INSTITUTO DE GESTÃO E TECNOLOGIA
DA INFORMAÇÃO

Fundamentos em Banco de Dados

Diego Bernardes de Lima Santos

2022

Fundamentos em Banco de Dados

Diego Bernardes de Lima Santos

Sumário_Toc104981469

Capítulo 1. Introdução

SGBD

Um Sistema de Gerenciador de banco de dados (SGBD) é um sistema de software de computador que interage com o usuário, outras aplicações e outros bancos de dados para armazenar, recuperar e analisar dados. Coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados.

Dados

Chamamos de Dados fatos conhecidos que podem ser registrados e possuem significado implícito (ex.: nome, telefone, CPF etc.).

Defino dado como uma sequência de símbolos quantificados ou quantificáveis. Portanto, um texto é um dado. De fato, as letras são símbolos quantificados, já que o alfabeto, sendo um conjunto finito, pode por si só constituir uma base numérica. Também são dados fotos, figuras, sons gravados e animação, pois todos podem ser quantificados a ponto de se ter eventualmente dificuldade de distinguir a sua reprodução, a partir da representação quantificada, com o original. É muito importante destacar que, qualquer texto constitui um dado ou um conjunto de dados.

Banco de Dados

Coleção de dados com informações relevantes que é armazenada em algum local onde possa ser recuperada posteriormente. Ex.: pastas (de papéis), arquivos (de

papéis), hd's, fitas etc. Bancos de dados possuem alguma ligação com o mundo real, com a fonte que gera as informações, usuários, algum ator que futuramente possa necessitar das informações armazenadas no banco de dados.

SGBDs X Bancos de Dados

O Sistema Gerenciador de Bancos de Dados é um sistema que é projetado para gerir os volumes de informações que são armazenados em um banco de dados.

O SGBD deve fornecer mecanismos de inserção, recuperação, alteração e remoção de todas as informações do banco de dados.

O sistema gerenciador de banco de dados é um sistema computacional que possui a tarefa de armazenar, organizar, indexar e permitir o fácil acesso aos bancos de dados, conforme eles devem ser interconectados.

O sistema gerenciador de banco de dados deve implementar linguagens específicas para recuperação de informação, em tempo aceitável ao usuário, e de maneira que a informação seja confiável e consistente, facilitando a utilização e manutenção dessas informações.

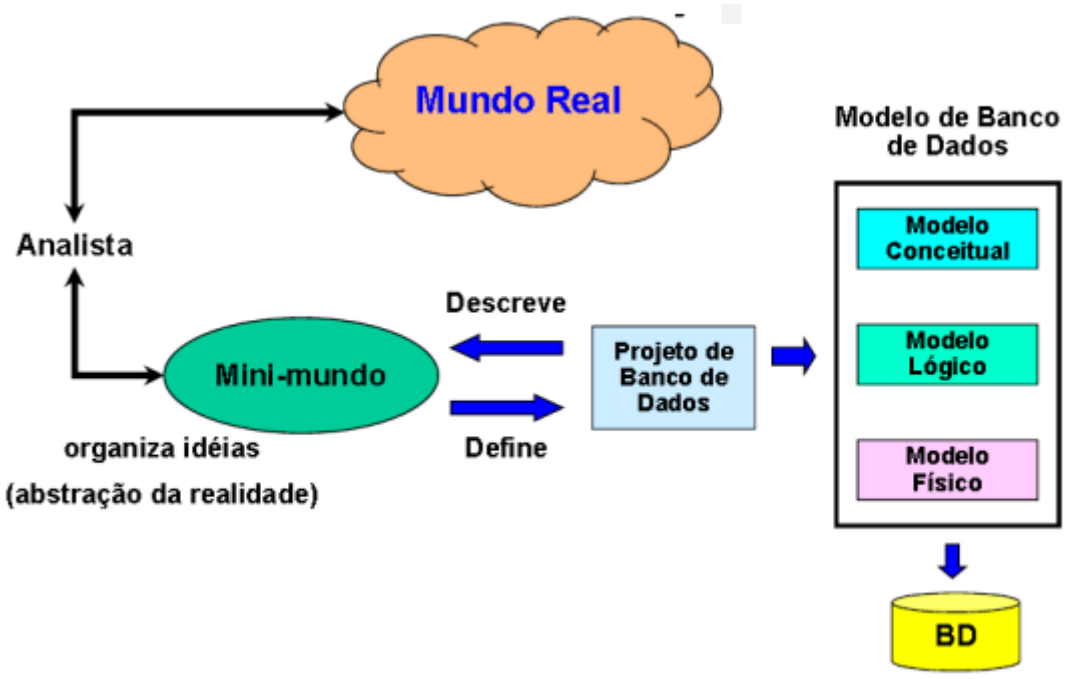
Modelagem de Dados

Desde o advento dos sistemas gerenciadores de bancos de dados (SGBDs), a modelagem relacional é, sem dúvida, a técnica mais utilizada em projetos de desenvolvimento de sistemas e aplicações apoiados em bancos de dados.

A modelagem de dados é um processo onde o analista procura compreender o ambiente em que a informação trafega, suas características, restrições e particularidades, para projetar um sistema que realizará o armazenamento dessas informações.

De uma forma geral, o processo de modelagem de dados pode seguir as quatro grandes etapas:

- Descrição do minimundo/Levantamento de Requisitos.
- Modelagem Conceitual.
- Modelagem Lógica.
- Modelagem Física.



Fonte: https://sites.google.com/site/giygiyhvhj/home/o-fatecano.

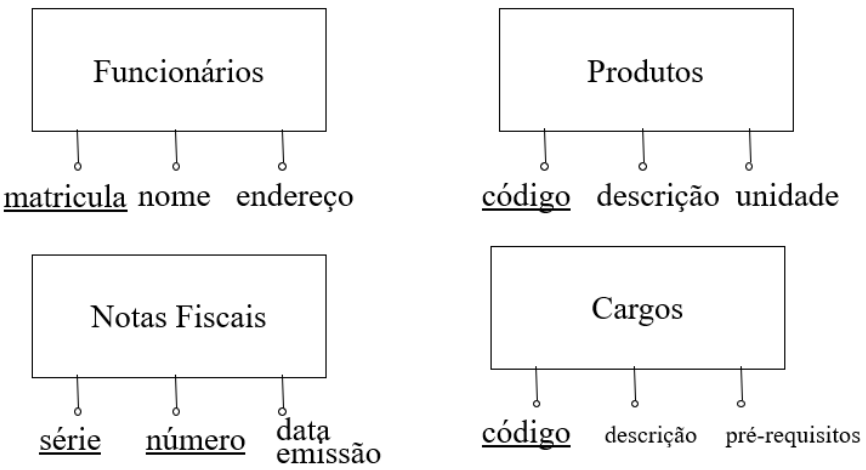
Modelo de Entidade Relacionamento

Entidades

Conjunto de objetos sobre os quais é preciso armazenar informações úteis, que podem englobar um conjunto de vários elementos. Conjuntos de elementos distinguíveis que aceitam um código para diferenciá-los. Conjuntos qualificativos (ex.: grau de instrução).

Entidades são representadas por retângulos, no modelo conceitual, e sua definição de nomes está dentro dos retângulos.

Exemplos:



Atributos

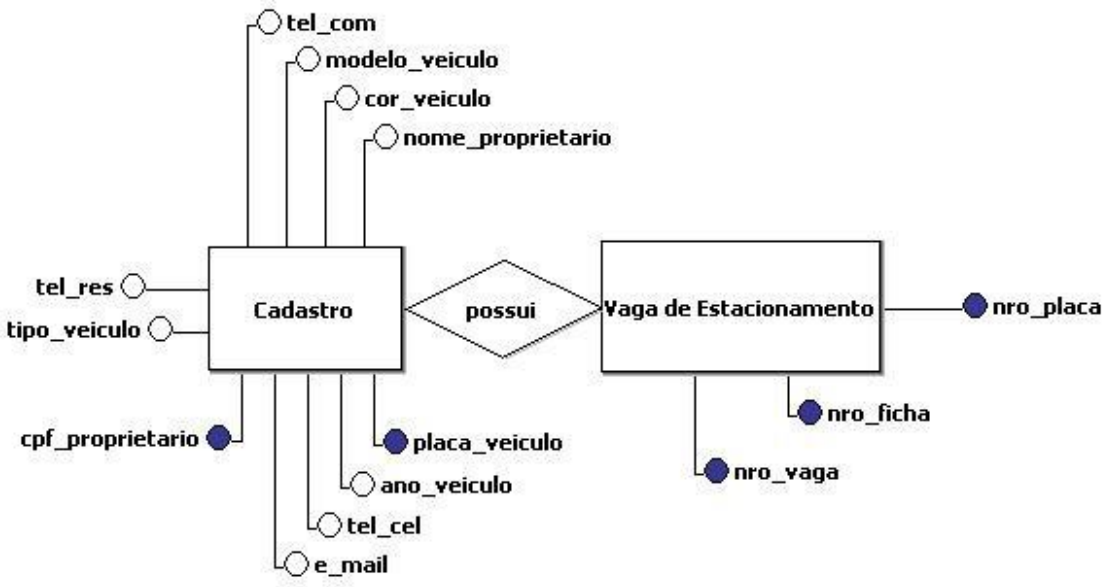
Informações úteis a respeito de uma entidade ou relacionamento.

Os atributos de uma entidade permanecem constantes para todos os seus relacionamentos. Os atributos de uma entidade são independentes de todas as demais entidades.

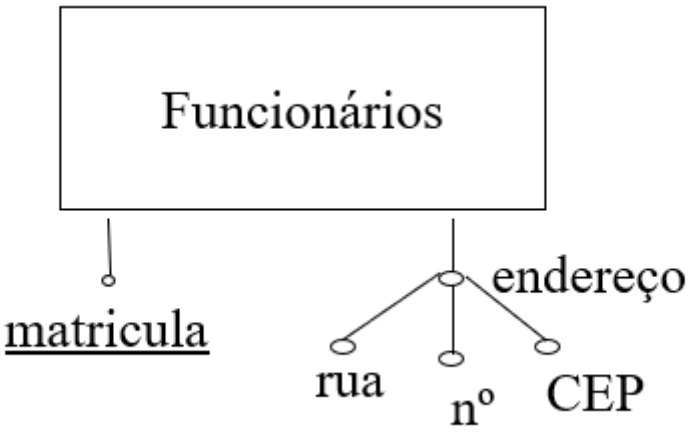
Tipos de atributos:

- Chave: Valor único que representa um elemento da entidade.
- Composto: Necessita ser dividido em subatributos, para que seu significado seja melhor compreendido.
- Multivalorado: Atributo que possui vários valores ou uma lista de valores.

Os atributos chave são identificados com o círculo preenchido (colorido) conforme exemplo a seguir:

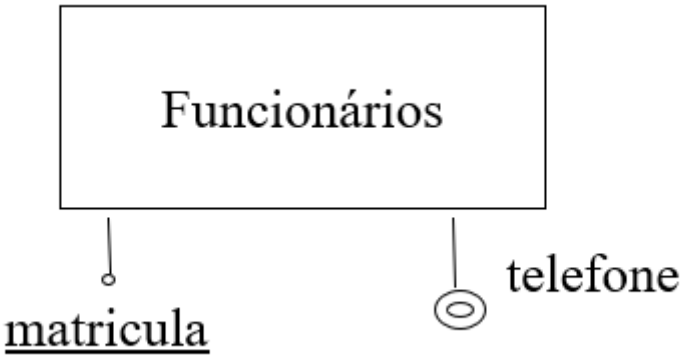


Os atributos compostos são representados da seguinte forma:



O campo “endereço”, no exemplo acima, é um atributo composto pelos dados: rua, número e CEP, embora seja apenas uma informação, ela pode ser desmembrada em três e representada com uma ligação dos sub-atributos ao atributo real.

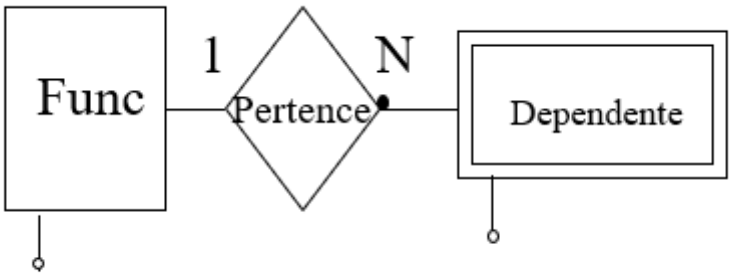
Os atributos multivalorados, que possuem apenas uma segmentação do dado, mas pode repetir n vezes dentro do mesmo campo, é representado da seguinte maneira:



O atributo telefone, no exemplo acima, possui vários valores e isso é representado pelo duplo círculo que se conecta à entidade, indicando essa possibilidade.

Entidades Fracas

As entidades fracas têm a mesma função de uma entidade comum, porém um registro delas só pode existir caso um registro de uma entidade forte exista também. Considere o exemplo abaixo:



A entidade “Dependente” é uma entidade fraca, pois um registro só pode existir nela caso exista um funcionário. Nesse caso não faz sentido cadastrar um dependente que não possua alguém responsável por ele. A representação gráfica de uma entidade fraca é o duplo retângulo, conforme exposto na entidade “Dependente”.

