

Banco de Dados NoSQL

PROFESSOR

Dr. Alexandre Savaris

ACESSE AQUI O SEU
LIVRO NA VERSÃO
DIGITAL!

EXPEDIENTE

FICHA CATALOGRÁFICA

Coordenador(a) de Conteúdo

Flavia Lumi Matuzawa

Projeto Gráfico e Capa

André Morais, Arthur Cantareli e
Matheus Silva

Editoração

Lucas Pinna Silveira Lima

Design Educacional

Laís Pinheiro de Souza Guelis

Curadoria

Elziane Vieira Alencar

Revisão Textual

Carlos Augusto Brito Oliveira

Ilustração

Eduardo Aparecido Alves
Wellington Vainer

Fotos

Shutterstock

C397 **CENTRO UNIVERSITÁRIO DE MARINGÁ**. Núcleo
de Educação a Distância. **SAVARIS**, Alexandre.

Banco de Dados NoSQL. Alexandre Savaris. Maringá -
PR: Unicesumar, 2022. Reimpresso em 2025.

180 p.

ISBN 978-85-459-2300-8

"Graduação - EaD".

1. Banco 2. Dados 3. NoSQL. 4. EaD. I. Título.

CDD - 22 ed. 005.1

Impresso por:

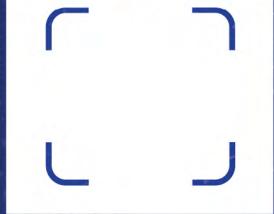
Bibliotecário: João Vivaldo de Souza CRB-9-1679



AVALIE ESTE LIVRO!



ACESSE O QR CODE



CRIAR MOMENTOS DE APRENDIZAGENS
INESQUECÍVEIS É O NOSSO OBJETIVO E POR ISSO,
GOSTARÍAMOS DE SABER COMO FOI SUA EXPERIÊNCIA.

Conta para nós! leva menos de 2 minutos. Vamos lá?!

DIGITE O CÓDIGO

02511311

Aa

RESPOnda A
PESQUISA

?

...

>>

MINHA HISTÓRIA

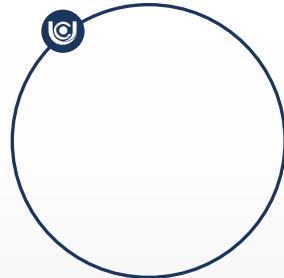
MEU CURRÍCULO



Alexandre Savaris

Olá, caro(a) aluno(a), sou o Prof. Alexandre Savaris, e estou aqui para lhe falar um pouco sobre mim. O gosto pela tecnologia veio cedo, a partir do meu primeiro contato com um computador pessoal aos 9 ou 10 anos de idade, durante um curso de informática na minha cidade natal. Somado aos videogames e à prática de voleibol, foram meus principais passatempos e atividades extraclasses durante a infância e adolescência. Hoje, os videogames diminuíram e o voleibol foi substituído pelo ciclismo (amador), mas a computação continua ocupando boa parte do meu tempo. Fiz graduação, especialização, mestrado e doutorado na área, e continuo sendo um aluno ativo e um consumidor quase compulsivo por tecnologia. Procurro equilibrar isso com muita leitura (preferencialmente de livros físicos, de papel), que considero insubstituíveis. A literatura de ficção (não apenas científica) ocupa boa parte do tempo que consigo dedicar a essa atividade, que complemento com algumas horas de filmes e séries.

Aqui você pode
conhecer um
pouco mais sobre
mim, além das
informações do
meu currículo.



<http://lattes.cnpq.br/7027246712839262>

PROVOCAÇÕES INICIAIS

BANCO DE DADOS NoSQL

Olá, caro(a) aluno(a), seja bem-vindo(a) à disciplina de Banco de Dados NoSQL. Apesar de serem naturalmente relacionados com mídias sociais, os cenários de Big Data que costumam utilizar esses bancos de dados estão presentes também nos ambientes institucionais. Empresas e instituições nas mais diversas áreas de atuação também são geradoras de conteúdo; esse conteúdo é operacional e decorre da sua atuação, demandando uma infraestrutura mínima que viabilize o seu funcionamento.

Neste livro, exploramos um possível cenário que gera e utiliza grandes volumes de dados em uma rede em que cada um de seus componentes busca ser categorizado como um Hospital sem Papel (Paperless Hospital). Essa categorização implica na escolha de tecnologias adequadas às demandas que surgem quando diferentes tipos de dados são gerados, quando fluxos de trabalho (workflows) são digitalizados, e quando resultados são apresentados de forma eficiente e ubíqua, sem atrasos e de forma consolidada em toda a rede.

Tudo começa com a definição de grandes volumes de dados a partir dos chamados “Vs” de Big Data, um conjunto de características presentes em cenários nos quais volume, velocidade e variedade na geração, aquisição, processamento e armazenamento de dados estão presentes. É apresentado também um teorema que objetiva auxiliar na classificação e na escolha de tecnologias com base em consistência, disponibilidade e tolerância a particionamentos.

O trabalho segue com a pesquisa de Sistemas Gerenciadores de Bancos de Dados (SGBDs) capazes de suportar as demandas provenientes dos Prontuários Eletrônicos dos Pacientes (PEPs), Registros Eletrônicos em Saúde (RES) e Sistemas de Registro Eletrônico em Saúde (S-RES). Como neles a organização costuma ser feita na forma de documentos, SGBDs orientados especificamente ao armazenamento desses artefatos são comentados e experimentos são feitos com um representante desta categoria de ferramenta (MongoDB).

Dados provenientes de equipamentos lotados em Unidades de Terapia Intensiva (UTIs) para monitoração e acompanhamento de condições de saúde agudas, ou mesmo dados provenientes de sensores para acompanhamento em regime de Telemedicina de condições de saúde crônicas são tema da Unidade 3, juntamente com a conceitua-

PROVOCACÕES INICIAIS

ção de séries temporais e a escolha e a experimentação de uma ferramenta capaz de gerenciá-los (InfluxDB).

Cenários de surto, endemia, epidemia ou pandemia são abordados a seguir. A declaração da epidemia de COVID-19 em março de 2020 trouxe às claras muitas das deficiências dos serviços de saúde globais, além da necessidade de se repensar os fluxos de trabalho. Em paralelo, a demanda por dados para acompanhamento, para controle e para rastreamento de pessoas e locais motivou a pesquisa de técnicas e a avaliação de ferramentas que auxiliassem no seu atendimento. É apresentado o Neo4j como opção de SGBD orientado a grafos para atender a esse tipo de rastreamento.

Finalmente, a gerência de grandes volumes de dados heterogêneos é abordada sob a ótica do modelo de dados orientado a colunas. Surgida como a evolução de uma proposta de armazenamento decomposto baseada no modelo relacional, a orientação a colunas provê a flexibilidade de esquema necessária e a liberdade para se armazenarem dados cuja composição, até o momento da sua disponibilização, é indefinida. O HBase é apresentado como o SGBD para, no cenário da rede de hospitais, ser o destino de imagens médicas e outros conteúdos semiestruturados e não estruturados.

A adoção de diferentes ferramentas para suporte a diferentes modelos de dados é prática comum, indicando que não existe uma solução única para todos os problemas presentes em cenários complexos.

Prontos para essa imersão em NoSQL e Big Data? Não deixe de acompanhar os experimentos e, se possível, repeti-los para uma maior fixação do conteúdo!

Ao final dos seus estudos, recapitule os conteúdos apresentados e considere a utilização deles como auxílio à solução de problemas de *Big Data* do seu cotidiano.

RECURSOS DE IMERSÃO



REALIDADE AUMENTADA

Sempre que encontrar esse ícone, esteja conectado à internet e inicie o aplicativo Unicesumar Experience. Aproxime seu dispositivo móvel da página indicada e veja os recursos em Realidade Aumentada. Explore as ferramentas do App para saber das possibilidades de interação de cada objeto.



RODA DE CONVERSA

Professores especialistas e convidados, ampliando as discussões sobre os temas.



PÍLULA DE APRENDIZAGEM

Uma dose extra de conhecimento é sempre bem-vinda. Posicionando seu leitor de QRCode sobre o código, você terá acesso aos vídeos que complementam o assunto discutido.



PENSANDO JUNTOS

Ao longo do livro, você será convidado(a) a refletir, questionar e transformar. Aproveite este momento.



EXPLORANDO IDEIAS

Com este elemento, você terá a oportunidade de explorar termos e palavras-chave do assunto discutido, de forma mais objetiva.



NOVAS DESCOBERTAS

Enquanto estuda, você pode acessar conteúdos online que ampliam a discussão sobre os assuntos de maneira interativa usando a tecnologia a seu favor.



OLHAR CONCEITUAL

Neste elemento, você encontrará diversas informações que serão apresentadas na forma de infográficos, esquemas e fluxogramas os quais te ajudarão no entendimento do conteúdo de forma rápida e clara

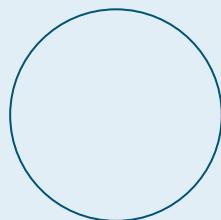
Quando identificar o ícone de **QR-CODE**, utilize o aplicativo **Unicesumar Experience** para ter acesso aos conteúdos on-line. O download do aplicativo está disponível nas plataformas:



Google Play



App Store



CAMINHOS DE APRENDIZAGEM

1

9

BIG DATA: VOLUME,
VELOCIDADE,
VARIEDADE, VALOR
E VERACIDADE

2

45

BANCOS DE DADOS
PARA DOCUMENTOS
ESTRUTURADOS E
SEMIESTRUTURADOS

3

77

BANCOS DE
DADOS PARA
SÉRIES TEMPORAIS
ORIENTADOS A
CHAVE-VALOR

4

107

CONEXÕES E
RELACIONAMENTOS
EM BANCOS DE
DADOS GRAFO

5

137

FLEXIBILIDADE DE
ARMAZENAMENTO
EM BANCO
DE DADOS
ORIENTADO A
COLUNAS

Big Data: Volume, Velocidade, Variedade, Valor e Veracidade

Dr. Alexandre Savaris

OPORTUNIDADES DE APRENDIZAGEM

Nesta unidade, serão apresentadas as características que melhor definem projetos com grandes volumes de dados (os chamados 'Vs' de Big Data). Serão relembrados os modelos computacionais centralizados e distribuídos (mainframes e minicomputadores, servidores e computadores pessoais), e a consequente adoção de sistemas monolíticos ou sistemas distribuídos. Especificamente para sistemas distribuídos, serão descritas as principais organizações dos seus componentes com relação a camadas e responsabilidades. Finalmente, o teorema CAP é descrito e utilizado como um classificador de Sistemas Gerenciadores de Bancos de Dados NoSQL.



Em dezembro de 2019, a OMS/WHO (Organização Mundial da Saúde/World Health Organization) foi notificada sobre casos de pneumonia na cidade de Wuhan, província de Hubei, na República Popular da China. Autoridades chinesas confirmaram, dias depois, tratar-se de um novo tipo de coronavírus. Essa confirmação levou à declaração de uma **Emergência de Saúde Pública de Importância Internacional (ESPII)** em 30 de janeiro de 2020, culminando com a caracterização de uma pandemia em 11 de março de 2020 – a **pandemia de COVID-19**.

Nos meses seguintes, os sistemas de saúde públicos e privados adotados por diferentes países ao redor do mundo passaram a receber a atenção da mídia e do público de forma generalizada, dada a sua importância na identificação dos casos positivos (por meio da testagem em larga escala), no acompanhamento de pacientes (por meio de visitas domiciliares ou internações) e na imunização (através da aplicação de vacinas).



Toda essa atenção trouxe à tona os inúmeros problemas que o setor de saúde enfrenta, incluindo (mas não se limitando) as questões organizacionais (como a falta de profissionais de saúde), as questões operacionais (como problemas de logística que atrasam ou impedem a chegada de insumos necessários às unidades que os necessitam), as questões estruturais (como o sucateamento e a falta de equipamentos, tanto de suporte à vida quanto voltados à administração de postos de saúde, clínicas e hospitais), e as questões financeiras, como a falta de recursos para a aquisição dos insumos básicos necessários ao funcionamento dos estabelecimentos e para a remuneração dos profissionais que atuam na linha de frente.

Por outro lado, surgiram oportunidades de avançar em discussões que até então haviam recebido pouca atenção, ou cuja tramitação estava travada pela burocracia, por exemplo, a **autorização em caráter emergencial do uso da Telemedicina no Brasil, sancionada pela Lei nº 13.989 de 15 de abril de 2020**.

Neste caso em específico, o uso de Tecnologias da Informação e Comunicação (TICs) na solução de problemas da saúde pública e privada passou a ser visto como uma necessidade, perfeitamente aplicável no contexto de atendimentos remotos visando a manutenção do distanciamento social.

Essas oportunidades não passaram despercebidas para Luísa e sua equipe. Luísa, formada em Ciência da Computação e com pós-graduação/MBA (Master in Business Administration) focado em gestão e inovação em serviços de saúde, atua como CTO (Chief Technology Officer) em uma rede de hospitais privados estabelecida a décadas.

Apesar de atuarem essencialmente no atendimento privado, os hospitais da rede fazem parceria com o SUS (Sistema Único de Saúde) do Brasil, disponibilizando um percentual de leitos e de recursos para receber pacientes oriundos da esfera pública. Compõem a rede hospitais de diferentes portes e com oferta de atendimento para diferentes especialidades, o que torna o conjunto bastante complexo e heterogêneo.

Essa complexidade e heterogeneidade pode ser observada também no nível da adoção de TICs, sendo que há hospitais com boa parte de seus processos informatizados; em contrapartida, há integrantes da rede em que o receituário ainda é impresso (passando de mão em mão até que os medicamentos relacionados sejam dispensados) e que os laudos de exames ainda são manuscritos pelos médicos responsáveis pelo atendimento aos pacientes.

O objetivo de Luísa é implantar, em toda a rede de hospitais, o conceito de **Hospital sem Papel** (Paperless Hospital). É um conceito que existe há muitos anos e cuja aplicação visa à sustentabilidade por meio da eliminação do papel (ou de qualquer outra mídia impressa) do fluxo de trabalho da instituição (OLIVEIRA; PASSOS, 2020).

Um hospital sem papel otimiza seus processos através da adoção de TICs, obtendo resultados positivos no tocante ao tempo (com a redução de atrasos na execução de procedimentos e, consequentemente, com a redução da permanência do paciente), no tocante a custos (com a eliminação da necessidade de impressão de todo e qualquer tipo de documento, incluindo filmes com as imagens de exames como raio-x e tomografia computadorizada, por exemplo) e no tocante à segurança (com o uso de boas práticas para o credenciamento e o acesso aos sistemas de informação sendo liberado apenas a usuários devidamente reconhecidos e previamente cadastrados) (SALOMI; MACIEL, 2015).

Como metodologia de trabalho, Luísa irá orientar sua equipe com base no conhecimento do histórico e da aplicação de conceitos reconhecidos e bem estabelecidos na área de TI para a saúde, limitando-os ao escopo de hospitais. Um primeiro passo para fundamentar o trabalho a ser realizado é conhecer os diferentes sistemas de informação que os hospitais utilizam, e aqui a sua ajuda é fundamental.

Faça uma pesquisa na internet, procurando entender o objetivo e identificar quais são os dados gerenciados por cada um dos cinco sistemas de uso mais comum por esses estabelecimentos, a saber: **Sistema de Informação Hospitalar** (Hospital Information System - HIS), **Sistema de Informação Radiológica** (Radiology Information System - RIS), **Sistema de Comunicação e Armazenamento de Imagens Médicas** (Picture Archiving and Communication System - PACS), **Sistema de Informação Clínica** (Clinical Information System - CIS) e **Sistema de Informação Laboratorial** (Laboratory Information System - LIS).

A implantação de um sistema de informação é uma tarefa árdua, que muitas vezes implica na reorganização dos processos executados pelas instituições que o adotam para que sejam obtidos os melhores resultados. Implementar cinco sistemas de forma orquestrada (gerenciada e coordenada de forma automatizada), então, é uma tarefa que demanda um controle rigoroso das ações a serem realizadas – além de uma definição de limites bastante clara para que o trabalho tenha um fim. Assim, é essencial que a equipe responsável tenha o conhecimento necessário sobre a **arquitetura** a ser adotada (se centralizada ou distribuída), sobre os impactos esperados a partir da adoção de cada uma dessas alternativas e das decisões de projeto relacionadas ao uso dos sistemas (como se tratará a questão da **consistência dos dados** relacionada à **disponibilidade dos sistemas**), e também sobre a forma como esses dados serão armazenados para uso individualizado ou consolidado (**sistemas de arquivo, bancos de dados relacionais, bancos de dados NoSQL**).

Em seu Diário de Bordo, tome nota de suas impressões a partir da pesquisa feita sobre os tipos de sistemas de informação utilizados em hospitais e as complemente, aos poucos, com as novas definições que serão apresentadas a seguir. É possível que suas anotações sejam úteis para que Luísa possa organizar o trabalho de sua equipe da melhor forma possível!

DIÁRIO DE BORDO

Para o pontapé inicial (*kick-off*) de projeto do Hospital sem Papel, Luísa convocou sua equipe para uma reunião pautada em um levantamento do histórico e de conceitos relacionados a tecnologias que poderiam ser aproveitados na fundamentação das atividades a serem desenvolvidas. Nesta reunião, Gaspar, o gerente de projetos que integra a equipe, ponderou se o projeto não estaria propondo a solução para um problema de **Big Data**. Se esse fosse o caso, soluções já conhecidas para esse tipo de problema poderiam ser avaliadas, selecionadas e adaptadas ao contexto dos hospitais da rede.

Há controvérsias sobre a origem exata do termo Big Data e sobre quem foi o responsável por utilizá-lo oficialmente pela primeira vez (e em que contexto) (DONTHA, 2018). A ideia que fundamenta o termo, porém, vem de décadas passadas. Edgar Frank Codd, por exemplo, em seu artigo sobre o modelo de dados relacional,

já abordava a organização de grandes *datasets* em idos de 1970 – organização essa que se mantém como uma das principais características de problemas resolvidos por essa abordagem (CODD, 1970). Junto a esta característica o fato de que os dados citados possuem diferentes formatos, são originados a partir de diferentes fontes e, por serem gerados por um número cada vez maior de produtores de conteúdo, ficam disponíveis de forma cada vez mais rápida (BASSO, 2020).

Em 2001, Doug Laney caracterizou cenários de Big Data como uma combinação de **Volume, Velocidade e Variedade** (**Volume, Velocity, Variety**, os três ‘Vs’ clássicos que foram naturalmente incorporados à própria definição do termo) (LANEY, 2012) (vide Figura 1). À época, os canais de *e-commerce* foram citados como catalisadores (dinamizadores ou estimuladores) desses cenários, responsáveis pelo aumento no volume dos dados trafegados e armazenados.

Além disso, a disseminação do uso desses mesmos canais contribuiu com um aumento na velocidade em que os dados das transações executadas fossem gerados, além de permitir a adoção de diferentes tipos de dados que resultaram em um aumento de variedade para os *datasets*. De lá para cá, a caracterização de problemas com relação à Big Data é possível para os mais diversos segmentos, porém, com as mesmas definições para os três ‘Vs’ citados anteriormente.



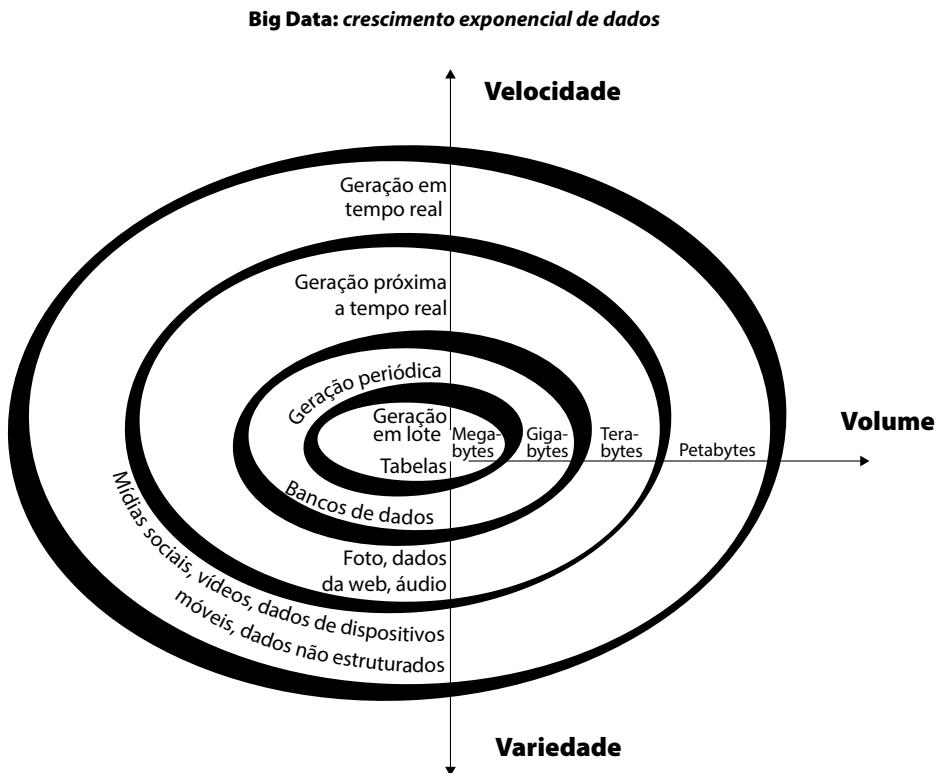


Figura 1 - Relação entre os 3 'Vs' do Big Data / Fonte: Wikimedia Commons (2018, on-line).

Descrição da Imagem: no topo da figura há o título "Big Data: crescimento exponencial de dados", na figura são apresentados os 3 'Vs' de Big Data, representados por setas que partem de um ponto central, de quatro círculos concêntricos. A seta correspondente à velocidade aponta para cima, e é acompanhada de alguns cenários (do mais lento para o mais rápido: 1. geração em lote; 2. geração periódica; 3. geração próxima a tempo real; 4. geração em tempo real). A seta correspondente ao volume aponta da esquerda para a direita, e é acompanhada de algumas unidades de medida de armazenamento (do menor para o maior: 1. megabytes; 2. gigabytes; 3. terabytes; 4. petabytes). A seta correspondente à variedade, por sua vez, aponta para baixo, e é acompanhada de algumas aplicações que geram ou consomem dados (1. tabelas; 2. bancos de dados; 3. fotos, áudio, dados da web; 4. mídias sociais, vídeos, dados de dispositivos móveis, dados não estruturados).

- **Volume (Volume)** - resultado da aquisição de grandes *datasets* provenientes de processos explícitos (como o preenchimento de um formulário por um usuário para uma compra on-line) ou implícitos (como o registro do número de cliques dados por esse mesmo usuário durante o processo de compra, ou a aquisição dos dados de localização do usuário enquanto compra os produtos). Dada a diversidade de opções e o custo reduzido

para armazenamento desses dados por tempo indeterminado, passou a fazer sentido adquirir, manter e executar análises descritivas (para revelar o que aconteceu através do acesso a dados conhecidos), preditivas (para sugerir o que irá acontecer com base em um histórico conhecido) ou prescritivas (para sugerir o que fazer com base em um histórico conhecido, oferecendo uma solução para um problema delimitado), objetivando responder às seguintes perguntas: O que aconteceu? Por que aconteceu? O que irá acontecer? Como pode se fazer acontecer?

Com relação ao entendimento do que é um ***dataset volumoso***, é necessário considerar o contexto de aplicação. Um exemplo pode ser dado a partir da Tabela 1, na qual estão relacionadas diferentes modalidades de exame de imagem, o tamanho aproximado das imagens geradas para cada modalidade e o tamanho aproximado dos arquivos onde essas imagens são gravadas. Se for considerada apenas a última coluna (“Tamanho aproximado do arquivo (por imagem)”), exames de “Radiografia Digital”, “Mamografia Digital” e “Radiografia Computadorizada” podem ser enquadrados como componentes de ***grandes datasets***. Porém, se considerarmos o número de imagens adquiridas por exame, teremos as modalidades de “Ressonância Magnética” e “Tomografia Computadorizada” como geradoras de ***datasets de tamanho considerável***. Isso acontece porque para cada exame destas duas modalidades são adquiridas dezenas, centenas ou mesmo milhares de imagens; exames de “Radiografia Digital”, “Mamografia Digital” e “Radiografia Computadorizada”, por sua vez, são executados com a aquisição de apenas algumas imagens. Ao final, *datasets* maiores são formados para exames com imagens de menor tamanho devido a maior quantidade de imagens envolvidas.

Modalidade	Matriz de imagem (em pixels)	Tamanho aproximado do arquivo (por imagem)
Ressonância Magnética	256 x 256	131KB
Tomografia Computadorizada	512 x 512	524KB
Ultrassom	512 x 512	262KB

Modalidade	Matriz de imagem (em pixels)	Tamanho aproximado do arquivo (por imagem)
Radiografia Digital	Até 3.000 x 3.000	Até 18MB
Mamografia Digital	Até 3.328 x 4.096	27MB
Radiografia Computadorizada	3.520 x 4.280	30MB

Tabela 1 - Tamanho e espaço de armazenamento necessário para imagens de diferentes modalidades de exame / Fonte: adaptado de Dandu (2008).

- **Velocidade (Velocity)** - corresponde à quantificação da frequência com que os volumes de dados são gerados. Essa quantificação resulta em números que podem ser utilizados para avaliar quantos dados são gerados em um intervalo de tempo delimitado, permitindo a sua comparação com outros métodos de aquisição e a adequação do método utilizado às necessidades do problema que está sendo resolvido. É comum representar essa característica por meio de comparações (vide Tabela 2), de forma a simplificar o entendimento e exemplificar o impacto dessa frequência com base em atividades do cotidiano.

Provedor/serviço	Dado/evento	Volume
Google	Buscas	5.700.000
Discord	Envio de mensagens	668.000
iMessage	Envio de mensagens	12.000.000
Clubhouse	Criação de salas	208
Snapchat	Envio de mensagens	2.000.000
Amazon	Compras (em US\$)	283.000,00
E-commerce	Pessoas comprando algum produto	6.000.000
Strava	Compartilhamento de atividades esportivas	1.500

Provedor/serviço	Dado/evento	Volume
Instacart	Gastos pelos usuários (em US\$)	67.000,00
Venmo	Envios pelos usuários (em US\$)	304.000,00
Slack	Envio de mensagens	148.000
Zoom	Transmissão de webinars (em minutos)	856
Teams	Usuários conectados	100.000
Netflix	Streaming (em horas)	452.000
Youtube	Transmissão de vídeo (em horas)	694.000
Facebook	Visualizações	44.000.000
Facebook	Fotos compartilhadas	240.000
Instagram	Fotos compartilhadas	65.000
Twitter	Envio de tweets	575.000
TikTok	Exibição de vídeos	167.000.000

Tabela 2 - Volume de dados gerados em alguns dos principais sites e serviços da internet no intervalo de um minuto / Fonte: o autor.

O aumento na velocidade da geração de dados tem demandado que o processo de análise se adapte às taxas de aquisição, transformando o já conhecido processo de *analytics* em *real-time analytics*. A análise em tempo real dos *datasets* que se tornam disponíveis objetiva reduzir o tempo da tomada de decisão por meio do cálculo de **indicadores-chave de performance (KPIs - Key Performance Indicators)**, indicadores esses que podem variar seus valores rapidamente de acordo com os dados recebidos. Em contextos nos quais decisões precisam ser tomadas imediatamente com base em dados dos últimos dias, horas ou minutos (estejam eles completos ou não), a velocidade com que se coleta e analisa conteúdo é fator determinante para o sucesso do empreendimento (DYKES, 2017).

- **Variedade (Variety)** - considerando grandes volumes de dados sendo adquiridos em velocidades cada vez maiores a partir de uma diversidade de origens e com diferentes finalidades, é natural assumir que esses dados sejam

disponibilizados em diferentes formatos. A multiplicidade de formatos implica em um esforço de quem os recebe e utiliza para entender a sua organização original, adaptar seu formato de origem para um formato compatível ou esperado (se esse for o caso) e, finalmente, utilizá-lo para processamento visando descrever, prever ou prescrever com base no seu conteúdo.

Apesar de teoricamente infinita em termos de possibilidades de organização, dados cuja estrutura pode variar acabam sendo classificados em três grupos (descritos a seguir e comparados no Quadro 1) (MONICA; KUMAR, 2013):

- 1. Dados estruturados** - seguem uma organização rígida definida por um conjunto de regras. A existência dessa organização permite que o conteúdo estruturado seja facilmente indexado para facilitar a execução de consultas, que possa ser processado de forma eficiente e sem ambiguidades, e possa ser entendido tanto por seres humanos quanto por algoritmos desenvolvidos com finalidades específicas. Um exemplo é a estrutura de tabela de um banco de dados relacional, que organiza o conteúdo de interesse em linhas (registros), colunas (campos) e aplica restrições sobre o conteúdo que pode ser armazenado em cada campo (tipo de dado).
- 2. Dados não estruturados** - são diametralmente opostos aos dados estruturados em termos de organização, não apresentando qualquer estrutura ou seguindo qualquer regra. Essas características fazem com que o processamento do conteúdo seja dificultado, o que pode gerar ambiguidades e impedir a obtenção de soluções aceitáveis para os problemas relacionados a esses dados. Apesar da dificuldade no seu tratamento, o fato de este grupo ser o mais volumoso quando comparado aos demais justifica o esforço e o direcionamento de recursos para a pesquisa e a obtenção de soluções para o seu uso efetivo. São incluídos neste grupo os dados multimídia como imagens, vídeos e sons, além de textos livres escritos de acordo com normas cultas ou mesmo de forma coloquial (informal).
- 3. Dados semiestruturados** - posicionam-se como um meio-termo entre os dados estruturados e os dados não estruturados, permitindo a adoção

de uma estrutura menos rígida combinada com a flexibilidade dos dados textuais e multimídia. Na prática adotam uma organização onde, para determinados níveis ou elementos, são permitidos conteúdos sem estrutura. Exemplos incluem dados em XML (*eXtensible Markup Language*) e JSON (*JavaScript Object Notation*), em que imagens, vídeos, sons e texto podem ser armazenados como conteúdo de tags ou chaves, respeitando uma estrutura prévia que costuma seguir um esquema predefinido.

	Dados estruturados	Dados semiestruturados	Dados não estruturados
Tecnologia	Tabela de banco de dados relacional	Arquivos em XML/RDF	Dados binários e do tipo caractere (textuais)
Gerenciamento de transações	Consolidado, adotando diferentes técnicas de concorrência	Gerenciamento de transações adaptado de bancos de dados relacionais	Sem concorrência ou gerenciamento de transações
Versionamento	Feito sobre tabelas e registros	Feito sobre tuplas e grafos	No conteúdo como um todo
Flexibilidade	Dependente de esquema rígido	Presença de esquema flexível	Ausência de esquema
Escalabilidade	Difícil devido à presença de esquema rígido	Simplificada devido à flexibilidade de esquema	Altamente escalável
Robustez	Altamente robusto	Em evolução	-
Desempenho em consultas	Consultas estruturadas permitindo junções complexas	Possibilita consultas em nodos anônimos	Permite apenas consultas textuais

Quadro 1 - Comparação entre dados estruturados, semiestruturados e não estruturados
Fonte: adaptado de Monica e Kumar (2013).



Com o passar do tempo e a disseminação de projetos de Big Data nas mais diferentes áreas, os três ‘Vs’ originais foram complementados por outras características que variam em número e significado (THIYAGARAJAN; VENKATA-CHALAPATHY, 2014; ISHWARAPPA; ANURADHA, 2015; HADI et al., 2015; OWAIS; HUSSEIN, 2016):

- **Valor (Value)** - sob a ótica de negócio, caracteriza os dados coletados com base na sua contribuição para a tomada de decisões. Quanto melhores as previsões ou prescrições feitas a partir do seu uso, maior será o retorno (lucro) realizado. Comumente, o valor está implícito nos *datasets* disponíveis, o que demanda a pesquisa de metodologias e o desenvolvimento de técnicas aplicáveis à sua extração.
- **Veracidade (Veracity)** - nem todo *dataset* é confiável; assim, a confirmação da veracidade do seu conteúdo é necessária para evitar inconsistências, ambiguidades ou incompletudes. Dados cuja procedência não seja verificável ou que possuam baixa precisão contribuem para a redução no valor obtido a partir do seu uso.
- **Viscosidade (Viscosity)** - dados provenientes de múltiplas fontes com múltiplos formatos podem dificultar o processo de integração e adequação de seu conteúdo ao fluxo de processamento, atrasando a obtenção do valor esperado.

- **Variabilidade (Variability)** - o fluxo de disponibilização de dados pode apresentar inconsistências em termos de velocidade ou de volume, quando relacionado ao tempo. Implica na ocorrência de picos na aquisição motivados por uma sazonalidade conhecida ou mesmo eventos inesperados. De forma complementar, o contexto de uso dos dados coletados pode ser diverso, o que permite variar a interpretação dada a um mesmo conteúdo.
- **Volatilidade (Volatility)** - o período de validade de um dado para análise, bem como o tempo de manutenção (armazenamento) desse dado, mudam conforme a área de interesse. Enquanto a análise em tempo quase-real é relevante para a tomada de decisões rápidas, a análise histórica de dados adquiridos em anos anteriores é relevante para estudos descritivos (que observam esse histórico visando entender ocorrências passadas).
- **Viabilidade (Viability)** - *datasets*, comumente, são compostos por dezenas ou mesmo centenas de atributos (valores) que, quando analisados em conjunto, levam ao sucesso (ou fracasso) na tomada de decisões. A viabilidade corresponde à seleção e manutenção entre todos esses atributos daqueles que são efetivos na geração de resultados positivos para o negócio. É importante também como forma de redução no consumo de recursos e no tempo de processamento, visto que a seleção dos atributos relevantes leva à redução no volume desses *datasets*.
- **Validade (Validity)** - dados completos e confiáveis podem não ser válidos para atender a um determinado caso de uso. Datasets são considerados válidos quando aplicados a um propósito conhecido, sendo compostos por atributos úteis à solução do problema. Aqui, o conhecimento do negócio a ser atendido é essencial para que a seleção dos dados seja acertada.
- **Viralidade (Virality)** - mensura a capacidade dos dados serem compartilhados, atingindo a maior audiência possível no menor intervalo de tempo possível. É relevante em cenários em que a divulgação de conteúdo é necessária à disseminação de algum conhecimento, ou mesmo à sinalização da existência de conteúdo para consumo (aluguel, compra).

Com o levantamento e a exposição das características, ficou claro para Luísa e a sua equipe que o projeto do Hospital sem Papel pode ser classificado como um projeto de Big Data. Nele, o **volume** de dados é obtido a partir do agendamento, execução e avaliação de exames, bem como do acompanhamento e monitoração dos pacientes atendidos. A **velocidade** de aquisição é observada no uso de equipamentos capazes de monitorar pacientes em tempo real ou quase real, gerando indicadores para suas condições físicas uma ou mais vezes por segundo.

Pelo fato de serem adotados diferentes padrões tanto na aquisição, quanto no processamento e no armazenamento dos dados coletados (como, por exemplo, DICOM® - *Digital Imaging and Communications in Medicine* – para imagens médicas digitais, e HL7® FHIR® - *Health Level Seven Fast Healthcare Interoperability Resources* – para a integração entre sistemas de saúde), a **variedade** dos datasets não pode ser ignorada.

O **valor** obtido a partir dos dados gerenciados pode ser observado em tratamentos melhores e mais ágeis aos pacientes, com base na experiência de anos anteriores, e será tão efetivo quanto a **veracidade** do que foi registrado a partir dessa experiência. Apesar de difícil, a integração entre os diferentes sistemas que serão utilizados pelos hospitais (**viscosidade**) é uma barreira a ser superada para que os resultados sejam atingidos, o que é potencializado pela sazonalidade de alguns eventos (**variabilidade**) já conhecidos (por exemplo, surtos de gripe e doenças correlatas no inverno).

A **volatilidade** dos dados manipulados varia de acordo com a sua natureza, sendo regulada por legislações específicas. Por fim, os dados adquiridos são naturalmente complexos, o que demanda simplificações para que se tornem **viáveis** e **válidos** para os casos de uso aos quais devem ser direcionados. A **viralidade** dos dados, porém, é questionável: por se tratar de dados pessoais referentes à saúde de indivíduos, o compartilhamento é limitado – inclusive por lei – e não deve ser considerada uma prática válida no contexto do problema que está sendo resolvido.



NOVAS DESCOBERTAS

DICOM® (*Digital Imaging and Communications in Medicine*) teve sua primeira versão publicada em 1993 e desde então se consolidou como o padrão internacional dominante para estruturação, transmissão e arquivamento de imagens médicas digitais e outros dados relacionados. Escaneie o Qr Code, a seguir, para conhecer o site.



NOVAS DESCOBERTAS

HL7® FHIR® (*Health Level Seven Fast Healthcare Interoperability Resources*) - foi publicado inicialmente em 2014 como evolução das versões 2, 3 e CDA (Clinical Document Architecture) e vem se firmando como uma das alternativas para interoperabilidade entre sistemas de saúde. Escaneie o Qr Code, a seguir, para conhecer o site.



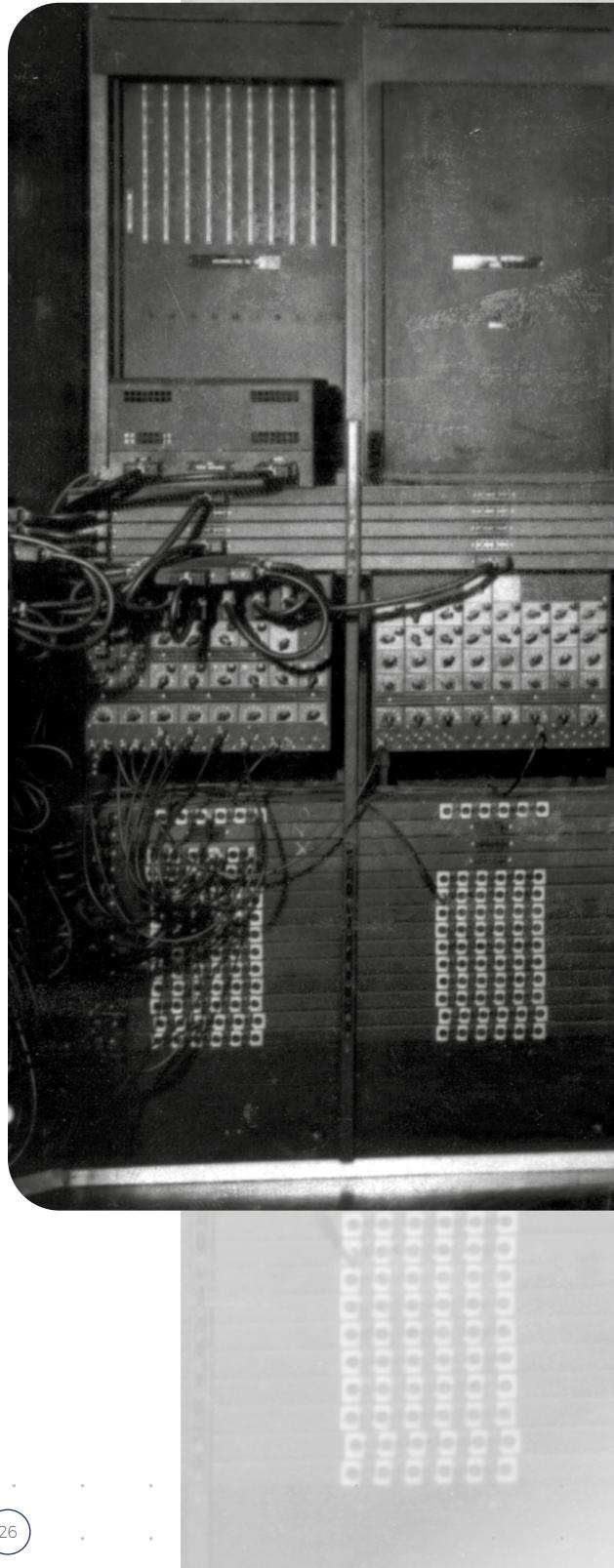
PENSANDO JUNTOS

Tanto o padrão DICOM® quanto o padrão HL7® FHIR® são considerados padrões abertos, para os quais existem guidelines de uso, mas não uma regulação ou fiscalização de uso correta para os produtos desenvolvidos que os implementam. Por esse motivo, podem ocorrer incompatibilidades e problemas de interoperabilidade entre esses produtos.

Concluída a caracterização do problema a ser resolvido pelo projeto de Hospital sem Papel, o foco da discussão da equipe de trabalho voltou-se a questões de arquitetura. Considerando que se trata de uma rede descentralizada de hospitais, foram feitos questionamentos sobre a melhor organização em alto nível a ser adotada durante o desenvolvimento do trabalho. Laura, a arquiteta de soluções recrutada por Luísa para integrar a equipe, resumiu a discussão a uma questão de escolha entre uma proposta **centralizada** (baseada em um modelo computacional monolítico) e uma proposta **distribuída** na forma de uma cloud (nuvem) privada projetada para uso dos hospitais da rede. Ambas as propostas têm base histórica que remete aos mainframes e culmina com a adoção massiva de sistemas distribuídos.

Nas décadas de 1950 e 1960, o cenário da computação comercial era caracterizado por um modelo de negócio no qual a aquisição de hardware e software e a contratação de serviços de uso e manutenção era feita com um mesmo fornecedor, dada a complexidade de integração entre esses componentes. Foi a era dos **mainframes**, computadores de grande porte desenvolvidos durante os anos anteriores por universidades (como o ENIAC - *Electronic Numerical Integrator and Computer* - na Figura 2) e mais tarde integrados ao cenário corporativo por empresas como a IBM (*International Business Machines Corporation*) e Bull (vide Figura 3).

O processamento e armazenamento disponibilizado por esses equipamentos eram centralizados, suas rotinas (programas a serem executados) organizadas em lotes (*batch*) e o seu acesso remoto disponibilizado via terminais utilizados como meros pontos de conexão à central de processamento (TIGRE; NORONHA, 2013).



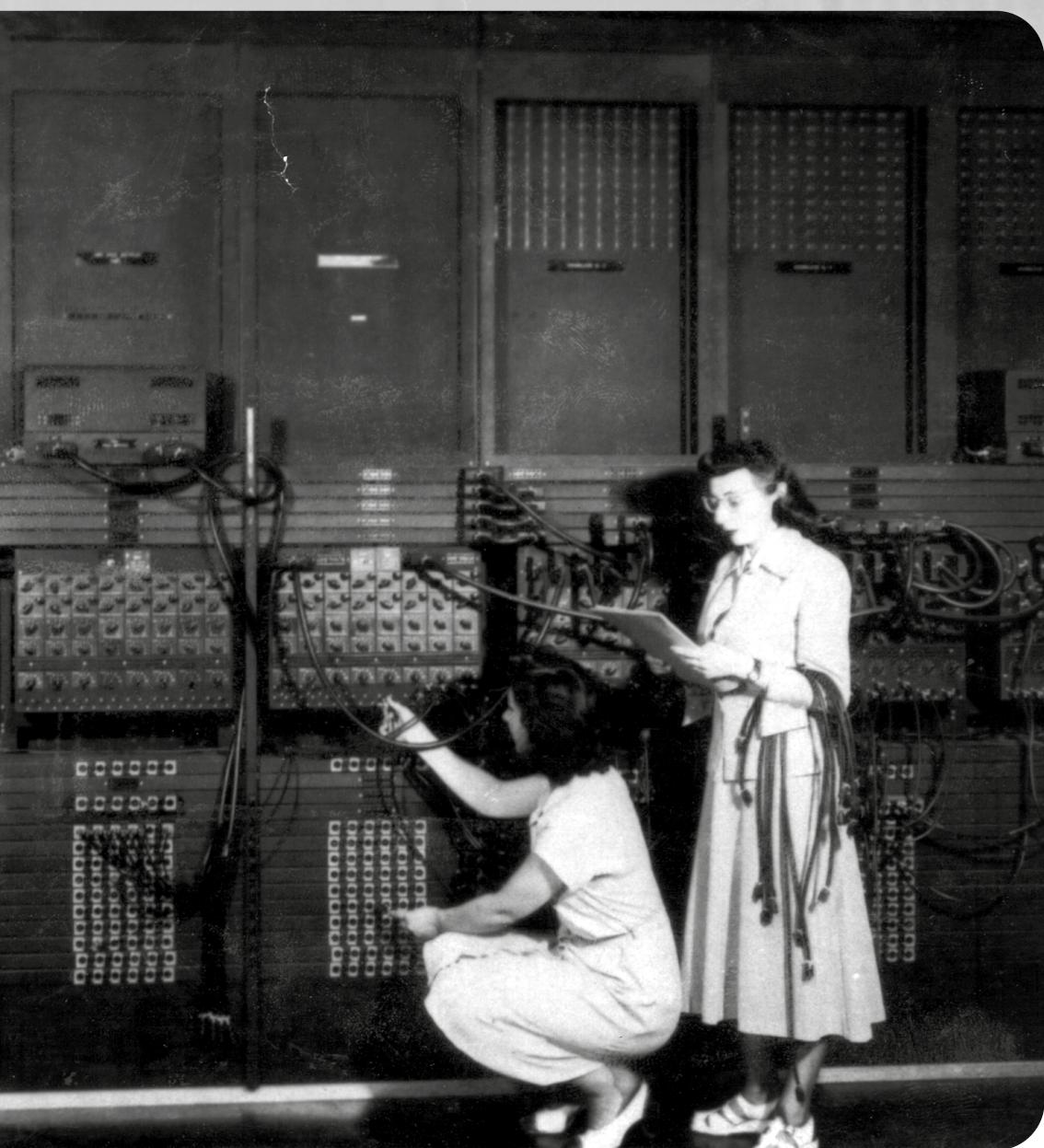


Figura 2 - O ENIAC (Electronic Numerical Integrator and Computer) em meados de 1946
Fonte: Wikimedia Commons ([1946], on-line).

