



Armazenamento de Dados

Bootcamp: Engenheiro de Dados

Ricardo Brito Alves

2021

Armazenamento de Dados

Bootcamp: Engenheiro de Dados

Ricardo Brito Alves

© Copyright do Instituto de Gestão e Tecnologia da Informação.

Todos os direitos reservados.

Sumário

Capítulo 1. Banco de Dados Relacional	8
Banco de Dados	8
Aplicação de Banco de Dados	8
Sistema de arquivos	9
SGBD – Sistema de Gerenciamento de Banco de Dados	10
Finalidade dos SGBDs.....	11
Banco de Dados Relacional.....	12
Benefícios dos Bancos de Dados Relacionais.....	13
Razões para usar Bancos de Dados Relacionais	14
Transações	14
Componentes Banco de Dados Relacional.....	15
Abstração de dados	15
Modelagem de Dados.....	16
Normalização de Dados.....	19
Formas normais	19
Desnormalização de Dados	20
Principais conceitos de Entidade - Relacionamento	21
Entidades	21
Atributos.....	21
Relacionamentos	23
Linguagem SQL.....	24
Capítulo 2. Banco de Dados NoSQL.....	26
Visão geral de NoSQL	26
Propriedades dos Bancos de Dados.....	26

Propriedades ACID	26
Propriedades BASE	27
CAP	27
Teorema CAP:	28
Escalando um banco de dados relacional	29
Recursos de bancos de dados NoSQL	30
Equívocos comuns de bancos de dados NoSQL	34
Por que usar o NoSQL?	36
Propriedades dos Bancos de Dados	37
Visão geral de NewSQL	39
Recursos e Técnicas em Bancos NoSQL	41
Tipos de Bancos NoSQL	46
NoSQL Chave-valor	46
NoSQL orientado a documentos	47
NoSQL Colunares	48
NoSQL Grafos	49
Ferramentas de mecanismo de consulta para NoSQL	50
Motivações do uso do NoSQL	50
Alguns bancos NoSQL	52
Capítulo 3. Armazenamento de Dados em Nuvem e Sistemas de Arquivos	54
Banco de Dados em Nuvem - DBaaS	54
Movimentando seus bancos de dados para a nuvem	55
Modelos em Nuvem	57
Introdução a AWS	59
Google Cloud Platform	61

Microsoft Azure	64
Arquitetura microserviços	65
Sistemas de Arquivos Distribuídos	68
Apache Hadoop	70
MongoDB na Nuvem	72
Capítulo 4. Data Warehouse e Data Lake	74
Definição de BI	74
Data warehouse	75
Data mart	76
Relatórios Ad hoc	76
Integração de dados	78
Definição de ETL	78
Como surgiu o ETL	79
Extração	80
Transformação	81
Carga	82
Importância do ETL	82
Benefícios do ETL	84
Principais conceitos em ETL	85
Ferramentas ETL	88
Big Data	89
Data Analytics	90
Pentaho	91
Visão geral do Pentaho	91
Pentaho Data Integration	91

Benefícios do Pentaho	92
Pentaho Data Integration Suite	93
Etapas do Projeto de BI	93
Componentes críticos para escolha da ferramenta.....	96
Requisitos para ETL	96
Componentes críticos	98
Definição de ambiente	99
Escalabilidade.....	100
Recuperabilidade ou Rerunnability	100
Chaves.....	100
Performance	101
Processamento paralelo	102
Ferramentas.....	103
Modelagem dimensional	103
Modelo Star Schema	104
Modelo Snowflake	105
Modelo Starflake.....	105
OLAP	107
Hierarquia de dimensões	108
Surrogate Key	109
Slowly Changing Dimension (SCD)	109
Tabela Fato.....	111
Fato Transacional	111
Fato Agregada	112
Fato Consolidada.....	112

Fato SnapShot Periódico	112
Fato SnapShot Acumulados	113
Fato sem Fato.....	113
Métricas	113
Tipos de métricas.....	114
Métrica aditiva.....	114
Métrica derivada	115
Métrica semi-aditiva	115
Métrica não-aditiva.....	115
ODS – Operational Data Store.....	115
Diferenças entre ODS e DW	116
Staging area.....	117
Tipos de Staging area	118
Staging 1	118
Staging 2.....	119
Staging 2 Aux.....	119
Tratamento de erros	119
Categoria de erros	120
Tipos de erros	121
Tratamento de erros	121
Referências.....	122

Capítulo 1. Banco de Dados Relacional

Banco de Dados

De acordo com Korth (2016), um banco de dados é uma coleção de dados inter-relacionados representando informações sobre um domínio específico. Ou seja, sempre que for possível agrupar informações que se relacionam e que tratam de um mesmo assunto, podemos dizer que temos um banco de dados.

Segundo Date (2004), um banco de dados, por si só, pode ser considerado como o equivalente eletrônico de um armário de arquivamento, ou seja, um repositório ou recipiente para uma coleção de arquivos de dados computadorizados.

Navathe (2010) afirma que um banco de dados deve possuir as seguintes propriedades implícitas:

- Um banco de dados deve representar algum aspecto do mundo real, o que chamamos de minimundo;
- Um banco de dados é uma coleção logicamente coerente de dados com algum significado inerente;
- Um banco de dados é projeto, construído e populado com dados para uma finalidade específica.

Bancos de dados possuem alguma ligação com o mundo real, com a fonte que gera as informações, com usuários e com algum ator que futuramente possa necessitar das informações armazenadas nele.

Aplicação de Banco de Dados

Para Navathe,

Um banco de dados é uma coleção de dados relacionados. Com dados, queremos dizer fatos conhecidos que podem ser registrados

e possuem significado implícito. De forma direta e simples, podemos dizer que um banco de dados é uma coleção de dados. Já um dado, por sua vez, é um fato que deve ser armazenado, ou seja, persistido e que tem um significado implícito. (NAVATHE, 2010)

Ou seja, um dado é um fato do mundo real que está registrado, como endereço e data. Uma informação é um fato útil que pode ser extraído direta ou indiretamente a partir dos dados, como um endereço de entrega ou uma idade.

Sistema de arquivos

Um sistema de arquivo (file system) é um conjunto de estruturas lógicas, feitas diretamente via software, que permite ao sistema operacional ter acesso e controlar os dados gravados no disco.

Cada sistema operacional lida com um sistema de arquivos diferente, e cada sistema de arquivos possui as suas peculiaridades – limitações, qualidades, velocidade, gerenciamento de espaço, entre outras características. É o sistema de arquivos que define como os bytes que compõem um arquivo serão armazenados no disco e de que forma o sistema operacional terá acesso aos dados. O constante crescimento da capacidade de armazenamento dos discos rígidos contribuiu para a variedade de sistemas de arquivos.

O sistema de gerenciamento por arquivos precede o modelo relacional. Podem ser citados alguns problemas com o uso de SGBDs em arquivos, como:

- Redundância e inconsistência de dados: Dificuldade em manter restrições de integridade automaticamente. Seriam checagens de determinadas condições a serem feitas pelo sistema sobre os dados armazenados.
- Dificuldade de acessar informação: O acesso à informação é dificultado em sistemas de arquivos, pois depende da aplicação e de suas regras de acesso, estando subordinado à sua disponibilidade.

- Anomalias de acesso concorrente (multiusuários): O acesso aos dados em arquivos não é feito com controle de acesso concorrente, o que possibilita a ocorrência de anomalias como dados incorretos no momento da consulta.
- Problemas de segurança e integridade: Sistemas em arquivos não possuem política de acesso aos dados (como concessão de privilégios), causando possíveis problemas de segurança.
- Isolamento de dados: Como não há controle de acesso concorrente, as transações não são isoladas, e uma pode influenciar a outra.
- Problemas de atomicidade: Uma vez que não há controle de transações, elas podem ser executadas de forma não completa, possibilitando inconsistências nos dados.

SGBD – Sistema de Gerenciamento de Banco de Dados

Um Sistema de Gerenciamento de Banco de Dados (SGBD) – ou Data Base Management System (DBMS) – é o conjunto de programas de computador responsáveis pelo gerenciamento de uma base de dados. Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados. O SGBD disponibiliza uma interface para que seus clientes possam incluir, alterar ou consultar dados previamente armazenados. Em bancos de dados relacionais, a interface é constituída pelas APIs (Application Programming Interface) ou drivers do SGBD, que executam comandos na linguagem SQL (Structured Query Language).

Figura 1– SGBD.



Fonte: <https://dicasdeprogramacao.com.br/o-que-e-um-sgbd/>.

Finalidade dos SGBDs

Existem muitas vantagens de se utilizar um SGBD, e cada um deles possui suas características. Você precisa analisar o que atenda melhor a sua organização.

- Reduzir a redundância e a inconsistência nos dados: O SGBD deve garantir que uma determinada informação esteja armazenada em apenas um local. As alterações nos dados deverão ser controladas e automaticamente propagadas a todos os usuários que acessam a informação.
- Reduzir a dificuldade no acesso aos dados: Os SGBDs fornecem linguagens e mecanismos eficientes para que uma determinada informação seja encontrada.
- Problemas de segurança: Os SGBDs possuem usuários e perfis de acesso, de modo que nem todos os usuários tem acesso a todas as informações.

- Garantir o isolamento dos dados: Manter todos os dados gravados no banco de dados com o mesmo tipo de formato, bem como impedir que o dado seja acessado de outro local que não seja o SGBD, garantindo, assim, um único ponto de acesso ao dado.
- Minimizar problemas de integridade: Algumas informações de uma empresa devem seguir algumas regras ou restrições, e o SGBD provê formas mais fáceis de garantir a integridade das informações através dessas restrições.
- Resolver problemas de atomicidade: O SGBD deve garantir todas as operações atômicas. Ou seja, todas as operações ou transações que devem acontecer completamente ou que devem ser desfeitas caso ocorram falhas, são gerenciadas pelo SGBD.
- Outros: Tempo de desenvolvimento de aplicações é reduzido, maior flexibilidade para realizar alterações (independência de dados) e maior economia, informações atualizadas, menor volume de papel.

Banco de Dados Relacional

Um banco de dados relacional é um tipo de banco de dados que armazena e fornece acesso a pontos de dados relacionados entre si. Bancos de dados relacionais são baseados no modelo relacional, uma maneira intuitiva e direta de representar dados em tabelas. Em um banco de dados relacional, cada linha na tabela é um registro com uma ID exclusiva chamada chave. As colunas da tabela contêm atributos dos dados e cada registro geralmente tem um valor para cada atributo, facilitando o estabelecimento das relações entre os pontos de dados.

O modelo relacional significa que as estruturas de dados lógicas — tabelas de dados, exibições e índices — são separadas das estruturas de armazenamento físico. Essa separação significa que os administradores de banco de dados podem gerenciar o armazenamento de dados físicos sem afetar o acesso a esses dados

como uma estrutura lógica. Por exemplo, a renomeação de um arquivo de banco de dados não renomeia as tabelas armazenadas nele.

A distinção entre lógico e físico também se aplica às operações do banco de dados, que são ações claramente definidas que permitem aos aplicativos manipular os dados e as estruturas do banco de dados. As operações lógicas permitem que um aplicativo especifique o conteúdo necessário, e as operações físicas determinam como esses dados devem ser acessados e, em seguida, executa a tarefa.

Para garantir que os dados sejam sempre precisos e acessíveis, os bancos de dados relacionais seguem determinadas regras de integridade. Por exemplo, uma regra de integridade pode especificar que linhas duplicadas não são permitidas em uma tabela, para eliminar o potencial de informações errôneas que entram no banco de dados.

O modelo relacional é composto, basicamente, pelos seguintes elementos: coleções de objetos ou relações que armazenam os dados; um conjunto de operadores que agem nas relações, produzindo outras relações; integridade de dados, para precisão e consistência.

Benefícios dos Bancos de Dados Relacionais

O modelo relacional simples, mas eficiente, é usado por organizações de todos os tipos e tamanhos para uma ampla variedade de necessidades de informações. Os bancos de dados relacionais são usados para rastrear inventários, processar transações de comércio eletrônico, gerenciar grandes quantidades de informações essenciais sobre o cliente e muito mais. Um banco de dados relacional pode ser considerado para qualquer necessidade de informações na qual os pontos de dados se relacionam e devem ser gerenciados de maneira segura e consistente, com base em regras.

Bancos de dados relacionais existem desde os anos de 1970. Atualmente, as vantagens do modelo relacional continuam a torná-lo o modelo mais amplamente aceito para bancos de dados.

Razões para usar Bancos de Dados Relacionais

Quatro propriedades essenciais definem as transações do banco de dados relacional: atomicidade, consistência, isolamento e durabilidade – referidos como ACID – que reduzem possíveis anomalias e protegem a integridade dos dados.

- **Atomicidade** define todos os elementos que compõem uma transação completa do banco de dados.
- **Consistência** define as regras para manter os pontos de dados em um estado correto após uma transação.
- **Isolamento** mantém o efeito de uma transação invisível para outras pessoas até ser confirmada, para evitar confusão.
- **Durabilidade** garante que as alterações dos dados se tornem permanentes quando a transação for confirmada.

Transações

Uma transação é um programa em execução que forma uma unidade lógica de processamento no banco de dados. Uma transação inclui uma ou mais operações de acesso ao banco de dados, o que engloba operações de inserção, exclusão, alteração ou recuperação. O sistema deverá garantir que todas as operações na transação foram completadas com sucesso e que seu efeito será gravado permanentemente no banco de dados — ou a transação não terá nenhum efeito sobre o banco de dados ou sobre quaisquer outras transações.

Componentes Banco de Dados Relacional

Os principais tipos de objetos que fazem parte de um banco de dados relacional são:

- Tabela: estrutura básica de armazenamento no SGBDR.
- Tupla ou Linha / Registro: representa todos os dados requeridos de uma entidade em particular. Cada linha em uma tabela deve ser identificada por uma chave primária, de modo a não haver duplicação de registros.
- Coluna ou Campo: unidade que armazena um tipo específico de dado (valor) – ou não armazena nada, com valor nulo. Esta é uma coluna não-chave, significando que seu valor pode se repetir em outras linhas da tabela.
- Relacionamento: associação entre as entidades (tabelas), conectadas por chaves primárias e chaves estrangeiras.
- Chave Primária (Primary Key/ PK): coluna (atributo) que identifica um registro de forma exclusiva na tabela.
- Chave Estrangeira (Foreign Key / FK): coluna que define como as tabelas se relacionam umas com as outras. Uma FK se refere a uma PK ou a uma chave única em outra tabela (ou na mesma tabela!).
- Restrições (Constraints): propriedades específicas de determinadas colunas de uma relação, se a coluna pode aceitar valores nulos ou não.
- Outros: índices, stored procedures, triggers etc.

Abstração de dados

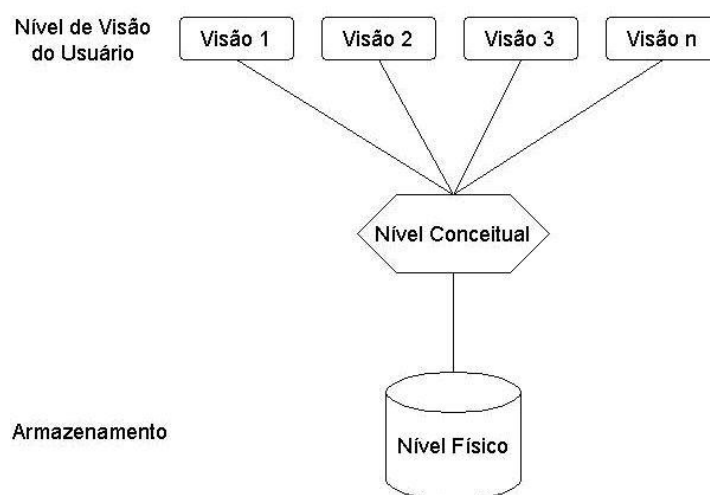
O sistema de banco de dados deve garantir uma visão totalmente abstrata do banco de dados para o usuário. Ou seja, para o usuário, pouco importa qual

unidade de armazenamento está sendo usada para guardar seus dados, contanto que eles estejam disponíveis no momento necessário.

Esta abstração se dá em três níveis:

- Nível de visão do usuário: as partes do banco de dados que o usuário tem acesso de acordo com a necessidade individual de cada usuário ou grupo de usuários;
- Nível conceitual: define quais os dados que estão armazenados e qual o relacionamento entre eles;
- Nível físico: é o nível mais baixo de abstração, que define efetivamente de que maneira os dados estão armazenados.

Figura 2–Níveis de Abstração.



Fonte: <https://www.devmedia.com.br/>.

Modelagem de Dados

A modelagem de dados é o ato exploratório dos dados coletados sobre pessoas e processos. Pode ser usada para uma variedade de objetivos, que vão desde modelos conceituais de alto nível até modelos físicos.

Esse processo crítico consiste em criar estruturas no sistema de armazenamento escolhido (físico ou cloud) para possibilitar a associação e o resgate de informações em um determinado padrão e momento.

É um requisito essencial no desenvolvimento de softwares, pois permite a integração de bancos de dados por sistemas independentes. Um erro durante a modelagem pode comprometer toda a usabilidade do sistema, gerar retrabalho de programação e a necessidade de reformulação de todo o banco, o que sempre aumenta o custo do projeto.

A modelagem de dados contempla todas as atividades relacionadas ao banco de dados para alcançar a um objetivo ou projeto:

- Identifica as entidades e atributos dos dados;
- Aplica uma convenção de nomes para padronizar ou hierarquizar os dados;
- Identifica relacionamentos e aspectos comuns entre os dados;
- Associa chaves para serem requisitados com mais facilidade;
- Normaliza o banco para reduzir a redundância dos dados;
- Diversifica os dados para melhorar o desempenho.

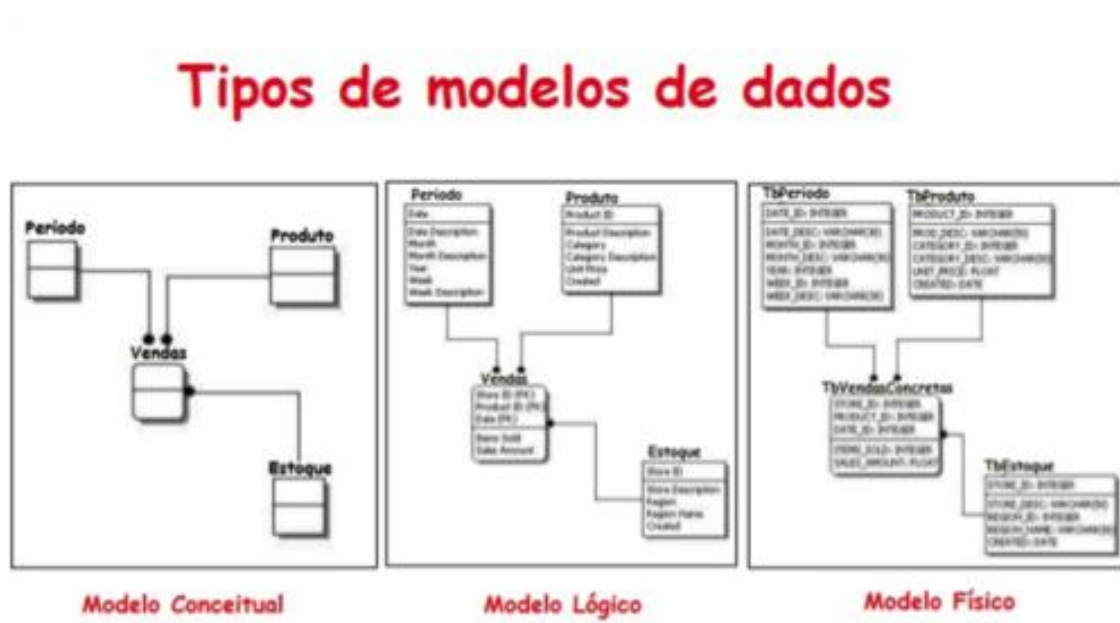
Existem três estilos básicos de modelos de dados.

- **Modelo conceitual:** também chamado de modelo de domínio, é criado para entendimento dos requisitos do sistema, pois explora as estruturas e os conceitos do negócio. É a primeira fase da modelagem, e serve para representar em alto nível as entidades e seus relacionamentos em um nível macro, considerando-se exclusivamente o ponto de vista do usuário gestor dos dados. Sua principal finalidade é capturar os requisitos de informação e regras de negócio sob o ponto de vista do negócio. É independente de

hardware ou software, ou seja, não depende de nenhum tipo de servidor de banco de dados.

- **Modelo lógico:** é usado para explorar os conceitos do domínio. Consiste em determinar quais informações serão necessárias ao banco de dados, divididas em Tabelas. Também serão definidos nesta fase os campos das tabelas, seus atributos e propriedades e ainda as chaves primárias e secundárias, seus relacionamentos, a normalização, a integridade referencial etc. Descreve como os dados serão armazenados no banco e também seus relacionamentos, mas ainda são independentes do SGBD.
- **Modelo físico:** é usado para projetar o esquema interno do banco de dados, e descreve as tabelas, suas colunas e o relacionamento estabelecido entre elas. Consiste na escolha de um SGBD. Parte do Modelo de Entidades e Relacionamentos (MER), que é um modelo abstrato cuja finalidade é descrever, de maneira conceitual, os dados a serem utilizados em um sistema de informações ou que pertencem a um domínio. A principal ferramenta do modelo é sua representação gráfica, o Diagrama Entidade Relacionamento – DER.

Figura 3 – Tipos de Modelo de Dados.



Fonte: <https://pt.stackoverflow.com/>.

Normalização de Dados

O processo de normalização compreende o uso de um conjunto de regras, chamados de formas normais. Ao analisarmos o banco de dados e verificarmos que ele respeita as regras da primeira Forma Normal, então podemos dizer que o banco está na “primeira Forma Normal”. Caso o banco respeite as primeiras três regras, então ele está na “terceira Forma Normal”. Mesmo existindo mais conjuntos de regras para outros níveis de normalização, a terceira Forma Normal é considerada o nível mínimo necessário para grande parte das aplicações. (MICROSOFT, 2007)

Normalização é o processo de modelar o banco de dados, projetando a forma como as informações serão armazenadas a fim de eliminar, ou pelo menos minimizar, a redundância no banco. Tal procedimento é feito a partir da identificação de uma anomalia em uma relação, decompondo-a em relações melhor estruturadas.

Normalmente, precisamos remover uma ou mais colunas da tabela – dependendo da anomalia identificada – e criar uma segunda tabela, obviamente com suas próprias chaves primárias, e relacionarmos a primeira com a segunda para tentarmos evitar a redundância de informações.

Formas normais

Primeira Forma Normal:

Uma relação está na primeira Forma Normal quando todos os atributos contêm apenas um valor correspondente, singular e não existem grupos de atributos repetidos, ou seja, não admite repetições ou campos que tenham mais que um valor. O procedimento inicial é identificar a chave primária da tabela. Então, devemos reconhecer o grupo repetitivo e removê-lo da entidade. Em seguida, criamos uma nova tabela com a chave primária da tabela anterior e o grupo repetitivo. Por exemplo, um atributo Endereço deve ser subdividido em seus componentes: Logradouro, Número, Complemento, Bairro, Cidade, Estado e CEP.

Segunda Forma Normal:

É dito que uma tabela está na segunda Forma Normal se ela atende a todos os requisitos da primeira Forma Normal e caso os registros na tabela, que não são chaves, dependam da chave primária em sua totalidade (e não apenas parte dela).

