WBA0755_v1.0

PROJETO EM CIÊNCIA DE DADOS COM SOLUÇÕES PARA PROCESSAMENTO PARALELO E DISTRIBUÍDO DE DADOS

Marcelo Tavares de Lima

Projeto em ciência de dados com soluções para processamento paralelo e distribuído de dados

1ª edição

Londrina
Editora e Distribuidora Educacional S.A.
2019

Dados Internacionais de Catalogação na Publicação (CIP)

Lima, Marcelo Tavares de
L732p Projeto em ciência de dados com soluções para processamento paralelo e distribuído de dados/ Marcelo Tavares de Lima, – Londrina: Editora e Distribuidora
102 p.

ISBN 978-85-522-1645-2

1. Processamento distribuído. 2. Big Data. I. Lima, Marcelo Tavares de. Título.

CDD 004

Thamiris Mantovani CRB: 8/9491

2019
Editora e Distribuidora Educacional S.A.
Avenida Paris, 675 – Parque Residencial João Piza
CEP: 86041-100 — Londrina — PR
e-mail: editora.educacional@kroton.com.br
Homepage: http://www.kroton.com.br/

SUMÁRIO

Apresentação da disciplina

Seja bem-vindo ao mundo da ciência de dados!

Esta disciplina apresentará a você os principais conceitos e definições de arquiteturas de redes, sistemas distribuídos de dados, tratamento de grandes volumes de dados e conceitos associados. Serão detalhados diversos tipos de arquiteturas de rede como sistemas *peer-to-peer*, *clusters* e *grids*. Também, serão apresentadas arquiteturas paralelas e de dados distribuídos, assim como serão identificadas as especificidades de cada uma. A intenção é tornar você conhecedor dos diversos tipos de sistemas existentes para que saiba como lidar com cada um ao longo de sua carreira profissional.

Também, serão apresentadas algumas ferramentas, dentre muitas existentes para a elaboração de imagens, que apresentarão suas informações de maneira clara, sucinta e eficiente. Dentre as ferramentas a serem apresentadas, algumas exigirão um certo conhecimento de linguagem de programação, como linguagem R e linguagem Python.

Serão apresentados, também, alguns conceitos de arquiteturas de dados como as ferramentas OLAP, apropriadas para consulta de grandes bases de dados, por meio de recursos diversos a depender da estrutura física disponível para a sua realização.

Desejamos que você aproveite bastante este momento de estudo do conteúdo. Que ele possa trazer *insights* para a sua vida, tanto acadêmica quanto profissional, e que ao final deste curso, você possa sair com um diferencial em sua formação! Bons estudos!

Introdução a soluções para processamento paralelo e distribuído de dados. Gerenciamento de *clusters* e *grids*

Autor: Marcelo Tavares de Lima

Objetivos

- Apresentar conceitos introdutórios de processamento paralelo de dados.
- Apresentar conceitos introdutórios de processamento distribuído de dados.
- Apresentar soluções para processamento paralelo e distribuído de dados e gerenciamento de *clusters* e *grids*.

1. Introdução

Olá aluno! Seja bem-vindo a esta aula. Nela iremos apresentar para você uma introdução a soluções para processamento paralelo e distribuído de dados e gerenciamento de *clusters* e *grids*. Portanto, para iniciar, vamos introduzir alguns conceitos básicos necessários para o bom desenvolvimento do tema.

O processamento de grandes bases de dados exige muito dos sistemas onde são executados. Por isso, é importante se preocupar com este item. Sistemas de armazenamento de dados precisam não somente ter grande capacidade de armazenamento, mas, também, devem ter capacidade de processamento em tempo adequado e hábil para o bom andamento do trabalho de analistas e cientistas de dados. Os sistemas distribuídos podem auxiliar nessa tarefa desafiadora.

Desejamos que esta aula traga conhecimento significativo para a sua formação e que esse conhecimento possa se tornar um diferencial em sua carreira profissional e acadêmica. Bons estudos!

2. Processamento paralelo e distribuído de dados

Está cada vez mais comum o uso de grandes bases de dados para o processo de tomada de decisões. Bases gigantescas de dados são usualmente conhecidas como *big data*. No entanto, com o surgimento de *big data* surgem novos desafios, o tempo de processamento, que é essencial para alavancar negócios, decisões etc.

O tempo de processamento de dados é fator decisivo para o pioneirismo de muitas atividades, sejam corporativas, comerciais ou acadêmicas. Por isso, surgiu o desafio da busca por processamentos mais rápidos e mais confiáveis de grandes bases de dados.

O surgimento dos outros tipos de processamento de dados, diferentes dos usuais, deu início à corrida pela busca do menor tempo de execução de dados. Um resultado dessa busca é o que se conhece por processamento paralelo, que é considerado um tipo de inovação essencial para o tratamento de grandes bases de dados.

O conceito de processamento paralelo é, de certa forma, simples e compreensível por qualquer pessoa, inclusive aquelas que não realizam análise de dados. É uma maneira mais eficiente de lidar com informações e, com ênfase, principalmente, na exploração de situações ou acontecimentos simultâneos durante a execução de um programa computacional. O que se pode dizer é que, na prática, é o uso de forma simultânea de mais de uma unidade de processamento de dados (CPU) para a realização de um trabalho.

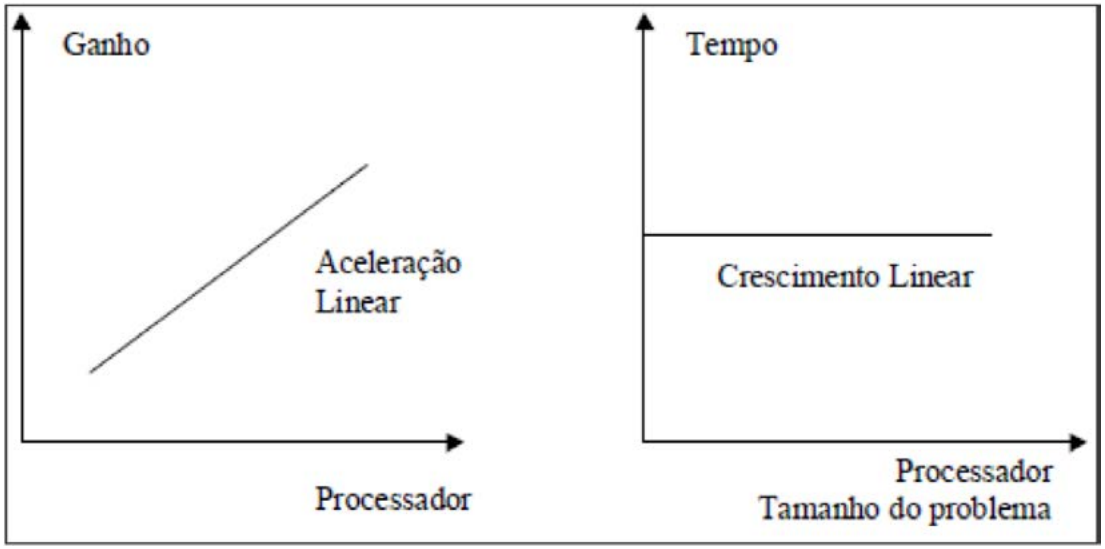
Considera-se como uma desvantagem do processamento paralelo, segundo Meyer (2006), o alto custo de investimento em hardware e software necessário para construir ambientes que se utilizem do recurso de processamento paralelo de maneira apropriada e significativa. Segundo Navaux, De Rose e Pilla (2011, p. 1) “a solução tendeu para o emprego de vários processadores trabalhando em conjunto na obtenção de uma maior capacidade de processamento”. Foi aí que surgiu o processamento paralelo com o propósito de utilizar técnicas diversas de concorrência que atendessem necessidades de aceleração de processamento de dados.