Descrição da Imagem: a foto exibe o ENIAC, que ocupava o espaço de uma sala inteira, com destaque a alguns de seus painéis utilizados para a inserção de novos programas. À direita há duas mulheres, uma delas conecta cabos nesses painéis - tarefa essa equivalente à programação de um computador pessoal feita nos dias de hoje; e outra está ao lado com uma prancheta nas mãos.



Figura 3 - O IBM System/360, mainframe de grande sucesso comercial até o início da década de 1980
Fonte: Wikimedia Commons (2005, on-line).

Descrição da Imagem: a foto exibe os componentes do IBM System/360. À esquerda, em segundo plano, estão localizadas as unidades de armazenamento de dados - equivalentes aos HDs (Hard Disks) da atualidade; em primeiro plano, da esquerda para a direita, estão localizadas as unidades de saída (impressororas), a unidade de processamento central - equivalente à CPU dos computadores pessoais - e os terminais para acesso ao sistema e a execução dos programas. O conjunto completo - exposto em um museu - ocupa o espaço de uma pequena sala.



NOVAS DESCOBERTAS

A IBM (*International Business Machines Corporation*) foi fundada em 1911 e dominou o mercado de computadores de grande porte (*mainframes*) nas décadas que precederam o processo de *downsizing* adotado por instituições em diferentes segmentos de mercado. Escaneie o Qr Code, a seguir, para conhecer momentos icônicos na história da IBM.

**EXPLORANDO IDEIAS**

A Bull é empresa francesa fundada em 1931 e adquirida pela Atos em 2014, dividiu com a IBM e outras o mercado de mainframes, diversificando seu catálogo de produtos e serviços com oferta de supercomputadores, servidores, sistemas de backup, sistemas de armazenamento e mobile.

Nas décadas de 1970 e 1980, a popularização dos microprocessadores contribuiu para o **downsizing** das instituições, processo que envolveu a reorganização de suas estruturas visando enxugá-las em termos de tamanho e custo. Nessa época, os *mainframes* passaram a ser substituídos por **servidores departamentais** (inicialmente baseados em **minicomputadores** (vide Figura 4), caracterizados por custar uma fração do valor dos primeiros e ocuparem um espaço consideravelmente menor.

Apesar de o software continuar proprietário, as interfaces de comunicação foram melhoradas de forma que periféricos de terceiros (monitores, impressoras, dentre outros) pudessem ser conectados e utilizados. Essa etapa foi seguida pela adoção dos **computadores pessoais (Personal Computers, os PCs)**, os quais assumiram as tarefas de servidor, terminal de conexão e estação de trabalho.

Por reduzirem ainda mais os custos e o espaço físico necessários à sua instalação, os PCs passaram para a categoria de bens de consumo (commodities) disponíveis tanto a pessoas físicas quanto a corporações, podendo ser montados combinando periféricos de diferentes fabricantes e executando softwares não proprietários desenvolvidos por empresas especializadas (TIGRE; NORONHA, 2013).





Figura 4 - O PDP-11 da DEC (Digital Equipment Corporation) / Fonte: Wikimedia Commons (2009, on-line).

Descrição da Imagem: a foto exibe os componentes de um minicomputador DEC PDP-11. Nas extremidades esquerda e direita do equipamento estão localizados leitores de fitas magnéticas, fitas essas armazenadas em rolos; ao centro estão localizados os periféricos para a entrada, processamento e saída de dados. O conjunto completo corresponde ao tamanho de um armário ou roupeiro médio. No limite direito da foto aparece um PC, posicionado sobre uma mesa.



PENSANDO JUNTOS

O termo *downsizing* é comumente associado a cortes de pessoal e diminuição de custos, o que o coloca como uma estratégia ameaçadora aos colaboradores de uma instituição. Surgiu então um termo alternativo – *rightsizing* – com foco em encontrar o tamanho certo da instituição, procurando manter a motivação das equipes em alta.

A substituição dos *mainframes* por um conjunto de minicomputadores e PCs alterou drasticamente o modelo de negócio relacionado à aquisição de hardware, software e serviços. Antes, um único fornecedor respondia por todo o ecossistema; após o *downsizing*, diferentes fornecedores (cada qual especializado em parte

da solução) tiveram seus produtos e serviços combinados para atender às demandas impostas pela redução de estrutura e orçamento (TIGRE; NORONHA, 2013).

Em paralelo, o modelo de processamento centralizado e em lotes (*batch*) foi substituído por um modelo de processamento descentralizado, distribuído; essa substituição, que em termos de sistemas de informação não é trivial, impôs demandas próprias relacionadas à melhoria da infraestrutura de redes de computadores, seja para dar suporte a conexões locais (LANs - *Local Area Networks*), em área metropolitana (MANs - *Metropolitan Area Networks*) ou mesmo estendidas a um país/continente (WANs - *Wide Area Networks*). Utilizando infraestrutura própria ou a própria internet como meio, a conectividade nesses ambientes descentralizados é fator sem o qual os sistemas não funcionam, tornando-a uma pré-condição.

Os sistemas construídos com base nos princípios da descentralização de recursos são conhecidos como **sistemas distribuídos**. Esses sistemas são uma combinação de computadores independentes e componentes de software organizados e conectados de forma que, para o usuário final, aparentam ser constituídos por um único computador e sistema (TANENBAUM; STEEN, 2007).

Para que sejam efetivos, sistemas construídos com base nessa arquitetura devem oferecer um acesso facilitado aos seus recursos de forma compartilhada (onde usuários dividem recursos comuns com base em políticas e permissões de acesso), transparência relacionada à distribuição desses recursos entre os componentes do sistema (de forma que os usuários possam utilizá-los independentemente da sua localização, seja em um computador próximo ou localizado em outro departamento, prédio ou cidade), abertura com relação à comunicação (permitindo a interoperabilidade e a portabilidade de dados entre diferentes computadores e/ou componentes de software) e escalabilidade (provendo um crescimento sustentável em tamanho, abrangência e gerenciamento).

A garantia de **acesso** e de **transparência** ao usuário final, juntamente com a **abertura** e a **escalabilidade** dos seus componentes, demanda que um sistema distribuído seja concebido e projetado considerando o modelo ideal de **interação entre componentes** e a sua **organização em camadas** (SOMMERVILLE, 2011).

Com relação à interação entre componentes, pode-se assumir uma comunicação síncrona baseada em chamadas remotas de procedimento (*Remote Procedure Calls - RPCs*), onde um software hospedado em um dos computadores do sistema invoca um serviço fornecido por outro software hospedado em outro computador, passando um conjunto mínimo de informações para que o serviço

seja executado corretamente e aguardando uma resposta. Esse modelo é conveniente em cenários onde o consumidor do serviço pode aguardar a resposta antes de continuar suas atividades, e o conjunto mínimo de informações é pequeno.

Para outros cenários, uma comunicação assíncrona baseada em mensagens é uma alternativa válida. **Mensagens** são estruturas de dados de tamanho variável que permitem o envio e o recebimento de um conjunto de dados comumente maior do que aquele disponível nas RPCs. Essas mensagens são organizadas em filas, de modo que sejam persistidas (gravadas, armazenadas) para que não se percam até terem sido enviadas ou utilizadas após o recebimento, o que dá robustez ao sistema que as utiliza.

Dada essa organização e a garantia de que dados não serão extraviados pelo emissor e pelo receptor, interações assíncronas costumam ser a base de projetos de interoperabilidade entre sistemas heterogêneos. Com relação à organização em camadas, os componentes do sistema podem ser dispostos de acordo com adaptações de outros sistemas conhecidos ou seguindo as orientações clássicas descritas na literatura especializada (como segue):



- **Middleware** - em sistemas distribuídos heterogêneos, fundamental para viabilizar a comunicação entre os demais componentes. Costuma ser instalado em cada um dos computadores do sistema, entre as aplicações utilizadas pelos usuários finais e o sistema operacional (vide Figura 5). Assume o papel de suporte às interações entre dois ou mais componentes (oferecendo transparência de localização, ou seja, permitindo que um software utilize os serviços de outro software sem necessariamente saber onde ele está instalado) e também de provedor de serviços comuns, no qual ele próprio disponibiliza funcionalidades aos demais componentes do sistema sem que estes precisem trocar mensagens ou executar RPCs entre si.

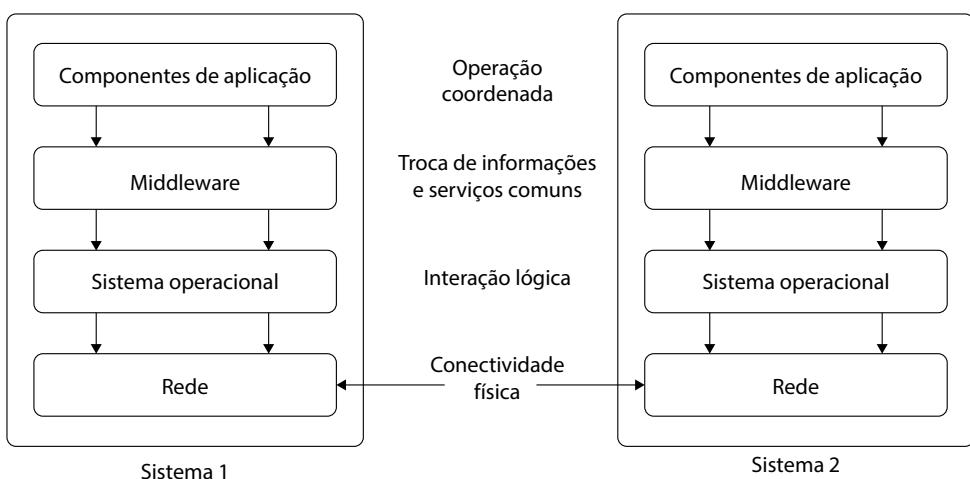


Figura 5 - O middleware como componente de um sistema distribuído
Fonte: adaptado de Sommerville (2011).

Descrição da Imagem: na figura são apresentados dois sistemas (Sistema 1 e Sistema 2) que compõem um sistema distribuído, ambos representados por duas caixas dispostas lado-a-lado. Cada um dos sistemas é decomposto em componentes menores, representados por retângulos, dispostos e conectados por setas direcionais de cima para baixo dentro de cada uma das caixas, como segue: "Componentes de aplicação" (que são executados entre os sistemas 1 e 2 de forma coordenada), "Middleware" (responsável pela troca de informações e serviços comuns), "Sistema operacional" (responsável pela interação lógica entre os sistemas) e "Rede" (que provê a conectividade física entre os sistemas).

- **Cliente-servidor** - organização na qual os componentes do sistema distribuído assumem o papel de clientes (equivalentes a consumidores de recursos) e servidores (responsáveis por prover esses recursos) (como

pode ser observado na Figura 6). O processamento pode ser dividido entre ambos os papéis, sendo essa divisão orientada pelo número de camadas que o sistema oferece. Menos camadas implicam em uma divisão mais equilibrada, enquanto múltiplas camadas costumam deixar para o cliente apenas a função de apresentação – transferindo para os servidores o gerenciamento, o processamento e o armazenamento de dados. É possível que servidores se comuniquem entre si para oferta e consumo de serviços, o que faz com que em determinados contextos um servidor se torne o cliente de outro servidor. O número de servidores em um sistema costuma variar de acordo com o volume de solicitação de serviços, o que implica em uma demanda natural por escalabilidade.

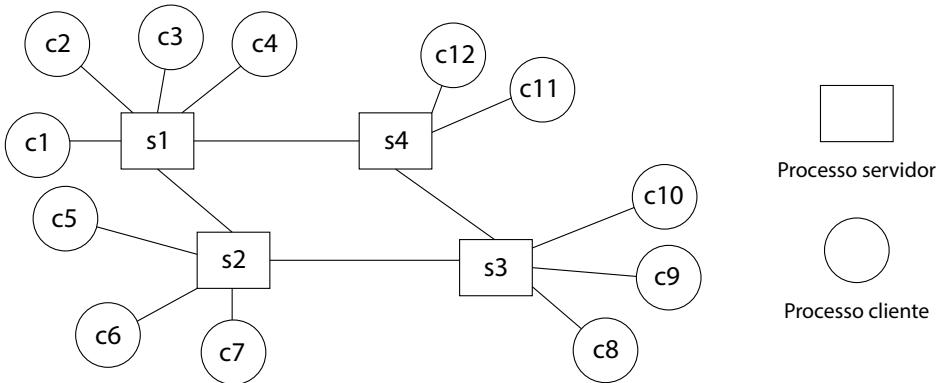


Figura 6 - Uma organização cliente-servidor genérica / Fonte: adaptado de Sommerville (2011).

Descrição da Imagem: na figura é apresentada uma organização genérica para sistemas cliente-servidor, com processos servidores sendo representados como quadrados, processos clientes sendo representados como círculos e as conexões sendo representadas por linhas. À direita da figura há a legenda dos elementos - quadrado: Processo servidor; círculo: Processo cliente. Quatro processos servidores (s1, s2, s3 e s4) estão conectados entre si sendo: s1 conectado a s2; s2 conectado a s3; s3 conectado a s4 e; s4 conectado a s1, e 12 processos clientes (numerados de c1 a c12) estão distribuídos e conectados a esses servidores: c1, c2, c3 e c4 conectados a s1; c5, c6 e c7 conectados a s2; c8, c9 e c10 conectados a s3; c11 e c12 conectados a s4.

- **Mestre-escravo (*master-slave*)** - organização na qual um conjunto de componentes especializados voltados à execução de tipos de processamento muito específicos (escravos) são coordenados por um componente central (mestre). Cabe ao componente mestre acessar ou receber dos com-

ponentes escravos, periodicamente, o resultado dos seus processamentos individuais ou os dados adquiridos em um intervalo de tempo determinado, consolidando-os para a solução do problema-alvo do sistema. É uma arquitetura comum em sistemas de tempo real ou quase-real onde existem restrições temporais severas, como controle de tráfego ou monitoração de dispositivos (como pode ser observado na Figura 7).

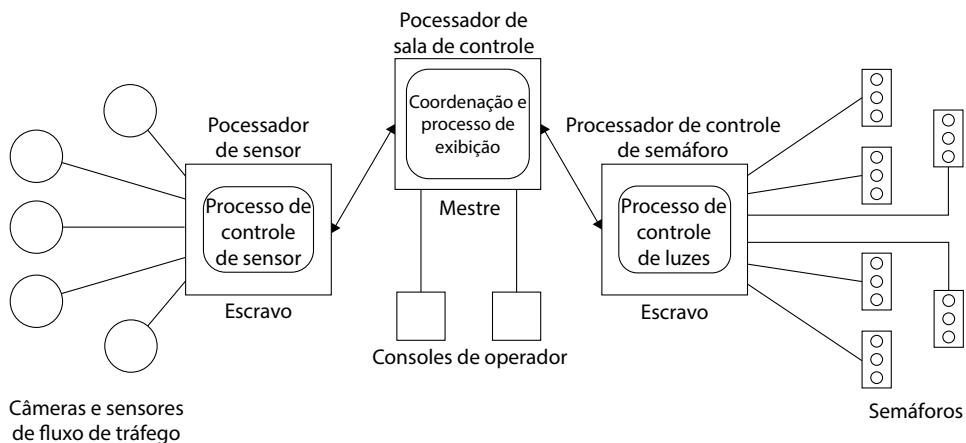


Figura 7 - Uma organização mestre-escravo para um sistema hipotético de controle de tráfego /
Fonte: adaptado de Sommerville (2011).

Descrição da Imagem: na figura é apresentado um exemplo de sistema de controle de tráfego organizado como mestre-escravo. Localizado no centro da figura, o Mestre (representado como um quadrado), nomeado como “Coordenação e processo de exibição” assume o papel de ‘Processador de uma sala de controle’ disponibilizando ‘Consoles de operador’ para acesso dos seus usuários (representado por dois quadrados menores ligado ao quadrado central por uma linha). À esquerda e à direita do mestre estão posicionados dois componentes do tipo ‘Escravo’ (também representados por como quadrados), conectados ao primeiro por com setas bidirecionais. O componente à esquerda é um Processador de sensor, nomeado como “Processo de controle de sensor”, e se conecta a um conjunto de cinco sensores (representados por círculos) nomeados como “Câmeras e sensores de fluxo de sensor de tráfego”. O componente à direita é um ‘Processador de controle de semáforos’ nomeado como “Processo de controle de luzes”, e se conecta a seis equipamentos desse tipo, nomeados como “Semáforos”, para atuar no funcionamento de suas luzes.

- **Ponto-a-ponto (peer-to-peer, ou P2P)** - arquitetura na qual todos os componentes do sistema possuem as mesmas atribuições, sendo responsáveis tanto pelo fornecimento quanto pelo consumo de serviços (vide Figura 8, à esquerda). Para processamento, adota a estratégia de **dividir para conquistar** (o problema a ser resolvido é dividido em problemas menores, independentes,

com cada um desses problemas sendo enviado a um dos componentes para resolução). As respostas aos problemas menores são consolidadas ao final. Para armazenamento de dados, os componentes disponibilizam seus recursos e permitem ser consultados para a recuperação de dados que porventura estejam sob sua guarda. Em sistemas organizados em torno de um número elevado de componentes pode ser necessária a promoção de alguns desses componentes a um status de superponto, o equivalente a um catálogo que relaciona componentes a serviços oferecidos. A descoberta e o acesso aos serviços do sistema, nesses cenários, ocorre via consulta a esses superpontos seguido de um acesso direto ao(s) componente(s) que os oferecem (vide Figura 8, à direita).

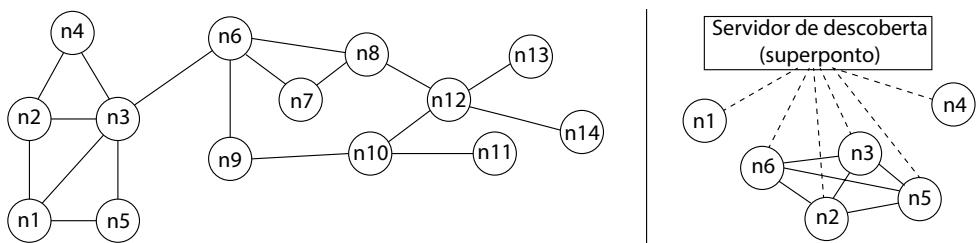


Figura 8 - À esquerda, componentes de um sistema distribuído com as mesmas atribuições conectados ponto a ponto; à direita, um sistema P2P com uma estrutura de supernodo para catalogação e descoberta de serviços / Fonte: adaptado de Sommerville (2011).

Descrição da Imagem: à esquerda da figura é representado um sistema distribuído formado por componentes idênticos, desenhados como círculos e numerados de n1 a n13. Os componentes estão conectados, dois a dois, por linhas: n1 com n2; n1 com n3; n1 com n5; n3 com n5; n2 com n3; n2 com n4; n3 com n4; n3 com n6; n6 com n7; n6 com n8; n6 com n9; n7 com n8; n9 com n10; n10 com n11; n8 com n12; n10 com n12; n12 com n13; n12 com n14. À direita da figura é representado um sistema distribuído formado por um superponto responsável pela catalogação e descoberta de serviços, desenhado como um retângulo nomeado como "Servidor de descoberta (superponto)". Esse superponto se conecta, por linhas tracejadas, a outros componentes desenhados como círculos, numerados de n1 a n6. Os componentes n2, n3, n5 e n6 se conectam entre si, dois a dois, através de linhas contínuas.

O levantamento histórico feito sobre centralização e distribuição, tanto para hardware quanto para software, trouxe os esclarecimentos necessários para a equipe sobre os rumos do projeto. Ainda utilizados por grandes corporações, os *mainframes* e os minicomputadores implicam em um custo proibitivo para o Hospital sem Papel. Esse custo provém não apenas da aquisição dos equipamentos, mas principalmente da manutenção da infraestrutura necessária para viabilizar sua

utilização e da manutenção da equipe técnica altamente qualificada necessária para operá-los em sua capacidade máxima.

Dessa forma, a adoção de servidores e PCs foi indicada como a melhor alternativa para a rede. Foi considerada a adoção de sistemas cliente-servidor em cada hospital, com cada setor de um hospital sendo atendido por um servidor (como, por exemplo, radiologia e patologia). Essa configuração permite que cada servidor possa ser especializado nas funções do setor que atende, e ser acessado por sistemas cliente desenvolvidos de acordo com as especificidades das tarefas que cada setor realiza.

Nas situações em que a monitoração contínua dos pacientes atendidos é uma necessidade, foi considerada a adoção de um modelo híbrido de (1) cliente-servidor e (2) mestre-escravo; dados coletados e tratados por (2) são consolidados por (1) para fins operacionais e administrativos.

Por fim, a visão macro da rede para a administração central será obtida pela consolidação dos dados adquiridos individualmente em cada hospital em uma estrutura de armazenamento centralizada, na forma de **índices** e um ***data warehouse*** (armazém de dados). Através dos índices será possível à administração acessar dados mais volumosos que permanecerão armazenados em cada hospital; o *data warehouse*, por sua vez, será a fonte de dados utilizada na geração de relatórios gerenciais em nível de hospital, em nível regional ou mesmo em nível global abrangendo toda a rede.



Com a definição de que serão adotados sistemas distribuídos baseados em uma plataforma convencional de servidores e clientes, Luísa e sua equipe iniciaram a discussão sobre os **SGBDs - Sistemas Gerenciadores de Banco de Dados** - mais qualificados para compor os alicerces do projeto.

Durante a discussão, que considerou diferentes cenários de uso com base no conhecimento registrado nas etapas anteriores, Samantha (a especialista em DevOps alocada para auxiliar nos trabalhos de instalação, configuração e entrega de soluções de hardware e software) ponderou que a escolha desses sistemas gerenciadores deveria observar critérios de **consistência, disponibilidade e tolerância a partições** (falhas) na rede de comunicação. Para tanto, ela sugeriu a observância ao **teorema CAP**, proposto no ano 2000 por Eric Brewer (BREWER, 2000) e provado formalmente por Gilbert e Lynch em 2002 (GILBERT; LYNCH, 2002).

O teorema CAP (**Consistency, Availability e Partition tolerance**) propõe que um sistema distribuído, em um determinado momento, pode assegurar o cumprimento de duas das três propriedades que o nomeiam (SILVA *et al.*, 2021):

- **C (consistency ou consistência)** - garantia de que todo componente do sistema distribuído retornará a resposta correta a uma requisição de dados. Significa que, caso uma sequência de alterações tenha sido feita nos dados, a versão mais recente desses dados é aquela que será retornada para quem a solicitar.
- **A (availability ou disponibilidade)** - garantia de que toda requisição que foi recebida pelo sistema por meio de um componente que não apresenta falhas será respondida em algum momento.
- **P (partition tolerance ou tolerância a partições)** - garantia de que o sistema distribuído continuará em

funcionamento, mesmo na presença de partições na rede de comunicação que isole um ou mais de seus componentes.

Ao serem consideradas em pares, as propriedades citadas anteriormente permitem antever o comportamento dos sistemas que as apresentam, como segue:

- **Sistemas CA** - são sistemas distribuídos consistentes, mas que não toleram partições na rede de comunicação - falhas desse tipo tornam o sistema inoperante. Apesar de não estar acessível, o sistema mantém a propriedade da disponibilidade; pode parecer contraditório, mas a afirmação está de acordo com a definição de availability (disponibilidade) citada anteriormente, onde são consideradas apenas as requisições recebidas pelo sistema.
- **Sistemas CP** - sistemas que toleram partições na rede de comunicação, porém, compromete a sua disponibilidade. Neles, as respostas corretas das requisições poderão ser enviadas com muito atraso, ou, dependendo da gravidade da falha de comunicação, nunca serem enviadas.
- **Sistemas AP** - são sistemas com alta disponibilidade capazes de tolerar partições na rede de comunicação, porém sacrificando a consistência das respostas enviadas a requisições. Neles, os componentes com dados desatualizados podem responder a requisições que receberem, sem aguardar por atualizações pendentes que, dependendo da gravidade da falha de comunicação, podem nunca ser executadas.

Quando abordados como sistemas distribuídos, os SGBDs NoSQL podem ser classificados de acordo com as proposições do teo-

rema CAP (vide Figura 9). Feita essa classificação, o comportamento desses sistemas com relação à disponibilidade e ao acesso aos dados armazenados pode ser conhecido com antecedência e auxiliar na escolha das melhores soluções a serem adotadas.

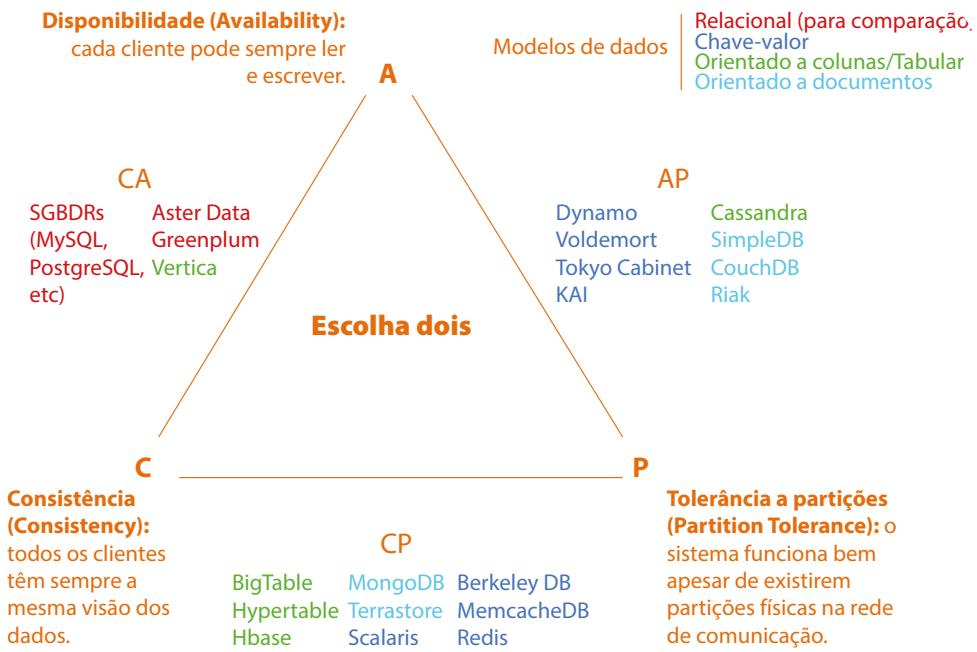
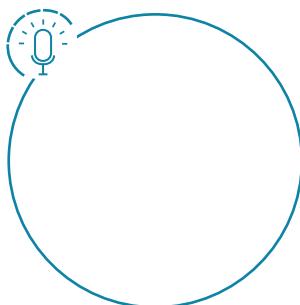


Figura 9 - O teorema CAP utilizado na classificação de bancos de dados NoSQL
Fonte: adaptado de Angadi e Gull (2013).

Descrição da Imagem: na figura, no canto superior direito, há a legenda para os “Modelos de Dados: Relacional (para comparação); Chave-Valor; Orientado a Colunas/Tabular e; Orientando a Documentos. No centro da figura o teorema CAP é apresentado como um triângulo com cada uma das suas propriedades ocupando o lugar de um vértice juntamente com uma breve descrição: A (availability ou disponibilidade): “cada cliente pode sempre ler e escrever” no topo; C (consistency ou consistência): “todos os clientes têm sempre a mesma visão dos dados” no canto inferior esquerdo; e P (partition tolerance ou tolerância a partições): “o sistema funciona bem apesar de existirem partições físicas na rede de comunicação” no canto inferior direito. Para cada combinação de propriedades, duas a duas, são relacionados diferentes SGBDs classificados de acordo com o seu modelo de dados principal. Para a combinação CA são listados os SGBDs relacionais: “Mysql, Postgres, Aster Data e Greenplum”, é listado também o SGBD orientado a colunas: “Vertica”. Para a combinação AP são listados os SGBDs chave-valor: “Dynamo, Voldemort, Tokyo Cabinet e KAI”, são listados também os SGBDs orientados a documentos: “SimpleDB, CouchDB e Riak”, além do SGBD orientado a colunas: “Cassandra”. Para a combinação CP são listados os SGBDs orientados a colunas: “BigTable, Hypertable e Hbase”, além dos SGBDs orientados a documentos: “MongoDB e Terrastore” e os SGBDs chave-valor: “Scalarmis, Berkeley DB, MemcacheDB e Redis”.



O termo **NoSQL** pode ser interpretado como “**no SQL**” (indicando a ausência de uma interface de acesso que utiliza a linguagem SQL - *Structured Query Language*, em português, Linguagem de Consulta Estruturada); porém, sua interpretação mais difundida e aceita é a de “**not only SQL**” – indicando que, além de SQL, outras interfaces de acesso estão disponíveis aos seus usuários.



A observância das propriedades de consistência (*consistency*), disponibilidade (*availability*) e tolerância a partições (*partition tolerance*) do teorema CAP é fundamental para a escolha acertada de um SGBD NoSQL. Acompanhe em nosso podcast os comentários feitos sobre como alguns dos SGBDs NoSQL mais conhecidos do mercado implementam essas propriedades e qual é o impacto para o usuário final. Não perca!

Com a análise do teorema CAP e a possível classificação dos SGBDs NoSQL baseada nas suas proposições, a equipe de trabalho considerou, em um primeiro momento, que a adoção de um único modelo de dados é insuficiente para atender às demandas abrangentes do projeto. A heterogeneidade prevista para as origens desses dados, a frequência de sua aquisição, a necessidade de consolidá-los com base nas políticas da rede de hospitais e de mantê-los durante os períodos definidos por lei demanda a construção de um ecossistema de soluções para armazenamento.

Luísa (como líder do projeto), Gaspar, Laura e Samantha (como líderes de equipe), compreenderam a necessidade de aprofundar o conhecimento de todos com relação aos diferentes modelos de dados que o Hospital sem Papel irá demandar, bem como das tecnologias existentes que sejam capazes de suportá-los.

O levantamento histórico e conceitual feito para **Big Data** e suas características permitiu que Luísa, Gaspar, Laura e Samantha classificassem o Hospital sem Papel como um projeto passível de ser abordado de acordo com as suas premissas.

O estudo conduzido sobre a **computação centralizada** em **mainframes** e **minicomputadores**, a **computação descentralizada** em **servidores** e **PCs** e os **sistemas distribuídos**, permitiu que a equipe decidisse sobre a melhor opção

de arquitetura capaz de atender às demandas do serviço, tanto no escopo local, a cada hospital componente da rede, quanto no escopo global referente à gestão da rede como um todo.

Por fim, o **teorema CAP** e a sua utilização na classificação de **Sistemas Gerenciadores de Banco de Dados NoSQL** foi suficiente para nivelar o conhecimento e o entendimento de todos os envolvidos. Com esse nivelamento, passa a ser possível que a equipe explore em detalhes os diferentes modelos de dados úteis ao projeto, bem como a sua implementação utilizando as ferramentas computacionais disponíveis no mercado. O trabalho começa com dados **estruturados** e **semiestruturados** na forma de **documentos**.

Chegamos ao fim da primeira etapa de nossa primeira imersão em Bancos de Dados NoSQL. Já aprendemos bastante, mas ainda temos muita coisa pela frente. Para relembrarmos e fixarmos ainda mais os conhecimentos vistos até agora, vamos realizar algumas atividades! Caso precise, retorne ao nosso conteúdo para rever os conceitos.



1. Assinale à opção que relaciona as características correspondentes aos três 'Vs' principais de Big Data:
 - a) Dados válidos por um período determinado; dados de fácil compartilhamento; dados em grande quantidade.
 - b) Dados gerados com uma frequência alta; subconjunto de dados úteis; dados cuja quantidade e frequência de aquisição variam.
 - c) Dados em grande quantidade; dados gerados com uma frequência alta; dados apresentados em diferentes formatos.
 - d) Dados confiáveis; dados difíceis de integrar com outros dados; dados que geram retorno (lucro) para a área de negócio.
 - e) Dados apresentados em diferentes formatos; dados válidos por um período determinado; subconjunto de dados úteis.
2. Com relação à organização de dados de forma estruturada, semiestruturada e não estruturada, qual das opções a seguir apresenta uma afirmação FALSA?
 - a) Tabelas de bancos de dados relacionais são exemplos de estruturas de armazenamento não estruturado.
 - b) Arquivos em JSON e XML são considerados veículos semiestruturados para organização de dados.
 - c) Dados semiestruturados são caracterizados pela ausência de estrutura em parte do seu conteúdo.
 - d) Imagens, vídeos e sons, pela sua característica não estruturada, são exemplos de dados que demandam mais pesquisa e recursos para uso efetivo.
 - e) Esquemas de bancos de dados relacionais são organizadores naturais para dados estruturados.



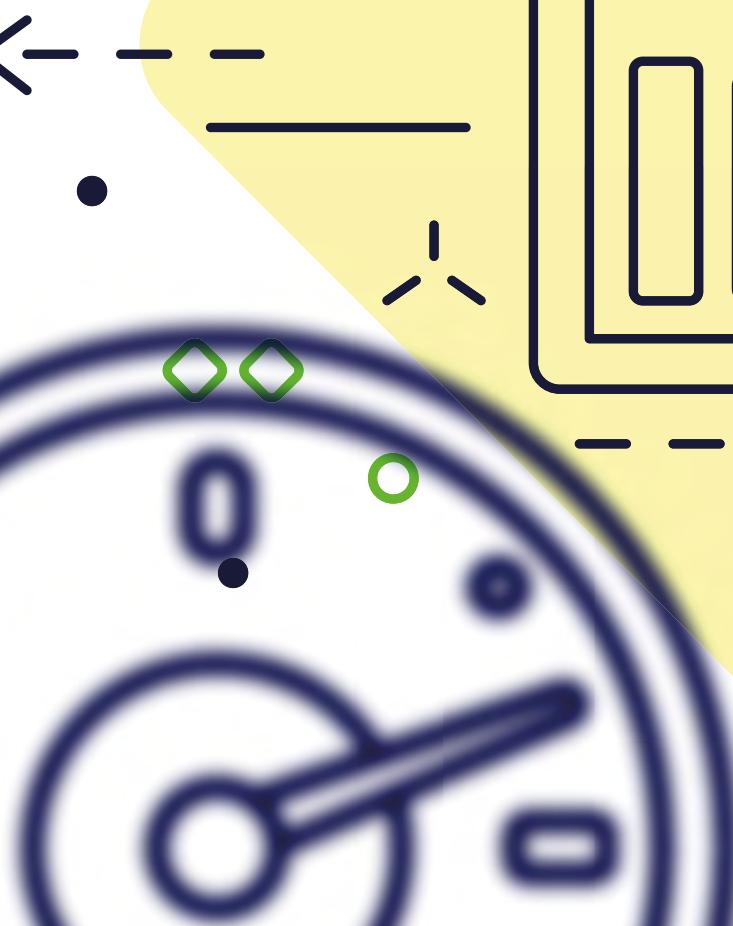
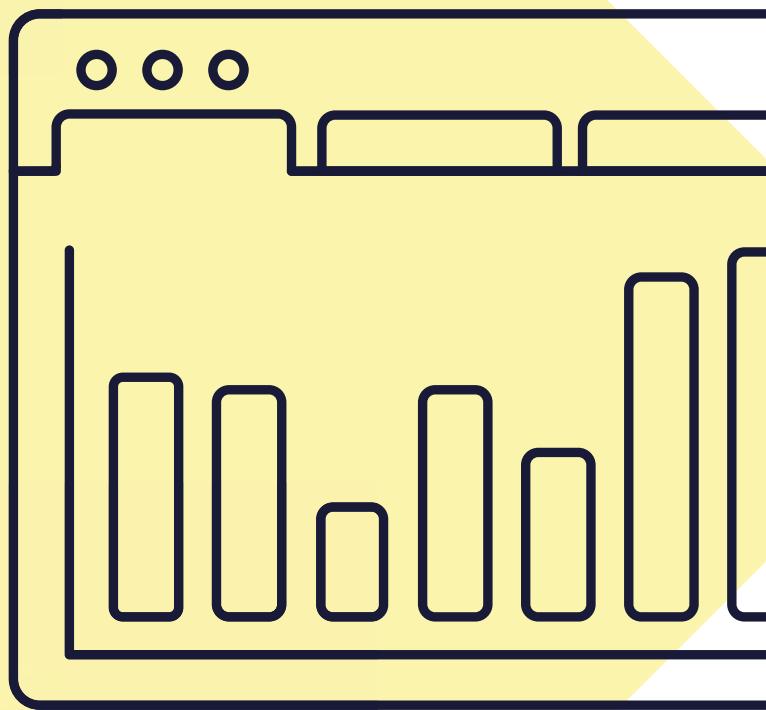
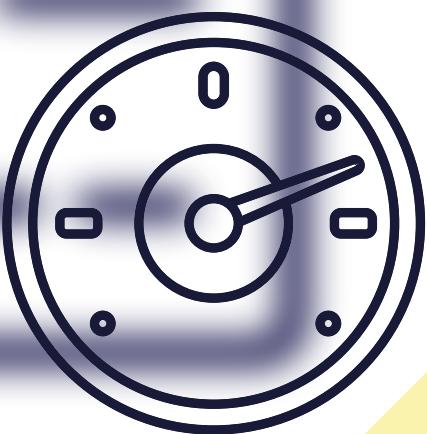
3. Considerando a aquisição, instalação e configuração de mainframes, a configuração de sistemas distribuídos e o processo de downsizing adotado por instituições, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
 - a) O downsizing das instituições nas décadas de 1970 e 1980 implicou na redução do número de equipamentos, levando à aquisição de mainframes e minicomputadores.
 - b) Servidores e computadores pessoais (PCs) dificultam a conexão de periféricos fabricados por terceiros.
 - c) Sistemas distribuídos são construídos para uso exclusivo em redes locais (LANs).
 - d) O downsizing contribuiu para a substituição de um modelo de processamento centralizado e em lotes (*batch*), executado por mainframes e minicomputadores, por um modelo de processamento descentralizado (executado por servidores e PCs).
 - e) O modelo de negócio de um fornecedor único para hardware, software e serviços foi mantido na troca de mainframes e minicomputadores por servidores e PCs..
4. Organização de sistemas distribuídos onde seus componentes assumem papéis de consumidores e provedores de recursos, podendo variar em número de camadas e demandar um aumento de desempenho através de escalabilidade:
 - a) Middleware..
 - b) Cliente-servidor.
 - c) Mestre-escravo (*master-slave*).
 - d) Ponto-a-ponto (*peer-to-peer*, ou P2P).
 - e) Nenhuma das alternativas.
5. Um SGBD NoSQL que apresenta consistência eventual (onde um cliente pode receber dados desatualizados, dependendo de qual componente do sistema está acessando) pode ser classificado, de acordo com o teorema CAP, a partir da combinação de quais propriedades?
 - a) CA.
 - b) CP.
 - c) AP.
 - d) CAP.
 - e) Nenhuma das alternativas.

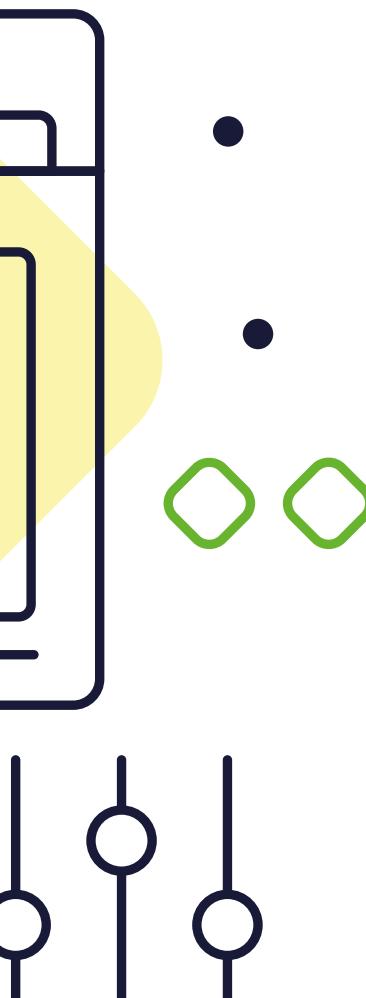
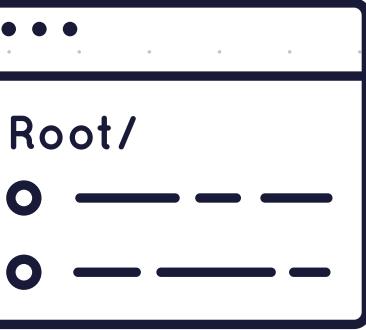
Bancos de Dados Para Documentos Estruturados e Semiestruturados

Dr. Alexandre Savaris

OPORTUNIDADES DE APRENDIZAGEM

Nesta unidade, serão introduzidas as definições gerais de *workflow*, PEP, RES e S-RES. Serão revisitados os conceitos fundamentais do modelo de dados relacional, além de apresentados os fundamentos dos Sistemas Gerenciadores de Bancos de Dados orientados a documentos. Como ferramenta para exemplificação e experimentação, será utilizado o MongoDB (SGBD orientado a documentos) para a construção de queries e instruções de inclusão, alteração e exclusão de documentos em um cenário simplificado e hipotético com dados de pacientes. É citada também a compatibilidade deste modelo de dados com padrões de dados em saúde internacionalmente reconhecidos.





A busca pela eficácia e eficiência em seus processos produtivos, processos de transformação, processos de prestação de serviços e/ou processos de aquisição, tratamento e manutenção de dados e informações (entre outros) orienta instituições de diferentes áreas a organizar suas atividades por meio da implementação de fluxos de trabalho (*workflows*).

Um **workflow** pode ser definido como a organização e a automação de procedimentos onde tarefas, documentos e informações são transmitidos entre os seus participantes obedecendo a um conjunto de regras, visando contribuir para que a instituição que o implementa atinja um objetivo previamente especificado (HOLLINGSWORTH, 1994). Suas origens remetem aos trabalhos de Frederick Taylor e Henry Gantt, que foram realizados no início do século XX voltados à organização racional de atividades de manufatura.

Especificamente na área de prestação de serviços em saúde, a adoção e o *fine-tuning* (ajuste fino ou configuração fina) de processos de *workflow* objetivam, grosso modo, **reduzir desperdícios e eliminar gargalos** perceptíveis no dia a dia das instituições (HRIBAR *et al.*, 2015). Embora seja comumente relacionado a insumos, o desperdício citado considera o tempo dos profissionais de saúde – tempo esse que é financeiramente caro, e cujo mau uso impacta negativamente a qualidade do atendimento recebido pelos pacientes. Os gargalos, por sua vez, são o resultado da repetição ou execução desnecessária de atividades, além do gasto excessivo de tempo especializado em atividades cuja execução pode ser simplificada.

Para os profissionais de saúde, um *workflow* não otimizado e deficiente em termos de organização de tarefas e comunicação de dados implica em uma sobrecarga no volume das atividades a serem executadas.

das. Essa sobrecarga contribui para o burnout, problema cada vez mais observado referente ao desgaste que prejudica os aspectos emocional e físico do indivíduo, causando seu esgotamento profissional (IMPROVING, 2022).

Para os pacientes, um *workflow* não efetivo aumenta o tempo de permanência no estabelecimento de saúde por atrasar o fluxo natural de admissão e registro, triagem, atendimento, diagnóstico, tratamento e alta. Além de comprometer o bem-estar do indivíduo, a ineficácia e a ineficiência no atendimento geram custos individuais com um efeito cascata para o estabelecimento (GRANJA *et al.*, 2014).

Para a rede de hospitais atendida por Luísa e sua equipe, a pesquisa, o projeto, a implementação, a implantação e a manutenção de um *workflow* eficiente tanto em nível local a cada estabelecimento quanto em nível global à rede visando a sua gestão são atividades compulsórias.

Pelo fato de estas atividades demandarem a efetiva aplicação de Tecnologias da Informação e Comunicação (TICs) para atender principalmente à questão da automação de procedimentos, a identificação dessas tecnologias, o seu estudo e a sua posterior adaptação/configuração visando a compatibilizá-las, tornam-se atividades essenciais a serem executadas pela equipe de TI.

Alguns fatores norteadores, necessários para a execução destas atividades, são o resultado de etapas anteriores do trabalho da equipe, como por exemplo as vantagens dos **sistemas distribuídos** quando comparados a sistemas monolíticos/centralizados, e a adoção de **Sistemas Gerenciadores de Bancos de Dados (SGBDs) NoSQL** para o atendimento aos diferentes cenários que se apresentarão durante a implementação do fluxo de trabalho local/global.

Apesar de o Hospital sem Papel (*Paperless Hospital*) contribuir com a substituição das mídias impressas pelo tráfego de dados 100% digital, os setores e os profissionais a serem atendidos pelo *workflow* em cada instituição permanecem os mesmos.

A entrada dos pacientes se dá pela emergência ou pela admissão através do registro e do agendamento de consultas e exames; os laboratórios ficam responsáveis pela execução das análises clínicas e pela disponibilização dos resultados; a farmácia assume a dispensação (distribuição) dos medicamentos; o setor de radiologia se encarrega da execução dos exames de imagem; o setor de patologia analisa e entrega resultados relacionados à análise dos órgãos, tecidos e fluidos corporais; e os especialistas se responsabilizam pela alta dos pacientes quando as consultas, exames e tratamentos forem concluídos (HOW, 2022).

Mesmo incompleta, essa lista de setores e profissionais utiliza dados comuns (além de gerar dados próprios, como resultados) necessários em etapas posteriores do fluxo de trabalho. A escolha acertada de um ou mais modelos de dados que tragam completude e flexibilidade ao armazenamento em cada etapa e à transmissão às etapas seguintes é um dos primeiros desafios a serem superados.