A segunda Forma Normal trabalha com essas irregularidades e previne que haja redundância no banco de dados. Para isso, devemos localizar os valores que dependem parcialmente da chave primária, criar tabelas separadas para conjuntos de valores que se aplicam a vários registros e relacionar estas tabelas com uma chave estrangeira. Se em algum momento tivermos que alterar o título de um filme, teríamos que procurar e alterar os valores em cada tupla (linha) da tabela. Isso demandaria trabalho e tempo desnecessários. Porém, ao criarmos uma tabela e a vincularmos com o recurso da chave estrangeira, tornamos o nosso banco mais organizado e ágil para as futuras consultas e manutenções que podem vir a ser necessárias.

Terceira Forma Normal:

Se analisarmos uma tupla e não encontrarmos um atributo não chave dependente de outro atributo não chave, podemos dizer que a entidade em questão está na terceira Forma Normal, contanto que esta não vá de encontro às especificações da primeira e da segunda Forma Normal. Como procedimento principal para configurar uma entidade que atenda as regras da terceira Forma Normal, nós identificamos os campos que não dependem da chave primária e dependem de um outro campo não chave. Então, separamos eles para criar uma outra tabela distinta, se necessário.

Desnormalização de Dados

Desnormalização é uma técnica aplicada a bancos de dados relacionais com o objetivo de otimizar a performance de consultas que envolvem muitas tabelas. Esse tipo de consulta normalmente requer a utilização de junções (JOINS)

entre tabelas para obter todos os dados necessários, o que acaba comprometendo o desempenho do banco de dados.

Para contornar esse problema em casos específicos, pode ser viável desnormalizar o banco, juntando os dados em uma única tabela (ou menos tabelas do que as que eram usadas originalmente). Apesar de isso acabar gerando redundância de informações, as aplicações serão beneficiadas com o ganho de desempenho devido a não ser mais necessário unir várias tabelas.

Principais conceitos de Entidade - Relacionamento

Entidades

Entidade pode ser entendida como algo da realidade modelada onde se deseja manter informações no banco de dados. É uma representação concreta ou abstrata de um objeto, com características semelhantes ao mundo real. Se algo existe e proporciona algum interesse em manter dados sobre ele, é caracterizado como uma Entidade do negócio.

Podemos dizer que uma entidade será uma tabela em nosso banco de dados. Geralmente, quando identificamos todas as entidades, estaremos definindo quais serão as tabelas que teremos que criar em nosso banco de dados. Exemplos: fornecedor, pessoa, imóvel, curso, aluno, professores, disciplinas.

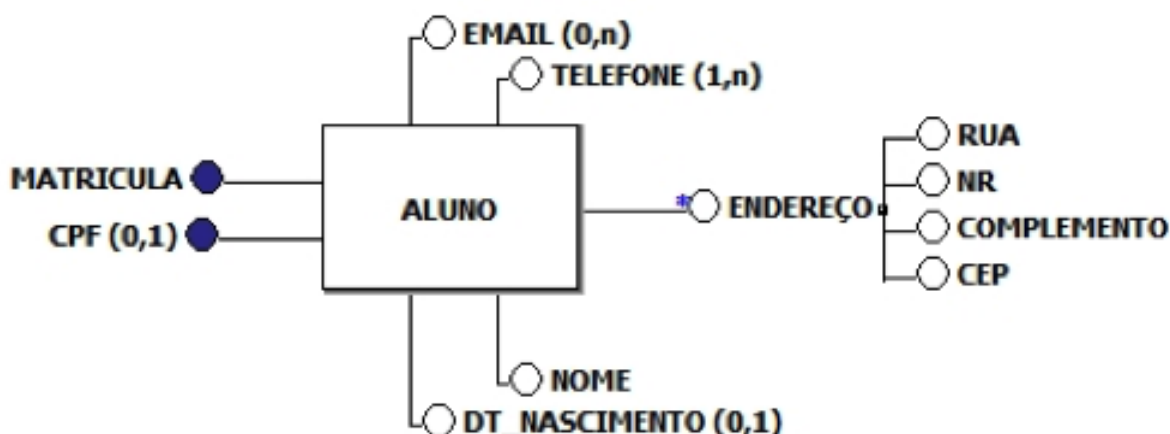
Atributos

São propriedades que identificam as entidades. Os atributos podem ser simples, compostos, únicos, não únicos, opcionais, obrigatórios, monovalorados, multivalorados ou determinantes.

- **Atributo Simples:** a maioria dos atributos serão simples. Quando um atributo não é composto, recebe um valor único como nome, por exemplo: nome, sexo, data de nascimento, dentre outros.

- **Atributo Composto:** é formado por vários itens menores. Exemplo: Endereço. Seu conteúdo poderá ser dividido em vários outros atributos, como: Rua, Número, Complemento, Bairro, Cep e Cidade. Conceitualmente, é aceito o endereço como um único atributo, mas na prática geralmente é feito este desmembramento para permitir a organização dos dados inseridos e facilitar a busca e a indexação destes.
- **Atributo Multivalorado:** o seu conteúdo é formado por mais de um valor, por exemplo: Telefone. Uma pessoa poderá ter mais de um número de telefone.
- **Atributo Determinante:** identifica de forma única uma entidade, ou seja, não pode haver dados repetidos. Exemplo: CNPJ, CPF, Código do fornecedor, Número da matrícula etc. Devemos considerar que toda tabela no banco de dados precisa ter um atributo determinante, que também chamamos de chave primária.
- **Atributo Identificador:** identifica unicamente cada entidade de um conjunto-entidade. Devem ser obrigatórios e únicos, como Cod_Func.
- **Atributo Derivado:** o seu valor pode ser calculado a partir do valor de outro(s) atributo(s). Ex.: idade (pode ser calculada a partir da data de nascimento).
- **Domínio de um atributo:** descrição de possíveis valores permitidos para um atributo. Ex.: Sexo {M, F}.
- **Tipo de um Atributo:** determina a natureza dos valores permitidos para um atributo. Ex.: inteiro, real, string etc.

Figura 4 – Exemplo de Atributo.



Esquema de um banco de dados é a especificação de sua estrutura, e Instância é o conjunto de ocorrências dos objetos de dados de um esquema em um dado momento do tempo.

Relacionamentos

Relacionamento é um conjunto de associações entre entidades.

Cardinalidade do relacionamento

- 1:1 (um para um - uma linha de uma tabela têm apenas um relacionamento com outra linha de outra tabela. Um aluno mora atualmente em um único endereço).
- 1:N (um para n - uma linha de uma tabela pode ter “n” relacionamentos com outra tabela - um pai pode ter “n” filhos)
- N:1 (idem anterior).
- N:N ou N:M (muitos para muitos) - 1 aluno cursa “n” disciplinas e uma disciplina pode conter “n” alunos).

Devemos observar alguns aspectos importantes que sinalizam erros na modelagem. É importante atentar a esses erros para que não haja acúmulo de inconsistências e para não tornar a modelagem um processo problemático.

1. Quando "sobram" entidades sem relacionamentos;
2. Quando ocorrem "ciclos fechados" de relacionamentos, por exemplo: Usuário relaciona-se com Empréstimo, que relaciona-se com Livro, que relaciona-se com Usuário, que relaciona-se com Empréstimo etc.
3. Entidades com muitos atributos relacionando-se com entidades com apenas alguns atributos;
4. Muitas entidades relacionando-se a uma mesma entidade;

Linguagem SQL

O Modelo Relacional prevê, desde sua concepção, a existência de uma linguagem baseada em caracteres que suporte a definição do esquema físico (tabelas, restrições etc.) e sua manipulação (inserção, consulta, atualização e remoção).

Os tipos da linguagem SQL são:

- DDL – Data Definition Language – Linguagem de Definição de Dados. São os comandos que interagem com os objetos do banco. São comandos DDL: CREATE, ALTER e DROP;
- DML – Data Manipulation Language – Linguagem de Manipulação de Dados. São os comandos que interagem com os dados dentro das tabelas. São comandos DML: INSERT, DELETE e UPDATE;
- DQL – Data Query Language – Linguagem de Consulta de dados. São os comandos de consulta. São comandos DQL: SELECT (é o comando de consulta). Aqui cabe um parêntese: alguns autores consideram que o SELECT fica na DML, mas recentemente há a visão que o SELECT faz parte da DQL;

- DTL – Data Transaction Language – Linguagem de Transação de Dados. São os comandos para controle de transação. São comandos DTL: BEGIN TRANSACTION, COMMIT E ROLLBACK;
- DCL – Data Control Language – Linguagem de Controle de Dados. São os comandos para controlar a parte de segurança do banco de dados. São comandos DCL: GRANT, REVOKE E DENY.

Capítulo 2. Banco de Dados NoSQL

Visão geral de NoSQL

Durante a era ponto com (.com), tivemos um uso maciço de bancos de dados relacionais. Em meados dos anos 2000, com a proliferação da Internet, empresas como a Amazon e o Google viram aumentos no tráfego e nos dados. Bancos de dados relacionais, como MySQL, Postgres e Oracle não conseguiam escalar bem.

A Amazon criou o SimpleDB e o Google apresentou o BigTable para superar as limitações do RDBMS. A entrada desses dois bancos de dados não relacionais despertou interesse na comunidade de tecnologia. Em 2009, Johan Oskarsson organizou um encontro para discutir bancos de dados não relacionais distribuídos. Para popularizar este encontro, ele usou uma hashtag #NoSQL no Twitter e isso deu origem aos bancos de dados NoSQL. Neste capítulo, vamos fazer um tour pelas limitações do RDBMS, o funcionamento, as capacidades e os equívocos do NoSQL DBS.

Propriedades dos Bancos de Dados

Propriedades ACID

- Atomicidade – estado em que as modificações no BD devem ser todas ou nenhuma feita. Cada transação é dita como “atômica”. Se uma parte desta transação falhar, toda a transação falhará.
- Consistência – estado que garante que todos os dados serão escritos no BD.
- Isolamento – requer que múltiplas transações que estejam ocorrendo “ao mesmo tempo” não interfiram umas nas outras.

- Durabilidade – garante que toda transação submetida (commit) pelo BD não será perdida.

Propriedades BASE

- Basically Available – Basicamente Disponível.
- Soft-State – Estado Leve.
- Eventually Consistent – Eventualmente Consistente.

Uma aplicação que funciona basicamente o tempo todo (basicamente disponível), que não tem de ser consistente o tempo todo (estado leve), e que o sistema se torna consistente no momento devido (eventualmente consistente).

BDs relacionais trabalham com ACID (Atomicidade, Consistência, Isolabilidade, Durabilidade). BDs NoSQL trabalham com BASE (basicamente disponível, estado leve, eventualmente consistente).

CAP

- Consistency – Consistência.
- Availability – Disponibilidade.
- Partition Tolerance – Tolerância ao Particionamento.

Consistência (Consistent): as escritas são atômicas, e todas as requisições de dados subsequentes obtém o novo valor. Assim, é garantido o retorno do dado mais atualizado armazenado logo após ter sido escrito. Isso independe de qual nó seja consultado pelo cliente – o dado retornado para a aplicação será igual.

Disponibilidade (Available): o banco de dados sempre retornará um valor desde que ao menos um servidor esteja em execução – mesmo que seja um valor defasado.

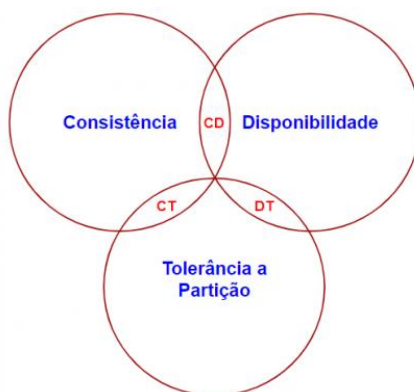
Tolerância ao Particionamento (Partition Tolerant) ou Tolerante a Falhas: o sistema funcionará mesmo se a comunicação com o servidor for temporariamente perdida. Assim, se houver falha de comunicação com um nó da rede distribuída, os outros nós poderão responder às solicitações de consultas de dados dos clientes.

Teorema CAP:

De acordo com o teorema CAP, um sistema distribuído de bancos de dados somente pode operar com dois desses comportamentos ao mesmo tempo, mas jamais com os três simultaneamente.

Consistência Eventual é um conceito interessante derivado do teorema CAP. O sistema prioriza as escritas de dados (armazenamento), sendo o sincronismo entre os nós do servidor realizado em um momento posterior – o que causa um pequeno intervalo de tempo no qual o sistema como um todo é inconsistente. Para isso, são implementadas as propriedades Disponibilidade e Tolerância a Partição.

Figura 5–Teorema CAP.



Exemplos de sistemas de bancos de dados que implementam a consistência eventual são o MongoDB, Cassandra e RavenDB (bancos NoSQL), entre outros.

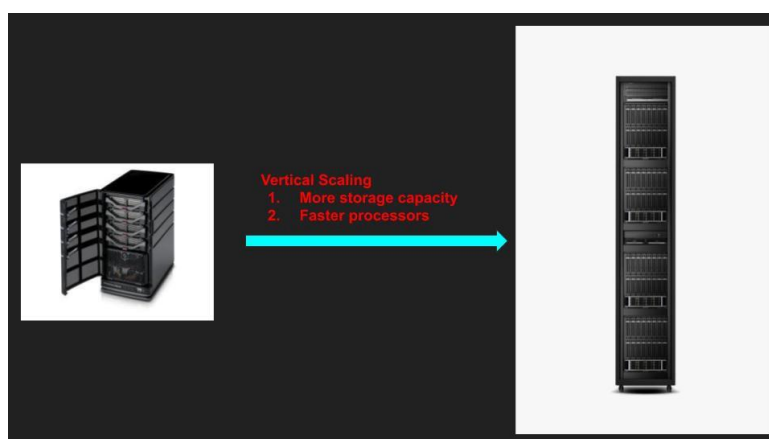
Em Bancos Relacionais, é muito comum implementar as propriedades Consistência e Disponibilidade. Como exemplo, citamos os SGBDRs Oracle, MySQL, PostgreSQL, SQL Server, entre outros.

Ao criar um banco de dados distribuído, é importante ter em mente o teorema CAP. Você terá de decidir se o banco será consistente ou disponível, pois bancos de dados distribuídos são sempre tolerantes à partição.

Escalando um banco de dados relacional

Vamos imaginar que começamos um negócio na Internet em meados dos anos 2000. Nosso negócio demonstra um crescimento exponencial no tráfego do site, e nossos clientes reclamam do carregamento lento das páginas da web. O que faremos a seguir? Pedimos aos nossos DBAs para otimizar as consultas de banco de dados e usar índices para melhorar o desempenho do site. Poucos meses depois, começamos a receber reclamações novamente. A escala vertical agora vem em nosso auxílio, e investir valores a mais na compra de servidores maiores resolve nosso problema.

Figura 6 –Escalonamento.



Existem limites reais sobre o quão longe podemos ir com a escala vertical de nossos bancos de dados. E se nosso site quiser entrar em um novo negócio e armazenar vídeos, imagens, chats e todas as formas de outros dados? A resposta é

simples. Como uma única máquina pode armazenar uma quantidade limitada de dados, temos que recorrer ao escalonamento horizontal. Compramos servidores grandes e distribuímos os dados e o tráfego por essas máquinas. No entanto, bancos de dados SQL não são projetados para escala horizontal. A união de conjuntos de dados e agregação de dados de muitas máquinas introduz complexidade em nosso projeto.

Bancos de dados relacionais tradicionais, como MySQL e PostgreSQL, oferecem suporte a transações A.C.I.D. (Atomicidade, Consistência, Isolamento e Durabilidade). Os bancos de dados No-SQL são compatíveis com B.A.S.E. (Basicamente disponível eventualmente consistente). Vamos fazer um tour pelos recursos dos bancos de dados NoSQL.

Recursos de bancos de dados NoSQL

Abaixo, elencamos alguns dos principais recursos de banco de dados NoSQL.

1. Sem esquema ou com esquema flexível:

Os bancos de dados relacionais têm um esquema rígido, e os usuários precisam passar por várias iterações para modelar os dados. Alterar o tipo de dados de um atributo se torna um pesadelo para desenvolvedores, clientes potenciais e DBAs. O NoSQL supera essa limitação, fornecendo um esquema flexível. Esses bancos de dados abstraem o armazenamento de dados e o trabalho interno dos usuários. Eles fornecem suporte para armazenar estruturas de dados definidas pelo usuário. Por exemplo: os dados podem ser armazenados na forma de um objeto JSON. Os usuários têm flexibilidade para adicionar, substituir ou remover atributos dos dados.

Por serem agnósticos quanto ao esquema, os bancos de dados NoSQL também são denominados bancos de dados schema-on-read. Você só precisa saber como os dados são armazenados durante a leitura dos dados.

O esquema flexível encurta o tempo de desenvolvimento. Você não precisa mais passar por muitas iterações de modelagem e design de dados, e os desenvolvedores podem armazenar e recuperar o que quiserem. A única desvantagem do design sem esquema é que ele aumenta o risco de inconsistência, pois há uma falta de controle.

Os bancos de dados SQL oferecem suporte a valores nulos para colunas. Por exemplo: A página da web de um aplicativo de banco tem muitos campos opcionais, como nome da rua. Se os usuários não preencherem os campos opcionais, o banco de dados ainda reservará espaço para essas colunas, caso os usuários os atualizem no futuro. Em bancos de dados NoSQL, você não passa as entradas nulas e, portanto, o armazenamento é otimizado.

Sem esquema não significa que qualquer lixo aleatório pode ser armazenado no banco de dados. Por exemplo, se uma coluna de banco de dados suporta o tipo de dados JSON, o JSON deve ser bem formatado. O aplicativo obterá um erro se tentar armazenar um objeto JSON mal-formatado.

2. Não Relacional:

Bancos de dados relacionais organizam dados em linhas e colunas. Você pode armazenar dados em muitas tabelas, e as tabelas podem ter relacionamentos diferentes. Para buscar os dados, você pode juntar as tabelas no valor de um atributo. O desempenho do aplicativo diminui quando o número de tabelas a serem unidas chega a dois dígitos ou mais. Há uma queda significativa na velocidade no caso de o aplicativo unir tabelas armazenadas em servidores de banco de dados diferentes.

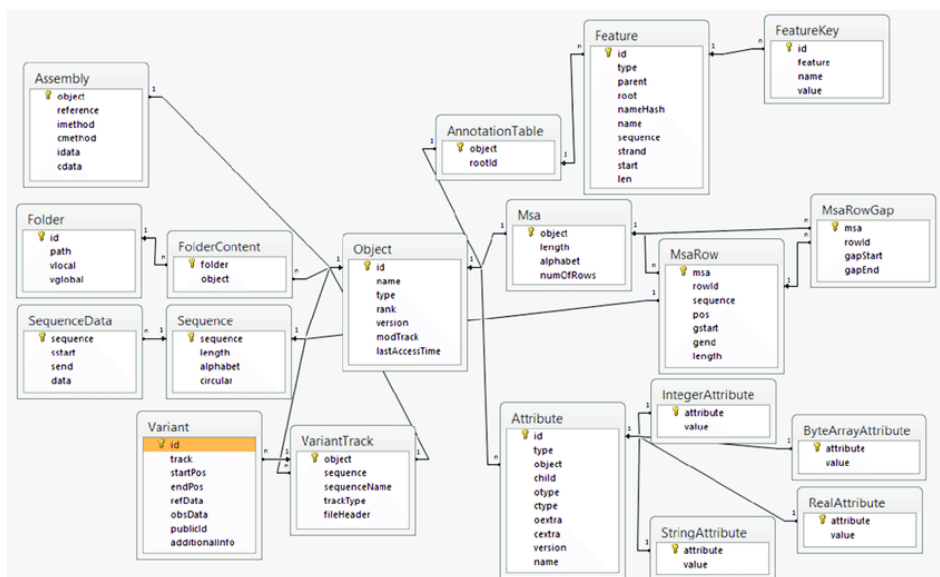
Os bancos de dados NoSQL são desnormalizados. Não existe um conceito de relacionamento entre registros em bancos de dados NoSQL. Isso significa que você pode armazenar os dados agregados em uma única tabela, em vez de espalhá-los por diferentes tabelas.

A seguir estão as principais vantagens da abordagem acima:

- Velocidade de consulta – A velocidade aumenta significativamente, pois apenas uma pesquisa no atributo-chave é necessária e não há necessidade de juntar muitas tabelas;
- Armazenamento e recuperação – basta salvar e obter um único registro.

Por exemplo, ao projetar um aplicativo de entrega de comida usando RDBMS, você criará várias tabelas – uma para usuários, restaurante, pedidos. Em um banco de dados NoSQL, uma única tabela de pedidos pode ter um restaurante, com os dados do usuário duplicados em várias linhas. A desvantagem da duplicação de dados é superada pelos benefícios mencionados acima.

Figura 7–Diagrama.



Você pode evitar criar diagramas ER complexos e escrever consultas SQL complicadas. Com os bancos de dados NoSQL, você pode acelerar seu desenvolvimento e se concentrar em fazer as coisas.

3. Alta escalabilidade e disponibilidade

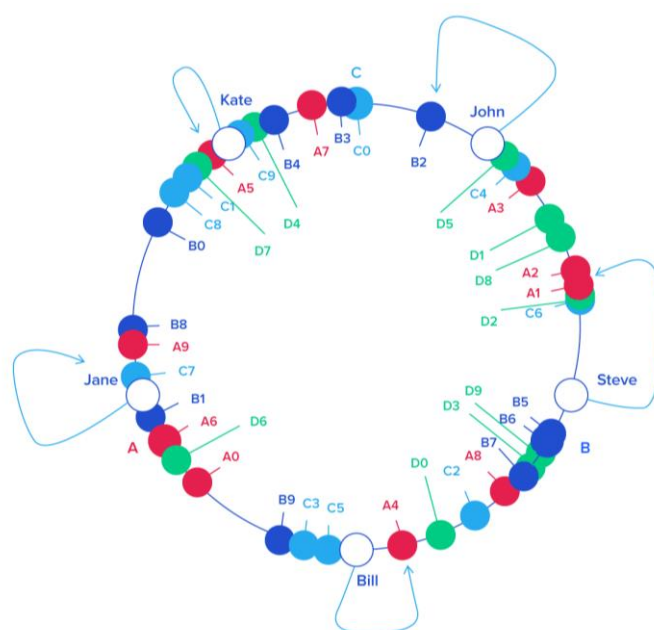
Quando o Google publicou seu artigo sobre o BigTable, ele o definiu como “um sistema de armazenamento distribuído para gerenciar dados estruturados que é

projetado para escalar para um tamanho muito grande”. O banco de dados NoSQL pode armazenar petabytes de dados em muitos computadores.

Com o aumento exponencial na quantidade de dados não estruturados, ficou difícil armazenar dados em uma única máquina. Os bancos de dados relacionais precisam de hardware especializado para atender à carga sem comprometer o desempenho. Portanto, para escalar, tornou-se essencial projetar um sistema que armazenaria dados em um cluster de computadores e os recuperaria de forma eficiente.

Os bancos de dados NoSQL usam servidores comuns, que são mais baratos do que servidores de alto desempenho. Conforme o requisito de armazenamento de dados aumenta, mais servidores de commodities podem ser adicionados. Os bancos de dados NoSQL distribuem os dados uniformemente em um cluster de servidores usando o algoritmo de hash consistente.

Figura 8 – Cluster.



Os bancos de dados NoSQL podem replicar os dados em muitas máquinas. Os dados, ainda, podem estar acessíveis se qualquer um dos servidores morrer ou travar. Portanto, os bancos de dados NoSQL estão altamente disponíveis.

