores. Além disso, os dados devem ser mantidos de forma confiável por uma solução de software **escalável** e de nível empresarial que seja executada em **hardware comum**. [A22]

Principais vantagens [A23] :

- Escalabilidade multi-TB em hardware comum;
- Indexação automática de valores XML e estrutura em um índice universal;
- Integração sobre HTTP.

```
xquery version "1.0-ml";

declare namespace qry =
"http://marklogic.com/cts/query";

let $binary-term :=

xdmp:plan (/binary ()) // qry:term-query /
qry:key/text ()

return cts:uris ((), (),
cts:term-query ($binary-term))
```

ORADOR:

MarkLogic Server é uma base de dados multimodelo, é uma solução de software para aproveitar conteúdos digitais numa única base de dados . Este possui documentos no lugar de um esquema. O seu modelo de dados pode ser tão flexível ou rígido quanto quisermos. O produto do MarkLogic é considerado uma base de dados NoSQL multimodelo devido à sua capacidade de armazenar, gerir e pesquisar documentos JSON e XML, bem como dados semânticos (triplas RDF).

A filosofia por detrás do Marklogic é que o armazenamento de dados é simplesmente uma parte da solução. Os dados também devem ser recuperados de forma rápida e fácil e que faça sentido para diferentes tipos de utilizadores. Sendo também possível observar um exemplo de código de marklogic no slide . As suas principais vantagens , é o facto de suportar XQuerys e outros padrões relevantes ,indexação automática de valores XML e estrutura em um índice universal, integração sobre HTTP e principalmente, escalabilidade multi-TB em hardware comum.

CONTEÚDO :

MarkLogic Server é uma base de dados multimodelo. Conforme citado na documentação oficial do MarkLogic Server . O

MarkLogic Server é uma solução de software poderosa para aproveitar o seu conteúdo digital numa única base de dados. MarkLogic possui documentos no lugar de um esquema. Entretanto, no MarkLogic, é possível simplesmente definir um modelo de documento de dados nos seus arquivos. O modelo de dados do seu documento pode ser tão flexível ou rígido quanto desejarmos. Em contraste com um esquema RDBMS, no modelo de registros de documentos, podemos criar os documentos após carregá-los no MarkLogic e posteriormente podem ser refinados conforme os nossos requisitos ou no caso de uso.[A22]

MarkLogic armazena documentos em seu próprio repositório transacional que foi desenvolvido especificamente com foco no máximo desempenho e integridade de dados. Como o banco de dados MarkLogic é transacional, você pode inserir ou atualizar um conjunto de documentos como uma unidade atômica e fazer com que a próxima consulta seja capaz de ver essas alterações com latência zero. MarkLogic suporta o conjunto completo de propriedades ACID.[A23]

MarkLogic, ao armazenar documentos JSON, oferece suporte a tipos de dados JSON padrão, incluindo matrizes e objetos. Os itens de menu do Hackolade, menus contextuais, dicas de ferramentas de ícones da barra de ferramentas e documentação são adaptados à terminologia e ao conjunto de recursos do MarkLogic.[A25]

A plataforma de Bases de dados Enterprise NoSQL da MarkLogic é utilizada em vários setores, incluindo publicidade, no governo e nas finanças, e é empregada em vários sistemas atualmente em produção.[A22]

Principais vantagens são o facto de ter escalabilidade multi-TB em hardware comum, suporte para XQuery, WebDAV, XPath 2.0, XML Schema e outros padrões relevantes, indexação automática de valores XML e estrutura em um índice universal, integração sobre HTTP, .NET e Java, suporte para atualizações em

nível de nó e funcionalidade de alerta oferece suporte a notificação, roteamento ou outro processamento de documentos de entrada para consultas sofisticadas. [A23]

[A22] - <https://en.wikipedia.org/wiki/MarkLogic>

[A23]- <https://docs.marklogic.com/guide/concepts/overview>

[A24]- <https://www.fpml.org/vendors/mark-logic-corporation/>

[A25]- https://en.wikipedia.org/wiki/MarkLogic_Server

Características principais do MarkLogic[F8]



- O Modelo de Documentos - Flexível e Orientado para Humanos
- Modelo de Base de Dados de Gráfico Semântico para Relações
- Capacidades de Pesquisa Geoespacial
- Vistas Estruturadas e Relacionais de Dados
- Índices Integrados
- Consultas Componíveis
- Plataforma Unificada

Notas de Orador:

Tal como todas as bases de dados, o MarkLogic também possui as suas características individuais. Algumas das suas características principais são o seu modelo de documentos, flexível e orientado para humanos, com outros benefícios como o desenvolvimento rápido, sem esquema definido, e os dados “desnormalizados” que aproveitam todos os atributos e consultam tudo no contexto, o seu modelo de base de dados de gráfico semântico para relações, enquanto o MarkLogic Server fornece um outro modelo de dados de gráfico semântico na forma de um armazenamento de triplas RDF incorporado, que armazena e gere dados semânticos. Para além disto, o MarkLogic tem uma capacidade de pesquisa geoespacial onde o seu modelo de dados de documentos fornece a flexibilidade para armazenar dados geospaciais, permitindo o MarkLogic armazenar, gerir e pesquisar dados geospaciais, nativamente, incluindo pontos de interesse, trajetos interceptados e regiões de interesse. Este permite ainda a vista estruturada e relacional de dados, através do suporte do SQL padrão. Os

Índices integrados do MarkLogic também são úteis, uma vez que bases de dados multimodelo fornecem uma interface de pesquisa unificada para consultar vários modelos de dados usando índices integrados. O MarkLogic permite ainda a consulta de dados utilizando Search, SQL, SPARQL ou API REST. Sendo uma plataforma de dados unificada, aumenta a produtividade dos programadores e a eficiência operacional.

Conteúdo:

[F8] As características principais do MarkLogic são o seu modelo de documentos, que é flexível e orientado para humanos, trazendo consigo outros benefícios tais como desenvolvimento rápido, sem esquema definido, dados "desnormalizados" que aproveitam todos os atributos e consultam tudo no contexto, sendo Ideal para a integração de dados; o seu modelo de base de dados de gráfico semântico para relações, já o MarkLogic Server fornece um modelo de dados de gráfico semântico na forma de um armazenamento de triplas RDF incorporado, que armazena e gere dados semânticos. Chamamos a essa capacidade de "MarkLogic Semantics". A semântica aprimora o modelo de documentos, fornecendo uma maneira inteligente de conectar e melhorar os documentos JSON e XML. Isso facilita a integração de dados e permite consultas mais poderosas para descobrir relações e fazer inferências. Para além disto, o MarkLogic tem uma capacidade de pesquisa geoespacial onde o seu modelo de dados de documentos fornece a flexibilidade para armazenar dados geoespaciais, permitindo o MarkLogic armazenar, gerir e pesquisar dados geoespaciais, nativamente, incluindo pontos de interesse, trajetos intercetados e regiões de interesse. Este permite ainda a vista estruturada e relacional de dados, através do suporte do SQL padrão. Os índices integrados do MarkLogic também são úteis, uma vez que bases de dados multimodelo fornecem uma interface de pesquisa unificada para consultar vários modelos de dados usando índices integrados. Normalmente, tem que escolher e gerir índices específicos para cada tipo de dados. Por outro lado, o

MarkLogic Server possui uma suíte integrada de índices que permitem um acesso rápido aos dados - imediatamente após o carregamento dos dados. Uma base de dados multimodelo funciona de forma semelhante ao Google - o Google não exige que as páginas da web se encaixem num determinado formato, apenas as indexa e torna acessíveis através de uma interface de pesquisa unificada. O MarkLogic permite ainda a consulta de dados utilizando Search, SQL, SPARQL ou API REST. Também suporta várias linguagens de programação como JavaScript, Node, Java e XQuery. Para além disto o MarkLogic é uma plataforma unificada, possuindo um único modelo de segurança de dados, governança e transicional. Sendo uma plataforma de dados unificada, aumenta a produtividade dos programadores e a eficiência operacional.

[F8] : [Benefits of a Multi-Model Database - MarkLogic Server - MarkLogic](#)

Apache CouchDB



Apache CouchDB é uma base de dados NoSQL orientada a documentos de open-source, implementada em Erlang, uma linguagem de programação de uso geral e um sistema para execução. [B1][B2]

O projeto CouchDB foi criado em abril de 2005 por Damien Katz, ex-desenvolvedor do Lotus Notes na IBM. Ele auto-financiou o projeto por quase dois anos e o lançou como um projeto open-source sob a Licença Pública Geral GNU. [B1]

Ela usa JSON para armazenar dados, JavaScript como linguagem de consulta (query) usando MapReduce e HTTP para uma API. [B1]

Couch é um acrónimo para “cluster of unreliable commodity hardware”, ou seja, “aglomerado de Hardware de comodidade não confiável”. [B3]

Conteúdo:

Apache CouchDB é uma base de dados NoSQL orientada a documentos open-source, implementada em Erlang. O CouchDB usa vários formatos e protocolos para armazenar, transferir e processar dados. Ela usa JSON para armazenar dados, JavaScript como linguagem de consulta usando MapReduce e HTTP para uma API. O projeto CouchDB foi criado em abril de 2005 por Damien Katz, ex-desenvolvedor do Lotus Notes na IBM. Ele auto-financiou o projeto por quase dois anos e o lançou como um projeto open-source sob a Licença Pública Geral GNU. Em fevereiro de 2008, tornou-se um projeto Apache Incubator e foi oferecido sob a licença Apache. Poucos meses depois, passou para um projeto de alto nível. Isso levou ao lançamento da primeira versão estável em julho de 2010. No início de 2012, Katz deixou o projeto para se concentrar no Couchbase Server. Desde a saída de Katz, o projeto Apache CouchDB continuou, lançando 1.2 em abril de 2012 e 1.3 em abril de 2013. Em julho de 2013, a comunidade CouchDB fundiu a base de código do BigCouch, a versão clusterizada do CouchDB do Cloudant, no projeto Apache. A estrutura de clustering BigCouch está incluída na versão atual do Apache CouchDB. [B1]

Erlang é uma linguagem de programação de uso geral e um sistema para execução. Foi desenvolvida pela Ericsson para suportar aplicações distribuídas e tolerantes a falhas a serem executadas em um ambiente de tempo real e ininterrupto. Ela suporta nativamente “hot swapping”, de forma que o código pode ser modificado sem parar o sistema. [B2]

Commodity Hardware às vezes conhecido como “hardware pronto-para-uso”, é um computador ou componente relativamente barato, amplamente disponível e basicamente intercambiável com outro hardware do seu tipo. Ao contrário do hardware desenvolvido especificamente para uma função específica, o commodity hardware pode executar muitas funções diferentes.

Um computador de comodidade, por exemplo, é um PC que não possui recursos excepcionais e está facilmente disponível para compra. Os discos rígidos de comodidade podem ser configurados como uma matriz redundante de discos independentes (RAID) para tolerância a falhas e “failover”. Em muitos ambientes, vários servidores de baixo custo compartilham a carga de trabalho. Os servidores de comodidade podem ser considerados descartáveis e normalmente são substituídos em vez de reparados por ser mais barato. [B3]

[B1] - https://en.wikipedia.org/wiki/Apache_CouchDB

[B2] -

[https://pt.wikipedia.org/wiki/Erlang_\(linguagem_de_programa%C3%A7%C3%A3o\)](https://pt.wikipedia.org/wiki/Erlang_(linguagem_de_programa%C3%A7%C3%A3o))

[B3] - <https://www.suse.com/suse-defines/definition/commodity-hardware/>

Vantagens do CouchDB



- Combina um modelo de armazenar documentos intuitivo com um motor de consulta poderoso; [B4]
- Tolerante a falhas; [B4]
- Escalabilidade extrema; [B4]
- Replicação incremental; [B4]
- Open-source; [B5]
- Integra-se facilmente a infra estruturas de gerenciamento de dados; [B5]
- Flexível; [B5]
- Usa um modelo de dados sem esquema; [B5]

"Com o CouchDB, nós simplesmente obtemos todos os nossos dados em JSON e os armazenamos na base de dados em, literalmente, minutos." [B15]

Conteúdo:

Capítulo 1, página 4:

"Nós acreditamos que o CouchDB irá drasticamente mudar a forma de como tu crias aplicações à base de documentos. CouchDB combina um modelo de armazenar documentos intuitivo com um motor de consulta poderoso de uma forma tão simples que provavelmente perguntarás, "Porque é que nunca ninguém criou algo assim?" O modelo do CouchDB baseia-se muito da arquitetura web e dos conceitos de recursos, métodos, e representações. Ele melhora isto com formas poderosas de consultar, mapear, combinar, e filtrar os teus dados. Adiciona tolerância a falhas, escalabilidade extrema, e replicação incremental, e o CouchDB define um ótimo lugar para bases de dados baseadas em documentos." [B4]

Ao contrário das bases de dados relacionais, o CouchDB usa um modelo de dados sem esquema, que simplifica o gerenciamento de registros em vários dispositivos de computação, telemóveis e navegadores web. Sendo um projeto open-source, o CouchDB é apoiado por uma comunidade ativa de desenvolvedores que melhoram continuamente o software com foco na facilidade de uso e na adoção da web. Para a maioria das empresas, decidir qual fornecedor usar ao adquirir tecnologia de gerenciamento de dados pode ser um desafio. O software proprietário não apenas impõe certas restrições de licenciamento ao uso da tecnologia, mas também há preocupações com a continuidade dos negócios ao mover todos os dados da empresa para um sistema de gerenciamento de base de dados (SGBD) de "tamanho único", sem visibilidade sobre sua estrutura interna.