Com o conhecimento referente aos cinco sistemas de uso mais comum em hospitais, é consenso entre os integrantes da equipe de TI de que um *workflow* para esse tipo de estabelecimento é o resultado de uma combinação organizada desses sistemas; nele, os dados gerenciados por cada um dos sistemas podem ser reutilizados e/ou complementados pelos demais, seguindo uma ordem lógica que esteja de acordo com o fluxo de recepção, atendimento e alta dos pacientes.

Para contribuir com esse raciocínio, sua ajuda é mais uma vez bem-vinda: procure relacionar cada um dos sistemas pesquisados com uma (ou mais) das etapas que foram apresentadas e ordenadas no parágrafo anterior, se possível especificando quais itens de dados podem ser encaminhados de etapa para etapa (ou de sistema para sistema) para reutilização ou complementação.

A compatibilização de diferentes sistemas para a composição de um *workflow* é um desafio mesmo para as equipes de trabalho que dominam tanto a área de negócio que está sendo atendida quanto as tecnologias envolvidas na construção das soluções necessárias. Especificamente no tocante aos dados gerenciados por esses sistemas, dificilmente será identificada uma *silver bullet* – expressão que denota uma solução simples, genérica e rápida para problemas grandes e de difícil resolução – ou *one-size-fits-all*, uma solução única para todos os problemas.

No contexto da rede de hospitais em questão, a finalidade dos sistemas utilizados implica a adoção de SGBDs cujos modelos de dados sejam orientados aos problemas endereçados por cada uma das etapas do fluxo de trabalho das instituições. O tratamento direcionado a dados alfanuméricos estruturados e semiestruturados (presente em praticamente todas as etapas do fluxo), por exemplo, é diverso ao tratamento direcionado a dados binários não estruturados (necessário em outras etapas). Aqui, a variedade de soluções NoSQL contribui para que cada problema seja abordado de acordo com suas especificidades.

Em seu diário de bordo, registre o seu entendimento e as suas opiniões a respeito da relação entre os sistemas utilizados em hospitais e a sua participação no *workflow* adotado por esse tipo de instituição, não esquecendo do impacto do gerenciamento e da escolha do modelo de dados em cada uma das etapas

do fluxo de trabalho. Posteriormente, revise esse registro com base no que será apresentado no restante deste módulo. Este conteúdo poderá ser utilizado pela equipe de TI como complemento ao material selecionado para embasar seu processo de tomada de decisão!

DIÁRIO DE BORDO

Com a identificação dos sistemas de informação de uso mais comum em hospitais e a necessidade de organizá-los em um *workflow*, Luísa passou a direcionar as atividades da sua equipe para o entendimento dos resultados a serem obtidos. Afinal, o que se espera do esforço de coordenar a execução de múltiplos sistemas, especificamente no tocante aos dados gerados por esses sistemas? Laura e Gaspar, após discutirem a questão com seus colaboradores, chegaram à conclusão de que um *workflow* bem ajustado, envolvendo sistemas diversos e complementares,

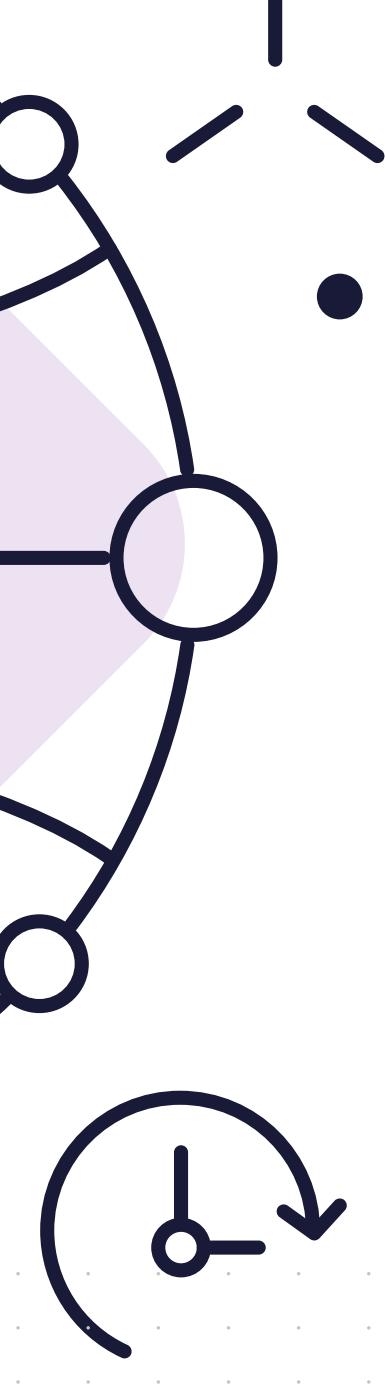
permite a construção de um **Prontuário Eletrônico do Paciente (PEP)** efetivo. O Conselho Federal de Medicina (CFM) define prontuário médico como:



[...] o documento único constituído de um conjunto de informações, sinais e imagens registradas, geradas a partir de fatos, acontecimentos e situações sobre a saúde do paciente e a assistência a ele prestada, de caráter legal, sigiloso e científico, que possibilita a comunicação entre membros da equipe multiprofissional e a continuidade da assistência prestada ao indivíduo (CFM, 2002, on-line).

Esta definição engloba tanto prontuários em papel quanto prontuários eletrônicos, estes últimos viabilizados pela utilização de Tecnologias da Informação e Comunicação em Saúde (TICS). Os PEPs, comumente implantados em um escopo local (um hospital, por exemplo), ao incorporarem funcionalidades de comunicação e interoperabilidade, evoluem para **Registros Eletrônicos em Saúde (RES)** gerenciados por **Sistemas de Registro Eletrônico em Saúde (S-RES)** (SBIS, 2012, on-line).

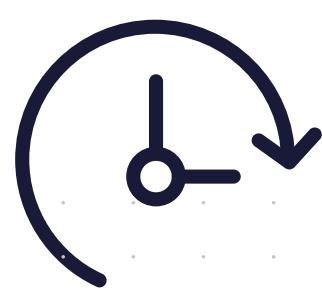




Quando implantados, RES e S-RES permitem o acesso facilitado ao histórico dos pacientes, englobando todos os atendimentos realizados e os procedimentos executados que tenham sido registrados em qualquer instituição que contribua com a organização e a disponibilização dos dados que gerencia. Os dados registrados, por sua vez, são provenientes de aquisições e observações executadas nas diferentes etapas do fluxo de atendimento ao paciente, sendo específicos a cada etapa e seguindo diretrizes e manuais de boas práticas que definem a sua estrutura e a forma de coleta (BRITO, VELOZO, PAVANELLI, 2016; COREN, 2022).

Para a implantação dos PEPs locais aos hospitais da rede, evoluindo em seguida para RES e S-RES, Samantha (*DevOps*) e Marco, o DBA (*Database Administrator* - Administrador de Banco de Dados) da equipe iniciaram a discussão referente aos modelos de dados e tecnologias de suporte a esses modelos.

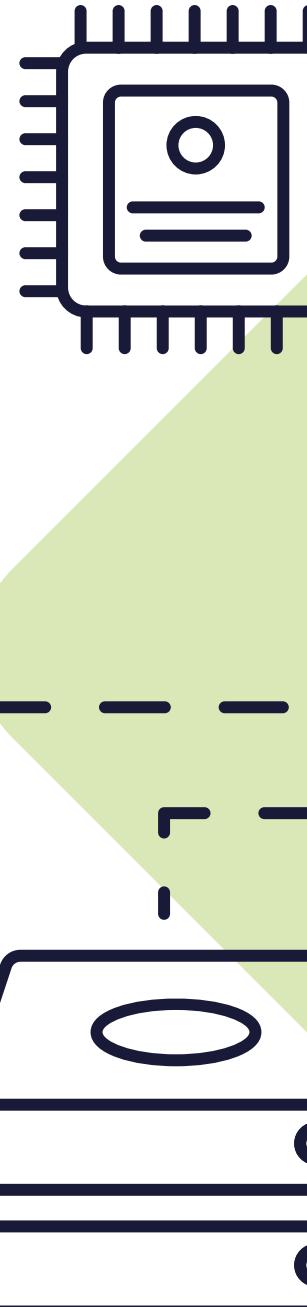
A partir da sua experiência profissional no projeto, modelagem e implantação de bancos de dados para os setores público e privado, e com o entendimento de que os dados a serem gerenciados são em sua maioria estruturados, Marco sugeriu a adoção de **bancos de dados relacionais** suportados pelos seus respectivos sistemas gerenciadores como alternativa. Samantha, por sua vez, ponderou que a adoção de arquiteturas distribuídas e a tendência dos PEPs, RES e S-RES trabalharem com **documentos** orientaria a adoção de tecnologias que utilizassem esses documentos como base para a organização e a persistência do conteúdo desses sistemas.



O **modelo de dados relacional** foi definido por Edgar Frank Codd na década de 1970 (CODD, 1970). Este modelo, atrativo pela sua base matemática e simplicidade, fomentou a implementação de bancos de dados (BDs) e os seus respectivos softwares gerencia-

dores (SGBDs) a partir da década de 1980. O modelo é organizado a partir das seguintes estruturas (vide sua representação tabular na Figura 1) (ELMASRI; NAVATHE, 2018):

- **Domínio** - um conjunto de valores atômicos (indivisíveis), utilizado para especificar o que é possível armazenar em cada parte componente do banco de dados. Exemplos: o número do CPF (Cadastro de Pessoa Física) de um paciente deve ser representado como um número inteiro positivo de até 11 dígitos; o nome de um paciente deve ser representado como um texto formado apenas por letras, limitado a 100 caracteres; a data de nascimento de um paciente deve ser representada como uma data cujo valor não pode ser superior à data do cadastro do paciente.
- **Atributo** - uma parte componente de um banco de dados que segue as especificações de um domínio, cujo papel é compor a estrutura de uma relação. Exemplos: número do CPF, nome e data de nascimento são atributos que juntos definem a estrutura de uma relação no banco de dados responsável por armazenar dados de pacientes.
- **Tupla** - uma lista ordenada de atributos pertencentes a uma relação. Os valores dos atributos de uma mesma tupla representam um elemento do cenário que está sendo modelado pelo banco de dados. Exemplo: uma tupla com o valor 99579629064 para o número do CPF; o valor ‘Marco Antônio’ para o nome; e o valor 29/02/1988 para a data de nascimento represen-



ta um paciente cujos dados são (ou serão) armazenados em uma relação no banco de dados.

- **Relação** - um conjunto de tuplas que obedece a um esquema bem definido (atributos em uma ordem previamente estabelecida, respeitando individualmente um domínio que especifica os dados a serem armazenados). Exemplo: a relação Paciente tem o objetivo de armazenar o nome, o número do CPF e a data de nascimento dos pacientes atendidos por um estabelecimento de saúde.

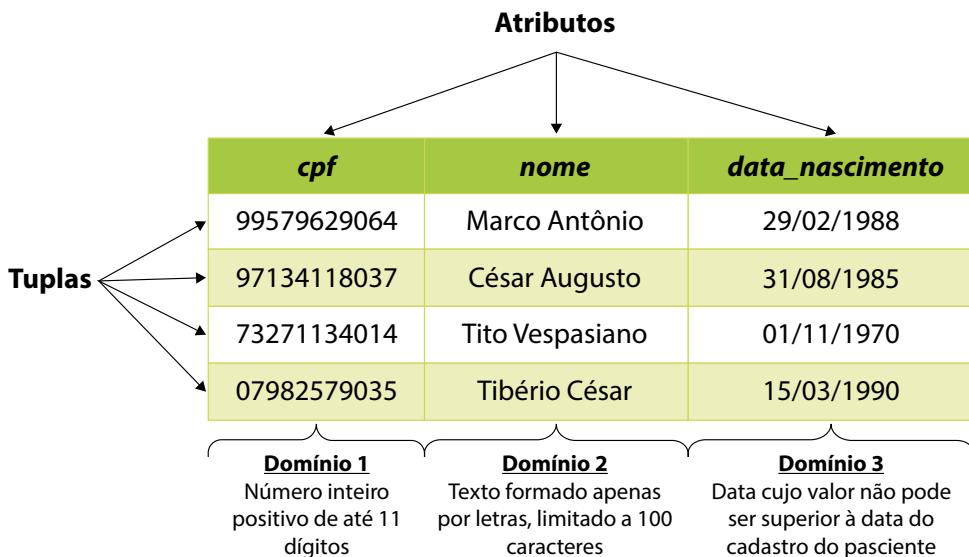


Figura 1 - Domínios, atributos e tuplas de uma relação / Fonte: adaptado de Elmasri e Navathe (2018a).

Descrição da Imagem: na figura é apresentada uma tabela com três colunas, uma linha de cabeçalho e quatro linhas com dados, representando uma relação que tem o objetivo de armazenar dados de pacientes de um estabelecimento de saúde. Cada uma das colunas corresponde a um atributo da relação (em ordem, 'cpf', 'nome' e 'data_nascimento'), e cada linha corresponde a um paciente cujos dados estão armazenados. O primeiro paciente possui o CPF 99579629064, chama-se Marco Antônio e nasceu em 29/02/1988; o segundo paciente possui o CPF 97134118037, chama-se César Augusto e nasceu em 31/08/1985; o terceiro paciente possui o CPF 73271134014, chama-se Tito Vespasiano e nasceu em 01/11/1970; e o quarto paciente possui o CPF 07982579035, chama-se Tibério César e nasceu em 15/03/1990. Na parte inferior da tabela, para cada coluna, é especificado o domínio de dados aceito para cada atributo. O primeiro atributo aceita números inteiros positivos de até 11 dígitos; o segundo atributo aceita textos formados apenas por letras, limitados a 100 caracteres; e o terceiro atributo aceita datas cujo valor não pode ser superior às datas do cadastro dos respectivos pacientes.

Os termos **relação**, **tupla** e **atributo**, definidos no trabalho de Codd, são utilizados no projeto conceitual de bancos de dados relacionais. Em tempo de implementação (quando um banco de dados é construído, nas etapas de projeto lógico e físico), a terminologia assumida é, respectivamente, a de **tabela**, **registro** e **campo**. É nestas etapas, também, que as ideias utilizadas no projeto conceitual são refinadas e adaptadas à ferramenta de software utilizada para gerenciar o BD.

Os refinamentos são executados em etapas sequenciais responsáveis por entregar, como resultado, um banco de dados normalizado. O processo de normalização demanda o respeito à definição das chamadas formas normais, que na prática são regras para a reorganização e a construção de tabelas que visam eliminar a redundância de campos e as inconsistências na distribuição de campos (CODD, 2022). O resultado obtido é um conjunto de tabelas que distribuem entre si, da melhor forma possível, os atributos referentes aos dados mapeados a partir do cenário modelado pelo BD. Um dos efeitos colaterais da normalização de um banco de dados relacional é o aumento no número de suas tabelas.

Esse aumento é decorrente da reorganização dos atributos (campos) originais, que em alguns casos acabam sendo copiados ou realocados para novas tabelas criadas com o objetivo de resolver as questões da múltipla ocorrência de um mesmo campo (por exemplo, um paciente com mais de um número de telefone, vide Figura 2), e questões que envolvem o relacionamento entre tabelas onde a cardinalidade do relacionamento (o número de registros envolvidos em cada um dos lados do relacionamento) é classificada como sendo de muitos-para-muitos (como, por exemplo, o relacionamento entre pacientes e médicos especialistas - vide Figura 3).

A normalização resolve o problema da distribuição dos atributos, mas impacta no desempenho da recuperação dos dados. Com os atributos distribuídos como campos de várias tabelas, recuperá-los passa a ser um exercício de junção entre as tabelas envolvidas (ELMASRI; NAVATHE, 2018b). A **junção entre tabelas (JOIN)** é considerada uma das operações mais custosas executada por um SGBD, custo esse que por vezes motiva a revisão do processo de normalização - e até mesmo a adoção do seu contraponto, a **desnormalização**.

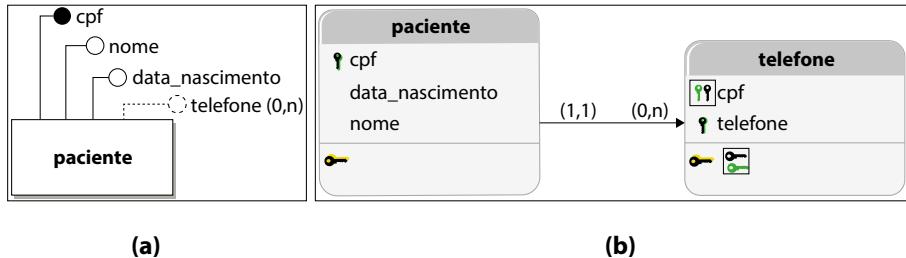


Figura 2 - Normalização da múltipla ocorrência do atributo `telefone` utilizando a regra da Primeira Forma Normal (1FN). Em (a) é exibido o diagrama conceitual da relação `paciente` com seus atributos; em (b) é exibido o diagrama lógico da mesma relação, com a criação da tabela `telefone` para armazenar as múltiplas ocorrências do referido atributo / Fonte: o autor.

Descrição da Imagem: na figura é apresentado à esquerda o diagrama conceitual da relação paciente. No diagrama a relação é exibida como um retângulo, ao qual estão conectados os atributos `cpf`, `nome`, `data_nascimento` e `telefone`. O atributo `cpf` é exibido como um círculo preenchido conectado à relação por uma linha contínua, indicando ser um atributo identificador; os atributos `nome` e `data_nascimento` são exibidos como círculos não preenchidos conectados à relação por linhas contínuas, indicando serem atributos convencionais; o atributo `telefone` é exibido como um círculo tracejado não preenchido conectado à relação por uma linha também tracejada, indicando ser um atributo de múltipla ocorrência. A cardinalidade do atributo é informada entre parênteses logo após o seu nome: `(0, n)`, ou múltipla ocorrência opcional. À direita é apresentado o diagrama lógico correspondente, onde a tabela paciente se conecta a uma nova tabela chamada `telefone`. A tabela paciente mantém os campos `cpf`, `data_nascimento` e `nome`, marcando o campo `cpf` como identificador; a tabela `telefone` é criada com os campos `cpf` e `telefone`, marcando a combinação destes campos como identificador da tabela e o campo `cpf` como uma chave estrangeira para a tabela paciente (com o objetivo de permitir o relacionamento entre ambas as tabelas quando necessário). As tabelas são conectadas por uma seta contínua direcionada de paciente para `telefone`, com as cardinalidades do relacionamento especificando que um paciente pode ter nenhum ou muitos telefones `(1,1)`, e que um telefone pertence a um único paciente `(0,n)`.



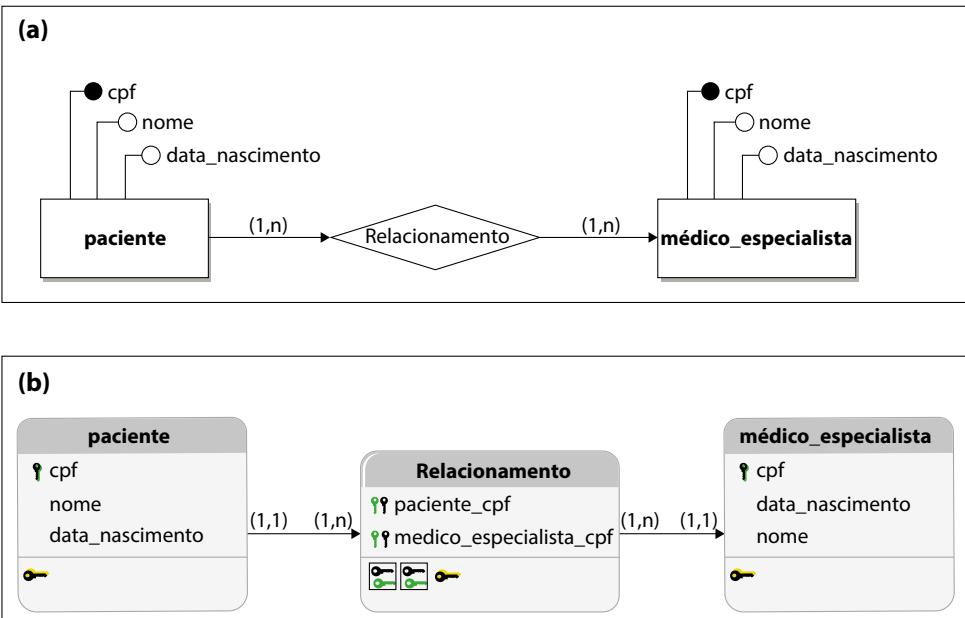


Figura 3 - Normalização de relacionamento muitos-para-muitos entre as relações/tabelas paciente e médico_especialista. Em (a) é exibido o diagrama conceitual do relacionamento entre as duas relações, juntamente com seus atributos; em (b) é exibido o diagrama lógico do mesmo relacionamento, com a criação da tabela Relacionamento para viabilizar a junção entre as duas tabelas originais.

Fonte: o autor.

Descrição da Imagem: na parte superior da figura é apresentado o diagrama conceitual do relacionamento entre as relações paciente e médico_especialista. No diagrama as relações são exibidas como retângulos, aos quais estão conectados os atributos cpf, nome e data_nascimento. O atributo cpf é exibido como um círculo preenchido conectado às relações por uma linha contínua, indicando ser um atributo identificador; os atributos nome e data_nascimento são exibidos como círculos não preenchidos conectados às relações por linhas contínuas, indicando serem atributos convencionais. O relacionamento entre paciente e médico_especialista é apresentado como um losango com o texto Relacionamento, conectado às relações por linhas contínuas. A cardinalidade do relacionamento é apresentada junto a cada relação sendo (1, n) - indicando que cada paciente está relacionado a um ou muitos médicos especialistas, e vice-versa. Na parte inferior da figura é apresentado o diagrama lógico correspondente, onde o losango do diagrama conceitual é convertido em uma tabela de relação. As tabelas paciente e médico_especialista mantém os campos cpf, data_nascimento e nome, marcando o campo cpf como identificador; a tabela Relacionamento é criada com os campos paciente_cpf e medico_especialista_cpf, marcando a combinação destes campos como identificador da tabela e cada campo em separado como uma chave estrangeira para as demais tabelas (com o objetivo de permitir a junção quando necessário). As tabelas são conectadas por setas contínuas direcionadas de paciente e médico_especialista para Relacionamento, com as cardinalidades do relacionamento especificando que cada paciente está relacionado a um ou muitos médicos especialistas, e vice-versa.

Apesar da sua simplicidade e da sua adaptabilidade a diferentes cenários de uso, o modelo de dados relacional possui uma limitação importante relacionada com a definição dos seus esquemas. O esquema de um banco de dados relacional compreende, para cada tabela definida, quais são seus campos, a ordem em que esses campos foram criados e a tipologia de dados e domínio que cada campo deve respeitar, o que o torna bastante rígido e até mesmo inflexível. Essa definição (ou conjunto de regras) compõe o **dicionário de dados do BD**, e é utilizada pelo SGBD para validar os dados inseridos e gerenciados pelo banco (TRIBUNELLA, 2002).

Caso a definição de um esquema mude como decorrência de adaptações em regras de negócio dos sistemas de informação que utilizam o banco de dados em questão, é necessário proceder com alterações explícitas nesse esquema para que as novas regras passem a valer.

Exemplos incluem a **inserção de novos campos** originalmente não previstos, a **alteração da tipologia de dados e/ou domínio** de um ou mais campos existentes, ou mesmo a **alteração na política de obrigatoriedade** de preenchimento de um ou mais campos. Alterações como essa são consideradas sensíveis, costumam demandar janelas de manutenção (que por sua vez podem levar a um *downtime* do BD, deixando-o indisponível) e podem ter impacto no histórico de dados que o banco já armazena.

Os **bancos de dados orientados a documentos**, por sua vez, podem ser considerados uma adaptação dos bancos de dados chave-valor (SILVA *et al.*, 2021). Neles, os valores armazenados são documentos estruturados ou semi-estruturados seguindo os padrões XML (*eXtensible Markup Language*), JSON (*JavaScript Object Notation*) (HAN *et al.*, 2011), ou mesmo YAML (*YAML Ain't Markup Language™*). Uma de suas características primordiais é a flexibilidade de esquema: cada documento armazenado pode ter uma estrutura particular, diferente das estruturas dos demais documentos.

O fato de os documentos serem autocontidos – possuírem tanto o nome dos atributos quanto o seu conteúdo – possibilita aos clientes do SGBD processarem o resultado de consultas sem depender de uma estrutura acessória (como o dicionário de dados de um SGBD relacional, por exemplo). Conforme (TRUICĂ, 2021), outras características que merecem destaque são:

- **Flexibilidade e adaptabilidade** - novos requisitos funcionais e regras de negócio dos sistemas de informação clientes podem ser facilmente atendidos, dada a possibilidade de que novos atributos sejam inseridos, de que atributos existentes sejam modificados e de que documentos com estruturas diversas coexistam em um mesmo BD. Essas alterações são permitidas on the fly, com o BD em uso e o SGBD em produção, um atrativo para cenários em que janelas de manutenção são escassas ou inexistentes.
- **Gerência de dados estruturados, semiestruturados e não estruturados** - a flexibilidade e adaptabilidade do item anterior viabilizam o armazenamento tanto de dados que possuem uma estrutura rígida (como por exemplo documentos gerados a partir de registros de um banco de dados relacional) quanto de dados que possuem uma estrutura parcial ou que não possuem estrutura, como por exemplo dados multimídia provenientes de mídias sociais, registros de log com formatos variados ou dados adquiridos por meio de sensores.
- **Escalabilidade** - concebidos como sistemas distribuídos, sistemas gerenciadores de bancos de dados orientados a documentos acomodam-se facilmente em clusters e grids, escalando horizontalmente seus bancos através da distribuição de documentos por meio de sharding (fragmentação). Com cada fragmento do BD distribuído e/ou replicado entre os nodos disponíveis na infraestrutura, a adoção de hardware convencional para esses nodos torna-se uma alternativa à aquisição de servidores de grande porte, consideravelmente mais caros.



Clusters são caracterizados como grupos de computadores conectados em rede, ocupando uma área geográfica delimitada (como uma sala ou prédio) e cujo hardware tende a ser padronizado. *Grids*, por sua vez, são caracterizados por um conjunto de computadores conectados em redes de larga escala (comumente a própria internet), não se limitam a áreas geográficas e são compostos por hardware heterogêneo.

**NOVAS DESCOBERTAS**

1. Foi definida pelo World Wide Web Consortium (W3C), a partir da década de 1990, como uma linguagem de marcação capaz de ser lida por software, baseada em SGML (*Standard Generalized Markup Language*) e *HyperText Markup Language* (HTML).

2. HTML (*HyperText Markup Language*) é uma linguagem de marcação utilizada para estruturação de conteúdo. Elementos são definidos por tags de abertura e fechamento, as quais definem o comportamento desses elementos (como exibição em negrito, como imagem, dentre outras).

3. JSON (*JavaScript Object Notation*) é um padrão aberto independente que define um formato compacto para troca de dados entre sistemas, especificado no ano 2000. O formato é baseado em pares atributo-valor autodescritivos para representação de conteúdo legível por humanos.

4. YAML (*YAML Ain't Markup Language*) é um formato para serialização de dados legível por humanos proposto em 2001. O formato assume que dados em sua completude podem ser representados como uma combinação de valores simples, mapas e listas.

Ao expor suas sugestões sobre SGBDs relacionais e SGBDs orientados a documentos para a equipe de desenvolvimento, Marco e Samantha receberam o feedback de que os primeiros são velhos conhecidos de todos, e foram estudados e utilizados profissionalmente em outros projetos pela maioria dos seus integrantes.

Entretanto, todos se mostraram interessados na avaliação e testes de ferramentas centradas em documentos, e sugeriram a adoção de um SGBD em particular que é considerado um expoente na área de NoSQL para esse tipo de organização de dados: o **MongoDB**. Em consenso com a opinião da equipe, Samantha e Marco autorizaram que fossem iniciados os estudos da ferramenta com foco em testes, utilizando a estrutura de paciente prevista em um PEP.

MongoDB é um SGBD orientado a documentos cuja criação foi motivada pela área de publicidade on-line. Nas décadas de 1990 e 2000, o uso em larga escala de sistemas para atender a demandas dessa área colocou em xeque os bancos de dados relacionais, cuja escalabilidade vertical mostrou-se cara e muitas vezes insuficiente. Inicialmente pensado como um subproduto de uma

PaaS (*Platform as a Service* (Plataforma como Serviço)), o SGBD foi promovido a produto principal e teve sua primeira versão liberada para *download* em fevereiro de 2009, seguindo um modelo de negócio onde o *software* é *open source* e o treinamento/suporte comercial é pago.

O SGBD oferece, desde as suas versões iniciais, funcionalidades que o destacam de seus pares: ausência de esquema, uso de *JavaScript* e **JSON/BSON (Binary JSON)** como, respectivamente, linguagem e formatos de comunicação e armazenamento de dados, e escalabilidade horizontal facilitada via particionamento e replicação.

Desde então e até as suas versões mais recentes, novos recursos são incorporados de forma a introduzir características de outros modelos (como o relacional, por exemplo) de forma a torná-lo cada vez mais um sistema gerenciador de bancos de dados multipropósito (DEJOY, 2020).



PENSANDO JUNTOS

A escalabilidade vertical implica no aumento dos recursos (comumente processador, memória principal e memória secundária) do computador ou computadores onde um sistema em questão é executado. A escalabilidade horizontal, por sua vez, implica no aumento do número de computadores dedicados a executar um sistema.



NOVAS DESCOBERTAS

BSON (Binary JSON) é a representação binária do formato JSON, criada em 2009. O formato incorpora tipagem e tamanho dos dados armazenados, o que favorece o desempenho do processo de parsing de conteúdo e indexação para acelerar a execução de queries.

Para fins de experimentação e testes, a versão em nuvem do MongoDB pode ser acessada gratuitamente. Conhecida como MongoDB Atlas, a versão gratuita oferece um ambiente de cluster limitado, porém suficiente para o aprendizado da ferramenta.

<https://www.mongodb.com/atlas/database>

O uso do MongoDB como SGBD NoSQL orientado a documentos inicia com a definição e criação de um banco de dados (*database*). O comando `use`, utilizado nessa criação, serve a dois propósitos: caso o BD já exista, passa a ser utilizado; caso ainda não exista, é definido para criação.

Aqui, pode ser observado um comportamento peculiar do SGBD quando comparado aos seus pares: um BD passa a existir efetivamente no MongoDB quando for feita a primeira inserção de dados em uma de suas coleções (vide Quadro 1). Na figura, o comando `show databases`, invocado logo após a criação, não exibe o novo banco - isso ocorre porque nenhum dado foi inserido.

```

1: > use pep
2: < 'switched to db pep'
3: > show databases
4: < admin 287 kB
      local 1.4 GB
5: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 1 - Criação de um banco de dados chamado pep no MongoDB. Apesar de criado, o banco não aparece na lista pelo fato de ainda não ter recebido qualquer dado

Fonte: o autor.

Com o banco de dados definido e criado, documentos podem ser inseridos em **coleções (collections)**. Uma coleção de documentos corresponde a uma tabela em um banco de dados relacional, onde documentos são equivalentes a registros.

Cada documento pode ser estruturalmente diferente dos demais documentos armazenados; porém, é possível que uma coleção defina um esquema a ser respeitado pelas operações de inserção e alteração. Coleções podem ser criadas explicitamente, ou criadas a partir da inserção de um documento (vide Quadro 2) ou mais de um documento (vide Quadro 3).

A inserção simultânea de múltiplos documentos, exemplificada no Quadro 3, é conhecida como *batch insert* (inserção em lotes) ou *bulk insert* (inserção de volume), e objetiva aumentar o desempenho da entrada de dados por executar um único comando sobre um array de documentos.

```

1: > db.paciente.insertOne({ "cpf": 99579629064,
  "nome": "Marco Antônio",
  "data_nascimento": new Date("1988-02-29") })
2: < T acknowledged: true,
   insertedId: ObjectId("6292019969a7ebfb-860c9e97")
3: > show databases

4: < pep      8.19 kB
  admin    287 kB
  local    1.4 GB
5: > show collections
6: < paciente
7: > db.paciente.find()
8: < [
  { _id: ObjectId("6292019969a7ebfb860c9e97"),
    cpf: 99579629064,
    nome: 'Marco Antônio',
    data_nascimento: 1988-02-29T00:00:00.000Z }
9: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 2 - Criação de uma coleção chamada paciente no banco pep. A criação, tanto do banco quanto da coleção, é feita a partir da inserção de um primeiro documento / Fonte: o autor.

```

1: > db.paciente.insertMany(
  [
    { "cpf": 97134118037, "nome": "César Augusto",
      "data_nascimento": new Date("1985-08-31") },
    { "cpf": 73271134014, "nome": "Tito Vespasiano",
      "data_nascimento": new Date("1970-11-01") },
    { "cpf": 07982579035, "nome": "Tibério César",
      "data_nascimento": new Date("1990-03-15") }
  ]
)
2: < [
  { acknowledged: true,
    insertedIds:
      { '0': ObjectId("6292050169a7ebfb860c9e98"),
        '1': ObjectId("6292050169a7ebfb860c9e99"),
        '2': ObjectId("6292050169a7ebfb860c9e9a") } }
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 3 - Inserção de múltiplos documentos na forma de batch ou bulk insert / Fonte: o autor.

Os identificadores únicos gerados para os documentos (atributos `_id`) obedecem à especificação do tipo de dado `ObjectId`, que por sua vez integra a especificação BSON. A metodologia que guia a geração objetiva que cada valor seja universalmente único - ou seja, que não se repita em qualquer outra instância de BD, gerenciada por qualquer outra instalação de SGBD.

Para isso, os identificadores são o resultado da concatenação de partes representadas como valores hexadecimais que possuem relação com o tempo cronológico capturado no momento da geração, com o equipamento (servidor) no qual o identificador foi gerado e com o processo no Sistema Operacional que gerou o identificador, além de incluírem uma parte aleatória e outra sequencial,

esta última cujo início também é aleatório (ALGER, 2022). A composição de um identificador é detalhada na Figura 4.

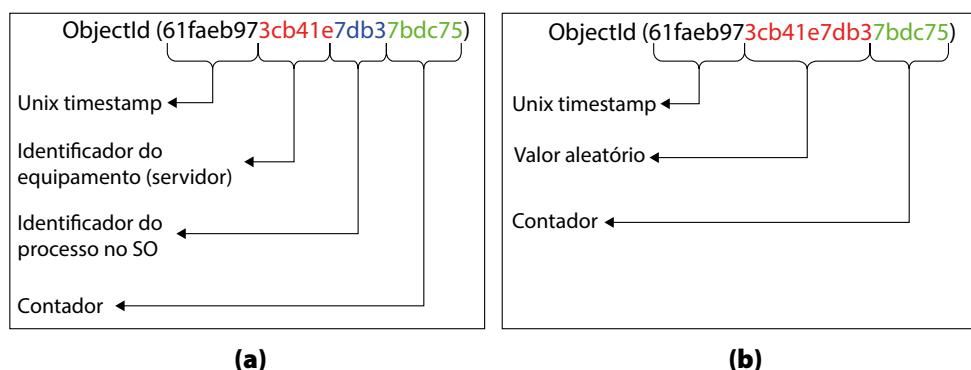


Figura 4 - Exemplos da composição de identificadores únicos. Em (a) uma versão anterior da metodologia, a qual utilizava dados do equipamento e do processo no SO onde o identificador estava sendo gerado; em (b), uma nova metodologia que substitui estes dados por um valor aleatório

Fonte: adaptado de Alger (2022).

Descrição da Imagem: na figura é apresentada à esquerda a composição de um identificador único gerado pelo MongoDB, utilizando uma versão anterior da função ObjectId. O identificador é composto por quatro partes em hexadecimal, como segue: quatro bytes com o valor do Unix timestamp do momento da geração do identificador (61faeb97); três bytes com a identificação do equipamento (servidor) onde o identificador único foi gerado (3cb41e); dois bytes com o identificador do processo no SO onde o identificador único foi gerado (7db3); e três bytes com um contador iniciado em um valor aleatório (7bdc75). À direita é apresentado o resultado da versão corrente (considerando a data de escrita deste material): quatro bytes com o valor do Unix timestamp do momento da geração do identificador (61faeb97); cinco bytes com um valor gerado aleatoriamente (3cb41e7db3); e três bytes com um contador iniciado em um valor aleatório (7bdc75).

A flexibilidade permitida pelo MongoDB na definição da estrutura dos documentos armazenados resolve, por exemplo, as questões de múltipla ocorrência e aninhamento de subdocumentos. Enquanto no modelo relacional a múltipla ocorrência de um atributo implica em normalização (ver Figura 2), em JSON é permitida a definição de um array para o armazenamento dos múltiplos valores (vide o atributo `telefone_completo` no Quadro 4).

O aninhamento de subdocumentos, por sua vez, que no modelo relacional implica na reestruturação da tabela ou tabelas atingidas pela demanda com a inclusão de novos campos, em um documento se resume à inclusão de um novo objeto aninhado à estrutura do documento original (vide o atributo `telefone` no Quadro 5).

```

1: > db.paciente.insertOne(
   {
      "cpf": 23326802099,
      "nome": "Caio César",
      "data_nascimento": new Date("2001-01-02"),
      "telefone_completo": [
         "(48)-98765-4321",
         "(49) 1234-5678"
      ]
   }
)
2: < { acknowledged: true,
       insertedId: ObjectId("6292077469a7ebfb860c9e9b")
}
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 4 - Exemplo da flexibilização na estrutura de documentos permitida pelo MongoDB, com a inserção de um atributo com múltipla ocorrência / Fonte: o autor.

```

1: > db.paciente.insertOne(
   {
      "cpf": 82822285080,
      "nome": "Nero Cláudio",
      "data_nascimento": new Date("1999-05-31"),
      "telefone": [
         {
            "ddd": 47,
            "numero": "67890-1234"
         }
      ]
   }
)
2: < { acknowledged: true,
       insertedId: ObjectId("629208e869a7ebfb860c9e9c")
}
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 5 - Exemplo da flexibilização na estrutura de documentos permitida pelo MongoDB, com a inserção de um subdocumento aninhado ao documento principal / Fonte: o autor.

A execução de consultas para a recuperação de documentos permite a construção de queries na forma de documentos JSON, onde são especificados os atributos a serem utilizados como parte do critério de busca e/ou como parte do resultado a ser retornado, além dos operadores que serão utilizados para a comparação dos valores dos atributos com os valores de interesse.

No Quadro 6, uma *query* é construída com o objetivo de recuperar todos os pacientes cujo nome possua “César” como parte, retornando o identificador do documento e o nome completo do paciente. No Quadro 7, por sua vez, a *query* especifica que devem ser retornados todos os pacientes cujo atributo ddd do telefone seja 47, retornando o identificador do documento, o nome completo do paciente e o telefone (com ddd e número).

```

1: > db.paciente.find(
   { "nome": /César/ },
   { "nome": 1 }
)
2: < { _id: ObjectId("6292050169a7ebfb860c9e98") ,
      nome: 'César Augusto' ,
      _id: ObjectId("6292050169a7ebfb860c9e9a") ,
      nome: 'Tibério César' ,
      _id: ObjectId("6292077469a7ebfb860c9e9b") ,
      nome: 'Caio César' }
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

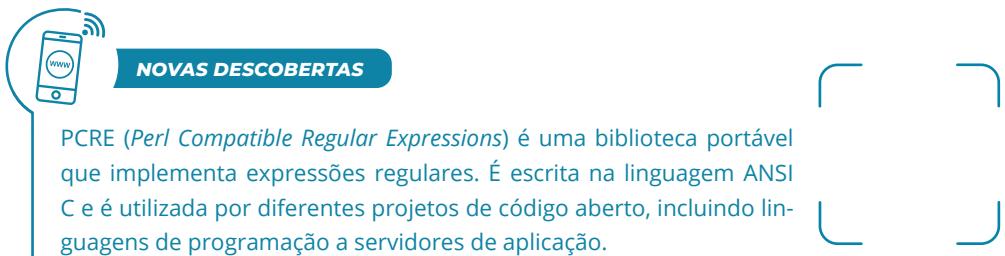
Quadro 6 - Exemplo de execução de *query* utilizando uma expressão PCRE (Perl Compatible Regular Expression) para a aplicação de um critério de busca por parte do nome do paciente / Fonte: o autor.

```

1: > db.paciente.find(
   { "telefone.ddd": 47 },
   { "nome": 1, "telefone": 1 }
)
2: < { _id: ObjectId("629208e869a7ebfb860c9e9c") ,
      nome: 'Nero Cláudio' ,
      telefone: [ { ddd: 47, numero: '67890-1234' } ] }
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 7 - Exemplo de execução de *query* explorando a hierarquia dos atributos nos documentos armazenados (*ddd* como parte de *telefone*) para a especificação do critério de seleção / Fonte: o autor.

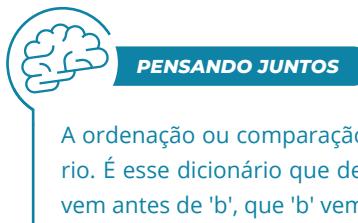


Na área de banco de dados existe um *trade-off* (perde-se de um lado, ganha-se de outro) entre **velocidade de armazenamento** e **velocidade de consulta**, e esse perde-ganha tem a ver com a **ordenação** dos dados envolvidos. As maiores velocidades de pesquisa e recuperação de dados em BD ocorrem quando esses dados estão ordenados (de forma crescente ou decrescente, numérica ou lexicográfica).

Em uma primeira avaliação, o problema seria resolvido se houvesse a possibilidade de se manter o conjunto de dados completo ordenado após cada inserção ou alteração de um de seus itens. Essa abordagem seria efetiva caso o conjunto de dados estivesse disponível na íntegra em memória principal, porém, devido

ao seu volume, comumente o armazenamento desses dados ocorre em memória secundária (disco e arquivos), muito mais lentos do que a memória principal, o que torna o custo de uma ordenação recorrente proibitivo.

Assume-se, então, que as inserções e as alterações não implicam em ordenação, sendo que as primeiras ocorrem ao final dos arquivos afetados e as últimas mantêm os dados no seu local de armazenamento de origem. Com isso, obtém-se o melhor desempenho possível para as escritas, sacrificando o desempenho das leituras.



A ordenação ou comparação lexicográfica é aquela executada com base em um dicionário. É esse dicionário que define a ordem de seus elementos, como por exemplo que 'a' vem antes de 'b', que 'b' vem antes de 'c' e assim por diante.

Considerando que, teoricamente, inserções e alterações de dados ocorrem menos do que consultas a esses mesmos dados, parece ser um contrassenso priorizar o desempenho de escrita em detrimento do desempenho de leitura. Por exemplo: em um PEP, o cadastro de um paciente é feito, de forma ideal, apenas 1 (uma) vez; esporadicamente, esse cadastro pode ser modificado (o paciente mudou de nome, ou o paciente mudou de endereço).

Porém, esse mesmo cadastro pode ser utilizado em inúmeros relatórios e aparecer como resultado de várias pesquisas feitas por diferentes sistemas de informação. São necessárias, então, alternativas para que as operações de inserção e alteração continuem sendo executadas com o máximo desempenho possível, e que operações de consulta sejam aceleradas para atender a demandas de rotina ou pontuais geradas a partir das necessidades dos serviços que consomem os dados. Essas alternativas existem, e são conhecidas como índices do banco de dados.

Índices são estruturas acessórias em um banco de dados utilizadas para acelerar operações de consulta. Essas estruturas são implementadas utilizando algoritmos e estruturas de dados (como árvores B e **tabelas Hash**, por exemplo), são mantidas atualizadas automaticamente pelo SGBD e são responsáveis por ordenar parte dos dados do banco para que consultas possam ser otimizadas.

Por serem opcionais, os índices podem ser criados, redefinidos e/ou apagados sob demanda sem prejuízo aos dados armazenados. Comumente, a escolha

e a implementação dos índices em um BD observa as consultas que apresentam maiores tempos de resposta ou que são executadas com uma maior frequência. A partir dessas consultas são identificados os atributos (campos) utilizados como critérios de seleção, que passam a ser candidatos à indexação.

No MongoDB, um índice é criado a partir da definição de um documento JSON onde são especificados o campo (ou campos) a serem indexados e as opções que definem o comportamento do índice (vide Quadros 8, 9 e 10). O identificador único dos documentos (atributo `_id`) é o único atributo indexado automaticamente pelo SGBD.

```

1: > db.paciente.createIndex(
2:   {
3:     cpf: 1
4:   }
5: )
6: < 'cpf_1'
7: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 8 - Exemplo de criação de índice, especificando que o atributo `cpf` deverá ser indexado em ordem crescente. O retorno `cpf_1` é o nome atribuído ao índice pelo SGBD / Fonte: o autor.

```

1: > db.paciente.createIndex(
2:   {
3:     "data_nascimento": -1,
4:     "telefone.ddd": 1, "telefone.numero": 1
5:   },
6:   {
7:     "sparse": true
8:   }
9: )
10: < 'data_nascimento_-1_telefone.ddd_1_telefone.numero_1'
11: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 9 - Exemplo de criação de índice para os atributos `data_nascimento` (com ordenação decrescente), `telefone.ddd` e `telefone.numero` ambos em ordem crescente. Todos os índices são esparsos - ou seja, neles são incluídos apenas os documentos que possuem os atributos especificados / Fonte: o autor.

```

1: > db.paciente.getIndexes()
2: < [
      {
        "v": 2, "key": { "id": 1 }, "name": "id",
        "v": 2, "key": { "cpf": 1 }, "name": "cpf_1",
        "v": 2, "key": { "data_nascimento": -1, "telefone.ddd": 1, "telefone.numero": 1 }, "name": "data_nascimento_-1_telefone.ddd_1_telefone.numero_1", "sparse": true
      }
    ]
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 10 - Recuperação de dados referentes aos índices criados na coleção paciente. Para cada índice é retornado um documento JSON com atributos que detalham a versão (v) de implementação do índice, a chave (key) que contempla os atributos indexados, o nome (name) do índice e outras opções de configuração utilizadas na sua criação (por exemplo, sparse) / Fonte: o autor.