4. Código Aberto

O desenvolvimento de código aberto torna o software NoSQL único. Poucos fornecedores de código aberto lançam um produto desse tipo e vendem recursos complementares corporativos. Essas empresas têm um modelo de negócios semelhante ao da RedHat.

A seguir está uma lista de alguns bancos de dados NoSQL de código aberto:

- MongoDB.
- Apache Cassandra.
- Redis.
- Voldemort.
- HyperTable.
- Neo4j.

Equívocos comuns de bancos de dados NoSQL

1 - NoSQL é um único tipo de banco de dados.

Os bancos de dados NoSQL são classificados quanto ao tipo de dados e quanto a seu funcionamento interno. A seguir estão os diferentes tipos de bancos de dados NoSQL:

- **Chave-valor:** esses bancos de dados funcionam como HashMap e podem armazenar qualquer tipo de valor. Exemplos: Redis, Voldemort e Aerospike.
- **Colunas:** os nomes e o formato das colunas podem variar nas linhas. Cassandra, BigTable e Hypertable são amplos armazenamentos de colunas.

- **Documentos:** bancos de dados como CouchDB, MongoDB e DocumentDB são capazes de armazenar dados na forma de documentos JSON, XML.
- **Grafos:** bancos de dados como Neo4j. Entidades de modelo interno são nós de gráfico, e os relacionamentos entre entidades são indicados por arestas entre os nós.

2 - Risco de perda de dados usando NoSQL.

Uma vez que os bancos de dados NoSQL comprometem a consistência ao invés da disponibilidade, pode haver casos em que cada leitura não segue a gravação mais recente. No entanto, esses bancos de dados são eventualmente consistentes e, portanto, garantem a durabilidade dos dados.

3 - NoSQL é apenas uma palavra da moda.

Amazon, Google, Microsoft, IBM, Oracle e muitas outras grandes corporações construíram bancos de dados NoSQL e estão alavancando seus recursos em sistemas de produção. Grandes empresas de software só investem em tecnologias se tiverem lucro, portanto, o NoSQL não é mais um exagero.

4 - RDBMS aprimorado irá substituir NoSQL.

Recursos altamente distribuídos do NoSQL estão sendo integrados à tecnologia RDBMS, o que resultou no surgimento de muitos bancos de dados NewSQL. Os bancos de dados NewSQL superaram a maioria das críticas relacionadas à tecnologia RDBMS. No entanto, os bancos de dados NoSQL são construídos para resolver diferentes problemas de dados usando diferentes estruturas de dados.

Portanto, NoSQL é um termo genérico que define bancos de dados não relacionais. Não quer dizer que seus modelos não possuem relacionamentos, e sim que não são orientados a tabelas (ou seja, não apenas SQL – Not Only SQL).

Bancos de dados NoSQL são cada vez mais usados em Big Data e, aplicações web de tempo real. Fornecem esquemas flexíveis e escalam facilmente

com grandes quantidades de dados e altas cargas de usuários. São bancos de dados de alto desempenho e altamente funcionais para fornecer excelentes experiências ao usuário. Por exemplo, empresas como Twitter, Facebook, Google coletam terabytes de dados do usuário todos os dias.

NoSQL é um DBMS não relacional, que não requer um esquema fixo, evita junções e é fácil de escalar. O objetivo de usar um banco de dados NoSQL é para armazenamentos de dados distribuídos com enormes necessidades de armazenamento de dados.

O RDBMS tradicional usa a sintaxe SQL para armazenar e recuperar dados para novas percepções. Em vez disso, um sistema de banco de dados NoSQL abrange uma ampla gama de tecnologias de banco de dados que podem armazenar dados estruturados, semiestruturados, não estruturados e polimórficos.

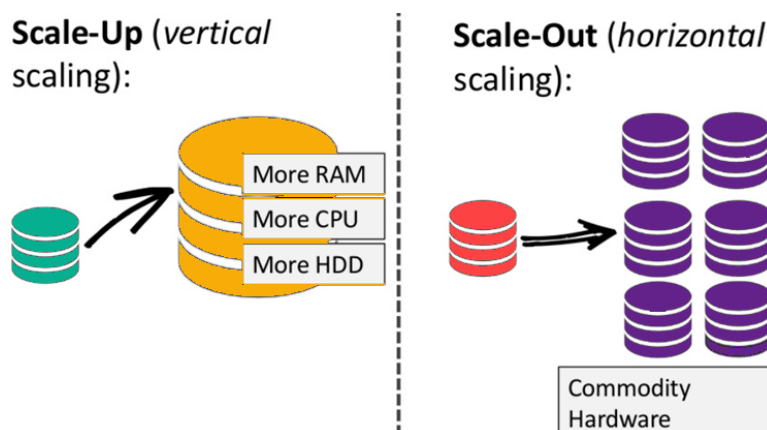
Por que usar o NoSQL?

O conceito de bancos de dados NoSQL se tornou popular entre gigantes da Internet como Google, Facebook, Amazon etc., que lidam com grandes volumes de dados. O tempo de resposta do sistema torna-se lento quando você usa RDBMS para grandes volumes de dados.

Para resolver esse problema, poderíamos “aumentar” nossos sistemas atualizando nosso hardware existente, mas esse processo é caro.

A alternativa para esse problema é distribuir a carga do banco de dados em vários hosts sempre que esta aumentar. Esse método é conhecido como “escalonamento”.

Figura 9 – Escalonamento Vertical e Horizontal.



O banco de dados NoSQL é melhor escalonável do que os bancos de dados relacionais, pois é projetado com aplicativos da web em mente.

Figura 10 – Banco NoSQL.



Propriedades dos Bancos de Dados

Os Bancos de Dados Relacionais são sistemas eficientes, o que os torna uma escolha comum para armazenar registros financeiros, informações logísticas, dados pessoais e outras informações em novas bases de dados. Por serem mais fáceis de entender e de usar do que os bancos de dados NoSQL, os bancos de dados relacionais também substituem frequentemente bancos de dados hierárquicos legados e bancos de dados de rede.

Os Bancos de Dados Relacionais têm as seguintes propriedades:

- Os valores são atômicos.
- Todos os valores em uma coluna têm o mesmo tipo de dados.
- Cada linha é única.
- A sequência de colunas é insignificante.
- A sequência de linhas é insignificante.
- Cada coluna tem um nome único.
- As restrições de integridade mantêm a consistência dos dados em várias tabelas.

O NoSQL é uma alternativa sem esquema para SQL e RDBMSs projetados para armazenar, processar e analisar quantidades extremamente grandes de dados não estruturados.

Nas bases de dados NoSQL, os princípios do ACID (atomicidade, consistência, isolamento e durabilidade) são reduzidos. Além disso, o processo de normalização não é obrigatório no NoSQL. Devido ao tamanho e velocidade dos dados modernos, é preferível que os bancos de dados NoSQL sejam desnormalizados.

Os bancos de dados NoSQL têm as seguintes propriedades:

- Têm maior escalabilidade.
- Usam computação distribuída.
- São econômicos.
- Suportam esquema flexível.
- São capazes de processar dados não estruturados e semiestruturados.
- Não há relações complexas, como as entre mesas em um RDBMS.

A tabela a seguir mostra os tipos de bancos de dados não relacionais e os recursos associados a eles:

Tipo	Desempenho	Escalabilidade	Flexibilidade	Complexidade
Chave-valor	Alto	Alto	Alto	Alto
Colunar	Alto	Alto	Moderado	Baixo
Documento	Alto	Variável para alto	Alto	Baixo
Grafo	Variável	Variável	Alto	Alto

Visão geral de NewSQL

NewSQL é uma classe de sistemas gerenciadores de banco de dados relacionais que busca fornecer a escalabilidade dos sistemas NoSQL para cargas de trabalho de processamento de transações on-line (OLTP), mas mantendo as garantias ACID de um sistema de banco de dados tradicional.

Muitos sistemas corporativos que lidam com dados de alto perfil (por exemplo, sistemas financeiros e de processamento de pedidos) são muito grandes para bancos de dados relacionais convencionais, mas têm requisitos transacionais e de consistência que não são práticos para sistemas NoSQL. As únicas opções disponíveis anteriormente para essas organizações eram comprar computadores mais poderosos ou desenvolver middleware personalizado que distribua solicitações sobre DBMS convencionais. Ambas as abordagens apresentam alto custo de infraestrutura e/ou de desenvolvimento. Os sistemas NewSQL tentam conciliar os conflitos.

O termo foi usado pela primeira vez pelo Group Matthew Aslett em um artigo de pesquisa de 2011 que discutiu o surgimento de uma nova geração de sistemas de gerenciamento de banco de dados. Um dos primeiros sistemas NewSQL foi o sistema de banco de dados paralelo H-Store.

Figura 11 – New SQL.



Aplicações típicas são caracterizadas por grandes volumes de transações OLTP:

- São de curta duração.
- Trabalham com pequenas quantidades de dados por transação.
- Usam pesquisas indexadas (sem varreduras de tabela).
- Têm um pequeno número de formulários (um pequeno número de consultas com diferentes argumentos).

No entanto, alguns suportam aplicações de processamento transacional/analítica híbrida (HTAP). Tais sistemas melhoram o desempenho e a escalabilidade, omitindo recuperação de pesos pesados ou controle de concorrência.

As duas características de distinção comuns das soluções de banco de dados NewSQL são que suportam a escalabilidade on-line dos bancos de dados NoSQL e o modelo de dados relacionais (incluindo a consistência ACID) usando SQL como sua interface primária.

Os sistemas NewSQL podem ser agrupados em três categorias:

- **Novas arquiteturas:** os sistemas NewSQL adotam várias arquiteturas internas. Alguns sistemas empregam um conjunto de nós de nada compartilhado, no qual cada nó gerencia um subconjunto dos dados. Eles

incluem componentes como controle de concorrência distribuída, controle de fluxo e processamento de consulta distribuído.

- **Motores SQL:** a segunda categoria são motores de armazenamento otimizados para SQL. Esses sistemas fornecem a mesma interface de programação do SQL, mas dimensionam melhor do que os motores embutidos.
- **Fragmento transparente:** esses sistemas dividem automaticamente os bancos de dados em vários nós, usando o algoritmo de consenso Raft ou Paxos.

Recursos e Técnicas em Bancos NoSQL

Temos como recursos nos bancos de dados NoSQL:

1. Não relacional.

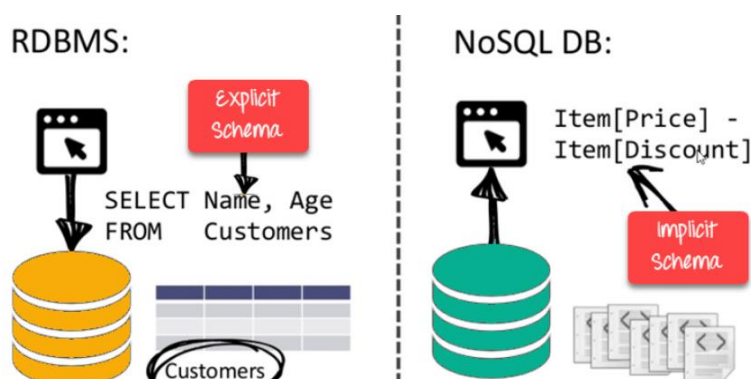
- Os bancos de dados NoSQL não seguem o modelo relacional.
- Não usam tabelas com registros de coluna fixa simples.
- Dados podem ser agregados, autocontidos ou BLOBs.
- Não requerem mapeamento relacional de objeto e normalização de dados.
- Sem recursos complexos, como linguagens de consulta, planejadores de consulta, junções de integridade referencial, ACID.

2. Esquema-livre ou flexível.

- Os bancos de dados NoSQL são livres de esquema ou têm esquemas flexíveis.
- Não requerem nenhum tipo de definição do esquema dos dados.

- Oferecem estruturas heterogêneas de dados no mesmo domínio.

Figura 12 – Esquema NoSQL.



3. API simples.

- Oferecem interfaces fáceis de usar para armazenamento e consulta de dados fornecidos.
- APIs permitem manipulação de dados de baixo nível e métodos de seleção.
- Protocolos baseados em texto usados principalmente com HTTP REST com JSON.
- Geralmente não usam linguagem de consulta padrão.
- Bancos de dados habilitados para web funcionando como serviços voltados para a internet.

4. Distribuído.

- Vários bancos de dados NoSQL podem ser executados de forma distribuída.
- Oferecem recursos de escalonamento automático e failover.
- Muitas vezes, o conceito de ACID pode ser sacrificado para manter a escalabilidade e performance.
- Quase sempre sem replicação síncrona entre nós distribuídos.

- Replicação multimestre assíncrona, ponto a ponto, Replicação HDFS.
- Fornecendo apenas consistência eventual.
- Arquitetura sem muito compartilhamento. Isso permite menos coordenação e maior distribuição.

Sobre as principais características nos bancos de dados NoSQL, é importante ressaltar algumas técnicas utilizadas para a implementação de suas funcionalidades.

Entre elas, estão:

- Map/reduce.
- Consistent hashing.
- MVCC - Multiversion Concurrency Control.
- Vector Clocks.

Map Reduce:

Permite a manipulação de enormes volumes de dados ao longo de nós em uma rede. Funciona da seguinte forma: na fase MAP, os problemas são particionados em pequenos problemas que são distribuídos em outros nós na rede. Quando chegam à fase REDUCE, esses pequenos problemas são resolvidos em cada nó filho e o resultado é passado para o pai, que sendo ele consequentemente filho, repassaria para o seu, até chegar à raiz do problema.

Hashing:

É o processo de mapear um dado (normalmente um objeto de tamanho arbitrário) para outro dado de tamanho fixo (normalmente um inteiro), conhecido como código hash ou simplesmente hash. Uma função é geralmente usada para mapear objetos para código hash conhecido como função hash.

Por exemplo, uma função hash pode ser usada para mapear strings de tamanho aleatório para algum número fixo entre $0 \dots N$. Dado qualquer string, ela sempre tentará mapeá-la para qualquer número inteiro entre 0 e N. Por exemplo, suponha que N seja 100: para qualquer string, a função hash sempre retornará um valor entre 0 e 100.

Hello ---> 60

Hello World ---> 40

Hashing consistente:

Hashing consistente é um tipo especial de hashing, que quando uma tabela hash é redimensionada, apenas as chaves precisam ser remapeadas em média, onde e é o número de chaves e E é o número de slots.

Suporta mecanismos de armazenamento e recuperação em que a quantidade de sites está em constante mudança. É interessante usar essa técnica, pois ela evita que haja uma grande migração de dados entre estes sites, que podem ser alocados ou desalocados para a distribuição dos dados.

Distributed Hashing:

Suponha que um número de funcionários continue crescendo, e que se torne difícil armazenar todas as informações dos funcionários em uma tabela hash que pode caber em um único computador. Nessa situação, tentaremos distribuir a tabela de hash para vários servidores para evitar a limitação de memória de um servidor. Os objetos (e suas chaves) são distribuídos entre vários servidores.

Esse tipo de configuração é muito comum para caches na memória como Memcached, Redis etc.

NAME	AGE	CAR	GENDER
jim	36	camaro	M
carol	37	345s	F
johnny	12	supra	M
suzy	10	mustang	F

PARTITION KEY	MURMUR3 HASH VALUE
jim	-2245462676723223822
carol	7723358927203680754
johnny	-6723372854036780875
suzy	1168604627387940318

MVCC - Multiversion Concurrency Control:

Oferece suporte a transações paralelas em banco de dados. Por não fazer uso de locks para controle de concorrência, faz com que transações de escrita e leitura sejam feitas simultaneamente.

Ao serem iniciados novos processos de leitura de um banco de dados no mesmo instante em que um outro processo está atualizando, pode acontecer que do processo de leitura ver apenas uma parte do que está sendo atualizado, ou seja, um dado inconsistente.

Vector clocks:

Ordenam eventos que ocorreram em um sistema. Como existe a possibilidade de várias operações estarem acontecendo simultaneamente, o uso de um log de operações informando suas datas se faz importante para informar qual versão de um dado é a mais atual.

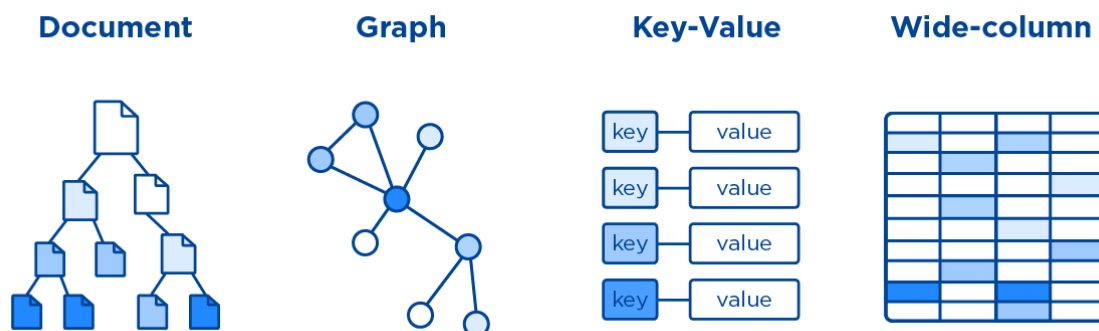
Tipos de Bancos NoSQL

Existem quatro grandes tipos de NoSQL:

- Armazenamento de chave-valor.
- Armazenamento de documentos.
- Banco de dados orientado a colunas.
- Banco de dados de gráficos.

Cada tipo resolve um problema que não pode ser resolvido com bancos de dados relacionais.

Figura 13 –Tipos de NoSQL.

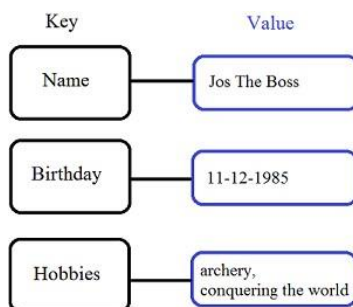


NoSQL Chave-valor

Os dados são armazenados em pares chave / valor. Ele foi projetado para lidar com muitos dados e cargas pesadas.

Os bancos de dados de armazenamento de par de chave-valor armazenam dados como uma tabela hash em que cada chave é única e o valor pode ser JSON, BLOB (objetos binários grandes), string etc.

Figura 14 – Chave Valor.



O valor em um armazenamento de chaves-valor pode ser qualquer coisa: uma string, um número, até um conjunto inteiramente novo de pares de chaves-valor encapsulados em um objeto.

Exemplos de armazenamentos de valores-chave são Redis, Voldemort, Riak e DynamoDB da Amazon.

```
{
  "internal data": [
    {
      "entities": [
        {
          "customer": [
            {
              "id": 1,
              "name": "Freddy"
            },
            {
              "id": 2,
              "name": "Fritz"
            }
          ]
        },
        {
          "legal entities": [
            {
              "id": 1,
              "company": "Maiton"
            }
          ]
        }
      ]
    },
    {
      "Products": [
        {
          "furniture": [
            {
              "id": 1,
              "name": "Octopus Table",
              "stock": 1
            }
          ]
        }
      ]
    }
  ]
}
```

NoSQL orientado a documentos

O armazenamento de documentos é um passo à frente em complexidade em relação aos armazenamentos de valores-chave: assume certa estrutura de documento que pode ser especificada com um esquema. Este parece ser o mais natural entre os tipos de banco de dados NoSQL, porque é projetado para armazenar documentos do dia a dia como estão e permitem consultas e cálculos complexos nesta forma de dados frequentemente já agregada.

A maneira como as coisas são armazenadas em um banco de dados relacional faz sentido do ponto de vista da normalização: tudo deve ser armazenado apenas uma vez e conectado por meio de chaves estrangeiras. Os armazenamentos de documentos se preocupam pouco com a normalização, desde que os dados estejam em uma estrutura que faça sentido. Um modelo de dados relacional nem sempre se encaixa bem com determinados casos de negócios.

Jornais ou revistas, por exemplo, contêm artigos. Para armazená-los em um banco de dados relacional, você precisa primeiro dividi-los: o texto do artigo vai para uma tabela, o autor e todas as informações sobre o autor em outra, e os comentários sobre o artigo quando publicado em um site vão para outra. Conforme mostrado abaixo, um artigo de jornal também pode ser armazenado como uma entidade única; isso diminui a carga cognitiva de trabalhar com os dados para aqueles acostumados a ver os artigos o tempo todo.

Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes são sistemas DBMS populares originados de documentos.

NoSQL Colunares

Bancos de dados orientados a colunas são baseados em papel BigTable do Google. Cada coluna é tratada separadamente, e os valores de bancos de dados de coluna única são armazenados de forma contígua.

ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

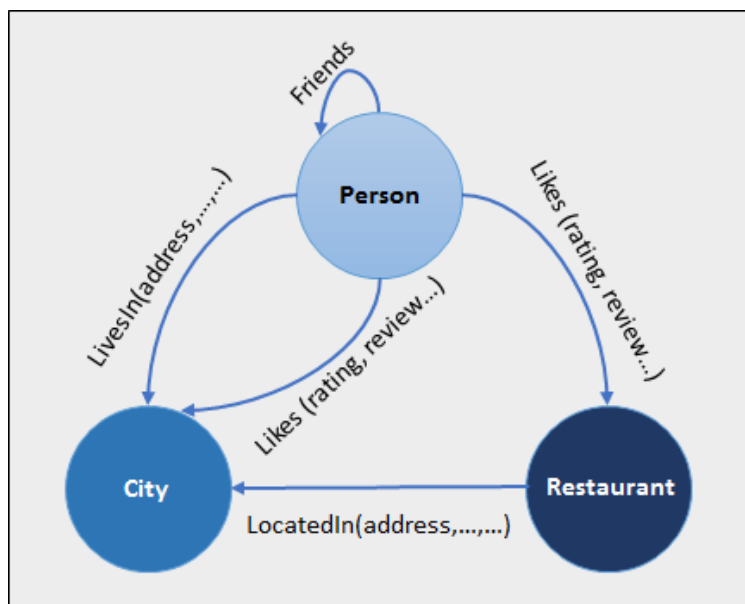
Eles oferecem alto desempenho em consultas de agregação como SUM, COUNT, AVG, MIN etc., pois os dados estão prontamente disponíveis em uma coluna.

Bancos de dados NoSQL baseados em colunas são amplamente usados para gerenciar data warehouses, Business Intelligence, CRM, catálogos de cartões de biblioteca. HBase, Cassandra, HBase, Hypertable são exemplos de bancos de dados baseado em colunas.

NoSQL Grafos

Um banco de dados do tipo grafo armazena entidades, bem como as relações entre essas entidades. A entidade é armazenada como um nó com o relacionamento como arestas. Uma aresta fornece um relacionamento entre os nós. Cada nó e borda possui um identificador exclusivo.

Figura 15–NoSQL Grafos.



Comparado a um banco de dados relacional em que as tabelas são fracamente conectadas, um banco de dados grafo é multi-relacional por natureza.

Atravessa relacionamentos tão rápido quanto estes são capturados no banco de dados, e não há necessidade de calculá-los.

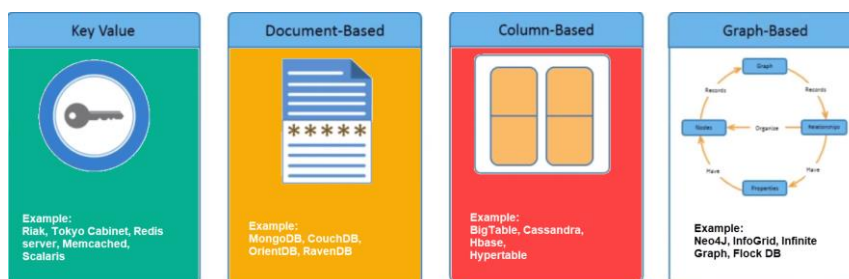
Bancos de dados de base de grafos são usados principalmente para redes sociais, logística e dados espaciais. Alguns exemplos populares são Neo4J, Infinite Graph, OrientDB e FlockDB.