Segundo Meyer (2006, p. 2), “a computação paralela é alcançada através da divisão do problema em tarefas menores que podem ser simultaneamente processadas por múltiplos processadores”. O autor exemplifica o conceito aplicando à soma de dois vetores A e B, que pode ser realizada com o uso de dois processadores, no qual o primeiro realiza a soma da primeira metade do vetor A com a primeira metade do segundo vetor, o vetor B. O segundo processador executa a soma da segunda metade do vetor A com a segunda metade do vetor B.

Existem duas métricas que são usadas para a avaliação da eficiência de um sistema paralelo (MEYER, 2006), que são a aceleração linear e o crescimento linear. O autor afirma que "a aceleração linear não leva em consideração o tamanho do problema e sim o tamanho do sistema" (MEYER, 2006, p. 2). O que ele quis dizer é que se o hardware for duplicado, uma tarefa poderá ser executada na metade do tempo se for executada com um processador apenas. Mede-se a aceleração linear pela razão entre o tempo de execução com um processador e o tempo de execução com mais de um processador, a qual representa o ganho por ser implantado.

A outra medida, o crescimento linear, é utilizada para medir a habilidade de crescimento do sistema e, também, do problema. Considerando que o hardware seja duplicado, espera-se que o sistema passe a ser capaz de executar um problema, duas vezes mais, considerando o mesmo intervalo de tempo (MEYER, 2006). O uso desta métrica serve para medir a habilidade de crescimento do sistema e, também, do problema. Na prática, é verificar quando o hardware é duplicado, se o sistema é capaz de executar um problema com o dobro de complexidade no mesmo intervalo de tempo que o sistema original. A Figura 1 apresenta, de forma ilustrativa, as duas propriedades, medidas pelas duas métricas descritas.

Figura 1 – Métricas de um sistema paralelo



Fonte: Meyer (2006).

Navaux, De Rose e Pilla (2011, p. 1) afirmam que

> o desenvolvimento da área de processamento paralelo levou ao aproveitamento destas técnicas no projeto de *chips* de processadores, microprocessadores e também em máquinas especializadas em realizar grandes quantidades de operações por segundo, conhecidas como supercomputadores. Estas últimas proporcionaram um maior desempenho na resolução de problemas, sendo, por isso, essa área conhecida também como processamento de alto desempenho – PAD (em inglês, *High Performance Computing* – HPC).

É crescente o investimento em tecnologias como os supercomputadores em todas as áreas do conhecimento, em especial, a ciência de dados e áreas especiais como a meteorologia (previsão do tempo), a busca por petróleo, área de simulações físicas e a matemática computacional. O processamento de alto desempenho é uma área da ciência da computação que veio para solucionar problemas complexos como os encontrados nessas áreas, que antes eram resolvidos com a simplificação dos modelos, resultando em respostas menos precisas e com margem de erro considerável (NAVAUX; DE ROSE; PILLA, 2011).

Em um processamento paralelo em que vários processadores são utilizados para resolver um mesmo problema, cada um “ataca” uma parte distinta dos dados e ficam se comunicando para realizar a troca de resultados parciais e para a junção final de seus resultados.

Existe uma série de recursos computacionais que trabalham de forma paralela, como por exemplo, uma aplicação em C ou Java com vários subprogramas conhecidos como *threads*, ou uma aplicação elaborada em Java que se utiliza de tecnologia RMI (*Remote Method Invocation*) para facilitar o desenvolvimento de aplicações realizadas de forma distribuída.

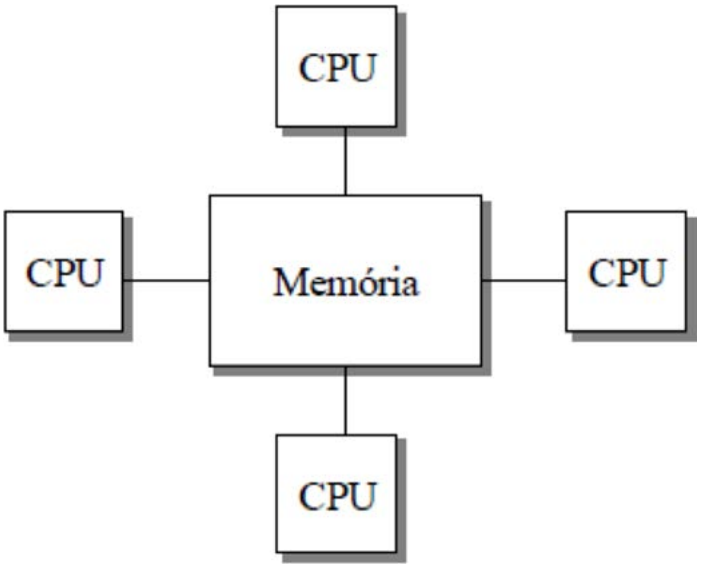
Estudiosos do assunto elencam uma série de vantagens em se utilizar processamento paralelo. Navaux, De Rose e Pilla (2011, p. 25) afirmam que dentre as motivações para o uso de processamento paralelo, estão:

- Desempenho: característica que surje com a redução do tempo de execução de um problema por utilização de diversas unidades de processamento aplicadas no mesmo problema.
- Tolerância a falhas: a probabilidade de ocorrência de falhas também se reduz, pois cada uma das unidades de processamento ativa executa o mesmo problema, permitindo a eleição do melhor resultado no final.
- Modelagem: é esperado que a complexidade da implementação de uma modelagem se reduza.
- Aproveitamento de recursos: é esperado que os recursos disponíveis sejam mais aproveitados na execução de resolução de problemas.

Em um processamento paralelo, a maneira como os processadores e dispositivos de memória realizam comunicação entre si é o que define a arquitetura de máquinas paralelas (MEYER, 2006). Segundo o autor, os principais modelos de memória são: memória compartilhada (*shared memory*) e memória distribuída (*shared nothing*).

Segundo Meyer (2006), em arquiteturas de memória compartilhada os processadores envolvidos na resolução do mesmo problema têm acesso à memória principal e às unidades de disco por uma rede de conexão entre eles. A Figura 2 apresenta um esquema de arquitetura em processamento paralelo com memória compartilhada.

Figura 2 – Modelo de memória compartilhada



Fonte: Meyer (2006).

Meyer (2006) afirma que como vantagem de um processamento paralelo com memória compartilhada pode-se citar o compartilhamento de dados e de tarefas de maneira rápida entre os processadores envolvidos e, como desvantagem, a expansibilidade limitada da possibilidade de processadores em uma mesma arquitetura e a disponibilidade de tráfego de dados entre a memória e os nós do sistema.

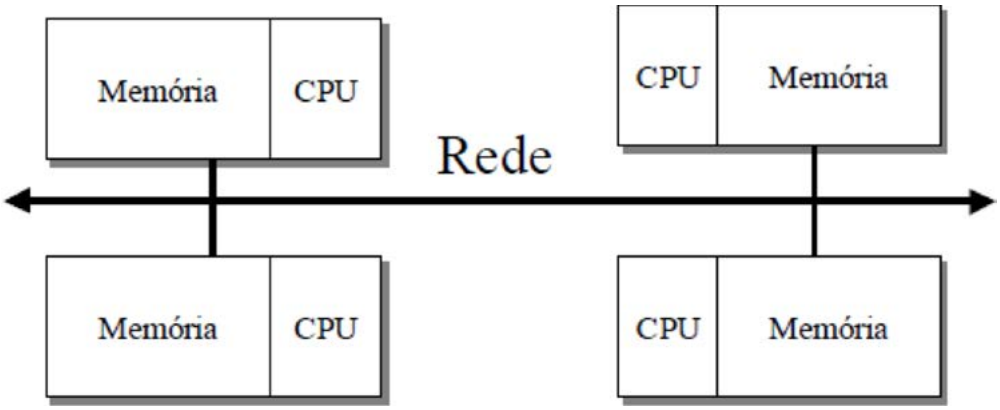
PARA SABER MAIS

É possível explorar sistemas em paralelo em diversos níveis. É o que se denomina de grão de paralelismo, que é um conceito importante e fundamental em modelagem por sistemas paralelos. Considerando-se o conceito de grão de paralelismo, pode-se afirmar que existem grão grosso, grão médio e grão fino.