Capítulo 2. Modelagem Relacional – Modelo Lógico

O modelo relacional foi criado por Edgar Frank Codd, em junho de 1970, com a publicação do seu artigo *A Relational Model of Data for Large Shared Data Banks*. Nesse artigo, Codd propôs um modelo onde a representação dos dados é independente de como eles são organizados internamente nos servidores, ou seja, um modelo lógico totalmente independente da estruturação dos dados.

Com um determinado registro (dados inter-relacionados) sendo visto como uma tupla (linha formada por uma lista ordenada de colunas), a representação tabular trouxe grande facilidade para manipular e visualizar dados, em detrimento dos modelos hierárquicos e de rede, em voga na época, contribuindo enormemente para o "boom" dos sistemas gerenciadores de bancos de dados relacionais.

Exemplo de uma tabela:

	EMPREGADO_ID	PRIMEIRO_NOME	ULTIMO_NOME	EMAIL	DATA_ADMISSAO	CARGO_ID	SALARIO	GERENTE_ID	DEPARTAMENTO_ID
1	100	Steven	King	SKING	2003-06-17	AD_PRES	24000.00	NULL	90
2	101	Neena	Kochhar	NKOCHHAR	2005-09-21	AD_VP	17000.00	100	90
3	102	Lex	De Haan	LDEHAAN	2001-01-13	AD_VP	17000.00	100	90
4	103	Alexander	Hunold	AHUNOLD	2006-01-03	IT_PROG	9000.00	102	60
5	104	Bruce	Ernst	BERNST	2007-05-21	IT_PROG	6000.00	103	60
6	105	David	Austin	DAUSTIN	2005-06-25	IT_PROG	4800.00	103	60
7	106	Valli	Pataballa	VPATABAL	2006-02-05	IT_PROG	4800.00	103	60
8	107	Diana	Lorentz	DLORENTZ	2007-02-07	IT_PROG	4200.00	103	60
9	108	Nancy	Greenberg	NGREENBE	2002-08-17	FI_MGR	12008.00	101	100
10	109	Daniel	Faviet	DFAVIET	2002-08-16	FI_ACCOUNT	9000.00	108	100

Junto com a teoria da representação tabular dos dados, Codd apresentou os conceitos e recursos a serem usados na modelagem relacional, descritos a seguir:

- TABELA: objeto correspondente à entidade do modelo conceitual, constituindo a estrutura onde os dados serão armazenados no modelo físico do banco de dados.

- DOMÍNIO: natureza dos valores permitidos para um determinado conjunto de dados. Exemplos:

- CODIGO CLIENTE - NUMBER
- NOME CLIENTE - STRING
- DATA NASCIMENTO – DATE

- COLUNA: correspondente ao conceito de atributo do modelo conceitual, acrescido do domínio do dado que será armazenado na coluna. Comumente chamado de campo.

- CHAVE PRIMÁRIA (“PRIMARY KEY” / “PK”): coluna (campo) ou conjunto de colunas (chave primária composta) que identificam unicamente a tupla (linha) em uma tabela. Exemplo: NÚMERO DO CPF.

- CHAVE CANDIDATA: que não repetem valor em uma tabela e que são candidatas à chave primária. Exemplos:
  - NÚMERO DO TÍTULO DO ELEITOR
  - NÚMERO DO RG
  - NÚMERO DO CPF

- CHAVE ESTRANGEIRA (“FOREIGN KEY” / “FK”): são o elo de ligação entre duas tabelas, se constituindo na(s) coluna(s) da chave primária da tabela estrangeira relacionada com a tabela em questão.

- RELACIONAMENTO: aplica-se o mesmo princípio do modelo conceitual para o mapeamento das relações entre as tabelas, sendo que alguns relacionamentos podem gerar novas tabelas.

- CARDINALIDADE: no modelo lógico relacional, assume uma importância ainda maior, pois além de representar a quantidade de elementos de uma entidade A que podem ser associados à elementos de uma entidade B e vice-versa, indicam também as restrições de nulidade das chaves estrangeiras.

- RESTRIÇÕES DE INTEGRIDADE: regra que garante a consistência dos dados em um relacionamento entre duas tabelas. São de três tipos:

  – Cascade (C): ao deletar um registro, deleta registro da chave estrangeira também (deleção em cascata).

  – Restrict (R): não deleta um registro que seja chave estrangeira.

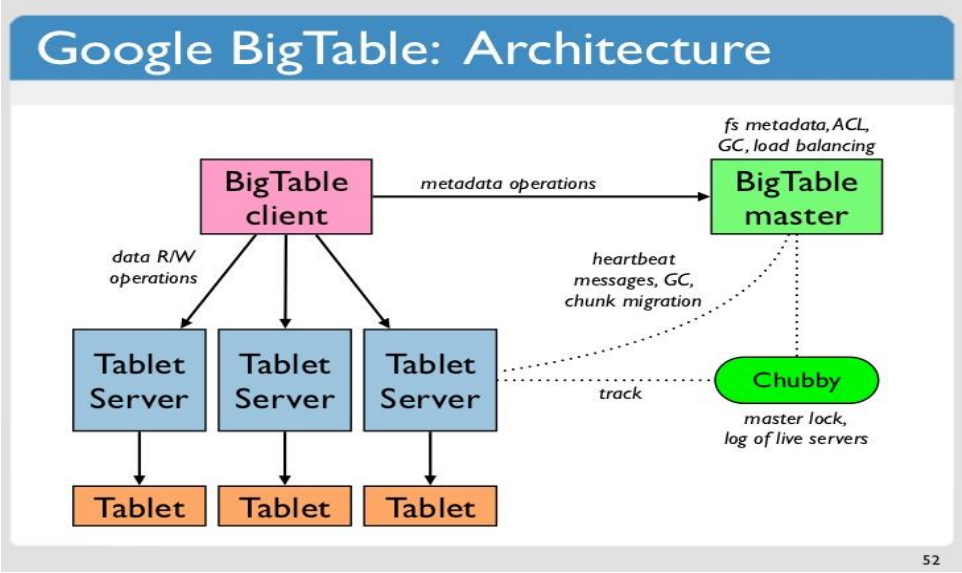
  – Set Null (S): deleta registro e seta chave estrangeira para nulo.

Capítulo 3. Bancos de Dados Não Relacionais

Os Bancos de Dados Não Relacionais, também conhecidos como Bancos de Dados NoSQL, são bancos de dados projetados para arquitetura em clusters (redes de computadores), de modo a permitir que o processamento dos dados seja realizado de maneira distribuída, com máquinas de baixo custo em detrimento de servidores dedicados e robustos dos modelos relacionais e de mercado.

O termo foi lançado mais ou menos em 1998 juntamente com um banco de dados que era controlado por linguagem SQL, fazendo referência ao fato de ser não relacional. Inicialmente, o termo significava No SQL (“sem SQL”), mas posteriormente, com a evolução das tecnologias envolvidas, se tornou Not Only SQL (“Não somente SQL”).

A seguir, uma arquitetura de um dos precursores da arquitetura NoSQL, o Google Big Table, que demonstra a destruição dos espaços de processamento, bem como armazenamento através de um cluster de computadores:



Fonte: Google.

É possível observar que há um nó orquestrador (Big Table Master), e cada nó “Tablet Server” tem seu espaço de processamento e armazenamento, com mecanismos de replicação para evitar a perda de dados, caso um nó tenha problemas de indisponibilidade.

Propriedades ACID

As propriedades ACID são implementadas em todos os bancos de dados relacionais, pois essas propriedades visam garantir que os dados sejam íntegros, consistentes, e que todas as interações dos usuários com os dados sejam refletidas corretamente ao armazená-las em disco.

De maneira específica, as propriedades ACID são:

- Atomicidade - A propriedade de atomicidade garante que as transações sejam atômicas (indivisíveis). A transação será executada totalmente ou não será executada, ou seja, se uma transação possui 1 ou N tarefas intermediárias, ela só será confirmada se todas as N tarefas forem executadas com sucesso, caso contrário toda a transação, bem como suas tarefas serão desfeitas.

- Consistência - A propriedade de consistência garante que o banco de dados passará de uma forma consistente para outra forma consistente, isto é, os dados antes e após as transações devem estar compatíveis com as restrições implementadas, por exemplo, um campo é obrigatório, esse tipo de validação deve continuar funcionando.

- Isolamento - A propriedade de isolamento garante que a transação não será interferida por nenhuma outra transação concorrente. Bases de dados convivem com acesso concorrente nos mesmos dados, o SGBD deve garantir que cada transação ocorrerá naturalmente sem uma influenciar a outra.

- Durabilidade - A propriedade de durabilidade garante que o que foi salvo, não será mais perdido. Uma vez que o dado foi confirmado (commit), o dado permanecerá em disco sem risco de perda de informação ao final das transações.

Propriedades BASE

Propriedades BASE indicam a expressão “basicamente disponível”, que indica que o banco de dados possui momentos de inconsistência, embora na maior parte do tempo o banco de dados esteja consistente. Esse estado intermediário busca flexibilizar as propriedades ACID, buscando maior desempenho, uma vez que Bancos de Dados NoSQL são bancos de dados projetados para altos volumes de informação.

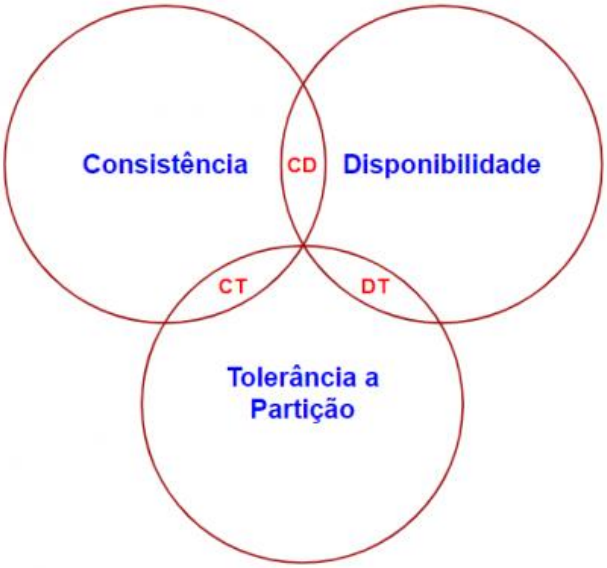
A seguir, uma comparação entre as propriedades ACID x BASE:

ACID	BASE
Consistência Forte	Consistência Fraca
Isolamento	Foco em Disponibilidade
Transações aninhadas	Repostas Aproximadas
Disponibilidade	Simples / Rápido
Conservador	Agressivo

As propriedades BASE são fundamentadas no Teorema CAP. O teorema CAP, de acordo com Eric Brewer, da Universidade da Califórnia, prevê que um sistema computacional distribuído é desejável obter consistência, disponibilidade e tolerância ao particionamento. Porém, no mundo real, essas propriedades são impossíveis de serem obtidas ao mesmo tempo. De acordo com esse teorema, que ficou conhecido

como teorema CAP (Consistency, Availability e Partition Tolerance), um sistema distribuído só pode garantir apenas duas dessas propriedades simultaneamente.

A seguir a demonstração gráfica desse comportamento, é interessante observar que não existe uma interseção entre as três propriedades de consistência, disponibilidade e tolerância à partição.



Como é possível observar na imagem anterior, o banco de dados distribuído oscila permanentemente entre as três propriedades, sempre garantido duas delas ocorrendo ao mesmo tempo, e mudando o foco de acordo com o momento.

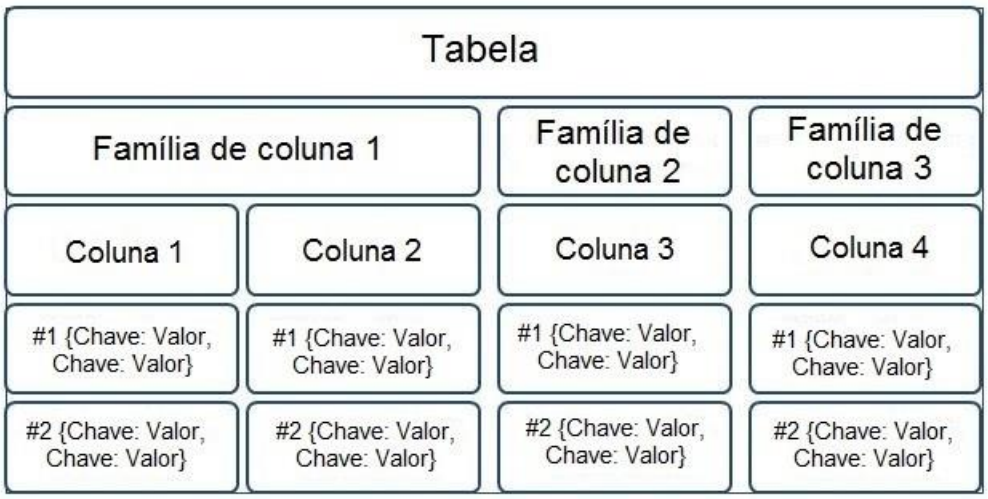
Bancos de Dados Colunares

Enquanto um banco de dados relacional é otimizado para armazenar linhas de dados, geralmente para aplicações transacionais, com propriedades ACID, um banco de dados colunar é otimizado para recuperação rápida de colunas de dados, normalmente em aplicativos analíticos.