Apesar de serem relacionados diretamente a melhorias no desempenho de consultas (queries), os índices também desempenham papel fundamental na aceleração das operações de alteração e exclusão. Comumente essas operações são realizadas em um único documento ou em um subconjunto de documentos de uma coleção. Antes de serem modificados ou excluídos, esses documentos precisam ser localizados – o que equivale a executar uma consulta utilizando os critérios de seleção especificados.

Se os atributos que atenderem aos critérios estiverem indexados, a consulta é acelerada e a alteração/exclusão pode ser efetivada em menos tempo. Os Quadros 11, 12, 13 e 14 exemplificam tanto alterações quanto exclusões, ambas utilizando atributos indexados previamente em exemplos anteriores.



```

1: > db.paciente.updateOne(
   {
      "_id": ObjectId("6292050169a7ebfb860c9e99")
   },
   {
      $set:
      {
         "nome": "Tito César Vespasiano Augusto"
      }
   }
)
2: < { acknowledged: true,
       insertedId: null,
       matchedCount: 1,
       modifiedCount: 1,
       upsertedCount: 0 }
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 11 - Alteração de dados em documentos da coleção paciente. O SGBD pode optar por utilizar o índice criado automaticamente para o atributo `_id` visando acelerar a busca pelo documento e, em seguida, proceder com a alteração / Fonte: o autor.

```

1: > db.paciente.updateOne(
   {
      "cpf": 7982579035
   },
   {
      $set:
      {
         "nome": "Tibério César Augusto",
         "telefone": "(45) 6789-0123"
      }
   }
)
2: < { acknowledged: true,
       insertedId: null,
       matchedCount: 1,
       modifiedCount: 1,
       upsertedCount: 0 }
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 12 - Alteração de dados em documentos da coleção paciente, utilizando o atributo `cpf` (também indexado) como critério de busca / Fonte: o autor.

```

1: > db.paciente.deleteOne(
   {
      "_id": ObjectId("6292050169a7ebfb860c9e9a")
   }
)
2: < { acknowledged: true, deletedCount: 1 }
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

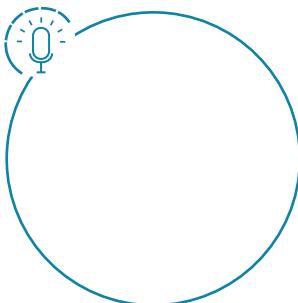
Quadro 13 - Exclusão de documentos da coleção paciente. O SGBD pode optar por utilizar o índice criado automaticamente para o atributo `_id` visando acelerar a busca pelo documento e, em seguida, proceder com a exclusão. / Fonte: o autor.

```

1: > db.paciente.deleteOne (
    {
        "cpf": 99579629064
    }
)
2: < { acknowledged: true, deletedCount: 1 }
3: Atlas atlas-ayr8k6-shard-0 [primary] pep>

```

Quadro 14 - Exclusão de documentos da coleção paciente, utilizando o atributo cpf (também indexado) como critério de busca / Fonte: o autor.



A escolha entre um SGBD relacional e um SGBD orientado a documentos pode ser motivada pelos detalhes de implementação e particularidades de cada ferramenta. Acompanhe em nosso *podcast* os comentários feitos sobre as semelhanças e diferenças entre os dois modelos de dados suportados por esses SGBDs, além de um apanhado de pontos positivos e negativos decorrentes da adoção de cada um. Aproveite!

NOVAS DESCOPERTAS

openEHR é uma tecnologia composta por software, modelos clínicos e especificações para a construção de padrões e soluções de interoperabilidade na área da saúde.



Os experimentos executados pelas equipes de Samantha e Gaspar ratificaram a opinião de boa parte dos envolvidos de que bancos de dados orientados a documentos são alternativas viáveis como *backend* de dados para sistemas PEP. Os exemplos de inserção, alteração e exclusão de documentos, bem como consultas (queries) e a definição de índices esclareceram aos participantes como se dá a comunicação de um software cliente (no caso, o Mongo Shell) para com o SGBD.

A estrutura de dados de facto que viabiliza essa comunicação é o documento JSON, utilizado tanto para encapsular os dados de interesse (nos exemplos, dados do cadastro de diferentes pacientes) quanto para encapsular parâmetros e configurações a serem utilizadas pelos comandos.

Outro fator que favorece a adoção dessa categoria de ferramenta é a aderência total a padrões de representação, gerenciamento, armazenamento, recuperação e troca de dados entre sistemas na área de saúde. Os padrões **HL7® FHIR®** (*Health Level Seven Fast Healthcare Interoperability Resources*) referenciado no Módulo 1 e **openEHR** definem resources (recursos) e archetypes (arquétipos) para organizar e estruturar os dados de interesse, cujas instâncias criadas a partir desses dados são apresentadas na forma de documentos XML ou JSON. Isso possibilita a criação de bancos de dados a partir dos documentos originais, sem que haja necessidade de um mapeamento para um modelo de dados diferente (como o modelo relacional, por exemplo).





1. Com relação aos conceitos de *workflow*; PEP, RES e S-RES, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
 - a) Um *workflow* automatiza e organiza procedimentos e tarefas através da transmissão desregrada de documentos sigilosos entre seus participantes e terceiros.
 - b) Por ter foco na transmissão de documentos entre pares, um *workflow* objetiva resolver gargalos ou mesmo reduzir/eliminar desperdícios nas organizações que o adotam.
 - c) PEPs são a representação digital dos prontuários de paciente, criados e armazenados exclusivamente em papel, é que comumente são utilizados no escopo local de um estabelecimento de saúde.
 - d) RES e S-RES são concorrentes dos PEPs por introduzirem recursos de interoperabilidade, o que impede o acesso aos dados dos pacientes que tenham sido gerados em múltiplos estabelecimentos de saúde.
 - e) PEPs, RES e S-RES organizam seu conteúdo na forma de pastas, o que dificulta a adoção de SGBDs orientados a documentos para a sua manutenção em detrimento dos SGBDs relacionais.
2. Relembrando as estruturas básicas que compõem o modelo de dados relacional, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
 - a) Os dados armazenados em uma mesma tupla não precisam ter, necessariamente, uma relação entre si; esses dados não representam qualquer elemento do cenário mapeado pelo BD.
 - b) Os domínios permitem especificar múltiplos valores (ou valores não atômicos) para os atributos de uma relação.
 - c) Uma relação corresponde a um conjunto de tuplas com esquema livre, onde cada elemento do conjunto pode ter um número diferente de atributos com tipos de dados e domínios também diferentes.
 - d) Os atributos são as estruturas básicas para a composição de uma relação, caracterizados por um domínio e organizados em uma ordem que define o esquema de uma relação.
 - e) De todas as estruturas básicas apresentadas, apenas o domínio é relevante para a composição de um esquema.



3. Com base nas características dos bancos de dados orientados a documentos, é CORRETO afirmar que:
- esses bancos adaptam-se melhor a cenários de escalabilidade vertical por serem sistemas monolíticos;
 - a adaptação dos esquemas de dados para esses bancos demanda, obrigatoriamente, uma janela de manutenção com *downtime*;
 - os esquemas passíveis de uso seguem a rigidez do modelo de dados relacional, exigindo que todos os documentos armazenados possuam os mesmos atributos;
 - um dicionário de dados é obrigatório para cada coleção de documentos;
 - os documentos cujo conteúdo é organizado nos formatos JSON, YAML e XML são passíveis de armazenamento.
4. Os comandos `db.<coleção>.insertMany`, `use`, `db.<coleção>.find`, `db.<coleção>.insertOne`, `db.<coleção>.getIndexes` e `db.<coleção>.createIndex` executam, respectivamente, as seguintes ações no MongoDB:
- inserção de múltiplos documentos em uma coleção, seleção/criação de um novo BD, execução de *query* (pesquisa) em uma coleção, inserção de um único documento em uma coleção, recuperação de dados dos índices de uma coleção e criação de um índice em uma coleção;
 - execução de *query* (pesquisa) em uma coleção, seleção/criação de um novo BD, inserção de um único documento em uma coleção, recuperação de dados dos índices de uma coleção e criação de um índice em uma coleção, inserção de múltiplos documentos em uma coleção;
 - inserção de um único documento em uma coleção, seleção/criação de um novo BD, execução de *query* (pesquisa) em uma coleção, recuperação de dados dos índices de uma coleção e criação de um índice em uma coleção, inserção de múltiplos documentos em uma coleção;
 - recuperação de dados dos índices de uma coleção, criação de um índice em uma coleção, inserção de múltiplos documentos em uma coleção, seleção/criação de um novo BD, execução de *query* (pesquisa) em uma coleção, inserção de um único documento em uma coleção;
 - inserção de múltiplos documentos em uma coleção, seleção/criação de um novo BD, execução de *query* (pesquisa) em uma coleção, recuperação de dados dos índices de uma coleção e criação de um índice em uma coleção, inserção de um único documento em uma coleção.

MEU ESPAÇO



MEU ESPAÇO



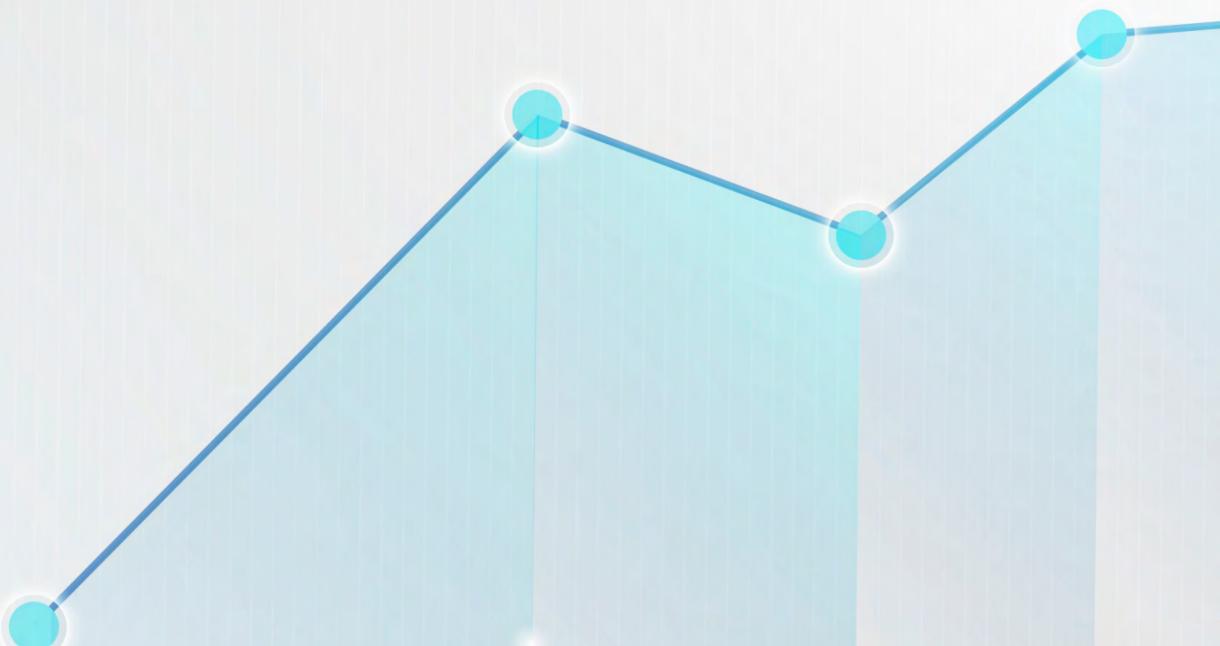
Bancos de Dados Para Séries Temporais Orientados a Chave-Valor

Dr. Alexandre Savaris

OPORTUNIDADES DE APRENDIZAGEM

Nesta unidade, o conceito de condição de saúde é apresentado de acordo com a sua orientação no tempo, que define condições agudas e condições crônicas. Para sua representação, é introduzido o conceito de série temporal, juntamente com os fundamentos dos Sistemas Gerenciadores de Bancos de Dados orientados a esse tipo de série. Como ferramenta para exemplificação e experimentação será utilizado o *InfluxDB*, cuja versão de testes gratuita em nuvem oferece recursos para a ingestão de dados, a construção de *dashboards* e a definição e configuração de alertas a partir dos dados armazenados.

As atividades de atenção à saúde executadas na esfera pública ou na esfera privada, utilizam o conceito e são guiadas pela **condição de saúde** da população. A condição de saúde caracteriza as circunstâncias – fatos, pormenores, particularidades – da saúde de um indivíduo, as quais implicam em demandas de respostas reativas ou proativas, pontuais ou contínuas, integradas ou fragmentadas, a serem dadas pelo sistema de saúde e pelos profissionais de saúde. Além de doenças, fazem parte dessas circunstâncias as condições fisiológicas (como uma gravidez, por exemplo), os eventos isolados (como as exposições a algum agente tóxico) e também os eventos cíclicos e duradouros que demandam acompanhamento de crianças, de adolescentes, de adultos e de pessoas idosas durante períodos de tempo estendidos (MENDES, 2022).





O tempo e a forma de enfrentamento são variáveis consideradas no atendimento a condições de saúde, e permitem classificá-las em duas categorias: **condições agudas** e **condições crônicas**. As condições de saúde agudas são caracterizadas por terem uma duração curta (até três meses), período no qual a doença se manifesta ou os eventos ocorrem, o diagnóstico (comumente facilitado) é fechado, o tratamento é realizado e um resultado é obtido. As condições de saúde crônicas, por sua vez, desenvolvem-se por períodos superiores a três meses (podendo, inclusive, serem vitalícias), e costumam ter uma evolução lenta, possuem diagnósticos mais difíceis de fechar e tratamentos de longo prazo que muitas vezes não visam a cura, mas a melhoria na qualidade de vida do indivíduo. A presença prolongada dessas circunstâncias costuma impactar as relações familiares do indivíduo e incluir fatores culturais, religiosos, sociais e psicológicos no processo de tratamento e acompanhamento (FREITAS; MENDES, 1999; MENDES, 2022).

Tanto as condições agudas quanto as condições crônicas demandam acompanhamento, o qual é feito de forma periódica com repetições estipuladas pelo diagnóstico e tratamento realizado. Todas as observações feitas e os dados coletados durante o acompanhamento possuem uma marca temporal – um instante no tempo, composto por uma data e uma hora com precisão de segundos, milissegundos, microssegundos ou mesmo nanosegundos – que auxiliam na composição de um histórico ordenado e que caracterizam uma **série temporal**. *Datasets* com essas características permitem análises diferenciadas, que para serem executadas adequadamente demandam a adoção de tecnologias que suportem o registro do tempo na escala desejada e sejam capazes de operar com grandes volumes de dados oriundos de aquisições em frequências elevadas.

No contexto do projeto a ser executado por Luísa e sua equipe, as séries temporais geradas durante o acompanhamento de casos agudos e crônicos irão demandar a escolha de tecnologias e a sua adaptação a técnicas para a correta aquisição, manutenção e processamento. Caberá à equipe de TI, com a sua ajuda, a pesquisa e a proposta de implementação dessas tecnologias visando atender aos requisitos locais (de cada hospital) e globais (da rede como um todo).

O acompanhamento feito a condições agudas e crônicas demanda monitoração, cuja execução é definida e organizada de acordo com a gravidade dos casos e o local de permanência dos indivíduos monitorados. Por serem caracterizadas pelo curto prazo e intensidade, as condições de saúde agudas costumam ser monitoradas nos estabelecimentos procurados no momento da ocorrência dos eventos; a gravidade desses eventos, por sua vez, é fator determinante do nível de cuidado a ser despendido (como o uso de Unidades de Tratamento Intensivo (UTIs), por exemplo), na especificação dos sinais vitais a serem acompanhados durante o tempo necessário à evolução do quadro da condição e na definição de alertas a serem disparados para sinalizar alterações importantes ao pessoal técnico qualificado (DAVOUDI *et al.*, 2019; COBUS; HEUTEN, 2019). As condições de saúde crônicas, por sua vez, caracterizadas em parte pelo longo prazo da sua evolução, dispensam a monitoração in loco feita nos estabelecimentos de saúde e permitem que os indivíduos utilizem de recursos pervasivos como dispositivos vestíveis (wearables) potencializados por recursos de IoT (*Internet of Things - Internet das Coisas*). Combinando processamento local e capacidade de comunicação, esses recursos permitem que o indivíduo seja monitorado remotamente com impacto mínimo à sua rotina (MALASINGHE; RAMZAN; DAHAL, 2019).

A rede que será base para a implantação do Hospital sem Papel (Paperless Hospital) já acompanha pacientes em regime de UTI, disponibilizando recursos para a monitoração de condições agudas em ambientes controlados. Com a parceria dos hospitais participantes junto ao Sistema Único de Saúde do Brasil (SUS), essa monitoração será estendida também às condições crônicas por meio de projetos viabilizados pelo uso de Telemedicina. Neles, os pacientes serão acompanhados remotamente com a monitoração de seus sinais vitais e marcadóres sendo feita por dispositivos especializados, selecionados de acordo com as particularidades de cada condição e com as especificidades físicas de cada indivíduo. Os dispositivos utilizados estarão conectados ao hospital selecionado para o acompanhamento, e enviarão periodicamente os dados adquiridos para consolidação e avaliação pelos profissionais de saúde disponíveis. Caberá à equipe de TI disponibilizar e manter a infraestrutura necessária para que o cenário proposto seja implantado e seja sustentável. Em particular, caberá à Luísa, Gaspar, Laura, Samanta e Marco definirem uma arquitetura que seja escalável – considerando que com o passar do tempo novos hospitais poderão oferecer o serviço de acompanhamento e novas especialidades poderão ser atendidas – além de selecionar tecnologias viáveis para o armazenamento do grande volume de dados esperado que será enviado pelos dispositivos de monitoração.

As características do processo de monitoração das condições de saúde dos indivíduos atendidos pelos hospitais da rede orientam, tecnicamente, a adoção de séries temporais para a organização, a representação e o armazenamento dos dados coletados, além de direcionar a pesquisa, a seleção, a instalação e a configuração de ferramentas de *Software* que suportem essa organização. De forma complementar, tão importante quanto viabilizar a aquisição e o armazenamento desses dados, é utilizá-los efetivamente no dia a dia do cuidado dispensado aos pacientes e na gestão dos estabelecimentos envolvidos. Para isso, conhecer e aplicar metodologias e técnicas de análise nessas séries é fundamental. Você pode colaborar com esse conhecimento (que será utilizado por Luísa para justificar o tempo e o esforço investidos no estudo e na adoção das tecnologias relacionadas) por meio de uma pesquisa feita na internet, com o objetivo de identificar essas metodologias e técnicas explicando brevemente o seu funcionamento. O trabalho pode ser iniciado pelo modelo ARIMA (*AutoRegressive Integrated Moving Average* - Modelo Auto-Regressivo Integrado)

de Médias Móveis) aplicado a dados de saúde (SATO, 2013; LEITE, 2021) e estendido a outros modelos que o substituam ou o complementam.

Durante o processo de modelagem conceitual, modelagem lógica e projeto físico de um banco de dados – seja ele relacional ou não relacional – a existência de uma ou mais de uma marca temporal é praticamente uma constante. Cenários em que se observa pelo menos uma data, um horário ou mesmo uma combinação de ambos (*timestamp*) são prevalentes, porém, é comum que essas marcas temporais sejam categorizadas como atributos comuns ou atributos convencionais, recebendo a mesma atenção que os demais atributos encontrados no domínio do problema modelado (por exemplo: nomes, endereços e números de telefone). Em se tratando de séries temporais, o entendimento é diametralmente oposto: as marcas temporais são obrigatórias, além de nortear todo o processo de organização e análise dos dados disponíveis.

Em seu diário de bordo, registre o seu entendimento e as suas opiniões sobre a promoção de datas e horas a atributos fundamentais na composição de datasets. Posteriormente, revise esse registro com base no que será apresentado no restante deste módulo. Este conteúdo poderá ser utilizado pela equipe de TI como parte do fundamento para a integração de séries temporais e suas tecnologias correlatas no ecossistema do hospital sem papel!

DIÁRIO DE BORDO

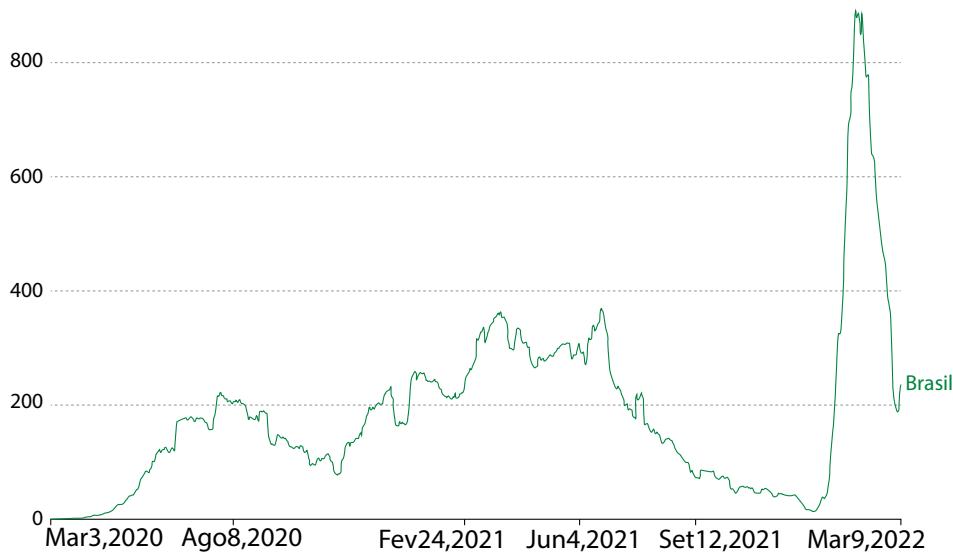


A adoção de séries temporais no projeto do Hospital sem Papel tirou a equipe de sua zona de conforto (em especial Samantha e Marco, responsáveis respectivamente pela área de DevOps e administração de banco de dados). *Datasets* com essas características levam a necessidades de infraestrutura e administração a outro patamar, dada a alta frequência de aquisição dos dados e a necessidade de organizá-los de forma que seja possível a sua utilização em tempo real para a composição de alertas, e posteriormente como subsídio para a geração de *dashboards* tanto em nível operacional quanto em nível gerencial. O entendimento de como essas séries devem ser construídas e a experimentação com ferramentas de *software* que as suportem, então, foram elencadas como as atividades da vez.

Séries temporais (*time series*) podem ser definidas como conjuntos ou coleções de observações feitas especificamente sobre um dado bem definido, de forma repetitiva no decorrer do tempo. Essas séries assumem características contínuas quando utilizadas na observação de fenômenos naturais (como temperatura e pressão), e discretas quando estabelecem uma frequência ou intervalo de tempo específico para a aquisição dos seus dados (MORETTIN; TOLOI, 2018; TIME, 2022). Durante a pandemia de COVID-19, a exibição de gráficos para a visualização dessas séries (vide Figura 1) passou a fazer parte da rotina cotidiana da população em geral, que os utilizou como ferramenta para acompanhar a evolução no número de casos (suspeitos e confirmados), das internações, das curas, das mortes e, mais recentemente, da evolução na aplicação de vacinas.

Novos casos diários confirmados de COVID-19 por milhão de pessoas

Média móvel de 7 dias. Devido às limitações de testagem, o número de casos confirmados é menor que o número real de infecções



Fonte: Johns Hopkins University CSSE COVID-19 Data

Figura 1 - Gráfico de linha exibindo uma série temporal relacionada ao número de casos diários confirmados de COVID-19 por milhão de habitantes no Brasil, de 03/03/2020 a 09/03/2022
 Fonte: GCDL ([2022], on-line).

Descrição da Imagem: no topo da figura é exibido o título "Novos casos diários confirmados de COVID-19 por milhão de pessoas", seguido pelo subtítulo "Média móvel de 7 dias. Devido às limitações de testagem, o número de casos confirmados é menor que o número real de infecções." A figura exibe um gráfico de linha onde no eixo y (vertical) são apresentados números de 0 a 800, separados de 200 em 200, e no eixo x (horizontal) são apresentadas datas de referência (03/03/2020, 08/08/2020, 24/02/2021, 04/06/2021, 12/09/2021 e 09/03/2022). Na base da figura é exibida a referência "Fonte: Johns Hopkins University CSSE COVID-19 Data".

À primeira vista, uma série temporal parece se resumir a uma combinação de tempo + valor que pode ser plotada em um gráfico como o da Figura 1. A análise dessas séries, porém, permite a identificação de **padrões** por meio de processos de **decomposição**. Conforme Hyndman e Athanasopoulos (2018), três padrões merecem destaque:

- **Tendência (Trend)** - comportamento de longo prazo que pode vir a ser observado em uma série temporal onde os dados aumentam ou diminuem indicando a manutenção ou a mudança de direção (de valores maiores para menores, ou vice-versa).
- **Sazonalidade (Seasonal)** - comportamento observado em séries temporais afetadas por fatores sazonais que ocorrem de forma fixa e conhecida (diariamente, semanalmente ou anualmente, por exemplo).
- **Ciclo (Cyclic)** - comportamento de médio/longo prazo (por exemplo, dois anos) exibindo uma flutuação de altos e baixos nos valores da série que ocorre sem uma frequência definida. É a indefinição dessa frequência que difere, essencialmente, um comportamento cíclico de um comportamento sazonal.

Os padrões citados podem ser obtidos com facilidade utilizando bibliotecas de *Software* para a análise de dados, como aquelas disponibilizadas pela linguagem de programação Python. Comumente, os dados analisados são provenientes de arquivos CSV (*Comma-Separated Values* - Valores Separados por Vírgula), utilizados para a decomposição da série em padrões e na exibição desses padrões em gráficos (vide Figura 2).

NOVAS DESCOPERTAS

CSV (*Comma-Separated Values*) é um formato de arquivo de texto caracterizado pela organização em linhas e pela separação por vírgula dos dados que compõem cada linha. Variantes permitem a adoção de outros separadores e de caracteres delimitadores de conteúdo.

Python é uma linguagem de programação multiparadigma (orientada a objetos, imperativa, funcional, procedural) criada em 1991 e usada extensivamente na área de ciência de dados graças à disponibilidade de um grande número de bibliotecas voltadas a esse fim.

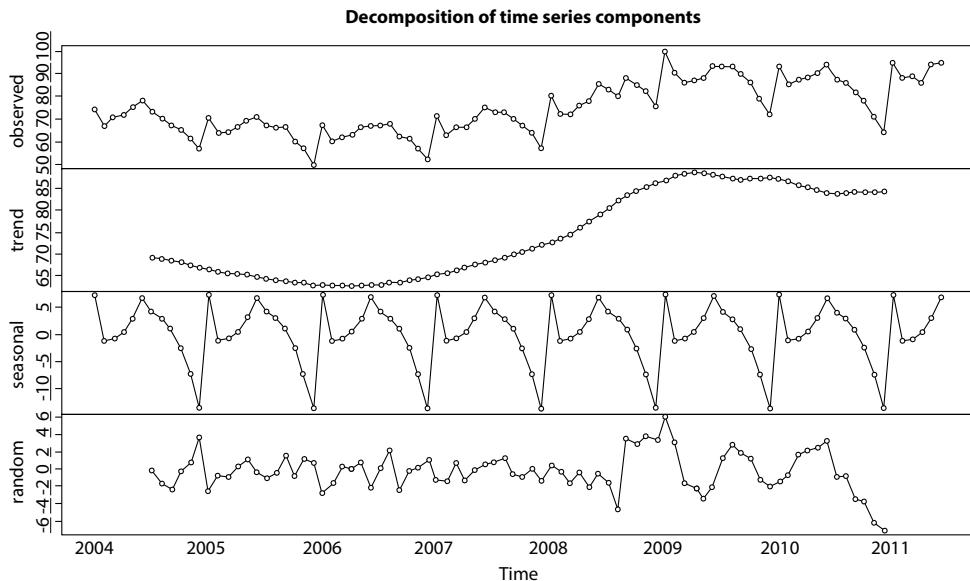


Figura 2 - Gráficos resultantes da decomposição de uma série temporal. De cima para baixo: 1) o conteúdo do dataset original; 2) a combinação dos padrões de tendência e de ciclo, operação comumente executada na análise desse tipo de série; 3) a sazonalidade observada na série; 4) o conteúdo residual após a subtração da tendência, ciclo e sazonalidade dos dados originais
 Fonte: adaptada de Jebb et al. (2015).

Descrição da Imagem: no topo da figura é exibido o título “Decomposition of Time Series Components”, seguido por um gráfico de linha para o conteúdo do dataset original, utilizando como legenda do eixo y (vertical) os valores 50, 60, 70, 80, 90 e 100, o título “observed” e como legenda do eixo x (horizontal) os valores 2004, 2005, 2006, 2007, 2008, 2009, 2010 e 2011 (com título “Time”). A legenda do eixo x (horizontal) é a mesma para todos os gráficos. Logo abaixo é exibido o gráfico de linha para a combinação dos padrões de tendência e de ciclo; o gráfico é intitulado “trend” e utiliza como legenda do eixo y (vertical) os valores 65, 70, 75, 80 e 85. A seguir é exibido o gráfico de linha para a sazonalidade; o gráfico é intitulado “seasonal” e utiliza como legenda do eixo y (vertical) os valores -10, -5, 0 e 5. Por fim é exibido o gráfico de linhas para o conteúdo residual; o gráfico é intitulado “random” e utiliza como legenda do eixo y (vertical) os valores -6, -4, -2, 0, 2, 4 e 6.

Com os conceitos básicos sobre séries temporais discutidos e entendidos pela equipe, Samantha e Marco passaram a direcionar os esforços de todos à experimentação. Para tanto, solicitaram que os envolvidos pesquisassem e apresentassem sugestões de SGBDs capazes de: 1) suportar datasets com características similares àquelas que serão encontradas nos hospitais da rede; 2) possam ser utilizados preferencialmente em nuvem, em um primeiro momento para testes de funcionalidades; 3) possuam APIs flexíveis para entrada e saída de dados, além

de uma GUI que permita explorar o conteúdo dos datasets e criar *dashboards* para análise em tempo quase-real. Por unanimidade, os participantes indicaram o **InfluxDB** como o SGBD a ser utilizado nos experimentos e, eventualmente, a ser adotado como solução componente do ecossistema do Hospital sem Papel.

NOVAS DESCOBERTAS

API (*Application Programming Interface*) corresponde a um conjunto de procedimentos e/ou funções, devidamente padronizados e documentados, que um programa de computador disponibiliza para que outros programas de computador possam utilizar para comunicar consigo.

GUI (*Graphical User Interface*) é uma interface utilizada por um usuário para interagir com dispositivos eletrônicos e/ou programas de computador por meio de elementos gráficos (rótulos, campos de texto, botões), substituindo as interações textuais via terminal de comando.

InfluxDB é um SGBD orientado a séries temporais (também caracterizado como um **Banco de Dados de Séries Temporais - Time Series DataBase - TSDB**) lançado em 2013 com o objetivo de gerenciar volumes massivos de dados orientados no tempo provenientes de sensores e dispositivos de IoT, sistemas de informação em geral, containers em ambientes de DevOps, Máquinas Virtuais (*Virtual Machines - VMs*), e redes de computadores. O SGBD integra uma Stack de soluções conhecida como TICK (Telegraf, *InfluxDB*, Chronograf e Kapacitor), com cada solução sendo responsável, respectivamente, pela coleta de métricas e envio ao SGBD, armazenamento e organização de dados, GUI para construção de gráficos e *dashboards*, e um engine para processamento de eventos e criação de alertas. Escrito em linguagem Go, o SGBD é diferenciado de outras soluções por oferecer uma ampla gama de bibliotecas e plugins para facilitar a integração com as mais diferentes linguagens de programação e fontes de dados, o que torna a sua inclusão como componente de um ecossistema uma tarefa de minutos (INFLUXDB, 2022b).



NOVAS DESCOBERTAS

Para fins de experimentação e testes, a versão em nuvem do InfluxDB pode ser acessada gratuitamente. Conhecida como InfluxDB Cloud, a versão gratuita oferece um ambiente de cluster limitado, porém suficiente para o aprendizado da ferramenta.

Go é uma linguagem de programação criada pelo Google em 2007 para atender a demandas internas de engenharia. Liberada como open source ao público em 2009, a linguagem é focada em produtividade e programação concorrente, e conquista adeptos pela facilidade de uso.



PENSANDO JUNTOS

As siglas SGBD e BD costumam ser utilizadas de forma intercambiada como referência para a mesma coisa, porém, há diferenças importantes de significado. Enquanto a primeira diz respeito aos *Softwares* utilizados para gerenciar e organizar o acesso aos dados disponíveis, a segunda é voltada especificamente a estruturas de dados criadas para armazenamento, juntamente com o seu conteúdo.

Internamente, os dados armazenados pelo InfluxDB são organizados combinando os seguintes elementos (INFLUXDB, 2022a):

- ***timestamp*** - armazena a marca temporal para cada dado inserido no BD. Clientes conectados ao SGBD podem especificar a precisão dos dados enviados (em segundos, em milissegundos, em microsegundos, ou em nanosegundos); porém, a precisão de armazenamento é sempre em nanosegundos. Por exemplo: a data/hora 13/03/2022 13:33:30 corresponde ao *timestamp* 1647178410000, assumindo a precisão de milissegundos.
- ***measurement (medida)*** - corresponde à especificação ou descrição dos dados armazenados. Por exemplo: uma medida chamada *uti* indica que os dados coletados e armazenados têm relação com o tratamento intensivo de pacientes em unidades especializadas.

- **fields (campos)** - permitem especificar os valores armazenados junto a um timestamp e a uma medida. Cada campo possui uma chave (*key*), um valor (*value*) e pertence a um conjunto de campos (*field set*). Por exemplo: HR=71, SpO2=93 indica a existência de um campo cuja chave é HR e cujo valor é 71, e de outro campo cuja chave é SpO2 e cujo valor é 93. Juntos, os campos formam um conjunto.
- **tags (etiquetas ou rótulos)** - permitem adicionar metadados à combinação de timestamp, medida e campos de forma a aumentar a especificidade dos dados armazenados, bem como propiciar a aplicação de filtros para buscas. Cada etiqueta possui uma chave (*key*), um valor (*value*) e pertence a um conjunto de etiquetas (*tag set*). Por exemplo: caso=05, hospital=HU_1 indica a existência de uma etiqueta cuja chave é caso e cujo valor é 05, e de outra etiqueta cuja chave é hospital e cujo valor é HU_1. Juntas, as etiquetas formam um conjunto.
- **series (série)** - formada por uma chave (*key*) composta por uma medida, por um conjunto de etiquetas e por uma chave de campo, além de um conjunto de pontos relacionados à chave. Por exemplo: uti caso=05, hospital=HU_1 HR é uma chave de série com a qual está relacionado o dado 1647178410000 71; uti caso=05, hospital=HU_1 SpO2, por sua vez, é uma chave de série com a qual está relacionado o dado 1647178410000 93.
- **points (pontos)** - estruturas que incluem uma chave de série, um valor de campo e um timestamp. Por exemplo: uti caso=05, hospital=HU_1 HR=71 1647178410000.
- **bucket (balde)** - combinação dos conceitos de banco de dados e de período de retenção (período este que define por quanto tempo será feita a persistência dos pontos). Por exemplo: um bucket chamado pep indica a existência de um espaço de armazenamento para dados de um prontuário eletrônico do paciente.

- **organization (organização)** - *workspace* (espaço de trabalho) para um grupo de usuários que engloba buckets, tarefas (*tasks*) e *dashboards*.



O UNIX timestamp corresponde à contagem de segundos a partir de 01/01/1970 00:00:00 (data/hora consideradas o início da chamada Era UNIX). Esse número de segundos é utilizado para representar datas e horas de forma numérica, aproveitando os benefícios do uso das operações aritméticas que essa representação permite.

Os SGBDs orientados a séries temporais costumam disponibilizar recursos próprios com o objetivo de facilitar a entrada de dados para o seu armazenamento, o que comumente ocorre a partir de origens variadas (como arquivos, sensores ou outros *Softwares*, por exemplo) e com volumes e frequências diversas. Em cenários de Big Data esse processo é conhecido como **ingestão de dados**, onde datasets estruturados, semiestruturados ou não estruturados são disponibilizados de forma ativa ou passiva, tendo como destino um repositório para o armazenamento do seu histórico, para a sua limpeza, transformação e consolidação, e finalmente para a execução de análises sobre o seu conteúdo (ALWIDIAN et al, 2020). Pode ser considerado como a evolução dos processos de **ETL** (*Extract, Transform, Load* - Extração, Transformação, Carga) e **ELT** (*Extract, Load, Transform* - Extração, Carga, Transformação), definidos e utilizados em cenários de Inteligência de Negócios (*Business Intelligence - BI*) e **DW** (*Data Warehousing*).

No *InfluxDB*, a ingestão de dados pode ser feita utilizando-se do *Telegraf* (*software* componente da stack TICK) por meio da importação de arquivos, ou por meio da execução de clientes escritos em diferentes linguagens de programação com o uso de bibliotecas específicas a esse fim. O Quadro 1 apresenta um cliente escrito em linguagem PHP que, a partir da leitura de um arquivo CSV, executa chamadas a funções de uma biblioteca disponibilizada para esta linguagem que permitem o envio dos dados lidos a uma instância do SGBD hospedada em nuvem. O dataset utilizado neste exemplo compreende dados adquiridos a partir de sensores em uma UTI, e seu conteúdo é organizado de forma que os elementos de dados enviados para armazenamento sejam os seguintes:

- **measurement (medida)** = uti;
- **fields (campos)** = HR, SpO2, NBP_Sys, NBP_Dia e etCO2, que correspondem respectivamente aos seguintes indicadores: frequência cardíaca, saturação de oxigênio, pressão sanguínea sistólica, pressão sanguínea diastólica e CO2 expirado;
- **tags (etiquetas ou rótulos)** = hospital e caso, com os valores HU_1 e 05;
- **bucket (balde)** = pep.



```

1: <?php
2:
3: require 'vendor/autoload.php';
4:
5: use InfluxDB2\Client;
6: use InfluxDB2\Model\WritePrecision;
7: use InfluxDB2\Point;
8:
9: $token = '<token gerado para acesso ao InfluxDB>';
10: $org = '<e-mail do usuário para acesso>';
11: $bucket = 'pep';
12:
13: try {
14:
15:     $cliente = new Client([
16:         "url" => "<URL para acesso ao InfluxDB>",
17:         "token" => $token,
18:     ]);
19:
20:     $writeApi = $cliente->createWriteApi();
21:
22:     $linha = 0;
23:     if (($arquivo = fopen("uq_vsd_case05_fulldata_01.csv", "r")) !== FALSE) {
24:         while (($dados = fgetcsv($arquivo, 2000, ",")) !== FALSE) {
25:             $linha++;
26:             if($linha == 1) {
27:                 continue;
28:             }
29:             $data = DateTime::createFromFormat('Y-m-d H:i:s.u', '2022-03-07 ' .
30: str_replace('_', '.', $dados[0]) . '000');
31:             $array = ['name' => 'uti',
32:                      'tags' => ['hospital' => 'HU_1', 'caso' => '05'],
33:                      'fields' => ['HR' => (strlen(trim($dados[3])) > 0 ? (int)
trim($dados[3])) : NULL,
34:                                     'SpO2' => (strlen(trim($dados[6])) > 0 ?
(int)trim($dados[6])),
35:                                     'NBP_Sys' => (strlen(trim($dados[11])) > 0 ?
(int)trim($dados[11])),
36:                                     'NBP_Dia' => (strlen(trim($dados[12])) > 0 ?
(int)trim($dados[12])),
37:                                     'etCO2' => (strlen(trim($dados[8])) > 0 ?
(int)trim($dados[8])),
38:                                     'time' => (int)$data->format('Uv'))];
39:             $writeApi->write($array, WritePrecision::MS, $bucket, $org);
40:         }
41:         fclose($arquivo);
42:     }
43:
44:     $cliente->close();
45:
46: }
47: catch(Exception $e) {
48:     echo $e->getMessage() . "\n";
49:     echo $e->getTraceAsString();
50:     die();
51: }
52:
53: ?>
```

Quadro 1 - Exemplo de programa escrito em linguagem PHP para a ingestão de dados no InfluxDB a partir de um arquivo com dados no formato CSV / Fonte: o autor.



NOVAS DESCOBERTAS

PHP (PHP: *Hypertext Preprocessor*) é uma linguagem de script open source de uso geral, passível de ser embutida em arquivos HTML (*HyperText Markup Language*) de forma a gerar conteúdo dinâmico, ou de ser utilizada no lado servidor para desenvolvimento backend.

Outra forma de acesso ao *InfluxDB*, mais pontual, pode ser realizada por meio de sua API HTTP. A API permite que rotinas administrativas, inserções, exclusões e pesquisas sejam executadas por meio de verbos/métodos HTTP, na forma de requisições e respostas suportadas pelo protocolo. No Quadro 2, por exemplo, é demonstrada a criação de um *bucket* chamado *pep* por meio do método POST, utilizando cabeçalhos para a passagem de parâmetros de autorização e especificação de conteúdo, e dados especificando o bucket na forma de um documento JSON.

```
INFLUX_TOKEN=<TOKEN_DE_ACESSO_À_API>
INFLUX_ORG_ID=<ID_DA_ORGANIZAÇÃO>

curl --request POST \
    "http://<url_do_servidor>/api/v2/buckets" \
    --header "Authorization: Token ${INFLUX_TOKEN}" \
    --header "Content-type: application/json" \
    --data `{
        "orgID": "${INFLUX_ORG_ID}",
        "name": "pep",
        "retentionRules": [
            {
                "type": "expire",
                "everySeconds": 86400,
                "shardGroupDurationSeconds": 0
            }
        ]
    }'
```

Quadro 2 - Exemplo de criação de bucket por meio da API HTTP do InfluxDB. Os dados especificam, além do nome do bucket, a política de retenção dos dados a serem armazenados
Fonte: adaptado de Influx Data ([2022]).



NOVAS DESCOBERTAS

HTTP (*Hypertext Transfer Protocol*) é um protocolo para transmissão de documentos hipermídia (como o HTML, por exemplo) localizado na camada de aplicação, que serve o modelo cliente-servidor de requisição e resposta.



PENSANDO JUNTOS

O acesso a serviços hospedados em intranets ou mesmo na web por meio de APIs HTTP é um facilitador para a interoperabilidade e a comunicação de dados entre diferentes sistemas de informação. Por utilizarem um protocolo padrão suportado pela maioria (se não todas) as linguagens de programação, essas APIs substituem clientes e bibliotecas específicas às linguagens - as quais demandam ajustes e configurações que oneram a sua adoção.

No Quadro 3 é demonstrada a inserção de um ponto utilizando o protocolo de linha do InfluxDB para a representação dos dados. Seguindo o protocolo, o primeiro elemento informado é a medida, separada por uma vírgula do conjunto de tags. A seguir é informado o conjunto de campos, seguido pelo timestamp com a precisão informada junto à URL de acesso ao serviço.

```
INFLUX_TOKEN=<TOKEN_DE_ACESSO_À_API>
INFLUX_ORG_ID=<ID_DA_ORGANIZAÇÃO>

curl --request POST \
  "http://<url_do_servidor>/api/v2/write?org=INFLUX_ORG_ID&bucket=pep&precision=ms" \
  --header "Authorization: Token INFLUX_TOKEN" \
  --header "Content-Type: text/plain; charset=utf-8" \
  --header "Accept: application/json" \
  --data-binary 'uti,hospital=HU_1,caso=05 HR=66,NBP_Sys=12,NBP_Dia=9,etcO2=27 1647548438000'
```

Quadro 3 - Exemplo de criação de ponto por meio da API HTTP do InfluxDB. Os dados são estruturados de acordo com o protocolo de linha do SGBD / Fonte: adaptado de Influx Data ([2022]).

No Quadro 4 é demonstrada a exclusão de pontos utilizando um critério de intervalo para o timestamp (que é obrigatório) e um predicado (opcional) que permite combinar via conjunção (AND) critérios de igualdade utilizando campos ou tags.

```

INFLUX_TOKEN=<TOKEN_DE_ACESSO_À_API>
INFLUX_ORG_ID=<ID_DA_ORGANIZAÇÃO>

curl --request POST \
"http://<url_do_servidor>/api/v2/delete?org=INFLUX_ORG_ID&bucket=pep" \
--header "Authorization: Token INFLUX_TOKEN" \
--header "Content-Type: application/json" \
--data '{
        "start": "2022-03-15T00:00:00Z",
        "stop": "2022-03-17T23:59:59Z",
        "predicate": "_measurement=\"uti\" AND hospital=\"HU_1\""
    }'

```

Quadro 4 - Exemplo de exclusão de pontos por meio da API HTTP do InfluxDB. Os dados a serem excluídos são localizados por um intervalo de tempo obrigatório e critérios opcionais envolvendo tags e campos / Fonte: adaptado de Influx Data ([2022]).

Seguindo o mesmo padrão de acesso utilizado para a criação de *buckets*, a inserção de pontos e a exclusão de pontos, a execução de consultas (*queries*) faz uso de um *endpoint* com a especificação dos predicados a serem aplicados pelo SGBD. Esses predicados podem ser escritos em *Flux* ou em *InfluxQL* (esta última em modo de compatibilidade), duas linguagens suportadas pelo backend do InfluxDB para a manutenção de dados. O Quadro 5 exemplifica uma requisição de consulta escrita em *Flux*, onde são especificados: 1) o bucket a partir do qual serão recuperados os dados de interesse (no exemplo, `pep`); 2) o período (range) de datas que será utilizado para filtrar os dados pelo seu timestamp (no exemplo, `start: -10d` indicando que deverão ser recuperados dados cujo timestamp esteja no período de 10 dias anteriores à data/hora de execução da query); 3) um primeiro filtro indicando qual é a medida de interesse (`uti`); 4) um segundo filtro indicando qual campo cujos dados serão retornados ao cliente (`HR`); 5) qual janela de tempo e função de agregação serão utilizadas sobre os dados brutos encontrados antes de disponibilizar o resultado ao cliente (no exemplo, `every: 1d, fn: mean` indicando que deverá ser calculada a média do valor do campo selecionado para cada dia presente no resultado). Os resultados obtidos podem ser visualizados no Quadro 6, dos quais vale destacar as seguintes colunas:

- **table** - número da tabela interna ao InfluxDB de onde os resultados foram recuperados;
- **_start** - timestamp inicial a partir do qual os dados foram recuperados;
- **_stop** - timestamp final até o qual os dados foram recuperados;
- **_time** - timestamp de referência para os valores obtidos (no exemplo, um timestamp de hora em hora para o qual é calculada a média dos valores do campo HR);
- **_value** - valor calculado de acordo com os predicados da query. No exemplo, foi utilizado o caractere ‘_’ para indicar ausência de valor;
- **_field** - campo recuperado de acordo com os predicados da query;
- **_measurement** - medida especificada na query;
- **caso e hospital** - tags relacionadas aos dados recuperados.