O modelo de memória distribuída permite que cada nó do sistema tenha acesso exclusivo à sua memória. Assim, cada processador pode realizar operações de maneira independente, superando a necessidade de um

endereçamento global de espaço de memória. No entanto, quando um dos nós do sistema necessitar de dados armazenados em outro nó, cabe ao programador definir a forma de obtenção. A Figura 3 apresenta um sistema paralelo com modelo de memória distribuída.

Figura 3 – Modelo de memória distribuída



Fonte: Meyer (2006).

Assim como o modelo de memória compartilhada, o modelo de memória distribuída também apresenta vantagens e desvantagens. Falando de suas vantagens, pode-se afirmar que, segundo Meyer (2006, p. 3), "a principal vantagem da memória distribuída é o baixo custo aliado a uma alta expansibilidade e disponibilidade". Ainda segundo o autor, tal vantagem permite que possa haver um crescimento incremental, o qual permite o suporte de um grande número de processadores, minimizando, assim, possíveis interferências que ocorrem quando há compartilhamento de recursos. Em se tratando de desvantagem, o autor cita a responsabilidade atribuída ao programador da necessidade, quando houver, de executar a troca de mensagens entre os nós do sistema.

A partir deste momento, focaremos em sistemas paralelos de modelo distribuídos, e para dar continuidade ao assunto, apresentamos a definição deste tipo de sistema segundo outros autores e estudiosos da temática.

Um sistema distribuído, segundo Colouris, Dollimore e Kindberg (2013, p. 1) “é aquele no qual os componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens”. Os mesmos autores afirmam que tal definição leva à percepção de algumas características importantes, que são a concorrência de componentes, a falta de um relógio global e a falha de componentes independentes.

A execução concorrente de programas é, praticamente, norma em uma rede de computadores. É possível que um analista realize um trabalho em seu computador enquanto outro analista também executa sua atividade em sua máquina local, compartilhando recursos como páginas de internet ou outros arquivos quando houver necessidade. A capacidade de um sistema gerenciar recursos compartilhados amplia-se quando mais recursos são adicionados ao sistema. Recursos podem ser entendidos como todo hardware ou software que agregue a um sistema, como um computador, por exemplo.

Entende-se como característica de inexistência de relógio global a falta de noção global única de tempo correto, apesar de a coordenação entre os componentes de um sistema dependerem de uma noção compartilhada do tempo em que as ações ocorram. Essa característica é, segundo Colouris, Dollimore e Kindberg (2013, p. 2) “uma consequência direta do fato de que a única comunicação se dá por meio do envio de mensagens em uma rede”.

Falhas independentes se referem à capacidade que cada componente do sistema tem em poder falhar independentemente, sendo que outros componentes podem permanecer em plena atividade. Em sistemas distribuídos, as falhas resultam em isolamento do componente que falhou, ou seja, os componentes que falham ficam isolados da rede, podendo não parar de funcionar, apenas ficarem isolados.

Colouris, Dollimore e Kindberg (2013, p. 2) afirmam que

> a principal motivação para construir e usar sistemas distribuídos é proveniente do desejo de compartilhar recursos. O termo “recurso” é bastante abstrato, mas caracteriza bem o conjunto de coisas que podem ser compartilhadas de maneira útil em um sistema de computadores interligados em rede. Ele abrange desde componentes de hardware, como discos e impressoras, até entidades definidas pelo software, como arquivos, bancos de dados e objetos de dados de todos os tipos.

Navaux, De Rose e Pilla (2011, p. 26) declaram que o uso de processamento distribuído tem como motivação “a modelagem e o aproveitamento de recursos”, nos quais os componentes do sistema comumente estão fisicamente mais afastados, o que traz um custo de comunicação maior, chamado de “alta latência”.

Sistemas distribuídos abrangem uma grande gama de recursos tecnológicos atualmente utilizados. Por exemplo a internet, o e-commerce, jogos online e redes sociais diversas. São recursos utilizados comumente e muito frequentemente por milhares de pessoas, o que torna necessário, para os profissionais da área de computação, um conhecimento fundamental e essencial.

Colouris, Dollimore e Kindberg (2013) apresentam uma série de exemplos de sistemas distribuídos. Um exemplo que se destaca são os sistemas de pesquisa na web. Dada a magnitude que a web atingiu, em termos de números, tornou-se um grande desafio realizar buscas de conteúdos específicos. Nessa área, o Google é o líder no mercado de tecnologia de pesquisas na web. Para se manter líder, o Google investiu muito em infraestrutura sofisticada de sistemas distribuídos focados no apoio à pesquisa.

Outra área de grande importância na utilização de sistemas distribuídos é a área de negócios em finanças. Ela assumiu o pioneirismo no uso de sistemas distribuídos dada a sua necessidade de acesso rápido a uma gama de informações econômicas, políticas, preços de ativos etc.

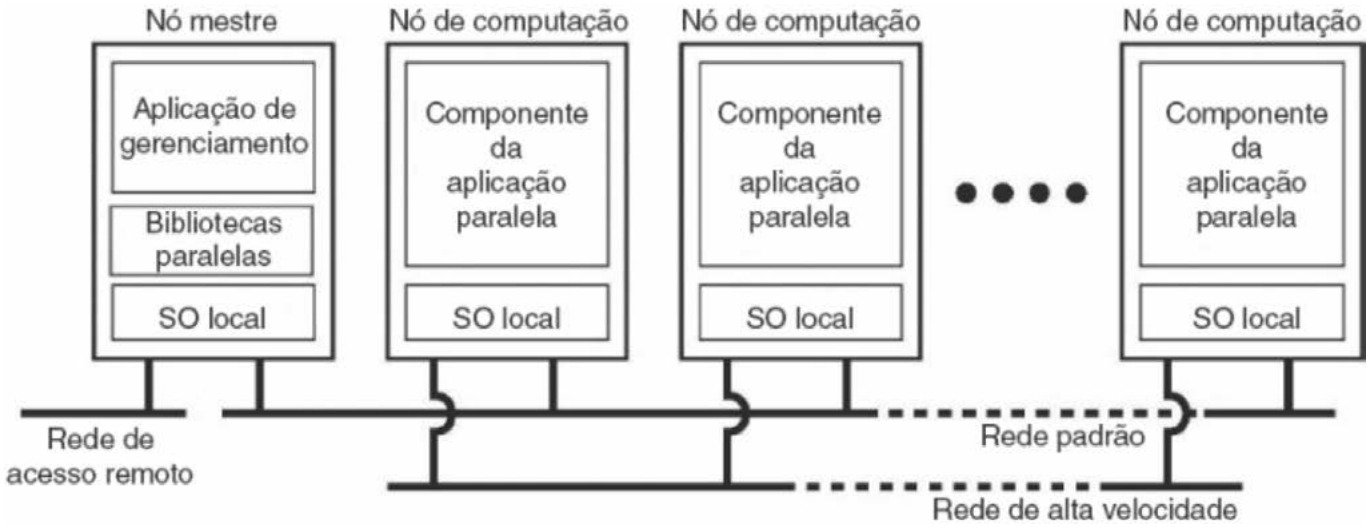
2.1 Gerenciamento de *clusters* e *grids*

Bacellar (2010, p. 3) define

> *cluster* é um sistema distribuído de computadores independentes e interligados, cujo o objetivo é suprir a necessidade de um grande poder computacional com um conjunto de computadores de forma transparente ao usuário.