O armazenamento orientado a colunas para tabelas do banco de dados é um fator importante para a performance de consulta analítica, pois ele reduz expressivamente os requisitos gerais de entrada e saída de disco (I/O) e diminui a quantidade de dados que você precisa carregar do disco na memória.

Assim como outros bancos de dados NoSQL, os bancos de dados em colunas foram criados para aumentar a escala horizontal usando clusters distribuídos de hardware de baixo custo para aumentar a taxa e transferência de dados, o que os torna ideais para sistemas de Business Intelligence e Big Data.

A modelagem colunar tem estrutura similar ao exemplo abaixo:



Nesse contexto as famílias de colunas podem ser organizadas por assuntos, por informações que geralmente são recuperadas juntas, por exemplo, dados pessoais de um funcionário como CPF, nome e e-mail, geralmente aparecem nas mesmas consultas, ao passo que dados como cargo, formação acadêmica não são tão comuns, nesse exemplo, poderíamos segmentar dados pessoais em uma família de colunas e dados profissionais em outra família de colunas, todavia todas essas informações seriam da mesma tabela: empregados.

A arquitetura colunar, portanto, favorece a recuperação de informação, logo um sistema com muitas transações e atualizações não é recomendável ser realizado em bancos de dados colunares, embora seja possível. O foco principal do banco colunar é o acesso rápido à informação.

Bancos de Dados de Documentos

Um banco de dados de documentos é um tipo de banco de dados NoSQL projetado para armazenar e consultar dados como documentos do tipo JSON. 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 da aplicação.

A natureza flexível, semiestruturada e hierárquica dos documentos e dos bancos de dados de documentos, permitem a evolução conforme as necessidades dos sistemas. O modelo de documentos funciona em casos de uso como catálogos, perfis de usuários e sistemas de gerenciamento de conteúdo, onde cada documento é único e evolui com o passar do tempo.

Os bancos de dados de documentos possibilitam uma indexação flexível, consultas ad hoc eficientes e análises de dados em grupos de documentos, e possuem melhor desempenho em situações em que os documentos são completamente independentes e não fazem relações de chave entre si, ou seja, onde operações como “joins” não sejam necessárias.

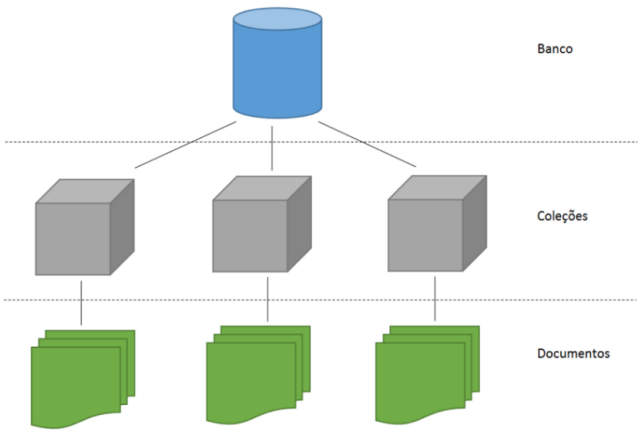
No exemplo a seguir, um documento semelhante ao JSON descreve um livro:

```
[
    {
        "year" : 2013,
        "title" : "Turn It Down, Or Else!",
        "info" : {
            "directors" : [ "Alice Smith", "Bob Jones"],
            "release_date" : "2013-01-18T00:00:00Z",
            "rating" : 6.2,
            "genres" : ["Comedy", "Drama"],
            "image_url" : "http://ia.media-imdb.com/images/N/O9ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@@._V1_SX400_.jpg",
            "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",
            "actors" : ["David Matthewman", "Jonathan G. Neff"]
        }
    },
    {
        "year": 2015,
        "title": "The Big New Movie",
        "info": {
            "plot": "Nothing happens at all.",
            "rating": 0
        }
    }
]
```


Fonte: www.amazon.com.

Como é possível observar, os atributos são separados em chave-valor, em uma estrutura hierárquica de armazenamento. Por exemplo, o campo year possui o valor 2013, e em uma estrutura hierarquicamente interna, temos o campo “info”, com informações específicas.

A arquitetura de um banco de dados de documento, embora resguardadas as particularidades de cada fornecedor, segue estrutura semelhante à imagem a seguir:



Fonte: MongoDB.

Na estrutura do MongoDB, temos a base de dados (banco) que centralizar a persistência de documentos, que são por sua vez divididos em coleções, geralmente criadas por projetos ou assuntos.

Bancos de Dados de Grafos

Grafos

Um grafo é uma representação abstrata de um conjunto de objetos e das relações existentes entre eles. É definido por um conjunto de nós ou vértices, e pelas ligações ou arestas, que ligam pares de nós. Uma grande variedade de estruturas do mundo real, podem ser representadas abstratamente através de grafos.

Um grafo basicamente tem duas estruturas principais, vértices e arestas:

- Vértice: Um vértice é uma estrutura que geralmente representa uma informação ou entidade do mundo real, que pode ou não se conectar com outras entidades semelhantes.

- Aresta: Aresta é um conector que indica uma ligação entre dois vértices, estabelece uma relação.

A imagem abaixo representa o mapa de um trecho do metrô da cidade de São Paulo, que podemos interpretá-la como um grafo:

Nesse contexto, os vértices são as estações, por exemplo, Ana Rosa, Brigadeiro, Trianon-Masp, são vértices nesse grafo, ao passo que as linhas verde, amarela e vermelha são as arestas.

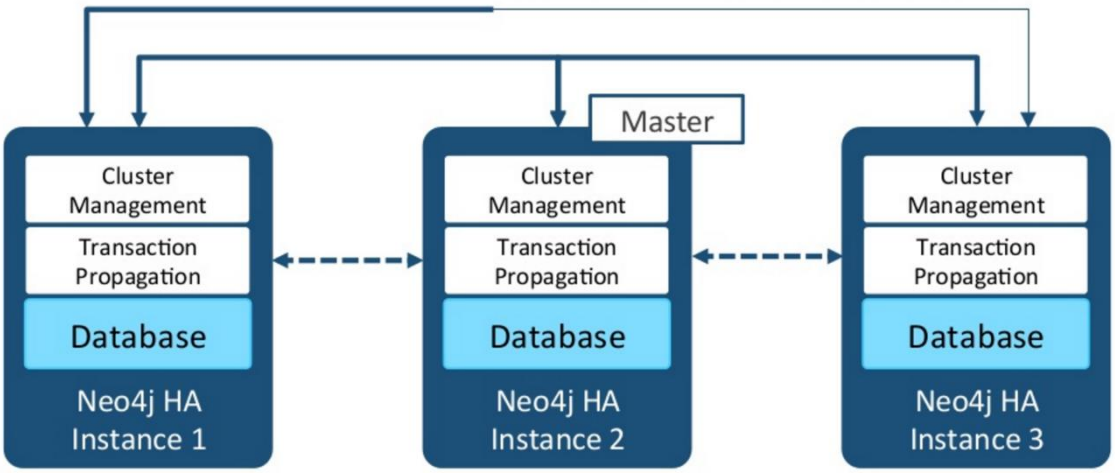
Por definição, uma aresta une um ponto do grafo ao outro, ou seja, existe uma aresta entre o vértice “Faria Lima” e o vértice “Fradique Coutinho”, porém não há um vértice que liga Faria Lima à “Oscar Freire”

Em um grafo é possível, adicionalmente, incluir peso nas arestas, indicando que a ligação é mais “cara” ou mais “barata” de acordo com seu peso, para objetivos de modelagem. Por exemplo, podemos dizer que a distância entre “Ana Rosa” e “Brigadeiro” é de 5km e de “Brigadeiro” à “Trianon-Masp” 8km, o que acarretaria em um peso maior, em caso de uma modelagem que pressupõe a definição de rotas mais

curtas. Por outro lado, podemos considerar o tempo de deslocamento de um vértice a outro, nesse caso o peso seria diferente, para calcular a melhor rota, portanto desconsiderando a distância entre os vértices.

Bancos de dados orientados a grafos são uma das categorias de bancos de dados (BDs) NoSQL propostas. A grande diferença para o modelo clássico está na representação explícita de relacionamentos entre os dados, através de um modelo com vértices, chamados de nós e arcos chamados de relações. Esperamos que esse modelo, menos genérico que o modelo relacional, nos forneça uma modelagem mais simples, uma linguagem mais natural para descrever as consultas.

Seguindo a proposta de soluções NoSQL, a arquitetura dos bancos de dados orientados a Grafos, exemplificada pela imagem a seguir, do banco de dados Neo4J, segue a arquitetura baseada em clusters:



Fonte: www.neo4j.com.

Capítulo 4. Tópicos Complementares

As discussões acerca dos bancos de dados relacionais e não relacionais é importante no contexto de processamento de informações, bem como análise de dados. É importante conhecer a arquitetura de Big Data, onde essas duas realidades convivem de maneira harmônica, dependendo sempre da característica dos dados e dos recursos.

Arquitetura de Dados

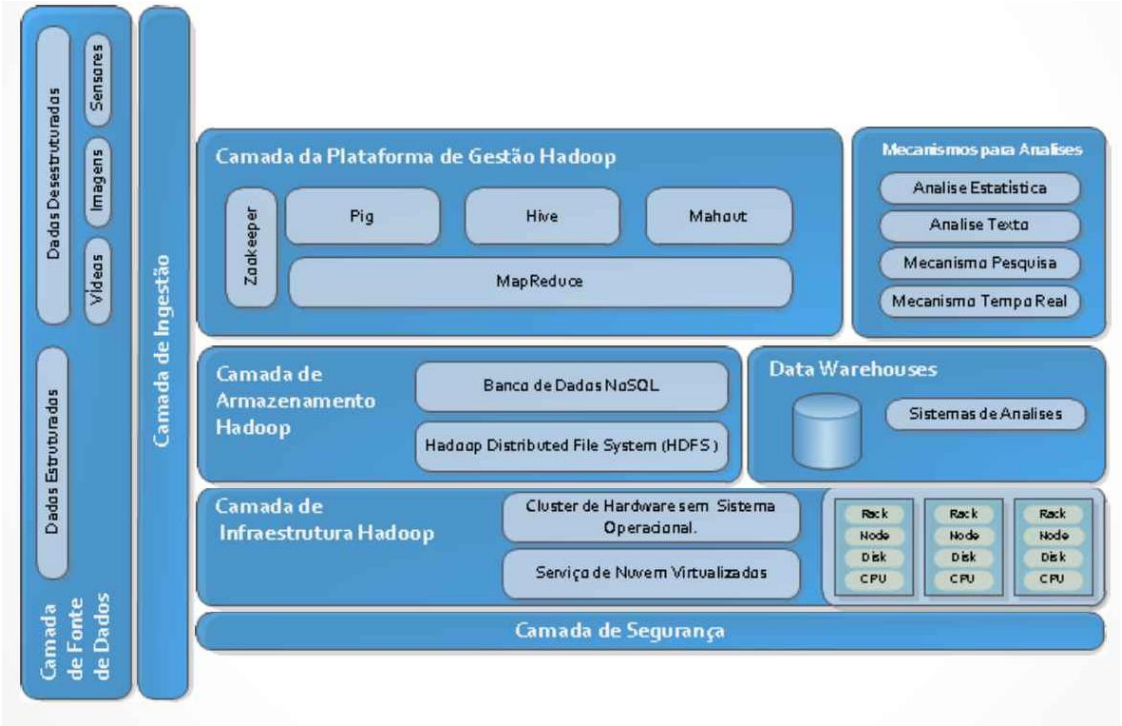
Arquitetura de dados é a estrutura dos componentes de dados de uma organização - considerados sob diferentes níveis de abstração, suas relações, bem como os princípios, diretrizes, normas e padrões que regem seu projeto e evolução ao longo do tempo. A arquitetura de dados é parte, ou grande parte, componente de análise de dados ou pipeline de dados.

Pipeline de Dados

Pipeline de dados é um processo de tratamento de informação, que contempla as etapas de engenharia de dados ou ingestão de dados, que consiste em organizar e orquestrar a coleta de dados, a preparação de dados que realiza a limpeza de dados incorretos ou inconsistentes, a padronização e enriquecimento dos dados com a geração de alguma informação derivada, além da análise dos dados, que é a última etapa do processo, que consiste em consumir, interpretar e aplicar regras estatísticas com o objetivo de obter conhecimento de negócios.

Big Data ou Arquitetura de Dados em Big Data, pode ser definido como todo o processo de tratamento da informação, contemplando a arquitetura dos dados e sistemas, bem como a definição de pipeline de dados, associados às ferramentas presentes em cada uma das etapas.