```

INFLUX_TOKEN=<TOKEN_DE_ACESSO_À_API>
INFLUX_ORG_ID=<ID_DA_ORGANIZAÇÃO>

curl --request POST \
"http://<url_do_servidor>/api/v2/query?org=INFLUX_ORG_ID&bucket=pep" \
--header "Authorization: Token INFLUX_TOKEN" \
--header 'Accept: application/csv' \
--header "Content-Type: application/vnd.flux" \
--data 'from(bucket:"pep")
|> range(start: -10d)
|> filter(fn: (r) => r._measurement == "uti")
|> filter(fn: (r) => r._field == "HR")
|> aggregateWindow(every: 1d, fn: mean)'

```

Quadro 5 - Exemplo de query executada por meio da API HTTP do InfluxDB. A especificação dos predicados é feita em linguagem Flux / Fonte: adaptado de Influx Data ([2022]).

```

result table _start _stop _time _value _field _measurement caso hospital
  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-09T00:00:00Z _ HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-10T00:00:00Z _ HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-11T00:00:00Z 59.45847430577721 HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-12T00:00:00Z _ HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-13T00:00:00Z _ HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-14T00:00:00Z _ HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-15T00:00:00Z _ HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-16T00:00:00Z _ HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-17T00:00:00Z _ HR uti 05 HU_1

  result 0 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-18T00:00:00Z _ HR uti 05 HU_1

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-09T00:00:00Z _ HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-10T00:00:00Z _ HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-11T00:00:00Z _ HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-12T00:00:00Z _ HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-13T00:00:00Z 86.7932356105433 HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-14T00:00:00Z _ HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-15T00:00:00Z _ HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-16T00:00:00Z _ HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-17T00:00:00Z _ HR uti 20 HU_3

  result 1 2022-03-08T17:41:27.518606037Z 2022-03-18T17:41:27.518606037Z
  2022-03-18T00:00:00Z _ HR uti 20 HU_3

```

Quadro 6 - Resultado da execução da query especificada no Quadro 5 por meio da API HTTP do InfluxDB / Fonte: o autor.

**NOVAS DESCOBERTAS**

Flux é uma linguagem de scripting e de consulta focada em séries temporais. Seus diferenciais incluem o acesso a diferentes datasources, a extensibilidade por meio de novas funções e bibliotecas, e a integração com diferentes ferramentas para a execução de análise de dados.

Como pode ser observado nos exemplos de acesso ao *InfluxDB* via API HTTP, não são utilizadas as convenções REST (*Representational State Transfer* - Transferência Representacional de Estado) com relação aos verbos/métodos HTTP (por exemplo: tanto a inserção quanto a exclusão de dados são realizadas via POST). Essa é uma decisão de projeto do SGBD que considera ser desnecessário seguir essas convenções dado o número reduzido de endpoints que são disponibilizados aos clientes (WRITING, 2022).

**NOVAS DESCOBERTAS**

REST (*Representational State Transfer*) é uma convenção ou conjunto de restrições de arquitetura para a construção de sistemas distribuídos escaláveis, eficientes e confiáveis. Esses objetivos são atingidos por meio da padronização de formatos e de operações de acesso ao conteúdo.

Como alternativa ao uso do Telegraf para a importação de arquivos, a execução de clientes escritos em diferentes linguagens de programação e ao uso da API HTTP visando à ingestão de dados, o InfluxDB disponibiliza uma interface web para facilitar esse processo de ingestão, além da visualização e análise de *dashboards* e a definição de alertas com base nos dados armazenados. A Figura 3 exibe os acessos a esses facilitadores.

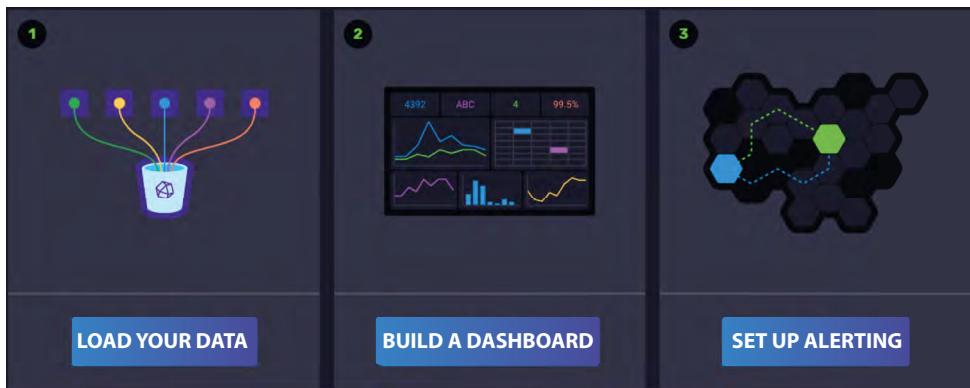


Figura 3 - Acessos às interfaces de ingestão de dados, construção de *dashboards* e configuração de alertas no InfluxDB Cloud / Fonte: o autor.

Descrição da Imagem: A figura exibe três caixas numeradas da esquerda para a direita de 1 a 3. A caixa de número 1 exibe um balde sendo alimentado por cinco linhas provenientes de cinco diferentes origens, representando a entrada de dados no InfluxDB a partir de diferentes fontes. Na parte inferior da figura é exibido um botão com o rótulo “LOAD YOUR DATA” que dá acesso às interfaces de ingestão de dados via web. A caixa de número 2 exibe um painel com gráficos, tabelas e indicadores, representando um dashboard para a visualização e a análise dos dados armazenados pelo SGBD. Na parte inferior da figura é exibido um botão com o rótulo “BUILD A DASHBOARD” que dá acesso às interfaces para a construção desses painéis de visualização. A caixa de número 3 exibe uma figura formada por hexágonos (figuras menores, com seis lados cada), das quais duas estão destacadas e conectadas por duas linhas tracejadas. Essa conexão é utilizada para representar que as ocorrências em uma das figuras sinalizam alterações na outra figura que está conectada. Aqui, o objetivo é indicar que caso sejam observados determinados padrões nos dados armazenados, alertas serão disparados para notificar os interessados. Na parte inferior da figura é exibido um botão com o rótulo “SET UP ALERTING” que dá acesso às interfaces para a configuração desses alertas.

A opção “LOAD YOUR DATA” dá aos usuários um acesso para o upload de arquivos em diferentes formatos visando a sua ingestão em um bucket criado previamente, além de exibir exemplos de código-fonte para que a ingestão de dados seja feita utilizando uma linguagem de programação de preferência a partir de bibliotecas construídas especificamente para esse fim. Há, ainda, a possibilidade de selecionar plugins do Telegraf para conectar e coletar dados a partir de uma grande variedade de fontes. A Figura 4 exibe os acessos ao upload de arquivos, ao código-fonte e (parcialmente) aos plugins do Telegraf.

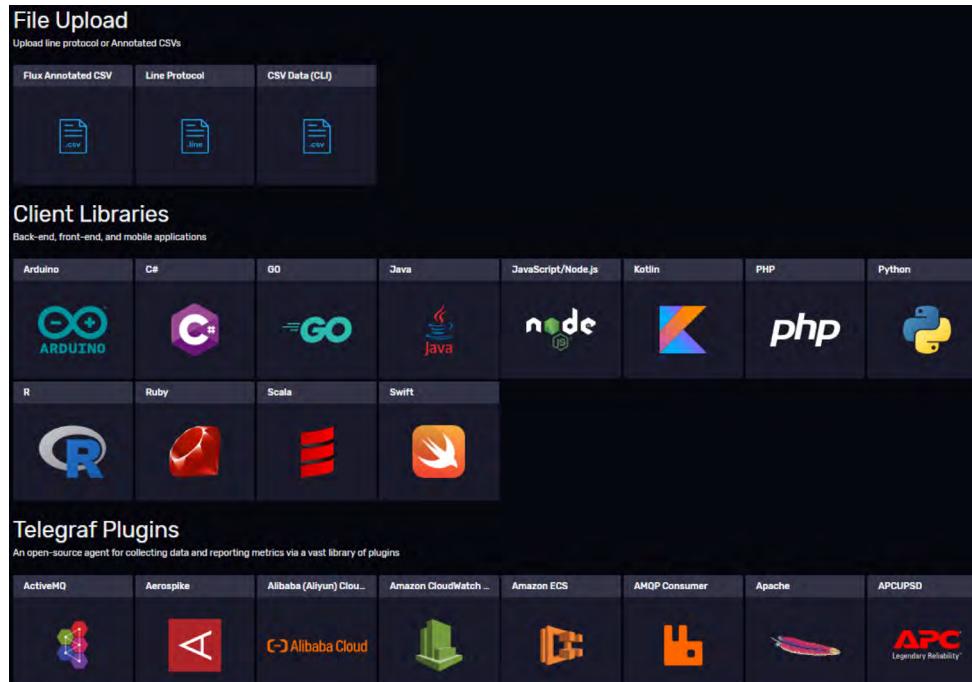


Figura 4 - Acessos ao upload de arquivos, ao código-fonte de clientes em diferentes linguagens de programação e a plugins do Telegraf no InfluxDB Cloud / Fonte: o autor.

Descrição da Imagem: a figura exibe, de cima para baixo, três tipos de acesso a opções para a ingestão de dados. O primeiro tipo é intitulado “File Upload”, seguido da instrução “Upload line protocol or Annotated CSVs”. Logo abaixo a essa instrução são exibidos, lado a lado, três botões com a figura de uma página impressa e, respectivamente, os seguintes rótulos: “Flux Annotated CSV”, “Line Protocol” e “CSV Data (Cli)”. O segundo tipo é intitulado “Client Libraries”, seguido da informação “Back-end, front-end, and mobile applications”. Logo abaixo a essa informação são exibidos, lado a lado, botões com o logotipo de diferentes linguagens de programação/ferramentas e, respectivamente, os seguintes rótulos: “Arduino”, “C#”, “GO”, “Java”, “JavaScript/Node.js”, “Kotlin”, “PHP”, “Python”, “R”, “Ruby”, “Scala” e “Swift”. O terceiro tipo é intitulado “Telegraf Plugins”, seguido da informação “An open-source agent for collecting data and reporting metrics via a vast library of plugins”. Logo abaixo a essa informação são exibidos, lado a lado, botões com o logotipo de diferentes fontes de dados e, respectivamente, os seguintes rótulos: “ActiveMQ”, “Aerospike”, “Alibaba (Aliyun) Cloud...”, “Amazon CloudWatch...”, “Amazon ECS”, “AMQP Consumer”, “Apache” e “APCUPSD”.

A opção “BUILD A DASHBOARD” permite que os usuários construam um painel visual de indicadores. Esse painel é formado por células, cada célula é configurada para assumir uma identidade visual (um gráfico, uma tabela de dados, dentre outras identidades possíveis) que irá exibir dados resultantes da aplicação de um ou mais critérios de seleção sobre os dados armazenados pelo SGBD. A Figura 5 exibe um dashboard construído a partir desta opção composto por quatro células, sendo as três primeiras referentes, respectivamente, aos hospitais HU_1,

HU_2 e HU_3, e a quarta sendo um consolidado de todos os hospitais. Nos gráficos apresentados são plotados, respectivamente, os valores para os campos: 1) HR e inO2; 2) HR, NBP_Dia e NBP_Sys; 3) HR e inO2; 4) HR, NBP_Dia, NBP_Sys, SpO2, etCO2 e inO2. Na Figura 6, por sua vez, é possível visualizar os parâmetros para a definição do primeiro gráfico.



Figura 5 - Dashboard para a exibição dos dados adquiridos nas Unidades de Tratamento Intensivo dos hospitais HU_1, HU_2 e HU_3 construído no InfluxDB Cloud / Fonte: o autor.

Descrição da Imagem: a figura exibe como título a palavra "UTI", seguida de botões, rótulos, caixas de seleção e campos com listas para seleção utilizadas na configuração do dashboard, como segue: um botão com o rótulo "ADD CELL" (utilizado para incluir uma nova célula no dashboard); um botão com o rótulo "ADD NOTE" (que permite ao usuário inserir anotações junto ao dashboard); uma caixa de seleção com o rótulo "Show Variables" (para habilitar ou desabilitar a exibição de variáveis relacionadas ao dashboard); uma caixa de seleção com o rótulo "Enable Annotations" (que permite inserir anotações em nível de célula); um botão com o rótulo "... que disponibiliza opções de visualização para o dashboard, como modo claro e escuro; um rótulo com o conteúdo "?" com dicas para ajuste de zoom e anotações; um botão com duas setas circulares como rótulo, utilizado para atualizar o conteúdo das células; um botão com o rótulo "SET AUTO REFRESH" (permitindo a configuração de intervalos para a atualização automática das células); um campo com uma lista para seleção intitulado "Local" (indicando se o timestamp a ser utilizado nas células é local ao SGBD ou UTC); e finalmente um campo com uma lista para seleção que permite especificar o período de tempo a ser exibido pelas células. A seguir são exibidos os gráficos com os dados dos hospitais: 1) título "HU_1", um gráfico de linhas para as variáveis HR e inO2, com os rótulos do eixo y exibidos de 10 em 10, iniciando em 30 e terminando em 90, e os rótulos do eixo x exibindo os timestamps "2022-03-10 20:00:00", "2022-03-10 20:05:00" e "2022-03-10 20:10:00"; 2) título "HU_2", um gráfico de linhas para as variáveis HR, NBP_Dia e NBP_Sys, com os rótulos do eixo y exibidos de 10 em 10, iniciando em 90 e terminando em 170, e os rótulos do eixo x exibindo os timestamps "2022-03-11 20:00:00" e "2022-03-11 20:15:00"; 3) título "HU_3", um mapa de calor para as variáveis HR e inO2, com os rótulos do eixo y exibidos de 10 em 10, iniciando em 30 e terminando em 100, e os rótulos do eixo x exibindo os timestamps "2022-03-12 20:00:00" e "2022-03-12 20:15:00"; 4) título "Todos os hospitais", um gráfico de linhas para as variáveis HR, NBP_Dia, NBP_Sys, SpO2, etCO2 e inO2, com os rótulos do eixo y exibidos de 10 em 10, iniciando em 10 e terminando em 140, e os rótulos do eixo x exibindo os timestamps iniciando em "2022-03-10 20:00:00" e terminando em "2022-03-10 20:10:00", variando de 1 em 1 minuto.

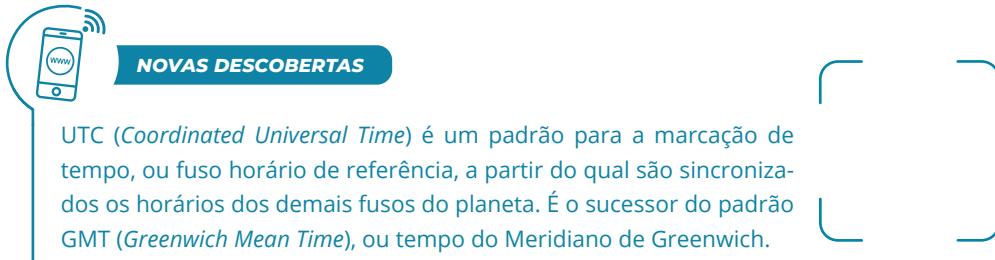


Figura 6 - Definição de parâmetros para a construção do gráfico referente ao hospital HU_1 no InfluxDB Cloud / Fonte: o autor.

Descrição da Imagem: A figura exibe como título a descrição “HU_1”, seguido de campos com listas para seleção e botões utilizados na configuração do dashboard, como segue: uma lista para a seleção do tipo de gráfico a ser utilizado; um botão com o rótulo “CUSTOMIZE” (utilizado para ajustar os parâmetros aplicáveis ao tipo de gráfico selecionado); um botão com o rótulo “x” para fechar a interface de definição de parâmetros; e um botão com o rótulo “√” utilizado para confirmar as alterações feitas e fechar a interface de definição de parâmetros. A seguir é exibido um gráfico de linhas para as variáveis HR e inO₂, com os rótulos do eixo y exibidos de 5 em 5, iniciando em 25 e terminando em 95, e os rótulos do eixo x exibindo os timestamps “2022-03-10 20:00:00”, “2022-03-10 20:05:00” e “2022-03-10 20:10:00”. Abaixo do gráfico são disponibilizados componentes para a inclusão de novas queries, além de um seletor com o rótulo “View Raw Data” que permite habilitar ou desabilitar a visualização dos dados brutos selecionados pelos filtros; um botão com o rótulo “CSV” que permite o download do dataset de resultado como um arquivo CSV; um botão com duas setas circulares como rótulo, utilizado para atualizar o conteúdo do gráfico; um campo com uma lista para seleção que permite especificar o período de tempo a ser exibido pelo gráfico; um botão com o rótulo “SCRIPT EDITOR” que dá acesso ao editor de scripts em linguagem Flux para a especificação de conteúdo e filtros do gráfico; e um botão com o rótulo “SUBMIT” que atualiza a estrutura do gráfico de acordo com as alterações ou as novas especificações de conteúdo e filtros. Finalmente, na parte inferior da tela, são disponibilizados componentes para a escolha do bucket a ser pesquisado, da medida a ser utilizada, dos campos a serem exibidos no gráfico e das tags a serem utilizadas como filtro, além da especificação da função de agregação a ser aplicada sobre os dados (como por exemplo, média) e a definição de janelas de tempo a serem utilizadas em conjunto a essa função.

A opção “SET UP ALERTING” disponibiliza aos usuários opções para a definição e a configuração de alertas a serem disparados a partir de observações feitas sobre o dataset armazenado pelo SGBD. A Figura 7 mostra as configurações para um alerta criado a partir do campo HR, configurado para enviar uma mensagem via HTTP POST a um *endpoint* caso seja encontrado um valor para este campo que seja maior do que 90. A verificação dos dados para o disparo do alerta é feita a cada cinco segundos.

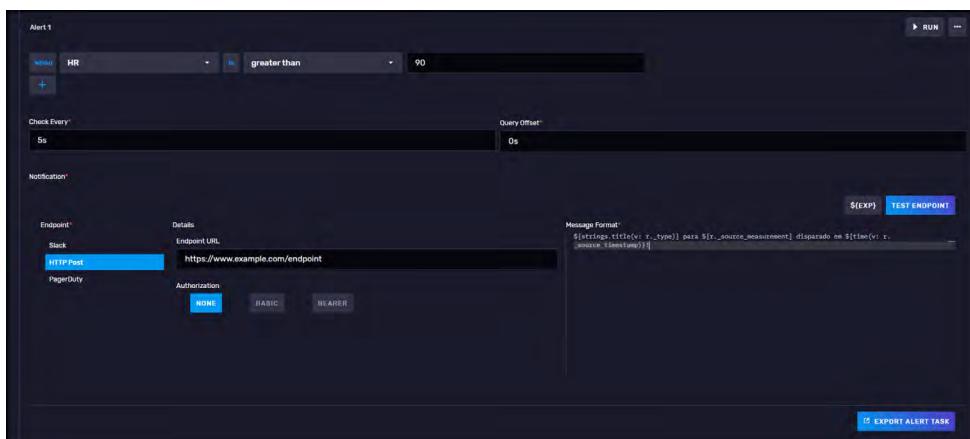
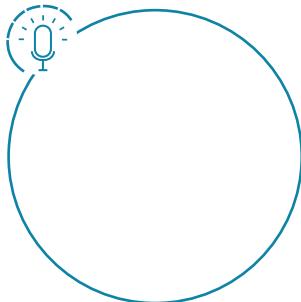


Figura 7 - Definição de parâmetros para a construção de um alerta vinculado a valores de frequência cardíaca (campo HR) do dataset armazenado no InfluxDB Cloud / Fonte: o autor.

Descrição da Imagem: a figura exibe como título a descrição “Alert 1”, seguido de campos com listas para seleção e botões utilizados na configuração do alerta, como segue: um botão com o rótulo “RUN”, utilizado para a execução imediata do alerta, seguido de um botão com o rótulo “...” que dá acesso a opções de configuração complementares, como exclusão e criação de uma cópia do alerta. Em seguida são disponibilizados campos para a especificação da condição a ser observada pelo alerta. Na figura, é escolhido o campo HR a partir de uma lista de seleção e a condição “greater than” a partir de uma segunda lista, com a especificação do valor 90 (o objetivo aqui é disparar o alerta quando for encontrado um valor maior do que 90 para o campo HR). A definição da frequência de checagem é feita a seguir, com o valor “5s” sendo especificado no campo intitulado “Check Every”, e com o valor “0s” sendo especificado no campo “Query Offset”. Por fim, é detalhada como será feita a notificação: a escolha do endpoint (a partir das opções “Slack”, “HTTP Post” ou “PagerDuty”), a especificação dos dados de conexão ao endpoint selecionado (no caso de um “HTTP Post”, por exemplo, qual a URL do endpoint e o método de autorização a ser utilizado - “NONE”, “BASIC” ou “BEARER”) e a definição do formato de mensagem que será enviado ao endpoint selecionado (na figura, “\${strings.title(v:r_type)} para \${r_source_measurement} disparado em \${time(v:r_source_timestamp)}!“). Para auxiliar na composição da mensagem, o botão com rótulo “\${EXP}” pode ser clicado para que o usuário tenha acesso a uma lista de variáveis e expressões. O botão com rótulo “TEST ENDPOINT” pode ser clicado para a execução de um teste de comunicação com o endpoint escolhido, e o alerta pode ser exportado como tarefa clicando no botão com rótulo “EXPORT ALERT TASK”.



Os SGBDs orientados a séries temporais ganham destaque quando comparados a outros sistemas baseados em modelos de dados diversos. Acompanhe em nosso podcast os comentários feitos sobre a importância desses SGBDs como suporte a cenários em que o registro de tempo, juntamente com a sua análise, são fatores primordiais. Não perca!

A experimentação executada pela equipe de TI utilizando o InfluxDB como SGBD orientado a séries temporais foi avaliada como extremamente positiva. Os testes utilizando datasets similares àqueles que serão encontrados na prática nos cenários de demanda mais sensível (UTIs) foram concluídos de forma muito satisfatória, atendendo inclusive às questões de definição e configuração de alertas - os quais podem fazer a diferença no acompanhamento de indivíduos em estado grave, onde decisões precisam ser tomadas de forma imediata.

A partir do feedback da equipe, Samantha e Marco efetivaram essa categoria de SGBD como componente do ecossistema do projeto Hospital sem Papel. Ficou decidido que todo e qualquer dataset que apresente características de temporalidade mandatórias, e cujas marcas temporais sejam o dado de referência, deverá ser organizado e persistido na forma de uma série temporal. Com isso, o projeto ganha respaldo da teoria já existente que define a metodologia de análise dessas séries, e abre caminho para que decisões com base em dados possam ser tomadas tanto no nível operacional (os diferentes setores dentro dos hospitais) quanto no nível gerencial (a administração de cada hospital, ou da rede como um todo).



1. Relembrando as características de séries temporais, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
 - a) As marcas temporais nesse tipo de série são opcionais e, quando existentes, têm a mesma importância dos demais dados que compõem os *datasets*.
 - b) A sazonalidade de uma série temporal corresponde a seu comportamento de médio/longo prazo, sem uma frequência definida.
 - c) O ciclo de uma série temporal corresponde ao seu comportamento de longo prazo que permite identificar a mudança ou a manutenção de sua direção de crescimento ou baixa.
 - d) A tendência de uma série temporal corresponde ao seu comportamento sazonal (diário ou semanal, por exemplo).
 - e) Séries temporais assumem características discretas ou contínuas, dependendo de quais fenômenos são observados e com qual frequência seus dados são adquiridos.
2. Com base na arquitetura e na organização do *InfluxDB*, é CORRETO afirmar que:
 - a) *Tags* (etiquetas ou rótulos) possuem relação apenas com timestamps, e são independentes de campos e medidas.
 - b) A precisão de um timestamp, tanto para a persistência quanto para a ingestão dos dados feita por um cliente, é em nanosegundos.
 - c) Um *bucket* (balde) corresponde a um banco de dados cujo conteúdo é mantido de acordo com uma política de retenção.
 - d) Cada valor armazenado é relacionado a apenas uma *tag*.
 - e) Um ponto é uma combinação de timestamp e um ou mais campos.
3. Observando as opções oferecidas pelo *InfluxDB* para a consulta de dados e a construção de *dashboards*, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
 - a) Campos e *tags* podem ser utilizados como filtros.
 - b) Alertas com base em dados não são permitidos.
 - c) Resultados de consultas são exibidos exclusivamente em gráficos.
 - d) O *InfluxDB* segue à risca as especificações *REST*.
 - e) A ingestão de dados no *InfluxDB* demanda a escolha e o uso de uma linguagem de programação de *script*.

MEU ESPAÇO



Conexões e Relacionamentos em Bancos de Dados Grafo

Dr. Alexandre Savaris

OPORTUNIDADES DE APRENDIZAGEM

Nesta unidade, o processo de rastreamento de pessoas e locais em cenários de surto, endemia, epidemia ou pandemia é descrito com base nas operações necessárias à sua execução. Por demandar a criação e a manutenção de relacionamentos entre os dados coletados, o conceito de grafo como estrutura de dados de suporte é revisado, acompanhado da apresentação de um Sistema Gerenciador de Banco de Dados com suporte a esse conceito (Neo4j). A versão em nuvem do SGBD, gratuita para testes, é utilizada na ingestão de um dataset para execução de queries e visualização pictórica e tabular de resultados.

762.66

835.01

105.08

Como interromper a transmissão de um vírus e a contaminação de novos indivíduos em cenários de surto, endemia, epidemia ou pandemia? De acordo com as orientações da OPAS (Organização Pan-Americana da Saúde), em conformidade à Organização Mundial da Saúde (OMS), utilizando como base o cenário da pandemia de COVID-19, a execução de atividades de **rastreamento de contatos** por meio dos **rastreadores de contato** (OPAS, 2021). Essas atividades compreendem a identificação de indivíduos que tiveram contato com outros indivíduos infectados, resultando em um encaminhamento dos primeiros para a quarentena. Ademais, ao considerar os locais ou eventos onde ocorreu esse contato, é possível identificar fontes de infecção para, em seguida, direcionar a implementação de medidas de saúde pública e sociais.

O processo é iniciado de forma **retrospectiva**; a partir de um caso confirmado é identificada uma fonte de infecção. A combinação de caso confirmado + fonte de infecção dá subsídio à execução do **rastreamento prospectivo**, atividade caracterizada pela identificação de contatos do indivíduo infectado, o encaminhamento desses contatos para uma quarentena e o acompanhamento dos indivíduos em quarentena durante o período estipulado (WORLD HEALTH ORGANIZATION, 2020a). É um processo recorrente, visto que, se durante a quarentena um dos contatos originais for diagnosticado como positivo, passam a ser considerados os seus próprios contatos para ingressar na quarentena, e assim por diante. O trabalho de prospecção demanda, além da constituição de uma equipe devidamente capacitada, o suprimento de equipamentos e logística mínima adequada à execução do trabalho, o envolvimento das comunidades onde o trabalho será realizado e, não menos importante, a organização e a análise dos dados coletados.

Em um cenário macro, os rastreamentos retrospectivos e prospectivos implicam na identificação de **relações entre pessoas e locais** (ou fontes de infecção e casos confirmados), de **relações de pessoas entre si** (casos confirmados e indivíduos não infectados), e da **combinação entre os anteriores em diferentes níveis** (como por exemplo a relação de uma pessoa com outras pessoas que visitaram dois ou três locais diferentes e tiveram contato com outras pessoas, que por sua vez visitaram quatro ou cinco locais diferentes, e assim por diante). Registrar essas cadeias de relações é imprescindível para que seja possível executar um rastreamento efetivo, isolando os grupos de interesse e demarcando potenciais fontes de infecção. Aliado a necessidade de registro, o desempenho do processo de análise é mandatório para que seja possível extrair dos dados obtidos os direcionamentos para o controle e a interrupção da transmissão. Esse desempenho pode ser alcançado com a adoção de ferramentas capazes de trabalhar com datasets que representem, da forma mais fiel possível, as relações observadas no mundo real.

A correta execução das atividades de rastreamento implica na definição de quais indivíduos considerar, bem como de quais locais observar. Indivíduos passíveis de monitoração (considerados suspeitos) são aqueles que atendem a um ou mais critérios clínicos (sintomas) e também critérios epidemiológicos (residência em locais de risco, trânsito ou viagem a locais de risco, ou execução de atividades em locais de risco). Somam-se a esse grupo os contatos feitos por esses indivíduos – outras pessoas com as quais houve aproximação durante uma janela de tempo definida, com as quais houve contato físico direto, ou com as quais houve convivência sem o uso dos Equipamentos de Proteção Individual (EPIs) adequados (WORLD HEALTH ORGANIZATION, 2020b). Com relação aos locais, são considerados aqueles utilizados

para o trânsito ou a permanência dos indivíduos suspeitos e dos seus contatos – o que inclui a própria comunidade em que esses indivíduos estão estabelecidos – além dos estabelecimentos de saúde responsáveis pelo primeiro atendimento a casos suspeitos, ou os estabelecimentos para os quais os casos confirmados são encaminhados para tratamento. Resultados efetivos são obtidos quando a testagem dos indivíduos suspeitos e seus contatos é feita de forma extensiva, abrangendo também as pessoas que tenham circulado pelos mesmos ambientes em que os primeiros reportaram ter frequentado.

Apesar de contemplar apenas o nível hospitalar, o projeto Hospital sem Papel (Paperless Hospital) desenvolvido por Luísa, Gaspar, Laura, Samanta e Marco (juntamente com suas equipes) tem potencial para colaborar com o processo de rastreamento prospectivo. Para tanto, casos encaminhados para tratamento e/ou internação podem ser utilizados na composição de uma base de dados cuja organização objetiva explicitar as relações interpessoais e interlocais dos indivíduos infectados. Considerando que os casos recebidos pelos hospitais da rede já foram confirmados em uma etapa anterior, os contatos estabelecidos por esses indivíduos e os locais frequentados durante o período da possível infecção podem ser mapeados e observados de forma mais assertiva, o que leva à economia de tempo e de recursos. A equipe de TI deve viabilizar a infraestrutura de hardware e software compatível com a necessidade de estabelecer vínculos entre pessoas, locais, e pessoas/locais, de forma a viabilizar os estudos que podem levar a atividades proativas de prevenção. Essa infraestrutura deve considerar estruturas de dados robustas, com foco na representação de relações entre diferentes entidades (pessoas e locais), capazes de escalar para o armazenamento e a composição de um histórico volumoso e extensível a períodos de tempo variáveis. A escolha do ferramental deve, também, considerar opções que facilitem a execução de consultas e análises em tempo quase-real.

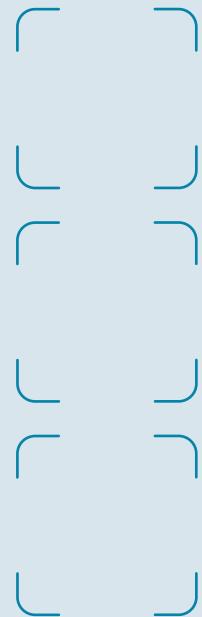
Tecnicamente, a representação dos elementos a serem rastreados (indivíduos com suspeita de infecção, os seus contatos e os locais por onde esses indivíduos e contatos estiveram), juntamente com as suas inter-relações, é compatível com o modelo matemático de **grafo**. Neste modelo, elementos chamados **vértices**, ou **nós**, relacionam-se entre si por meio de **arestas**; vértices e arestas formam, respectivamente, dois conjuntos que, quando vistos de forma consolidada, compõem um grafo completo (NETTO; JURKIEWICZ, 2009). Em uma analogia direta com o cenário de rastreamento, os indivíduos, os seus contatos e os locais frequentados

por ambos compõem o conjunto de vértices. Suas relações de proximidade ou frequência, por sua vez, compõem as arestas que os interligam. À equipe de TI cabe identificar ferramentas de software capazes de representar dados na forma de elementos e relações, e que ofereçam facilidades para a análise do conteúdo armazenado com base na teoria de grafos.

A sua ajuda nesta etapa pode ser útil pesquisando nos Qr Codes ao lado. Você pode colaborar com Luísa e os demais, indicando técnicas de análise em teoria dos grafos que podem ser utilizadas para auxiliar no rastreamento prospectivo. Procure relacionar as técnicas vistas, descrevendo brevemente o seu funcionamento.

Os Sistemas Gerenciadores de Bancos de Dados (SGBDs) relacionais costumam ser utilizados como solução para diferentes tipos de problema, nos mais variados cenários de uso. Seja para armazenar dados convencionais (letras e números) ou mesmo dados não convencionais (dados multimídia e documentos complexos), a flexibilidade oferecida pelo modelo relacional explorada durante a modelagem conceitual, lógica e física favorece a adoção de ferramentas que implementam esse modelo. Por que não assumir, então, que esses SGBDs são a solução ideal para representar indivíduos, contatos, locais e suas relações, como necessário para auxiliar o processo de rastreamento prospectivo? Possivelmente porque apesar de facilitar o armazenamento desses dados, os SGBDs relacionais não oferecem um ferramental para a análise eficiente deste conteúdo.

Em seu diário de bordo, procure registrar a sua opinião sobre o uso dos SGBDs relacionais como ferramentas de referência para o armazenamento de dados, independentemente do problema abordado.



Mais tarde, avalie de forma crítica esse registro com base nos conhecimentos adquiridos no restante do módulo, especificamente no que compete à organização, persistência e análise do conteúdo de grafos. A sua opinião original e crítica são importantes para que a equipe de TI possa fundamentar a escolha por ferramentas não convencionais, que se adaptem melhor ao problema que precisa ser abordado.

DIÁRIO DE BORDO

A necessidade de armazenar elementos e as relações entre esses elementos para posterior análise uma vez mais levou a equipe de TI a relembrar conceitos básicos vistos em seus cursos de graduação e pós-graduação. Diferentemente dos objetivos almejados com a adoção de tecnologias para o uso de documentos estruturados e semiestruturados e de séries temporais representadas como pares chave-valor, as atividades de rastreamento a serem executadas com o suporte do projeto Hospital sem Papel visam à identificação antecipada de potenciais indivíduos, contatos e locais para ações de saúde assertivas. Para tanto, a escolha de ferramentas altamente especializadas e fundamentadas em modelos matemáticos reconhecidos é uma necessidade. Apresentam-se, então, os conceitos definidos pela teoria dos grafos.

A **teoria dos grafos** é um ramo da matemática que estuda as relações entre os objetos de um conjunto determinado. Surgido no século XVIII com o trabalho de Leonhard Euler visando à solução do problema das pontes de Königsberg (vide Figura 1), esse ramo foi desenvolvido nos séculos seguintes por cientistas como Gustav Robert Kirchhoff, Arthur Cayley e William Rowan Hamilton (entre outros), que definiram novas características e utilizaram o conceito de grafo na resolução de problemas em que o estabelecimento de relações entre elementos é factível.

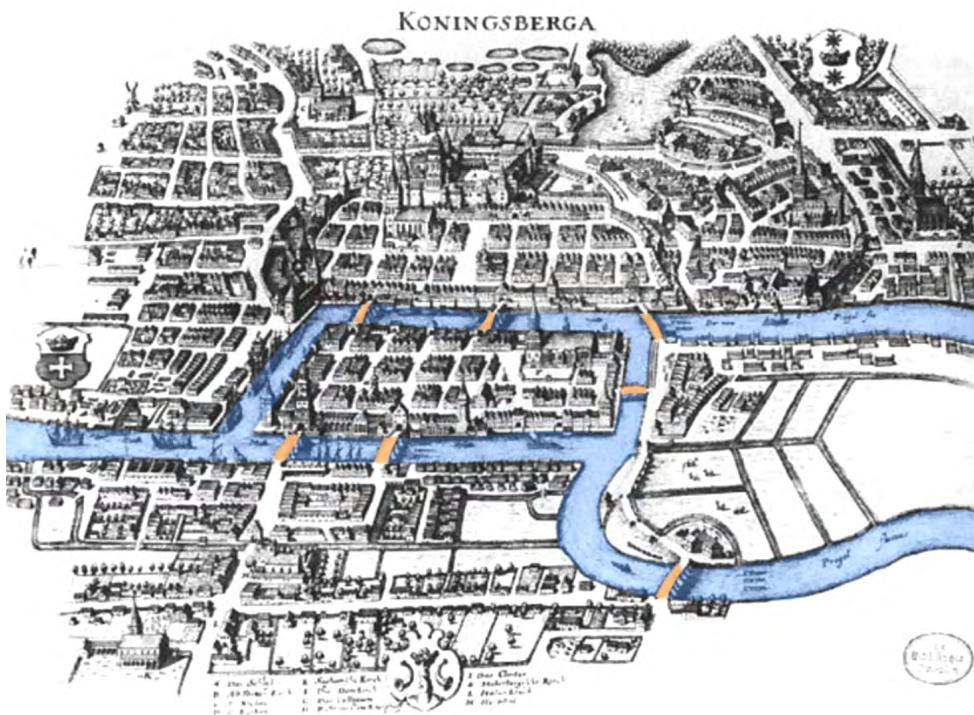


Figura 1 - Mapa de Königsberg. Acreditava-se que seria possível, em uma caminhada contínua, cruzar as sete pontes (identificadas na figura, em amarelo) sem passar duas vezes por qualquer uma delas. Euler, utilizando a notação de grafos, provou que o problema não possui solução / Fonte: Wikimedia Commons ([2022], on-line).

Descrição da Imagem: no topo da figura é exibido o título "KONINGSBERGA", seguido de um mapa da cidade de Königsberg à época de Euler. A cidade é cruzada por um rio, orientado da esquerda para a direita, que a divide em ilhas e uma parte continental. O rio se divide em dois cursos: um desses cursos volta a se conectar com o fluxo original, e o outro divide-se em dois mais uma vez. Há sete pontes cruzando o rio, que interligam as ilhas entre si e as ilhas com a parte continental da cidade.

Um grafo (estrutura em torno da qual o ramo é definido) $G = G(V, E)$ corresponde a um conjunto discreto finito e não vazio V de vértices, pontos ou nós ligados dois a dois por meio de arestas, linhas ou arcos que formam um conjunto E (vide Figura 2 para a representação em grafo do problema das pontes de Königsberg) (OSTROSKI; MENONCINI, 2009). As características das ligações entre os nós permitem classificar instâncias de grafos como sendo direcionados (onde dois vértices conectados entre si formam um par ordenado, sendo visitados na ordem, ou direção assumida pela aresta) ou não direcionados, onde esta ordem/direção não existe; essas conexões podem também ser rotuladas ou receberem um peso indicativo do custo relacionado a percorrer o caminho entre os dois vértices. Quando tratadas em conjunto ou em sequência, as arestas de um grafo delimitam e caracterizam caminhos (sequências de vértices conectados entre uma origem e um destino), que podem ser cíclicos (quando origem e destino correspondem ao mesmo vértice) ou não, gerando estruturas parcial ou completamente conectadas. Nós, por sua vez, são caracterizados de acordo com o número de conexões de entrada e saída que possuem (que somadas definem o seu grau), e quando observados em conjunto permitem classificar a estrutura que compõem como sendo regular (todos os nós conectados a um mesmo número de vizinhos), como sendo fortemente conectada, fracamente conectada ou mesmo unilateralmente conectada. A combinação de características entre nós e arestas, por sua vez, contribui para a definição de algoritmos aplicáveis à resolução de problemas representáveis na forma de grafos. Um exemplo de combinação está na identificação do emparelhamento em grafos, sendo o emparelhamento equivalente a um conjunto de arestas que não possuem nós comuns na sua origem ou destino. Um grafo é dito perfeitamente emparelhado quando todos os seus nós estão conectados ao conjunto de emparelhamento; além disso, um emparelhamento é dito maximal quando não é possível adicionar novos vértices ao conjunto, e o emparelhamento máximo corresponde ao conjunto que possuir o maior número de arestas não adjacentes. Encontrar o emparelhamento em um grafo é solução para problemas de otimização combinatória, onde busca-se correlacionar vértices de forma a atender restrições de combinação (como por exemplo a distribuição de hóspedes em quartos de hotel de acordo com suas preferências de acomodação).

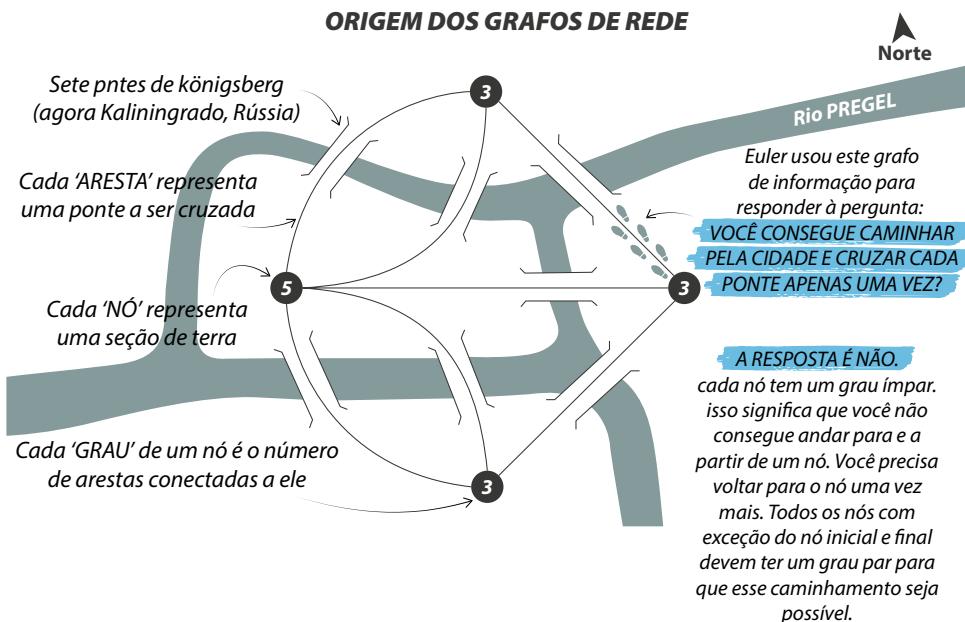
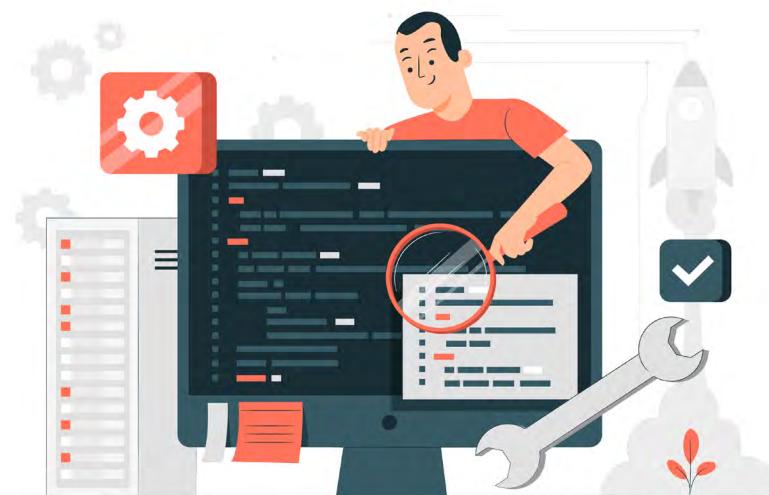


Figura 2 - O problema das pontes de Königsberg representado por um grafo com quatro nós e sete arestas / Fonte: The Reliants Project (2022, on-line).

Descrição da Imagem: a figura exibe um desenho simplificado do mapa da cidade de Königsberg à época de Euler. No topo da figura é exibido o título "ORIGEM DOS GRAFOS DE REDE". No canto superior direito da figura é exibida uma seta apontando e nomeada "Norte", a ser utilizada como referência para o mapa. O nome do rio "PREGOLYA" é exibido dentro do curso do mesmo. Cada uma das regiões de terra delimitadas pelo curso do rio é identificada por um nó do grafo (sendo quatro no total). Cada um desses nós, por sua vez, é rotulado com o número de arestas que o conecta aos outros nós. Ao nó posicionado à esquerda do grafo estão conectadas cinco arestas; a cada um dos demais nós estão conectadas três arestas. As arestas do grafo cruzam as pontes do mapa original. Há textos explicativos junto ao mapa; no seu lado esquerdo, de cima para baixo, os textos são os seguintes: 1. "SETE PONTES DE KÖNIGSBERG (AGORA KALININGRADO, RÚSSIA)", seguido de uma seta apontando para uma das pontes do mapa; 2. "CADA 'ARESTA' REPRESENTA UMA PONTE A SER CRUZADA", seguido de uma seta apontando para uma das arestas do grafo; 3. "CADA 'NÓ' REPRESENTA UMA SEÇÃO DE TERRA", seguido de uma seta apontando para um dos nós do grafo; 4. "CADA 'GRAU' DE UM NÓ É O NÚMERO DE ARESTAS CONECTADAS A ELE. O GRAU DESTE NÓ É 3.", seguido de uma seta apontando para um dos nós do grafo com rótulo igual a três. No seu lado esquerdo, de cima para baixo, os textos são os seguintes: 5. "EULER USOU ESTE GRAFO DE INFORMAÇÃO PARA RESPONDER À PERGUNTA: 'VOCÊ CONSEGUE CAMINHAR PELA CIDADE E CRUZAR CADA PONTE APENAS UMA VEZ?';", seguido de uma seta apontando para uma aresta do grafo onde aparecem algumas figuras de pegadas (marcas de pés); 6. "A RESPOSTA É NÃO. CADA NÓ TEM UM GRAU ÍMPAR. ISSO SIGNIFICA QUE VOCÊ NÃO CONSEGUE ANDAR PARA E A PARTIR DE UM NÓ. VOCÊ PRECISA VOLTAR PARA O NÓ UMA VEZ MAIS. TODOS OS NÓS COM EXCEÇÃO DO NÓ INICIAL E FINAL DEVEM TER UM GRAU PAR PARA QUE ESSE CAMINHAMENTO SEJA POSSÍVEL.".

De forma similar às estruturas de dados baseadas em árvore, grafos permitem diferentes modalidades de caminhamento com o objetivo de visitar seus nós componentes para a resolução de algum problema relacionado. Dentre essas modalidades, destaca-se a pesquisa em largura (*Breadth-first search*, ou BFS) efetiva para que se descubra, por exemplo, o caminho mínimo entre dois nós e a pesquisa em profundidade (*Depth-first search*). Essas operações possuem um desempenho proporcional ao tamanho do grafo ao qual são aplicadas. Quando o número de nós e arestas é considerável, algoritmos voltados especificamente à descoberta do caminho mais curto (de menor peso, ou de menor custo) entre nós são preferíveis por resolverem o problema em um tempo aceitável.

Como parte da equipe não havia tido contato com a teoria dos grafos durante a sua formação, a leitura dos trabalhos de Euler e a conceituação da estrutura de dados utilizada na resolução dos problemas relacionados permitiu um nivelamento satisfatório do conhecimento de seus integrantes. A partir desse nivelamento foi possível que todos participassem da pesquisa de ferramentas e da escolha de um SGBD com suporte tanto a teoria quanto a estrutura, de forma a aproximar a implementação dos requisitos do problema de rastreamento a ser resolvido pelo projeto Hospital sem Papel. Considerando a disponibilidade do SGBD em nuvem (para experimentação), as funcionalidades oferecidas em sua versão padrão e o suporte à conectividade de ferramentas externas para interação com o modelo de dados, o **Neo4j** foi selecionado para a execução de testes e avaliação.



Neo4j é um SGBD orientado a grafos implementado na linguagem de programação Java cuja primeira versão open source foi disponibilizada em 2007. O SGBD integra a chamada **Neo4j Graph Data Platform**, que consolida o banco de dados (**Neo4j**), uma plataforma para ciência de dados (**Neo4j Graph Data Science**), uma ferramenta visual interativa para a visualização e a exploração de grafos (**Neo4j Bloom**) e uma linguagem de consulta (**Cypher Query Language**), além de conectores e integradores para uso com diferentes ferramentas externas e ferramentas de desenvolvimento. Em seu núcleo (core), o SGBD organiza o conteúdo armazenado como nós, arestas e atributos. O fato de os relacionamentos (arestas) serem armazenados explicitamente permite a obtenção de um alto desempenho na execução de queries, o que pode ser melhorado ainda mais com o uso de indexação. Além do desempenho, as consultas são escritas de forma simplificada quando comparadas a pares escritas em SGBDs relacionais que, dependendo do objetivo, demandam cadeias de JOINs para uma solução (THE NEO4J GRAPH DATA PLATFORM, 2022; NEO4J GRAPH DATABASE, 2022).



NOVAS DESCOBERTAS

1. Java é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems na década de 1990. Atualmente um produto da Oracle Corporation, a linguagem é caracterizada por ter seu código-fonte compilado para bytecode para, posteriormente, ser executado em uma JVM (*Java Virtual Machine*).
2. Para fins de experimentação e testes, a versão em nuvem do Neo4j pode ser acessada gratuitamente. Conhecida como AuraDB, a versão gratuita oferece um ambiente de cluster limitado, porém suficiente para o aprendizado da ferramenta.

A partir do cadastro de uma conta, é possível acessar o Neo4j AuraDB de forma gratuita para a criação de uma instância de Banco de Dados (BD) limitada (vide Figura 3). As limitações são aplicadas no número de nós (até 50.000) e no número de relacionamentos (até 175.000).

Figura 3 - Informações sobre a instância criada no Neo4j AuraDB, logo após o cadastro de uma nova conta e acesso / Fonte: o autor.

Descrição da Imagem: a figura exibe a tela inicial do Neo4j AuraDB com informações sobre as instâncias criadas e vinculadas à conta do usuário. Na parte superior, da esquerda para a direita, é exibido um botão com um 'X' (o qual exibe/oculta um menu localizado logo abaixo, à esquerda). Neste menu, é exibido o item "Instances" ("Instâncias"), que quando clicado permite visualizar as instâncias criadas pelo usuário. Abaixo é exibido o item "Connect" ("Conectar"), que dá acesso ao código-fonte de exemplo para conexão e acesso aos dados das instâncias disponíveis nas linguagens de programação e/ou frameworks Python, JavaScript, GraphQL, Java, Spring Boot, .NET e Go. Ao final do menu é disponibilizado o item "Feedback" ("Comentários"), que possibilita ao usuário descrever sua experiência e fazer sugestões sobre a ferramenta. Segundo na parte superior da tela é exibido o nome da ferramenta ("Neo4j AuraDB"), tendo à sua direita um botão de seleção entre as opções "AuraDB" (o SGBD orientado a grafos) e "AuraDS" (a plataforma de ciência de dados baseada em grafos). Ainda à direita é exibido o menu "GET HELP" ("CONSIGA AJUDA"), um botão com o ícone de um megafone (que quando clicado exibe anúncios e novidades sobre a ferramenta) e a foto do perfil do usuário conectado. Na parte central da tela são exibidas as instâncias mantidas pelo usuário, com o rótulo "Instances" ("Instâncias") e um botão com o rótulo "New Instance" ("Nova Instância"). Um resumo para cada instância disponível é exibido individualmente em uma caixa, onde consta o nome da instância (exemplo: "COVID-19"), se a instância é de uso gratuito (exemplo: "FREE" - "GRATUITO"), a situação quanto à sua execução (exemplo: "Running" - "Executando"), a versão do Neo4j utilizada (exemplo: "Neo4j version 4"), o número de nós disponíveis na instância (exemplo: "Nodes: 0 / 50000 (0%)"), e o número de relacionamentos disponíveis na instância (exemplo: "Relationships: 0 / 1750000 (0%)"). À direita dessas informações são disponibilizados botões com os rótulos "Explore" ("Explorar"), "Query" ("Inquirir"), "Import" ("Importar"), além de um botão com a figura de uma lixeira (o que permite excluir a instância) e outro botão com "..." como rótulo, para a execução de configurações complementares.

Para validar a utilização do Neo4j como ferramenta de auxílio ao processo de rastreamento, a equipe de TI optou por selecionar datasets construídos e disponibilizados gratuitamente na internet, se possível acompanhados de exemplos de como acessar os dados que os compõem. Após uma pesquisa de conteúdo com essas características foi selecionado o trabalho de Rik Van Bruggen, focado especificamente na atividade de **contact tracing** (rastreamento de contatos) para casos de COVID-19. O dataset utilizado consiste em dados de pessoas, locais e visitas, e disponibiliza exemplos de acesso aos grafos construídos a partir de sua importação.

Como é comum em ambientes de nuvem, o Neo4j AuraDB disponibiliza ferramentas para a importação de dados e acesso aos dados importados (via execução de queries e visualização dos grafos construídos). Para a importação do dataset citado no parágrafo anterior foi utilizada a ferramenta para execução de queries (vide Figura 4), a qual disponibiliza acesso aos comandos do SGBD para a importação de dados em diferentes formatos (bem como de comandos complementares para que os dados importados possam inicialmente ser convertidos em nós, e posteriormente conectados via relacionamentos), resultando em um ou mais grafos.

The screenshot shows the Neo4j AuraDB interface. In the top-left editor, the following Cypher code is written:

```

1 load csv with headers from
2 "https://docs.google.com/spreadsheets/u/0/d/1R-XVuynPsOWcXSderLpq3DacZdk10PZ8v6FiyGTncIE/export?format=csv&id=1R-XVuynPsOWcXSderLpq3DacZdk10PZ8v6FiyGTncIE&gid=0" as csv
3 create (:Person {id: csv.PersonId, name:csv.PersonName, healthstatus:csv.Healthstatus, confirmtimedate:datetime(csv.ConfirmedTime)});
```

Below the editor, the interface displays several sections:

- Server version:** Neo4j/4.4-aura
- Server address:** c61fcbea databases.neo4j.io:7687
- Query:** The same Cypher code as in the editor.
- Summary:** Shows the command executed: "load csv with headers from 'https://docs.google.com/spreadsheets/u/0/d/1R-XVuynPsOWcXSderLpq3DacZdk10PZ8v6FiyGTncIE&gid=0' as csv create (:Person {id: csv.PersonId, name:csv.PersonName, healthstatus:csv.Healthstatus, confirmtimedate:datetime(csv.ConfirmedTime)});".
- Response:** Displays the message: "Added 501 labels, created 501 nodes, set 2004 properties, completed after 1498 ms."

Figura 4 - Tela da ferramenta para a execução de queries no Neo4j AuraDB. Aqui, a ferramenta está sendo utilizada para importar conteúdo no formato CSV (Comma-Separated Values - Valores Separados por Vírgula) referente às pessoas a serem rastreadas, gerando como resultado um conjunto de nós de um grafo / Fonte: o autor.

Descrição da Imagem: a figura exibe a tela do Neo4j AuraDB utilizada para a execução de queries. Na parte superior da tela é disponibilizado um editor de texto onde os comandos a serem executados podem ser digitados (para este exemplo, vide conteúdo apresentado no Quadro 1). Logo abaixo deste editor são disponibilizadas duas formas de visualização dos resultados: “Table” (“Tabela”) e “Code” (“Código”). A exibição dos resultados em código disponibiliza as seguintes informações: “Server version” (“Versão do servidor” - exemplo: “Neo4j/4.4-aura”); “Server address” (“Endereço do servidor” - exemplo: “c61fcbea.databases.neo4j.io:7687”); “Query” (“Inquirição” ou “Consulta”) com o mesmo conteúdo do editor de texto; “Summary” (“Sumário” ou “Resumo”) com o mesmo conteúdo do editor de texto, seguido de outras mensagens complementares relacionadas à execução da query; e “Response” (“Resposta” - se houver). No final da tela é exibida uma mensagem com os resultados quantitativos da execução da query (exemplo: “Added 501 labels, created 501 nodes, set 2004 properties, completed after 1498 ms.” - “Adicionados 501 rótulos, criados 501 nós, atribuídas 2004 propriedades, completo após 1498 ms.”)

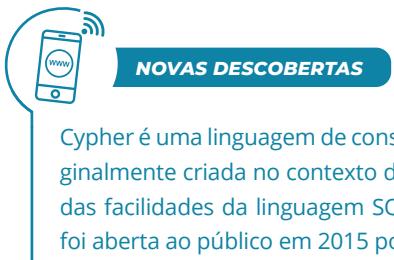
O Quadro 1 exibe o código-fonte escrito em linguagem **Cypher** utilizado na ingestão dos dados das pessoas a serem rastreadas. A linha 1 especifica que a leitura será feita a partir de uma origem em formato CSV considerando a existência de

uma linha de cabeçalho; a linha 2 indica a origem do conteúdo a ser lido – neste caso, um arquivo armazenado no Google Docs disponível publicamente; a linha 3 instrui o SGBD sobre o que fazer com o conteúdo lido. Neste caso, para cada linha do arquivo CSV, criar um nó com o rótulo “Person”, inserindo também as propriedades “id”, “name”, “healthstatus” e “confirmedtime” (onde cada propriedade assume o valor de um dos valores da linha do arquivo).

```

1: load csv with headers from
2: "https://docs.google.com/spreadsheets/u/0/d/1R-XVuynPsOWcXSder-
Lpq3DacZdk10PZ8v6FiYGTncIE/export?format=csv&id=1R-XVuynPsOWcXSder-
Lpq3DacZdk10PZ8v6FiYGTncIE&gid=0"
as csv
3: create (p:Person {id: csv.PersonId, name:csv.PersonName, health-
status:csv.Healthstatus, confirmedtime:datetime(csv.ConfirmedTime)});
```

Quadro 1 - Exemplo de código-fonte escrito em linguagem Cypher para a criação de nós de um grafo a partir do conteúdo de um arquivo CSV3. Cada nó representa uma pessoa, recebe um rótulo e armazena dados na forma de propriedades / Fonte: Bruggen ([2022], on-line).



NOVAS DESCOPERTAS

Cypher é uma linguagem de consulta declarativa orientada a grafos, originalmente criada no contexto de uso do Neo4j. Projetada para dispor das facilidades da linguagem SQL (*Structured Query Language*), Cypher foi aberta ao público em 2015 por meio do projeto openCypher.

O Quadro 2 exibe o código-fonte utilizado na ingestão dos dados dos locais frequentados pelas pessoas a serem rastreadas. É similar ao código-fonte apresentado no Quadro 1, variando a origem dos dados, o rótulo dos nós criados e as propriedades de cada nó. O Quadro 3, por sua vez, exibe o código-fonte utilizado na criação de índices a partir dos dados armazenados nos nós na forma de propriedades.

```

1: load csv with headers from
2: "https://docs.google.com/spreadsheets/u/0/d/1R-XVuynPsOWcXSderLpq3DacZdk-
10PZ8v6FiYGTncIE/export?format=csv&id=1R-XVuynPsOWcXSderLpq3DacZdk10PZ8v6Fi-
YGTncIE&gid=205425553"
as csv
3: create (p:Place {id: csv.PlaceId, name:csv.PlaceName, type:csv.PlaceType,
location:point({x: toFloat(csv.Lat), y: toFloat(csv.Long)}))};

```

Quadro 2 - Exemplo de código-fonte escrito em linguagem Cypher para a criação de nós de um grafo a partir do conteúdo de um arquivo CSV3. Cada nó representa um local, recebe um rótulo e armazena dados na forma de propriedades. Latitudes e longitudes podem ser utilizadas para georreferenciar os locais importados / Fonte: Bruggen ([2022], on-line).

```

1: create index on :Place(id);
2: create index on :Place(location);
3: create index on :Place(name);
4: create index on :Person(id);
5: create index on :Person(name);
6: create index on :Person(healthstatus);
7: create index on :Person(confirmedtime);

```

Quadro 3 - Exemplo de código-fonte escrito em linguagem Cypher para a criação de índices a partir das propriedades dos nós de um grafo, com o objetivo de agilizar consultas e o estabelecimento de relacionamentos entre os nós³ / Fonte: Bruggen ([2022], on-line).

Com os dados das pessoas e locais devidamente importados e utilizados na criação dos nós do grafo, é possível estabelecer os relacionamentos a partir dos dados das visitas. O Quadro 4 exibe o código-fonte utilizado para essa importação, no qual vale destacar a seguinte sequência de comandos (executada para cada linha do arquivo CSV):

- **linha 3:** são recuperadas as referências aos nós criados anteriormente para uma pessoa e para um local, a partir das suas respectivas propriedades “*id*” (cujos valores vêm originalmente do arquivo com os dados das visitas);

- **linha 4:** é criado um relacionamento chamado “*PERFORMS_VISIT*” entre o nó de uma pessoa (cuja referência foi recuperada na linha 3) e um novo nó que representa uma visita, o qual recebe o rótulo “Visit” e as propriedades “*id*”, “*starttime*” e “*endtime*”. A seguir, é criado um relacionamento chamado “*LOCATED_AT*” entre este novo nó e o nó de um local (cuja referência foi recuperada na linha 3);
- **linha 5:** é criado um relacionamento chamado “*VISITS*” entre o nó de uma pessoa e o nó de um local (cujas referências foram recuperadas na linha 3). Este relacionamento recebe as propriedades “*id*”, “*starttime*” e “*endtime*”;
- **linhas 6 e 7:** os nós criados na linha 4 e os relacionamentos criados na linha 5 recebem como propriedade o tempo de duração da respectiva visita, calculado a partir das propriedades “*starttime*” e “*endtime*”.

```

1: load csv with headers from
2:"https://docs.google.com/spreadsheets/d/1R-XVuynPsOWcXSderLpq3DacZdk-
10PZ8v6FiYGTncIE/export?format=csv&id=1R-XVuynPsOWcXSderLpq3DacZdk10PZ-
8v6FiYGTncIE&gid=1261126668"
as csv
3: match (p:Person {id:csv.PersonId}), (pl:Place {id:csv.PlaceId})
4:create(p)-[:PERFORMS_VISIT]->(v:Visit{id:csv.VisitId, starttime:datetime(csv.StartTime), endtime:datetime(csv.EndTime)})-[:LOCATED_AT]->(pl)
5:create(p)-[vi:VISITS{id:csv.VisitId, starttime:datetime(csv.StartTime), endtime:datetime(csv.EndTime)}]->(pl)
6: set v.duration=duration.inSeconds(v.starttime,v.endtime)
7: set vi.duration=duration.inSeconds(vi.starttime,vi.endtime);

```

Quadro 4 - Exemplo de código-fonte escrito em linguagem Cypher para a criação de relacionamentos entre os nós de um grafo, relacionamentos esses representados por arestas³ / Fonte: Bruggen ([2022], on-line).



Em um BD relacional, o relacionamento entre duas tabelas é resolvido em tempo de execução através de junções (JOINS) de diferentes tipos (INNER, LEFT, FULL), de acordo com o resultado a ser obtido. Em um BD com dados organizados na forma de grafos, relacionamentos entre nós existem na forma de arestas, podendo, inclusive, armazenar propriedades que caracterizam a conexão.

Finalmente, o Quadro 5 exibe o código-fonte utilizado para referenciar geograficamente, no escopo de cidade, país e continente, todos os locais importados a partir do código exibido no Quadro 2. São criados nós para representar uma cidade (“Antwerp”), um país (“Belgium”) e um continente (“Europe”), bem como os relacionamentos entre esses nós. Por fim, todos os locais importados anteriormente são relacionados ao nó criado para a cidade.

```

1: create (r:Region{name:"Antwerp"}) -[:PART_OF] -> (c:Country {name:"Belgium"}) -[:PART_OF] -> (co:Continent {name:"Europe"});
2: match (r:Region {name:"Antwerp"}), (pl:Place)
3: create (pl) -[:PART_OF] -> (r);

```

Quadro 5 - Exemplo de código-fonte escrito em linguagem Cypher para a criação de nós individuais e relacionamentos entre esses novos nós e nós já existentes no grafo³. / Fonte: Bruggen ([2022], on-line).

Com o grafo construído a partir dos dados das pessoas, locais e visitas, é possível explorar o seu conteúdo usando queries. Essas queries são expressas por meio de referências aos nós e aos relacionamentos, permitindo a aplicação de critérios de filtro sobre as propriedades dos nós e dos relacionamentos, bem como a recuperação do conteúdo dessas propriedades para exibição. O Quadro 6 exibe um exemplo de código-fonte para uma query escrita com o objetivo de identificar quais pessoas podem ter sido infectadas por outras pessoas. Aqui, parte-se do pressuposto de que a infecção pode ter ocorrido caso uma pessoa não infectada tenha visitado um local que tenha sido visitado também por uma pessoa infectada. As linhas a seguir merecem destaque:

- **linha 1:** são recuperadas as referências aos nós cujo valor para a propriedade “*healthstatus*” seja “*Sick*” (pessoas já infectadas);
- **linha 4:** são recuperadas as referências aos nós cujo valor para a propriedade “*healthstatus*” seja “*Healthy*” (pessoas saudáveis), que tenham frequentado os mesmos locais que uma pessoa infectada. O relacionamento chamado “*VISITS*” é utilizado para estabelecer as relações das pessoas com os locais;

- **linha 5:** são recuperados os valores de propriedades a partir dos nós que armazenam dados das pessoas já infectadas, das pessoas saudáveis, dos locais e também dos relacionamentos dessas pessoas com os locais.

```

1: match (p:Person {healthstatus:"Sick"})
2: with p
3: limit 1
4:match (p)-[v1:VISITS]->(pl:Place)<-[v2:VISITS]-(p2:Person
{healthstatus:"Healthy"})
5: return p.name as Spreader, v1starttime as SpreaderStarttime,
v1endtime as SpreaderEndtime, pl.name as PlaceVisited, p2.name
as Target, v2starttime as TargetStarttime, v2endtime as
TargetEndtime;
```

Quadro 6 - Exemplo de código-fonte escrito em linguagem Cypher para a consulta de nós e relacionamentos em um grafo³ / Fonte: Bruggen ([2022], on-line).

O resultado da execução da query exibida no Quadro 6 pode ser visualizado na Figura 5. Como o retorno da query é formado por um conjunto de dados de propriedades, a exibição padrão assume o formato de uma tabela. Com alterações sutis (que podem ser visualizadas no Quadro 7), a consulta retorna o grafo resultante da aplicação dos critérios de filtro e relacionamentos. A Figura 6 exibe o grafo resultante da execução da query modificada.

```

1: match (p:Person {healthstatus:"Sick"})
2: with p
3: limit 1
4: match path = (p)-[v1:VISITS]->(pl:Place)<-[v2:VISITS]-(p2:Person
{healthstatus:"Healthy"})
5: return path;
```

Quadro 7 - Exemplo de código-fonte escrito em linguagem Cypher para a consulta de nós e relacionamentos em um grafo, retornando a representação pictórica da estrutura de dados organizada em nós e relacionamentos³ / Fonte: Bruggen ([2022], on-line).

Spreader	SpreaderStarttime	SpreaderEndtime	PlaceVisited	Target	Targetstarttime	TargetEndtime
"Landyn Greer"	"2022-04-21T03:39:58Z"	"2022-04-21T16:01:31Z"	"Place nr 37"	"Serenity Jacobs"	"2022-05-04T04:20:06Z"	"2022-05-04T09:10:13Z"
"Landyn Greer"	"2022-04-21T03:39:58Z"	"2022-04-21T16:01:31Z"	"Place nr 37"	"Paisley Brock"	"2022-04-09T19:08:00Z"	"2022-04-09T23:24:04Z"
"Landyn Greer"	"2022-04-21T03:39:58Z"	"2022-04-21T16:01:31Z"	"Place nr 37"	"Semaj Roach"	"2022-04-20T14:01:14Z"	"2022-04-21T00:42:19Z"
"Landyn Greer"	"2022-04-21T03:39:58Z"	"2022-04-21T16:01:31Z"	"Place nr 37"	"Marlie Dunlap"	"2022-04-12T17:03:02Z"	"2022-04-13T05:56:41Z"
"Landyn Greer"	"2022-04-21T03:39:58Z"	"2022-04-21T16:01:31Z"	"Place nr 37"	"Paige Richards"	"2022-04-29T12:30:00Z"	"2022-04-30T02:05:14Z"
"Landyn Greer"	"2022-04-21T03:39:58Z"	"2022-04-21T16:01:31Z"	"Place nr 37"	"Duncan Garcia"	"2022-04-23T03:21:49Z"	"2022-04-26T03:40:16Z"

Figura 5 - Execução de query com retorno em formato de tabela. Cada coluna exibe os dados de uma propriedade, de acordo com o retorno definido no texto da consulta / Fonte: o autor.

Descrição da Imagem: a figura exibe a tela do Neo4j AuraDB utilizada para a execução de queries. Na parte superior da tela é disponibilizado um editor de texto com a query a ser executada (idêntica àquela apresentada no Quadro 6). Logo abaixo deste editor é disponibilizado o resultado da execução da query no formato de uma tabela. São exibidas seis linhas numeradas com as seguintes colunas: "Spreader" (nome da pessoa infectada que poderia "espalhar" o vírus), com o valor "Landyn Greer" em todas as linhas; "SpreaderStarttime" (data/hora em que a pessoa infectada iniciou a visita a um determinado local), com o valor "2022-04-21T03:39:58Z" em todas as linhas; "SpreaderEndtime" (data/hora em que a pessoa infectada encerrou a visita a um determinado local), com o valor "2022-04-21T16:01:31Z" em todas as linhas; "Place-Visited" (nome do local visitado), com o valor "Place nr 37" em todas as linhas; "Target" (nome da pessoa que pode ter sido infectada por ter frequentado o mesmo local), com os seguintes valores para as linhas: "Serenity Jacobs", "Paisley Brock", "Semaj Roach", "Marlie Dunlap", "Paige Richards" e "Duncan Garcia"; "TargetStarttime" (data/hora em que a pessoa saudável iniciou a visita ao mesmo local visitado pela pessoa infectada), com os seguintes valores para as linhas: "2022-05-04T04:20:06Z", "2022-04-09T19:00:00Z", "2022-04-20T14:01:14Z", "2022-04-12T17:03:02Z", "2022-04-29T12:30:00Z", "2022-04-25T05:21:40Z"; e TargetEndtime" (data/hora em que a pessoa saudável encerrou a visita ao mesmo local visitado pela pessoa infectada), com os seguintes valores para as linhas: "2022-05-04T09:10:13Z", "2022-04-09T23:24:04Z", "2022-04-21T00:42:19Z", "2022-04-13T05:56:41Z", "2022-04-30T02:05:14Z", "2022-04-26T03:40:16Z". A tabela apresenta uma barra de rolagem que permite a visualização dos demais resultados da consulta. Ao final da tela é exibida a mensagem "Started streaming 172 records after 1 ms and completed after 10 ms." ("Iniciada a transmissão de 172 registros após 1 ms, completa após 10 ms.").

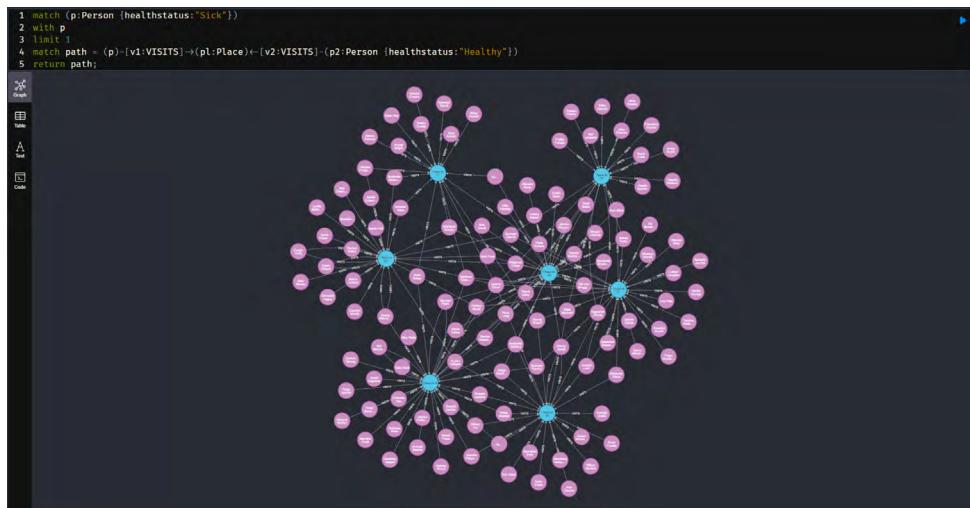


Figura 6 - Execução de query com o retorno em formato de grafo. Os nós que representam pessoas são coloridos de forma diferenciada dos nós que representam locais, e os relacionamentos aparecem como arestas do grafo / Fonte: o autor.

Descrição da Imagem: a figura exibe a tela do Neo4j AuraDB utilizada para a execução de queries. Na parte superior da tela é disponibilizado um editor de texto com a query a ser executada (similar àquela apresentada no Quadro 7). Logo abaixo deste editor é disponibilizado o resultado da execução da query no formato de um grafo. São exibidos neste grafo 125 nós (seis deles correspondendo a locais, e os restantes a pessoas infectadas ou saudáveis), todos como círculos, e 344 relacionamentos indicando visitas de pessoas a diferentes locais (todos como setas direcionadas das pessoas aos locais).

O resultado obtido pela execução da query exibida nos Quadros 6 e 7 considera que visitas feitas aos mesmos locais por pessoas saudáveis e infectadas podem levar à infecção das primeiras. Apesar de válida, a suposição feita na pesquisa leva a um resultado abrangente que não considera o fato de que o tempo em que as visitas ocorreram é significativo para que uma infecção ocorra. A probabilidade de que uma pessoa saudável seja infectada é maior quando ocorre o contato dessa pessoa com outra infectada, o que demanda ambas estarem em um mesmo local ao mesmo tempo. Para restringir o resultado desta query e identificar quando ocorreram possíveis contatos diretos entre as pessoas, o Quadro 8 exibe uma consulta que atende a estes critérios na qual valem destacar as linhas a seguir:

- **linha 1:** são recuperadas as referências aos nós cujo valor para a propriedade “*healthstatus*” seja “*Sick*” (pessoas já infectadas), ao nós referentes aos locais visitados por essas pessoas, e aos relacionamentos entre essas pessoas e locais;
- **linha 4:** o caminho no grafo formado pelo relacionamento entre pessoas infectadas e locais visitados por essas pessoas é complementado pelo relacionamento desses locais com pessoas saudáveis (nós cuja propriedade “*healthstatus*” seja “*Healthy*”);
- **linha 5:** a partir do caminho completo formado na linha 4, são calculados os valores necessários à verificação de uma sobreposição dos tempos de visita das pessoas saudáveis com as pessoas infectadas. Uma sobreposição ocorre quando o horário final de visita de uma pessoa saudável é igual ou superior ao horário inicial de visita de uma pessoa infectada.

```

1: match (p:Person {healthstatus:"Sick"})-[v1:VISITS]->(pl:Place)
2: with p,v1,pl
3: limit 10
4:match path = (p)-[v1]->(pl)<-[v2:VISITS]-(p2:Person {healthstatus:"Healthy"})
5: with path, apoc.coll.max([v1starttime.epochMillis, v2starttime.epochMillis]) as maxStart,
6: apoc.coll.min([v1endtime.epochMillis, v2endtime.epochMillis]) as minEnd
7: where maxStart <= minEnd
8: return path;

```

Quadro 8 - Exemplo de código-fonte escrito em linguagem Cypher para a consulta de nós e relacionamentos em um grafo, retornando a representação pictórica da estrutura de dados organizada em nós e relacionamentos, de acordo com um critério especificado via cláusula WHERE³.

Fonte: Bruggen ([2022], on-line).

A Figura 7 exibe o grafo resultante da execução da query relacionada no Quadro 8. São retornados 18 nós (dos quais 11 são pessoas e sete são locais), conectados por meio de 22 relacionamentos correspondentes a visitas.

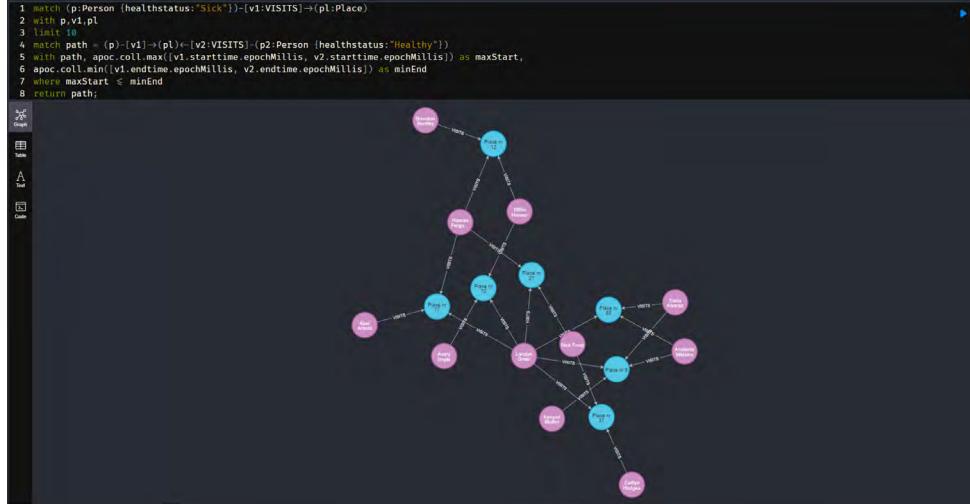


Figura 7 - Execução de query com o retorno em formato de grafo. Os nós que representam pessoas são coloridos de forma diferenciada dos nós que representam locais, e os relacionamentos aparecem como arestas do grafo. O resultado é restrito às visitas de pessoas saudáveis e pessoas infectadas feitas aos mesmos locais cujos períodos de tempo possuem alguma coincidência / Fonte: o autor.

Descrição da Imagem: a figura exibe a tela do Neo4j AuraDB utilizada para a execução de queries. Na parte superior da tela é disponibilizado um editor de texto com a query a ser executada (similar àquela apresentada no Quadro 8). Logo abaixo deste editor é disponibilizado o resultado da execução da query no formato de um grafo. São exibidos neste grafo 18 nós (sete deles correspondendo a locais, e os restantes a pessoas infectadas ou saudáveis), todos como círculos, e 22 relacionamentos indicando visitas de pessoas a diferentes locais (todos como setas direcionadas das pessoas aos locais), que ocorreram em períodos de tempo com alguma coincidência.

Além da identificação de pessoas infectadas e dos seus contatos, o conhecimento de quais locais foram visitados por pessoas infectadas é importante para que sejam demarcadas as regiões que podem potencializar a disseminação de um vírus. Esses locais, conhecidos como *hotspots*, são candidatos a uma monitoração mais ativa, resultando em um maior controle de frequência. O código-fonte exibido no Quadro 9 permite identificar os hotspots existentes no *dataset* a partir de visitas feitas por pessoas infectadas. Essa identificação compara as datas de confirmação da infecção da pessoa com a data de início da visita para garantir que sejam selecionadas apenas pessoas capazes de transmitir o vírus. Diferentemente das *queries* anteriores, o resultado de sua execução gera mais de um grafo como resultado (vide Figura 8).

```

1: match (p:Person {healthstatus:"Sick"})-[visited]->(pl:Place)
2: where p.confirmedtime < visitedstarttime
3: return p, visited, pl
4: limit 10;

```

Quadro 9 - Exemplo de código-fonte escrito em linguagem Cypher para a consulta de nós e relacionamentos, retornando mais de um grafo simultaneamente³ / Fonte: Bruggen ([2022], on-line).

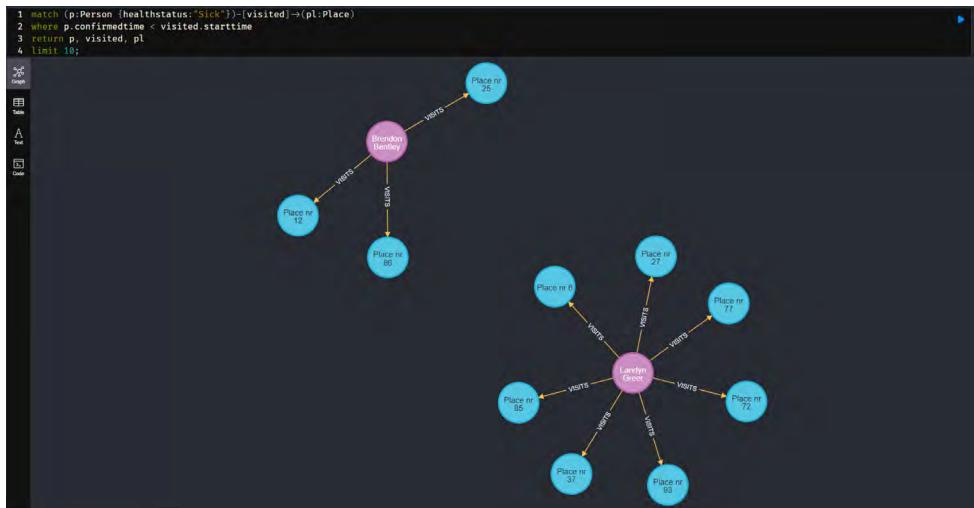


Figura 8 - Execução de query com o retorno simultâneo de dois grafos. Os nós que representam pessoas são coloridos de forma diferenciada dos nós que representam locais, e os relacionamentos aparecem como arestas dos grafos. O resultado é restrito aos locais visitados por pessoas infectadas
Fonte: o autor.

Descrição da Imagem: a figura exibe a tela do Neo4j AuraDB utilizada para a execução de queries. Na parte superior da tela é disponibilizado um editor de texto com a query a ser executada (similar àquela apresentada no Quadro 9). Logo abaixo deste editor é disponibilizado o resultado da execução da query no formato de grafos. São exibidos dois grafos: o primeiro com quatro nós (sendo uma pessoa e três locais); e o segundo com oito nós (sendo uma pessoa e sete locais). As pessoas exibidas estão infectadas, e os locais visitados por essas pessoas são identificados para um possível monitoramento.

Há, também, a possibilidade de se destacarem os principais hotspots com base em uma maior frequência de visitas por pessoas infectadas. O código-fonte exibido no Quadro 10 auxilia na seleção desses locais, com destaque para as seguintes linhas:

- **linha 1:** são recuperadas as referências aos nós cujo valor para a propriedade “healthstatus” seja “Sick” (pessoas já infectadas), ao nós referentes aos locais visitados por essas pessoas, e aos relacionamentos entre essas pessoas e locais;
- **linha 2:** locais são agrupados por seu nome, e é calculada uma contagem de quantas visitas recuperadas na linha 1 foram feitas a cada um desses locais;
- **linha 3:** o resultado é ordenado pelo número de visitas de forma decrescente, para que os locais mais visitados sejam considerados primeiro.

```

1: match (p:Person {healthstatus:"Sick"})-[v:VISITS]->(pl:Place)
2: with distinct pl.name as placename, count(v) as nrofsickvisits, pl
3: order by nrofsickvisits desc
4: limit 10
5: match path = (pl)<-[v]-(p:Person)
6: return path;

```

Quadro 10 - Exemplo de código-fonte escrito em linguagem Cypher para a consulta de nós e relacionamentos, utilizando uma função de contagem para a ordenação dos resultados³.

Fonte: Bruggen ([2022], on-line).

O grafo resultante desta consulta pode ser visualizado na Figura 9. Nele são destacados os oito principais hotspots identificados a partir do dataset (selecionados com base no número de visitas recebidas), juntamente com um subconjunto das pessoas infectadas que os visitaram.

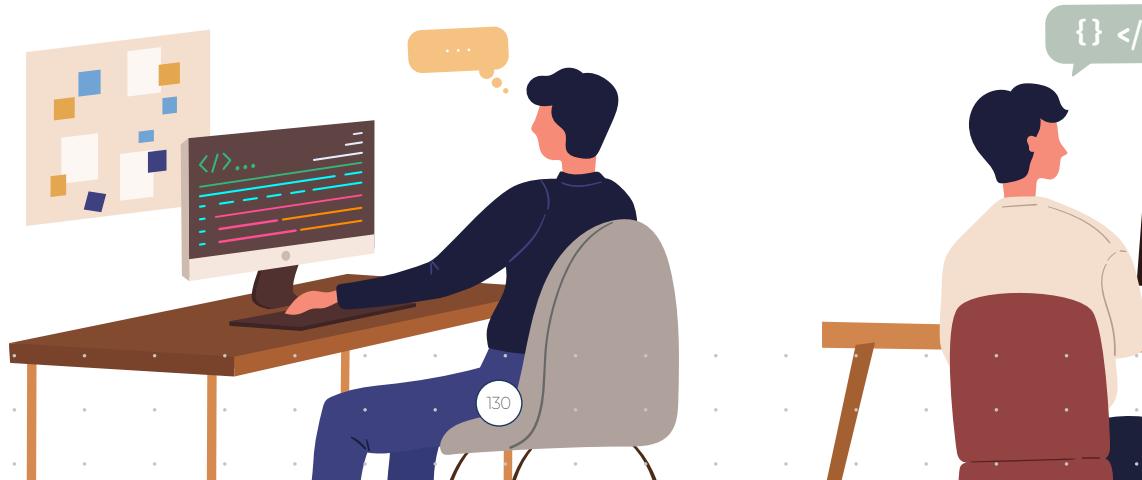




Figura 9 - Execução de query com o retorno de um grafo identificando hotspots. Os nós que representam pessoas são coloridos de forma diferenciada dos nós que representam locais, e os relacionamentos aparecem como arestas do grafo. O resultado é parcial, devido a configurações de exibição feitas na ferramenta para a execução de queries, e exibe os locais que receberam um maior número de visitas de pessoas infectadas / Fonte: o autor.

Descrição da Imagem: a figura exibe a tela do Neo4j AuraDB utilizada para a execução de queries. Na parte superior da tela é disponibilizado um editor de texto com a query a ser executada (similar àquela apresentada no Quadro 10). Logo abaixo deste editor é disponibilizado o resultado da execução da query no formato de um grafo. O grafo é exibido com 300 nós (sendo oito locais e 292 pessoas infectadas); esses oito locais são os principais hotspots do dataset por terem recebido um maior número de visitas de pessoas infectadas.



Nos exemplos anteriores o relacionamento entre as pessoas saudáveis e as pessoas infectadas era definido por uma visita a um local comum. Essa visita pode ser utilizada para complementar o grafo existente com um novo tipo de relacionamento, desta vez construído diretamente entre pessoas. O critério a ser utilizado é o mesmo daquele visto no Quadro 8: se houver sobreposição nos seus horários de visita, pessoas infectadas e pessoas saudáveis podem ter seus respectivos nós conectados diretamente. O Quadro 11 exibe o código-fonte responsável por criar esse novo relacionamento (chamado “*MEETS*”), o qual recebe como propriedade o tempo de sobreposição dos horários de visita (ou seja, por quanto tempo as pessoas envolvidas puderam ter ficado em contato). A Figura 10, por sua vez, exibe uma nova visão para o grafo onde são considerados apenas os relacionamentos entre pessoas.

```

1: match (p1:Person)-[v1:VISITS]->(pl:Place)<-[v2:VISITS]-(p2:Person)
2: where id(p1)<id(p2)
3: with p1, p2, apoc.coll.max([v1starttime.epochMillis, v2starttime.
epochMillis]) as maxStart,
4: apoc.coll.min([v1endtime.epochMillis, v2endtime.epochMillis]) as minEnd
5: where maxStart <= minEnd
6: with p1, p2, sum(minEnd-maxStart) as meetTime
7: create (p1)-[:MEETS {meettimes: duration({seconds: meetTime/1000})}]->(p2);

```

Quadro 11 - Exemplo de código-fonte escrito em linguagem Cypher para a construção de um novo relacionamento entre os nós de um grafo existente a partir de relacionamentos prévios³.

Fonte: Bruggen ([2022], on-line).

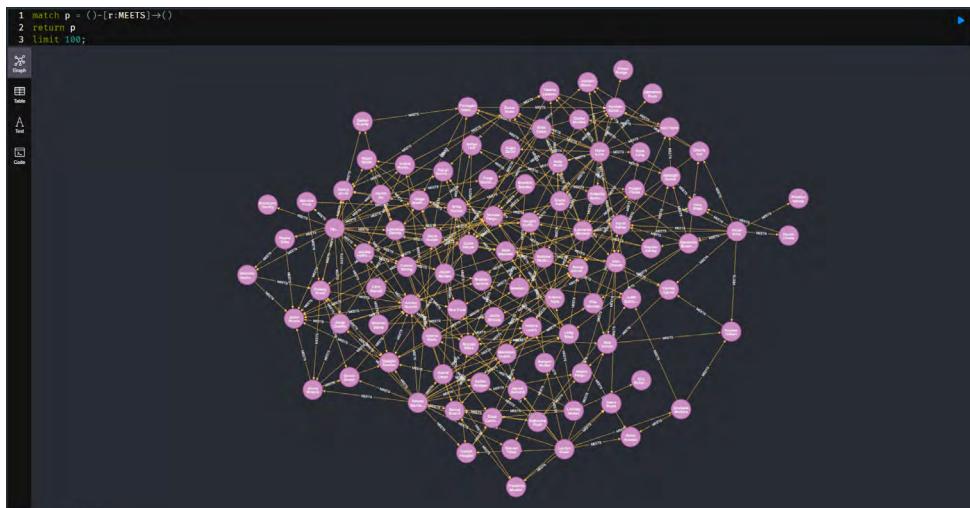


Figura 10 - Execução de query com o retorno de um grafo com relações interpessoais diretas O resultado é parcial devido a uma limitação de retorno incluída como parte da consulta / Fonte: o autor.

Descrição da Imagem: a figura exibe a tela do Neo4j AuraDB utilizada para a execução de queries. Na parte superior da tela é disponibilizado um editor de texto com a query a ser executada (composta por três linhas: "match p = ()-[r:MEETS]->()", "return p" e "limit 100;"). O grafo resultante é exibido com 97 nós (todos representando pessoas) e 100 relacionamentos, devido à limitação presente no texto da query.

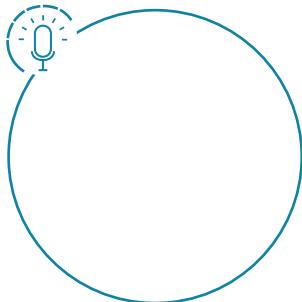
Em conjunto com as ferramentas disponibilizadas on-line pelo *Neo4j AuraDB*, opções para uso em ambiente desktop podem ser selecionadas visando à execução de *analytics* ou mesmo georreferenciamento. Este último é atendido pela combinação do *Neo4j Desktop* com *neomap*, uma aplicação construída com o objetivo de visualizar nós dotados de atributos geográficos (comumente, latitude e longitude). A partir de uma query como a apresentada no Quadro 12, os pontos a serem georreferenciados podem ser plotados para visualização e contextualização.