A Figura 4 apresenta um exemplo bastante típico de arquitetura em *clusters*, a qual, segundo Martins (2019), tornou-se muito popular quando "a razão preço/desempenho de computadores pessoais e estações de trabalho melhorou" (MARTINS, 2019, p. 4).

Figura 4 – Exemplo típico de *cluster*



Fonte: Martins (2019).

O nó de um *cluster* pode representar uma única máquina (computador) ou um computador com diversos processadores simétricos (memória compartilhada). Como principais características, pode-se afirmar que cada um dos nós pertencentes a um *cluster* tem a sua própria memória, seus dispositivos de entrada/saída e seu próprio sistema operacional. Pode-se dizer, também, que fisicamente, cada nó pode ser montado em um único gabinete ou os nós podem estar separados entre si,

conectados unicamente por uma rede local (LAN). Em relação ao seu tamanho, podem ser classificados como *clusters* pequenos quando possui um pequeno número de nós, ou grandes, quando constituídos por milhares de processadores.

A arquitetura de computadores em *cluster*, segundo Meyer (2006), surgiu por volta da década de 90, com motivação semelhante ao que afirmou Martins (2019), motivada por baixo custo dos componentes necessários para sua elaboração.

Para que funcione, segundo Bacellar (2010, p. 3), "o cluster possui três condições primordiais para o seu devido funcionamento, sistema operacional, hardware e biblioteca de comunicação". Os nós de um cluster são controlados por um único nó mestre, o qual tem como tarefa típica, a manipulação da alocação dos nós controlados, segundo Martins (2019, p. 6), "a determinado programa paralelo, manter fila de *jobs*, e fornecer uma interface para o usuário".

Meyer (2006, p. 6) afirma que

> a construção de um cluster é tida como extremamente fácil. Entretanto, fazer com que computadores individuais operem como um sistema integrado com o propósito de resolver um dado problema é mais difícil.

O autor também declara que dentre os desafios existentes para a elaboração desse tipo de arquitetura, um dos maiores se refere ao desenvolvimento de aplicações que possam tirar algum proveito da infraestrutura de um cluster, pois precisam ser paralelizadas para estarem aptas a retirar algum tipo de proveito do processamento paralelo, além de precisarem ser desenvolvidas com ferramentas apropriadas para o seu ambiente de instalação.

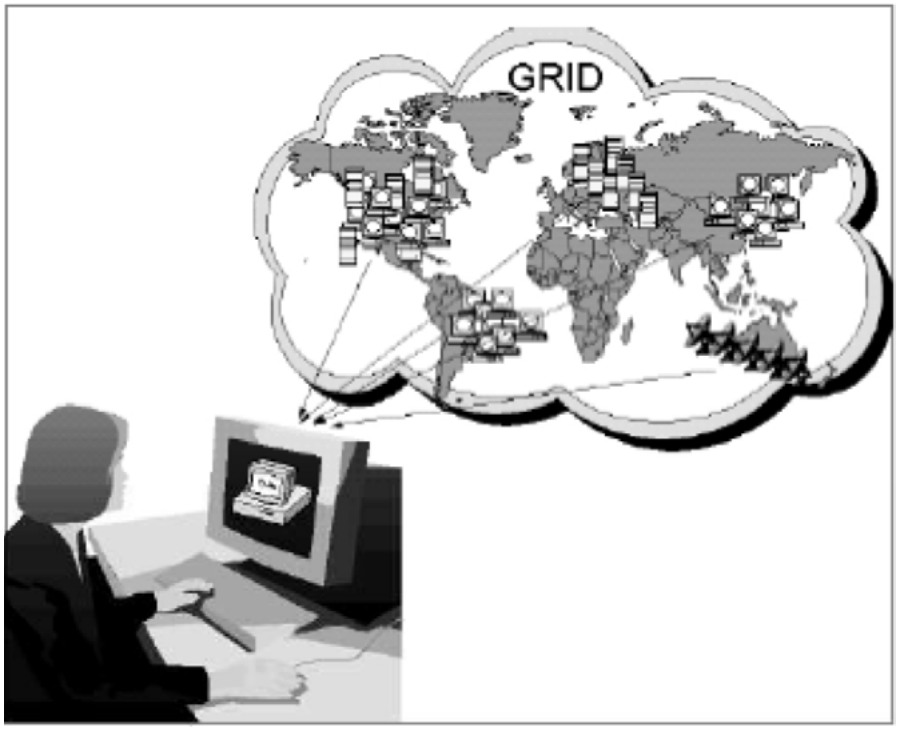
ASSIMILE

Diversas aplicações têm sido criadas para lidar com a paralelização de estruturas paralelas de *clusters*. As mais aceitas, segundo Meyer (2006), são as estratégias baseadas em troca de mensagens explícitas, principalmente por terem tido mais sucesso em ampla variedade de aplicações e de plataformas.

Outro modelo de computação existente é o de computação em *grid* ou grade. Sua descrição é realizada por meio de uma analogia ao conceito de redes de transmissão de eletricidade. Com este exemplo, podemos perceber que ao se ligar um aparelho elétrico na tomada, não temos a mínima ideia de onde a eletricidade ali utilizada é gerada. A concessionária de energia elétrica responsável pelo fornecimento do serviço disponibiliza interface para o sistema, que é complexo, de geração e de transmissão da energia utilizada.

Com a analogia de uma rede de distribuição de energia para descrever um modelo de *grid* ou grades, pode-se afirmar, segundo Meyer (2006, p. 8), que "a visão de *Grid* é similar (ou almeja ser), ou seja, diversos recursos computacionais geograficamente distribuídos podem ser agregados para formar um supercomputador virtual". A Figura 5 apresenta um esquema de modelo em *grid* ou grades sob o ponto de vista do usuário.

Figura 5 – Esquema de modelo de *grid* para o usuário



Fonte: Meyer (2006).

O que a Figura 5 tenta apresentar é que os recursos utilizados para estruturar uma rede em *grid* não precisam ser visíveis para o seu usuário, por isso, utilizam a analogia com o fornecimento de energia elétrica, pois o usuário não tem a mínima ideia de onde a energia é gerada para que lhe possa ser fornecida.

Um bom exemplo de rede de *grid* trata-se da internet, a qual é uma infraestrutura de rede que conecta milhões de computadores espalhados por todo o mundo. A rede *World Wide Web*, segundo Meyer (2006, p. 8), "é um serviço de compartilhamento de informações construídos sobre a internet". De maneira análoga, o *grid* também é um serviço construído sobre a internet. A diferença é que este último vai um passo à frente, embasado por sua tecnologia, que vai além de informação etc., que pode ser compartilhada.

Tanebaum e Steen (2008 *apud* PEREIRA, 2019, p. 35) descrevem computação em *grid* como "um conjunto de máquinas com características diferentes, podendo o hardware e os sistemas

operacionais serem de fabricantes diferentes". Tal característica atribui o adjetivo heterogêneo aos sistemas de computação em *grid*, o qual também pode agregar vários *clusters*. O autor apresenta como exemplo de *grid* o CineGrid Brasil (CineGrid, 2019), que é uma comunidade interdisciplinar com foco em pesquisa, desenvolvimento e produção de ferramentas colaborativas multimídias.