No contexto de Big Data, é importante, ao definir um pipeline de dados, escolher qual banco de dados é adequado à necessidade, seja relacional ou não, dependendo da característica da informação a ser armazenada. Considerando a questão de arquitetura, a imagem abaixo ilustra uma usual arquitetura de Big Data:



É importante destacar, entre os demais componentes, a camada de armazenamento, onde as bases de dados estão desenhadas nas soluções de Big Data. O bloco de data warehouses também é componente da arquitetura dos dados, uma vez que comporta dados analíticos.

As camadas de armazenamento e de data warehouses podem ser desenhadas com bancos de dados relacionais, colunares, documentos e mesmo arquivos. Por isso a importância de assertividade na escolha da base adequada, uma vez que ela se integra aos demais componentes da arquitetura de dados e dos pipelines de informação, onde tempo e carga são componentes fundamentais, principalmente em ambientes de Big Data.

Capítulo 5. Armazenamento e Processamento Distribuído

Introdução

Bancos de Dados em nuvem

Os sistemas em nuvem, por definição, não são sistemas distribuídos em sua essência, entretanto os serviços em nuvem apresentam vantagens tradicionalmente inerentes aos sistemas distribuídos, como, por exemplo, a escalabilidade horizontal e alta disponibilidade, além de incluírem outras vantagens como escalabilidade vertical e maior flexibilidade em aumentar ou diminuir o tamanho da solução de acordo com o consumo.

Computação em nuvem comercialmente é utilizada como serviço, ou seja, o usuário final contrata e utiliza os recursos sem se preocupar com instalação, manutenção e administração.

É possível classificar os serviços através dos seguintes tipos:

- IaaS: Infraestrutura como serviço. Esse serviço consiste na contratação de máquinas, com requisitos de hardware definidos pelo usuário. Sistema operacional e ferramentas devem ser instaladas pelo usuário da máquina.

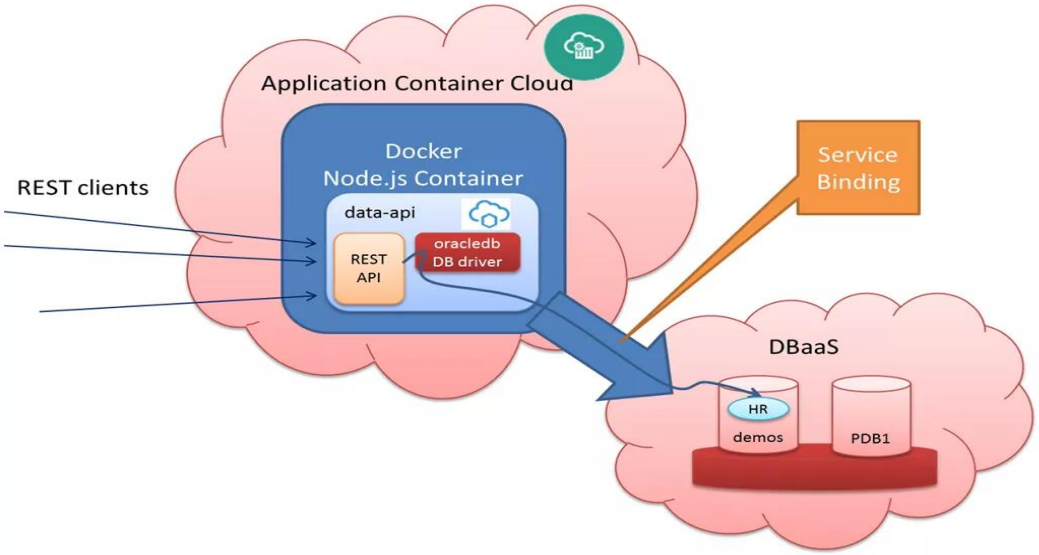
- PaaS: Plataforma como serviço. Este serviço disponibiliza ao usuário uma plataforma completa de desenvolvimento, como serviço, onde o usuário recebe as máquinas com a capacidade computacional e todas as ferramentas necessárias para desenvolvimento de software.

- SaaS: Software como serviço. Este serviço disponibiliza ao usuário uma solução completa. Aplicações na nuvem que o usuário se autentica e utiliza, sem a necessidade de instalar e desenvolver nada. Ex.: Google Docs, conjunto de

ferramentas para criação de textos, planilhas, cronogramas, que o usuário precisa apenas de se autenticar para utilizar.

- DBaaS: Bancos de dados como serviço. Esse serviço provê a solução de banco de dados para o usuário, o SGBD já vem instalado e configurado, o usuário precisará apenas criar seus objetos, de acordo com a necessidade de suas aplicações. A figura 1 demonstra como seria uma arquitetura de banco de dados como serviço, em nuvem.

Figura 1 - Bancos de dados em nuvem.



Na arquitetura proposta na figura 1, o banco de dados está sendo exposto como serviço em uma nuvem. As aplicações apenas se conectam e fazem uso do serviço, de maneira transparente, sem que seja necessária qualquer intervenção para configuração, instalação e administração desse ambiente.

Algumas das principais características dos bancos de dados em nuvem são:

– Infraestrutura de servidores terceirizada.

- Redução do tamanho das equipes.
- Alocação de recursos sob demanda (Elasticidade).
- Automação de atividades de rotina.
- Integração com outros recursos disponíveis em nuvem.

Os serviços em nuvem possuem, entretanto, alguns riscos:

- Modo de armazenamento: A utilização do sistema de arquivos, bem como as políticas de conformidade, fica a cargo do fornecedor da nuvem, o que pode gerar algum risco de acesso indevido ou incompatibilidade com os padrões de segurança dos clientes.
- Acesso aos dados pelo fornecedor: O fornecedor de nuvem administra o ambiente e, portanto, tem acesso aos dados de maneira irrestrita. Os clientes devem fazer uso de políticas de criptografia para evitar acesso a dados sensíveis ao negócio.
- Localização física dos dados: Muitas vezes os clientes não sabem onde seus dados estão armazenados.
- Dados confidenciais: Dados confidenciais, por exemplo, cadastros de cartões de crédito, eventualmente podem ficar acessíveis. Políticas adicionais de segurança se fazem necessárias.

A utilização de serviços de bancos de dados em nuvem modifica a forma de gestão dos ambientes, trazendo alguns benefícios, porém existem novos desafios, tais como:

- Disponibilidade: Manter a disponibilidade dos serviços não é um desafio para o cliente, mas sim para o fornecedor. Os serviços em nuvem fazem replicação das informações, distribuindo o ambiente em vários locais, evitando indisponibilidade.

- Monitoramento: O monitoramento do desempenho da solução de banco de dados deixa de ser uma preocupação, o foco se volta ao monitoramento dos usuários, regras de negócio, perfis de acesso e aplicação das políticas de segurança.

- Desempenho: O desempenho é outra característica que o fornecedor deve garantir. Em nuvem, o conceito de elasticidade visa garantir tempos de resposta adequados, aumentando a capacidade computacional em momentos de pico e reduzindo em momentos de baixa demanda.

- Confidencialidade: A confidencialidade é um requisito sensível ao alternar a solução de banco de dados para a nuvem. O acesso aos dados fica vulnerável aos funcionários da empresa que fornece o serviço em nuvem, porém os SGBD's modernos fornecem funcionalidades de criptografia e máscara dos dados, que devem ser utilizados pelos clientes para aumentar a segurança da informação.

- Gestão de usuários: Os usuários das aplicações e bancos de dados permanecem da mesma forma, os administradores de dados, do cliente, devem ter atenção especial aos acessos administrativos. Políticas de auditoria devem ser adotadas nesse tipo de solução.

- Segurança de rede: O acesso ao banco de dados passa pela internet, portanto um risco adicional. A latência de rede é um requisito que deve ser avaliado na implantação do acesso aos serviços. Se todas as camadas de serviço estiverem

na nuvem, o problema de latência é mitigado. Conexões SSL podem contribuir para melhor segurança.

Sistemas de arquivos distribuídos

Os Sistemas de arquivos distribuídos são soluções onde os dados ficam divididos fisicamente, muitas vezes em locais geograficamente distintos. Esse tipo de solução permite que o acesso simultâneo seja mais eficiente, além de ter os seguintes benefícios:

- Alta disponibilidade: O sistema de arquivos deve permanecer sempre disponível, sem interrupção dos serviços de acesso e escrita das informações.
- Escalabilidade: A escalabilidade em sistemas de arquivos permite que o espaço de armazenamento, bem como de redundância, possa ser expandido dinamicamente.
- Tolerância a falhas: Em situações de falha, o sistema de armazenamento deve possuir mecanismos de recuperação que não causem impacto no acesso aos dados, ou seja, mesmo que um erro aconteça, as informações devem estar acessíveis.
- Redundância: O sistema distribuído tem a necessidade de realizar cópias dos dados em locais diferentes, para garantir a disponibilidade do serviço. A redundância é um requisito importante, pois garante todas as características mencionadas acima.

Os sistemas de arquivos distribuídos fornecem mecanismos de segurança, tais como controle de acesso de usuários, permissões de leitura e escrita em nível de

arquivos e criptografia dos dados, essencial para evitar o acesso indevido. Essas características são configuráveis e permitem que o acesso aos dados tenha a restrição adequada, bem como a hierarquização necessária.

Além de mecanismos de segurança, existem outras ferramentas de controle de acesso. Na maioria dos casos o mecanismo de cache, no lado do cliente, permite que o servidor possua menos carga, além de prover melhor desempenho para os usuários.

O controle de acesso realizado possui travamento de arquivos, onde o usuário acessa os arquivos e apenas ele pode efetuar modificações até que ocorra a liberação desse arquivo.

A replicação dos dados ocorre de forma automática. O cluster além de realizar a réplica dos arquivos, controla as alterações, fazendo com que cada modificação seja refletida em todas as réplicas.

Google File System

O Google, em razão do enorme volume de dados que possui, desenvolveu um sistema de arquivos próprio para atender às demandas de suas aplicações.

A inexistência de soluções de bancos de dados de mercado que suportassem o volume de dados do Google, fez com que a alternativa da empresa para manter seus dados fosse desenvolvida internamente. Era necessário um sistema de arquivos escalável, com baixa latência e que suportasse petabytes de dados.

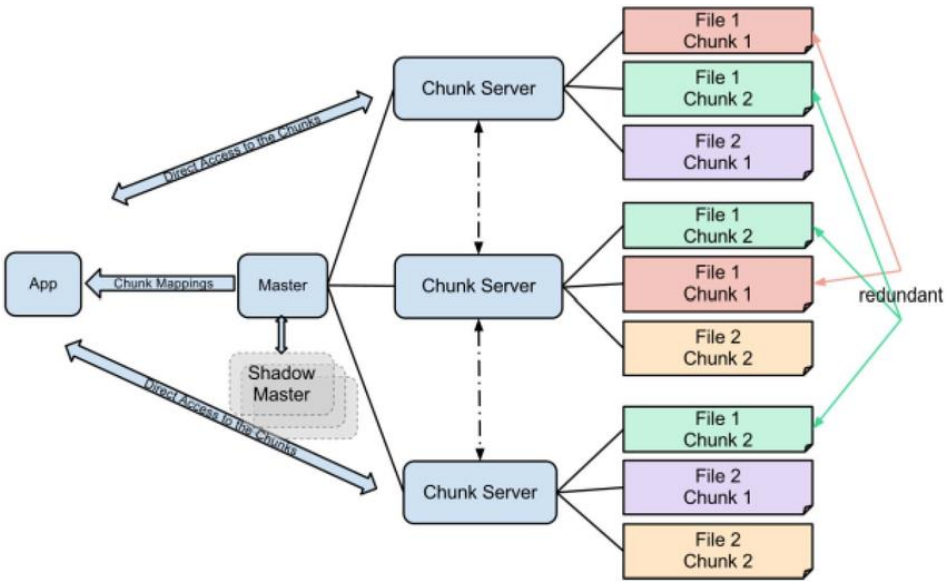
O Google File System foi construído com o objetivo de prover desempenho no armazenamento e recuperação de suas informações, e é um software privado. O GFS é parte da arquitetura das soluções de serviços da Google.

Considerando o perfil de suas aplicações, a Google definiu as seguintes características para seu sistema de arquivos:

- Tolerância a falhas.
- Recuperação automática.
- Foco em arquivos grandes (A maioria em GB).
- Crescimento rápido do conjunto de dados.
- Anexações recorrentes (record append).
- Leitura sequencial de arquivos.

A figura 2 demonstra a arquitetura do Google File System:

Figura 2 - Google File System.



Os principais componentes, demonstrados na figura 2, são:

- Master Server: Componente responsável pela coordenação do cluster. Este componente mantém os Metadados dos arquivos e realiza a orquestração dos chunkservers, além de controlar a replicação dos dados.

- Chunkserver: Componente responsável pelo armazenamento dos arquivos

- Chunk: Componentes similares a blocos de dados. Um arquivo pode conter um ou mais chunks. Um chunk possui partes de um arquivo, porém um arquivo pode conter vários chunks. O chunk é o menor nível de informação presente no GFS.