Ferramentas de mecanismo de consulta para NoSQL

O mecanismo de recuperação de dados mais comum é a recuperação baseada em REST de um valor com base em sua chave / ID com recurso GET.

Bancos de dados de armazenamento de documentos oferecem consultas mais difíceis, pois entendem o valor em um par de chave-valor. Por exemplo, CouchDB permite definir visualizações com MapReduce.

Figura 16–Ferramentas NoSQL.



Motivações do uso do NoSQL

Podemos citar as vantagens e desvantagens no uso de banco de dados NoSQL, o que seria uma boa visão para entender o que motiva seu uso.

Vantagens do NoSQL:

- Pode ser usado como fonte de dados primária ou analítica.

- Capacidade de Big Data.
- Nenhum ponto único de falha.
- Replicação fácil.
- Não há necessidade de camada de cache separada.
- Fornece desempenho rápido e escalabilidade horizontal.
- Pode lidar com dados estruturados, semiestruturados e não estruturados com o mesmo efeito.
- Programação orientada a objetos, fácil de usar e flexível.
- Não precisam de um servidor dedicado de alto desempenho.
- Suporte aos principais idiomas e plataformas do desenvolvedor.
- Mais simples de implementar do que usar RDBMS.
- Pode servir como fonte de dados primária para aplicativos online.
- Lida com Big Data – gerencia a velocidade, a variedade, o volume e a complexidade dos dados.
- Excelente em banco de dados distribuído e operações de multi-data center.
- Elimina a necessidade de uma camada de cache específica para armazenar dados.
- Oferece um design de esquema flexível, que pode ser facilmente alterado sem tempo de inatividade ou interrupção do serviço.

Desvantagens:

- Sem regras de padronização.
- Recursos de consulta limitados.

- Bancos de dados e ferramentas RDBMS são comparativamente maduros.
- Ele não oferece nenhum recurso de banco de dados tradicional, como consistência quando várias transações são realizadas simultaneamente.
- Quando o volume de dados aumenta, é difícil manter valores únicos, pois as chaves se tornam difíceis.
- Não funciona tão bem com dados relacionais.
- A curva de aprendizado é difícil para novos desenvolvedores.
- As opções de código aberto não são tão populares para empresas.

Alguns bancos NoSQL

Aerospike: banco de dados NoSQL que oferece uma vantagem de velocidade de memória, atraindo empresas de anúncios de alta escala e aquelas que precisam de tempos de resposta em milissegundo. Aerospike está apostando em novas categorias, incluindo jogos, e-commerce e segurança, em que a baixa latência é tudo.

Apache Cassandra: os pontos fortes são a modelagem de dados NoSQL e escalabilidade linear flexível em hardware por conta do uso de cluster.

Amazon DynamoDB: foi desenvolvido pela Amazon para incrementar o seu e-commerce, possibilitando ter seus serviços altamente escaláveis. Inspirou o Cassandra, Riak, e outros projetos NoSQL.

MongoDB: é o banco de dados mais popular NoSQL, com mais de sete milhões de downloads e centenas de milhares de implantações. Sua popularidade se deve à facilidade de desenvolvimento e ao manejo flexível dos dados. Muito utilizado em aplicações de redes sociais web e móvel.

HBase: é o banco de dados que roda em cima do HDFS (Hadoop Distributed File System – sistema de arquivos distribuído). Por isso, dá aos usuários a capacidade única de trabalhar diretamente com os dados armazenados no Hadoop. As características incluem grande escalabilidade.

Redis: é o banco de dados NoSQL do tipo chave-valor mais conhecido. No mercado podemos encontrar diversas outras soluções que também são mecanismos de armazenamento baseado em chave-valor.

Capítulo 3. Armazenamento de Dados em Nuvem e Sistemas de Arquivos

Banco de Dados em Nuvem - DBaaS

O uso de Cloud Computing pelas empresas já se popularizou graças a seus inúmeros benefícios. O Banco de Dados Híbridos, ou Database as a Service (DBaaS), assim como os outros sistemas que funcionam como serviço, pode ajudar uma organização em seus processos. O DBaaS é baseado em sistemas na Nuvem e é capaz de oferecer plataformas flexíveis, escalonadas e sob demanda para seus usuários. A entrega do software de banco de dados é feita pelo fornecedor do serviço, que é responsável por alocar fisicamente o Data Center. Esse tipo de serviço tem se tornado, cada vez mais, essencial para os negócios, visto que a maioria das empresas já migraram para a Cloud, aumentando a importância de se ter uma base de dados on-line.

Os bancos de dados baseados na nuvem permitem que os usuários armazenem, gerenciem e recuperem seus dados estruturados, não estruturados e semiestruturados, por meio de uma plataforma na nuvem acessível pela Internet. Os bancos de dados em nuvem são também conhecidos como banco de dados como serviço (DBaaS), pois normalmente são oferecidos como serviços gerenciados.

Por que usar bancos de dados na nuvem?

- Os custos de TI podem ser reduzidos: a empresa não precisa investir, manter e dar suporte a um ou mais bancos de dados.
- Tecnologias automatizadas: seu banco de dados colhe benefícios de diversos processos automatizados, como failover automático, recuperação de falha e auto-dimensionamento.
- Maior acessibilidade: desde que haja conexão com a Internet, é possível acessar seu banco de dados na nuvem a partir de qualquer lugar, a qualquer momento.

- Experiência em TI: os projetos de banco de dados são famosos por serem desafiadores. Você pode aprimorar as habilidades relacionadas a banco de dados da sua equipe de TI com as de um provedor de serviços, ou confiar totalmente na experiência em banco de dados da equipe do provedor de serviços.

As principais características do DBaaS são:

- Disponibilidade sob demanda: o serviço deve estar disponível 24 horas por dia, 7 dias por semana para os usuários. Não há a necessidade de se instalar e configurar hardwares ou softwares. A utilização é totalmente virtual.
- Pagamento por assinatura: a empresa que contratar o DBaaS realizará o pagamento por meio de uma assinatura, conforme estabelecido em contrato. Normalmente, o pagamento é mensal e de acordo com a capacidade de armazenamento utilizada, processamento e desempenho.
- O Fornecedor é responsável pela gestão: uma das principais características desse serviço na Nuvem é que a responsabilidade de todo o gerenciamento é do fornecedor. Assim, a empresa não precisa se preocupar em manter, atualizar ou administrar seu banco de dados.

O grande diferencial do DBaaS, em relação a um banco de dados comum, é seu autoatendimento, que torna a gestão e uso das ferramentas uma tarefa fácil. Ele ainda permite que os usuários consumam, operem e configurem o banco de dados por meio de conjuntos de abstrações, mesmo que não tenham conhecimento técnico para isso.

Movimentando seus bancos de dados para a nuvem

Como a maioria das transformações organizacionais, mover para a nuvem não é algo que se realiza da noite para o dia. Você deve escolher um projeto para experimentar em um provedor de nuvem escolhido. Quase todas as organizações

que se movem para a nuvem fazem uma POC (prova de conceito) com um banco de dados não crítico.

Embora cada migração seja única, você provavelmente seguirá as etapas:

▪ **Planejamento:**

- Coleta de requisitos.
- Determinando recursos para atender aos requisitos.
- Avaliação de quais bancos de dados mover e quais mudanças podem ser necessárias para o banco de dados ou os aplicativos que o utilizam.
- Estabelecendo critérios de sucesso e critérios de reversão.

▪ **Movimentando os dados:**

- Replicação.
- Incorporação de mudanças desde que a réplica foi criada.
- Teste de aplicação.
- Verificações pós-migração.

▪ **Otimização:**

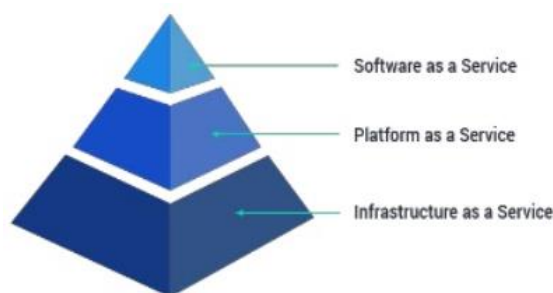
- Ajuste de desempenho.
- Projetando alta disponibilidade.
- Determinar quais eventos registrar e monitorar.
- Criação de um plano de recuperação de desastres.

Modelos em Nuvem

Para se falar em Nuvem, é importante abordar os três tipos de modelos em nuvem:

- SaaS é caracterizado principalmente pela não-aquisição de licenças de software.
- PaaS envolve um ambiente virtual para criação, hospedagem e controle de softwares e bancos de dados.
- IaaS apenas abstrai aspectos relacionados à parte física de servidores e redes.

Figura 17 – Modelos de Nuvem.



IaaS — Infrastructure as a Service (Infraestrutura como Serviço):

Nesse exemplo dos modelos de nuvem, a empresa contrata uma capacidade de hardware que corresponde à memória, armazenamento, processamento etc. Podem entrar nesse pacote de contratações os servidores, roteadores, racks, entre outros.

Dependendo do fornecedor e do modelo escolhido, a sua empresa pode ser tarifada, por exemplo, pelo número de servidores utilizados e pela quantidade de dados armazenados ou trafegados. Em geral, tudo é fornecido por meio de um data center com servidores virtuais, em que você paga somente por aquilo que usar.

PaaS — Platform as a Service (Plataforma como Serviço):

Nesse cenário, o PaaS surge como o ideal porque é, como o próprio nome diz, uma plataforma que pode criar, hospedar e gerir esse aplicativo.

Nesse modelo de nuvem, contrata-se um ambiente completo de desenvolvimento, no qual é possível criar, modificar e otimizar softwares e aplicações.

Aqui, a grande vantagem é que a equipe de desenvolvimento só precisa se preocupar com a programação do software, pois o gerenciamento, manutenção e atualização da infraestrutura ficam a cargo do fornecedor e às várias ferramentas de desenvolvimento de software que são oferecidas na plataforma.

SaaS — Software as a Service (Software como Serviço):

Nesse modelo de nuvem, você pode ter acesso a um software sem comprar a sua licença, utilizando-o a partir da Cloud Computing.

Muitos CRMs ou ERPs trabalham no sistema SaaS. Assim, o acesso a esses softwares é feito usando a internet. Os dados, contatos e demais informações podem ser acessados de qualquer dispositivo, dando mais mobilidade à equipe.

O Google Docs e o Office 365 funcionam dessa maneira.

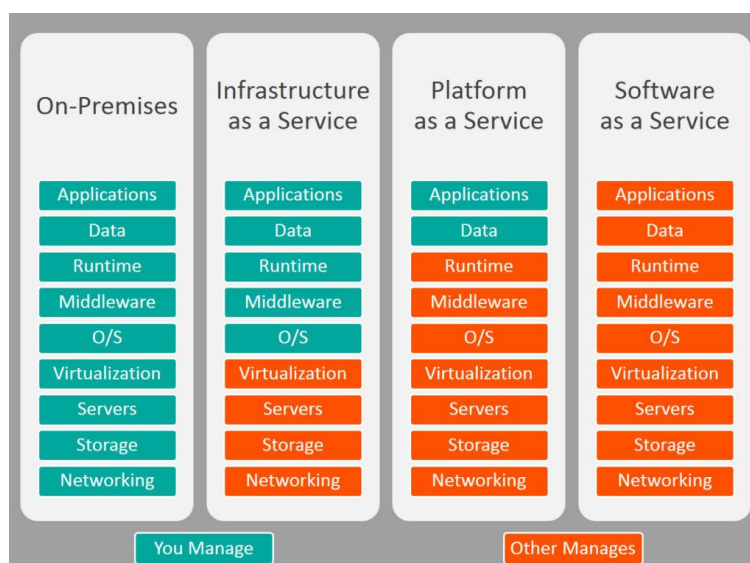
On-Premises:

Um servidor on-premises é aquele em que a própria empresa tem a responsabilidade de processar suas aplicações de hardware e software.

Em outras palavras, toda a infraestrutura, customização, configuração e atualização é feita internamente.

A figura a seguir denota as responsabilidades de gerenciamento para cada modelo em Nuvem.

Figura 18 – Gerenciamento modelo em Nuvem.
















Introdução a AWS

Amazon Web Services, também conhecido como AWS, é uma plataforma de serviços de computação em nuvem, que formam uma plataforma de computação na nuvem oferecida pela Amazon.com. Os serviços são oferecidos em várias geográficas distribuídas pelo mundo.

EC2: o Amazon Elastic Compute Cloud (Amazon EC2) oferece uma capacidade de computação escalável na Nuvem da Amazon Web Services (AWS). O uso do Amazon EC2 elimina a necessidade de investir em hardware inicialmente, portanto, você pode desenvolver e implantar aplicativos com mais rapidez. Você pode usar o Amazon EC2 para executar o número de servidores virtuais que precisar, configurar a segurança e a rede, e gerenciar o armazenamento. O Amazon EC2 também permite a expansão ou a redução para gerenciar as alterações de requisitos ou picos de popularidade, reduzindo, assim, a sua necessidade de prever o tráfego do servidor.

Abaixo temos uma visão geral de instâncias indicadas por situação.

General Purpose	Compute Optimised	Memory Optimised	Accelerated Computing	Storage Optimised
 ARM based core and custom silicon	 Compute - CPU intensive apps and DBs	 RAM - Memory intensive apps and DB's	 Processing optimised - Machine Learning	 High Disk Throughput - Big data clusters
 Tiny - Web servers and small DBs		 Xtreme RAM - For SAP/Spark	 Graphics Intensive - Video and streaming	 IOPS - NoSQL DBs
 Main - App servers and general purpose		 High Compute and High Memory - Gaming	 Field Programmable - Hardware acceleration	 Dense Storage - Data Warehousing

AWS - S3:

O Amazon Simple Storage Service é armazenamento para a Internet. Ele foi projetado para facilitar a computação de escala na web para os desenvolvedores. O Amazon S3 tem uma interface simples de serviços da web que você pode usar para armazenar e recuperar qualquer quantidade de dados, a qualquer momento, em qualquer lugar da web.

Ela concede acesso a todos os desenvolvedores para a mesma infraestrutura altamente dimensionável, confiável, segura, rápida e econômica que a Amazon utiliza para rodar a sua própria rede global de sites da web. O serviço visa maximizar os benefícios de escala e poder passar esses benefícios para os desenvolvedores.

AWS – RDS:

O Amazon Relational Database Service (Amazon RDS) facilita a configuração, a operação e a escalabilidade de bancos de dados relacionais na nuvem. O serviço oferece capacidade econômica e redimensionável, e automatiza tarefas demoradas de administração, como provisionamento de hardware, configuração de bancos de dados, aplicação de patches e backups. Dessa forma, você pode se concentrar na performance rápida, alta disponibilidade, segurança e conformidade que os aplicativos precisam.

Mecanismos de banco de dados do Amazon RDS



As formas de cobrança da AWS podem ser:

- On-demand: o cálculo é feito em cima de horas ou segundos utilizados (no mínimo 60 segundos) e somente as instâncias EC2 que forem utilizadas.
- Instâncias reservadas (RIs): o preço por hora é fixo, independentemente do uso, e existe um prazo pré-determinado de contratação. Essa forma de pagamento tem o benefício de obter desconto, uma vez que o cliente tem o compromisso de um a três anos.
- Instâncias spot: por serem instâncias extras, ou seja, instâncias de capacidade extra na Nuvem AWS, o preço é muito mais atrativo. Por outro lado, se o EC2 precisar de capacidade, o cliente que utiliza Instância Spot será notificado 2 minutos antes que suas instâncias serão interrompidas.

Google Cloud Platform

Google Cloud Platform é uma suíte de computação em nuvem oferecida pelo Google, funcionando na mesma infraestrutura que a empresa usa para seus produtos dirigidos aos usuários, dentre eles o Buscador Google e o YouTube.

Juntamente com um conjunto de ferramentas de gerenciamento modulares, fornecem uma série de serviços incluindo, computação, armazenamento de dados, análise de dados e aprendizagem de máquina.

Compute Engine

O Google Compute Engine é o componente Infraestrutura como serviço do Google Cloud Platform, construído sobre a infraestrutura global que executa o mecanismo de pesquisa do Google, Gmail, YouTube e outros serviços.

O Google Compute Engine permite que os usuários iniciem máquinas virtuais sob demanda. Uma máquina virtual é um computador emulado em outra máquina. Nesse caso, o servidor em nuvem executa vários sistemas operacionais em diversas outras máquinas, em vez de cada computador ter um sistema operacional próprio.

O Google Compute Engine oferece máquinas virtuais em execução nos centros de dados inovadores do Google e na rede mundial de fibra. O suporte a ferramentas e fluxo de trabalho do Compute Engine permite dimensionar de instâncias únicas para computação em nuvem balanceada de carga global. As VMs do Compute Engine são inicializadas rapidamente, vêm com opções de disco persistentes e locais de alto desempenho, oferecem desempenho consistente.

- Instâncias.
- Configurações.
- Deploy de aplicação e Jobs.
- Alertas automáticos.

Storage & Databases:

A ideia de Plataforma como um Serviço (PaaS) já é bem difundida em muitos setores, especialmente no que diz respeito à gestão e ao armazenamento de dados. É similar ao Google Drive, porém, em escala bem maior, o que permite a você armazenar e organizar todos os arquivos da empresa e acessá-los a partir de qualquer máquina com permissão de acesso. É ótimo para evitar a perda de documentos e para minimizar o uso do espaço físico em seu negócio.

Armazenamento de produtos escaláveis, resilientes e de alto desempenho. Bancos de dados para suas aplicações.

- Armazenamento de arquivos.
- Gerenciamento de buckets.

- Bucket Locations.
- Multi Regional.
- Regional.
- NearLine.
- CodeLine.
- Linha de comando, API, Console.

App Engine:

Plataforma para criação de aplicativos Web escaláveis e backends para dispositivos móveis. O App Engine fornece serviços integrados e APIs, como datastores NoSQL, memcache e uma API de autenticação de usuário, comum à maioria dos aplicativos.

- Construir aplicações de desenvolvimentos escaláveis.
- Instâncias automáticas.
- Integrações.
- Deploy automatizado.

Big Data:

Quando precisamos tomar uma decisão importante para a empresa, é importante ter dados que deem suporte a essa escolha. Ainda mais, atualmente, em um mercado cada vez mais complexo e volátil.

Totalmente gerenciado por data warehousing, lote e processamento de fluxo, exploração de dados, Hadoop/Spark e mensagens confiáveis.

- Análise de dados em bancos NoSQL.
- Construir bancos via: linha de comando, console e API's.

- Importar dados: csv, txt, integrações.

Machine Learning:

Serviços de ML rápidos, escaláveis e fáceis de usar. Use modelos pré-treinados ou treine modelos personalizados em seus dados.

- Máquina de conhecimento, como funciona.
- Exemplo Case Google: E-mail e Spam.
- Google Cloud Vision API.
- Google Translate API.
- Google Speech API.

Microsoft Azure

O Microsoft Azure é uma plataforma destinada à execução de aplicativos e serviços, baseada nos conceitos da computação em nuvem.

A apresentação do serviço foi feita no dia 27 de outubro de 2008 durante a *Professional Developers Conference*, em Los Angeles e lançado em 1 de fevereiro de 2010 como Windows Azure, para então ser renomeado como Microsoft Azure em 25 de março de 2014.

Sua computação em nuvem é definida como uma combinação de software como serviço (SaaS) com computação em grid. A computação em grid dá o poder de computação e alta escalabilidade oferecida para as aplicações, através de milhares de máquinas (hardware) disponíveis em centros de processamento de dados de última geração. De software como serviço tem-se a capacidade de contratar um serviço e pagar somente pelo uso, permitindo a redução de custos operacionais, com uma configuração de infraestrutura realmente mais aderente às necessidades.

Além dos recursos de computação, armazenamento e administração oferecidos pelo Microsoft Azure, a plataforma também disponibiliza uma série de serviços para a construção de aplicações distribuídas, além da total integração com a solução on-premise (local) baseada em plataforma .NET.

Entre os principais serviços da plataforma Windows Azure há o SQL Azure Database, Azure AppFabric Platform e uma API de gerenciamento e monitoramento para aplicações colocadas na nuvem.

As formas de cobrança da Azure podem ser:

- On-demand: Seus custos são realizados em cima dos minutos utilizados. Neste modelo, não é necessário compromisso de tempo mínimo de contratação. Como o pagamento é feito de acordo com a utilização, é possível aumentar e diminuir recursos sem limite.
- Contrato pré-definido: Como um determinado tempo de utilização é acordado e o custo é reduzido.
- Acordo empresarial: Nessa modalidade, o pagamento é realizado antecipadamente, por esse motivo, há benefícios e desconto. O uso adicional é pago de forma separada, mas com desconto nas taxas.

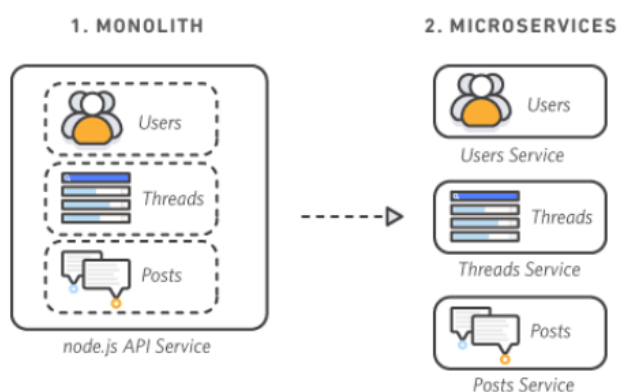
Arquitetura microsservicos

Arquiteturas monolíticas: Com as arquiteturas monolíticas, todos os processos são altamente acoplados e executam como um único serviço. Isso significa que se um processo do aplicativo apresentar um pico de demanda, toda a arquitetura deverá ser escalada. A complexidade da adição ou do aprimoramento de recursos de aplicativos monolíticos aumenta com o crescimento da base de código. Essa complexidade limita a experimentação e dificulta a implementação de novas ideias. As arquiteturas monolíticas aumentam o risco de disponibilidade de

aplicativos, pois muitos processos dependentes e altamente acoplados aumentam o impacto da falha de um único processo.

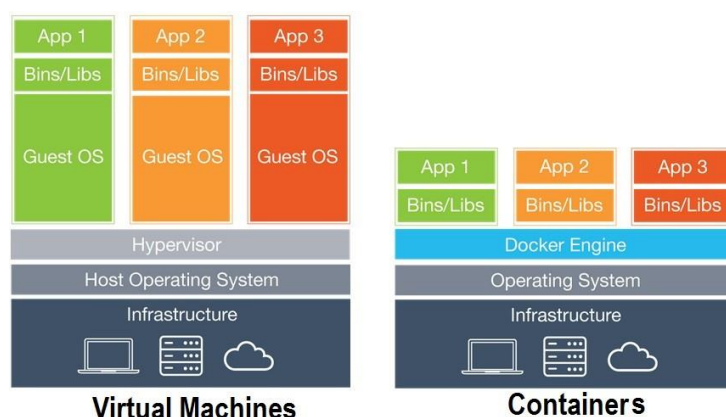
Arquitetura de microsserviços: com uma arquitetura de microsserviços, um aplicativo é criado como componentes independentes que executam cada processo do aplicativo como um serviço. Esses serviços se comunicam por meio de uma interface bem definida usando APIs leves. Os serviços são criados para recursos empresariais e cada serviço realiza uma única função. Como são executados de forma independente, cada serviço pode ser atualizado, implantado e escalado para atender a demanda de funções específicas de um aplicativo.