O CouchDB é diferente. Ao contrário do software proprietário que pode correr o risco de “aprisionamento de fornecedor”, o CouchDB open-source, de uso gratuito e integra-se facilmente à sua infraestrutura atual de gerenciamento de dados. Por ter mais controle sobre o software, também tem-se mais flexibilidade ao adaptá-lo às necessidades exclusivas do seu negócio. Seja exigindo um armazenamento de documentos de uso geral, permitindo a sincronização eficiente de dados ou adotando uma mentalidade “Offline First”, o CouchDB oferece às empresas a flexibilidade necessária para criar infra estruturas duráveis, confiáveis e escaláveis. [B5]

Com o CouchDB, nós simplesmente obtemos todos os nossos dados em JSON e os armazenamos na base de dados em, literalmente, minutos. Não é preciso fazer nenhuma normalização ou algo parecido, e o transporte é HTTP, então temos muitas opções de cliente. [B15]

[B4] -

https://books.google.pt/books?hl=pt-PT&lr=&id=G4N-DPk9R5sC&oi=fnd&pg=PR7&dq=couchdb&ots=OkR1IijANY&sig=CYTlIHx2rFHyMTt08uR5rL_jWSg&redir_esc=y#v=onepage&q&f=false

[B5] - <https://www.ibm.com/topics/couchdb>

[B15] -

<https://stackoverflow.com/questions/644695/what-are-the-advantages-of-couchdb-vs-an-rdbms>

Desvantagens do CouchDB



- Ocupa muito espaço para sobrecarga (overhead) [B7];
- Queries arbitrárias são caras; [B6]
- Visualizações temporárias em grandes conjuntos de dados são muito lentas; [B6]
- Não suporta transações; [B6]
- Replicação de grandes bases de dados pode falhar; [B6]
- Mais lento que SGBDs de memória (in-memory); [B16]
- As atualizações locais exigem lógica do lado do servidor (update handlers); [B16]

“Os dados estão em JSON, o que significa que os documentos são muito grandes e ter nomes de chaves descritivos, na realidade prejudica, pois aumentam o tamanho do documento.” [B16]

Conteúdo:

Desvantagens do CouchDB incluem as seguintes:

CouchDB ocupa muito espaço para sobrecarga (*overhead), o que é uma enorme desvantagem quando comparado a outras bases de dados; Queries arbitrárias são caras; Tem um pouco de espaço de sobrecarga extra para o CouchDB quando comparado com a maioria das alternativas; Visualizações temporárias em grandes conjuntos de dados são muito lentas; Não suporta transações; Replicação de grandes bases de dados pode falhar. [B6]

*overhead - Na ciência da computação, “overhead” ou sobrecarga é qualquer combinação de tempo de computação excessivo ou indireto, memória, largura de banda ou outros recursos necessários para executar uma tarefa específica. É um caso especial de sobrecarga de engenharia. A sobrecarga pode ser um fator decisivo no design de software, no que diz respeito à estrutura, correção de erros e inclusão de recursos. Exemplos de sobrecarga computacional podem ser encontrados em Programação Orientada a Objetos (OOP), programação funcional, transferência de dados e estruturas de dados. [B7]

“Em anexo está uma lista de pontos a ter em atenção:

Postagens de blogs de 2010 afirmam “não ser maduro o suficiente” (seja lá o que isso valha); Mais lento que o SGBDs de memória; As atualizações locais exigem lógica do lado do servidor (update handlers); Troca disco por velocidade: As bases de dados podem se tornar enormes em comparação com outros SGBDs (embora exista

funcionalidade de compactação); "Apenas" consistência eventual; As visualizações temporárias em grandes conjuntos de dados são muito lentas; A replicação de grandes bases de dados pode falhar; O paradigma mapear/reduzir requer repensar (apenas para ser completo).

O único ponto que me preocupa é o nº 3 (atualizações locais), porque é bastante inconveniente." [B16]

"Os dados estão em JSON, o que significa que os documentos são muito grandes e ter nomes de chaves descritivos, na realidade prejudica, pois aumentam o tamanho do documento; Não suporta transações, Isso significa que impor a exclusividade de um campo em todos os documentos não é seguro, por exemplo, impor que um nome de usuário seja único. Outra consequência da incapacidade do CouchDB de suportar a noção típica de transação é que coisas como aumentar/diminuir um valor e salvá-lo de volta também são perigosas. Não há muitos casos em que gostaríamos de simplesmente aumentar/diminuir algum valor onde não pudéssemos simplesmente armazenar os documentos individuais separadamente e agregá-los com uma visualização." [B16]

[B6] - <https://www.geeksforgeeks.org/introduction-to-apache-couchdb/>

[B7] - [https://en.wikipedia.org/wiki/Overhead_\(computing\)](https://en.wikipedia.org/wiki/Overhead_(computing))

[B16] - <https://stackoverflow.com/questions/7858699/disadvantages-of-couchdb>

ArangoDB



ArangoDB é um sistema de base de dados multi-modelo desenvolvido pela ArangoDB Inc. É multi-modelo uma vez que suporta três modelos de dados (gráficos, documentos JSON e chave/valor) com um núcleo de base de dados e uma linguagem de consulta unificada AQL (ArangoDB Query Language). [B8]

Documentos no ArangoDB são objetos JSON que contêm dados estruturados ou semiestruturados. [B9]

Fundada em 2015 por Claudius Weinberger e Frank Celler. Eles originalmente chamaram o sistema de base de dados de "A Versatile Object Container", ou AVOC, o que os levou a chamar a base de dados de AvocadoDB. Mais tarde, foi mudado para ArangoDB. A palavra "arango" referência um tipo de abacate pouco conhecida de Cuba. [B8]

Conteúdo:

ArangoDB é um sistema de base de dados gráfico desenvolvido pela ArangoDB Inc. ArangoDB é um sistema de base de dados multi-modelo uma vez que suporta três modelos de dados (gráficos, documentos JSON e chave/valor) com um núcleo de base de dados e uma linguagem de consulta unificada AQL (ArangoDB Query Language). AQL é principalmente uma linguagem declarativa e permite a combinação de diferentes padrões de acesso a dados em uma única consulta. ArangoDB é um sistema de base de dados NoSQL mas o AQL é semelhante de muitas maneiras ao SQL, ele utiliza o RocksDB como um motor de armazenamento. [B8]

ArangoDB Inc foi fundada em 2015 por Claudius Weinberger e Frank Celler. Eles originalmente chamaram o sistema de base de dados de "A Versatile Object Container", ou AVOC, o que os levou a chamar a base de dados de AvocadoDB. Mais tarde, foi mudado para ArangoDB. A palavra "arango" referência um tipo de abacate pouco conhecido de Cuba.

Em janeiro de 2017, a ArangoDB levantou uma ronda inicial de investimento de 4,2 milhões de euros liderada pela Target Partners. Em março de 2019, ArangoDB levantou 10 milhões de dólares em financiamento da série A liderado pela Bow Capital. Em outubro de 2021, ArangoDB levantou 27,8 milhões de dólares em financiamento da série B liderado pela Iris Capital. [B8]

Documentos no ArangoDB são objetos JSON que contêm dados estruturados ou

semi-estruturados. Eles são armazenados em coleções. Cada documento possui uma chave imutável que o identifica numa coleção e um identificador, derivado da chave, que o identifica exclusivamente dentro de uma base de dados.

Os documentos podem armazenar valores primitivos, listas de valores e objetos aninhados (em qualquer profundidade). JSON e, portanto, ArangoDB suporta os seguintes tipos de dados: Null para representar a ausência de um valor, também conhecido como nil ou none; True e False, os valores booleanos, para representar sim e não, ativado e desativado, etc; Números, para armazenar valores inteiros e de ponto flutuante (floats); Strings, para armazenar sequências de caracteres para texto, codificadas como UTF-8; Matrizes, para armazenar listas que podem conter qualquer um dos tipos de dados suportados como elementos, incluindo matrizes e objetos aninhados; Objetos, para mapear chaves para valores como um dicionário, também conhecido como matrizes associativas ou mapas hash. As chaves são strings e os valores podem ser qualquer um dos tipos de dados suportados, incluindo matrizes e objetos aninhados. [B9]

[B8] - <https://en.wikipedia.org/wiki/ArangoDB>

[B9] - <https://docs.arangodb.com/3.11/concepts/data-structure/documents/>

Vantagens do ArangoDB



- Base de dados multi-modelo flexível; [B11]
- Escalonamento de performance simplificado; [B11]
- Tolerância a falhas; [B11]
- Forte consistência dos dados; [B11]
- Bom construtor de queries; [B10]
- Utiliza AQL para queries complexas; [B10]
- Modelos abstratos para substituir SGBD; [B10]

"Ao escolher o ArangoDB, podemos utilizar uma única base de dados para as funcionalidades: chats, relacionamentos, compartilhamento de documentos, diversos tipos de entidades (empresas, pessoas, universidades, funcionários, fundos de risco, etc.)." [B17]

Conteúdo:

Razões para escolher o ArangoDB: Capacidades multi-modelo com junções (joins); Apoio comunitário ativo; Construtor de queries do ArangoDB; Utiliza AQL para queries complexas; Modelos abstratos para substituir SGBD, se necessário. [B10]

Por ser uma base de dados multi-modelo nativa, o ArangoDB elimina a necessidade de implantar várias bases de dados e, assim, diminui o número de componentes e sua manutenção. Consequentemente, reduz a complexidade da pilha de tecnologia para o aplicativo. Além de consolidar as suas necessidades técnicas gerais, esta simplificação leva à redução do custo total de propriedade e ao aumento da flexibilidade.

Com o crescimento das aplicações ao longo do tempo, o ArangoDB pode atender às crescentes necessidades de desempenho e armazenamento, escalonando de forma independente com diferentes modelos de dados. Como o ArangoDB pode ser dimensionado tanto vertical quanto horizontalmente, caso seu desempenho exija uma diminuição (uma desaceleração deliberada e desejada), seu sistema back-end pode ser facilmente reduzido para economizar hardware e custos operacionais. Se forem usados várias bases de dados de modelo único, a consistência dos dados pode se tornar um problema. Essas bases de dados não foram projetadas para se comunicarem entre si; portanto, alguma forma de funcionalidade de transação precisa ser implementada para manter seus dados consistentes entre diferentes modelos. Oferecendo suporte a transações ACID, o ArangoDB gerencia seus diferentes modelos de dados com um único back-end, fornecendo forte consistência

em uma única instância e operações atômicas ao operar no modo cluster. É um desafio construir sistemas tolerantes a falhas com muitos componentes não relacionados. Este desafio torna-se mais complexo quando se trabalha com clusters. É necessária experiência para implantar e manter esses sistemas, usando diferentes tecnologias e/ou pilhas de tecnologia. Além disso, a integração de múltiplos subsistemas, concebidos para funcionar de forma independente, inflige grandes custos operacionais e de engenharia. Como uma pilha de tecnologia consolidada, o banco de dados multi-modelo apresenta uma solução elegante. Projetado para permitir arquiteturas modernas e modulares com diferentes modelos de dados, o ArangoDB também funciona para uso em cluster. [B11]

“Frequentemente lidamos com projetos que enfrentam um alto nível de incerteza. Com aplicativos de rede, não se pode prever quantas conexões terá no futuro e quão complexas elas se tornarão. Portanto, pode ser difícil escolher uma pilha de tecnologia ideal. No entanto, pode haver uma base de dados perfeita para tais casos. Esta base de dados chama-se ArangoDB. Ao escolher o ArangoDB, podemos utilizar uma única base de dados para as funcionalidades: chats, relacionamentos, compartilhamento de documentos, diversos tipos de entidades (empresas, pessoas, universidades, funcionários, fundos de risco, etc.).” [B17]

[B10] - <https://brainhub.eu/library/arangodb-use-case>

[B11] - https://www.tutorialspoint.com/arangodb/arangodb_advantages.htm

[B17] - <https://www.mindk.com/blog/arangodb/>

Desvantagens do ArangoDB



- Transações no ArangoDB não podem ser aninhadas (não podem começar outra transação); [B12]
- Pequena comunidade; [B13]
- Ecossistema ainda em evolução; [B13]
- Falta de documentação; [B14]
- Dificuldade em encontrar suporte; [B14]
- Erros aleatórios sob condições de carga pesada; [B14]
- Grande consumo de memória; [B18]
- Escalabilidade limitada; [B18]

"Embora o ArangoDB tenha muitos pontos fortes, pode não ser a melhor escolha para todas as situações. É importante avaliar cuidadosamente os nossos requisitos específicos e considerar outras opções antes de decidir usar o ArangoDB." [B18]

Conteúdo:

As transações no ArangoDB não podem ser aninhadas, ou seja, uma transação não deve iniciar outra transação. Se for feita uma tentativa de chamar uma transação de dentro de uma transação em execução, o servidor lançará o erro 1651 (transações aninhadas detectadas). Também não é permitido executar transações do usuário em algumas das coleções do próprio sistema do ArangoDB. Isso não deve ser um problema para o uso regular, pois as coleções do sistema não conterão dados do usuário e não há necessidade de acessá-los a partir de uma transação do usuário. [B12]

Comunidade menor: Embora a documentação do produto seja bastante completa, existem alguns cenários que não são abordados. Nós encontramos perguntas respondidas no StackOverflow em alguns casos; Ecossistema ainda em evolução: A integração do GraphQL é limitada e o esquema precisa ser gerado manualmente. Alguns drivers de idioma não são mantidos. Embora exista integração com Spring Data, ela não oferece funcionalidade gráfica completa e não há outro suporte oficial semelhante a ORM ou OGM; [B13]