Uma aplicação, ao requisitar um arquivo, acessa o Master Server e solicita o arquivo através de seu nome. Uma vez que o Master Server localiza o chunkserver onde o arquivo se encontra, o Master Server retorna o caminho do chunkserver para aplicação e esta acessa o arquivo diretamente.

O Google File System, possui as seguintes operações:

- Leitura: A operação de leitura ocorre quando a aplicação solicita, através do nome do arquivo, as informações relacionadas aos dados. O Master responde para o cliente o caminho de uma das réplicas das informações solicitadas, e a partir desse momento a aplicação consegue manipular os arquivos diretamente.

- Gravação: A aplicação solicita permissão para escrever um arquivo. O Master Server retorna o “chunk handle”, para que o arquivo possa ser manipulado, bem como o caminho das réplicas daquele arquivo. O cliente confirma as alterações no arquivo e os arquivos atualizados são enviados para todas as réplicas. Para realizar a escrita, no entanto, é necessário que todos os nós que

possuem cópias do arquivo emitam sinal de autorização para escrita, evitando, assim, inconsistências.

- Snapshot: Criação de cópias de um arquivo ou uma árvore de diretório com baixo custo, ou seja, quase que em tempo real para minimizar interrupções de mudanças.

- Record Append: Realização do controle de acesso a um arquivo, permitindo acesso concorrente, garantindo a sequência de alterações de maneira semelhante ao modo sequencial.

- Garbage Collector: Quando um arquivo é excluído no GFS, a exclusão física não é feita imediatamente. Ocorre a exclusão lógica e periodicamente, o garbage collector faz uma varredura nos níveis de chunks a fim de procurar dados já excluídos e realiza a exclusão física.

O GFS possui um mecanismo de geração de logs. O sistema de logs registra todas as instruções de modificação de dados, de modo a manter um histórico de alterações em ordem cronológica. Os logs são utilizados em operações de recuperação, onde os dados não haviam sido atualizados, as instruções contidas no log são reexecutadas e as informações, desta forma, não são perdidas.

O Google File System é um sistema de arquivos que modificou a forma como as aplicações mantêm as informações, utilizando clusters com milhares de máquinas, provendo alta disponibilidade, escalabilidade e desempenho. O GFS foi uma arquitetura referência para outras soluções de sistemas de arquivos.

HDFS - Hadoop Distributed File System

O Hadoop Distributed File System (HDFS) é um sistema de arquivos distribuído desenvolvido com o objetivo de implementar a arquitetura do Google File System,

através de uma solução de código aberto, que se tornou um projeto da Apache Software Foudation.

O HDFS, inspirado no GFS, foi projetado considerando os seguintes requisitos:

- Funcionar de maneira distribuída, em hardware de baixo custo.
- Ser tolerante a falhas.
- Possuir bom desempenho na transferência de dados.
- Considerar grandes volumes de dados.

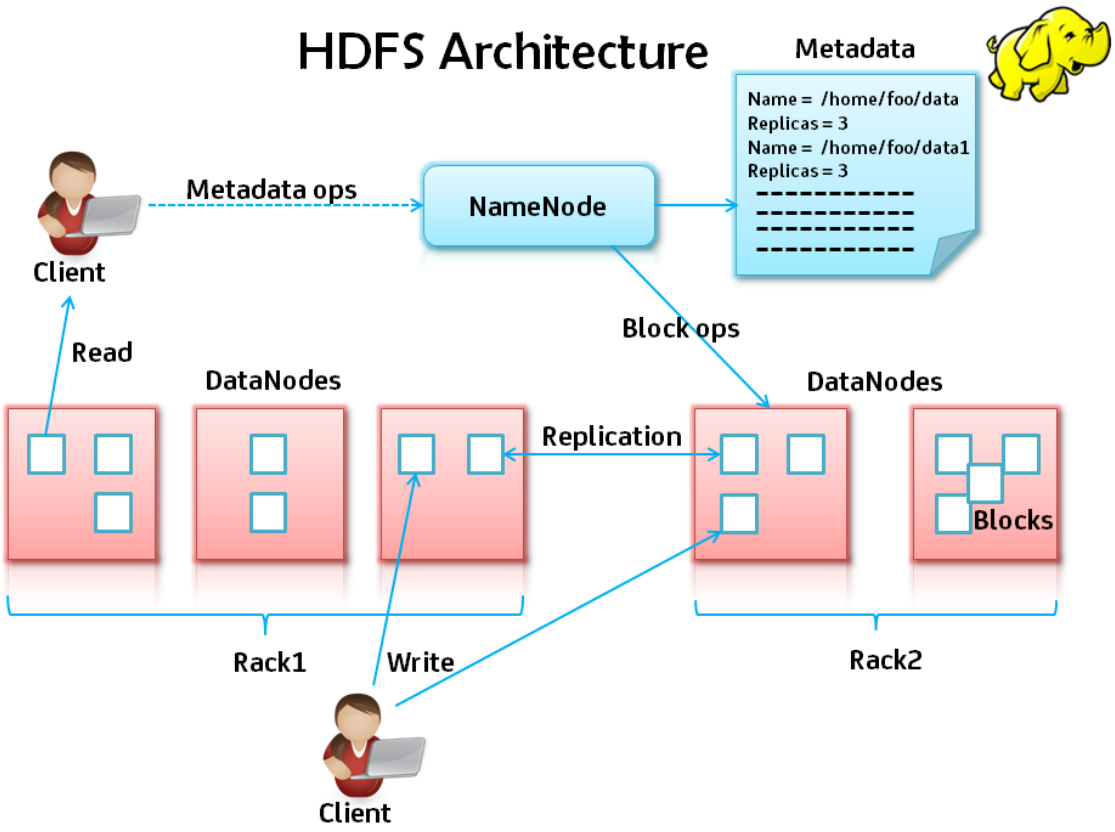
O HDFS possui um modelo simples, com poucos componentes e poucas operações de manipulação, facilitando a gestão do ambiente. O conceito de processamento é a transferência dos algoritmos de processamento para os locais onde os dados estão, invertendo a lógica de processamento e reduzindo drasticamente o tráfego dos dados.

O HDFS possui três componentes principais:

- NameNode: Esse componente é o responsável pelo gerenciamento do sistema. O NameNode possui o armazenamento dos Metadados e as definições de acesso, tais como permissões e grupos de usuários.
- DataNode: Os DataNodes são os conjuntos de blocos de dados, geralmente existe um DataNode para cada nó do cluster.
- Namespace: É o nome lógico do sistema de arquivos, exposto aos usuários do sistema.

A figura 6 demonstra a arquitetura do HDFS, quando um cliente solicita um serviço de leitura ou escrita, que é intermediado pelo NameNode. Uma vez que o NameNode indica uma das réplicas para acesso dos clientes, o acesso ocorre diretamente ao DataNode indicado.

Figura 3 - Arquitetura HDFS.

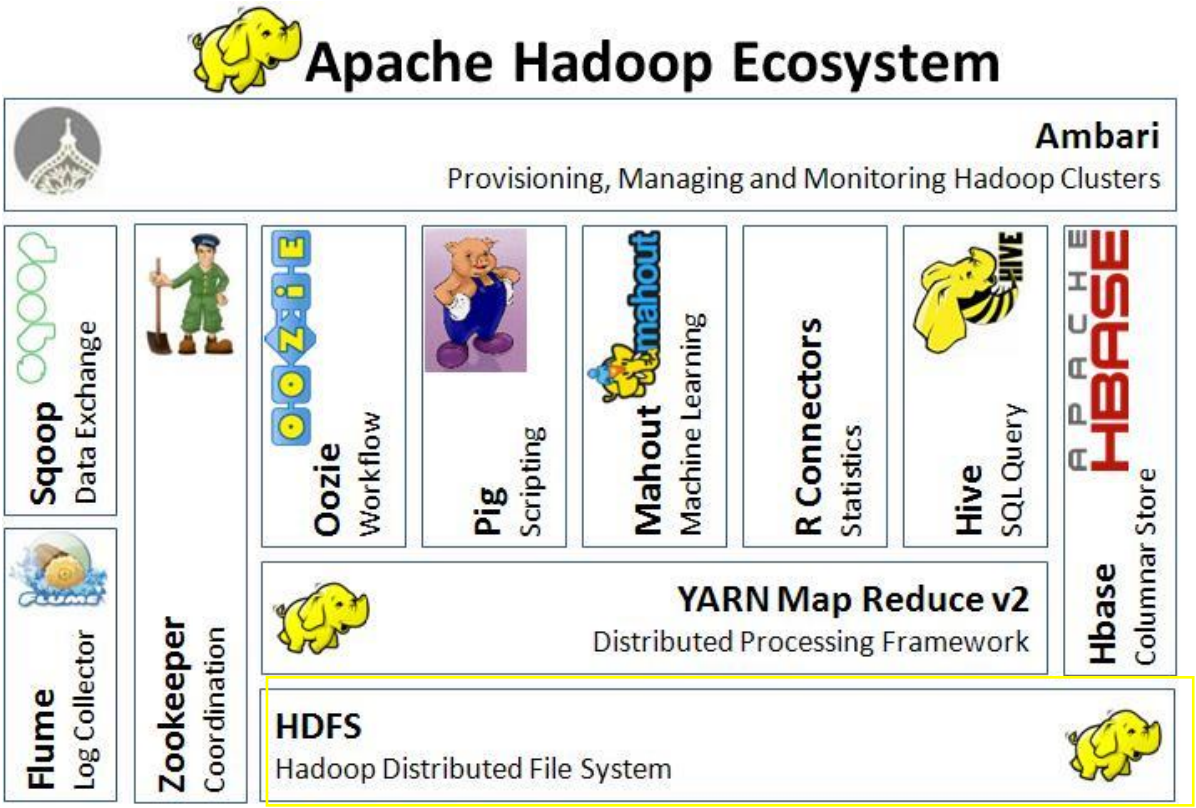


A replicação dos dados é realizada pelo NameNode periodicamente. Por padrão, as réplicas são armazenadas em racks distintos, para minimizar os riscos inerentes às falhas no sistema.

O HDFS é o sistema de arquivos utilizado por grande parte das soluções do ecossistema Hadoop, conforme demonstrado na figura 4. Existem, atualmente, soluções

que armazenam os dados diretamente no HDFS e, também, soluções que fazem a interface de aplicações com o HDFS, provendo facilitadores de consultas.

Figura 4 - Ecossistema Hadoop.



Abaixo algumas aplicações que utilizam o HDFS como sistema de arquivos:

- Facebook.
- Adobe.
- EBay.
- Google.
- IBM.

- ImageShack.
- Last.fm.
- LinkedIn.

O HDFS, portanto, é um sistema de arquivos desenvolvido como solução distribuída, tolerante a falhas e escalável. É um sistema de arquivos baseado na solução Google File System, com o objetivo de prover capacidade de armazenamento elevada, baixa latência e baixo custo de hardware. Suporta a maioria das aplicações do ecossistema Hadoop.

Amazon S3

O Amazon Simple Storage Service, conhecido como Amazon S3, é um armazenamento de objetos baseado em web service. O objetivo do projeto foi prover um serviço de armazenamento em nuvem. O Amazon S3 é um produto do portfólio de soluções em nuvem da Amazon.

O Amazon S3 é um serviço, em nuvem, de armazenamento escalável, com suporte a versionamento de arquivos, bem como acesso via HTTP ou HTTPS.

O acesso aos arquivos pode ser realizado através de protocolos de webservices, como REST e SOAP, facilitando integração entre sistemas, por exemplo.

A solução S3 com o objetivo de garantir alta disponibilidade realiza replicação dos arquivos em outros nós e realiza o monitoramento de arquivos corrompidos, de modo a corrigir os erros nestes arquivos.

O S3 pode ser utilizado em diferentes situações:

- Backup e recuperação: O S3 oferece recursos de versionamento de dados, bem como ciclo de vida das cópias dos dados, permitindo uma política de backup, com ciclos de retenção de dados.

- Análise de dados: O S3 oferece integração com soluções do ecossistema Hadoop, permitindo arquiteturas de análise de grandes volumes de dados, gerando indicadores e tendências nos dados.

- Arquitetura de armazenamento híbrida: Existem algumas implementações de armazenamento híbrido, onde a empresa mantém parte de seus dados em infraestrutura própria e outra parte dos dados em nuvem. Esse tipo de abordagem define o local de armazenamento de uma informação de acordo com a sua relevância, dados menos importantes ou pouco acessados podem ser enviados à nuvem, enquanto dados sensível e de acesso frequente na infraestrutura interna das empresas.

A estrutura de dados do S3 possui os seguintes conceitos principais:

- Buckets: É o componente que agrupa os arquivos armazenados. Cada arquivo inserido no sistema está relacionado a um bucket.