Figura 19 – Microsserviço.



Container: em vez de usar um sistema operacional para cada estrutura, como na virtualização, os containers são blocos de espaços divididos pelo Docker em um servidor, o que possibilita a implementação de estruturas de microsserviços que compartilham o mesmo sistema operacional. Porém, de forma limitada (conforme a demanda por capacidade). O fato de os containers não terem seus próprios sistemas operacionais, permite que eles consumam menos recursos e, com isso, sejam mais leves.

Figura 20 – Containers.

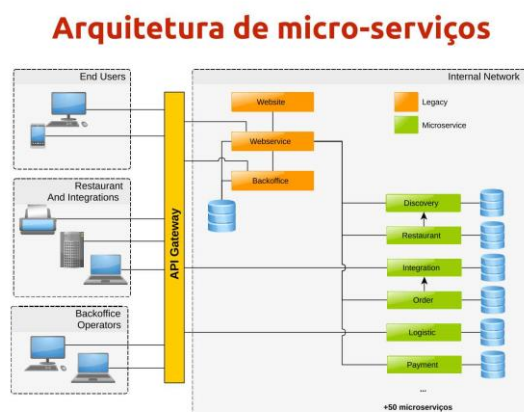


As ferramentas de orquestração de containers são aplicações em nuvem que permitem fazer o gerenciamento de múltiplos containers. Seus principais objetivos são:

- Cuidar do ciclo de vida dos containers de forma autônoma, subindo e distribuindo, conforme nossas especificações ou demandas.
- Gerenciar volumes e rede, que podem ser local ou no cloud provider de sua preferência.

O Kubernetes, ECS e o Docker são as principais plataformas de gerenciamento de containers. Dessa forma, essa é uma ferramenta para viabilizar a utilização de containers e microsserviços em servidores com mais facilidade, pois permite empacotar os aplicativos para que possam ser movimentados facilmente. O Docker permite, por exemplo, que uma biblioteca possa ser instalada em diferentes containers sem que haja qualquer interdependência entre eles. Essa característica tem o objetivo de facilitar o gerenciamento de códigos e aplicativos.

Figura 21 – Arquitetura de microserviços.



Sistemas de Arquivos Distribuídos

Os sistemas de arquivos foram originalmente desenvolvidos como um recurso do S.O que fornece uma interface de programação conveniente para armazenamento em disco. São responsáveis pela organização, armazenamento, recuperação, atribuição de nomes, compartilhamento e proteção de arquivos. Projetados para armazenar e gerenciar muitos arquivos, com recursos para criação atribuição de nomes e exclusão de arquivos.

“Um sistema de arquivos distribuídos permite aos programas armazenarem e acessarem arquivos remotos exatamente como se fossem locais, possibilitando que os usuários acessem arquivos a partir de qualquer computador em uma rede” (COULOURIS, et. al., p. 284).

- Objetivo: permitir que os programas armazenem e acessem arquivos remotos exatamente como se fossem locais.
- Permitem que vários processos compartilhem dados por longos períodos, de modo seguro e confiável.

- O desempenho e segurança no acesso aos arquivos armazenados em um servidor devem ser compatíveis aos arquivos armazenados em discos locais.

Os requisitos de um sistema de arquivos distribuídos são:

- Transparência.
- Atualização concorrente de arquivos.
- Replicação de arquivos.
- Heterogeneidade.
- Tolerância a falha.
- Consistência.
- Segurança.
- Eficiência.

Arquitetura do SAD:

Modelo abstrato de arquitetura que serve para o Network File System (NFS) e Andrew File System (AFS).

Divisão de responsabilidades entre três módulos.

- Cliente.
- Serviço de arquivos planos.
- Serviço de diretórios.

Design aberto:

- Diferentes módulos cliente, podem ser utilizados para implementar diferentes interfaces.
- Simulação de operações de arquivos de diferentes S.O.

- Otimização de performance para diferentes configurações de hardware de clientes e servidores.

Funções de um Sistema de Arquivos Distribuído:

- Armazenar e compartilhar programas e dados:
 - Funções idênticas as de um sistema centralizado (local).
- Ênfase na disponibilidade, confiabilidade e segurança.
- Desempenho:
 - Questão importante porque acessos remotos podem ser significativamente mais lentos que os locais.
 - Não se pretende, em geral, que o SAD seja mais rápido que um SA local, mas sim que a degradação seja aceitável.

Sistemas de arquivo distribuídos devem ser vistos pelos clientes como um sistema de arquivo local. A transparência é muito importante para seu bom funcionamento. É necessário um bom controle de concorrência no acesso. Cache é importante.

Apache Hadoop

Hadoop é uma plataforma de software de código aberto para o armazenamento e processamento distribuído de grandes conjuntos de dados, utilizando clusters de computadores com hardware commodity.

Os serviços do Hadoop fornecem armazenamento, processamento, acesso, governança, segurança e operações de Dados.

Algumas das razões para se usar Hadoop é a sua capacidade de armazenar, gerenciar e analisar grandes quantidades de dados estruturados e não estruturados de forma rápida, confiável, flexível e de baixo custo.

- Escalabilidade e desempenho – é distribuído tratamento de dados local para cada nó em um cluster Hadoop, permite armazenar, gerenciar, processar e analisar dados em escala petabyte.
- Confiabilidade – clusters de computação de grande porte são propensos a falhas de nós individuais no cluster. Hadoop é fundamentalmente resistente – quando um nó falha de processamento, é redirecionado para os nós restantes no cluster e os dados são automaticamente re-replicado em preparação para falhas de nó futuras.
- Flexibilidade – ao contrário de sistemas de gerenciamento de banco de dados relacionais tradicionais, você não tem que esquemas estruturados criados antes de armazenar dados. Você pode armazenar dados em qualquer formato, incluindo formatos semiestruturados ou não estruturados, e em seguida, analisar e aplicar esquema para os dados quando ler.
- Baixo custo – ao contrário de software proprietário, o Hadoop é open source e é executado em hardware commodity de baixo custo.

O HDFS (Hadoop Distributed File System) é um sistema de arquivos distribuído, projetado para armazenar arquivos muito grandes, com padrão de acesso aos dados streaming, utilizando clusters de servidores facilmente encontrados no mercado e de baixo ou médio custo.

Não deve ser utilizado para aplicações que precisem de acesso rápido a um determinado registro e sim para aplicações nas quais é necessário ler uma quantidade muito grande de dados.

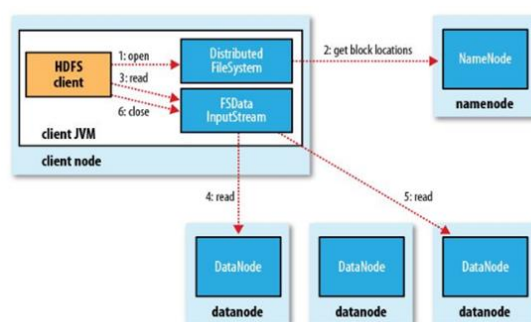
Outra questão que deve ser observada é que não deve ser utilizado para ler muitos arquivos pequenos, tendo em vista o overhead de memória envolvido.

O HDFS tem 2 tipos de nós: Master (ou Namenode) e Worker (ou Datanode).

- O Master armazena informações da distribuição de arquivos e metadados.
- Já o Worker armazena os dados propriamente ditos. Logo, o Master precisa sempre estar disponível. Para garantir a disponibilidade, podemos ter um backup (similar ao Cold Failover) ou termos um Master Secundário em um outro servidor. Nesta segunda opção, em caso de falha do primário, o secundário pode assumir o controle muito rapidamente.

Tal como um sistema Unix, é possível utilizar o HDFS via linha de comando.

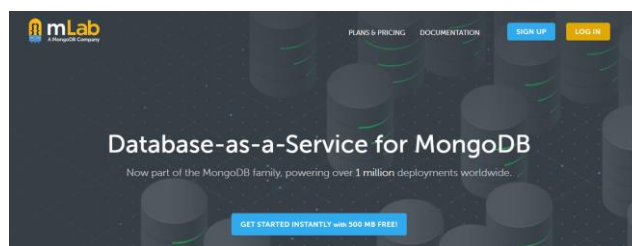
Figura 22 – HDFS.



MongoDB na Nuvem

Conhecido um tempo atrás como MongoLab, o mLab é um serviço de banco de dados gerenciável que hospeda na nuvem um banco de dados MongoDB e é executado em provedores como a Amazon Web Services (AWS), Google Cloud e Microsoft Azure.

A parte mais interessante é que o serviço tem um plano gratuito que oferece 0,5 Gb para armazenamento de dados.

Figura 23 – Mongo.

Capítulo 4. Data Warehouse e Data Lake

Definição de BI

BI é um termo criado pelo Gartner Group, que define como "um termo genérico que inclui aplicações, infraestrutura, ferramentas e melhores práticas que permite o acesso e análise de informações para melhorar e otimizar decisões e desempenho". É um processo de coleta, transformação, análise e distribuição de dados para melhorar a decisão dos negócios.

Descreve a capacidade de a empresa ter acesso e explorar seus dados, desenvolvendo percepção e conhecimento, o que leva à melhora do processo de tomada de decisões. Sua infraestrutura tecnológica é composta de data warehouse ou data marts, data mining e ODS, além das ferramentas pertinentes. Os principais processos uma aplicação de Business Intelligence são:

- Modelagem dimensional.
- ETL.
- OLAP.

O BI pode ser usado para adquirir insights táticos para otimizar processos de negócios, identificando tendências, anomalias e comportamentos que requerem ação de gerenciamento e visão estratégica, para alinhar vários processos de negócios aos principais objetivos de negócios por meio de gerenciamento e análise de desempenho integrados. Além disso, o BI proporciona uma tomada de decisão baseada em fatos e uma visão única dos dados.

Abrange os processos, ferramentas e tecnologias necessárias para transformar dados corporativos em informações e informações em conhecimento que podem ser usados para aprimorar a tomada de decisões e criar planos acionáveis que conduzem a uma atividade comercial eficaz.

O BI proporciona uma tomada de decisão baseada em fatos e visão única dos dados.

Figura 24– Conceito de pirâmide do BI.



Fonte: <https://www.dreamstime.com>.

Data warehouse

Data warehouse é um depósito de dados digitais que serve para armazenar informações detalhadas relativamente a uma empresa, criando e organizando relatórios através de históricos que são depois usados pela empresa para ajudar a tomar decisões importantes com base nos fatos apresentados.

O data warehouse serve para recolher informações de uma empresa para que essa possa controlar melhor um determinado processo, disponibilizando uma maior flexibilidade nas pesquisas e nas informações que necessitam.

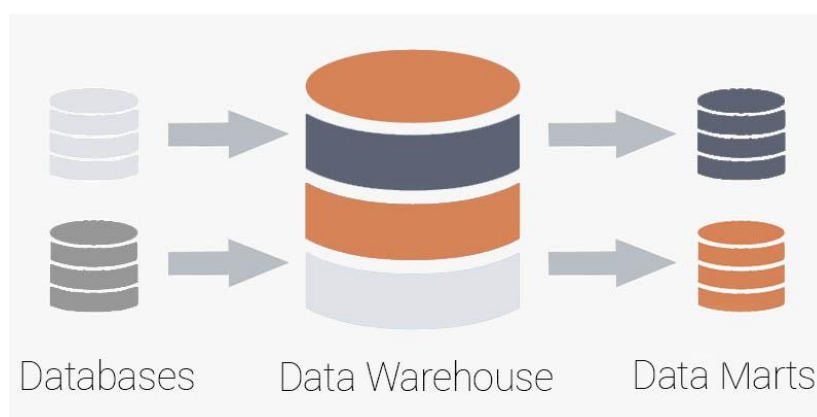
Para além de manter um histórico de informações, o data warehouse cria padrões melhorando os dados analisados de todos os sistemas, corrigindo os erros e reestruturando os dados sem afetar o sistema de operação, apresentando somente um modelo final e organizado para a análise.

Data mart

Um data mart é uma subdivisão ou subconjunto de um DW. Os data marts são como pequenas fatias que armazenam subconjuntos de dados, normalmente organizados para um departamento ou um processo de negócio.

Normalmente o data mart é direcionado para uma linha de negócios ou equipe, sendo que a sua informação costuma pertencer a um único departamento.

Figura 25 – Data warehouse x Data mart.



Fonte: <https://www.cetax.com.br/data-warehouse/>.

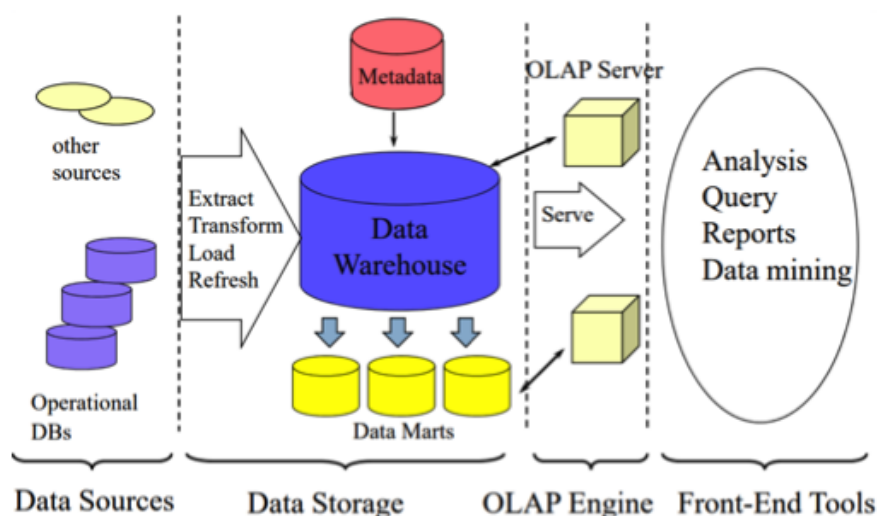
Relatórios Ad hoc

Bill Inmon (2002) conceitua a consulta ad-hoc como: são consultas com acesso casual único e tratamento dos dados segundo parâmetros nunca antes utilizados, geralmente executado de forma iterativa e heurística. Isso tudo nada mais é do que o próprio usuário gerar consultas de acordo com suas necessidades de cruzar as informações de uma forma não vista e com métodos que o levem a descoberta daquilo que procura. (INMON, 2002)

Uma das características que devem estar presentes em ferramentas OLAP é a capacidade de efetuar algumas operações, como:

- **Drill Across:** ocorre quando o usuário pula um nível intermediário dentro de uma mesma dimensão. Por exemplo, a dimensão tempo é composta por ano, semestre, trimestre, mês e dia. A operação Drill Across é executada quando o usuário passa de ano direto para trimestre ou mês.
- **Drill Down:** ocorre quando o usuário aumenta o nível de detalhe da informação, diminuindo a granularidade. Ela influencia diretamente na velocidade do acesso às informações e no volume de dados armazenados.
- **Drill Up:** é o contrário do Drill Down, ocorre quando o usuário aumenta a granularidade, diminuindo o nível de detalhamento da informação.
- **Drill Throught:** ocorre quando o usuário passa de uma informação contida em uma dimensão para outra. Por exemplo: inicia na dimensão do tempo e no próximo passo analisa a informação por região.
- **Slice and Dice:** é uma das principais características de uma ferramenta OLAP. Como a ferramenta OLAP recupera o cubo, surgiu a necessidade de criar um módulo, que se convencionou de Slice and Dice, para ficar responsável por trabalhar esta informação. Ele serve para modificar a posição de uma informação, trocar linhas por colunas de maneira a facilitar a compreensão dos usuários e girar o cubo sempre que tiver necessidade.

Figura 26– Componentes do BI.



Integração de dados

A integração de dados ou *data integration* envolve a combinação de dados residentes em diferentes fontes e fornece aos usuários uma visão unificada desses dados. Esse processo se torna significativo em uma variedade de situações, incluindo domínios comerciais (como quando duas empresas similares precisam mesclar seus bancos de dados) e científicos (combinando resultados de pesquisa de diferentes repositórios de bioinformática, por exemplo). A integração de dados aparece com frequência crescente à medida que o volume (ou seja, big data) e a necessidade de compartilhar dados existentes explodem. Tornou-se o foco de extenso trabalho teórico, e numerosos problemas em aberto permanecem sem solução. A integração de dados incentiva a colaboração entre usuários internos e externos.

Definição de ETL

Na fase de integração dos dados executamos o ETL. Essa sigla significa *Extract, Transform and Load*, ou seja, Extração, Transformação e Carga.

Comumente utilizado para construir um data warehouse. Nesse processo, os dados são retirados (extraídos) de um ou mais sistemas-fonte ou origens, convertidos (transformados) em um formato que possa ser analisado, e armazenados (carregados) em um armazém ou outro sistema.

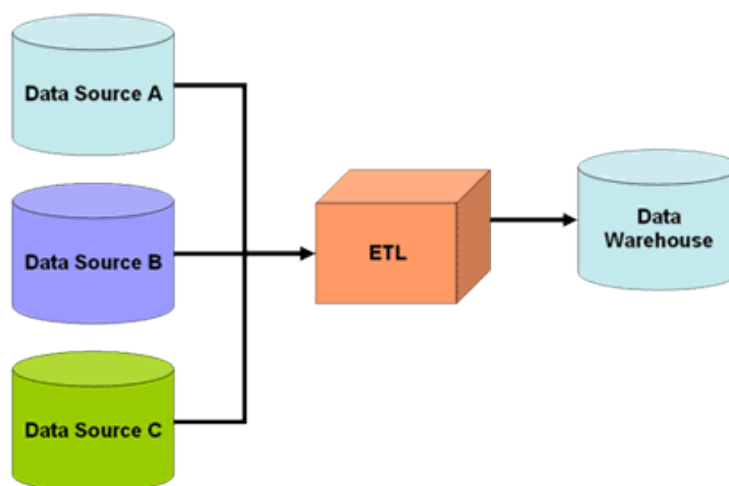
Conforme Ralph Kimball: um sistema ETL projetado adequadamente extrai dados dos sistemas de origem, aplica padrões de qualidade e consistência de dados, alinha os dados para que fontes separadas possam ser usadas em conjunto e, finalmente, entrega dados em um formato pronto para apresentação, para que os desenvolvedores possam criar aplicativos e os usuários finais podem tomar decisões. (KIMBALL, 2004)

A extração e carga são obrigatórias para o processo, sendo a transformação/limpeza opcional, mas que são boas práticas, tendo em vista que os

dados já foram encaminhados para o sistema de destino. É considerada uma das fases mais críticas do data warehouse e/ou data mart.

Os projetos de data warehouse consolidam dados de diferentes fontes. A maioria dessas fontes tende a ser bancos de dados relacionais ou arquivo de texto (texto plano), mas podem existir outras fontes. Uma aplicação ETL tem que ser capaz de se comunicar com as bases de dados e ler diversos formatos de arquivos utilizados por toda a organização. Essa pode ser uma tarefa não trivial, e muitas fontes de dados podem não ser acessadas com facilidade.

Figura 27 – Ferramenta de ETL.



Fonte: <http://gedxml.com.br/>.

Como surgiu o ETL

ETL ganhou popularidade nos anos 1970, quando as organizações começaram a usar múltiplos repositórios ou bancos de dados para armazenar diferentes tipos de informações de negócios. A necessidade de integrar os dados que se espalhavam pelos databases cresceu rapidamente. O ETL tornou-se o método padrão para coletar dados de fontes diferentes e transformá-los antes de carregá-los no sistema-alvo ou destino.

No final dos anos 1980 e início dos 1990, os data warehouses entraram em cena. Sendo um tipo diferente de banco de dados, eles forneceram um acesso integrado a dados de múltiplos sistemas – computadores mainframes, minicomputadores, computadores pessoais e planilhas. Mas diferentes departamentos costumam usar diferentes ferramentas ETL com diferentes armazéns. Adicione isso a junções e aquisições, e muitas empresas acabam com distintas soluções ETL, que não foram integradas.

Com o tempo, o número de formatos, fontes e sistemas de dados aumentou muito. Extrair, transformar e carregar é, hoje, apenas um dos vários métodos que as organizações utilizam para coletar, importar e processar dados. ETL e ELT são, ambos, partes importantes de uma estratégia ampla de data integration das empresas.

Extração

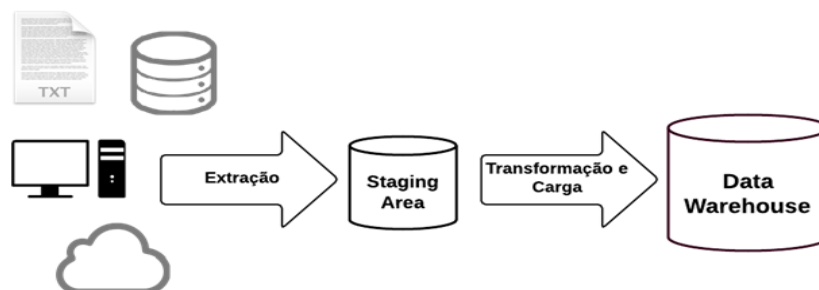
A extração é a primeira etapa de um processo de ETL, sendo a coleta de dados dos sistemas de origem (Data Sources ou sistemas operacionais) extraíndo-os e transferindo-os para o ambiente de DW, onde o sistema ETL pode operar independente dos sistemas de origem. Um processo ETL precisa ser capaz de se comunicar com bases de dados, visto que os projetos de data warehouse consolidam dados de diferentes fontes, e ler diversos formatos de arquivos utilizados por toda a organização.

O primeiro passo é definir as fontes de extração e os dados podem vir das mais diversas fontes, por exemplo: Sistemas de gestão (SIG, ERP, CRM etc.), SGBD's (Oracle, SQLSERVER, DB2 etc.), arquivos como planilhas do Excel e documentos de texto.

A etapa de extração é a fase onde os dados são extraídos dos OLTPs (Online Transaction Processing) e conduzidos para a staging area (área temporária), onde são convertidos para um único formato. A conversão se faz

necessária devido a heterogeneidade existente nas informações oriundas desses sistemas, sendo essencial a conformação prévia para o tratamento adequado.

Figura 28 – Etapa de Extração de Dados.



Fonte: <https://canaltech.com.br/>.

Transformação

Após a extração teremos subsídios para iniciar a etapa de transformação e limpeza dos dados. Nessa fase são corrigidos, padronizados e tratados os desvios e inconsistências, transformando os dados de acordo com as regras do negócio.

É nesta etapa que realizamos os devidos ajustes, podendo assim, melhorar a qualidade dos dados e consolidar dados de duas ou mais fontes. O estágio de transformação aplica uma série de regras ou funções aos dados extraídos para ajustar os dados a serem carregados.

O processo de transformação de dados é composto por vários subprocessos, como por exemplo:

- Limpeza: inconsistências e valores ausentes nos dados são resolvidos.
- Padronização: regra de formatação é aplicada ao conjunto de dados.
- Eliminar duplicados: dados redundantes são excluídos ou descartados.
- Verificação: dados inutilizáveis são removidos e anomalias são sinalizadas.
- Classificação: os dados são organizados de acordo com o tipo.

- Outras tarefas: quaisquer regras adicionais/opcionais podem ser aplicadas para melhorar a qualidade dos dados.

Carga

A etapa de carga ocorre em sequência com a de transformação. Assim que são efetuados os tratamentos necessários nos dados, a carga no DW é iniciada. Essa fase se resume em persistência dos dados na base consolidada. Consiste em estruturar e carregar os dados.