Falta de documentação: Alguns usuários descobriram que falta documentação para ArangoDB em certas áreas, como clustering e técnicas de otimização AQL. Outros consideraram-no ambíguo e mal escrito, tornando-o difícil de entender e usar; Dificuldade em encontrar suporte: Há uma grave falta de ferramentas e uma pequena comunidade em torno do ArangoDB, o que pode tornar difícil para alguns usuários

encontrar suporte para problemas que encontram ao usar o software; Erros aleatórios sob carga pesada: Alguns usuários experienciam erros aparentemente aleatórios e sobrecargas de aplicativos ao usar o ArangoDB sob condições de carga pesada, fazendo com que a base de dados travesse. [B14]

Grande consumo de memória: ArangoDB pode usar mais memória do que outras bases de dados, portanto, pode não ser a melhor escolha para aplicativos que exigem baixa latência ou exigem que uma base de dados seja capaz de escalar rapidamente; Escalabilidade limitada: ArangoDB não foi projetado para escalar horizontalmente, portanto, se precisar de lidar com grandes quantidades de dados, pode não ser a escolha certa. [B18]

“Embora o ArangoDB tenha muitos pontos fortes, pode não ser a melhor escolha para todas as situações. É importante avaliar cuidadosamente os nossos requisitos específicos e considerar outras opções antes de decidir usar o ArangoDB.” [B18]

[B12] - <https://docs.arangodb.com/3.11/develop/transactions/limitations/>

[B13] - <https://cycode.engineering/blog/aws-neptune-neo4j-arangodb-or-redisgraph-how-we-at-cycode-chose-our-graph-database/>

[B14] - <https://www.trustradius.com/products/arangodb/reviews?qs=pros-and-cons#reviews>

[B18] - <https://www.quora.com/Are-there-any-cons-to-using-ArangoDB>

CouchDB vs ArangoDB (Comparação)



CouchDB:

- A compactação é ativada manualmente por base de dados; [B19]
- Escalabilidade extrema;
- Usa MapReduce como linguagem de query; [B20]
- Armazenamento de documentos JSON nativo; [B21]
- Replicação multifonte; [B21]

ArangoDB:

- Coleta de lixo automática; [B19]
- Escalabilidade limitada;
- Usa AQL como linguagem de query;
- Permite gráficos e documentos em uma base de dados; [B20]
- Replicação de réplica de origem com fator de replicação configurável; [B21]

Conteúdo:

Os arquitetos de bases de dados podem escolher entre diferentes estratégias para lidar com atualizações e exclusões. Especialmente nas primeiras bases de dados (relacionais), a abordagem de “atualização no local” foi amplamente utilizada. “Atualização no local” significa que um registro é substituído in-situ por uma nova versão deste registro. O MVCC, por outro lado, descreve uma semântica de acréscimo: a nova versão de um registro será anexada a um registro cronológico e deixará os dados antigos intactos.

ArangoDB 1.0 e 1.1 armazenam todas as revisões de documentos apenas com acréscimos. A versão mais recente de um documento está disponível para o usuário final. Nestas versões não há simultaneidade de escrita por coleção, portanto não é um MVCC “real”.

Uma base de dados que usa uma estratégia somente de acréscimo, como o MVCC, precisa cuidar de versões antigas e não mais utilizadas de um documento. Essas versões devem ser removidas regularmente para manter a base de dados o menor possível. No CouchDB a compactação é acionada manualmente por base de dados. ArangoDB possui uma coleta de lixo automática. [B19]

Os desenvolvedores descrevem o ArangoDB como “uma base de dados distribuída de código aberto com um modelo de dados flexível para documentos, gráficos e valores-chave”. Crie aplicativos de alto desempenho usando uma linguagem de consulta conveniente semelhante a SQL ou extensões JavaScript. Por outro lado, o CouchDB é detalhado como “base de dados de documentos HTTP + JSON com

visualizações Map Reduce e replicação baseada em pares". Apache CouchDB é uma base de dados que usa JSON para documentos, JavaScript para índices MapReduce e HTTP regular para sua API. CouchDB é uma base de dados que abraça completamente a web. Armazena seus dados com documentos JSON. Acesse seus documentos e consulte seus índices com seu navegador, via HTTP. Indexe, combine e transforme seus documentos com JavaScript. [B20]

Métodos de replicação do ArangoDB: Replicação de réplica de origem com fator de replicação configurável. Métodos de replicação do CouchDB: Replicação de réplica de origem e replicação multifonte. [B21]

[B19] -

<https://arangodb.com/2012/11/comparing-arangodb-with-mongodb-and-couchdb/>

[B20] - <https://stackshare.io/stackups/arangodb-vs-couchdb>

[B21] - <https://db-engines.com/en/system/ArangoDB%3BCouchDB>

CouchDB vs ArangoDB (Casos de uso)



CouchDB:

- Aplicativos com acumulação e alteração ocasional de dados com consultas predefinidas; [B23]
- Oferecer suporte para dispositivos móveis; [B23]
- Armazenar dados de série temporal para uma rede de sensores de teste; [B24]
- Oferecer um armazenamento de dados operacional sempre ativo; [B25]

ArangoDB: [B22]

- Detecção de fraude;
- Gráficos de conhecimento;
- Motores de recomendação;
- Gerenciamento de identidade e acesso;
- Operações de rede e IT;
- Gerenciamento de mídia social;
- Gerenciamento de tráfego;

Conteúdo:

ArangoDB como base de dados gráfica é ideal para casos de uso como detecção de fraude, gráficos de conhecimento, motores de recomendação, gerenciamento de identidade e acesso, operações de rede e IT, gerenciamento de mídia social, gerenciamento de tráfego e muito mais.

ArangoDB permite olhar além de pontos de dados individuais em fontes de dados diferentes, permitindo integrar e harmonizar dados para analisar atividades e relacionamentos todos juntos, para uma visão mais ampla dos padrões de conexão, para detectar comportamentos fraudulentos complexos, como redes de fraude. Sugira produtos, serviços e informações aos usuários com base em relacionamentos de dados. Por exemplo, você pode usar o ArangoDB junto com o PyTorch Geometric para construir um sistema de recomendação de filmes, analisando os filmes que os utilizadores assistiram e, em seguida, prevendo links entre os dois com uma rede neural gráfica (GNN).

Os dispositivos de rede e a forma como eles se interconectam podem ser naturalmente modelados como um gráfico.

Pode-se mapear um organograma como um gráfico e usar o ArangoDB para determinar quem está autorizado a ver quais informações. Coloque os recursos gráficos do ArangoDB em funcionamento para implementar listas de controle de acesso e herança de permissão. [B22]

Quais são os casos de uso do CouchDB? O CouchDB é adequado para aplicativos com dados acumulados e que mudam ocasionalmente com consultas predefinidas.

Se precisas de uma base de dados que funcione em dispositivos móveis, precisa de replicação mestre-mestre, ou durabilidade de servidor único, então o CouchDB é uma grande escolha. [B23]

O CouchDB resolve um problema interessante para nós: nós o usamos para armazenar dados de séries temporais para uma rede de sensores de teste e é uma ótima introdução aos conceitos e tecnologias NoSQL. [B24]

Um dos melhores casos de uso do Apache CouchDB é como um armazenamento de dados operacional sólido e sempre ativo. Ele foi desenvolvido para tolerância a falhas, armazenando várias cópias de seus dados em servidores separados para que um cluster CouchDB possa suportar a perda de vários nós sem perda de serviço. [B25]

[B22] - <https://docs.arangodb.com/3.11/introduction/use-cases/>

[B23] -

<https://thecustomizewindows.com/2019/03/what-ae-the-use-cases-for-couchdb/>

[B24] -

<https://www.trustradius.com/products/apache-couchdb/reviews?qs=product-usage#reviews>

[B25] -

https://medium.com/@glynn_bird/couchdb-and-the-data-warehouse-506b8ce81674

Firestore Realtime Database [A5]



O **Firestore Realtime Database** é uma base de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real para cada cliente conectado . Quando criamos aplicações do tipo multiplataforma com os nossos SDKs para Apple, Android e JavaScript, todos os seus clientes compartilham uma instância do Realtime Database e recebem automaticamente atualizações com os dados mais recentes.[A10]

Os dados são mantidos **localmente** e, mesmo **off-line**, os eventos em tempo real continuam a ser acionados, proporcionando uma experiência responsiva ao utilizador final. Quando o dispositivo recupera a conexão, o Realtime Database **sincroniza** as alterações feitas nos dados locais com as atualizações remotas que ocorreram enquanto o cliente estava off-line, mesclando qualquer conflito automaticamente.[A11]

ORADOR:

O Firestore Realtime Database é uma base de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real para cada cliente conectado. Quando criamos aplicações do tipo multiplataforma com os nossos SDKs para Apple, Android e JavaScript, todos os seus clientes compartilham uma instância do Realtime Database e recebem automaticamente atualizações com os dados mais recentes. Uma das principais vantagens deste tipo de bases de dados , é o facto de permitir o seu uso offline , sem qualquer restrição , os dados são mantidos localmente e assim que o utilizador conecta se a internet os dados são atualizados na bases de dados , sendo os dados sincronizados com as alterações que ocorreram enquanto este utilizador estava offline . Corrigindo qualquer tipo de conflito efetuado .

CONTEÚDO:

O Firestore Realtime Database é uma base de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real para cada cliente conectado. Quando criamos aplicações do tipo multiplataforma com os nossos SDKs para Apple, Android e JavaScript, todos os seus clientes

compartilham uma instância do Realtime Database e recebem automaticamente atualizações com os dados mais recentes.

Como funciona?

No Firebase Realtime Database é possível criar aplicações avançadas e colaborativas, ao conceder acesso seguro a base de dados diretamente do código do cliente. Os dados são mantidos localmente e, mesmo off-line, os eventos em tempo real continuam sendo acionados, proporcionando uma experiência responsiva ao utilizador final. Quando o dispositivo recupera a conexão, o Realtime Database sincroniza as alterações feitas nos dados locais com as atualizações remotas que ocorreram enquanto o cliente estava off-line, mesclando qualquer conflito automaticamente.

O Realtime Database é um banco de dados NoSQL e, por isso, tem otimizações e funcionalidades diferentes de uma base de dados relacional. [A10]

Principais recursos :

Em tempo real : Em vez de solicitações HTTP típicas, o Firebase Realtime Database usa a sincronização de dados. Sempre que os dados são alterados, todos os dispositivos conectados recebem essa atualização em milissegundos. Crie experiências colaborativas e imersivas sem se preocupar com códigos de rede.

Off-line : Os apps do Firebase permanecem responsivos mesmo off-line, pois o SDK do Firebase Realtime Database mantém seus dados em disco. Quando a conectividade é restabelecida, o dispositivo cliente recebe as alterações perdidas e faz a sincronização com o estado atual do servidor[A510]

Acessível em dispositivos clientes: É possível acessar o Firebase Realtime Database diretamente de um dispositivo móvel ou navegador da Web. Não é necessário um servidor de aplicativos. A segurança e a validação dos dados estão disponíveis por meio das Regras de segurança do Firebase Realtime Database, baseadas em expressão e executadas quando os dados são lidos ou gravados.[A11]

[A10]- <https://firebase.google.com/docs/database?hl=pt-br>

[A11]-

<https://console.cloud.google.com/marketplace/product/google-cloud-platform/firebase-realtime-database?pli=1>

Vantagens de Firebase Realtime DB [A12]



A base de dados Firebase Real-time usa sincronização de dados em vez de solicitações HTTP. Qualquer dispositivo conectado recebe as atualizações em milissegundos. Não pensa em código de rede e oferece experiências colaborativas e imersivas. [A39]

Principais vantagens [A13] :

- Sincronização de dados JSON em tempo real;
- Colaboração fácil entre vários dispositivos;
- Desenvolvimento de apps sem servidor;
- Segurança forte baseada no utilizador;
- Otimizado para uso off-line;

ORADOR:

Principais vantagens deste tipo de bases de dados, algumas já mencionadas como o facto deste tipo de bases de dados permitir que haja uma sincronização em tempo real , ficando mais fácil para os utilizadores acederem aos dados em qualquer sítio com um dispositivo , sendo também possível a colaboração entre si em tempo real . O realtime DB acompanha os SDKs (software development kits) e os dispositivos móveis . Ou seja , permite a criação de apps sem ser necessário servidores . O firebase também proporciona serviços de hospedagem bons, seguros e rápidos . Se desejarmos hospedar os seus micro servicos Express.js,HTML,... o suporte de hospedagem do firebase está sempre presente , conseguindo assim concluir que o firebase consegue hospedar uma grande variedade de conteúdos . E para finalizar , como já dito , quando os utilizadores estão offline os SDKs utilizam o cache local do dispositivo para armazenar as alterações e quando está de volta online , os dados são sincronizados .

CONTEÚDO:

A base de dados Firebase Real-time usa sincronização de dados em vez de solicitações HTTP. Qualquer dispositivo conectado

recebe as atualizações em milissegundos. Não pensa em código de rede e oferece experiências colaborativas e imersivas.