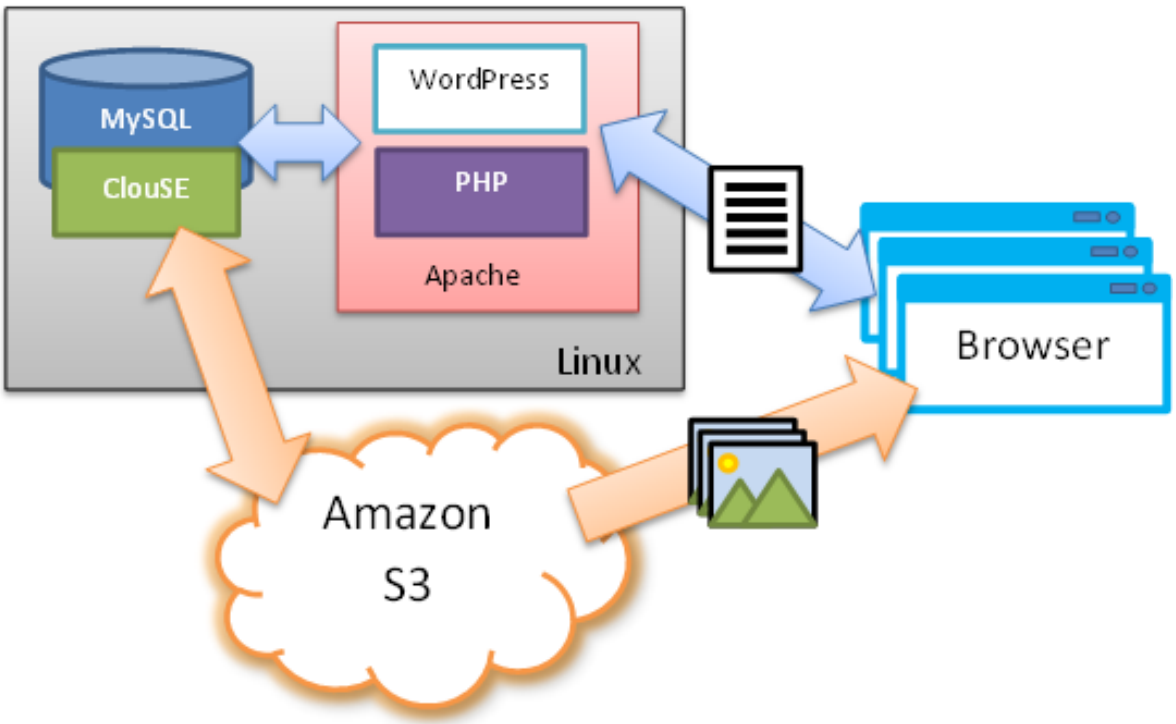
- Objetos: Os Metadados do sistema são armazenados nos “objects”.

- Chaves: A chave é um valor único, que identifica um bucket. Um objeto pode ser acessado através da combinação do endereço do serviço na web com o nome do bucket. Exemplo: *http://doc.s3.amazonaws.com/teste/AmazonS3.wsdl*
  “doc” é o nome do bucket e “teste/amazons3.wsdl” é a chave.

- Regiões: As regiões são os locais, geograficamente definidos, onde os buckets são armazenados. O usuário pode, manualmente, definir quais os locais de sua preferência.

A Figura 5 demonstra uma aplicação onde o S3 é comumente utilizado. Em serviços de blog, como o Wordpress, como solução de armazenamento adicional. O S3 possui integração com esses serviços, que por padrão possuem limitações de armazenamento. Em alguns blogs, o serviço de armazenamento S3 é utilizado para manutenção dos dados.

Figura 5 - Solução de armazenamento adicional – S3 Wordpress.



Empresas que utilizam o S3:

– Netflix: Data Lake da plataforma de vídeos.

- Airbnb: Utiliza o S3 como solução de backup.
- Thomsom Reuters: Utiliza como infraestrutura para seus sistemas de Analytics.

Google Cloud Storage

O Google Cloud storage é uma solução de armazenamento de dados, em nuvem, da Google, com proposta semelhante ao Amazon S3.

A plataforma de serviços em nuvem da Google, o Google Cloud Platform, opera de forma redundante, em diversos datacenters, em várias regiões do mundo. O objetivo do produto é fornecer toda a solução de infraestrutura e plataforma de uma empresa como serviço. A solução permite o desenvolvimento e integração de novos softwares às soluções da Google.

A plataforma de nuvem da Google, possui 5 grandes segmentos:

- Computação: Serviço de virtualização que disponibiliza máquinas virtuais configuráveis e escaláveis para diferentes tipos de usuários. Serviço e IaaS cobrado por tempo de utilização que possui integração com a solução Google Kubernetes para gerenciamento de aplicativos baseados em microsserviços, conforme demonstrado na Figura 6. O app Engine, deste mesmo componente, é um motor de criação de aplicações web com persistência em SGBDs NoSQL. Possui ferramentas de desenvolvimento, tais como: Eclipse, IntelliJ, Maven, Git, Jenkins e PyCharm.
- Armazenamento e Banco de dados: Serviços de armazenamento de dados, como storage em nuvem, mas também soluções de bancos de dados. Ex.: Cloud Bigtable.

- Rede: Serviços da camada de rede e integração. Possui serviços de balanceamento de carga para aplicações na nuvem e também o protocolo interconnect, para integração entre aplicativos da nuvem Google.

- Big Data e Internet das Coisas: Esses produtos fornecem soluções de armazenamento e consultas de dados, com a linguagem SQL, bem como ferramentas de processamento de fluxos de dados, como o Data Flow.

- Aprendizado de Máquina: Possui bibliotecas para aprendizagem de máquina, como o TensorFlow, ferramenta utilizada no treinamento de redes neurais. Permite a elaboração de modelos de classificadores, com diferentes algoritmos para treinamento.

A figura 7 demonstra uma arquitetura que integra vários serviços da plataforma em nuvem da Google, com entrada de dados sendo processada pelos aplicativos da Google App Engine, desde a transferência das informações através do DataFlow, armazenamento no Cloud Storage e BigQuery até o uso das informações pelas ferramentas de Big Data e Machine Learning.

Figura 6 - Arquitetura de gestão de microserviços.

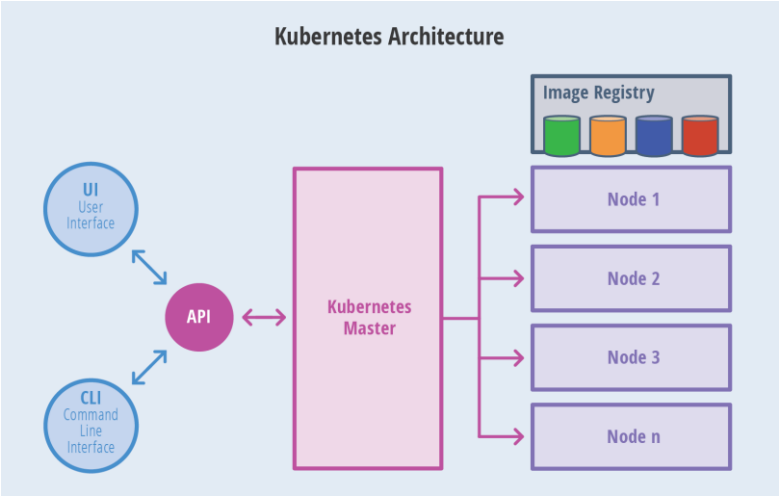
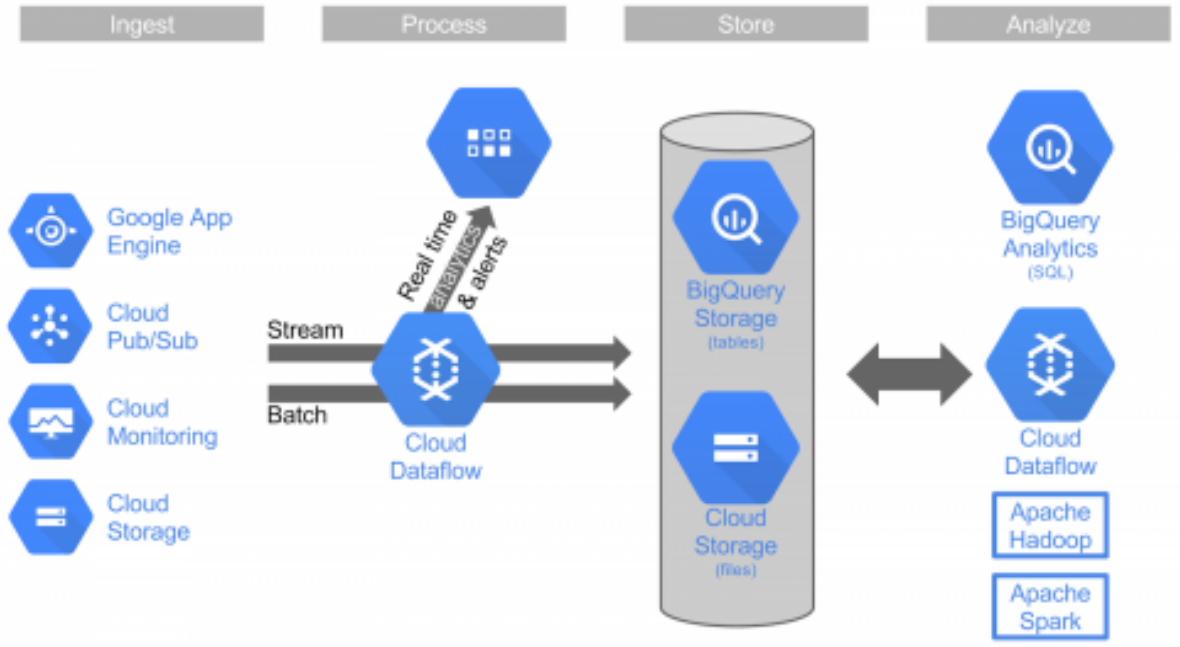


Figura 10: Arquitetura de solução utilizando Google Cloud Platform.



Bancos de dados distribuídos

A distribuição dos dados, processamento e armazenamento oferece algumas vantagens e desafios. Os principais fornecedores de SGBDs de mercado possuem soluções distribuídas.

A origem dos SGBDs distribuídos foi motivada, principalmente, após a internacionalização de grandes empresas. Nesse contexto, alguns problemas referentes aos dados foram identificados:

- Cada filial terá uma cópia dos dados?
- Cada filial terá uma parte dos dados?

– Como armazenar os dados comuns entre filiais?

– Como minimizar o tráfego de dados entre as filiais?

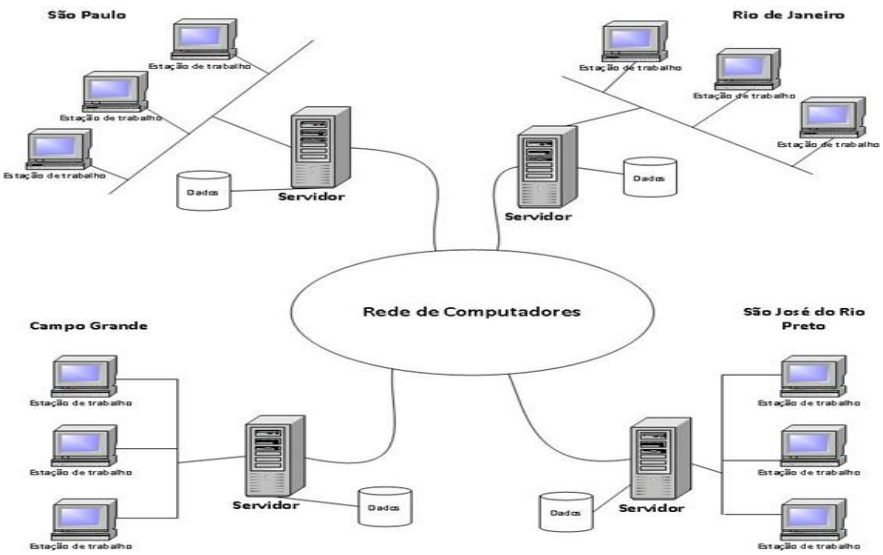
– Como manter a integridade referencial entre as filiais?

Os bancos de dados distribuídos surgiram, nesse contexto, como solução para o aumento da disponibilidade dos dados, manter a proximidade de uma filial com seus dados, facilidade para escalabilidade dos dados, inclusão de novas filiais, problemas decorrentes de empresas distribuídas geograficamente por vários países.

Os bancos de dados distribuídos foram criados para realizar a interligação de vários bancos de dados, através de uma estrutura única de rede, mantendo o processamento local.

Os sistemas gerenciadores de bancos de dados distribuídos, SGBDD, gerenciam os bancos de dados clusterizados ou distribuídos, mantendo a integração e sincronismo dos dados. A figura 8 demonstra uma implementação de bancos de dados distribuídos.

Figura 8 - Arquitetura de Bancos de dados distribuídos.



A forma de implementação dos SGBDDs trouxe novos desafios:

- Como fazer o controle de acesso concorrente em ambiente distribuído?
- Como realizar de forma eficiente a replicação dos dados através do cluster?
- Como garantir as propriedades ACID no banco de dados?