Dentro de um mesmo DW temos diferentes períodos de execução para cada tipo de processo de carga. Alguns são mensais, outros diários. Neste momento também é definida a latência das informações, isso pode variar para cada tabela a ser carregada. Latência é sinônimo de atraso, é uma expressão de quanto tempo leva para um pacote de dados ir de um ponto designado para o outro.

As cargas podem ser full ou incrementais. A full ou total, se trata da carga completa dos dados toda vez que há a execução de um novo processo de ETL. Nesse tipo de carga todos os dados da origem são extraídos e transformados, recarregando os dados antigos e incrementando com os novos. Já a incremental considera apenas os novos registros dos sistemas operacionais no ETL, inserindo-os ao repositório do DW. A carga total é mais custosa que a incremental, tanto em tempo de carga, quanto em processamento (CPU).

Importância do ETL

Há anos, inúmeras empresas têm confiado no processo de ETL para obter uma visão consolidada dos dados que geram as melhores decisões de negócios. Hoje, esse método de integrar dados de múltiplos sistemas e fontes, ainda é um componente central do kit de ferramentas de data integration de uma organização.

- Quando utilizado com um data warehouse corporativo (dados em repouso), o ETL fornece o contexto histórico completo para a empresa.
- Ao fornecer uma visão consolidada, o ETL facilita para os usuários corporativos a análise e a criação de relatórios sobre dados relevantes às suas iniciativas.
- O ETL pode melhorar a produtividade de profissionais analíticos, porque ele codifica e reutiliza processos que movem os dados sem que esses profissionais possuam a capacidade técnica de escrever códigos ou scripts.
- O ETL evoluiu ao longo do tempo para suportar os requisitos emergentes de integração para coisas como streaming data.
- As organizações precisam tanto de ETL quanto ELT para unir dados, manter a precisão e fornecer a auditoria necessária para armazenar dados, criar relatórios e realizar análises.

O processo ETL é uma das fases mais críticas na construção de um sistema DW, visto que:

- Grandes volumes de dados são processados.
- Serão implementadas as regras e fórmulas dos indicadores que irão compor as tabelas de Fato.
- É essencial para a criação das estruturas de Dimensões e Fatos no ambiente do DW. É ele que faz a “ponte” de ligação entre o operacional e o DW.

As ferramentas, que darão suporte ao processo, devem ser bem escolhidas, pois são essenciais para a correta execução das atividades do ETL. A janela de operação do ETL deve ser analisada, pois não é em qualquer momento que ele poderá ser executado e também analisar a periodicidade de execução.

O ETL é fundamental para qualquer iniciativa de DW. Porém, deve ser planejado com cuidado para não comprometer os sistemas transacionais (OLTP) das empresas. Um bom ETL deve ter escalabilidade e ser passivo de ser mantido.

Estudos relatam que o ETL e as ferramentas de limpeza de dados consomem um terço do orçamento num projeto de DW, podendo, no que tange ao tempo de desenvolvimento de um projeto de DW, chegar a consumir 80% desse valor. Outros estudos mencionam, ainda, que o processo de ETL consome 55% do tempo total de execução do projeto de DW.

As ferramentas de ETL podem ser utilizadas para fazer todo tipo de trabalho de importação, exportação, transformação de dados para outros ambientes de banco de dados ou para outras necessidades a serem endereçadas. Exemplo: Importação de base de outra empresa.

Benefícios do ETL

Histórico dos dados: ao reunir as informações do negócio em um repositório, o ETL fornece um histórico da empresa e auxilia a produção de relatórios e análises sobre dados relevantes para iniciativas corporativas.

Fácil de usar: Ferramentas de ETL eliminam a necessidade de programar o código para extrair e processar dados. Isso já é feito pela ferramenta, basta especificar as fontes dos dados e as regras para a sua transformação.

Funções avançadas para categorizar e limpar dados: Ferramentas ETL possuem muitas funções para auxiliar na limpeza e categorização de dados. Além disso, o uso dessas ferramentas apresenta vantagens na transferência e transformação de um grande volume de dados com regras complexas, simplificando seu cálculo, mudança e integração.

Meta-informação: os metadados, ou seja, informações sobre como identificar, localizar e compreender os dados, são gerados automaticamente pelas aplicações ETL. Isso reduz falhas e auxilia a administração desses dados.

Principais conceitos em ETL

Business Intelligence (BI): ou inteligência de negócios, refere-se ao processo de coleta, organização, análise, compartilhamento e monitoramento de informações que oferecem suporte a gestão de negócios. É o conjunto de teorias, metodologias, processos, estruturas e tecnologias que transformam uma grande quantidade de dados brutos em informação útil para tomadas de decisões estratégicas.

Descreve a capacidade de a empresa ter acesso e explorar seus dados, desenvolvendo percepção e conhecimento, o que leva à melhora do processo de tomada de decisões. Sua infraestrutura tecnológica é composta de data warehouse ou data marts, Data Mining e ODS, Benchmarking, além das ferramentas pertinentes.

Figura 29 – Composição do BI.



Fonte: <https://www.oficinadanet.com.br>.

Modelagem Dimensional: também chamada de modelagem multidimensional, é a técnica de modelagem de banco de dados para o auxílio às consultas do data warehouse nas mais diferentes perspectivas. A visão multidimensional permite o uso mais intuitivo para o processamento analítico pelas ferramentas OLAP (Online Analytical Processing). Toda modelagem dimensional possui dois elementos imprescindíveis: as tabelas Fatos e as tabelas Dimensões. Ambas são obrigatórias e possuem características complementares dentro de um data warehouse.

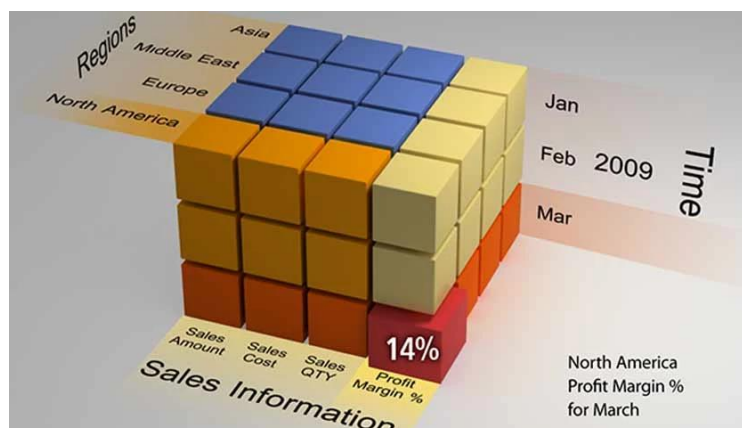
Data warehouse: é um depósito de dados digitais que serve para armazenar informações detalhadas relativamente a uma empresa, criando e organizando relatórios através de históricos que são depois usados pela empresa para ajudar a tomar decisões importantes com base nos fatos apresentados.

Data mart: é uma subdivisão ou subconjunto de um DW. Os data marts são como pequenas fatias que armazenam subconjuntos de dados, normalmente organizados para um departamento ou um processo de negócio. Normalmente o data mart é direcionado para uma linha de negócios ou equipe, sendo que a sua informação costuma pertencer a um único departamento.

OLAP: Online Analytical Processing ou Processo Analítico em Tempo Real, é uma das ferramentas mais usadas para a exploração de um data warehouse. O OLAP possibilita alterar e analisar grandes quantidades de dados em várias perspectivas diferentes. As funções básicas do OLAP são: visualização multidimensional dos dados, exploração, rotação, vários modos de visualização.

OLAP e o data warehouse são destinados a trabalharem juntos, enquanto o DW armazena as informações de forma eficiente, o OLAP deve recuperá-las com a mesma eficiência, porém com muita rapidez. Um cubo OLAP é uma estrutura de dados montada de forma multidimensional, e que proporciona uma rápida análise de valores quantitativos ou medidas relacionadas com determinado assunto, sob diversas perspectivas diferentes.

Figura 30 – Demo de Cubo com Dimensões e Medidas.



Fonte: <https://ayltoninacio.com.br/blog/>.

Staging Area: é uma localização temporária onde os dados dos sistemas de origem são copiados, facilitando a integração dos dados antes de sua atualização DW. Tem como função agilizar o processo de consolidação, proporcionando um melhor desempenho na fase da atualização dos dados.

A staging area é o único lugar para determinar os valores que vêm efetivamente dos sistemas legados. A staging area deve ser usada para limpeza dos dados que entram no processo de extração e transformação.

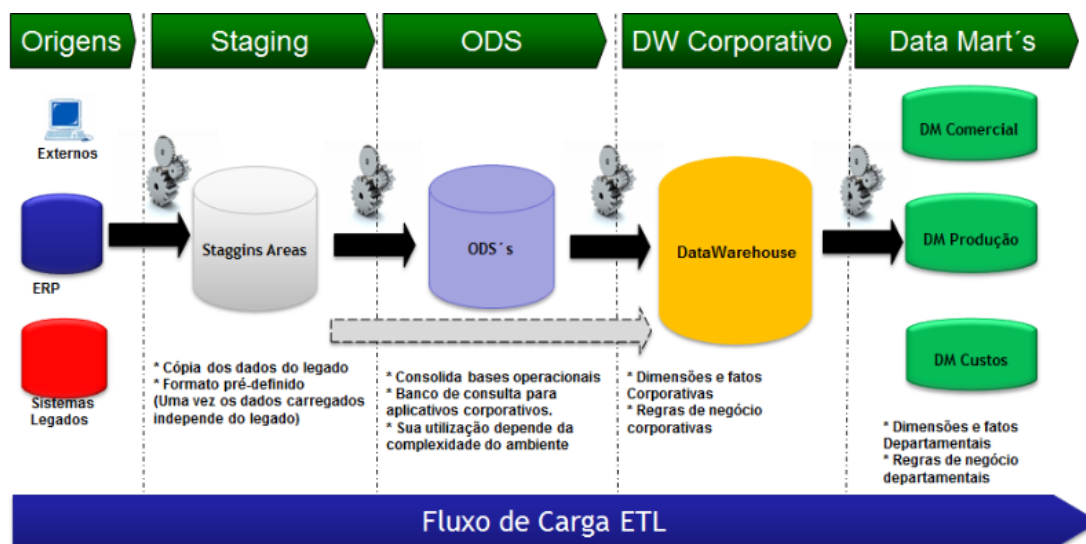
ODS: Operational Data Store é um repositório de dados onde são colocados os dados que a empresa trabalha no seu dia a dia, para que sejam consultados por outros sistemas, ou por áreas de inteligência. Um ODS reúne dados de várias aplicações e não é semelhante a um data warehouse, pois não tem o compromisso de armazenar histórico de dados e de servir para processos de auditoria sobre esses dados. Entretanto o ODS deve armazenar dados que tem “valor” para seus consumidores e de manter-se atualizado.

Uma área de preparação normal destina-se apenas ao recebimento dos dados operacionais das origens transacionais, a fim de transformar os dados e carregá-los no armazém de dados. Um ODS também oferece essa funcionalidade, mas, além disso, pode ser consultada diretamente.

Dessa forma, as ferramentas de análise que precisam de dados mais próximos do tempo real podem consultar os dados do ODS à medida que são recebidos dos respectivos sistemas de origem, antes de operações demoradas de transformação e carregamento.

Data Lake: é um repositório que centraliza e armazena todos os tipos de dados gerados pela e para a empresa. Eles são depositados ali ainda em estado bruto, sem o processamento e análise e até mesmo sem uma governance. A ideia é manter na organização dados que podem ser estrategicamente úteis, mesmo que eles, na realidade, não sejam requeridos em nenhum momento posterior. O data lake seria, em alguns casos já é, o local de armazenamento dessas informações.

Figura 31 – Processos do ETL.



Fonte: <https://www.igti.com.br/>.

Ferramentas ETL

Algumas das ferramentas mais conhecidas de ETL são:

- IBM InfoSphere DataStage.
- Informática Power Center.

- SAP Business Objects Data Services.
- Microsoft SQL Server Integration Services (SSIS).
- Oracle Data Integrator (ODI).
- Pentaho Data Integration (PDI).

Temos que considerar as particularidades de cada projeto, como orçamento, tamanho, recursos, quantidade de usuários.

Big Data

Razões para usar o Big Data:

- Entender padrões;
- Prever situações;
- Criar fronteiras;
- Informar coleções de dados;
- Estimar parâmetros escondidos;
- Calibrar.

O Big Data traz novos desafios na gestão de dados, como a manutenção de uma linearidade dos dados, sua integridade e qualidade, a fim de que eles possam ser transformados em informação útil.

Soluções de Inteligência Operacional podem correlacionar e analisar dados de fontes variadas em várias latências (desde o batch, até o tempo real), para revelar informações importantes.

Data Analytics

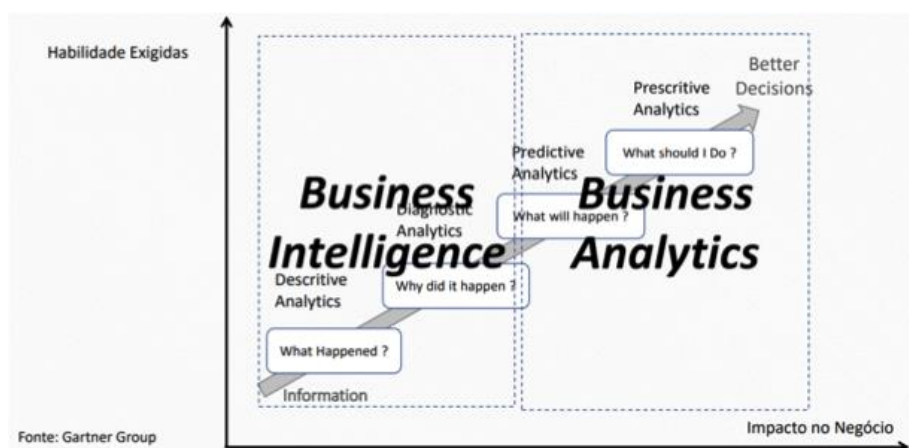
O Data Analytics pode ser usado em vários segmentos de mercado. Os bancos usam essa estratégia para evitar possíveis fraudes. Na educação, você pode medir o progresso dos alunos e avaliar a eficácia do sistema. No varejo, o principal uso é rastrear as características sociais e comportamentais dos clientes, de modo a prever tendências e hábitos.

Geralmente, o processo de análise do fluxo de dados segue as seguintes etapas:

- Coleta.
- Ingestão e transformação.
- Armazenamento.
- Análise.
- Desenvolvimento de algoritmos.
- Visualização.

É importante avaliar quais são os principais insights desejados na etapa de visualização, ou até mesmo o levantamento de quais problemas de negócio você gostaria de resolver.

Figura 32 – Business Intelligence x Business Analytics.



Fonte: Gartner Group.

Pentaho

A empresa americana Pentaho, desenvolveu uma suíte de softwares, para o desenvolvimento de uma solução em BI, desde a parte de levantamento de requisitos e DW, até a geração de dashboards para os usuários, essa suíte, chamada de Pentaho BI Suite Community Edition é bastante complexa e inclui ETL, OLAP, metadata, data mining, relatórios e dashboards.

Visão geral do Pentaho

- Pentaho é um software de código aberto comercial para negócios de Inteligência (BI). Foi desenvolvido desde 2004.
- Pentaho fornece relatórios abrangentes, OLAP análise, painéis, integração de dados, dados mineração e uma plataforma de BI.
- Benefícios:
 - Plataforma com código aberto (open source).
 - Tem uma comunidade de suporte aos usuários.
 - Funciona bem em multiplataforma - Windows, Linux, Macintosh, Solaris, Unix.
 - Tem pacote completo de relatórios, ETL para armazenamento de dados, mineração de dados do servidor OLAP e também painel de controle.

Pentaho Data Integration

O Pentaho Data Integration (PDI) é uma solução de extração, transformação e carregamento (ETL) que utiliza uma abordagem inovadora orientada por metadados.

- Migração de dados entre diferentes bancos de dados e aplicativos.
- Carga de grandes conjuntos de dados em bancos de dados, aproveitando ao máximo os ambientes de processamento em nuvem, em cluster e em paralelo.
- Limpeza de dados com etapas que variam de transformações muito simples a muito complexas.
- Integração de dados, incluindo a capacidade de alavancar ETL em tempo real como fonte de dados para o Pentaho Reporting.
- Prototipagem rápida de esquemas ROLAP.
- Funções do Hadoop: execução e programação de tarefas do Hadoop, design simples do Hadoop MapReduce, integração do Amazon EMR.

Benefícios do Pentaho

- Designer gráfico fácil de usar, com mais de 100 objetos de mapeamento prontos para uso, incluindo entradas, transformações e saídas.
- Arquitetura de plug-in simples para adicionar suas próprias extensões personalizadas.
- Designer integrado (Spoon) que combina ETL com modelagem de metadados e visualização de dados, fornecendo o ambiente perfeito para o desenvolvimento rápido de novas soluções de Business Intelligence.
- A arquitetura do mecanismo de streaming oferece a capacidade de trabalhar com volumes de dados extremamente grandes.
- Desempenho e escalabilidade de classe corporativa com uma ampla variedade de opções de implantação, incluindo servidores ETL dedicados, em cluster e/ou baseados em nuvem.

Pentaho Data Integration Suite

Kettle é um acrônimo para “Kettle E.T.T.L. Environment”.

- Extração, transformação, transporte e carregamento de dados.
- Spoon é uma interface gráfica do usuário que permite projetar transformações e tarefas que podem ser executadas com as ferramentas Kettle - Pan e Kitchen.
- Pan é um mecanismo de transformação de dados que executa diversas funções, como ler, manipular e gravar dados em/e de várias fontes de dados.
- É um programa que executa tarefas projetadas pelo Spoon em XML ou em um repositório de banco de dados.
- As tarefas geralmente são agendadas no modo de lote para serem executadas automaticamente em intervalos regulares.
- Faz parte da suíte do Data Integration, e utiliza as técnicas de ETL (Extract-Transform-Load), para a obtenção dos dados que virão das várias fontes de dados, e que obrigatoriamente teremos de cruzá-las em algum momento dentro do ciclo de ETL.
- Ele é capaz de ler e escrever em vários formatos de SGBD, como Oracle, PostgreSQL, SQLServer, MySql, entre outros, e importar arquivos texto como CSV, planilhas Excel e bases de dados ODBC (apenas em Windows).

Etapas do Projeto de BI

Esse é o passo a passo da implementação de um projeto de BI. São várias atividades necessárias para o levantamento, desenvolvimento e implantação do Business Intelligence. Essas atividades estão intrinsecamente ligadas e são imprescindíveis para o êxito do projeto

1. Levantamento de necessidades.

Essa é a etapa inicial. Nela, a equipe de especialistas irá conversar com os gestores para entender quais suas necessidades e que informações eles gostariam de extrair de seus dados. Selecione as pessoas-chave, de todos os segmentos da sua empresa, que irão utilizar a ferramenta e inclua-os nessa etapa. A primeira pergunta nesse momento é: o que você quer responder através do Business Intelligence? Liste todos os objetivos que deseja alcançar, sem se preocupar se já possui os dados para tanto ou não.

2. Identificação das fontes de dados e requisitos de informação.

Após mapear as necessidades, os analistas irão checar se o banco de dados da empresa já possui os dados necessários para responder às perguntas dos gestores. Nessa etapa, a equipe irá conversar com os gestores dos departamentos da empresa envolvidos, buscando entender quais os sistemas computacionais usados, onde estão e qual a qualidade dos dados. Aqui há 2 cenários possíveis:

- A empresa já possui um data warehouse. Os dados nesse local já estão tratados, portanto são confiáveis, se correlacionam quando possível e podem ser acessados facilmente. Empresas que possuem um data warehouse já estão com os dados prontos para serem usados e prontas para implementar a solução de BI.
- A empresa ainda não possui um data warehouse. Nesse caso, a equipe responsável pela implementação do BI irá auxiliar também na construção do armazém de dados. Será um pouco mais demorado, porém nada que não possa ser feito.

3. Planejamento.

Nessa etapa ocorre o detalhamento e a documentação de toda a estrutura do projeto. Feito isso, a equipe irá validar o projeto com os gestores.

4. Implementação da ferramenta.

Aqui o software de BI escolhido pela empresa será de fato implementado. Os especialistas irão desenvolver as aplicações necessárias para responder às perguntas levantadas na primeira etapa. Essa etapa pode ser dividida em 3 subetapas:

- **Carga de dados:** ou ETL, onde serão extraídos os dados dos locais definidos na etapa 2, transformá-los segundo certos critérios e carregá-los no banco de dados.
- **Desenvolvimento:** utilizar os dados gerados no ETL e criar as dashboards, levando sempre em consideração as necessidades e regras de negócio definidas na etapa 1.
- **Testes:** essa etapa é realizada simultaneamente com a implementação do BI para evitar grande quantidade de retrabalho. Serão realizados diversos testes para checar se tudo está saindo como o esperado.

5. Disponibilidade aos usuários.

Essa etapa consiste na entrega do produto ao usuário final. Essa etapa envolve também o processo de capacitação dos colaboradores que irão utilizar a ferramenta no dia a dia. Como essa capacitação irá ocorrer dependerá da equipe contratada: pode ser por meio da criação de manuais, de treinamentos ou de suporte.

6. Retroação.

Aqui o projeto de BI já está implementado e funcionando. O processo de retroação é a melhoria contínua das funcionalidades da ferramenta. A equipe estará disponível para fazer ajustes necessários, melhorar aplicações já feitas, e personalizar cada vez os recursos de acordo com o cenário da empresa.

Componentes críticos para escolha da ferramenta

Carregamento incremental: permite atualizar o DW com novos dados sem fazer uma recarga completa de todo o conjunto de dados.

Auditoria e registro: é necessário registro detalhado no pipeline ETL para garantir que os dados possam ser auditados após serem carregados e que os erros possam ser depurados.

Manipulação de vários formatos de origem: para obter dados de diversas fontes, como a API do Salesforce, o aplicativo financeiro de back-end e bancos de dados como MySQL e MongoDB, o processo precisa ser capaz de lidar com uma variedade de formatos de dados.

Tolerância a falhas: em qualquer sistema, ocorrem inevitavelmente problemas. Os sistemas ETL precisam ser capazes de se recuperar normalmente.

Suporte para notificação: se você deseja que sua organização confie nas análises, é necessário criar sistemas de notificação para alertá-lo quando os dados não forem precisos.

Baixa latência: algumas decisões precisam ser tomadas em tempo real, portanto a atualização dos dados é fundamental. Embora existam restrições de latência impostas por integrações de dados de origem específicas, os dados devem fluir pelo seu processo de ETL com a menor latência possível.

Requisitos para ETL

Antes de iniciar um projeto de ETL é necessário que os seguintes itens estejam bem alinhados.

- Requisitos de Negócio.
 - Foco na decisão.

- Questões de informação.
- Prioridade.
- Viabilidade dos dados.
 - Performance.
- Latência dos dados.
 - Histórico.
- Política de compliance e segurança.
 - Segurança.

Foco na decisão: este deve ser o objetivo inicial de um sistema de apoio à decisão. As primeiras questões direcionadas aos usuários devem ser elaboradas de maneira a identificar: Quais os objetivos do negócio que necessitam análises? Quais decisões devem ser tomadas a fim de satisfazer estes objetivos? Com estas questões respondidas, já demos um grande passo na formalização dos objetivos que devem nortear o desenvolvimento do projeto.

Questões relacionadas a informação: com objetivos e decisões identificadas, precisamos perguntar ao usuário: Que informações você necessita para tomar estas decisões? Quais medidas (indicadores, KPIs etc.) melhor refletem estas informações? Com informações e medidas na mão, temos identificados grande parte dos fatos e dimensões de negócio.