[A39]

O Firebase Realtime tem como principais vantagens as seguintes . É uma base de dados NoSQL hospedado na nuvem. Com ele, você armazena e sincroniza dados entre os seus utilizadores em tempo real, com a sincronização em tempo real, fica mais fácil para os utilizadores acederem aos dados em qualquer lugar, seja na Web ou em dispositivos móveis. Além disso, eles podem colaborar entre si, o Realtime Database acompanha os SDKs para Web e dispositivos móveis. Assim, você desenvolve apps sem precisar de servidores. Além disso, você executa códigos de back-end que respondem a eventos acionados pela sua base de dados com o Cloud Functions para Firebase, outra vantagem atraente do Firebase são seus serviços de hospedagem seguros e rápidos. A hospedagem Firebase oferece suporte a todos os tipos de conteúdo, incluindo aplicativos da web, conteúdo dinâmico e estático. Além disso, se desejarmos hospedar os seus microsserviços Express.js, HTML, CSS ou APIs, o suporte de hospedagem do Firebase está sempre lá. Isso significa que o Firebase hospeda uma grande variedade de conteúdo.[A13] Quando os utilizadores ficam off-line, os SDKs do Realtime Database usam o cache local no dispositivo para aplicar e armazenar alterações. Quando o dispositivo volta a ficar on-line, os dados locais são sincronizados automaticamente. [A12]

Embora o Google Analytics seja uma ferramenta gratuita fornecida pelo Google, sua integração se torna perfeita quando você trabalha com o Firebase. O Google Analytics é compatível com iOS, Android, Web, C ++ e configurações do Unity. Esta solução analítica gratuita relata aos desenvolvedores como os utilizadores estão se comportando em relação aos seus dispositivos móveis e aplicações web. O Firebase Analytics também é benéfico para melhorar as taxas de retenção e engajamento de utilizadores para os seus aplicativos .

Existem também vários recursos de teste gratuitos vinculados ao Firebase, onde podemos testar e visualizar seu projeto gratuitamente. Com certeza, após certo limite de uso de serviços ou consumo de memória da base de dados, você teria que escolher um plano pago para Firebase. No entanto, os pacotes pagos do Firebase também são fáceis de usar e você pode até usar uma calculadora de preços para estimar o custo do seu projeto.[A13]

[A12]- <https://firebase.google.com/products/realtime-database?hl=pt-br>

[A13]- <https://blog.back4app.com/pt/vantagens-do-firebase/>
[A39]- <https://www.javatpoint.com/firebase-realtime-database>



BaseX

BaseX é um sistema que gere bases de dados XML , nativo e leve e processador XQuery, desenvolvido como um projeto comunitário no GitHub. É especializado em armazenar, consultar e visualizar grandes documentos e coleções XML. BaseX é independente de uma plataforma .[A15]

Principais características :

- Serve como uma excelente estrutura para a construção de aplicativos da web complexos com uso intensivo de dados.[A16]
- Permite aos utilizadores explorar dados de forma interativa e avaliar consultas em tempo real , de forma rápida e flexível . [A17]

Query:

```
json:serialize(doc('input.xml'), map { 'format': 'jsonml' })
```

Input XML

```
<address id='1'>
  <!-- comments will be discarded -->
  <last_name>Smith</last_name>
  <age>25</age>
  <address xmlns='will be dropped as well'>
    <street>21 2nd Street</street>
    <city>New York</city>
    <code>10021</code>
  </address>
  <phone type='home'>212 555-1234</phone>
</address>
```

Orador:

É um sistema que gera bases de dados XML , leve e de alto desempenho e um processador de XQuery . Foi desenvolvido como um projeto comunitário no GitHub . Tem como particularidades especializado em armazenar , consultar e visualizar grandes documentos e coleções XML e JSON . Este permite que os utilizadores explorem dados de forma interativa e avaliar consultas em tempo real . Além de dois modos locais autônomos, este sistema , oferece uma arquitetura cliente/servidor para lidar com operações simultâneas de leitura e gravação de vários utilizadores. Tem como linguagem de implementação o java , suportado também mais algumas linguagens como C, C#, python,... . Os seus principais recursos incluem a importação de grandes recursos únicos, bem como coleções de recursos em vários formatos, por ex. CSV, HTML e texto simples . Além disso, o BaseX fornece vários tipos de índices para melhorar o desempenho de operações de caminho, pesquisas de atributos, comparações de texto e pesquisas de texto completo.

CONTEÚDO:

BaseX é um mecanismo de bases de dados XML robusto e de alto desempenho e um processador XQuery 3.1 altamente

compatível com suporte total às extensões W3C Update e Full Text. Ele serve como uma excelente estrutura para a construção de aplicativos da web complexos com uso intensivo de dados.[A16]

Ele se concentra no armazenamento, consulta e visualização de grandes coleções e documentos XML e JSON. Um vista frontal permite aos utilizadores explorar dados de forma interativa e avaliar consultas em tempo real .

Além de dois modos locais autônomos, o BaseX oferece uma arquitetura cliente/servidor para lidar com operações simultâneas de leitura e gravação de vários utilizadores.

BaseX é independente de uma plataforma.[A17]

Os principais recursos do BaseX incluem a importação de grandes recursos únicos, bem como coleções de recursos em vários formatos, por ex. CSV, HTML e texto simples. BaseX pode ser operado a partir da linha de comando, no modo cliente/servidor ou usando a Interface Gráfica do Usuário (GUI). Além da funcionalidade principal do banco de dados, a GUI oferece front-ends gráficos inovadores para visualizar dados XML. Um editor XQuery avançado incorporado é acoplado às visualizações, fornecendo resultados instantâneos e feedback detalhado de erros.[A18]

Os operadores em Xbase são implementados como métodos Java seguindo uma convenção de nomes.

A linguagem de implementação da Base X é o Java , suportando também linguagens de programação como C,C#, Visual Basic, PHP, Python , JavaScript e muitas mais .[A21]

BaseX usa uma representação tabular de estruturas de árvore XML para armazenar documentos XML. A base de dados atua como um contentor para um único documento ou uma coleção de documentos. O esquema de codificação do XPath Accelerator e o Staircase Join Operator foram tomados como inspiração para acelerar as etapas de localização do XPath. Além disso, BaseX fornece vários tipos de índices para melhorar o desempenho de

operações de caminho, pesquisas de atributos, comparações de texto e pesquisas de texto completo. [A36]

[A14]- https://www.w3schools.com/xml/xquery_intro.asp

[A15]-

<https://files.basex.org/publications/Shadura%20%5B2012%5D,%20Input%20and%20Output%20with%20XQuery%20and%20XML%20Databases.pdf>

[A16]- <https://basex.org/>

[A17] - <https://www.linuxlinks.com/basex/>

[A18]-

<http://www.linuxtag.org/2011/de/program/themenschwerpunkte/system-management/vortragsetails-talkid383.html>

[A19]- <https://pt.wikipedia.org/wiki/XQuery>

[A21]- <https://db-engines.com/en/system/BaseX%3BTeradata>

[A36]- <https://en.wikipedia.org/wiki/BaseX>

Imagens[A27] - https://docs.basex.org/wiki/JSON_Module

O eXistdb (ou eXist, abreviadamente) é um projeto de software open source para bases de dados NoSQL construídas em tecnologia XML. É classificado como um sistema de bases de dados orientada a documentos NoSQL e uma base de dados XML nativo (e fornece suporte para documentos XML, JSON, HTML e binários). Ao contrário da maioria dos sistemas que gerem bases de dados relacional (RDBMS) e bases de dados NoSQL, o eXist-db fornece XQuery e XSLT como linguagens de consulta e programação de aplicações .

Principais características :

- Sendo uma solução completa, o eXistdb integra-se totalmente ao XForms para desenvolvimento de formulários complexos.
- Desenvolva aplicativos inteiros em XQuery usando o rico conjunto de bibliotecas do eXistdb.

ORADOR:

eXistDB é um projeto de software open source , foi construída para bases de dados nosql em formato XML. Também fornece suporte para documentos JSON,HTML . Fornece 2 principais tipos de linguagens de consulta , XQuery e XSLT(XSLT é uma linguagem de marcação XML usada para criar documentos XSL que, por sua vez, definem a apresentação dos documentos XML nos browsers e outros aplicativos que os suportem), Integrase totalmente ao Xforms para o desenvolvimento de formulários complexos . Tem um rico conjunto de bibliotecas , permitindo o desenvolvimento de aplicações inteiras em XQuery. As aplicações eXist-db são empacotadas como arquivos compactados únicos que são instalados diretamente na base de dados. A implantação, a atualização para novas versões e a distribuição tornam-se muito fáceis com eXistDB. Ja foi a melhor base de dados com formato XML em 2006.

CONTEÚDO:

eXist-db (ou eXist, abreviadamente) é um projeto de software open source para bases de dados NoSQL construídas em tecnologia XML. É classificado como um sistema de bases de dados orientada a

documentos NoSQL e uma base de dados XML nativo (e fornece suporte para documentos XML, JSON, HTML e binários). Ao contrário da maioria dos sistemas que gerem bases de dados relacional (RDBMS) e bases de dados NoSQL, o eXist-db fornece XQuery e XSLT como linguagens de consulta e programação de aplicativos.

eXist-db permite que desenvolvedores de software persistam documentos XML/JSON/Binários sem escrever middleware extenso. eXist-db segue e estende muitos padrões XML do W3C, como XQuery. eXist-db também suporta interfaces REST para interface com formulários web do tipo AJAX. Aplicativos como XForms podem salvar seus dados usando apenas algumas linhas de código. A interface WebDAV para eXist-db permite aos usuários "arrastar e soltar" arquivos XML diretamente na base de dados eXist-db. eXist-db indexa documentos automaticamente usando um sistema de indexação de palavras-chave. [A38]

Principais características :

Sendo uma solução completa, o eXistdb integra-se totalmente ao XForms para desenvolvimento de formulários complexos.

Desenvolva aplicativos inteiros em XQuery usando o rico conjunto de bibliotecas do eXistdb. [A26]

XSLT é uma linguagem de marcação XML usada para criar documentos XSL que, por sua vez, definem a apresentação dos documentos XML nos browsers e outros aplicativos que os suportem.[A38]

Os aplicativos eXist-db são empacotados como arquivos compactados únicos que são instalados diretamente na base de dados. A implantação, a atualização para novas versões e a distribuição tornam-se muito fáceis.

eXist-db é totalmente baseado em padrões abertos e código aberto, tornando-o uma escolha sustentável e preparada para o futuro.

Sendo Open Source desde 2001, o desenvolvimento do

eXist-db sempre foi impulsionado pelas necessidades de uma grande comunidade de usuários.[A40]

Curiosidade : eXist-db foi premiado como a melhor base de dados XML do ano pela InfoWorld em 2006.[A38]

[A26]-

<https://www.predictiveanalyticstoday.com/top-nosql-document-data-bases/>

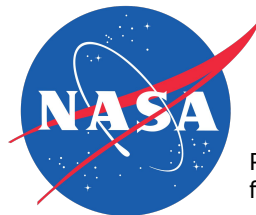
[A38]- <https://pt.wikipedia.org/wiki/XSLT>

[A40]- <https://exist-db.org/exist/apps/homepage/index.html>

RethinkDB



A RethinkDB é um sistema de base de dados que teve o seu lançamento em Junho de 2009 e tendo a sua versão mais recente em 2022. [D9] As suas principais vantagens são que as suas queries alteram a base de dados em tempo real, sendo isto útil para várias coisas desde jogos multiplayer a mercados online, pode ser utilizado com diversas linguagens, como Ruby, Python, Node.js e sendo open-source, está em permanente evolução e adaptação a necessidades novas. [D7]



Por exemplo, a NASA faz uso do RethinkDB

Conteúdo:

O RethinkDB é um sistema de gestão de base de dados de documentos distribuído, gratuito e de código aberto, originalmente criado pela empresa com o mesmo nome. A base de dados armazena documentos em formato JSON com esquemas dinâmicos e foi concebida para facilitar a transmissão de atualizações em tempo real dos resultados de consultas para aplicações. Inicialmente financiado pela Y Combinator em junho de 2009, a empresa anunciou em outubro de 2016 que não tinha conseguido estabelecer um negócio sustentável e que os seus produtos seriam disponibilizados no futuro como código aberto, sem suporte comercial. A CNCF (Cloud Native Computing Foundation) adquiriu então os direitos de código-fonte do RethinkDB e contribuiu para a Linux Foundation. O RethinkDB foi fundado em 2009 e tornou-se de código aberto na versão 1.2 em 2012. Em 2015, lançou a versão 2.0, declarando que estava pronta para uso em produção. Em 5 de outubro de 2016, a empresa anunciou que estava a encerrar as operações, transferindo membros da sua equipa de engenharia para a Stripe e deixando de oferecer suporte de produção. Em 6 de fevereiro de 2017, a Cloud Native Computing Foundation adquiriu os direitos do código-fonte e re-licenciou-o sob a Licença Apache 2.0. O RethinkDB utiliza a linguagem de consulta ReQL, uma linguagem de domínio interna (embutida) oficialmente disponível para Ruby, Python, Java e JavaScript (incluindo Node.js). Tem suporte para junções de tabelas, agrupamentos, agregações e funções. Existem também controladores não oficiais suportados pela comunidade para outras linguagens, incluindo C#, Clojure, Erlang, Go, Haskell, Lua e PHP. De acordo com a classificação DB-Engines, em fevereiro de 2016, era a 46ª base de dados mais popular. Em comparação com outras bases de dados de documentos, uma

característica distintiva do RethinkDB é o suporte de primeira classe para feeds de alterações em tempo real. Uma consulta de alteração retorna um cursor que permite solicitações bloqueantes ou não bloqueantes para acompanhar uma potencial corrente infinita de alterações em tempo real. [D9] O RethinkDB envia dados JSON em tempo real para as suas aplicações. Quando a sua aplicação faz sondagens para obter dados, torna-se lenta, não escalável e difícil de manter. O RethinkDB é a base de dados escalável de código aberto que torna a construção de aplicações em tempo real dramaticamente mais fácil. Trabalhe com a sua pilha de tecnologia favorita. Consulte documentos JSON com Python, Ruby, Node.js ou dezenas de outras linguagens. Construa aplicações modernas utilizando o seu framework web favorito, combinado com tecnologias em tempo real como Socket.io ou SignalR. Arquitetura robusta. Aplicações web e móveis reativas. Aplicações web como o Google Docs, Trello e Quora pioneirizaram a experiência em tempo real na web. Com o RethinkDB, pode construir incríveis aplicações em tempo real com muito menos esforço de engenharia. O RethinkDB integra os mais recentes avanços na tecnologia de bases de dados. Tem uma arquitetura distribuída moderna, uma cache de buffer altamente otimizada e um mecanismo de armazenamento de última geração. Todos esses componentes trabalham juntos para criar uma base de dados robusta, escalável e de alto desempenho. Tudo o que precisa para construir aplicações modernas. Expresse relações usando junções, construa aplicações sensíveis à localização ou armazene dados multimédia e de séries temporais. Faça análises com agregação e map/reduce e acelere as suas aplicações usando indexação flexível. Construído com amor pela comunidade de código aberto. Originalmente desenvolvido por uma equipa central de especialistas em bases de dados e mais de 100 contribuidores de todo o mundo, o RethinkDB é moldado por programadores como você que participam num processo de desenvolvimento de código aberto na comunidade. [D7]

[D7]: <https://rethinkdb.com/>

[D9]: <https://en.wikipedia.org/wiki/RethinkDB>

Linguagem ReQL [D8]



O RethinkDB utiliza como linguagem a ReQL, que manipula documentos no formato JSON. Ela tem três princípios como base, sendo estes: 1. Todas as queries podem ser chamadas em sequência 2.