O controle de concorrência é o mecanismo que garante o acesso aos dados de maneira sincronizada, de modo a garantir que as operações realizadas por diferentes usuários sejam concluídas de maneira consistente, mantendo a cronologia e veracidade no resultado final. O controle de concorrência é utilizado para garantir as propriedades ACID:

- Atomicidade: Propriedade que garante que uma transação é indivisível, ou seja, deve ser integralmente concluída ou totalmente desfeita ao seu final, não são permitidas alterações parciais nos dados.
- Consistência: É a garantia de que as integridades relacionais definidas no modelo de dados serão respeitadas ao final de cada transação.
- Isolamento: Cada transação acontece sem interferência de outras, ou seja, deve ocorrer como se fosse única naquele conjunto de dados.
- Durabilidade: É a garantia de que os dados serão corretamente armazenados e atualizados após as transações de banco de dados.

A implementação do controle de concorrência e garantia das propriedades ACID, foram criados novos protocolos, entre eles:

- Two-phase-commit (2PC)

- Locks (Bloqueios)
- Two-phase-locks (2PL)

O protocolo two-phase-commit é um protocolo que separa a operação de confirmação (commit) dos dados seja realizada em duas etapas:

1. Preparação
    a. A transação envia uma mensagem preliminar, para “avisar” que irá acontecer um commit.
    b. Cada nó recebe a mensagem e responde se consegue consolidar essa alteração localmente.
2. Confirmação
    a. Após a resposta de todos os nós do SGBD, o gerenciador de transações deve decidir se confirma as alterações.
    b. Se um nó solicita que a transação deve ser abortada, então a transação será abortada em todos os nós.
    c. Se todos os nós confirmarem a confirmação da transação, então o commit acontece.

Outro protocolo muito utilizado, tanto em SGBDs quanto em SGBDDs, são os locks. É um mecanismo simples, que bloqueia um conjunto de registros, após solicitação de uma transação, e apenas a transação que possui o bloqueio pode modificar os dados:

- A transação que vai modificar os dados solicita o bloqueio desses dados ao nó gerenciador.

- O nó gerenciador solicita aos demais nós, o bloqueio desses registros para outras transações.
- O nó gerenciador controla os objetos bloqueados
- Quando a transação recebe a confirmação do bloqueio, realiza as alterações.
- O bloqueio é retirado quando o commit ou rollback da transação acontece.

As soluções de bancos de dados distribuídos de mercado implementam os conceitos de formas variadas, algumas soluções distribuem os dados, outras o processamento e ainda existem soluções que implementam as duas, além dos mecanismos de controle de transações distribuídas.

Google Big Table

O Google Bigtable é o sistema gerenciador de banco de dados, projetado pela Google, para suportar:

- Aplicações com grandes volumes de dados semiestruturados.
- Resolver problemas de escala nas aplicações da Google.
- Não existia nenhum SGBD capaz de suportar o volume de dados e transações.

Otimizado com o sistema de armazenamento da Google, Google File System, o que provê desempenho.

O Google Big Table é um SGBD que possui capacidade de manipulação de grandes volumes de dados, que é executado de maneira distribuída em milhares ou

milhões de máquinas em um cluster, para diferentes tipos de aplicações e que possui suporte a operações MapReduce.

Principais características:

- SGBD tolerante a falhas.
- Modelo de dados orientado a colunas.
- Escalabilidade:
    - Milhares de servidores;
    - Terabytes de dados em memória;
    - Petabytes de dados em disco;
    - Milhões de operações de leitura e escrita.
- Ajustes dinâmicos.
- Compactação de dados em disco.

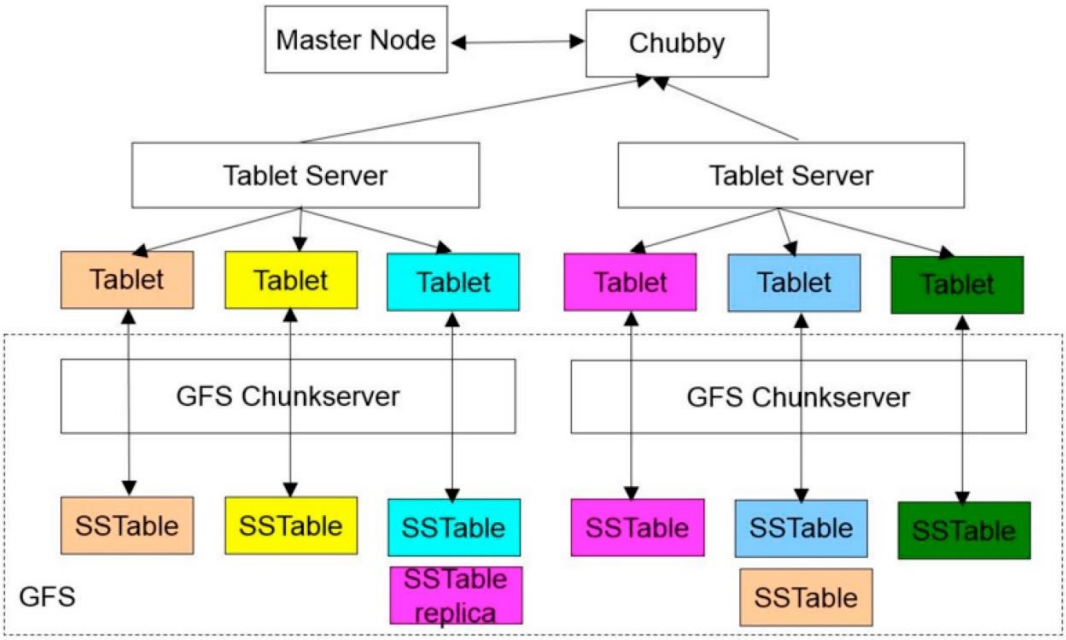
Principais componentes:

- Chubby:
    - Realiza o controle de concorrência;
    - Implementa política de Locks;
    - Mantém a permissão de acessos.
- Google File System:

- Armazenamento dos dados fisicamente;
- Balanceamento de carga em nível de disco.

A arquitetura do Google Big Table está demonstrada na Figura 9:

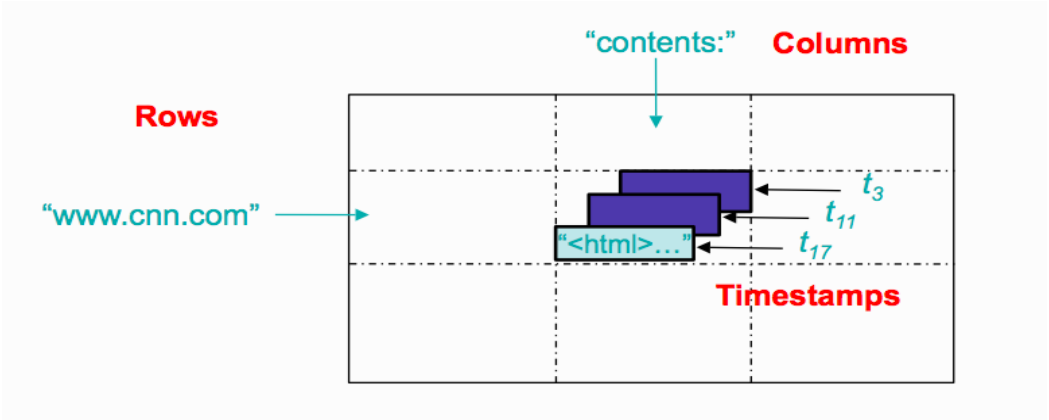
Figura 9 - Arquitetura do Google Big Table.



O modelo de dados do Google Big Table é baseado em famílias de colunas, onde as colunas ficam agrupadas de acordo com o acesso, ou seja, colunas que são consultadas frequentemente juntas, devem permanecer juntas em seu armazenamento.

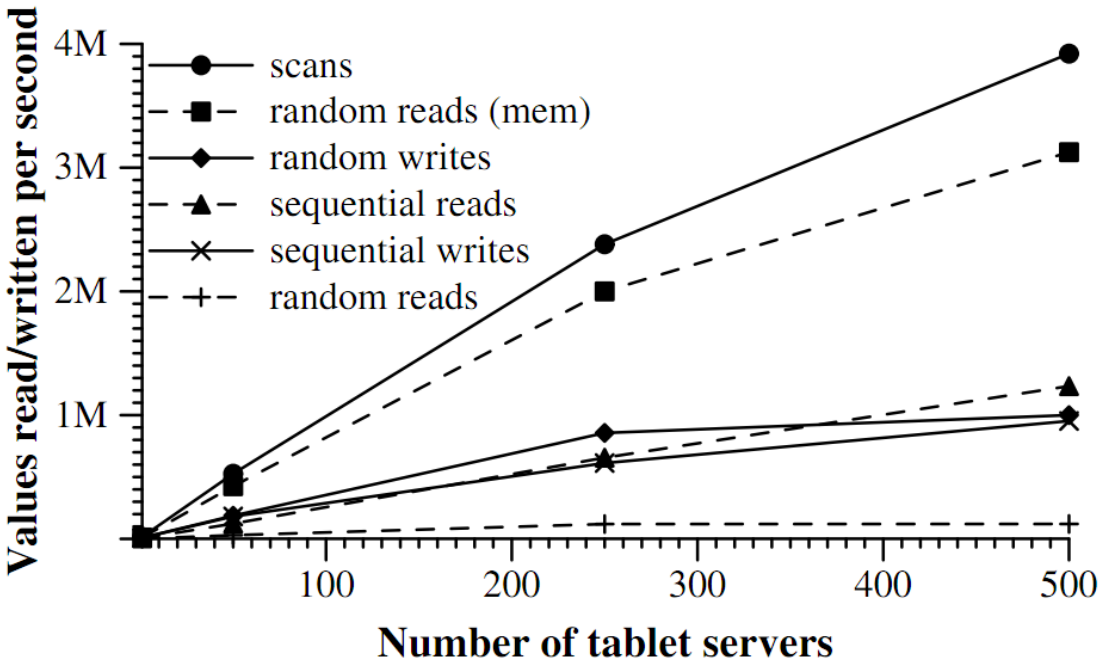
As colunas são indexadas por linha, coluna e timestamp que identifica a versão da informação contida na coluna. A figura 10 demonstra o esquema de armazenamento e versionamento do modelo de dados do Google Big Table.

Figura 10 - Modelo de Dados – Google Big Table.



Conforme demonstrado na figura 11, o Google Big Table aumenta o número de servidores dinamicamente de acordo com a demanda, mantendo o desempenho no mesmo nível, antecipando a perda de performance, mantendo a linha abaixo de funções lineares.

Figura 11 - Análise de desempenho x aumento de demanda.



O Google Big Table possui mecanismos de balanceamento de carga, algoritmo de rebalanceamento que reduz a movimentação de registros em disco e, também, mantém grande parte dos registros em memória, tornando as leituras mais eficientes.

O Big Table é o SGBD desenvolvido para atender à demanda de SGBD da Google, possui mecanismos de escalabilidade, integração nativa com o Google File System, modelo de dados flexível, baseado em colunas, que serviram de modelo para desenvolvimento de outras soluções NoSQL.

SGBD Distribuído Oracle

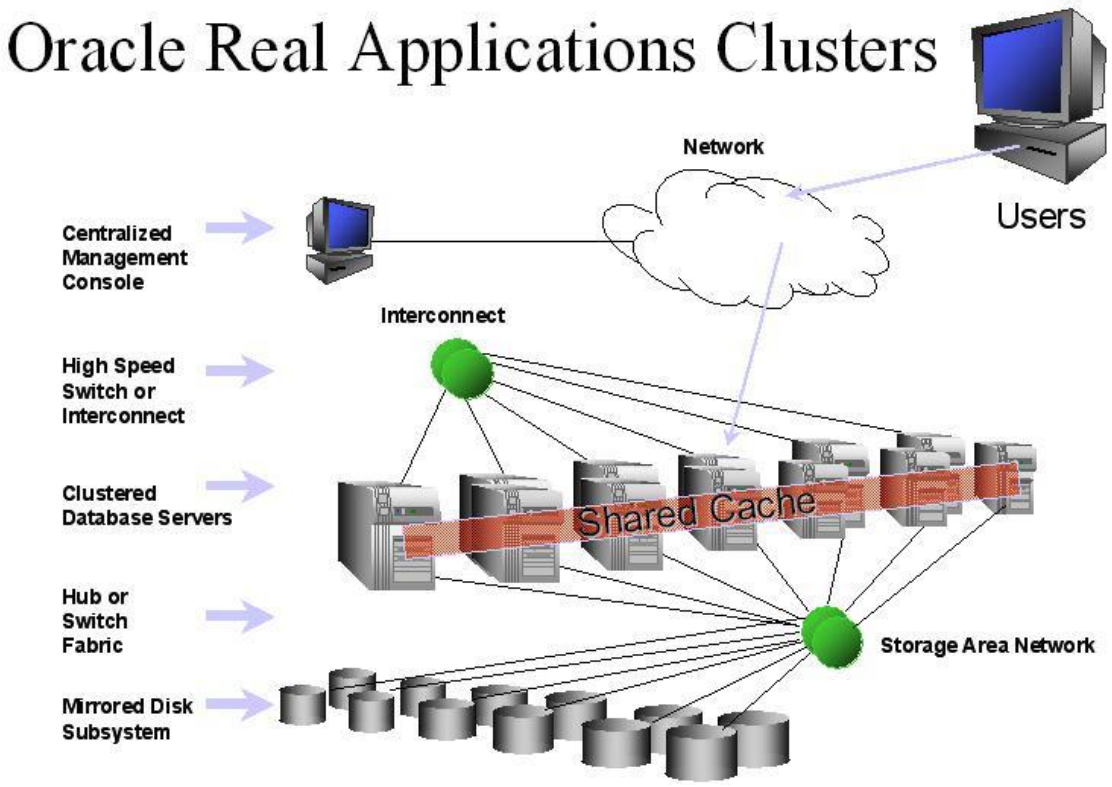
O SGBD Oracle, a partir da sua versão 9i, com a crescente demanda de usuários, necessitava de alternativas para suportar maiores volumes de dados, em decorrência do crescimento de empresas multinacionais.