Prioridade: quão importante para o negócio são as decisões e as informações identificadas nos tópicos anteriores? Identificar as prioridades é essencial, principalmente se no futuro, por qualquer motivo, tivermos que ‘fatiar’ ou reduzir o escopo do projeto.

Performance: quanto tempo o usuário tem para tomar determinada decisão? Com que frequência deve ser tomada esta decisão? Quanto tempo é aceitável entre o momento que um evento acontece no mundo real e o momento

que este evento é observado no BI? Essas questões nos ajudam a definir as necessidades de tempo de resposta, frequência de integração e latência.

Histórico: quanto do passado é necessário estar disponível para auxiliar a tomada de decisão? Requisitos de histórico para cada decisão podem explicitar oportunidades de estratégias heterogêneas de histórico.

Segurança: existe restrição de quem deve acessar as informações? Muitas vezes, questionamentos associadas a políticas de segurança são feitos tardiamente, somente no momento de desenvolvimento das aplicações de acesso as informações. Não raro, os requisitos se mostram bem complexos, trazendo surpresa desagradáveis de impacto ao cronograma. Portanto, esta etapa tão importante, executada no início do projeto, deve direcionar todo o desenvolvimento.

Componentes críticos

- Carregamento incremental.
- Auditoria e registro.
- Manipulação de vários formatos de origem.
- Tolerância a falhas.
- Suporte para notificação.
- Baixa latência.
- Escalabilidade.
- Precisão.

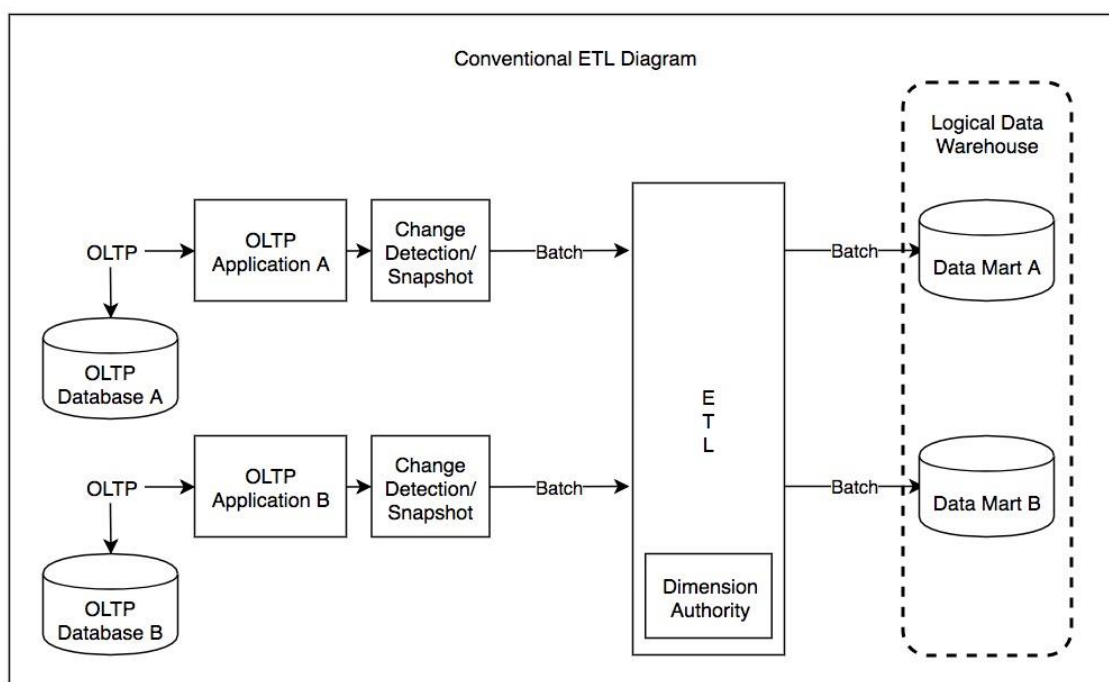
Definição de ambiente

Os processos de ETL podem envolver complexidade considerável e problemas operacionais significativos podem ocorrer com sistemas de ETL projetados incorretamente.

O intervalo de valores ou qualidade dos dados em um sistema operacional pode exceder as expectativas dos projetistas no momento em que as regras de validação e transformação são especificadas.

O perfil de dados de uma fonte, durante a análise de dados, pode identificar as condições de dados que devem ser gerenciadas pelas especificações de regras de transformação, levando a uma alteração das regras de validação implementada de forma explícita e implícita no processo ETL.

Figura 33 – Diagrama ETL.



Fonte: <https://www.wikiwand.com/>.

Escalabilidade

Os data warehouses são normalmente montados a partir de uma variedade de fontes de dados com diferentes formatos e finalidades. O ETL é um processo essencial para reunir todos os dados em um ambiente padrão e homogêneo.

Escalabilidade: a análise de projeto deve estabelecer a escalabilidade de um sistema ETL durante toda a vida útil de seu uso - incluindo a compreensão dos volumes de dados que devem ser processados nos contratos de nível de serviço. Alguns sistemas ETL precisam ser dimensionados para processar terabytes de dados para atualizar os data warehouses com dezenas de terabytes de dados.

O aumento do volume de dados pode alterar o que foi pensado como arquitetura do BI.

Recuperabilidade ou Rerunnability

Os procedimentos de Data Warehousing geralmente subdividem um grande processo de ETL em partes menores, executando sequencialmente ou em paralelo. Para acompanhar os fluxos de dados, faz sentido marcar cada linha de dados com "row_id" e marcar cada parte do processo com "run_id". Em caso de falha, essas IDs ajudam a reverter e executar novamente a peça com falha.

A melhor prática também exige pontos de verificação, que são estados em que determinadas fases do processo são concluídas. Uma vez em um ponto de verificação, é uma boa ideia gravar tudo em disco, limpar alguns arquivos temporários, registrar o estado etc.

Chaves

Chaves exclusivas desempenham um papel importante em todos os bancos de dados relacionais, pois unem tudo. Uma chave exclusiva é uma coluna que

identifica uma determinada entidade, enquanto uma chave estrangeira é uma coluna em outra tabela que se refere a uma chave primária.

As chaves podem compreender várias colunas; nesse caso, são chaves compostas.

Em muitos casos, a chave primária é um número inteiro gerado automaticamente que não tem significado para a entidade de negócios que está sendo representada, mas existe apenas para o objetivo do banco de dados relacional - geralmente chamado de chave substituta.

Performance

Os fornecedores de ETL avaliam seus sistemas de registro em vários TB (terabytes) por hora (ou ~ 1 GB por segundo) usando servidores poderosos com várias CPUs, vários discos rígidos, várias conexões de rede gigabit e muita memória.

A parte mais lenta de um processo ETL geralmente ocorre na fase de carregamento do banco de dados. Os bancos de dados podem ter um desempenho lento porque precisam cuidar da simultaneidade, manutenção da integridade e índices. Mesmo usando operações em massa, o acesso ao banco de dados geralmente é o gargalo no processo ETL. Alguns métodos comuns usados para aumentar o desempenho são:

- Observe as tabelas de partição (e índices): tente manter partições de tamanho semelhante.
- Observe o catálogo.
- Faça toda a validação na camada ETL antes do carregamento.
- Desativar gatilhos (triggers) nas tabelas de banco de dados de destino durante o carregamento.

- Gere IDs na camada ETL (não no banco de dados).
- Dedicar tempo à modelagem.
- Monitorar os comandos SQL durante o ETL.
- Utilizar índices em colunas muito acessadas.
- Observar as boas práticas nos comandos SQL:
 - Evite o SELECT *.
 - Desvantagem do ORDER BY e DISTINCT. Só utilize se for realmente necessário.
 - Joins em excesso.
 - Valores calculados devem ser gravados?
 - Comando Truncate Table x Delete.
 - Conversões com UPPER, TO_CHAR e etc em cláusulas WHERE.

Processamento paralelo

Um desenvolvimento recente no software ETL é a implementação do processamento paralelo. Ele permite vários métodos para melhorar o desempenho geral do ETL ao lidar com grandes volumes de dados. Os aplicativos ETL implementam três tipos principais de paralelismo:

- **Dados:** dividindo um único arquivo sequencial em arquivos de dados menores para fornecer acesso paralelo.
- **Pipeline:** permitindo a execução simultânea de vários componentes no mesmo fluxo de dados, por exemplo procurando um valor no registro 1 ao mesmo tempo em que adiciona dois campos no registro 2.

- **Componente:** a execução simultânea de vários processos em diferentes fluxos de dados no mesmo trabalho, por exemplo classificando um arquivo de entrada enquanto remove duplicatas em outro arquivo.

Todos os três tipos de paralelismo geralmente operam combinados em um único trabalho ou tarefa.

Ferramentas

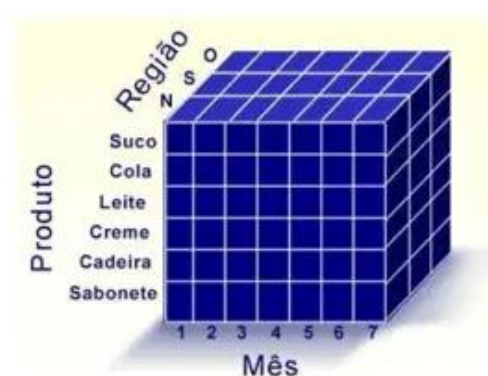
Ao usar uma estrutura estabelecida de ETL, é possível aumentar as chances de terminar com melhor conectividade e escalabilidade. Uma boa ferramenta de ETL deve ser capaz de se comunicar com os diversos bancos de dados relacionais e ler os vários formatos de arquivo usados em uma organização.

As ferramentas ETL, na maioria dos casos, contêm uma GUI que ajuda os usuários a transformar convenientemente os dados, usando um mapeador de dados visual, em vez de gravar programas grandes para analisar arquivos e modificar tipos de dados.

Modelagem dimensional

A modelagem dimensional é a técnica de modelagem de banco de dados para o auxílio às consultas do data warehouse, nas mais diferentes perspectivas, onde as informações se relacionam de maneira que podem ser representadas metaforicamente como um cubo.

Figura 34 – Cubo OLAP.



Fonte: Nardi (2007).

Podemos fatiar este cubo e aprofundar em cada dimensão ou eixo para extrair mais detalhes sobre os processos internos que ocorrem na empresa, que em um modelo relacional torna-se muito complicados de serem extraídos e muitas vezes até impossíveis de serem analisadas.

O modelo dimensional permite visualizar dados abstratos de forma simples e relacionar informações de diferentes setores da empresa de forma muito eficaz.

Toda modelagem dimensional possui dois elementos imprescindíveis: as tabelas Fatos e as tabelas Dimensões. Ambas são obrigatórias e possuem característica complementares dentro de um data warehouse.

Modelo Star Schema

O Star Schema, idealizado por Ralph Kimball, é o modelo mais utilizado na modelagem dimensional para dar suporte à tomada de decisão e melhorar a performance de sistemas voltados para consulta.

O modelo é composto no centro por uma tabela Fato, rodeada de dimensões, ficando parecido com a forma de uma estrela.

Dessa forma, as tabelas dimensionais devem conter todas as descrições que são necessárias para definir uma classe como Produto, Tempo ou Loja nela mesma, ou seja, as tabelas de dimensões não são normalizadas no modelo estrela, então campos como Categoria, Departamento, Marca contém suas descrições repetidas em cada registro, assim aumentando o tamanho das tabelas de dimensão por repetirem estas descrições de forma textual em todos os registros.

Modelo SnowFlake

No modelo SnowFlake, defendido por Bill Inmon, as tabelas dimensionais relacionam-se com a tabela de Fatos, mas algumas dimensões relacionam-se apenas entre elas. Isto ocorre para fins de normalização das tabelas dimensionais, visando diminuir o espaço ocupado por estas tabelas, então informações como Categoria, Departamento e Marca tornaram-se tabelas de dimensões auxiliares.

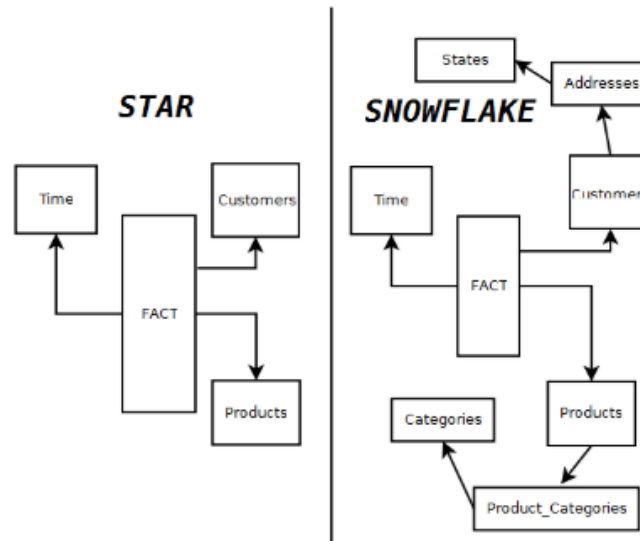
Construindo a base de dados desta forma, passamos a utilizar mais tabelas para representar as mesmas dimensões, mas ocupando um espaço em disco menor do que o modelo Estrela.

Este modelo chama-se Snow Flake, pois cada dimensão se divide em várias outras tabelas, onde organizadas de certa forma lembra um floco de neve.

Modelo StarFlake

É um híbrido entre os modelos Star Schema e Snowflake, aproveitando o melhor de cada um.

Figura 35 – Star Schema x Snowflake.



Fonte: <http://rpblogbi.blogspot.com/2018>.

Figura 36 – Considerações sobre os modelos dimensionais.

	Resumo	
	Star Schema	Snowflake
Clareza	mais fácil	mais difícil
Número de tabelas	Menor <	Maior >
Complexidade de consultas	Consultas mais simples	Consultas mais complexas
Desempenho	Menor número de chaves estrangeiras consequentemente mais rápido a consulta	Maior número de chaves estrangeiras, consequentemente mais lento
Forma Normal	2FN Desnormalizada	A 3FN normalização das dimensões também tem efeito negativo
Joins	Poucos	Muitos
Manutenção	Possui redundância	Sem redundância, pois está na 3FN
Tipo DW	Pequenos	Grandes
Tabelas Dimensão	Somente uma para cada dimensão	Mais que uma para cada dimensão
Normalização	Tanto as dimensões quanto as fatos DESNORMALIZADAS	Tabelas dimensão normalizadas, porém a fato desnormalizada
Modelo	Top Down	Bottom-Up
Quando usar	Quando as tabelas dimensões forem menores	Quando as tabelas dimensões forem maiores
Idealizador	Ralph Kimball	Bill Inmon

Fonte: <http://rpblogbi.blogspot.com/2018>.

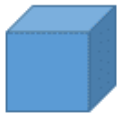

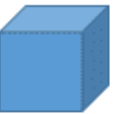
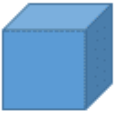
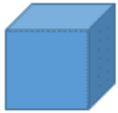




OLAP



A visão multidimensional permite o uso mais intuitivo para o processamento analítico pelas ferramentas OLAP (Online Analytical Processing).

O OLAP possui um conjunto de técnicas para o tratamento dos dados contidos na visão multidimensional do data warehouse. As ferramentas OLAP podem suportar os diferentes tipos de arquitetura: MOLAP, ROLAP ou HOLAP.

O MOLAP é um tipo de arquitetura que utiliza estrutura de banco de dados multidimensional. O ROLAP utiliza a arquitetura relacional dos dados, onde o banco de dados possui a estrutura tradicional. Já o HOLAP (Híbrido) é a junção das duas anteriores, utilizando os melhores aspectos e recursos de cada um dos dois tipos.

Figura 37 – Considerações sobre os modelos dimensionais.

	MOLAP	HOLAP	ROLAP
Cube Structure			
Preprocessed Aggregates			
Detail-Level Values			

 Multidimensional Storage
  Relational Storage

Fonte: <https://fard-solutions.com/2016/>.

MOLAP	ROLAP	HOLAP
▪ Mais rápido (quase	▪ Mais lento (quase	▪ Tão rápido quanto

sempre). <ul style="list-style-type: none"> ▪ Mais armazenamento total em disco. ▪ Instantâneo. 	sempre). <ul style="list-style-type: none"> ▪ Suporta tempo real (com algumas ressalvas). 	MOLAP para agregações. <ul style="list-style-type: none"> ▪ Tão lento quanto ROLAP para dados no nível da “folha”.
---	--	---

Hierarquia de dimensões

É o conjunto de atributos que possuem uma ordem lógica do maior ao menor nível, é uma forma hierárquica de organizar os dados nas dimensões, por exemplo:

- Ano, mês e dia.
- Categoria, subcategoria e produto.
- Capitão, sargento, cabo e soldado.

As dimensões podem relacionar-se através de **hierarquias**. É comum existir apenas uma hierarquia por dimensão, mas podem existir múltiplas, se necessário. A hierarquia existe apenas nas dimensões, porque a métrica é só um valor, e quem vai dizer se esse valor está correspondendo a um determinado nível da hierarquia é o atributo.

O grão, ou detalhe, é o menor nível da hierarquia da dimensão. É a informação base, o menor detalhe da informação. É muito comum o cliente querer esse tipo de hierarquia para poder ir descendo ou subindo o nível da informação, é o que chamamos de drill down e drill up. Assim podemos ter uma visualização por ano e fazer um drill down, tendo a visualização por mês e depois por dia (que seria o grão).

Surrogate Key

Em um banco de dados, as chaves são usadas para identificar as linhas de uma tabela e fazer as conexões entre elas. No data warehouse, temos a Surrogate Key nas dimensões, que é a chave artificial utilizada para conectar a tabela na Fato. A Surrogate Key nada mais é que a Primary Key da dimensão.

A Surrogate Key é uma chave artificial e auto incremental. A palavra artificial vem do tipo, porque ela não existe em lugar nenhum, não está lá no transacional como a Natural Key (PK que vem do legado), ela é criada no data warehouse. E é auto incremental porque toda vez que é chamada, troca de número, então ela começa com 1 e vai indo para 2, 3, 4 e assim por diante. Ela é gerada automaticamente na hora da carga, quando você carrega a dimensão no ETL.

Na tabela Fato, essa Surrogate Key vai ser uma Foreign Key, a chave que serve para relacionar os dados entre duas tabelas, sempre apontando para uma Primary Key em outra tabela, que no caso da dimensão, vai ser a Surrogate Key. Assim, a tabela Fato receberá apenas o código da Surrogate Key da linha que ela está referenciando e não os atributos.

Slowly Changing Dimension (SCD)

Slowly Changing Dimensions (SCD ou Dimensões que Mudam Lentamente) retrata as dimensões que sofrem atualizações em seus campos e os classifica pelo tipo de mudança existente em cada uma delas. Todas as dimensões são SCD, porque elas vão precisar atualizar para se manterem sincronizadas com o transacional, a única exceção é a dimensão de tempo, que a gente chama de tipo 0, porque depois que os dados foram inseridos, não precisa mais atualizar. Apesar do SCD tipo 2 ser predominante e normalmente utilizado, podem haver situações onde outros tipos possuam melhores aplicabilidades. Cabe analisar cada um e verificar a melhor estratégia para o versionamento dos dados, mantendo, assim, a base histórica do DW com alto grau de precisão e confiabilidade.

Vários tipos de SCD podem ser identificados no DW, variando de acordo com as características de atualizações das Dimensões. As alternativas mais comuns de SCD são:

- SCD Tipo 1.
- SCD Tipo 2.
- SCD Tipo 3.
- SCD Híbrido.

SCD Tipo 1: é a alteração que não armazena histórico na Dimensão, ou seja, não é feito o versionamento do registro modificado. Trata-se do tipo mais simples, pois não há nenhum controle específico para a atualização dos dados, havendo apenas a sobreposição.

SCD Tipo 2: é a técnica mais utilizada para atualizações de dimensões. Nesse tipo de SCD é adicionado um novo registro com as mudanças, preservando sempre os dados anteriores. Dessa forma, os registros da tabela Fato vão apontar para a versão correspondente nas dimensões de acordo com a data de referência.

SCD Tipo 3: permite manter as modificações no mesmo registro. Essa técnica funciona com a adição de uma nova coluna na tabela de Dimensão, onde é armazenada a atualização, mantendo na antiga coluna o valor anterior.

SCD Híbrido: conhecido também como SCD Tipo 6, combina todas os SCD anteriores. Isso o torna bastante flexível para as atualizações das dimensões, porém com um grande custo de complexidade. Na solução híbrida é combinado os SCD de acordo com a estratégia e conveniência, sendo mais completo que os demais SCD. Dessa forma é flexibilizado as atualizações, de maneira que melhor se adeque às modificações dos dados nas dimensões.

Tabela Fato

A tabela Fato é a principal tabela do data warehouse, ela vai se conectar nas dimensões, podem existir uma ou mais tabelas Fato. Elas armazenam principalmente:

- Métricas – que são os fatos propriamente ditos (tudo que a empresa for mensurar é uma métrica).
- Foreign key – chave estrangeira que serve para relacionar os dados das Dimensões com a Fato.

As tabelas Fatos são classificadas pelas suas granularidades, mas independentemente de sua granularidade, cada métrica em uma tabela Fato deve estar exatamente no mesmo nível de detalhe. Existem 6 tipos de fatos:

- Fato transacional.
- Fato agregada.
- Fato consolidada.
- Fato snapshot periódico.
- Fato de snapshot acumulado.
- Fato sem Fato.

Fato Transacional

As tabelas Fatos transacionais são as mais comuns, geralmente utilizam métricas aditivas, aquelas que somam por todas as Dimensões. Há 2 formas de armazenar os dados em uma tabela Fato transacional.

- Em uma forma de transação por linha.

- Em uma forma de linha por transação – é a mais utilizada.

Fato Agregada

A tabela Fato agregada serve para juntar uma grande quantidade de dados quando não precisar analisar no nível do grão, com a função de acelerar o desempenho das consultas. Assim, cria uma Fato agregada com os dados que precisa e poderia permanecer com uma Fato normal e outra agregada. Exemplo: Uma tabela Fato visitantes e Dimensões de data e hora de um determinado site. A Fato vai monitorar sempre que alguém acessa o site, mas não os visitantes por minuto, porém posso precisar ver de forma consolidada.

Fato Consolidada

A tabela Fato consolidada é bem parecida com a agregada, mas serve para combinar 2 tipos de processos (área de negócio, área de assunto, processo de negócio). A Fato consolidada serve para consolidar duas tabelas Fato. No processamento do ETL, na hora de carregar a tabela Fato, carrega uma, carrega a outra, e mistura as duas. O grão precisa ser o mesmo entre as duas. Exemplo: tem uma Fato Venda e depois precisa juntar ela com uma Fato Venda Orçada. Depois disso, cria uma Fato Consolidada e coloca nela o valor real e o orçado, que antes estavam em fatos diferentes.

Fato SnapShot Periódico

As tabelas snapshots periódico registram dados que é instantâneo em um período de tempo predefinido, podendo ser diariamente, semanalmente ou mensalmente. Como o nome indica, tira-se uma "imagem do momento" em que o fato ocorreu, os dados de origem da tabela snapshot periódico são dados de uma tabela de Fato transacional em que se escolhe um período a ser capturado.

Fato SnapShot Acumulados

As tabelas de snapshots acumulado descreve a atividade de um processo de negócios que possui início e fim. Esse tipo de tabela de Fato possui várias colunas de data para representar marcos no processo. A medida em que as etapas do processo forem sendo concluídas, o registro correspondente na tabela de Fato é atualizado.