```

1: match (pl:Place)
2: return pl.location.x as latitude, pl.location.y as longitude
3: limit 10000

```

Quadro 12 - Exemplo de código-fonte escrito em linguagem Cypher para a recuperação das coordenadas geográficas armazenadas em nós, com o objetivo de utilizá-las para georreferenciamento. Fonte: Bruggen ([2022], on-line).



Os SGBDs orientados a grafos são diferenciados por persistir, além dos dados dos nós – correspondentes aos registros de um BD relacional – os dados dos relacionamentos entre eles. Com isso, é possível complementar esses relacionamentos com propriedades que os definem. Acompanhe em nosso podcast os comentários feitos sobre essa diferença tão importante nos quesitos de desempenho e representação de dados. Não perca!

O *dataset* selecionado para os experimentos com o SGBD orientado a grafos Neo4j permitiu à equipe de TI simular cenários muito próximos ao uso cotidiano esperado. O processo de rastreamento prospectivo tanto de pessoas quanto de locais foi considerado atendido, com a ferramenta entregando resultados úteis tanto para o acompanhamento de possíveis infecções quanto para a identificação e a sinalização dos locais de risco (*hotspots*), que contribuem como pontos de disseminação do agente infeccioso.

Com os resultados positivos obtidos, grafos e as ferramentas que os implementam passaram a compor o ecossistema tecnológico que será adotado pela rede de hospitais. Ficou claro aos participantes dos testes e aos seus superiores que a existência de relações entre dados, relações essas que por si só possuem dados próprios que as definem, compõem *datasets* passíveis de serem tratados utilizando essas ferramentas. O rastreamento de infecções é apenas um dos cenários onde essa abordagem é factível; cabe aos especialistas de domínio juntamente com os analistas da equipe de TI identificar similaridades entre problemas ainda não atendidos e os cenários construídos para teste, aproveitando a expertise adquirida na resolução dos primeiros.

AGORA É COM VOCÊ



1. Com relação ao processo de rastreamento de contatos, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
 - a) A execução do processo de rastreamento gera resultados suficientes e pode ser interrompida ao final da execução de uma primeira interação (identificar apenas os contatos diretos de uma pessoa infectada).
 - b) No rastreamento prospectivo, parte-se de um caso confirmado para que seja possível identificar uma fonte de infecção.
 - c) O rastreamento caracteriza-se pela identificação dos contatos feitos por um indivíduo infectado com outras pessoas, encaminhando essas pessoas à quarentena para acompanhamento durante um período.
 - d) Rastrear contatos é um processo aplicável exclusivamente a epidemias.
 - e) No rastreamento retrospectivo buscam-se os contatos de um indivíduo infectado para acompanhamento.
2. Com relação aos processos de rastreamento retrospectivo e prospectivo no que compete à identificação de relações, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
 - a) Consideram-se não apenas as relações interpessoais.
 - b) Consideram-se relações que ocorrem apenas com contato indireto.
 - c) Consideram-se relações interpessoais sem levar em conta visitas aos mesmos locais.
 - d) Consideram-se apenas relações entre locais.
 - e) Consideram-se apenas relações entre locais e contato indireto.
3. Com relação à teoria dos grafos e à estrutura de um grafo em si, é CORRETO afirmar que:
 - a) O conjunto de nós de um grafo pode ser infinito.
 - b) Os grafos podem ser classificados como direcionados (quando a ordem em que os nós são visitados importa) e não direcionados (quando essa ordem não existe).
 - c) É possível construir um grafo sem nós.
 - d) Uma mesma aresta pode conectar simultaneamente até quatro nós.
 - e) O grau de um nó corresponde ao número de conexões de entrada que o nó recebe. Desta forma, é válido apenas para grafos direcionados.



4. Observando as queries apresentadas no decorrer da unidade, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
- a) Na notação "(p:Person {id:csv.PersonId})", id corresponde ao rótulo do nó p.
 - b) A notação "[vi:VISITS {id:csv.VisitId...}" (as reticências indicam que o comando continua) é inválida, visto que uma aresta não aceita propriedades.
 - c) Índices são criados apenas a partir dos valores dos rótulos dos nós e relacionamentos, e não a partir de suas propriedades.
 - d) A notação "-[v:VISITS]->" implica na criação de uma aresta (relacionamento) entre dois nós, sendo que essa aresta é direcionada do nó indicado à esquerda para o nó indicado à direita.
 - e) O resultado das queries pode ser visualizado apenas na forma de um grafo.

Flexibilidade de Armazenamento em Banco de Dados Orientado a Colunas

Dr. Alexandre Savaris

OPORTUNIDADES DE APRENDIZAGEM

Nesta unidade, a variedade de dados em cenários de Big Data é revisada sob a ótica de bancos de dados orientados a colunas. É apresentado o modelo de dados decomposto, uma tentativa de flexibilizar esquemas sem abandonar o padrão já estabelecido. Embora não utilizado na prática (devido a suas limitações de desempenho), esse modelo foi base para o desenvolvimento de ferramentas como Bigtable e HBase, apresentadas em termos de arquitetura e acesso. Exemplos de instruções para esse acesso são apresentadas e comentadas.



A variedade (variety) em cenários de Big Data impõe desafios à aquisição, organização, armazenamento e acesso a dados cujas fontes são diversas, os formatos não são padronizados e o conteúdo pode ser altamente estruturado, semiestruturado ou mesmo não estruturado (MONICA; KUMAR, 2013). A aquisição impacta na infraestrutura de entrada, que precisa se adaptar a diferentes protocolos e suportar uma vazão (*throughput*) adequada às necessidades das aplicações que consumirão os dados recebidos. Uma vez internalizados, os dados recebidos passam a ser organizados de acordo com a necessidade de uso, podendo ser escalonados para consumo em etapas e também ser pré-processados para atender a demandas internas dos consumidores. O armazenamento dos dados recebidos e processados, por sua vez, segue as políticas internas e a legislação vigente que regula as questões de disponibilização e temporalização, ou seja, por quanto tempo os dados estarão disponíveis e por quem esses dados poderão ser acessados nesse tempo. Finalmente, o acesso aos dados adquiridos, organizados e armazenados demanda recursos suficientes para que o tempo de resposta seja adequado às necessidades dos clientes; esse tempo de resposta varia de acordo com o tipo de operação executada, desde consultas interativas até a emissão de relatórios em lote (batch) com tempo de execução de minutos ou horas.

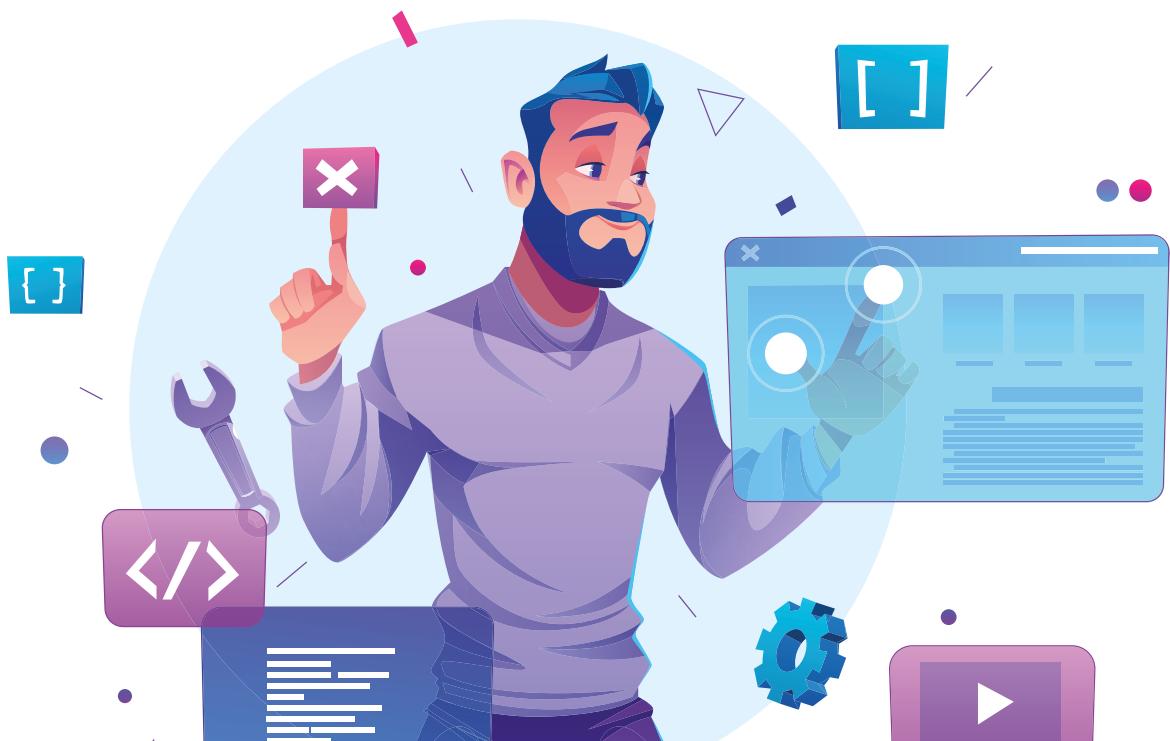


É comum que soluções para essas demandas sejam visualizadas sob a perspectiva do **modelo de dados relacional**, onde todo e qualquer dado é organizado na forma de relações (tabelas), atributos (campos) e tuplas (registros) seguindo um esquema rígido. O fato de esse modelo ser implementado por um sem-número de Sistemas Gerenciadores de Banco de Dados (SGBDs), além de disponibilizar uma linguagem de consulta de conhecimento geral (SQL - *Structured Query Language*), favorece essa perspectiva. No entanto, apesar da aparente facilidade em se resolver as questões de aquisição, organização, armazenamento e acesso a dados por meio desse modelo, problemas relacionados à escalabilidade e flexibilidade surgem com o aumento dos datasets, com a especificação de limites de tempo para recuperação de dados e respostas a consultas e com a necessidade de se armazenarem dados para os quais não se conhece, previamente, a estrutura.

Em sistemas de informação em saúde, especificamente aqueles onde são registradas as atividades relacionadas ao diagnóstico médico por imagem, a multiplicidade de formatos aliada à geração de datasets cada vez maiores (oriundos de equipamentos médicos com uma maior capacidade e uma maior qualidade de aquisição) demandam soluções adaptáveis adequadas a necessidades de uso observadas em postos de saúde, Unidades de Pronto Atendimento (UPAs), clínicas

e hospitais. A aquisição de datasets heterogêneos nesses estabelecimentos é um passo importante para que, em algum momento, possa ser feita a consolidação desses dados com uma visão centrada no paciente. Assim, independentemente de onde um exame tenha sido feito, ele poderia ser recuperado para compor um Prontuário Eletrônico do Paciente (PEP) unificado, com a possibilidade de evoluir para a construção de Registros Eletrônicos em Saúde (RES) gerenciados por Sistemas de Registro Eletrônico em Saúde (S-RES) (RESOLUÇÃO, 2022; CARTILHA, 2022). A escolha de tecnologias adequadas para uma correta representação dessa heterogeneidade, aliada às necessidades de escalabilidade e desempenho de acesso, então, é fundamental; o resultado da escolha orienta a adoção das ferramentas que, por sua vez, disponibilizam funcionalidades para atender aos cenários propostos.

Caro(a) aluno(a), você já consegue perceber a importância da utilização dos bancos de dados em vários tipos de empresas, certo? Essa percepção é facilitada considerando o volume de dados que é adquirido em curtos intervalos de tempo (por exemplo, em hospitais) e como as soluções NoSQL contribuem de forma positiva no armazenamento desses dados. Continue seus estudos para auxiliar Luisa (a CTO – *Chief Technology Officer* – da rede de hospitais), e Marco, o DBA (Database Administrator - Administrador de Banco de Dados), juntamente com o restante da equipe de TI, no desenvolvimento do projeto do Hospital sem Papel!



Há anos, esforços de padronização voltados aos dados de saúde de um indivíduo são conduzidos por entidades como o Colégio Americano de Radiologia (*American College of Radiology - ACR*)¹, a Associação Nacional de Fabricantes de Equipamentos Elétricos (*National Electrical Manufacturers Association - NEMA*)², o Instituto Americano de Padrões Nacionais (*American National Standards Institute - ANSI*)³, a Organização Internacional para Padronização (*International Organization for Standardization - ISO*)⁴, e a *Health Level Seven International (HL7)*⁵, resultando em especificações como DICOM (*Digital Imaging and Communications in Medicine*, um padrão reconhecido internacionalmente para a aquisição e comunicação de imagens médicas digitais)⁶ e o HL7® FHIR® (*HL7 Fast Healthcare Interoperability Resources*)⁷, um modelo de representação para dados a serem trafegados na forma de recursos (resources), além de interfaces RESTful (*Representational State Transfer*) para a comunicação – envio e recebimento – de instâncias desses recursos entre pontos interessados em interoperar dados de seus ecossistemas em saúde. A adoção desses padrões e especificações facilita a interoperabilidade de conteúdo, permitindo que diferentes sistemas de informação possam adquirir e transmitir/receber esses dados de forma facilitada. Apesar dos padrões estabelecidos, porém, diferenças continuam presentes – o que pode ser observado na diferença de conteúdo entre exames distintos, por exemplo, ultrassonografia e tomografia computadorizada. Exames distintos capturam dados distintos, que acabam sendo armazenados de forma diferenciada mesmo respeitando os padrões citados anteriormente.

Para o projeto Hospital sem Papel, abordar essa heterogeneidade de conteúdo é mandatório. As instituições integrantes da rede atendem a diferentes especialidades, gerando como resultado dados alfanuméricos estruturados, semiestruturados e não estruturados, bem como dados multimídia como imagens e vídeos. Em conjunto, esses dados (quando analisados) indicam a condição de saúde dos indivíduos atendidos, sendo parte importante do seu histórico de saúde. Manter esse histórico é essencial para que um acompanhamento de médio e longo prazo seja possível, e mais um desafio para Luísa e a equipe de TI. A formação de seus integrantes no que compete armazenamento de dados é, essencialmente, fundamentada no modelo relacional. Como citado, porém, as limitações desse modelo podem inviabilizar o cenário heterogêneo a ser atendido, e soluções alternativas devem ser pelo menos consideradas. Cabe, então, à equipe como um todo a pesquisa e a identificação de modelos de dados alternativos que permitam

adaptar-se a conteúdos para os quais não se conhece, a priori, uma estrutura. O cenário é desafiador quando comparado com aqueles atendidos prontamente pelo modelo relacional, em que um esquema fixo é suficiente para o atendimento do problema. A adoção de um esquema fixo para o armazenamento de dados cuja estrutura é desconhecida pode implicar em sérios problemas de escalabilidade e desempenho, e deve ser substituída por uma solução maleável e adaptável aos datasets gerados no decorrer do tempo.

A demanda por se armazenar dados cuja estrutura é desconhecida levou a equipe de TI a tentar entender como seria possível estender (ou mesmo melhorar) o modelo relacional com o intuito de facilitar a persistência desse conteúdo. Ficou claro a todos que essa melhoria ou extensão não é factível, visto que em essência o modelo relacional se apoia em um esquema considerado por muitos como imutável, ou seja, uma vez definido, permanece com a mesma estrutura indefinidamente. Apesar de sua dita imutabilidade, alterações e/ou correções de esquema são possíveis, porém, feitas a partir de demandas muito específicas e em casos de real necessidade. Assim, como encaminhamento, foi determinado a todos que se atualizassem em relação aos **SGBDs orientados a colunas** (SILVA *et al.*, 2021) (uma opção que surgiu quando Marco, o DBA da equipe, leu um artigo sobre a decomposição do modelo relacional em colunas) (COPELAND; KHOSHAFFIAN, 1985). O modelo apresentado no artigo, apesar de utilizar tabelas de estrutura fixa, permite dinamizar o uso de colunas (campos) pela decomposição de tabelas em tabelas menores. Apesar de eficiente para a decomposição, o uso do modelo implica em um custo elevado no momento da recuperação de dados relacionados.

Para auxiliar a equipe de TI em sua busca, execute uma pesquisa na internet com o objetivo de diferenciar SGBDs orientados a linhas e SGBDs orientados a colunas. Essa diferenciação é importante e pode esclarecer a equipe o porquê de um SGBD que ofereça flexibilidade de esquema pode ser a solução procurada para o problema da rede de hospitais.

Pode parecer um contrassenso afirmar que um SGBD relacional é uma ferramenta flexível e adaptável a diferentes tipos de problemas, visto que a sua tecnologia é direcionada à disponibilização de um esquema rígido para o armazenamento de dados. Essa flexibilidade diz respeito à liberdade dada aos projetistas de definir tantas tabelas quanto necessário, bem como a forma de relacionar essas tabelas entre si. De forma complementar, a diversidade de tipos de dados ofertada também facilita o uso da ferramenta, permitindo que letras, números,

documentos, imagens, sons (entre outros) possam ser armazenados de forma similar. A rigidez de esquema, no entanto, permanece: cada tabela criada possui uma estrutura fixa pré-definida; por um lado, favorece o desenvolvimento de soluções que utilizam o banco de dados (BD) – as soluções sabem exatamente o que irão encontrar em termos de estrutura, descartando “surpresas”. Por outro, qualquer necessidade que extrapole o esquema definido vai demandar alterações estruturais, as quais são onerosas e perigosas, se executadas de forma leviana.

Em seu diário de bordo, procure registrar a sua opinião sobre a rigidez de esquemas no modelo relacional, e como você avalia que essa rigidez pode influenciar na adoção de SGBDs que implementam esse modelo para a solução de problemas. Mais tarde, compare as características desse modelo com o modelo orientado a colunas que será apresentado no decorrer do módulo - identificando as vantagens e desvantagens de um modelo com relação ao outro. A sua contribuição dará suporte à equipe de TI na defesa da escolha de uma solução (relacional ou não) que auxilie o gerenciamento de datasets heterogêneos provenientes dos hospitais que participam do projeto.

DIÁRIO DE BORDO

Em modelos de dados nos quais há uma flexibilização de esquema, pode-se observar que a ideia de instância ou objeto do mundo real a ser armazenado continua existindo da mesma forma que em modelos de dados com esquemas rígidos. O que muda é a **composição** dessas instâncias ou objetos: o número de atributos (campos) pode mudar de instância para instância, e o armazenamento ocorre sem que sejam geradas lacunas (como aquelas observadas no modelo relacional quando campos de um registro não possuem conteúdo).

O artigo lido por Marco propõe que a estrutura das tabelas de um BD relacional seja revista, de forma que por meio de uma operação de decomposição seja possível obter um resultado onde as lacunas são eliminadas e cada instância ou objeto seja armazenado de acordo com seu conteúdo particular. O resultado desse processo de decomposição pode ser visualizado na Figura 1. Nele, uma tabela original com uma chave substituta (*surrogate key*) e outros três campos é decomposta em três tabelas, uma para cada campo. A chave substituta é replicada em cada uma das tabelas resultantes para permitir a recomposição dos dados completos para cada registro armazenado por meio de junções.



R	sur	a1	a2	a3
s1	v11	v21	v31	
s2	v12	v22	v32	
s3	v13	v23	v33	

a1	sur	val	a2	sur	val	a3	sur	val
s1	v11		s1	v21		s1	v31	
s2	v12		s2	v22		s2	v32	
s3	v13		s3	v23		s3	v33	

Figura 1 - Exemplo do modelo de dados decomposto. Nele, uma tabela original dá origem a n tabelas que a substituem, onde n corresponde ao seu número de campos. A chave substituta existente na tabela original é replicada para fins de recomposição de registros completos

Fonte: Copeland e Khoshafian (1985, [s.p.]).

Descrição da Imagem: no topo da figura é exibida a tabela original, antes do processo de decomposição, formada por cinco colunas. A primeira coluna apresenta o cabeçalho "R", indicando que este é o nome da relação (ou tabela) que será decomposta; a segunda coluna apresenta o cabeçalho "sur", indicando ser o atributo (ou campo) correspondente à chave substituta, seguido de três linhas com os valores "s1", "s2" e "s3"; a terceira coluna apresenta o cabeçalho "a1", indicando ser o primeiro atributo (ou campo) da tabela, seguido de três linhas com os valores "v11", "v12" e "v13"; a quarta coluna apresenta o cabeçalho "a2", indicando ser o segundo atributo (ou campo) da tabela, seguido de três linhas com os valores "v21", "v22" e "v23"; finalmente, a quinta coluna apresenta o cabeçalho "a3", indicando ser o terceiro atributo (ou campo) da tabela, seguido de três linhas com os valores "v31", "v32" e "v33". Na parte de baixo da figura é exibido o resultado do processo de decomposição: três tabelas, uma para cada campo da tabela original. Na primeira tabela, a primeira coluna apresenta o cabeçalho "a1" (referente ao primeiro atributo - ou campo - da tabela original); a segunda coluna apresenta o cabeçalho "sur", indicando ser o campo correspondente à chave substituta, seguido de três linhas com os valores "s1", "s2" e "s3"; a terceira coluna apresenta o cabeçalho "val", indicando receber os valores provenientes do atributo - ou campo - da tabela original, seguido de três linhas com os valores "v11", "v12" e "v13". As duas tabelas que complementam o resultado do processo de decomposição seguem o mesmo padrão da primeira, respectivamente com os valores "a2" e "a3" no cabeçalho da primeira coluna e os mesmos valores de cabeçalho e chaves substitutas para a segunda coluna. A terceira coluna, por sua vez, recebe o cabeçalho "val" e, respectivamente, os valores "v21", "v22" e "v23" para a segunda tabela e "v31", "v32" e "v33" para a terceira tabela.

O modelo decomposto atende à demanda de armazenar registros com conteúdos diferentes e que representam instâncias ou objetos do mundo real de um mesmo tipo. Para isso, os valores dos campos existentes em cada instância ou objeto são direcionados especificamente a uma tabela; caso um determinado campo não exista ou não esteja presente em uma instância, o direcionamento não ocorre. Essa estratégia resolve também o problema das lacunas: como dados não existentes ou não preenchidos não são inseridos, não existe a possibilidade de se

gerarem espaços não ocupados nas tabelas que distribuem os dados entre si. Há, porém, considerações a serem feitas relacionadas ao processo de recuperação dos dados de uma instância. A seleção de dois ou mais campos de uma instância implicam o acesso a duas ou mais tabelas, com a posterior junção dos resultados parciais via chave substituta para a obtenção de um resultado final. Tanto o acesso a múltiplas tabelas quanto a junção dos resultados parciais são etapas custosas, cuja execução pode inviabilizar a decomposição de esquemas mais complexos (tabelas com dezenas de campos, por exemplo).

Em resumo, um esquema de BD decomposto construído a partir do modelo relacional possui as seguintes vantagens e desvantagens:

- Permitem armazenar registros com estruturas diferentes, mesmo representando os mesmos tipos de instâncias ou objetos;
- Eliminam lacunas;
- Ainda devem respeitar o número máximo de colunas e a especificação de tipagem e domínio das colunas da tabela original. Flexibilizar essa restrição implica na criação de tabelas em tempo de execução, para que novas colunas (não previstas) possam ter seu conteúdo armazenado;
- O tempo de resposta a consultas que envolvem duas ou mais tabelas tende a ser estendido, dada a necessidade de se acessarem os dados em dois ou mais locais diferentes e proceder, em seguida, com uma junção dos resultados intermediários visando recompor a(s) instância(s) ou objeto(s) resultante(s) da consulta.

O artigo encontrado por Marco referente ao modelo decomposto, compartilhado com o restante da equipe, trouxe consenso sobre a dificuldade de uso de SGBDs relacionais para o armazenamento dos dados heterogêneos que serão gerados pelos hospitais. Ficou claro a todos que uma abordagem NoSQL parece ser mais conveniente, e a médio/longo prazos pode valer o investimento em pesquisa e implantação necessários para sua efetivação. Porém, qual entre os diversos modelos NoSQL adotar? Objetivando responder a esta pergunta, Samantha, a especialista em DevOps, trouxe ao conhecimento da equipe outro artigo focado em armazenamento de dados distribuído. Nele, é apresentada a arquitetura e os princípios de funcionamento de uma ferramenta chamada ***Bigtable***.

Bigtable é, por definição, um sistema para armazenamento de dados distribuído projetado para dados estruturados, ou seja, dados que possam ser organizados na forma de linhas e colunas, que podem ser instalados em clusters de servidores convencionais capazes de suportar volumes de dados na faixa dos *petabytes*. À época de sua concepção no Google, o sistema foi utilizado como repositório de dados para diferentes aplicações, como o Google Earth, o Google Finance e mesmo como repositório para o resultado da indexação de páginas utilizado na ferramenta de busca. Apesar da heterogeneidade dos dados submetidos para armazenamento, e independentemente do que é armazenado (por exemplo URLs - *Uniform Resource Locators*, imagens de satélite, páginas web), a estrutura do sistema distribuído é a mesma (CHANG *et al.*, 2006).



NOVAS DESCOBERTAS

Petabyte é um múltiplo da unidade de representação e armazenamento de informações byte, correspondendo a décima-quinta potência (10^{15}) de unidades desta última. A unidade é utilizada em projetos de Big Data como forma de mensurar o volume de dados disponível em um cenário de aquisição, processamento e/ou armazenamento.



**NOVAS DESCOBERTAS**

URL (Uniform Resource Locator) corresponde ao endereço de um recurso na internet, juntamente com a especificação do protocolo necessário para acessá-lo. Exemplo: <http://google.com.br>.

A estrutura de dados utilizada pelo *Bigtable* (um mapa ordenado multidimensional esparsa e distribuída) pode ser visualizada na Figura 2. Nela, é exemplificado o armazenamento do resultado da *web crawling* que coleta as referências a uma página web a partir de outras páginas, o que permite compor o ranqueamento utilizado pelo buscador do Google para o cálculo de relevância de cada página indexada. Os seguintes elementos podem ser destacados a partir da arquitetura da solução:

- **Chave de linha:** é o equivalente a uma chave primária do modelo de dados relacional. Suportando textos de até 64 quilobytes (kB), as chaves de linha são ordenadas de forma lexicográfica; no exemplo apresentado, a chave “com.cnn.www” é utilizada para referenciar unicamente uma linha onde serão armazenadas todas as referências encontradas em outras páginas que levem à URL da chave.
- **Família de colunas:** é um agrupador de colunas e a base do acesso aos dados tanto em memória quanto em disco, e precisam ser conhecidas e criadas com antecedência. No exemplo apresentado, existem duas famílias de coluna previstas: “contents.” e “anchor.”. A primeira é utilizada para o armazenamento do conteúdo das páginas web referenciadas, e a segunda para armazenar o texto encontrado nas páginas web que fazem referência às primeiras. Sintaticamente, o nome de uma família de colunas sempre aparece à esquerda da marca de dois-pontos.
- **Coluna:** é uma divisão de uma família, permitindo separar conteúdos de acordo com o contexto de uso da estrutura. A criação de colunas é dinâmica e executada pelo próprio sistema de informações que acessa a estru-

tura; assim, a responsabilidade pela criação e uso das colunas criadas é dos sistemas que o acessam, e não do sistema de armazenamento distribuído. Não há limite para o número de colunas que possa ser criado, havendo apenas a necessidade de que a nomenclatura dentro de uma família seja exclusiva para cada coluna. No exemplo, existem duas colunas nomeadas dentro da família “anchor”: “cnnsi.com” e “my.look.ca”. Sintaticamente, o nome de uma coluna sempre aparece à direita da marca de dois-pontos.

- **Célula:** corresponde à interseção de uma coluna com uma linha, sendo utilizada efetivamente para o armazenamento dos dados da tabela. Os dados armazenados são tratados como cadeias de bytes sem formatação ou tratamento por parte do sistema de armazenamento distribuído; assim, cabe ao sistema de informações dar contexto aos dados armazenados e recuperados a partir da estrutura. No exemplo, a célula localizada na interseção entre a chave de linha “com.cnn.www” e a família de colunas “contents:” é utilizada para armazenar o conteúdo da página web localizada em “www.cnn.com”. As células localizadas na interseção entre a chave de linha “com.cnn.www” e as famílias de colunas/colunas “anchor:cnnsi.com” e “anchor:my.look.ca” são utilizadas para armazenar os textos encontrados nas páginas (cujos endereços são o próprio nome das colunas) que possuem um link para a página web localizada em “www.cnn.com”.
- **Timestamp:** marca temporal que permite versionar o conteúdo de uma célula. Um sistema de informações conectado ao sistema de armazenamento distribuído pode definir versões no tempo para o conteúdo de interesse, ou ignorar essas versões, o que faz com que o sistema de armazenamento, em resposta a consultas, retorne sempre a versão mais recente. No exemplo, a célula localizada na interseção entre a chave de linha “com.cnn.www” e a família de colunas “contents:” possui três versões relacionadas aos timestamps “t3”, “t5” e “t6”. As células localizadas na interseção entre a chave de linha “com.cnn.www” e as famílias de colunas/colunas “anchor:cnnsi.com” e “anchor:my.look.ca”, por sua vez, possuem respectivamente uma versão cada relacionadas aos timestamps “t9” e “t8”.

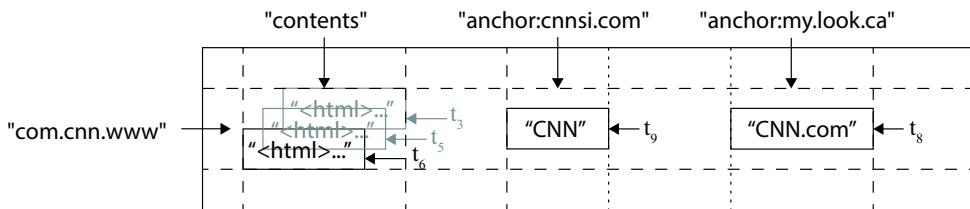


Figura 2 - Estrutura de dados utilizada pelo Bigtable, utilizada (neste exemplo) para o armazenamento do conteúdo de páginas web e referências a essas páginas a partir de outras páginas

Fonte: Chang et al. (2006, p. 2).

Descrição da Imagem: a figura exibe uma tabela com uma linha e três colunas. À esquerda da tabela é exibido o texto “com.cnn.www”, uma URL reversa que serve como chave única para a linha. À direita do texto existe uma seta apontando também para a direita, onde se encontra o conteúdo das colunas. A primeira coluna possui o cabeçalho “contents:” e armazena o conteúdo de uma página web. Abaixo do cabeçalho existe uma seta apontando também para baixo, onde está o conteúdo armazenado. Esse conteúdo é organizado com relação ao tempo, e possui três versões geradas respectivamente nos tempos “t3”, “t5” e “t6” (tempos esses que aparecem na tabela). A segunda coluna possui o cabeçalho “anchor:cnnsi.com” e armazena o texto encontrado em uma outra página web que referencia a URL reversa utilizada como chave para a linha. Abaixo do cabeçalho existe uma seta apontando também para baixo, onde está armazenado o conteúdo “CNN”. Esse conteúdo é organizado com relação ao tempo, e possui uma versão no tempo “t9” (tempo esse que aparece na tabela). A terceira coluna possui o cabeçalho “anchor:my.look.ca” e armazena o texto encontrado em uma outra página web que referencia a URL reversa utilizada como chave para a linha. Abaixo do cabeçalho existe uma seta apontando também para baixo, onde está armazenado o conteúdo “CNN.com”. Esse conteúdo é organizado com relação ao tempo, e possui uma versão no tempo “t8” (tempo esse que aparece na tabela).



PENSANDO JUNTOS

A estrutura em três dimensões oferecida pelo *Bigtable* é um diferencial importante quando o modelo de dados da solução é comparado com o modelo relacional. Neste último, o versionamento de registros no tempo costuma ser feito em duas dimensões, com o tempo assumindo a posição de um campo convencional, o que dá origem a bancos de dados temporais (não confundir com séries temporais).



NOVAS DESCOBERTAS

Web crawler é um programa de computador desenvolvido para procurar páginas web de acordo com um conjunto de critérios. O processo de busca, visita e coleta de dados é chamado de *web crawling*.



NOVAS DESCOBERTAS

Quilobyte é um múltiplo da unidade de representação e armazenamento de informações byte, correspondendo a 1.024 unidades desta última.

O endereçamento de células por meio de chaves de linha, famílias de colunas e colunas (e também, opcionalmente, por *timestamps*) facilita a distribuição do mapa em um cluster. Essa característica de distribuição faz parte da arquitetura da solução, e pode ser visualizada na Figura 3. De acordo com a especificação *Bigtable*, a estrutura de dados em três níveis (similar à estrutura de uma árvore B^+) é organizada como segue:

- **Chubby files:** arquivo(s) mantido(s) por um serviço de bloqueio (*lock*) distribuído. O serviço é utilizado para a manutenção e o acompanhamento do cluster, e os arquivos gerenciados armazenam dados referentes às tabelas (mapas) distribuídas pelo cluster. Dentre esses dados destacam-se as famílias de colunas criadas antecipadamente em cada mapa (equivalente ao esquema criado para o mapa), listas de controle de acesso (ACLs - *Access Control Lists*) com as permissões dos usuários e os endereços de uma das partes de cada mapa para acesso;
- **Root tablete:** tabela de metadados responsável por manter atualizados os dados de localização de todas as demais tabelas de metadados que compõem um mapa. É equivalente a um catálogo ou a um índice de outras tabelas de metadados, disponibilizado para acelerar a navegação pelo conteúdo da estrutura;
- **Other metadata tablets:** tabelas de metadados responsáveis por manter atualizada a localização de todas as tabelas que armazenam os dados do mapa. Além da localização, os metadados incluem conteúdo sobre as chaves de linha encontradas nas tabelas de dados. Como no *Bigtable* o acesso é feito primordialmente pela chave de linha, metadados com esse conteúdo permitem uma navegação direcionada às tabelas que efetivamente conterão os dados de interesse;

- **UserTable1..UserTableN:** tabelas de dados que, juntas, compõem o mapa. Cada uma dessas tabelas armazena um intervalo de chaves de linha, organizadas lexicograficamente para acelerar o processo de busca.

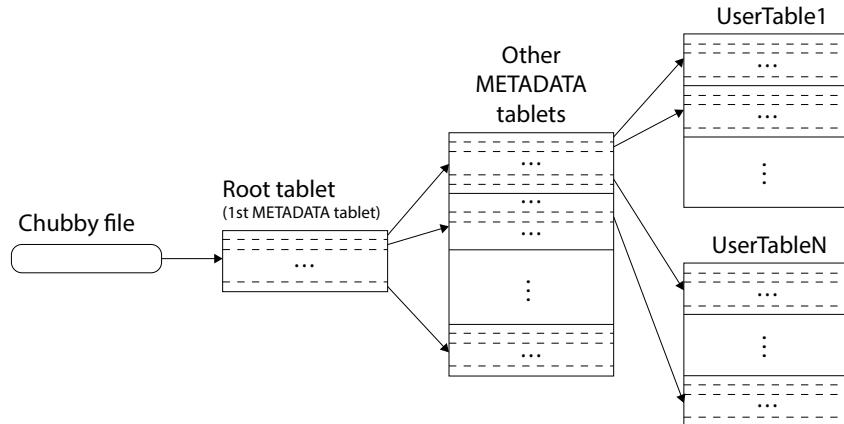


Figura 3 - Organização de um mapa ordenado multidimensional esparsos e distribuídos em um cluster, seguindo a especificação de três níveis definida pela arquitetura do serviço
Fonte: Chang et al. (2006, p. 4).

Descrição da Imagem: a figura exibe a estrutura em três níveis do Bigtable para distribuição de dados em cluster. À esquerda da figura é exibido um retângulo com o título “Chubby file”, que representa um dos arquivos mantido pelo serviço de bloqueio (lock) distribuído. Este retângulo aponta com uma seta à direita para outro retângulo intitulado “Root tablet (1st METADATA tablet)”, correspondendo ao primeiro nível de metadados do mapa. Este retângulo, por sua vez, aponta com três setas à direita para outros retângulos desenhados uns sobre os outros com o título “Other METADATA tablets”, fazendo referência às tabelas de metadados intermediárias. O primeiro retângulo desses metadados intermediários aponta com duas setas à direita, cada uma para um novo retângulo, intitulados “UserTable1” (tabelas onde efetivamente os dados são armazenados). O segundo retângulo desses metadados intermediários aponta com uma seta à direita para um novo retângulo, intitulado “UserTableN” (tabela onde efetivamente os dados são armazenados).

O acesso aos dados do *Bigtable* para escrita utiliza primitivas e não a linguagem SQL para ser executado. O Quadro 1 apresenta um trecho de código-fonte escrito na linguagem C++ que executa operações para modificação de conteúdo, como segue:

- **Linha 2:** um ponteiro para a tabela (mapa) desejada é criado, permitindo a abertura de uma conexão visando a manipulação do seu conteúdo;

- **Linha 5:** é especificada a chave de linha que receberá as alterações;
- **Linha 6:** é atribuída a uma coluna chamada “www.c-span.org”, pertencente à família de colunas “anchor”, o valor “CNN”;
- **Linha 7:** é excluída a coluna chamada “www.abc.com”, pertencente à família de colunas “anchor”;
- **Linha 9:** as operações de atribuição e exclusão são executadas.

```

1: // Open the table
2: Table *T = OpenOrDie("/bigtable/web/webtable");
3:
4: // Write a new anchor and delete an old anchor
5: RowMutation r1(T, "com.cnn.www");
6: r1.Set("anchor:www.c-span.org", "CNN");
7: r1.Delete("anchor:www.abc.com");
8: Operation op;
9: Apply(&op, &r1);

```

Quadro 1 - Exemplo de código-fonte escrito em linguagem C++ para a atualização de dados em uma tabela (mapa) armazenada no Bigtable / Fonte: Chang et al. (2006, p. 3).



NOVAS DESCOPERTAS

C++ é uma linguagem de programação multiplataforma orientada a objetos utilizada na criação de aplicações de alto desempenho. Como extensão da linguagem C, é dado ao programador um alto nível de controle sobre a memória e outros recursos do sistema (o que em outras linguagens de programação de mesmo nível é abstrato).

A leitura de dados, por sua vez, utiliza abstrações que permitem o acesso às linhas de uma tabela (mapa) via chaves de linha, famílias de colunas, colunas e timestamps especificando versões. O Quadro 2 apresenta um trecho de código-fonte escrito na linguagem C++ que executa operações para a pesquisa e a recuperação de conteúdo, como segue:

- **Linha 1:** é criado um objeto para acesso de leitura à tabela definida no Quadro 1;
- **Linha 3:** a partir do objeto criado, é recuperado um ponteiro para um objeto que permitirá o acesso ao conteúdo da família de colunas “anchor”;
- **Linha 4:** é especificado ao objeto que todas as versões dos dados armazenados deverão ser retornadas na consulta;
- **Linha 5:** a linha cuja chave é “com.cnn.www” é acessada;
- **Linha 6:** utilizando uma estrutura de repetição, são percorridas todas as colunas que pertencem à família especificada anteriormente;
- **Linhas 7 a 11:** são exibidos em tela, respectivamente, a identificação da linha, o nome da coluna, o timestamp relacionado ao dado recuperado e, finalmente, o dado propriamente dito.

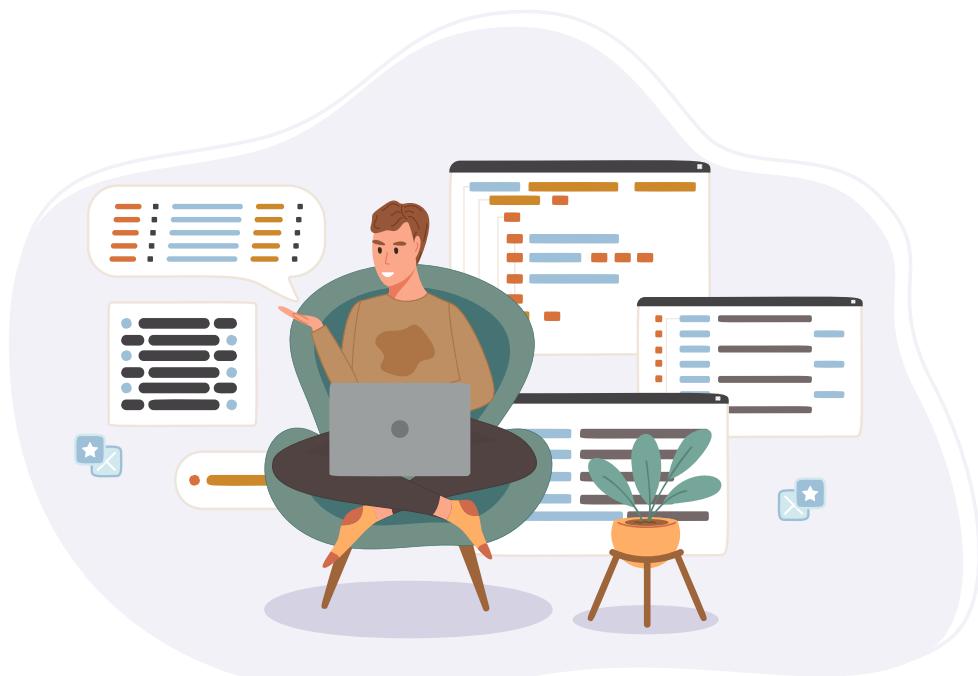
```