O desafio, portanto, era desenvolver um SGBD redimensionável, que possibilitasse menor gasto em licenças e eliminação de ponto central de falhas.

O nome adotado para essa solução foi o Oracle Real Application Clusters, o Oracle RAC. Uma arquitetura de banco de dados clusterizada.

A figura 12 demonstra a arquitetura do Oracle RAC, com vários nós de processamento e um ponto de armazenamento. Nesse exemplo, o armazenamento possui espelhamento, mas existem situações em que o armazenamento é realizado em um único disco, em desenho mais simplificado.

Figura 12 - Arquitetura Oracle RAC.



O Oracle RAC possui os seguintes componentes principais:

- Oracle Clusteware: Componente desenvolvido para realizar a integração de forma transparente e uniforme entre os servidores. Esse componente é um pré-requisito para a instalação do Oracle RAC. Realiza o monitoramento dos nós do cluster, inicializando automaticamente as instâncias e listeneres e agindo em situações de falha, reiniciando nós problemáticos.

- Automatic Storage Management (ASM): O sistema de arquivos da Oracle, é um componente opcional no Oracle RAC, porém possui melhor gerenciamento e desempenho nas operações de leitura e escrita. Possui, também, balanceamento de carga interno e funciona em todos os sistemas operacionais suportados pela Oracle.

- Endereço VIP: Endereço de IP interno, exclusivo do Cluster, que cada nó recebe. A faixa de IP Vip não é utilizada na rede local onde o SGBD está instalado. Com o IP virtual, a recuperação de erros ocorre mais rapidamente, pois não há a necessidade de aguardar a recuperação dos componentes de rede, o IP virtual é atribuído ao nó e com esse endereço já é possível iniciar a instância.

- Interconnect: O interconnect é um protocolo que realiza a troca de mensagens e faz a função de heartbeat e keep alive, identificando se os nós estão respondendo.

- Automatic Diagnostic Repository: Repositório centralizado, onde os eventos e problemas do SGBD são registrados. Esse repositório centralizado facilita a identificação de problemas por parte dos administradores do banco de dados.

O Oracle RAC possui as seguintes vantagens:

- Alta disponibilidade: Com vários nós no cluster, quando ocorre problema em algum dos nós, outros nós assumem as atividades do nó problemático.

- Escalabilidade: O Oracle RAC oferece a possibilidade de incluir novos nós na estrutura do banco de dados.

- A Oracle possui ferramentas de gestão com grande variedade de informações e indicadores.

- O Oracle RAC possui balanceamento de carga, equilibrando as requisições entre os nós do cluster.

A Oracle possui, ainda, versões do banco de dados em nuvem, na solução Oracle Database Coud Service, que fornece alta disponibilidade e escalabilidade na solução de banco de dados, tanto em arquitetura padrão, quando no Oracle RAC.

SGBD Distribuído SQL Server

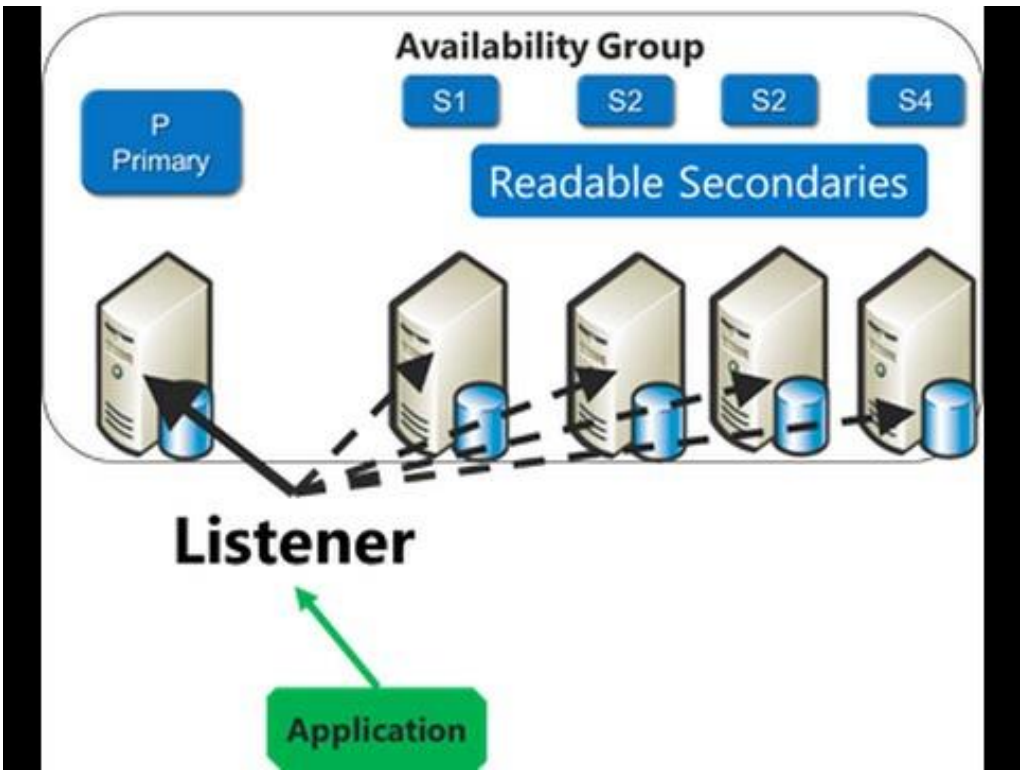
A Microsoft desenvolveu o banco de dados SQL Server, com o nome Always On, para prover serviços de bancos de dados em alta disponibilidade. AlwaysOn é o nome dado ao mecanismo de alta disponibilidade, baseado em replicação de dados.

Grupos de Disponibilidade:

- Um grupo de disponibilidade suporta um ambiente replicado para um conjunto de bancos de dados, conhecidos como bancos de dados de disponibilidade.
- Cada grupo de disponibilidade possui um banco de dados primário e até 08 réplicas ou bancos secundários.
- O banco primário realiza as transações dos usuários, OLTP normalmente, e envia seus logs ao banco secundário.

A figura 13 demonstra a arquitetura Always On, que define um banco de dados aberto aos usuários e aplicações, concentrando as operações OLTP e que, através desse banco central, as informações são replicadas para outras bases secundárias, de “Stand By”.

Figura 13 - Arquitetura SQL Server Always On.



O SQL Server possui dois modos de disponibilidade:

- Asynchronous-commit mode: Registra as transações sem confirmação. Os logs com as informações das transações são enviados aos nós secundários, mas o nó principal não aguarda a confirmação dessa tarefa.
- Synchronous-commit mode: Registra as transações após confirmação das instâncias secundárias.

Tipos de Failover:

- Failover manual planejado: Um failover manual ocorre depois que um administrador de banco de dados emite um comando de failover e faz com

que uma réplica secundária sincronizada faça a transição para a função primária.

- Failover automático: Um failover automático ocorre em resposta a uma falha que faz com que uma réplica secundária sincronizada faça a transição para a função primária.

As instâncias secundárias podem ser utilizadas para que o backup seja realizado, eliminando o consumo de recursos do banco de dados principal.

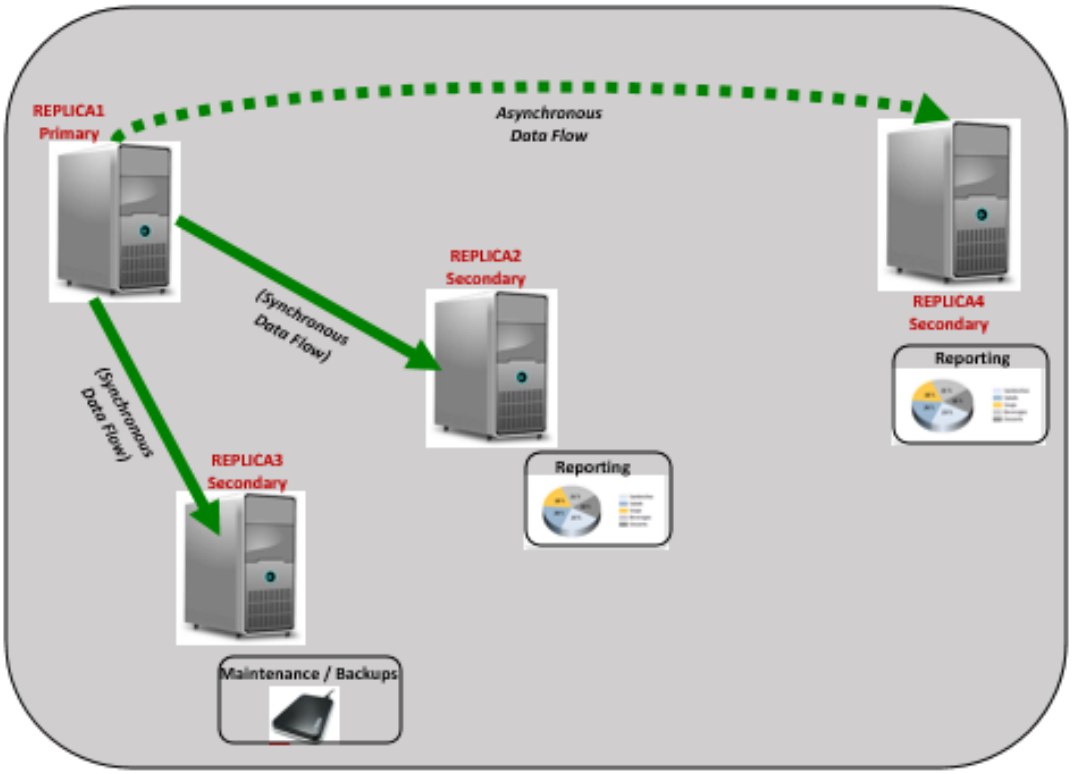
As instâncias secundárias podem, também, ser acessadas em modo somente leitura, permitindo que sejam utilizadas para sistemas de relatórios ou sistemas de business intelligence.

Benefícios do Always On:

- Apoio à política de disaster recovery: A alteração do nó principal pode permitir a recuperação do sistema mais rapidamente.
- Alta disponibilidade: Caso ocorram problemas com o nó principal, automaticamente outro nó se torna o nó principal.
- Tempo de recuperação menor: A eliminação da atividade de retornar backups torna a recuperação mais rápida.
- Transparência em tarefas de manutenção: As intervenções de manutenção podem não ser percebidas pelos usuários, uma vez que se alterna o nó principal e as atividades são executadas sem impactar as transações das aplicações.

A figura 14 demonstra uma eventual arquitetura, onde os nós secundários são utilizados para sistemas de relatórios e operações de backup, mantendo o nó principal com as transações OLTP.

Figura 14 - Arquitetura SQL Server Always On.



O SQL Server Aways On é uma solução de alta disponibilidade da Microsoft, que garante alta disponibilidade do banco de dados. Não é integralmente um cluster, pois o processo não é distribuído, mas pode ser utilizado para melhorar o desempenho, utilizando bases secundárias como leitura.

O sincronismo dos dados é realizado por logs, de forma automática, mantendo as bases secundárias com as mesmas informações da base principal.

Referências

APACHE Cassandra. Disponível em: <https://cassandra.apache.org/>. Acesso em: 27 dez. 2021.

ELMASRI, Ramez et al. **Sistemas de banco de dados**. São Paulo: Pearson Addison Wesley, 2005.

MONGODB. Disponível em: <https://www.mongodb.com/>. Acesso em: 27 dez. 2021.

SILBERSCHATZ, Abraham; SUNDARSHAN, S.; KORTH, Henry F. **Sistema de banco de dados**. Elsevier Brasil, 2016.

BROWNE, Julian. Brewer's CAP Theorem. **:julianbrowne**, 11 jan. 2009. Disponível em: <http://www.julianbrowne.com/article/viewer/brewers-cap-theorem>. Acesso em: 27 dez. 2021.

NEO4J. Disponível em: <https://neo4j.com/>. Acesso em: 27 dez. 2021.