Fato sem Fato

As tabelas de Fato sem fato são encontradas na modelagem de data warehouse, esta tabela não possui nenhuma medida, ela contém apenas chaves estrangeiras para tabelas dimensionais, sendo suficiente para responder a questões relevantes, ou seja, seria uma tabela Fato sem métricas. Ela também é chamada de Fato de Associação ou de Intersecção, mas o termo técnico é Fato sem Fato ou Factless Fact Table.

Serve para fazer uma intersecção de Dimensões. Às vezes a gente quer comparar ou cruzar algo somente entre duas ou mais Dimensões e não tem uma métrica para fazer essas comparações. Essa tabela Fato é exceção, só é usada quando se precisa fazer uma intersecção entre as Dimensões.

Métricas

Métricas são medidas brutas que servem de subsídios aos indicadores. São compostas por vários tipos, como valor, quantidade, peso, volume ou outro formato quantitativo. Tudo que a empresa quer mensurar é métrica, geralmente sendo o que o usuário quer medir, por exemplo:

- Número de vendas.
- Seguidores em determinada rede social.

- Quantidade de propostas implantadas.
- Quantidade de beneficiários.

Tipos de métricas

Por mais que todas as métricas sejam numéricas, elas têm significados e interpretações diferentes, e é para identificar isso e saber que tipo de cálculos e cruzamentos pode fazer com cada uma, que servem os tipos de métricas. Os 4 tipos de métricas são:

- Aditivas.
- Derivadas.
- Semi-aditivas.
- Não-aditivas.

Métrica aditiva

São as métricas que permitem operações matemáticas como soma e subtração por todas as dimensões. Dentro da Fato há diversas linhas, e as métricas aditivas devem poder somar todas elas. Alguns exemplos de métricas aditivas:

- Quantidade de vendas.
- Valor da venda (se não for calculado).
- Quantidade de colaboradores.
- Quantidade de demissões.
- Quantidade de admissões.

Métrica derivada

São métricas que já estavam na Fato e que são calculadas, criando uma nova métrica, que chamamos de derivada. Por exemplo, uma métrica derivada pode ser a multiplicação da quantidade da venda com o preço unitário da Dimensão de produto. Ela pode ser armazenada diretamente na Fato ou calculada em tempo de execução nos cubos na ferramenta OLAP.

Métrica semi-aditiva

Métrica semi-aditiva pode ser somada por todas as dimensões menos de tempo, exceto se colocar um filtro que diga que ele só pegue o último registro. Saldo de estoque e saldo bancário, quando representado de forma monetária, são métricas semi-aditivas bem comuns, porque são aditivas em todas as dimensões, exceto no tempo.

Métrica não-aditiva

São métricas tipo percentual, algum cálculo feito em tempo de execução, que não podem ser somadas por nenhuma Dimensão.

ODS – Operational Data Store

Operational Data Store (armazenamento de dados operacional) é um repositório de dados, onde são colocados os dados que a empresa trabalha no seu dia a dia, para que sejam consultados por outros sistemas. Geralmente os dados são armazenados em um ODS através de ferramentas de softwares ETL, que extraem dados de banco de dados de diversas origens e coloca, de forma integrada, no ODS. Além disso, há estratégias que utilizam o ODS como base origem para os data warehouse.

Um ODS difere de uma base de dados de uma aplicação, pois reúne dados de várias aplicações e não é semelhante a um data warehouse, pois não tem o compromisso de armazenar histórico de dados e de servir para processos de auditoria sobre esses dados. Entretanto o ODS deve armazenar dados que tem “valor” para seus consumidores e de manter-se atualizado.

ODS é uma base de dados integrada, volátil, de valores correntes e que contém somente dados detalhados. Também pode ser entendido como uma visão integrada do mundo operacional. Normalmente, sua construção adota bases de dados relacionais. Algumas características do ODS:

- Possibilitar a integração de dados de várias aplicações.
- Ter alto desempenho na hora de armazenar seus dados e, principalmente, na hora de consultas sobre esses dados.
- Ter dados de negócio atualizados e ao mesmo tempo servir para processos decisórios.

Um dos benefícios do ODS é poder otimizar a criação do DW e possibilitar a realização de consultas relacionais sobre dados históricos.

Diferenças entre ODS e DW

As principais diferenças entre ODS e DW são que ODS é voltado para consultas com granularidades maiores, enquanto um DW normalmente é utilizado para consultas complexas e em dados agregados.

Outra diferença é que o ODS é para relatórios operacionais que exigem informações próximas ou em tempo real, enquanto um DW é destinado a análise de tendência histórica e, geralmente, trabalha com grande volume de dados. Além do que, ODS contém apenas uma pequena janela de dados, enquanto um DW contém todo o histórico de dados.

Um ODS fornece informações para decisões operacionais e táticas em dados em tempo real atuais ou próximo, enquanto um DW fornece um feedback para as decisões estratégicas que levam a melhorias gerais do sistema.

A frequência de carga de dados da ODS poderia ser a cada poucos minutos ou por hora, enquanto que em um DW a frequência das cargas de dados pode ser diária, semanal, mensal ou trimestral.

Staging area

Staging Area é uma localização temporária onde os dados dos sistemas de origem são copiados. Desta forma, ao invés de acessar os dados diretamente da fonte, o processo de “transformação” do ETL pega os dados da Staging Area para tratar e entregar os dados.

Em alguns casos, ao invés da “Staging” ser uma tabela temporária, pode ser uma view materializada que pode ser executada (manualmente ou mediante programação de carga) para ter os dados sempre atualizados.

Como alguns projetos precisam ter várias fontes de dados, a necessidade de termos uma “STAGING” acaba sendo obrigatório para que possa reunir o máximo de dados possível e, com isso, selecionar os dados, transformando-os em informação.

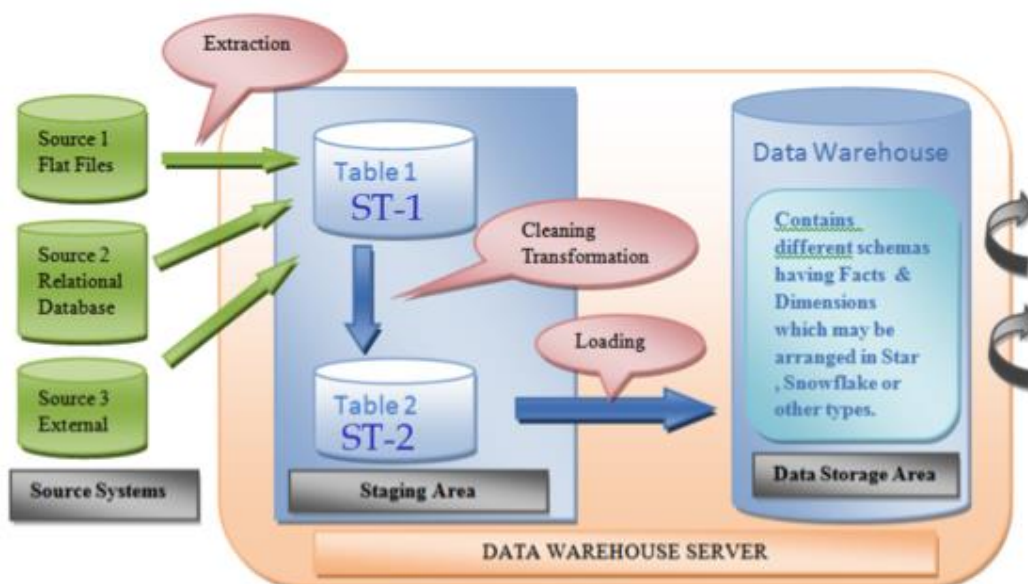
O benefício deste recurso é poder armazenar os dados em sua origem “bruta” para poder trabalhar em cima deles, ao invés de ficar sempre tendo que acessar a Fonte de dados. Assim, conseguimos evitar uma série de problemas, como baixa performance, por exemplo. Além disso, você tem a segurança que os dados estarão à sua disposição até que você execute algum processo que limpe a STAGING. A Staging area é uma área de armazenamento intermediário situada dentro do processo de ETL. Auxilia a transição dos dados das origens para o destino final no DW.

Tipos de Staging area

A Staging area pode possuir três tipos de stagings:

- Staging 1: Dados sem nenhuma transformação. Cópia exata da origem.
- Staging 2 Aux: Query com transformações. Visão do dia.
- Staging 2: Query com transformações. Visão histórica.

Figura 38 – Tipos de Staging Area.



Staging 1

A Staging 1 serve para executar a extração, busca os dados sem nenhuma transformação, cópia exata da origem. A extração, basicamente, seria buscar as informações dos sistemas legados e fontes externas da empresa, e colocá-las na Staging Área para validação, transformação e carga. Existem várias técnicas para fazer isso. O importante é termos as informações novas ou atualizadas, tendo assim um retrato dia a dia do que foi incluído, excluído e alterado. A partir disso não

precisamos mais do banco de dados de produção, ou seja, não corremos o risco de concorrer consumindo assim recursos dos sistemas legados.

Staging 2

A Staging 2 refere-se à transformação. Com os dados na Staging area podemos fazer as transformações necessárias. Essas transformações vão variar dependendo da modelagem e das fontes de dados. Ela utiliza os dados da staging 1 e utiliza os métodos da carga insert/update.

Staging 2 Aux

A Staging 2 Aux também se refere à transformação. A diferença que utiliza o método de carga Truncate.

Figura 39 – Comparativo dos Tipos de Staging Area.

Staging 1	Staging 2 Aux	Staging 2
Cópia dos dados	Query com as transformações	
Origem: dados transacionais e arquivos	Origem: ST1	
Toda a informação pertinente	Apenas campos que serão utilizados na dimensão	
Não possui chave primária	Possui chave primária	
Metodo Truncate	Metodo Truncate	Metodo Update / Insert
Pouco espaço de armazenamento	Pouco espaço de armazenamento	Maior espaço de armazenamento
Latência pequena	Latência pequena	Latência grande
ST1 Aux tem como objetivo copiar os dados da origem para não sobrecarregá-los nas operações necessárias ao ETL	ST2 Aux tem como objetivo otimizar o processo de carga diário	ST2 Aux tem como objetivo servir de base histórica para possíveis reprocessamentos

Tratamento de erros

Quando um componente de fluxo de dados aplica uma transformação aos dados da coluna, extrai dados de fontes ou carrega dados nos destinos, podem ocorrer erros. Frequentemente, os erros ocorrem por causa de valores de dados

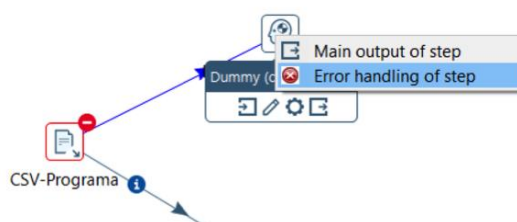
inesperados. Por exemplo, uma conversão de dados falha porque uma coluna contém uma cadeia de caracteres em vez de um número, uma inserção em uma coluna de banco de dados falha porque os dados são uma data e a coluna tem um tipo de dados numéricos, ou uma expressão não é avaliada porque o valor de uma coluna é zero, resultando em uma operação matemática que não é válida.

Categoria de erros

Em geral, os erros se enquadram nas categorias a seguir:

- **Erros de conversão de dados:** ocorrem se uma conversão resulta em perda de dígitos significativos, perda de dígitos insignificantes e truncamento de cadeias de caracteres. Os erros de conversão de dados também ocorrem se não houver suporte para a conversão solicitada.
- **Erros de avaliação de expressão:** ocorrem se expressões avaliadas em tempo de execução realizarem operações inválidas ou se tornarem sintaticamente incorretas devido a valores de dados ausentes ou incorretos.
- **Erros de pesquisa:** ocorrem se uma operação de pesquisa não conseguir localizar, por exemplo, uma tabela de pesquisa correspondente.

Figura 40 – Saídas de Erro.



Fonte: Pentaho.

Tipos de erros

Os erros fazem parte de uma de duas categorias: erros ou truncamentos.

- **Erros:** um erro indica uma falha inequívoca e gera um resultado NULL. Esses erros podem incluir erros de conversão de dados ou erros de avaliação de expressão.
- **Truncamentos:** um truncamento é menos grave que um erro. Um truncamento gera resultados que podem ser utilizáveis ou até mesmo desejáveis. Pode-se optar por tratar truncamentos como erros ou como condições aceitáveis.

Tratamento de erros

Opção	Descrição
Falha no Componente	A tarefa Fluxo de Dados falha quando ocorre um erro ou um truncamento. Falha é a opção padrão para um erro e um truncamento.
Ignorar Falha	O erro ou truncamento é ignorado e a linha de dados é direcionada para a saída da transformação ou fonte.
Redirecionar Linha	O erro ou truncamento da linha de dados é direcionado para a saída de erro da fonte, transformação ou destino.

Referências

ACADEMIA IN. *O que é e qual a importância de aprender sobre modelagem de dados?*. 2018. Disponível em: <<https://blog.academiain1.com.br/o-que-e-e-qual-a-importancia-de-aprender-sobre-modelagem-de-dados/>>. Acesso em: 05 fev. 2021.

ALENCAR, Felipe. *Entenda o que é sistema de arquivos e sua utilidade no PC e no celular*. TechTudo, 2016. Disponível em: <<https://www.techtudo.com.br/dicas-e-tutoriais/noticia/2016/02/entenda-o-que-e-sistema-de-arquivos-e-sua-utilidade-no-pc-e-no-celular.html>>. Acesso em: 05 fev. 2021.

ALVES, Gustavo Furtado de Oliveira. *O que é um SGBD?*. Dicas de Programação, 2012. Disponível em: <<https://dicasdeprogramacao.com.br/o-que-e-um-sgbd/>>. Acesso em: 05 fev. 2021.

BIX TECNOLOGIA. *Como funciona a implementação de um projeto BI?*. 2018. Disponível em: <<https://www.bixtecnologia.com.br/home/index.php/4886/como-implementar-um-projeto-de-bi/>>. Acesso em: 05 fev. 2021.

DANIEL, Eduardo Jose. *Glossário Pentaho Business Intelligence*. Ambiente Livre, 2020. Disponível em: <<https://www.ambientelivre.com.br/tutoriais-pentaho-bi/glossario-pentaho-business-intelligence.html>>. Acesso em: 05 fev. 2021.

DATA INTEGRATION. In: *Wikipédia*, a enciclopédia livre. Flórida: Wikimedia Foudation, 2020. Disponível em: <https://en.wikipedia.org/wiki/Data_integration#cite_note-refone-1>. Acesso em: 05 fev. 2021.

DATE, C. J.. *Introdução a Sistemas de Banco de Dados*. Elsevier Editora, 2004.

DEVMEDIA. *Desnormalização de bancos de dados*. Disponível em: <<https://www.devmedia.com.br/desnormalizacao-de-bancos-de-dados/38623>>. Acesso em: 05 fev. 2021.

DEV MEDIA. Disponível em: <<http://www.devmedia.com.br/>>. Acesso em: 05 fev. 2021.

EASIER. *Ferramenta de ETL*. Disponível em: <<http://gedxml.com.br/asa2/index.php/ferramenta-de-etl?view=featured>>. Acesso em: 05 fev. 2021.

ECKERSON, Wayne; WHITE, Colin. *Evaluating ETL and Data Integration Platforms*. USA, 2003.

ELIAS, Diego. *Entendendo o processo de ETL*. Canaltech, 2014. Disponível em: <<https://canaltech.com.br/business-intelligence/entendendo-o-processo-de-etl-22850/>>. Acesso em: 05 fev. 2021.

ELIAS, Diego. *O que é Business Intelligence? BI na prática*. Disponível em: <<https://www.binapratica.com.br/o-que-e-bi/>>. Acesso em: 05 fev. 2021.

ELMASRI, R.; NAVATHE, S. B.. *Sistemas de Banco de Dados*. 4. ed., Pearson-Addison-Wesley, 2005.

ERIKA. *Um estudo sobre as ferramentas OLAP*. DevMedia, 2007. Disponível em: <<https://www.devmedia.com.br/um-estudo-sobre-as-ferramentas-olap/6691>>. Acesso em: 05 fev. 2021.

E-SETORIAL. *Levantamento de requisitos para BI: uma questão de seguir o roteiro*. 2012. Disponível em: <<https://www.e-setorial.com.br/blog/88-levantamento-de-requisitos-para-bi-uma-questao-de-seguir-o-roteiro>>. Acesso em: 05 fev. 2021.

GAITONDE, Animesh. *NoSQL databases – Na Introduction*. Medium Analytics Vidhya, 2019. Disponível em: <<https://medium.com/analytics-vidhya/no-sql-databases-an-introduction-eb9706fbe3>>. Acesso em: 05 fev. 2021.

GODOI, Douglas. *O que é data warehouse?*. Cetax. Disponível em: <<https://www.cetax.com.br/data-warehouse/>>. Acesso em: 05 fev. 2021.

HITACHI. *Pentaho*. Disponível em: <https://community.hitachivantara.com/s/pentaho>>. Acesso em: 05 fev. 2021.

INACIO, Aylton. *Criando e alimentando um cubo OLAP físico no MySQL*. AyltonInacio Blog, 2018. Disponível em: <https://ayltoninacio.com.br/blog/criando-e-alimentando-um-cubo-olap-fisico-no-mysql>>. Acesso em: 05 fev. 2021.

KIMBALL GROUP. Disponível em: <http://www.kimballgroup.com/>>. Acesso em: 05 fev. 2021.

KIMBALL, Ralf. *The data warehouse Toolkit: The Complete Guide to Dimensional Modeling*. USA, 2002.

LUCIDCHART. *O que é um esquema de banco de dados?*. Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-um-esquema-de-banco-de-dados>>. Acesso em: 05 fev. 2021.

MACHADO, Felipe Nery Rodrigues. *Tecnologia e Projeto de data warehouse: Uma visão multidimensional*. São Paulo, Brasil, 2008

MUSARDO, Igor. *Modelos de consultas e relatórios AD-HOC para Sistemas de BI*. Musardos, 2008. Disponível em: <https://musardos.com/modelo-de-consultas-e-relatorios-ad-hoc-para-sistemas-de-bi/>>. Acesso em: 05 fev. 2021.

NOVATO, Douglas. *O que é Business Intelligence?* Oficina da Net, 2017. Disponível em: <https://www.oficinadanet.com.br/post/13153-o-que-e-business-intelligence>>. Acesso em: 05 fev. 2021.

OLIBONI, Daniel. *O que é SGBD?*. Oficina da Net, 2016. Disponível em: <https://www.oficinadanet.com.br/post/16631-o-que-e-um-sqbd>>. Acesso em: 05 fev. 2021.

ORACLE. *O Que É um Banco de Dados Relacional*. Disponível em: <https://www.oracle.com/br/database/what-is-a-relational-database/>>. Acesso em: 05 fev. 2021.

PITON, Rafael. *Tabela Dimensão: os 5 tipos que você deve conhecer*. Piton, 2017. Disponível em: <<https://rafaelpiton.com.br/blog/data-warehouse-tipos-dimensoes/>>. Acesso em: 05 fev. 2021.

POSITIVO. *O que é data lake*. 2018. Disponível em: <<https://www.meupositivo.com.br/panoramapositivo/o-que-e-data-lake/>>. Acesso em: 05 fev. 2021.

PRIMAK, Fábio Vinicius. *Decisões com B.I.: Business Intelligence*. Rio de Janeiro: Editora Ciência Moderna, 2008.

PROFESSOR DIGITAL. *Entidade: Atributos simples, compostos e multivalorados*. <<https://www.luis.blog.br/analise-de-entidade-atributos-simples-compostos-multivalorados.html>>. Acesso em: 05 fev. 2021.

RACKSPACE. *Diferenças e aprendizados sobre as ofertas da Rackspace*. Disponível em: <<https://www.rackspace.com/pt-br/library/>>. Acesso em: 05 fev. 2021.

REDAÇÃO FLEXA. *Como aplicar o business intelligence para pequenas e médias empresas*. Flexa, 2018. Disponível em: <<https://flexa.cloud/como-aplicar-o-business-intelligence-para-pequenas-e-medias-empresas/>>. Acesso em: 05 fev. 2021.

RENAN, Lucas. *Propriedades ACID*. Blog Lucas Renan, 2010. Disponível em: <<http://blog.lucasrenan.com/propriedades-acid/>>. Acesso em: 05 fev. 2021.

RICARDO. *Conceitos Fundamentais de Bancos de Dados*. DevMedia, 2006. Disponível em: <<https://www.devmedia.com.br/conceitos-fundamentais-de-banco-de-dados/1649>>. Acesso em: 05 fev. 2021.

RODRIGO. *Modelagem Dimensional – Snowflake*. RP Blog BI, 2018. Disponível em: <<http://rpblogbi.blogspot.com/2018/10/modelagem-dimENSIONAL-snowflake.html>>. Acesso em: 05 fev. 2021.

SANTANA, Eduardo. *Você já ouviu falar em Staging Area? Veja a contribuição dela no ETL*. Bufallos, 2017. Disponível em: <http://bufallos.com.br/bg_br/staging-area/>. Acesso em: 05 fev. 2021.

SIERRO, Sérgio. *O que é um Banco de Dados?*. Gran Cursos Online, 2020. Disponível em: <<https://blog.grancursosonline.com.br/o-que-e-um-banco-de-dados/>>. Acesso em: 05 fev. 2021.

SOFTWARE TESTING HELP. *Metadata In data warehouse (ETL) Explained Whith Examples*. 2020. Disponível em: <<https://www.softwaretestinghelp.com/metadata-in-data-warehouse-etl/>>. Acesso em: 05 fev. 2021.

SOLVIMM. *O que é ETL*. 2018. Disponível em: <<https://solvimm.com/blog/o-que-e-etl/>>. Acesso em: 05 fev. 2021.

SOUZA, Alexandre Morais. *Critérios de Seleção de Sistemas de Gerenciamento de Banco de Dados Não Relacionais em Organizações Privadas*. Dissertação (Mestrado em Ciências) – Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo. São Paulo, p. 151. 2013. Disponível em: <https://www.teses.usp.br/teses/disponiveis/100/100131/tde-26112013-181716/publico/Dissertacao_Alexandre_Souza_final.pdf>. Acesso em: 05 fev. 2021.

SYNNEX WESTCON. *O que é banco de dados como serviço (DBAAS)*. Disponível em: <<https://blogbrasil.westcon.com/o-que-e-banco-de-dados-como-servico-dbaas>>. Acesso em: 05 fev. 2021.

UESB, Informática. *data warehouse*. Disponível em: <<https://sites.google.com/site/informaticaesb/oracoes>>. Acesso em: 05 fev. 2021.

VIEIRA, Marcio Junior. Kimball University: As 10 Regras Essenciais para Modelagem de Dados Dimencional. In: *Tutoriais Pentaho Business Intelligence e Analytics*. Ambiente Livre, 2020. Disponível em: <<https://www.ambientelivre.com.br/tutoriais-pentaho-bi/kimball-university-as-10->

[regras-essenciais-para-a-modelagem-de-dados-dimENSIONAL.html](#)>. Acesso em: 05 fev. 2021.

VLAD, V. et al. *An Introduction to Cloud Databases*. O'Reilly Media, Inc, 2019.

WIKIWAND. *Extract, transform, load*. Disponível em: <[https://www.wikiwand.com/en/Extract, transform, load](https://www.wikiwand.com/en/Extract,_transform,_load)>. Acesso em: 05 fev. 2021.

ZEINA, Adem. *4 Types of NoSQL Databases*. Medium The Startup, 2020. Disponível em: <<https://medium.com/swlh/4-types-of-nosql-databases-d88ad21f7d3b>>. Acesso em: 05 fev. 2021.