Pode parecer que *clusters* e *grids* são a mesma coisa, no entanto, existe uma característica que os diferencia fundamentalmente. Trata-se da homogeneidade atribuída aos sistemas distribuídos *clusters* e da heterogeneidade atribuída aos *grids*. Segundo Pereira (2019 p. 36), *clusters* "são criados para executar alguma tarefa específica que, em geral, necessita de um alto poder de processamento", por exemplo, o treinamento de redes neurais artificiais de aprendizagem profunda (*deep learning*) para uso em ferramentas de *chat* online. Já os *grids*, segundo o mesmo autor, "são criados para executarem diferentes tarefas, de certa maneira relacionadas entre si, formando um centro de pesquisas de caráter multidisciplinar" (PEREIRA, 2019, p. 37). Podem ser pensados com um conjunto de *clusters*, no qual cada *cluster* é responsável por uma atividade específica.

O estudo de sistemas distribuídos é amplo e não se esgota neste texto. Eles são extremamente úteis para estruturar sistemas de armazenamento, tratamento e consulta de dados. Também são responsáveis por coleta de informações, como os sistemas de medição meteorológica.

Dada a grande extensão do assunto, sugerimos que você investigue mais sobre ele. Que possa buscar e encontrar mais informações e com elas entender o funcionamento de sistemas distribuídos para extrair deles os melhores resultados possíveis. Desejamos bons estudos!

TEORIA EM PRÁTICA

Imagine que você trabalha no departamento de pesquisa de mercado em uma empresa. Sua responsabilidade é gerenciar uma equipe de funcionários aptos para lidar com grandes bases de dados, os *big data*. Os equipamentos tecnológicos que sua equipe utiliza estão ficando obsoletos para lidar com bases de dados tão grande quanto às que vocês conseguem manipular, e o sistema de gerenciamento de dados também está se tornando obsoleto. A partir deste cenário, você reúne sua equipe e começa a planejar estratégias de melhorias do ferramental tecnológico e da rede de dados que fazem uso. Você precisa decidir qual o melhor sistema de rede para implantar, de acordo com as necessidades de sua equipe de trabalho. Uma rede com processamento paralelo resolve? Caso escolha esse tipo de sistema, é melhor um sistema de memória compartilhada ou distribuída? Mas, vocês podem concluir que é melhor um sistema de computação em *grids*, pois você precisa estar em rede com outras unidades da empresa. Enfim, você precisa avaliar tudo isso dentre outras características e necessidades para o seu trabalho e o da sua equipe. É um desafio, já que todo o trabalho é realizado com prazos muito curtos, o que exige demais de você e sua equipe. Por isso, precisa estar bem aparelhado! Bem!
O desafio está em suas mãos! Bom trabalho!

VERIFICAÇÃO DE LEITURA

1. O uso de grandes bases de dados, conhecidas como *big data*, tem se tornado cada vez mais frequente. Para encarar desafios de retiradas de informações e de

descoberta de conhecimentos com elas, busca-se cada vez mais a otimização de um item muito importante na corrida competitiva de mercado para o pioneirismo. De qual item estamos nos referindo?

Assinale a alternativa CORRETA.

a. Média dos dados.

b. Tempo de processamento.

c. Tamanho da base de dados.

d. Número de computadores.

e. Profissionais técnicos.

2. Métricas de avaliação de um sistema paralelo foram criadas para avaliar a sua eficiência. Como se chama a métrica utilizada para avaliar um sistema de processamento paralelo em termos do tamanho do sistema?

   Assinale a alternativa CORRETA.

   a. Crescimento linear.

   b. Memória compartilhada.

   c. Aceleração linear.

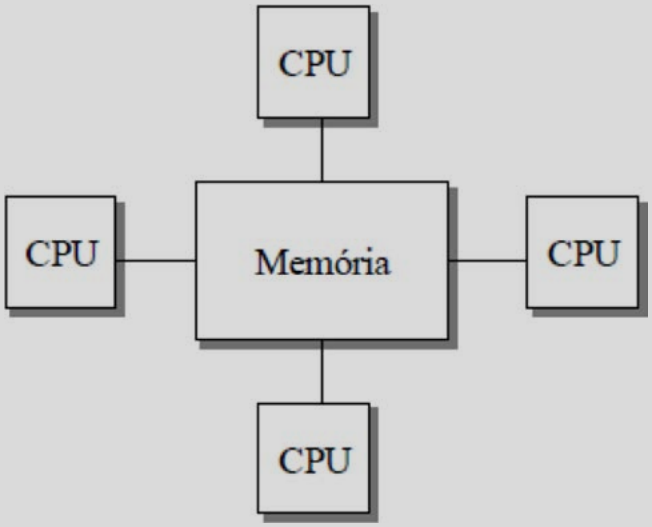
   d. Aceleração intervalar.

   e. Memória distribuída.

3. Em sistemas de processamento paralelo existem dois principais modelos. A Figura 6 a seguir apresenta um esquema de um destes modelos. Qual o nome do modelo representado pela figura?

Assinale a alternativa CORRETA.

Figura 6 – Esquema de computação paralela



Fonte: Meyer (2006).

a. Sistema distribuído.

b. Modelo em *grids*.

c. Memória distribuída.

d. Memória compartilhada.

e. Sistema compartilhado.

Referências bibliográficas

BACELLAR, H. V. **Cluster:** computação de alto desempenho. Campinas: Instituto de Computação, Universidade Estadual de Campinas, 2010. Disponível em: http://www.ic.unicamp.br/~ducatte/mo401/1s2010/T2/107077-t2.pdf. Acesso em: 17 set. 2019.

CINEGRID. **CineGrid Brasil**. [S.l., s.d.]. Disponível em: https://cinegridbr.org/. Acesso em: 18 set. 2019.

COLOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas distribuídos:** conceitos e projeto. 5. ed. Porto Alegre: Bookman, 2013. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582600542/cfi/0!/4/2@100:0.00. Acesso em: 17 set. 2019.

MARTINS, S. L. **Sistemas distribuídos**. Departamento de Ciência da Computação. Niterói: Universidade Federal Fluminense, 2019. Notas de aula. Disponível em: http://www.ic.uff.br/~simone/sd/contaulas/aula2.pdf. Acesso em: 17 set. 2019.

MEYER, L. A. V. C. **Uma visão geral dos sistemas distribuídos de cluster e grid e suas ferramentas para o processamento paralelo de dados**. 2006. IBGE [s.d.]. Disponível em: https://www.censo.gov.br/confest_e_confege/pesquisa_trabalhos/CD/palestras/368-1.pdf. Acesso em: 17 set. 2019.

NAVAUX, P. O. A.; De ROSE, C. A. F.; PILLA, L. L. **Fundamentos das arquiteturas para rocessamento paralelo e distribuído**. 2011. Laboratório de Banco de Dados. Departamento de Ciência da Computação – UFMG. Disponível em: http://www.lbd.dcc.ufmg.br/colecoes/erad-rs/2011/003.pdf. Acesso em: 17 set. 2019.

PEREIRA, C. S. **Sistemas distribuídos**. Londrina: Editora e Distribuidora Educacional S.A., 2019. 184 p.

Gabarito

Questão 1 – Resposta B

A corrida pela redução do tempo de processamento de grandes volumes de dados acelera o processo de obtenção de resultados e coloca em posição pioneira quem chega na frente.

Questão 2 – Resposta C

A aceleração linear é uma métrica utilizada para avaliar um processamento paralelo em termos do tamanho do sistema.

Questão 3 – Resposta D

O esquema gráfico apresentado se refere a um processamento paralelo de memória compartilhada.

Monitoramento e depuração de programas paralelos. Sistemas de armazenamento de dados distribuídos

Autor: Marcelo Tavares de Lima

Objetivos

- Conhecer conceitos e definições associados ao monitoramento e depuração de programas paralelos.
- Compreender conceitos fundamentais de bancos de dados.
- Compreender conceitos fundamentais de armazenamento de dados distribuídos.