Todas as queries são chamadas diretamente no servidor 3. As queries são chamadas diretamente na linguagem, descartando a necessidade de construir objetos especializados para ligar à base de dados.

```
# Get an iterable cursor to the 'users' table (we've seen this above)
r.table('users').run(conn)

# Return only the 'last_name' field of the documents
r.table('users').pluck('last_name').run(conn)

# Get all the distinct last names (remove duplicates)
r.table('users').pluck('last_name').distinct().run(conn)

# Count the number of distinct last names
r.table('users').pluck('last_name').distinct().count().run(conn)
```

Exemplo de comandos a serem sequenciados em ReQL .

Conteúdo:

O ReQL é a linguagem de consulta do RethinkDB. Oferece uma forma muito poderosa e conveniente de manipular documentos JSON. Este documento é uma introdução suave aos conceitos do ReQL. Não é necessário lê-lo para ser produtivo com o RethinkDB, mas ajuda a entender alguns conceitos básicos. O ReQL é diferente de outras linguagens de consulta NoSQL. Baseia-se em três princípios-chave: 1. O ReQL incorpora-se na sua linguagem de programação. As consultas são construídas fazendo chamadas de função na linguagem de programação que já conhece. Não é necessário concatenar strings ou construir objetos JSON especializados para consultar a base de dados. 2. Todas as consultas ReQL são encadeáveis. Começa com uma tabela e encadeia transformadores incrementalmente até o final da consulta usando o operador ".". 3. Todas as consultas são executadas no servidor. Enquanto as consultas são construídas no cliente numa linguagem de programação familiar, são executadas inteiramente no servidor de base de dados assim que chama o comando "run" e lhe passa uma ligação ativa à base de dados. O ReQL incorpora-se na sua linguagem de programação, começa-se a usar o ReQL no seu programa de forma semelhante à forma como usaria outras bases de dados, mas aqui a semelhança termina. Em vez de construir strings e enviá-las para o servidor de base de dados, acede ao ReQL usando métodos do pacote RethinkDB. Toda a consulta ReQL, desde filtros até atualizações e junções de tabelas, é feita chamando métodos apropriados. Esta abordagem tem as seguintes vantagens: Pode usar o mesmo ambiente de programação e ferramentas a que já está habituado, Aprender a linguagem não é diferente de aprender qualquer outra biblioteca, Há poucas ou nenhuma possibilidade de problemas de segurança causados por ataques

de injeção de strings. Quase todas as operações ReQL são encadeáveis. Pode pensar no operador "." de forma semelhante a um pipe Unix. Seleciona os dados da tabela e encaminha-os para um comando que os transforma. Pode continuar a encadear transformadores até que a sua consulta esteja completa. No ReQL, os dados fluem da esquerda para a direita. Mesmo se tiver um cluster de nós RethinkDB, pode enviar as suas consultas para qualquer nó, e o cluster criará e executará programas distribuídos que obtêm os dados dos nós relevantes, realizam os cálculos necessários e apresentam os resultados finais sem que precise preocupar-se com isso. Esta abordagem tem as seguintes vantagens: A linguagem é fácil de aprender, ler e modificar, é uma forma natural e conveniente de expressar consultas, pode construir consultas de forma incremental, encadeando transformações e examinando resultados intermediários. [D8]

[D8] - <https://rethinkdb.com/docs/introduction-to-reql/>

Oracle NoSQL



A Oracle NoSQL database é uma base de dados criada pela Oracle. Os seus principais destaques são as suas respostas rápidas, replicação rápida entre bases de dados [D13], a sua configuração elástica, que se refere à expansão dinâmica do cluster, onde o utilizador tem a possibilidade de manipular a topologia da base de dados (aumentando o espaço, ou redistribuindo os dados dentro da DB), a sua capacidade de atualização enquanto que a base continua em funcionamento e o facto de esta ser feita num client-server, baseado em partições onde os dados destas são passadas para cada nodo. [D14]

Conteúdo:

O Oracle NoSQL Database Cloud Service facilita o desenvolvimento de aplicativos para os desenvolvedores usando modelos de banco de dados de documento, esquema fixo e chave-valor, oferecendo tempos de resposta previsíveis de milissegundos de dígito único com replicação de dados para alta disponibilidade. O serviço oferece replicação regional ativo-ativo, transações ACID, dimensionamento sem servidor, segurança abrangente e preços baixos de acordo com o uso, tanto para modos de capacidade sob demanda quanto para capacidade provisionada, incluindo compatibilidade de 100% com o Oracle NoSQL Database local. [D13] O Oracle NoSQL Database é construído com base no mecanismo de armazenamento de alta disponibilidade Oracle Berkeley DB Java Edition. Ele adiciona serviços para oferecer um armazenamento de chave/valor distribuído e altamente disponível, adequado para aplicativos com grandes volumes de dados e baixa latência. O sistema é composto por uma arquitetura cliente-servidor, é particionado e segue o princípio de compartilhamento de recursos. Cada partição, conhecida como "shard", possui dados replicados em todos os nós que fazem parte dela. O mecanismo usa um algoritmo de hash para determinar a qual shard um determinado registro pertence. É projetado para permitir a alteração dinâmica do número de shards, sem a necessidade de interrupção do sistema. Se o número de shards muda, os pares de chave-valor são redistribuídos dinamicamente para as novas shards, tudo isso sem a necessidade de desligar e reiniciar o sistema. Cada shard é composta por um nó mestre eleito para atender a solicitações de leitura e escrita, bem como por vários nós réplicas (geralmente dois ou mais) que podem atender a solicitações de leitura. As réplicas são mantidas atualizadas por meio de replicação em tempo real. Cada

alteração feita no nó mestre é registrada localmente e propagada para as réplicas. O Oracle NoSQL Database oferece alta disponibilidade e tolerância a falhas por meio da replicação de banco de dados com um único mestre e várias réplicas. Os dados transacionais são entregues a todos os nós réplicas com políticas de durabilidade flexíveis por transação. Em caso de falha do nó mestre, é acionado um processo automatizado de failover baseado em PAXOS para minimizar o tempo de inatividade. Assim que o nó mestre falho é reparado, ele retorna à shard, atualizado e pronto para processar solicitações de leitura. A arquitetura também se preocupa com o balanceamento de carga transparente, com a partição de dados em tempo real e distribuição uniforme entre os nós de armazenamento, considerando a topologia de rede e a latência para otimizar a distribuição de carga e desempenho. A administração e o monitoramento do Oracle NoSQL Database podem ser realizados por meio de uma interface web ou uma interface de linha de comando. Esses serviços suportam funções como configuração, inicialização, desligamento e monitoramento de nós de armazenamento, tudo sem a necessidade de arquivos de configuração ou scripts. Além disso, suporta Java Management Extensions (JMX) ou agentes Simple Network Management Protocol (SNMP) para monitoramento. O Oracle NoSQL Database é altamente configurável, permitindo a expansão dinâmica do cluster. Isso inclui adicionar nós de armazenamento para aumentar a capacidade, o desempenho e a confiabilidade. A topologia do cluster pode ser ajustada enquanto o banco de dados está online, aumentando a distribuição de dados, o fator de replicação e o equilíbrio do armazenamento, conforme necessário. O sistema suporta implantações em várias zonas para melhorar a confiabilidade em cenários de falha de hardware, rede e energia. Isso é feito por meio de zonas primárias, que contêm nós mestres e réplicas conectados por interconexões de alta velocidade, e zonas secundárias, onde os nós servem apenas como réplicas, permitindo acesso de leitura com baixa latência ou a execução de cargas de trabalho somente leitura, como análises, geração de relatórios e troca de dados. O formato de dados do Oracle NoSQL Database é baseado em JSON e oferece suporte à serialização de dados Avro. Os esquemas são definidos em JSON e a evolução de esquemas é suportada. O sistema utiliza uma "Smart Topology" configurável, permitindo que os administradores aloquem inteligentemente os nós de replicação com base na capacidade disponível em cada nó de armazenamento. Isso é feito para um melhor equilíbrio de carga e uso eficiente de recursos do sistema, minimizando o impacto em caso de falha de nós de armazenamento. O Smart Topology também suporta data centers, garantindo que um conjunto completo de réplicas seja alocado inicialmente para cada data center. O Oracle NoSQL Database oferece a capacidade de realizar atualizações de sistema em linha, permitindo que um administrador atualize os nós do cluster enquanto o banco de dados permanece disponível. O sistema é configurável para atender aos critérios CAP (Consistency, Availability, Partition Tolerance). Se as gravações são configuradas para serem executadas de forma síncrona em todas as réplicas, o sistema atende ao critério C/P, ou seja, uma falha de partição ou nó torna o sistema indisponível para gravações. Se a replicação for executada de forma assíncrona e as leituras forem configuradas para serem atendidas por qualquer réplica, o sistema atende ao critério A/P, o que significa que o

sistema está sempre disponível, mas não há garantia de consistência. Entre os recursos do banco de dados, está o modelo de dados de tabela, que simplifica a modelagem de dados de aplicativos. Isso é obtido por meio de uma estrutura tabular que se baseia na estrutura distribuída de chave/valor, herdando suas vantagens e simplificando o design de aplicativos ao permitir a integração com aplicativos SQL. O Oracle NoSQL Database também oferece suporte a índices secundários, permitindo caminhos de acesso de baixa latência que não se baseiam apenas na chave primária. Além disso, suporta objetos grandes (LOBs) para leitura e gravação de arquivos de áudio e vídeo sem a necessidade de materializar o arquivo inteiro na memória, o que ajuda a reduzir a latência de operações em cargas de trabalho com objetos de tamanhos variados. Transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade) são fornecidas para operações completas de criação, leitura, atualização e exclusão (CRUD), com garantias ajustáveis de durabilidade e consistência. Um conjunto de operações pode ser tratado como uma única unidade atômica, desde que todos os registros afetados compartilhem o mesmo caminho de chave principal. A integração é suportada por meio de APIs em Java, C, Python, C# e REST, permitindo que os desenvolvedores de aplicativos realizem operações de criação, leitura, atualização e exclusão. Essas bibliotecas também incluem suporte a Avro, permitindo que os desenvolvedores serializem e deserializem registros de chave/valor de forma intercambiável entre aplicativos em C e Java. [D14]

[D13] - <https://www.oracle.com/database/nosql/#rc30p1>

[D14] - https://en.wikipedia.org/wiki/Oracle_NoSQL_Database

InterSystems Caché/IRIS



A bases de dados Caché, e a sua “sequência” IRIS, bases de dados da empresa InterSystems, são bases de dados de objetos de alta performance e que podem escalar de forma massiva [D10][D11], sendo capazes de também criar bases de dados documents, sendo estas chamadas de DocDB. Estas bases de dados são baseadas em JSON, sendo o principal destaque o facto de ser uma database sem esquema pré-definido, isto é, cada documento tem estrutura única, o que pode diferir de outros documentos da mesma base de dados, sendo uma diferença para o SQL normal, que precisa de uma estrutura pré-definida. [D12]

Conteúdo:

O Caché é projetado para superar as limitações do modelo relacional, ao mesmo tempo que oferece uma trajetória de atualização evolutiva para milhares de aplicativos de banco de dados relacionais existentes, bem como suporte para muitas ferramentas de relatórios baseadas em SQL disponíveis no mercado. Além de ser um banco de dados de objetos de alto desempenho, o Caché também é um banco de dados relacional completo. Todos os dados dentro de um banco de dados Caché estão disponíveis como tabelas relacionais verdadeiras e podem ser consultados e modificados usando SQL padrão via ODBC, JDBC ou métodos de objeto. Devido ao poder do mecanismo subjacente do banco de dados Caché, acreditamos que o Caché é o banco de dados relacional mais rápido, confiável e escalável disponível hoje. [D11] O InterSystems IRIS é uma plataforma de dados unificada que apresenta um banco de dados altamente escalável, análises integradas, ferramentas de integração e interoperabilidade, além de mecanismos automatizados para implantação fácil. No cerne disso, o banco de dados InterSystems IRIS é tanto altamente eficiente quanto incrivelmente escalável. Com estruturas de dados efetivas e exclusivas para velocidade e flexibilidade, juntamente com recursos intuitivos que permitem dimensionar seus sistemas vertical e horizontalmente, não há problema grande demais para o InterSystems IRIS lidar. [D10] A plataforma de dados InterSystems IRIS® DocDB é uma instalação para armazenar e recuperar dados de banco de dados. É compatível com, mas separado do armazenamento e recuperação tradicional de dados de tabelas e campos SQL (classe e propriedade). É baseado no JSON (JavaScript Object Notation), o que oferece suporte para a troca de dados baseada na web. O InterSystems IRIS fornece suporte para o

desenvolvimento de bancos de dados e aplicativos do DocDB em REST e ObjectScript, além de oferecer suporte SQL para criar ou consultar dados do DocDB. Por sua natureza, o Document Database do InterSystems IRIS é uma estrutura de dados sem esquema. Isso significa que cada documento tem sua própria estrutura, que pode ser diferente de outros documentos no mesmo banco de dados. Isso tem várias vantagens quando comparado ao SQL, que requer uma estrutura de dados pré-definida. [D12]

[D10] - <https://learning.intersystems.com/course/view.php?name=InterSystemsIRIS>

[D11] -

https://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GIC_intro

[D12] -

https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=GD OCDB_intro#:~:text=By%20its%20nature%2C%20InterSystems%20IRIS,a%20pre%2Ddefined%20data%20structure.