1: Scanner scanner(T);
2: ScanStream *stream;
3: stream = scanner.FetchColumnFamily("anchor");
4: stream->SetReturnAllVersions();
5: scanner.Lookup("com.cnn.www");
6: for (; !stream->Done(); stream->Next()) {
7:   printf("%s %s %lld %s\n",
8:         scanner.RowName(),
9:         stream->ColumnName(),
10:        stream->MicroTimestamp(),
11:        stream->Value());
12: }
```

Quadro 2 - Exemplo de código-fonte escrito em linguagem C++ para a recuperação de dados a partir de uma tabela (mapa) armazenada no Bigtable / Fonte: Chang et al. (2006, p. 3).

Após anos de sua concepção e utilização, *Bigtable* é oferecido hoje como um serviço em nuvem altamente gerenciável e escalonável. A sua proposta, à época da publicação do artigo com a sua especificação, foi disruptiva; a sua arquitetura, por sua vez, serviu como fundamento para a construção de outros produtos com foco nas mesmas características de flexibilidade de esquema, escalabilidade e alta distribuição. Dos produtos resultantes, um merece destaque por ter se posicionado como o SGBD de referência para um ecossistema inteiro de soluções para Big Data: **HBase™**.

O HBase⁽¹¹⁾ é um SGBD open source orientado a colunas escrito na linguagem de programação Java. Concebido inicialmente como parte de um projeto para Processamento de Linguagem Natural (PLN), teve sua primeira release disponibilizada em 2007. Em 2008 o projeto foi incorporado ao projeto Hadoop e, em 2010, foi promovido a um projeto *Apache independente*. Em ecossistemas Hadoop de complexidade variada, HBase e HDFS (*Hadoop File System*) compõem a infraestrutura de armazenamento de dados. Essa infraestrutura oferece altos *throughputs* de escrita e leitura sequencial (além da redundância a falhas oferecida pelo HDFS), e flexibilidade na definição de esquemas provida pelo HBase. Como uma pilha de serviços (vide Figura 4), o HBase é executado sobre o HDFS - o que permite que o primeiro utilize os serviços oferecidos pelo segundo.



**NOVAS DESCOBERTAS**

- 1.** PLN (Processamento de Linguagem Natural) corresponde ao ramo da Inteligência Artificial (IA) que procura dotar os computadores com a capacidade de entender textos falados e/ou escritos, de forma análoga aos seres humanos. Para esse objetivo são utilizadas técnicas de Machine Learning, técnicas estatísticas e técnicas de *deep learning* (entre outras) para que executem, por exemplo, reconhecimento de fala e análise de sentimentos.
- 2.** Hadoop é um *framework* open source direcionado ao armazenamento e ao processamento de grandes volumes de dados (em escala de *gigabytes* a *petabytes*), utilizando a infraestrutura de clusters para esse fim. Implementado na linguagem de programação Java, organiza-se em quatro módulos principais: um sistema de arquivos distribuído (HDFS), um monitor e gerenciador de recursos (YARN), um *framework* de processamento (MapReduce) e bibliotecas utilitárias.
- 3.** HDFS (*Hadoop File System*) é um sistema de arquivos distribuído projetado para instalação, configuração e uso em hardware convencional. Altamente tolerante a falhas, o sistema gerencia com eficiência grandes volumes de dados, oferecendo alta vazão (*throughput*) para processos de escrita e leitura.



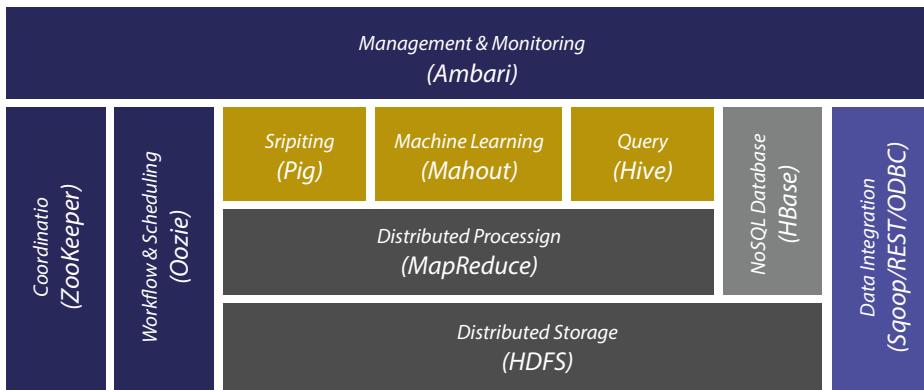


Figura 4 - Exemplo de um ecossistema Hadoop, com seus componentes organizados na forma de uma pilha de softwares executados de forma colaborativa / Fonte: adaptada de Abualkishik (2019).

Descrição da Imagem: a figura exibe um conjunto de blocos (retângulos) que compõem um exemplo de ecossistema Hadoop. Cada retângulo apresenta um software, e o seu conjunto está organizado como segue: (1) o retângulo da parte inferior exibe o texto "Distributed Storage (HDFS)" ("Armazenamento Distribuído (HDFS)"); (2) o retângulo posicionado logo acima exibe o texto "Distributed Processing (MapReduce)" ("Processamento Distribuído (MapReduce)"); logo acima, são exibidos retângulos com os textos "Scripting (Pig)" (não há uma tradução formal para "scripting", e "Pig" é o nome do software) (3), "Machine Learning (Mahout)" ("Aprendizado de Máquina (Mahout)") (4), e "Query (Hive)" ("Pesquisa ou Inquirição (Hive)") (5); à esquerda dos retângulos (1), (2) e (3), orientado na vertical, há um retângulo com o texto "Workflow & Scheduling (Oozie)" ("Fluxo de trabalho & Agendamento (Oozie)") (6); à esquerda do retângulo (6), orientado na vertical, há um retângulo com o texto "Coordination (ZooKeeper)" ("Coordenação (ZooKeeper)") (7); à direita dos retângulos (2) e (5), orientado na vertical, há um retângulo com o texto "NoSQL Database (HBase)" ("Banco de dados NoSQL (HBase)") (8); à direita dos retângulos (1) e (8), orientado na vertical, há um retângulo com o texto "Data Integration (Sqoop/REST/ODBC)" ("Integração de dados (Sqoop/REST/ODBC)") (9). Na parte superior, acima dos demais blocos, é exibido um retângulo com o texto "Management & Monitoring (Ambari)" ("Gerenciamento & Monitoração (Ambari)") (10).



A pilha de serviços disponibilizada pelo Hadoop permite ao HBase especializar suas funcionalidades voltadas ao particionamento e à gerência de chaves de linha, famílias de colunas e colunas. Replicação e alta disponibilidade ficam a cargo do HDFS, por meio das suas próprias funcionalidades e definições de arquitetura. A Figura 5 mostra a organização do HDFS com relação à distribuição de dados e replicação; a Figura 6, por sua vez, exibe a arquitetura do HBase e o posiciona acima do HDFS como consumidor dos seus serviços.

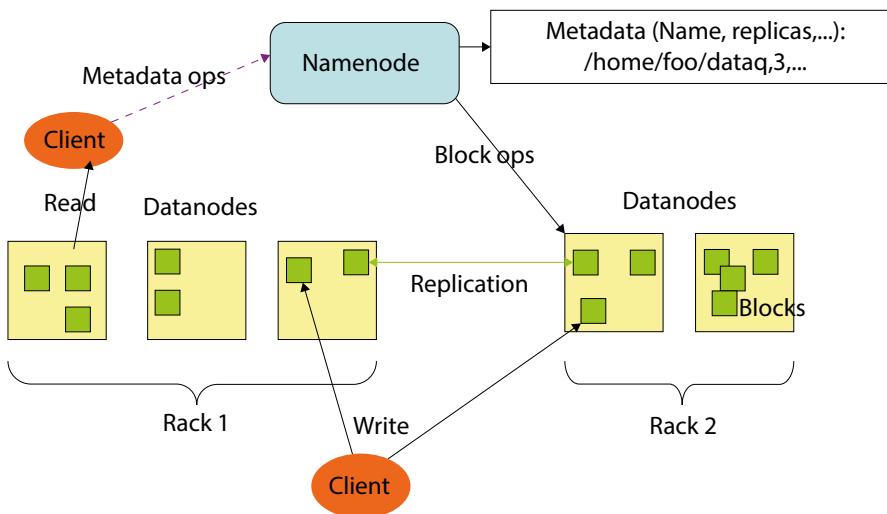


Figura 5 - Arquitetura do HDFS para a versão 1.2.1 do Hadoop. Fica evidente a característica distribuída do sistema de arquivos, que organiza seu conteúdo em racks, datanodes e blocos (blocks)

Fonte: Hadoop ([2022], on-line).

Descrição da Imagem: a figura exibe a arquitetura distribuída do HDFS, juntamente com a representação dos acessos de escrita e leitura que são suportados. Na parte intermediária da figura são exibidos cinco quadrados, cada um equivalente a um datanode (nó de dados) utilizado para armazenamento. Os três primeiros nós estão agrupados em um rack (denominado rack 1), e os demais estão agrupados em outro rack denominado rack 2. Dentro de cada um desses quadrados estão dispostos outros quadrados menores, equivalentes a blocos de dados (blocks). O número de blocos por nó de dados varia; na figura, são três para o primeiro e quarto nós, dois para o segundo e terceiro nós, e quatro para o quinto nó. Há uma seta bidirecional entre dois desses blocos armazenados, respectivamente, no terceiro e quarto nós, que indica uma replicação (replication), ou seja, uma cópia completa do bloco entre nós. Na parte inferior da figura um cliente (client) representado por uma elipse executa operações de escrita (write) em dois blocos armazenados, respectivamente, nos nós três e quatro (operações essas que aparecem como setas). Acima do primeiro nó está posicionado um outro cliente (também representado por uma elipse) que executa uma operação de leitura (operação essa que aparece como uma seta). Esse mesmo cliente está conectado com uma a um retângulo posicionado à sua direita, pouco acima, denominado Namenode. A seta que os conecta possui o rótulo "Metadata ops" (operações com metadados). Esse Namenode, por sua vez, se conecta com uma seta ao quarto nó; a seta possui o rótulo "Block ops" (operações com blocos). Por fim, o Namenode aponta com uma seta a outro retângulo, posicionado à sua direita, com os dizeres "Metadata (Name, Replicas, ...): /home/foo/data, 3, ..." ("Metadados (Nome, Réplicas, ...): /home/foo/data, 3, ...").

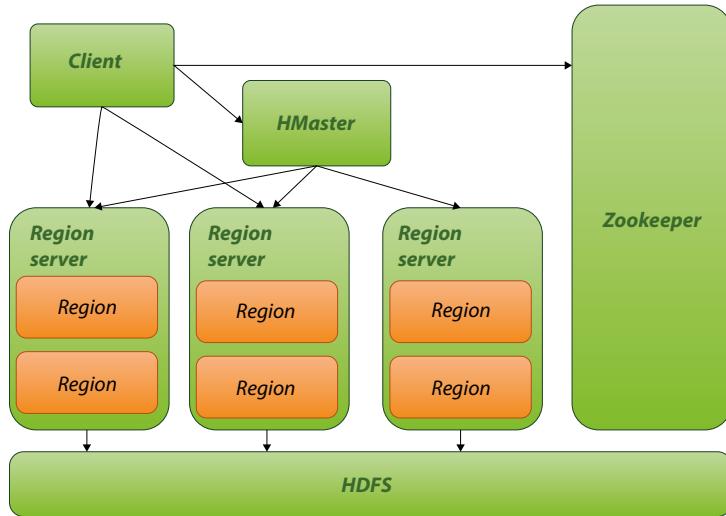


Figura 6 - Arquitetura do HBase apoiada sobre o HDFS para a utilização dos seus serviços de distribuição e replicação / Fonte: adaptado de Nima (2018).

Descrição da Imagem: a figura exemplifica uma instalação e configuração do HBase utilizando o HDFS como base. Na parte inferior, um retângulo com o HDFS é exibido indicando que o SGBD está apoiado nesse sistema de arquivos. Logo acima, há três retângulos com o rótulo “Region server” (“Servidor regional”), todos apontando com setas individuais para o HDFS. Em cada um desses retângulos há dois outros com o rótulo “Region” (“Região”). Acima dos servidores regionais há um retângulo com o rótulo “HMaster”, o qual se conecta com setas individuais unidirecionais aos servidores regionais. À esquerda deste último retângulo, levemente acima, há um outro retângulo com o rótulo “Client” (“Cliente”), que se conecta usando setas individuais unidirecionais com o “HMaster”, com os servidores regionais e com o “ZooKeeper”, retângulo posicionado à direita dos demais.

De forma similar ao Bigtable, o acesso ao HBase se dá por primitivas – e não pelo uso da linguagem SQL como nos SGBDs relacionais. Um dos acessos possíveis é disponibilizado via linha de comando (*shell*), que de forma interativa permite a execução dessas primitivas, como pode ser observado no Quadro 3:

- **Linha 1:** é criada uma tabela chamada “blog”, na qual é definida uma família de colunas chamada “postagens”. Estruturalmente, são as definições mínimas para que o SGBD possa ser utilizado;
- **Linha 2:** a tabela “blog” é alterada para que o conteúdo da família de colunas “postagens” seja compactado utilizando o formato GZIP;
- **Linha 3:** a tabela “blog” é descrita para verificação da sua estrutura.

```

1: hbase> create 'blog', 'postagens'
2: hbase> alter 'blog', { NAME => 'postagens', COMPRESSION => 'GZ' }
3: hbase> desc 'blog'

```

Quadro 3 - Execução interativa de comandos em um shell oferecido pelo HBase. São exibidos comandos de criação, configuração e exibição da estrutura de uma tabela / Fonte: o autor.



GZIP é um formato *lossless* (sem perda) para a compactação de dados, bem como o nome dado a softwares que implementam o formato. Sua definição é ser independente de CPU, sistema operacional, sistema de arquivos e conjuntos de caracteres de forma a tornar seu conteúdo interoperável.

Com a estrutura de tabela e família(s) de colunas criada, é possível utilizá-la para a persistência de conteúdo. No Quadro 4 são exibidos exemplos de comandos executados de forma interativa para essa persistência. É importante lembrar que, assim como no *Bigtable*, o *HBase* oferece o versionamento de células utilizando um *timestamp*. O uso desse versionamento é opcional: quando não especificado na consulta, é responsabilidade do SGBD retornar a versão mais recente do dado que estiver armazenado na(s) célula(s) recuperada(s).

- **Linha 1:** o texto “Bem-vindo ao Blog!” é inserido na tabela “blog”, em uma linha identificada unicamente pela chave “Primeira postagem”, na família de colunas chamada “postagens”. Caso a linha com essa chave ainda não exista, é criada. No exemplo, o valor não pertence a uma coluna, mas à própria família de colunas;
- **Linha 2:** é solicitada a recuperação do conteúdo armazenado na tabela “blog”, na linha identificada unicamente pela chave “Primeira postagem”, e na família de colunas “postagens”;
- **Linha 3:** é solicitada a recuperação do conteúdo completo da tabela “blog”.

```

1: hbase> put 'blog', 'Primeira postagem', 'postagens:', 'Bem-vindo
ao Blog!'

2: hbase> hbase> get 'blog', 'Primeira postagem', 'postagens:'

3: hbase> scan 'blog'

```

Quadro 4 - Execução interativa de comandos em um shell oferecido pelo HBase. São exibidos comandos para a inserção e a recuperação do conteúdo armazenado em uma tabela / Fonte: o autor.



PENSANDO JUNTOS

O acesso às tabelas no *HBase* (assim como no *Bigtable*) é feito por referência a uma linha, a uma família de colunas, a uma coluna dentro dessa família (opcional, caso o valor tenha sido armazenado na própria família de colunas) e a um timestamp (indicando a versão desejada, também opcional). Em suma, cada célula da tabela pode ser referenciada para a execução das operações desejadas.

Manutenções estruturais e configurações de tabelas no HBase são possíveis, porém, com impacto na sua disponibilidade. Dada a distribuição das tabelas em um cluster ou grid, alterações desse tipo devem ser propagadas para que a estrutura distribuída mantenha sua consistência. O Quadro 5 exemplifica uma manutenção desse tipo, onde a estrutura afetada permanece offline durante o processo de ajuste:

- **Linha 1:** a tabela “blog” é desabilitada para manutenção. Enquanto não for habilitada novamente, a tabela permanece inacessível aos clientes do SGBD;
- **Linha 2:** a tabela “blog” é modificada para que a família de colunas chamada “postagens” passe a armazenar todas as versões possíveis do seu conteúdo (todos os timestamps possíveis). A constante `org.apache.hadoop.hbase.HConstants::ALL_VERSIONS` corresponde ao atributo `MAX_VALUE` da classe `Integer` na linguagem Java. Em sua versão 11, esse valor é igual a $2^{31}-1$, ou 2.147.483.647. Esse é o número total de versões que poderão ser armazenadas;

- **Linha 3:** a tabela “blog” é modificada para a criação de uma nova família de colunas chamada “revisoes”. Essa nova família de colunas já é criada com total versionamento de seu conteúdo.
- **Linha 4:** a tabela “blog” é habilitada novamente, ficando disponível para acesso pelos clientes do SGBD.

```

1: hbase> disable 'blog'
2: hbase> alter 'blog', { NAME => 'postagens', VERSIONS => org.
 apache.hadoop.hbase.HConstants::ALL_VERSIONS }
3: hbase> alter 'blog', { NAME => 'revisoes', VERSIONS => org.
 apache.hadoop.hbase.HConstants::ALL_VERSIONS }
4: hbase> enable 'blog'

```

Quadro 5 - Execução interativa de comandos em um shell oferecido pelo HBase. São exibidos comandos para alteração estrutural e configuração de uma tabela / Fonte: o autor.

O acesso pontual via shell interativo ao HBase pode ser complementado com a escrita de programas para a execução de rotinas mais elaboradas. O Quadro 6 exibe um exemplo de um programa escrito na linguagem JRuby cuja execução pode ser feita linha-a-linha diretamente em shell (de forma também interativa ou através de chamadas via sistema operacional):

- **Linhas 1 e 2:** são importadas classes para o acesso a tabelas e para a execução de gravações no *HBase*;
- **Linhas 3, 4 e 5:** definem uma estrutura que será utilizada para organizar os dados a serem inseridos em uma tabela;
- **Linha 6:** instancia um objeto que abstrai a tabela chamada “blog”, dando acesso às suas operações de escrita e leitura;
- **Linha 7:** instancia um objeto que abstrai a operação de inserção via primitiva PUT. No momento da instanciação é especificada a chave de linha que será utilizada (no exemplo, “Segunda postagem”);

- **Linha 8:** adiciona conteúdo à instância criada na linha 7, especificando que a família de colunas “postagens” receberá o conteúdo “Texto da segunda postagem!”;
- **Linha 9:** adiciona conteúdo à instância criada na linha 7, especificando que será criada uma coluna chamada “autor” pertencente à família de colunas “revisoes”. O conteúdo a ser armazenado nesta coluna é “Alex”;
- **Linha 10:** adiciona conteúdo à instância criada na linha 7, especificando que será criada uma coluna chamada “comentarios” pertencente à família de colunas “revisoes”. O conteúdo a ser armazenado nesta coluna é “Meu primeiro comentário!”;
- **Linha 11:** a primitiva PUT é executada na tabela.

```

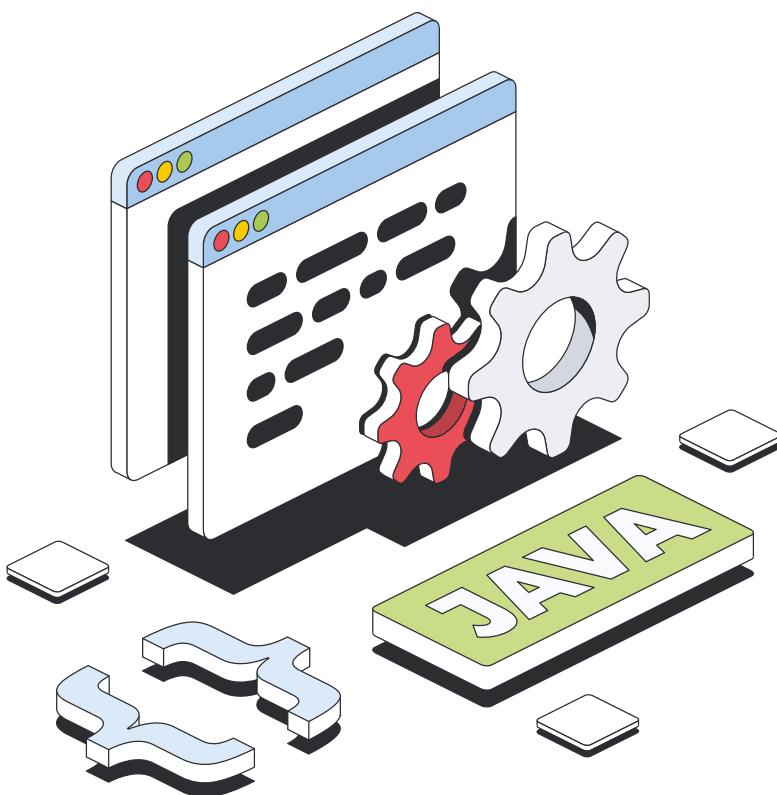
1: hbase> import 'org.apache.hadoop.hbase.client.HTable'
2: hbase> import 'org.apache.hadoop.hbase.client.Put'
3: hbase> def jbytes(*args)
4: hbase> args.map {|arg| arg.to_s.to_java_bytes}
5: hbase> end
6: hbase> table = HTable.new(@hbase.configuration, "blog")
7: hbase> p = Put.new(*jbytes("Segunda postagem"))
8: hbase> p.add(*jbytes("postagens", "", "Texto da segunda
9: hbase> postagem!"))
10: hbase> p.add(*jbytes("revisoes", "autor", "Alex"))
11: hbase> p.add(*jbytes("revisoes", "comentarios", "Meu
primeiro comentário!"))
12: hbase> table.put(p)

```

Quadro 6 - Execução interativa de comandos em um shell oferecido pelo HBase. São exibidos comandos em JRuby para a criação de colunas e execução de um PUT na tabela “blog” / Fonte: o autor.

**NOVAS DESCOBERTAS**

JRuby é a implementação da linguagem de programação Ruby para execução em máquinas virtuais Java (*Java Virtual Machines - JVMs*). Disponibilizada como software livre, a linguagem permite uma forte integração com Java, de forma que possa ser embarcada junto a aplicações escritas nesta última - o que permite a interoperabilidade entre o código de ambas as linguagens.



Finalmente, primitivas para a exclusão tanto de conteúdos quanto de estruturas também estão disponíveis para execução interativa. O Quadro 7 exemplifica o uso dessas primitivas:

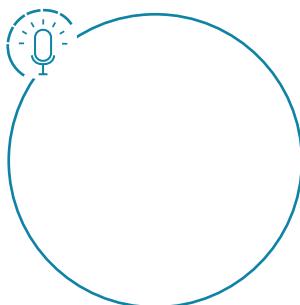
- **linha 1:** é excluído o conteúdo da coluna “autor” pertencente à família de colunas “revisoes” armazenado na linha cuja chave é “Primeira postagem” (tabela “blog”);
- **linha 2:** é excluído todo o conteúdo da linha cuja chave é “Primeira postagem” existente na tabela “blog”;
- **linha 3:** a tabela “blog” é desabilitada para manutenção. Enquanto não for habilitada novamente, a tabela permanece inacessível aos clientes do SGBD;
- **linha 4:** a tabela “blog” é apagada.

```

1: hbase> delete 'blog', 'Primeira postagem', 'revisoes:autor'
2: hbase> deleteall 'blog', 'Primeira postagem'
3: hbase> disable 'blog'
4: hbase> drop 'blog'

```

Quadro 7 - Execução interativa de comandos em um shell oferecido pelo HBase. São exibidos comandos para a exclusão de conteúdo e, por fim, de uma tabela inteira / Fonte: o autor.

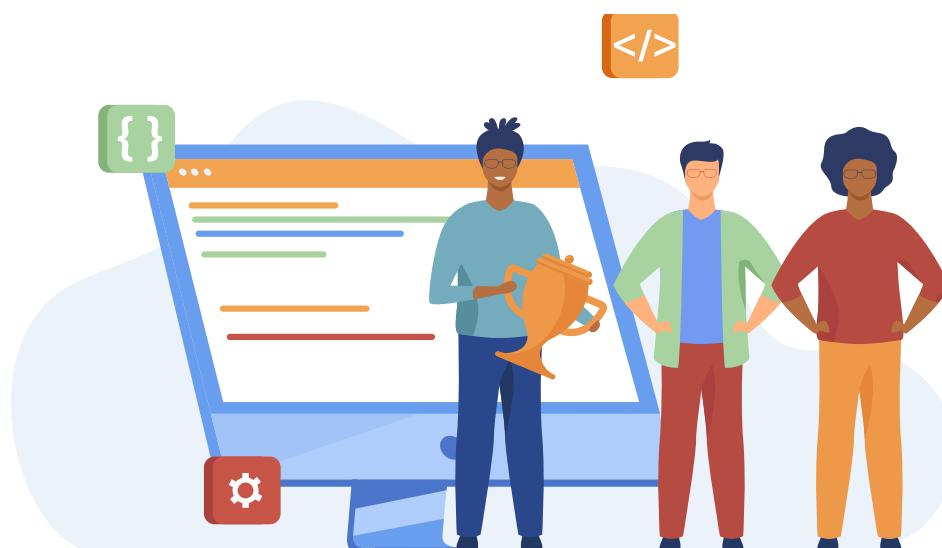


Os SGBDs orientados a colunas são construídos de forma a permitir que os sistemas de informação que os utilizam tenham a liberdade de criar tantas colunas quanto necessário, sob demanda. Acompanhe em nosso podcast os comentários feitos sobre essa transferência de responsabilidade do SGBD para a aplicação, bem como sobre os impactos decorrentes dessa arquitetura. Não perca!

A leitura dos artigos sobre o modelo de dados decomposto (definido sobre o modelo de dados relacional original) e sobre o projeto e implementação do Bigtable nivelou a equipe de TI em termos de conhecimento, e permitiu que seus integrantes pudessem identificar ferramentas capazes de auxiliar no armazena-

mento de dados heterogêneos para os quais uma estrutura prévia completa não é conhecida. No projeto Hospital sem Papel, datasets com essas características são comuns, e as ferramentas identificadas são opções viáveis para atender à demanda. Seja para o armazenamento de imagens de exames, textos de laudos ou mesmo documentos referentes a prontuários completos, SGBDs orientados a colunas foram considerados soluções viáveis.

Considerado por Luísa como uma ferramenta adequada para compor uma primeira versão focada no armazenamento do histórico de imagens DICOM dos pacientes atendidos pelos hospitais da rede, o HBase foi relacionado como componente do ecossistema de soluções que está sendo construído por sua equipe. O suporte a grandes volumes de dados, sua alta disponibilidade e sua integração com um sistema de arquivos robusto e que pode ser disponibilizado em uma infraestrutura convencional (HDFS) foram as características principais que fundamentaram a decisão; de forma secundária, mas não menos importante, o modelo de dados com esquema flexível e o desempenho previsto a partir dos critérios de acesso a serem utilizados (a identificação dos pacientes como chaves de linha, “imagens” como família de colunas e colunas dinâmicas para cada exame) completaram a justificativa para a adoção da ferramenta. Ao explorar as possibilidades de configuração para clusters e grids, é possível considerar que cada hospital participante possa contribuir com um ou mais nós da infraestrutura, e interligá-los com o objetivo de prover uma visão global de todo o conteúdo disponível.





1. Com relação à variedade (variety) em cenários de Big Data, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
 - a) Escalonamento e pré-processamento são obrigatórios após o processo de coleta de grandes volumes de dados ser concluído.
 - b) SGBDs relacionais são os sistemas indicados para o atendimento às demandas de cenários de Big Data.
 - c) Sistemas de informação em saúde são exemplos de sistemas onde há predominância de dados altamente estruturados.
 - d) A variedade implica em fontes diversas, formatos não padronizados e conteúdo podendo ser altamente estruturado, semiestruturado ou mesmo não estruturado (os quais podem ocorrer simultaneamente, em datasets diversos presentes em um mesmo cenário).
 - e) A legislação vigente não precisa ser observada para a tomada de decisões relacionada a questões de temporização.
2. Sobre a padronização de dados em saúde, é CORRETO afirmar que:
 - a) Logo após a avaliação dos dados de um paciente, esses dados podem ser descartados - não é necessário mantê-los como um histórico para acompanhamento de médio e longo prazos.
 - b) A interoperabilidade de dados implica apenas na transmissão de conteúdo alfanumérico.
 - c) A interoperabilidade de dados entre sistemas deve ser feita via conexões TCP/IP ponto a ponto, não utilizando RESTful services ou webservices SOAP disponibilizados para acesso via HTTP.
 - d) Esquemas fixos de BD são a solução ideal para problemas de interoperabilidade (troca de dados entre sistemas).
 - e) Apesar dos esforços de padronização, ainda é possível encontrar diferenças de representação entre os mesmos dados em diferentes sistemas.



3. Sobre o modelo de dados decomposto, é CORRETO afirmar que:
- a) Decomposição, para o modelo de dados, significa a divisão de uma tabela original em tabelas menores, uma tabela para cada campo.
 - b) Com a divisão da tabela original em tabelas menores, a tipagem de dados das colunas não precisa mais ser respeitada.
 - c) Qualquer consulta é resolvida acessando, no máximo, duas tabelas.
 - d) O modelo garante alta performance, tanto para gravações quanto para leituras (consultas).
 - e) São geradas lacunas quando valores nulos são inseridos nas tabelas resultantes do processo de decomposição.
4. Com relação ao modelo de dados orientado a colunas implementado pelo Bigtable, qual das opções a seguir apresenta uma afirmação VERDADEIRA?
- a) A manutenção de versões com timestamps é obrigatória, e deve ser tratada pelo back-end dos sistemas de informação que acessam o SGBD.
 - b) Famílias de colunas são agrupadores de colunas no modelo de dados implementado pelo SGBD.
 - c) Chaves de linha aceitam duplicidades, desde que o restante do conteúdo das linhas das linhas da tabela seja diferente.
 - d) Não é possível inserir valores diretamente em uma família de colunas. Sempre deve haver uma coluna relacionada.
 - e) Quando consultado sem especificação de timestamp, o SGBD retorna os dados mais antigos armazenados nas tabelas.
5. Considerando os comandos utilizados para manutenção e acesso ao HBase exibidos nesta unidade, é CORRETO afirmar que:
- a) A exclusão de dados exige que seja especificado o endereço de uma célula (linha + família de colunas + coluna); além de um timestamp;
 - b) A sintaxe utilizada como parte do referenciamento de uma célula é “coluna:família_de_colunas”.
 - c) A compressão de dados em uma tabela pode ser configurada, individualmente, para cada família de colunas.
 - d) Para a inclusão de dados é necessário informar, além de uma família de colunas, o nome de uma coluna já existente.
 - e) A criação de uma nova família de colunas em uma tabela pode ser feita com a tabela on-line (ativa), sem downtime.

MEU ESPAÇO



REFERÊNCIAS



UNIDADE 1

ANGADI, A. B.; ANGADI, A. B.; GULL, K. C. Growth of New Databases & Analysis of NOSQL Databases. **International Journal of Advanced Research in Computer Science and Software Engineering**, v. 3, n. 6, jun. 2013.

BASSO, D. E. **Big Data**. Curitiba: Contentus, 2020.

BRASIL. Lei nº 13.989, de 15 de Abril de 2020. **Dispõe sobre o uso da telemedicina durante a crise causada pelo coronavírus (SARS-CoV-2)**. 2020. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2019-2022/2020/lei/L13989. Acesso em: 13 nov. 2021.

BREWER, E. A. Towards robust distributed systems. **PODC '00**: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing. 2000. Disponível em: https://sites.cs.ucsb.edu/~rich/class/cs293-cloud/papers/Brewer_podc_keynote_2000.pdf. Acesso em: 14 dez. 2021.

CODD, E. F. A Relational Model of Data for Large Shared Data Banks. **Communications of the ACM**, v. 13, n. 6, jun. 1970.

DANDU, R. V. Storage media for computers in radiology. **Indian Journal of Radiology and Imaging**, v. 18, n. 4, nov. 2008.

DONTHA, R. Who Came Up With The Name Big Data? **Digital Transformation PRO**. 2018. Disponível em: <http://digitaltransformationpro.com/name-big-data/>. Acesso em: 18 nov. 2021.

DYKES, B. Big Data: Forget Volume and Variety, Focus On Velocity. **Forbes**. 2017. Disponível em: <https://www.forbes.com/sites/brentdykes/2017/06/28/big-data-forget-volume-and-variety-focus-on-velocity/?sh=be665c26f7d6>. Acesso em: 30 nov. 2021.

GILBERT, S; LYNCH, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. **Acm Sigact News**, v. 33, n. 2, 51-59. 2002. Disponível em: <https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.1495>. Acesso em: 14 dez. 2021.

HADI, H. J. et al. BIG DATA AND FIVE V'S CHARACTERISTICS. **International Journal of Advances in Electronics and Computer Science**, v. 2, n. 1, Jan. 2015.

HISTÓRICO da pandemia de COVID-19. **OPAS**. 2021?. Disponível em: <https://www.paho.org/pt/covid19/historico-da-pandemia-covid-19>. Acesso em: 13 nov. 2021.

ISHWARAPPA; ANURADHA, J. A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology. **Procedia Computer Science**, v. 48, 2015.

LANEY, D. Deja VVVu: Gartner's Original "Volume-Velocity-Variety" Definition of Big Data. **AIIM**. 2012. Disponível em: <https://community.aiim.org/blogs/doug-laney/2012/08/25/deja-vvvu-gartners-original-volume-velocity-variety-definition-of-big-data>. Acesso em: 19 nov. 2021.

MONICA, M.; KUMAR, K. R. Survey on Big Data by Coordinating Mapreduce to Integrate Variety of Data. **International Journal of Science and Research**, v. 2, n. 11, nov. 2013.

REFERÊNCIAS



OLIVEIRA, A. C.; PASSOS, M. M. Sustentabilidade Hospitalar: hospital sem papel e outras tendências. **Educação Sem Distância - Revista Eletrônica da Faculdade Unyleya**, v. 1, n. 2, 22 dez. 2020.

OWAIS, S. S.; HUSSEIN, N. S. Extract Five Categories CPIVW from the 9V's Characteristics of the Big Data. **International Journal of Advanced Computer Science and Applications**, v. 7, n. 3, 2016.

SALOMI, M. J. A.; MACIEL, R. F. Gestão de documentos e automação de processos em uma instituição de saúde sem papel. **Journal of Health Informatics**, v. 8, n. 1, 21 ago. 2015.

SILVA, L. F. C. et al. Banco de Dados Não Relacional. [S.I.]: [S.n.], 2021.

SOMMERVILLE, I. **Engenharia de Software**. 9 ed. São Paulo: Pearson Prentice Hall, 2011.

TANENBAUM, A. S.; STEEN, M. V. **Sistemas Distribuídos**: princípios e paradigmas. São Paulo: Pearson Prentice Hall, 2007.

THIYAGARAJAN, V. S.; VENKATACHALAPATHY, K. Isolating values from Big Data with the help of four VS. **International Journal of Research in Engineering and Technology**, v. 4, n. 1, jan. 2014.

TIGRE, P. B.; NORONHA, V. B. Do mainframe à nuvem: inovações, estrutura industrial e modelos de negócios nas tecnologias da informação e da comunicação. **Revista de Administração**, v. 48, n. 1, mar. 2013.

WIKIMEDIA COMMONS. **[Sem título]**. [1946]. 1 fotografia. Disponível em: https://commons.wikimedia.org/wiki/File:Reprogramming_ENIAC.png. Acesso em: 10 ago. 2022.

WIKIMÉDIA COMMONS. **[Sem título]**. 2005. 1 fotografia. Disponível em: <https://commons.wikimedia.org/wiki/File:DM-IBM360.jpg>. Acesso em: 10 ago. 2022.

WIKIMEDIA COMMONS. **[Sem título]**. 2009. 1 fotografia. Disponível em: <https://commons.wikimedia.org/wiki/File:Pdp-11-at-tnmoc.jpg>. Acesso em: 10 ago. 2022.

WIKIMEDIA COMMONS. **[Sem título]**. 2018. 1 ilustração. Disponível em: https://commons.wikimedia.org/wiki/File:Infográfico_Big_Data,_crescimento_exponencial_de_dados.png. Acesso em: 10 ago. 2022.

UNIDADE 2

ALGER, K. W. **Quick Start**: BSON Data Types - ObjectId. Disponível em: <https://www.mongodb.com/developer/quickstart/bson-data-types-objectid/>. Acesso em: 5 fev. 2022.

ANOTAÇÕES de Enfermagem. **COREN/SP**. Disponível em: <https://www.portaldaenfermagem.com.br/downloads/manual-anotacoes-de-enfermagem-coren-sp.pdf>. Acesso em: 24 jan. 2022.

BRITO, N. M. R.; VELOZO, B. C.; PAVANELLI, R. J. **Manual de Orientação**: Anotações de Enfermagem. Hospital das Clínicas da Faculdade de Medicina de Botucatu. Disponível em: <http://biblio>

REFERÊNCIAS



teca.cofen.gov.br/wp-content/uploads/2020/09/Manual-Anotacao-Enfermagem.pdf. Acesso em: 24 jan. 2022.

CARTILHA sobre Prontuário Eletrônico - A Certificação de Sistemas de Registro Eletrônico de Saúde. **Conselho Federal de Medicina e Sociedade Brasileira de Informática em Saúde**. Disponível em: http://www.sbis.org.br/certificacao/Cartilha_SBIS_CFM_Prontuario_Eletronico_fev_2012.pdf. Acesso em: 23 jan. 2022.

CODD, E. F. A Relational Model of Data for Large Shared Data Banks. **Communications of the ACM**, v. 13, n. 6, jun. 1970.

CODD, E.F. **Further normalization of the data base relational model**. Disponível em: <https://forum.thethirdmanifesto.com/wp-content/uploads/asgarosforum/987737/00-efc-further-normalization.pdf>. Acesso em: 29 jan. 2022.

DEJOY, P. **A Short History of MongoDB**. Disponível em: <https://petedejoy.com/writing/mongo-db>. Acesso em: 7 fev. 2022.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. São Paulo: Pearson Education do Brasil, 2018a.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. São Paulo: Pearson Education do Brasil, 2018b.

GRANJA, C. et al. An optimization based on simulation approach to the patient admission scheduling problem using a linear programming algorithm. **Journal of Biomedical Informatics**, v. 52, dez. 2014. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1532046414001889>. Acesso em: 28 maio 2022.

HAN, J. et al. Survey on NoSQL Database. **2011 6th International Conference on Pervasive Computing and Applications**. 2011. Disponível em: http://faculty.washington.edu/wlloyd/courses/tcss562/papers/Spring2017/team7_NOSQL_DB/Survey%20on%20NoSQL%20Database.pdf. Acesso em: 5 fev. 2022.

HOLLINGSWORTH, D. The Workflow Reference Model. **Workflow Management Coalition**. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.198.5206&rep=rep1&type=pdf>. Acesso em: 15 jan. 2022.

HOW to Improve Hospital Workflows. **HIPAA Journal**. Disponível em: <https://www.hipaajournal.com/improve-hospital-workflows/>. Acesso em: 16 jan. 2022.

HRIBAR, M. R. et al. Secondary Use of EHR Timestamp data: Validation and Application for Workflow Optimization. **AMIA... Annual Symposium Proceedings**. 2015. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4765636/>. Acesso em: 28 mai. 2022.

IMPROVING Clinical Staff Workflow in Hospitals. **HIPAA Journal**. Disponível em: <https://www.hipaajournal.com/improving-clinical-staff-workflow-in-hospitals/>. Acesso em: 15 jan. 2022.

REFERÊNCIAS



CFM Resolução nº 1.638/2002. **Conselho Federal de Medicina.** Disponível em: <https://sistemas.cfm.org.br/normas/visualizar/resolucoes/BR/2002/1638>. Acesso em: 23 jan. 2022.

SILVA, L. F. C. et al. Banco de Dados Não Relacional. [S.I.]: [S.n.], 2021.

TRIBUNELLA, T. Designing Relational Database Systems. **The CPA Journal.** Disponível em: https://www.researchgate.net/publication/272489096_Designing_Relational_Database_Systems. Acesso em: 30 jan. 2022.

TRUICĂ, C. O. et al. The Forgotten Document-Oriented Database Management Systems: An Overview and Benchmark of Native XML DODBMSes in Comparison with JSON DODBMSes. **Big Data Research**, v. 25, jul. 2021.

UNIDADE 3

ALWIDIAN, J. et al. Big Data Ingestion and Preparation Tools. **Modern Applied Science**, v. 14, n. 9, 2020.

COBUS, V.; HEUTEN, W. To Beep or Not to Beep? Evaluating Modalities for Multimodal ICU Alarms. **Multimodal Technologies and Interaction**, v. 3, n. 1, 2019.

DAVOUDI, A. et al. Intelligent ICU for Autonomous Patient Monitoring Using Pervasive Sensing and Deep Learning. **Scientific Reports**, v. 9, 2019.

FREITAS, M. C.; MENDES, M. M. R. Condições crônicas de saúde e o cuidado de enfermagem. **Rev. Latino-Am. Enfermagem**, v. 7, n. 5, dez. 1999.

GCDL. Daily new confirmed COVID-19 cases per million people. **Our World in Data**, [2022]. Disponível em: <https://ourworldindata.org/explorers/coronavirus--data-explorer?zoomToSelection=true&time=2020-03-01..latest&facet=none&pickerSort=asc&pickerMetric=location&Metric=Confirmed+cases&Interval=7-day+rolling+average&Relative+to+Population=true&Color+by+test+positivity=false&country=~BRA>. Acesso em: 10 ago. 2022.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting:** Principles and Practice. 2nd edition. Melbourne, Austrália: OTexts, 2018. Disponível em: <https://otexts.com/fpp2/>. Acesso em: 12 mar. 2022.

INFLUX DATA: **Documentation**. [2022]. Disponível em: <https://docs.influxdata.com/influxdb/v2.1/>. Acesso em: 10 ago. 2022.

INFLUXDB data elements. influxdata Documentation. Disponível em: <https://docs.influxdata.com/influxdb/v2.1/reference/key-concepts/data-elements/>. Acesso em: 13 mar. 2022.

INFLUXDB System Properties. **DB-ENGINES**. Disponível em: <https://db-engines.com/en/system/InfluxDB>. Acesso em: 13 mar. 2022.

JEBB, A. T. et al. Time series analysis for psychological research: examining and forecasting change. **Frontiers in Psychology**, v. 6, art. 727, 2015.

REFERÊNCIAS



LEITE, G. F. M. **Previsão, por meio de análise de Séries Temporais, de leitos de Unidades de Terapia Intensiva - UTI na rede de atendimento do Sistema Único de Saúde - SUS no município de Goiânia.** TCC (MBA em Ciências de Dados) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo, 2021.

MALASINGHE, L. P.; RAMZAN, N.; DAHAL, K. Remote patient monitoring: a comprehensive study. **Journal of Ambient Intelligence and Humanized Computing**, v. 10, 2019.

MENDES, E. V. O conceito de condições de saúde. Disponível em: <https://www.nescon.medicina.ufmg.br/biblioteca/imagem/3936.pdf>. Acesso em: 27 fev. 2022.

MORETTIN, P. A.; TOLOI, C. M. C. **Análise de séries temporais.** 3^{ed}. São Paulo: Blucher, 2018.

SATO, R. C. Gerenciamento de doenças utilizando séries temporais com o modelo ARIMA. **Revendo Ciências Básicas**, v. 11, n. 1, mar. 2013.

TIME Series Analysis: The Basics. **Australian Bureau of Statistics**. Disponível em: <https://www.abs.gov.au/websitedbs/d3310114.nsf/home/time+series+analysis:+the+basics>. Acesso em: 8 mar. 2022.

WRITING Data with the HTTP API. **Influxdata Docs Archive**. Disponível em: https://archive.docs.influxdata.com/influxdb/v1.2/guides/writing_data/. Acesso em: 17 mar. 2022.

UNIDADE 4

BRUGGEN, R. van. Contract tracing wexample #cypher #neo4j. **GitHubGist**, [2022]. Disponível em: <https://gist.github.com/rvanbruggen/a916e640336f91767be23ea9288fccc3>. Acesso em: 10 ago. 2022.

NEO4J GRAPH DATABASE. Disponível em: <https://neo4j.com/product/neo4j-graph-database/?ref=product>. Acesso em: 6 maio. 2022.

NETTO, P. O. B; JURKIEWICZ, S. **Grafos:** introdução e prática. São Paulo: Blucher, 2009.

ORGANIZAÇÃO PAN-AMERICANA DA SAÚDE. Rastreamento de contatos no contexto da COVID-19. Orientação provisória. 2021. Disponível em: <https://iris.paho.org/handle/10665.2/53893>. Acesso em: 21 abr. 2022.

OSTROSKI, A.; MENONCINI, L. Teoria dos grafos e aplicações. **Synergismus scyentifica UTFPR**, v. 4, n. 2, 2009.

THE NEO4J GRAPH DATA PLATFORM. Disponível em: <https://neo4j.com/product/>. Acesso em: 6 maio 2022.

THE RELIANTS PROJECT. Concept 1: Seven Bridges of Konigsberg and the Origin of Network Graphs. Disponível em: <https://www.reliantsproject.com/2020/06/07/concept-1-seven-bridges-of-konigsberg-and-the-origin-of-network-graphs/>. Acesso em: 4 maio 2022.

REFERÊNCIAS



WIKIMEDIA COMMONS. [Sem título]. [2022]. 1 gravura Disponível em: https://commons.wikimedia.org/wiki/File:Konigsberg_Bridge.png. Acesso em: 10 ago. 2022.

WORLD HEALTH ORGANIZATION. Considerations in the investigation of cases and clusters of COVID-19. Disponível em: <https://www.who.int/publications/i/item/considerations-in-the-investigation-of-cases-and-clusters-of-covid-19>. Acesso em: 21 abr. 2022.

WORLD HEALTH ORGANIZATION. Public health surveillance for COVID-19. Disponível em: <https://apps.who.int/iris/handle/10665/333752>. Acesso em: 29 abr. 2022.

UNIDADE 5

ÂBUALKISHIK, A. Z. Hadoop and Big Data challenges. **Journal of Theoretical and Applied Information Technology**, v. 97, n. 12, jun. 2019.

CARTILHA sobre Prontuário Eletrônico - A Certificação de Sistemas de Registro Eletrônico de Saúde. **Conselho Federal de Medicina e Sociedade Brasileira de Informática em Saúde**. Disponível em: http://www.sbis.org.br/certificacao/Cartilha_SBIS_CFM_Prontuario_Eletronico_fev_2012.pdf. Acesso em: 17 maio 2022.

CFM. **Resolução CFM nº 1.638/2002**. Disponível em: <https://sistemas.cfm.org.br/normas/visualizar/resolucoes/BR/2002/1638>. Acesso em: 17 maio 2022.

CHANG, F. et al. Bigtable: A Distributed Storage System for Structured Data. **Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)**. 2006. Disponível em: <https://static.googleusercontent.com/media/research.google.com/pt-BR//archive/bigtable-osdi06.pdf>. Acesso em: 18 maio 2022.

COPELAND, G. P.; KHOSHAFIAN, S. N. A DECOMPOSITION STORAGE MODEL. **ACM SIGMOD Record**, v. 14, n. 4, 1985.

HADOOP. **HDFS Architecture Guide**. [2022]. Disponível em: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Acesso em: 10 ago. 2022.

MONICA, M.; KUMAR, K. R. Survey on Big Data by Coordinating Mapreduce to Integrate Variety of Data. **International Journal of Science and Research**, v. 2, n. 11, nov. 2013.

NIMA, P. Comparative study on MongoDB and HBase. **Technical Report**. dez. 2018. Disponível em: https://www.researchgate.net/publication/330837495_Comparative_study_on_MongoDB_and_HBase. Acesso em: 22 maio 2022.

CFM. **Resolução CFM nº 1.638/2002**. Disponível em: <https://sistemas.cfm.org.br/normas/visualizar/resolucoes/BR/2002/1638>. Acesso em: 17 maio 2022.

SILVA, L. F. C. et al. Banco de Dados Não Relacional. [S.I.]:[S.n.], 2021

CONFIRA SUAS RESPOSTAS



UNIDADE 1

1. C. As características apresentadas representam Volume, Velocidade e Variedade.
2. A. Tabelas de bancos de dados relacionais são conhecidas por sua estrutura rígida, servindo como repositório de dados estruturados.
3. D. A afirmação apresentada representa uma das características do processo de downsizing adotado por instituições nas décadas de 1970 e 1980.
4. B. A descrição é a de sistemas cliente-servidor padrão.
5. C. Consistência eventual implica sacrificar consistência em prol de disponibilidade e tolerância a partções na rede de comunicação. Portanto, AP.

UNIDADE 2

1. B. Reduzir desperdícios e eliminar gargalos são dois objetivos passíveis de serem atingidos com a adoção de um workflow.
2. D. A afirmação apresentada corresponde com a definição correta de um atributo no modelo relacional.
3. E. Bancos de dados orientados a documentos aceitam, de acordo com suas especificações individuais, documentos nos formatos citados na alternativa.
4. A. A ordem das ações apresentadas nesta alternativa corresponde com a ordem dos comandos listados na questão.

UNIDADE 3

1. E. A afirmação apresentada corresponde à tipificação de séries temporais com relação ao que se observa e com qual frequência.
2. C. A afirmação apresentada corresponde ao conceito de bucket, tanto para a sua finalidade quanto para a retenção dos dados que armazena.

CONFIRA SUAS RESPOSTAS



3. A. O conteúdo de campos e tags é passível de uso como filtro, visando limitar os resultados obtidos por meio da execução de consultas.

UNIDADE 4

1. C. A descrição condiz com a definição do processo de rastreamento de contatos.
2. A. As relações consideradas no processo de rastreamento incluem pessoas e locais, ocorrendo de forma direta e indireta em diferentes níveis.
3. B. Os grafos podem ser classificados como direcionados ou não direcionados, dependendo da definição ou não de um sentido de origem/destino nas suas arestas.
4. D. A notação define uma aresta direcionada da esquerda à direita.

UNIDADE 5

1. D. A descrição confere com o conceito de variedade em cenários de Big Data.
2. E. Embora tenha avançado muito nos últimos anos, a padronização ainda não resolreu todas as questões envolvendo as diferenças de representação de dados com a mesma semântica.
3. A. O significado de decomposição para o modelo de dados está correto.
4. B. A descrição confere com o conceito de famílias de colunas.
5. C. A compressão é configurável para cada família de colunas de uma tabela.

MEU ESPAÇO



MEU ESPAÇO



MEU ESPAÇO