1. Introdução

Olá aluno! Seja bem-vindo a esta leitura, na qual serão apresentados conceitos fundamentais de bancos de dados e de monitoramento e depuração de programas paralelos, assim como conceitos sobre armazenamento de dados distribuídos.

Para o bom aproveitamento é importante lembrar de conceitos básicos de sistemas paralelos e de sistemas distribuídos, assim como conceitos relacionados à arquitetura de cada um deles.

Desejamos que você tenha um excelente momento de estudos e que ele possa agregar conhecimento de forma significativa na sua formação profissional e acadêmica. Bons estudos!

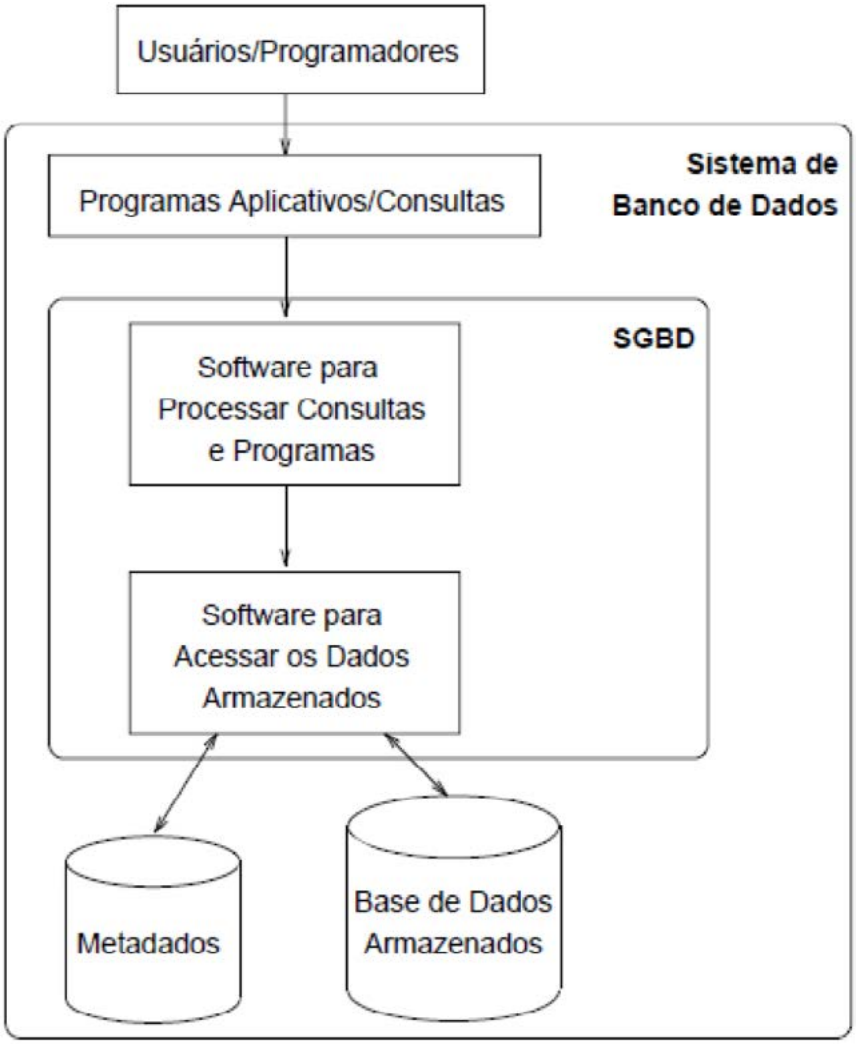
2. Conceitos fundamentais de bancos de dados

Segundo Ricarte (1996, p. 1), "um banco de dados é uma coleção de dados relacionados que pode ser armazenada sob alguma forma física". Associado a essa definição, o autor afirma ainda que "os dados armazenados em um banco de dados representam algum aspecto específico do mundo real" (RICARTE, 1996, p. 1).

Outro conceito bastante importante e que está associado ao banco de dados é o seu sistema de gerenciador de bancos de dados (SGBD), o qual é, basicamente, um software que manipula os dados armazenados com a execução de tarefas como criar, manipular e manter, além de realizar manutenção sobre tais dados. Já a pessoa ou grupo de pessoas responsável pela criação e manutenção de um banco de dados é referenciado como administrador do banco de dados.

Abrangendo todos os conceitos e elementos, um sistema de banco de dados (SBD) constitui-se pelo banco de dados em si e pelo sistema que o gerencia, ou seja, é uma aplicação que agrega outras aplicações sobre ela. A Figura 1 apresenta um esquema de sistema de banco de dados.

Figura 1 – Sistema simplificado de banco de dados



Fonte: Ricarte (1996).

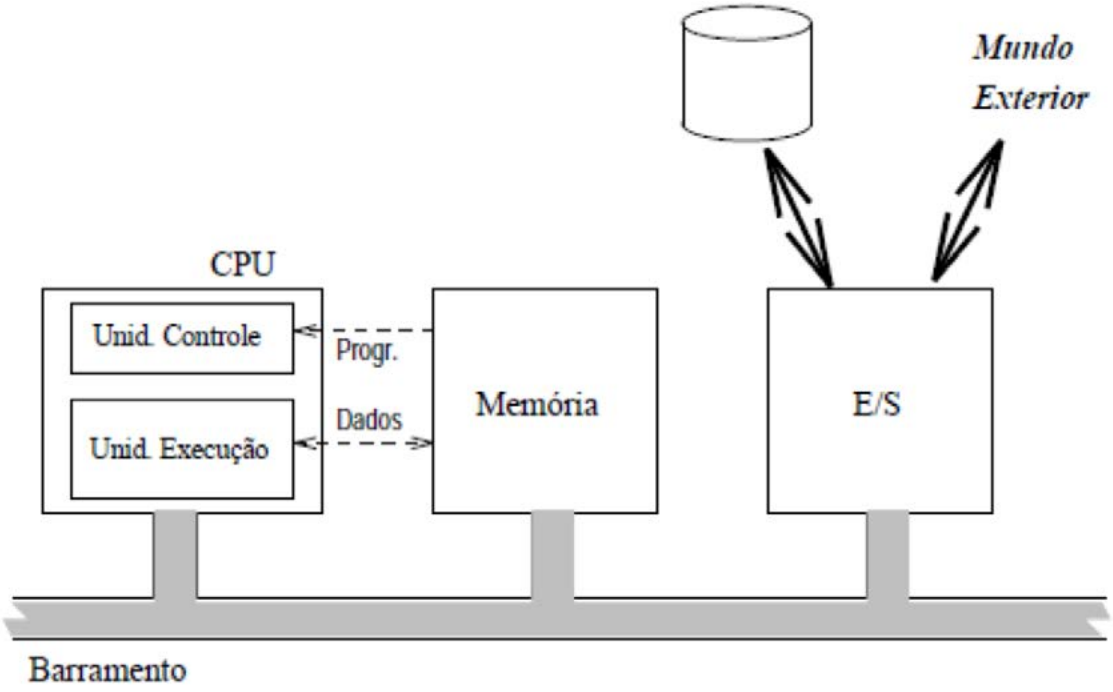
Os SBD são de extrema importância em tempos de *big data*, dado que a exigência que estes fazem sobre aqueles, no sentido de tratamento adequado para armazenamento e análise, incentivou a busca por melhores ferramentas, hardwares, softwares etc., já que falar em dados não significa mais falar em número ou algo mais convencional. Significa falar de uma gama de informações que pode ser registrada e coletada em diversos formatos.

Ricarte (1996, p. 2) afirma que "a grande motivação para manter um SBD em um sistema de processamento paralelo é manter seu desempenho em níveis razoáveis". Pode-se considerar que a afirmação do autor confirma o que foi dito no parágrafo anterior. É uma corrida

infinita por um melhor ferramental, tanto de equipamento quanto de conhecimento humano em forma de aplicativos mais eficientes e profissionais mais qualificados.