Características/ Vantagens do IS Caché/IRIS [D12]



- Como mencionado no slide anterior, a não existência de um esquema pré-definido, o que permite à IRIS ser das mais rápidas databases a capturar informações, já que esta captura não é rendida por estruturas.
- São bases de dados muito eficientes a captar dados pequenos, já que não criam espaços vazios em outros documents se houver informação extra em alguns deles
- A hierarquia pode ser guardada de maneira desnormalizada, isto é, não precisa de ser guardada em várias tabelas
- Chaves podem ter os seus tipos de dados alterados, por exemplo uma chave X pode ser associada a um tipo de data em um document e noutro document a um tipo diferente

Conteúdo:

Algumas das principais características do InterSystems IRIS DocDB incluem:

Flexibilidade da Aplicação: Documentos não requerem um esquema pré-definido.

Isso permite que as aplicações configurem rapidamente seus ambientes de dados e se adaptem facilmente a mudanças na estrutura de dados. Isso permite a captura rápida de dados. O Document Database pode começar a capturar dados

imediatamente, sem a necessidade de definir uma estrutura para esses dados. Isso é ideal para fontes de dados imprevisíveis, como as encontradas frequentemente em fontes de dados baseadas na web e em mídias sociais. Se, ao capturar um conjunto de dados, estruturas dentro desses dados se tornarem evidentes ou surgirem como úteis, a estrutura de dados do documento pode evoluir. Os dados capturados existentes podem coexistir com essa representação mais estruturada dos dados.

Cabe à sua aplicação determinar a estrutura de dados para cada documento e processá-la adequadamente. Uma maneira de fazer isso é estabelecer um par chave:valor que represente a versão da estrutura do documento. Assim, a conversão de dados de uma estrutura JSON para outra pode ser realizada gradualmente, sem interromper a captura ou acesso de dados, ou a conversão de dados não pode ser realizada. **Eficiência de Dados Esparsos:** Documentos são muito eficientes para armazenar dados esparsos, pois atributos com uma chave específica podem aparecer em alguns documentos de uma coleção, mas não em outros. Um documento pode ter um conjunto de chaves definidas; outro documento na mesma coleção pode ter um conjunto muito diferente de chaves definidas. Em contraste, o SQL exige que cada registro contenha todas as chaves; em dados esparsos, muitos registros têm chaves com valores NULL. Por exemplo, um registro médico de

paciente em SQL fornece campos para muitos diagnósticos, condições e testes; para a maioria dos pacientes, a maioria desses campos está em branco. O sistema aloca espaço para todos esses campos não utilizados. Em um registro médico de paciente do DocDB, somente as chaves que contêm dados reais estão presentes.

Armazenamento de Dados Hierárquicos: O DocDB é muito eficiente para armazenar dados estruturados hierarquicamente. Em um par chave:valor, os dados podem ser aninhados dentro dos dados a um número ilimitado de níveis. Isso significa que dados hierárquicos podem ser armazenados desnormalizados. No modelo relacional SQL, os dados hierárquicos são armazenados normalizados usando várias tabelas.

Tipos de Dados Dinâmicos: Uma chave não possui um tipo de dados definido. O valor atribuído à chave possui um tipo de dados associado. Portanto, um par chave:valor em um documento pode ter um tipo de dados, e um par chave:valor para a mesma chave em outro documento pode ter um tipo de dados diferente. Como os tipos de dados não são fixos, você pode alterar o tipo de dados de um par chave:valor em um documento em tempo de execução atribuindo um novo valor que tenha um tipo de dados diferente. Essas características do DocDB têm importantes implicações para o desenvolvimento de aplicativos. Em um ambiente SQL tradicional, o design do banco de dados estabelece a estrutura de dados que é seguida no desenvolvimento de aplicativos. No DocDB, a estrutura de dados é amplamente fornecida nas próprias aplicações. [D12]

[D12] -

https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=GD OCDB_intro#:~:text=By%20its%20nature%2C%20InterSystems%20IRIS,a%20pre%2Ddefined%20data%20structure.

SGBD NoSQL - Documents

Os seis SGBD NoSQL Documents mais usados são [C1][C2]:

1º-MongoDB



2º- Amazon DynamoDB



3º- Databricks



4º- MSFT Azure Cosmos DB



5º- Couchbase



6º- Firebase Realtime Database



Orador:

Boa tarde a todos,

Estes são os seis SGBD NoSQL Documents mais usados, sendo que muitas destes já foram referidos e explicadas anteriormente.

Esta classificação que podemos ver aqui está presente na página DB-Engines Ranking no qual este ranking é feito tendo em atenção vários pontos: menções ao sistema em sites, frequência de pesquisa no Google Trends, número de ofertas de emprego em que o sistema é mencionado e entre muitas mais.

CONTEÚDO:

Segundo o site DB-Engines Ranking, a classificação dos SGBD NoSQL Document é feita e atualizada mensalmente de acordo com a sua popularidade.

usando os seguintes parâmetros: Número de menções ao sistema em sites, medido como o número de resultados em consultas em motores de busca. Atualmente são usados o Google e o Bing para esta medição. Para contar apenas os resultados relevantes, é procurado por <nome do sistema> junto com o termo base de dados, por exemplo. "Oracle" e "base de dados"; Interesse geral no sistema(SGBD). Para esta medição, é utilizada a frequência de pesquisas no Google Trends; Número de discussões técnicas sobre o sistema. É usado o número de perguntas efetuadas e o número de utilizadores interessados no sistema(SGBD) nos sites Stack Overflow e DBA Stack Exchange.; Número de ofertas de emprego em que o sistema é mencionado. Utilizamos o número de vagas nos motores de busca de emprego Even e Simply Hired; Número de perfis em redes profissionais em que o sistema é mencionado. É utilizada a rede profissional internacional mais popular, o LinkedIn; Relevância nas redes sociais. É contado o número de tweets do Twitter (X) em que o sistema é mencionado; O valor de popularidade de um sistema é calculado padronizando e calculando a média de parâmetros individuais. Estas transformações matemáticas são feitas de forma que a distância dos sistemas individuais seja preservada. Ou seja, quando o sistema A tem um valor duas vezes mais alto no DB-Engines Ranking que o sistema B, ele é duas vezes mais popular quando a média dos critérios de avaliação individuais é calculada.;Para eliminar os efeitos causados pela alteração das quantidades das próprias fontes de dados, a pontuação de popularidade é sempre um valor relativo, que só deve ser interpretado em comparação com outros sistemas; O DB-Engines Ranking não mede o número de instalações de sistemas, nem seu uso em sistemas de TI.

[C1]: <https://db-engines.com/en/ranking/document+store>

[C2]: https://db-engines.com/en/ranking_definition

Comparativo geral

	Open Source	Serviço na nuvem	Casos de uso	Linguagem Query
MongoDB	Sim	Não	IA, Edge computing, internet das coisas, aplicações móveis, sistemas de pagamentos, jogos... [C3]	MongoDB Query language(MQL)[C9]
Amazon DynamoDB	Não	Sim	Retalho, finanças e bancos, internet e software [C4]	PartiQL (compatível com SQL)[C10]
Databricks	Sim	Sim	IA, streaming, data warehousing... [C5]	Databricks SQL[C11]
MSFT Azure Cosmos DB	Não	Sim	Internet das coisas, retalho, jogos, aplicações web e móveis...[C6]	SQL[C12]
Couchbase	Sim	Não	Internet das coisas, inventário de produtos, jogos... [C7]	SQL++ [C13]
Firebase Realtime Database	Não	Sim	Sincronização em tempo real (ex. uber)[C8]	NoSQL

C - Leonardo Ferreira - 2122422

131

ORADOR:

Neste slide vemos uma tabela que faz um comparativo geral dos seis SGBD visto anteriormente.

Destes seis, três são Open Source sendo eles MongoDB, Databricks e Couchbase. Quatro são serviços na nuvem que são o Amazon DynamoDB, Databricks, Azure Cosmos DB e o Firebase Realtime Database. O serviço na nuvem quer dizer que é um serviço (software) disponibilizado online.

Temos também, aqui, os casos de uso mais usuais de cada um destes SGBD. Por exemplo, os casos de uso do MongoDB são: IA, internet das coisas, aplicações móveis, jogos, entre outros.

Temos ainda a linguagem de query de cada um deles. Podemos ver que por exemplo a linguagem utilizada para efetuar consultas de dados no Cosmos DB é o SQL.

CONTEÚDO:

iniciativas estratégicas para o negócio às suas aplicações mais importantes.

Casos de uso: Inteligência artificial- agilizar a construção de aplicações enriquecidas com IA; Edge computing- desbloquear os benefícios do “edge computing” simplificando a gestão de dados; Internet das coisas- Analisar e actuar em dados do mundo físico; Móvel- Desenvolvimento de aplicações móveis tornado fácil e rápido; Pagamentos- É crítico modernizar a sua arquitetura de pagamentos; Desenvolvimento serverless- desenvolva aplicações serverless de qualquer escala: Vista única (single view)- vistas em tempo real de todos os seus dados mais importantes; Personalização- conteúdo relevante apresentado a todos os seus utilizadores; Catálogo- catálogos de produtos, gestão de ativos e mais; Gestão de conteúdos- Guarde, edite e apresente todo o tipo de conteúdo; Modernização dos Mainframe (servidores)- mover cargas de trabalho do mainframe (servidor principal); Jogos- video jogos que são globais, confiáveis e escaláveis.

Linguagem Query [C9]:

Consultando coleções do MongoDB. A linguagem de consulta MongoDB (MQL) usa a mesma sintaxe dos documentos, tornando-a intuitiva e fácil de usar até mesmo para consultas avançadas.

Casos de uso Amazon DynamoDB [C4]: Retalho- As empresas de retalho usam o DynamoDB para oferecer baixa latência para casos de uso de missão crítica. O aumento e a redução permitem que esses clientes paguem apenas pela capacidade de que precisam e mantêm recursos técnicos preciosos focados em inovações e não em operações; Banca e finanças- As empresas bancárias e financeiras usam o DynamoDB para aumentar a agilidade, reduzir o tempo de lançamento no mercado e minimizar a sobrecarga operacional, mantendo a segurança e a confiabilidade das suas aplicações; Software e internet- O DynamoDB tem um histórico como o melhor de

capacidade de lidar com casos de uso de Internet e dos seus requisitos, mantendo uma latência consistente de milissegundos (de um dígito). Com tabelas globais, os clientes do DynamoDB podem expandir facilmente as suas aplicações para diversas regiões da AWS para obter exposição global e continuidade de negócio.

Linguagem Query (Consulta) [C10]:

PartiQL – uma linguagem de consulta compatível com SQL para Amazon DynamoDB

O Amazon DynamoDB oferece suporte a PartiQL, uma linguagem de consulta compatível com SQL, para selecionar, inserir, atualizar e remover dados no Amazon DynamoDB. Utilizando o PartiQL, podemos interagir facilmente com tabelas do DynamoDB e executar consultas ad hoc usando o AWS Management Console, o NoSQL Workbench, a AWS Command Line Interface e as APIs do DynamoDB para PartiQL.

As operações do PartiQL fornecem a mesma disponibilidade, latência e desempenho que as outras operações do DynamoDB.

Casos de uso Databricks[C5]: Partilha de dados Engenharia de dados; Gestão de dados; Armazenamento de dados; Inteligência artificial; Data science; Streaming em tempo real; Mercado.

Linguagem Query (Consulta) [C11]:

O que é armazenamento de dados no Databricks? A plataforma Databricks Lakehouse fornece uma solução de armazenamento de dados completa. A plataforma Databricks Lakehouse é baseada em padrões abertos e APIs. O Databricks Lakehouse combina as transações ACID e a gestão dos data warehouses empresariais com a flexibilidade e a eficiência de custos dos data lakes. O Databricks SQL descreve o data warehouse empresarial integrado à plataforma Databricks Lakehouse que fornece uma experiência única para analisar e agir sobre os dados.

Casos de uso Azure Cosmos DB [C6]: IoT e telemática- Os casos de uso de IoT geralmente partilham alguns padrões de como ingerem, processam e armazenam dados. Primeiro, esses sistemas precisam ingerir rajadas de dados de sensores de dispositivos de vários locais. Em seguida, esses sistemas processam e analisam dados de streaming para obter insights em tempo real. Os dados são então arquivados em armazenamento frio para análise em lote; Retalho e marketing- O Azure Cosmos DB é amplamente utilizado em plataformas de comércio eletrônico da Microsoft que executam a Windows Store e Xbox Live. Também é usado no setor retalho para armazenar dados de catálogos e processamento de pedido; Jogos- A camada de bases de dados é um componente crucial dos jogos. Os jogos modernos realizam processamento gráfico em clientes móveis/consolas, mas dependem da nuvem para fornecer conteúdo customizado e personalizado, como estatísticas do jogo, integração de redes sociais e placares de classificação com pontuações mais altas. Os jogos geralmente exigem latências de um milissegundo para leituras e gravações de forma a fornecer uma experiência envolvente no jogo. Uma base de dados de jogos precisa de ser rápido e capaz de lidar com picos massivos nas taxas de solicitação durante o lançamento de novos jogos e atualizações de recursos; Aplicações web e móveis- O Azure Cosmos DB é frequentemente usado em aplicações Web e móveis e é adequado para modelar interações sociais, integrar-se a serviços de terceiros e criar experiências personalizadas avançadas.

Linguagem Query (Consulta) [C12]: SQL

Casos de uso Couchbase [C7]: Cliente 360- agregar dados na plataforma obtidos de diferentes fontes para construir uma visão única dos seus clientes ou negócio; Catálogo e inventário- Publique novos conteúdos de produtos e inventário em tempo real e dimensione milhões de produtos e solicitações

por segundo para apresentar os dados certos no momento certo; Trabalho de campo- Disponibiliza aos funcionários de campo uma plataforma para gerir dados de diferentes fontes, levar esses dados até o limite e garantir que os dados estejam disponíveis online e offline; Gestão de dados IoT- Gira, suporte e gere insights de dados em tempo real com bases de dados integrados e na nuvem, sincronização e disponibilidade garantida de dados.

Linguagem Query (Consulta) [C13]: O Couchbase Server pode ser consultado usando SQL++, a linguagem de consulta do Couchbase Server.

Casos de uso Firebase Realtime Database[C8]: Sincronização em tempo real

Linguagem Query (Consulta) [C14]:

O documento explica os conceitos básicos sobre recuperação, ordenação e consultas simples de dados (base de dados) sendo NoSQL. Disponibiliza duas formas: **Listeners assíncronos- Este modo é compatível** com SDKs Admin para Java, Node.js e Python; **Leituras de método de bloqueio-** Este modelo é compatível com o SDKs Admin para Python e Go.

[C3]: <https://www.mongodb.com/solutions/use-cases>

[C4]:

https://aws.amazon.com/pm/dynamodb/?trk=bf64c969-685f-4fc4-b36b-4bcbda56cee7&sc_channel=ps&ef_id=Cj0KCQjwm66pBhDQARIsALIR2zBAaHSPe-cm699UVuPZkKmPud6h_10OoNeIxJnoYR3z1k4SFSlt_I0aAlMqEALw_wcB:G:s&s_kwcid=AL!4422!3!536324221413!p!!g!!aws%20dynamodb!12195830303!119606857400

[C8]: <https://firebase.google.com/docs/database>

[C9]: <https://www.mongodb.com/basics/examples>

[C10]:
<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-reference.html>

[C11]: <https://docs.databricks.com/en/sql/index.html>

[C12]:
<https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/tutorial-query>

[C13]:
<https://docs.couchbase.com/server/current/n1ql/query.html#:~:text=Couchbase%20Server%20can%20be%20queried,form%20of%20the%20data%20model.>

[C14]:
<https://chat.google.com/dm/9762XgAAAAE/Es2OQpUE7a8/Es2OQpUE7a8?cls=10>

MongoDB VS DynamoDB

Instalação ou serviço?

MongoDB [C15][C19]- hospedado localmente ou numa nuvem pública ou privada:

- MongoDB Atlas- serviço totalmente gerido;
- MongoDB Enterprise- versão autogerida e paga;
- MongoDB Community- versão autogerida e gratuita.

DynamoDB [C16][C20]- Só pode ser executado na nuvem AWS. É totalmente gerido, oferece:

- Rápido desempenho;
- Elevada escalabilidade.

Tipos de Dados

MongoDB [C17][C21]: Formato BSON.

- Scalar Types;
- Document Types;
- Set Types;
- Date

DynamoDB [C17][C22]- Formato JSON:

- Scalar Types;
- Document Types;
- Set Types;

ORADOR:

Nos dois slides seguintes irei fazer uma pequena comparação entre os dois SGBD NoSQL Document mais utilizados (MongoDB e DynamoDB).

O MongoDB, de acordo com a versão, poderá ser hospedado localmente ou numa nuvem pública ou privada enquanto que o DynamoDB só pode ser executado na nuvem AWS (Amazon). No DynamoDB é totalmente gerido, Isso significa que os clientes não precisam se preocupar com a configuração, manutenção, atualizações ou qualquer aspecto operacional da solução. Oferece um rápido desempenho e uma elevada escalabilidade.

Em relação ao tipo de dados, o MongoDB usa o formato BSON que quer dizer "Binary JSON", este adiciona alguns tipos de dados que o formato JSON não possui entre eles o Date. E o DynamoDB usa o formato JSON.

Instalação ou serviço?

Podem ser criadas bases dados MongoDB nos seguintes ambientes [C19]: MongoDB Atlas- O serviço totalmente gerido para implementações (instalação) do MongoDB na nuvem; MongoDB Enterprise- Versão autogerida e baseada em assinatura do MongoDB e MongoDB Community – Versão MongoDB autogerida, disponível e de uso gratuito.

[C20]

O Amazon DynamoDB é um serviço de bases de dados NoSQL totalmente gerido que oferece rápido desempenho com uma elevada escalabilidade . O DynamoDB liberta-nos das tarefas administrativas de operar uma base dados distribuída e o seu dimensionamento pelo que não é necessário nos preocupar com o hardware, instalação e configuração, instalação de updates de software e dimensionamento de um cluster, entre outros. O DynamoDB distribui automaticamente os dados e o tráfego por um determinado número de servidores de forma a manter um desempenho consistente e rápido de acordo com requisitos da aplicação. Todos os dados são armazenados em discos SSD e replicados automaticamente para várias zonas de disponibilidade de uma zona AWS, proporcionando alta disponibilidade e durabilidade dos dados.

[C15]

MongoDB[C15]: Pode ser hospedado localmente ou em qualquer nuvem pública ou privada. DynamoDB[C16]: DynamoDB: Só pode ser executado na nuvem AWS, pelo que não é a opção ideal para utilizadores que não trabalham com serviços AWS. Sem opções locais.

Tipos de dados MongoDB[C21]:

O servidor MongoDB armazena dados usando o formato BSON que suporta alguns tipos de dados adicionais que não estão disponíveis quando os dados são formatados JSON.

atributos de uma tabela. Podem ser organizados da seguinte forma:

- **Scalar Types:** Number; String; Binary; Boolean; Null
- **Document Types:** List; Map
- **Set Types:** String Set; Number Set; Binary Set

[C17]

- MongoDB- Qualquer tipo de dados na especificação BSON, incluindo datas. O DynamoDB- A maioria dos tipos de dados padrão, excluindo datas

[C15]:

<https://www.mongodb.com/compare/mongodb-dynamodb>

[C16]: <https://www.educative.io/blog/mongo-db-vs-dynamo-db-nosql-databases>

[C17]: <https://kinsta.com/blog/dynamodb-vs-mongodb/>

[C19]: <https://www.mongodb.com/docs/manual/introduction/>

[C20]: [What is Amazon DynamoDB? - Amazon DynamoDB](#)

[C21]: [Data Types — MongoDB Shell](#)

[C22]: [Supported data types and naming rules in Amazon DynamoDB - Amazon DynamoDB](#)

MongoDB VS DynamoDB

Índices

Os índices têm como função aumentar a velocidade de pesquisa

MongoDB:

- Cria índice `_id` por defeito[C23];
- Máximo de 64 índices por coleção [C18][C24].

DynamoDB [C18][C25][C26]:

- Usa chaves primárias (Partition key and sort key);
- Máximo de 20 índices secundários.

Queries[C15]

MongoDB:

- Consultas por chaves únicas;
- Pesquisas detalhadas;
- Consultas geoespaciais.
- ...

DynamoDB:

- A query só pode ser feita através da chave primária[C27];
- Alguns tipos de pesquisa implicam a replicação de dados para outros serviços AWS;
- Chave primária pode ter dois atributos dificultando a flexibilidade da pesquisa.

ORADOR:

Os índices têm como função aumentar a velocidade de pesquisa. O MongoDB cria por defeito o índice `_id` e permite no máximo 64 índices por coleção. O DynamoDB usa chaves primárias e tem um máximo de 20 índices secundários.

Em relação às queries o MongoDB permite consultas por chaves únicas, pesquisas detalhadas, consultas geoespaciais, etc. No DynamoDB a query só pode ser feita através da chave primária, a chave primária pode ter dois atributos dificultando a flexibilidade da pesquisa e alguns tipos de pesquisa implicam a replicação de dados para outros serviços AWS.

CONTEÚDO:

Índices MongoDB [C23] :

Os índices suportam a execução eficiente de consultas no MongoDB. Sem índices, o MongoDB deverá verificar todos os

consulta, o MongoDB usa esse índice de forma a limitar o número de documentos que deve verificar.

Embora os índices melhorem o desempenho da consulta, adicionar um índice tem um impacto negativo no desempenho das operações de gravação. Para coleções com uma alta proporção de gravação leitura, os índices poderão tornar a operação mais lenta porque cada inserção deverá atualizar os índices.

Restrições:

São aplicadas algumas restrições aos índices, como o comprimento das chaves de índice ou o número de índices por coleção.

Default index:

O MongoDB cria um índice único no campo `_id` durante a criação de uma coleção. O índice `_id` impede que os clientes insiram dois documentos com o mesmo valor para o campo `_id`. Não é possível apagar este índice.

Index Build Performance

As aplicações podem ter uma queda no desempenho durante a criação de índices, incluindo acesso limitado de leitura/gravação à coleção.

Número de índices por coleção [C24]:

Uma única coleção não pode ter mais de 64 índices.

Número de índices DynamoDB [C25]:

Existe um limite de 20 índices secundários globais por tabela.

[C26]

No DynamoDB, os principais componentes são tabelas, itens e atributos. Uma tabela é uma coleção de itens e cada item é

unicamente cada item na tabela.

Pode ser de dois tipos: Partition key- um atributo; Partition key and sort key- A chave primária é constituída por dois atributos, o primeiro é o Partition key e o segundo sort key.

[C18]

O MongoDB suporta até 64 índices mutáveis por coleção, permitindo que a estrutura do documento mude dinamicamente. O DynamoDB oferece suporte até 20 índices globais mutáveis por tabela, que não são compatíveis com os dados subjacentes, e até 5 índices locais que não podem ser modificados após a criação da tabela.

[C27]

Uma query é uma pesquisa baseada num determinado critério. A primary key indica a localização dos dados na tabela DynamoDB, de forma podermos usar os atributos dessa primary key para efetuar pesquisas à base dados.

A primary key pode ser de dois tipos: Chave primária simples (simple primary key)- partition key; chave primária composta (composite primary key)- dois atributos: partition key e sort key.

Podemos efetuar pesquisas de duas formas: usando a partition key e usando a partition key e filtrando pela sort key

[C15]:

<https://www.mongodb.com/compare/mongodb-dynamodb>

[C18]:<https://rockset.com/blog/mongodb-vs-dynamodb-head-to-head-which-should-you-choose/>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html>

[C27]:

<https://www.educative.io/answers/how-to-query-data-in-dynamodb>

MSFT Azure Cosmos DB VS Firebase Realtime Database

Instalação ou serviço?[C28][C33]

Quer o MSFT Azure Cosmos DB quer Firebase Realtime Database são hospedados na nuvem sendo serviços totalmente geridos.

Tipos de dados

MSFT Azure Cosmos DB[C29]:

Fornecer suporte aos tipos NoSQL document (JSON) e MongoDB (BSON). São formatos diferentes mas oferecem modelagem de dados semelhantes.

Firebase Realtime Database[C30][C34]:

Usa o formato JSON e é sincronizado em tempo real para cada cliente conectado.

ORADOR:

Nestes slides faço a comparação entre o Azure Cosmos DB e o Firebase Realtime Database.

Ambos são hospedados na nuvem, sendo serviços totalmente geridos.

O Azure Cosmos DB permite dois tipos de NoSQL Document, JSON se for uma base de dados NoSQL Document ou BSON se for escolhida o tipo MongoDB. Estes dois formatos são diferentes mas oferecem modelagem de dados semelhantes. O Firebase usa o formato JSON e é sincronizado em tempo real para cada cliente conectado.

CONTEÚDO:

Quer o MSFT Azure Cosmos DB quer Firebase Realtime Database são hospedados na nuvem. [C28]

O Azure Cosmos DB é uma base de dados NoSQL e relaciona

milissegundos de um dígito, escalabilidade automática e instantânea, além de velocidade garantida em qualquer escala. A continuidade dos negócios é garantida com disponibilidade apoiada por SLA e a segurança de nível empresarial.

Como um serviço totalmente gerido, o Azure Cosmos DB tira a administração da base de dados das mãos do utilizador, com gestão, atualizações e patches automáticos, também lida com a gestão da capacidade com opções económicas de escalonamento automático e serverless que respondem às necessidades das aplicações [C33].

O modelo MSFT Azure Cosmos DB oferece suporte a API SQL e APIs MongoDB. Estas duas APIs fornecem o modelo de dados documento. Estas APIs são diferentes uma da outra, embora sejam semelhantes na modelagem de dados. A API SQL permite construir stored procedures e funções definidas pelo utilizador. A API SQL armazena entidades em JSON num documento hierárquico de key value. A API MongoDB armazena em BSON (versão codificada em binário do JSON, que estende o JSON com tipos de dados adicionais e suporte multilingua) [C29].

O Firebase Realtime Database é uma base de dados que está hospedada na nuvem. Os dados são armazenados no formato JSON e sincronizados em tempo real para cada cliente conectado.