Os aspectos que têm impacto direto no desempenho de um SBD são melhores compreendidos quando se conhece a estrutura dos sistemas computacionais que armazenam o banco de dados. Uma organização considerada como clássica, utilizada desde a primeira geração de computadores, é a organização conhecida como “arquitetura de von Neumann”, a qual é apresentada de forma esquemática na Figura 2.

Figura 2 – Arquitetura de von Neumann



Fonte: Ricarte (1996).

PARA SABER MAIS

Ao pesquisador John von Neumann (1903-1957) é atribuída a proposta do armazenamento de dados e programas em uma memória comum, que se trata do conceito de programa armazenado. Era matemático nascido na Hungria,

onde trabalhou como consultor no projeto ENIAC (*Electronic Numeric Integrator and Calculator*), e de sua proposta de organização (1945) resultou a criação do computador EDVAC (*Electronic Discrete Variable Computer*).

A sequência de execução de um programa ou aplicação em máquinas esquematizadas pela arquitetura de von Neumann possui o código executável do programa (que é uma sequência de instruções para a máquina executar), assim como a disponibilidade de dados pré-definidos, tem o seu armazenamento em arquivo no disco. Para a sua execução, tais conteúdos precisam ser carregados (transferidos) do disco para a memória da CPU.

Em aplicações de bancos de dados, o processamento não é o único desafio a ser encarado. A manipulação de grandes volumes de dados também é um grande desafio, pois exige acessos contínuos a sistemas secundários de armazenamento, tais como discos magnéticos, discos ópticos, fitas magnéticas etc. Para tal acesso, há uma necessidade de transferência de dados, a qual impõe diretamente limitações de desempenho.

3. Depuração de sistemas paralelos

Segundo Ricarte (1996, p. 4), "um sistema paralelo pretende agilizar o processamento através da replicação de recursos computacionais, de forma a permitir a execução concorrente de instruções". A principal motivação para o seu surgimento, segundo o mesmo autor, foi a execução de programas científicos, com uma grande capacidade de processamento numérico.

Em se tratando de aplicações de bancos de dados, o processamento não é o único desafio a ser enfrentado. A manipulação ou tratamento de grandes bases de dados, a qual requer acessos contínuos a

sistemas secundários de armazenagem para a transferência de dados, impõe, de certa forma, limitação no seu desempenho. Os sistemas de processamento paralelo foram criados, de certa forma, para superar tais limitações.

Segundo Schnorr (2014, p. 1), “o objetivo da análise de desempenho de aplicações paralelas é a identificação de regiões do programa que tem uma baixa exploração dos recursos computacionais”. O autor afirma ainda que o sucesso da utilização de aplicações paralelas, na resolução de problemas, assim como o desempenho em termos de tempo de execução, tem dependência direta da maneira como se faz o mapeamento entre os requerimentos da aplicação e, também, dos recursos disponíveis.

É usual partir para a análise de desempenho de programas paralelos após a sua implementação, livre de qualquer possível erro, momento em que o desenvolvedor ou administrador do banco de dados está apto para realizar esta etapa do trabalho. A análise de desempenho se manifesta de forma cíclica, pois o desenvolvedor precisa realizar execuções experimentais do programa para obter informações sobre ela, para, enfim, realizar as modificações necessárias para a melhoria do desempenho.

Schnorr (2014) apresenta duas fases para a realização da análise do desempenho de uma aplicação paralela: fase de coleta e fase de análise de dados. A primeira fase ocorre ao longo da execução da aplicação e tem como objetivo o registro de informações importantes sobre o comportamento do sistema. A segunda fase tem como objetivo a identificação de problemas de desempenho durante a execução da aplicação, assim como a identificação das causas desses possíveis problemas. Dentre as técnicas utilizadas para análise dos dados de desempenho, podemos citar as técnicas de visualização de rastros e a análise estatística.

Muitas das vezes as fases da análise de desempenho de programas paralelos ocorrem de forma separada. No entanto, é muito comum realizá-las de forma paralela também, ou seja, de forma simultânea, pois pode-se coletar os dados e analisá-los concomitantemente quando se realiza o processo por longos períodos.

Por ser uma etapa de muita importância, o processo de análise de desempenho deve ser realizado com bastante rigor e auxiliado por métricas apropriadas, pois somente assim o desenvolvedor terá subsídios para afirmar para uma determinada configuração do sistema e com isso a aplicação alcançará um bom nível de desempenho.

Um conceito básico associado à análise de desempenho de aplicações paralelas é o efeito de sonda, o qual é, segundo Schnorr (2014, p. 4), "o tempo gasto no registro do comportamento da aplicação". É o impacto causado pela intrusão de sonda de monitoramento dentro da aplicação, com o intuito de registrar o seu comportamento. O impacto pode ocorrer por diversas causas, como com o uso de memória, unidade central de processamento (CPU), disco ou outro recurso computacional importante e necessário para o registro do comportamento.

Ainda sobre o efeito de sonda, é adotado o método de coleta para uma aplicação paralela cujo impacto seja o menor possível dentre aqueles apropriados para o sistema avaliado. Em geral, mede-se o impacto por medida percentual de tempo em relação ao comportamento usual da aplicação (SCHNORR, 2014). Para exemplificar, é possível obter como resultado do efeito de sonda para um determinado sistema de coleta, um impacto de 4% do tempo de execução. Com esta medida identificada, pode-se realizar comparação entre vários sistemas de coleta, para escolher aquele que apresentar o menor impacto medido.

Para uma escolha mais apropriada do sistema de coleta, Schnorr (2014) afirma que se deve levar em conta outros fatores correlacionados com o efeito sonda. É importanto considerar, também, a ocupação da memória, o tempo de processamento e uso do disco, pois podem ser fatores determinantes para alguns sistemas cujos recursos são limitados.

A análise de desempenho de sistemas paralelos pode combinar as duas fases de duas maneiras (SCHNORR, 2014). A primeira trata-se da abordagem online, a qual realiza a coleta de dados e realiza a análise de forma simultânea. Já a outra abordagem, a offline, que também é conhecida como *post-mortem*, executa as duas fases de maneira separada, sendo que a análise ocorrerá sempre "após o fim da execução da aplicação cujo comportamento será analisado" (SCHNORR, 2014, p. 5). As abordagens online e offline para análise de desempenho de aplicações paralelas também são conhecidas como análise dinâmica (STRINGHINI, 2003).

A análise online, segundo Schnorr (2014), tem como principais vantagens a ausência de custo de manutenção de dados e a interatividade na análise de dados. O autor declara que a ausência de custo de manutenção de dados tem o significado de que não há necessidade do registro de dados em disco, já que estes são analisados em tempo de execução. Já como desvantagem, ainda segundo Schnorr (2014), tem-se a falta de escalabilidade e o custo de transferência de dados. Com respeito à escalabilidade, temos que a quantidade de informação analisada em tempo de execução cresce de acordo com o tamanho das aplicações paralelas analisadas. Em tempos de *big data*, não é raro lidar com aplicações compostas por milhares de processos. "A rede de interconexão é utilizada de forma contínua para transferir os dados coletados, eventualmente centralizando-os para analisá-los" (SCHNORR, 2014, p. 5), o que pode aumentar consideravelmente o efeito de sonda. A alteração do comportamento natural da aplicação paralela também é outra desvantagem do tipo de análise online.

Sobre análise offline, Stringhini (2003, p. 28) afirma que é

> caracterizada pela gravação de informações durante a execução, tem como objetivo fornecer ao usuário uma representação esquemática da execução. Este processo pode ser dividido em duas partes: gravação de eventos durante a execução e interpretação das informações após a execução.