O Realtime Database é uma base de dados NoSQL e, por isso, tem otimizações e funcionalidades diferentes de uma base de dados relacional. A API do Realtime Database foi desenvolvida para autorizar apenas operações que possam ser executadas com rapidez. Isso possibilita uma ótima experiência em tempo real que atende a milhões de utilizadores sem comprometer a capacidade de resposta. Por isso, é importante analisar como é que os utilizadores precisam aceder aos dados e estruturá-los adequadamente[C30].

Todos os dados do Firebase Realtime Database são armazenados como objetos JSON. Pense na base de dados como uma árvore JSON hospedada na nuvem. Ao contrário de uma base de dados SQL, não há tabelas nem registros. Quando adicionamos dados à árvore JSON, eles tornam-se na estrutura JSON existente com a chave associada. É possível fornecer chaves próprias [C34].

[C28]:

<https://db-engines.com/en/system/Firebase+Realtime+Database%3BMicrosoft+Azure+Cosmos+DB>

[C29]:

<https://www.red-gate.com/simple-talk/cloud/azure/overview-of-azure-cosmos-db/>

[30]: <https://firebase.google.com/docs/database?hl=pt>

[C33]:

<https://learn.microsoft.com/en-us/azure/cosmos-db/introduction>

[C34]:

<https://firebase.google.com/docs/database/unity/structure-data?hl=pt-br>

MSFT Azure Cosmos DB VS Firebase Realtime Database

Índices

MSFT Azure Cosmos DB [C31]:

Indexa todas as propriedades sem definir qualquer esquema e sem ter que configurar índices secundários. O Azure Cosmos DB suporta três tipos de índices: Índice de Intervalos, Índice espacial e Índices compostos.

Firebase Realtime Database [C35]:

Cria sempre um índice automático na chave de um nó. Podemos adicionar índices usando `orderByChild` e `orderByValue`.

Queries

MSFT Azure Cosmos DB:

Permite consultar os dados através de consultas SQL [C37].

Firebase Realtime Database:

Mais limitado. Consultas por chave, valor, valor de filho e filtrar o resultado ordenado por um número específico de resultados ou um intervalo de chaves ou valores [C36].

ORADOR:

Neste slide é feita a comparação em relação a índices e queries.

O Cosmos DB Indexa todas as propriedades sem definir qualquer esquema e sem ter que configurar índices secundários e suporta três tipos de índices, Índice de Intervalos, Índice espacial e Índices compostos.

O Firebase cria sempre um índice automático na chave de um nó. Podemos adicionar índices usando `orderByChild` e `orderByValue`.

Para consultas o Cosmos DB disponibiliza a linguagem SQL. O Firebase é mais limitado, permite consultas por chave, valor, valor de filho e filtrar o resultado ordenado por um número específico de resultados ou um intervalo de chaves ou valores.

CONTEÚDO:

sem ter que configurar índices secundários. O Azure Cosmos DB suporta três tipos de índices: Índice de Intervalos, Índice espacial e Índices compostos.

Índice de Intervalos: Os índices de intervalo baseiam-se numa estrutura ordenada semelhante a uma árvore. O tipo de índice de intervalo é utilizado para: consulta de igualdade, correspondência de igualdade num elemento de matriz, consultas de intervalo, verificar a presença de uma propriedade, funções do sistema de cadeias, order by e join.

Índice espacial: Os índices espaciais permitem consultas eficientes em objetos geoespaciais, como pontos, linhas, polígonos e multipolígonos. Estas consultas utilizam palavras-chave ST_DISTANCE, ST_WITHIN ST_INTERSECTS.

Índices compostos : Os índices compostos aumentam a eficiência quando executa operações em vários campos. O tipo de índice composto é utilizado para: order by, consultas com filtro e order by e consultas com um filtro em duas ou mais propriedades em que pelo menos uma propriedade é um filtro de igualdade [C31].

O Firebase permite que realizar consultas ad-hoc usando uma chave filha arbitrária. Se soubermos com antecedência quais serão os índices, podemos defini-los pela regra .indexOn nas regras de segurança do Firebase Realtime Database para melhorar o desempenho das consultas.

Definir índices de dados

O Firebase fornece ferramentas eficientes para classificação e consulta de dados. Especificamente, O Firebase permite a criação de consultas ad-hoc em uma coleção de nós usando qualquer chave filha comum. Conforme a aplicação é expandida, o desempenho dessa consulta é degradada-se. No entanto, se indicarmos ao Firebase as chaves a consultar, ele as indexará nos servidores, melhorando o desempenho das

consultas.

[C36]

É possível usar a classe Query do Realtime Database para recuperar dados ordenados por chave, valor ou valor de filho. Também é possível filtrar o resultado ordenado por um número específico de resultados ou um intervalo de chaves ou valores.

Ordenar dados

Para recuperar dados ordenados, começamos pela especificação de um dos métodos de ordenação para determinar como os resultados são ordenados:

orderByChild(): Ordenar resultados pelo valor de uma chave filha específica ou caminho filho aninhado.

orderByKey(): Ordenar resultados por chaves filhas.

orderByValue(): Ordenar resultados por valores filhos.

O Azure Cosmos DB para NoSQL permite consultar dados ao escrever consultas com a linguagem SQL (Structured Query Language) como uma linguagem de consulta JSON. Por exemplo por palavra chave, funções matemáticas etc[C37].

[C31]:

<https://learn.microsoft.com/pt-pt/azure/cosmos-db/index-overview>

[C35]:

<https://firebase.google.com/docs/database/security/indexing-data?hl=pt-br>

[C36]:

https://firebase.google.com/docs/database/web/lists-of-data?hl=pt-br#sorting_and_filtering_data

[C37]:

[https://learn.microsoft.com/pt-pt/azure/cosmos-db/nosql/quer
y/](https://learn.microsoft.com/pt-pt/azure/cosmos-db/nosql/quer
y/)

CouchDB VS Couchbase

Instalação ou serviço? [C38]

Quer o **CouchDB** quer o **Couchbase** são hospedados localmente, podendo serem instalados na nuvem em máquinas virtuais.

Tipos de dados

CouchDB[C41]:

Usa o formato JSON.

Couchbase[C43][C45]:

Usa o formato JSON. Suporta tipos de dados básicos, como números e strings; e tipos complexos, como documentos e arrays incorporadas.

ORADOR:

Nestes últimos dois slides é feita a comparação entre o CouchDB e o Couchbase. Os dois são hospedados localmente havendo a possibilidade de ter na nuvem, pública ou privada, mas numa máquina virtual.

O CouchDB e o Couchbase usam o formato JSON. O JSON suporta tipos de dados básicos, como números e strings; e tipos complexos, como documentos e arrays incorporadas.

CONTEÚDO:

[C38]

Quer o CouchDB quer o Couchbase são hospedados localmente.

[C41]

mesmos, através de um navegador web (internet browser), via HTTP. As consultas, e alterações aos documentos são efetuadas utilizando JavaScript. O CouchDB funciona bem com aplicações modernas web e móveis. Permite distribuir os dados, eficientemente usando a replicação incremental. O CouchDB suporta configurações mestre-mestre com detecção automática de conflitos. Vem com um conjunto de recursos, como a transformação de documentos e notificações de alterações em tempo real, que facilitam o desenvolvimento web, vem ainda com uma consola de administração web fácil de usar. O CouchDB é altamente disponível e tolerante a partições, mas também é eventualmente consistente e possui um mecanismo de armazenamento tolerante a falhas que coloca a segurança dos dados em primeiro lugar.

[C43][C45]

O modelo de dados do Couchbase é baseado em JSON, o que fornece uma notação simples, leve e legível por humanos. Ele suporta tipos de dados básicos, como números e strings, e tipos complexos, como documentos incorporados e arrays. O JSON oferece serialização e desserialização rápidas, é nativo do JavaScript e constitui o tipo de retorno mais comum para APIs REST. Consequentemente, o JSON é extremamente conveniente para programação de aplicações web. Um documento individual muitas vezes representa uma única instância de um objeto no código da aplicação. Um documento pode ser considerado equivalente a uma linha em uma tabela relacional, sendo que cada atributo do documento é equivalente a uma coluna. No entanto, o Couchbase oferece maior flexibilidade do que bancos de dados relacionais, pois pode armazenar documentos JSON com esquemas variados. Os documentos podem conter estruturas aninhadas. Isso permite que os desenvolvedores expressem relacionamentos muitos para muitos sem precisar de uma tabela de referência ou de junção, e é naturalmente expressivo para dados hierárquicos.

[C38]:

<https://db-engines.com/en/system/CouchDB%3BCouchbase>

[C41]: <https://docs.couchdb.org/en/stable/intro/index.html>

[C43]:

<https://docs.couchbase.com/server/current/getting-started/try-a-query.html>

[C45]:

<https://docs.couchbase.com/server/current/learn/data/document-data-model.html>

CouchDB VS Couchbase

Índices

CouchDB[C42]:

Os índices são criados e mantidos através de views.

Couchbase[C44]:

No Couchbase estão presentes vários índices, entre eles: primário, secundário, full text, análise e o view.

Queries

CouchDB[C41][C42]:

O CouchDB usa o JavaScript.

Couchbase [C39][C40]:

O Couchbase Server pode ser consultado usando SQL++

SQL++

```
SELECT a.country FROM default:'travel-sample'.inventory.airline a
WHERE a.name = "Excel Airways";
```

ORADOR:

Por fim temos a comparação dos índices e das queries destes dois SGBD's.

No CouchDB os índices são criados e mantidos através de views (consultas) e em relação ao Couchbase é possível termos vários tipos de índices entre eles: primário, secundário, full text, view, etc.

Nas queries, o JavaScript é a linguagem utilizada pelo CouchDB e o SQL++ é utilizado no Couchbase. Temos aqui um exemplo de uma query em SQL++. O resultado de uma query deste género será no formato JSON.

CONTEÚDO:

O CouchDB [C42]

As "views" são definidas usando funções JavaScript que atuam como a parte "map" de um sistema de map-reduce. Uma função de view recebe um documento CouchDB como

view. As "views" são uma representação dinâmica do conteúdo real dos documentos de uma base de dados, e o CouchDB facilita a criação de visualizações úteis dos dados. No entanto, gerar uma visualização de uma base de dados com centenas de milhares ou milhões de documentos consome tempo e recursos, não é algo que o sistema deva fazer do zero sempre que existam alterações. Para manter a consulta rápida, o mecanismo de visualização mantém índices e atualiza-os de forma a refletir as alterações na base de dados. O desenho do CouchDB foi otimizado para a necessidade de criação eficiente e incremental de consultas e dos seus índices. As consultas (views) e as suas funções são definidas dentro dos documentos "design" especiais, e um desses documentos de "desenho" pode conter várias "view functions" com nomes únicos. Quando um utilizador abre uma consulta (view) e o seu índice é atualizado automaticamente, todas as consultas no mesmo documento de design são indexadas como um único grupo.

O Couchbase[C44]

O Couchbase tem vários índices sendo estes: o **primary index**- baseia-se na chave exclusiva de cada item numa coleção especificada. Cada índice primário é mantido de forma assíncrona. Um índice primário destina-se a ser usado para consultas simples, que não possuem filtros ou predicados; **secondary index**- é baseado num atributo dentro de um documento. O valor associado ao atributo pode ser de qualquer tipo: escalar, objeto ou array. Um Índice Secundário é frequentemente referido como Índice Secundário Global, ou GSI. Este é o tipo de índice utilizado com mais frequência no Couchbase Server, para consultas realizadas em SQL++; **full text**- este é um índice especialmente projetado, que contém alvos derivados do conteúdo textual dos documentos dentro de um ou mais espaços-chave especificados. Podem ser efetuadas pesquisas correspondentes ao texto com diferentes graus de exatidão. Os valores de texto de entrada e de destino podem

ser eliminados pelos caracteres irrelevantes (como sinais de pontuação ou tags HTML); **analytics**- este é um caminho de acesso realizado para os dados sombra numa coleção do Analytics. Os índices do Analytics podem ser usados para acelerar consultas de seleção do Analytics e consultas de junção. Se as alterações nos dados operacionais resultarem em modificações correspondentes nos dados sombra, os índices do Analytics serão atualizados automaticamente; **view**- as visualizações estão antigas no Couchbase Server 7.0 e serão removidas numa versão futura.

[C40] [C39]

Couchbase SQL++

O Couchbase Server pode ser consultado usando SQL++ (a linguagem de consulta do Couchbase Server). A implementação Couchbase do SQL++ era conhecida anteriormente como N1QL ("níquel"), cujo nome deriva da primeira forma não normal do modelo de dados.

SQL++ é uma linguagem SQL expressiva, poderosa e completa que permite consultar, transformar e manipular dados JSON. Baseado em SQL, é familiar aos programadores que lhes permite começar a desenvolver rapidamente aplicações avançadas.

SQL++ adota o modelo de documento JSON e usa uma sintaxe semelhante ao SQL. No SQL++, trabalhamos em documentos JSON e o resultado da pesquisa é outro documento JSON. Podemos executar consultas SQL++ na linha de comando, usando a ferramenta cbq ou no Query Workbench no Couchbase Server Web Console.

Uma consulta SQL++ básica possui as seguintes componentes: SELECT — Os campos de cada documento a serem devolvidos; FROM — A fonte de dados na qual procurar; WHERE — As condições que o documento deve satisfazer.

<https://docs.couchbase.com/server/current/getting-started/try-a-query.html>

[C40]:

<https://docs.couchbase.com/server/current/n1ql/query.html>

[C42]:

<https://docs.couchdb.org/en/stable/intro/overview.html#javascript-view-functions>

[C44]:

<https://docs.couchbase.com/server/current/learn/services-and-indexes/indexes/indexes.html>