Tratando-se das vantagens e desvantagens da análise offline, pode-se afirmar, segundo Schnorr (2014), como desvantagem, o esforço necessário para o gerenciamento de dados, ou seja, o custo de manutenção é maior em comparação à análise online. É usual em uma análise offline de programas paralelos a não permissão de interatividade durante a fase de observação e registro, o que exigirá do desenvolvedor ou analista um planejamento apropriado dos experimentos a serem realizados para a extração de informações importantes, o que poderá ser um problema em aplicações de longa duração. Uma das principais vantagens de uma análise offline tem relação com a escalabilidade que, com a escolha de técnicas de observação e registro apropriadas, permitem o uso da abordagem offline em aplicações de larga escala.

Existem diversas técnicas de observação e registro de comportamento de aplicações paralelas para a execução da primeira fase de uma análise de desempenho. As mais conhecidas, segundo Schnorr (2014), apropriadas para uma análise offline, são: técnicas de observação (monitoramento, geração de índices estatísticos, definição de um perfil de execução e observação comportamental), técnicas de coleta e registro (amostragem, cronometragem, contagem e rastreamento).

Cada tipo de análise, online e offline, é apropriada para estudos específicos em aplicações paralelas. Por exemplo, a abordagem online coleta a observação ou registro em tempo de execução, permite a realização de análise de aplicação mais relacionada com trabalhos de depuração de erros, semelhantes à depuração de programação sequencial, do que com a análise de desempenho, de fato. Já a abordagem offline ou *post-mortem* tem como foco a análise exclusiva no desempenho, já que o seu objetivo principal é reduzir, o máximo possível, a sua intrusão no sistema. Podemos considerar estas características como justificativas para a realização de análise separadas feitas por grande parte das ferramentas paralelas.

ASSIMILE

Quando se tem diversas técnicas para aplicação em um trabalho, sempre surge o interesse em saber qual a melhor delas. Assim como em muitas situações similares, a resposta para, por exemplo, identificar a melhor técnica de coleta e registro de dados para realizar uma análise de desempenho, pode ser apresentada como: a melhor técnica é aquela que melhor atende aos objetivos que se tem, e depende diretamente do nível de conhecimento do comportamento da aplicação que se deseja analisar.

Em se tratando de técnicas para análise de desempenho, Schnorr (2014) afirma existirem várias, as quais dependem diretamente das formas de coleta de dados. Tais técnicas podem ser elaboradas pelo analista e analisadas a partir de índices estatísticos obtidos a partir de dados coletados por contadores, ou utilizarem recursos de visualização de dados, que podem ser coletados através de rastreamento.

A grande variabilidade de técnicas de análise de desempenho pode ser considerada como complemento para fornecerem uma visão global do desempenho do programa, enquanto outras poderão ser utilizadas para a exploração de pontos locais da aplicação paralela. A escolha da técnica a ser adotada também tem dependência direta das questões de desempenho que estão sendo investigadas.

É óbvio que, em muitas situações, não existirá uma única técnica apropriada para a análise de desempenho. A escolha, então, da melhor técnica, deve-se basear na técnica mais simples, mas que também possa fornecer uma visão geral do comportamento do programa. Detalhes e pormenores, caso sejam de interesse, devem ser procurados e analisados como complementação.

De forma geral, pode-se elencar os objetivos de uma análise de desempenho, segundo apresenta Schnorr (2014): (1) Melhora de desempenho da aplicação paralela e; (2) Aumento da eficiência de utilização de recursos.

Em se tratando do objetivo da melhora de desempenho, de maneira geral, da aplicação, pode-se afirmar que a mensuração pode ser feita por diversos fatores correlacionados, tais como, tempo de execução, aceleração e eficiência. O que se busca são informações capazes de serem utilizadas para melhorar os índices de desempenho.

A busca pelo aumento da eficiência de utilização de recursos computacionais é de extrema importância na execução de programação paralela. Para isso, a análise de desempenho precisa fornecer subsídios para o analista, pontos de baixa eficiência no uso dos recursos alocados, os quais podem, juntos com outros fatores, ser o motivo de baixo desempenho da aplicação.

4. Sistemas de armazenamento de dados distribuídos

Segundo Varajão (2016, p. 12), "os sistemas de informação distribuídos (SID) ficaram mais populares depois da explosão da Internet em 1993 e, desde então, estes sistemas não param de crescer". É possível perceber que o uso crescente de sistemas distribuídos ocorre tanto em ambientes acadêmicos ou de pesquisa quanto em ambientes corporativos e empresariais.

Dentre os aspectos motivadores para o crescente uso de sistemas distribuídos pode-se enumerar a facilidade no compartilhamento de uma série de componentes de redes, tais como, impressoras, arquivos eletrônicos, páginas da internet e dados distribuídos. Varajão (2016, p. 12) define sistema distribuído como "um conjunto de processos concorrentes acessando recursos distribuídos, os quais podem ser compartilhados ou replicados, através de troca de mensagens em um ambiente de rede".

> Com o avanço das técnicas de comunicação de dados, aliado ao desenvolvimento e barateamento de hardwares, foi possível migrar os sistemas de informação da tradicional forma centralizada para uma arquitetura distribuída. (ELLER, 1997, p. 1)

Um sistema de arquivos distribuídos permite às aplicações ou programas a possibilidade de armazenarem e de acessarem arquivos diversos de forma remota como se fossem arquivos locais, permitindo que os usuários do sistema possam acessar arquivos a partir de qualquer máquina acoplada em uma rede.

Uma das maneiras de armazenamento de dados, com o propósito de suprir grandes demandas, é a utilização de redes com interface de união para os usuários, os quais poderão tanto acessar quanto compartilhar dados (ALMEIDA; DUTRA, 2019, p. 173). Os autores afirmam que é um desafio o desenvolvimento de um sistema de armazenamento de dados distribuídos em ampla escala para um processamento e análise eficientes. No entanto, afirmam que um sistema computacional descentralizado, para ser utilizado, precisa apresentar três propriedades, conforme a seguir:

1. Consistência: em um sistema de armazenamento distribuído de dados, há um particionamento e uma replicação dos dados nos múltiplos nós do sistema e que cooperam entre si, no intuito de garantir disponibilidade. O sistema como um todo precisa garantir que as réplicas produzidas sejam idênticas em qualquer momento para evitar que dados desatualizados sejam lidos.

2. Disponibilidade: a justificativa para a replicação dos dados considera que há uma necessidade desta tarefa para, em situações de falhas do sistema, ele não ficar inoperante se algum nó falhar. Desta forma, as requisições do usuário poderão ser atendidas em qualquer momento, sem a necessidade de se conhecer o nó que contém os dados solicitados.

3. Tolerância à partição: sabemos que os nós de um sistema de armazenamento distribuído são conectados por uma rede. Sabemos também que a rede pode apresentar falhas em qualquer momento e até mesmo congestionamento temporário. No entanto, tais situações não deverão afetar que o sistema funcione como um todo. Por isso, a intenção é que ele funcione bem, mesmo com o particionamento da rede.

O mundo ideal é que o sistema possua as três características apresentadas. No entanto, Almeida e Dutra (2019) apresentam um teorema o qual indica que um sistema distribuído não pode garantir as três propriedades de maneira simultânea. Trata-se do teorema CAP (Consistência, Disponibilidade e Tolerância a Partição – *Consistency, Availability, Partition tolerance*). Os autores afirmam que apenas duas das três propriedades conseguem ser oferecidas de maneira simultânea. Na prática, o que querem afirmar é que um sistema classificado como CA desconsidera a tolerância à partição; já um sistema classificado como CP desconsidera a disponibilidade e, um sistema AP, desconsidera a consistência. A Figura 3 apresenta um esquema das três propriedades, onde apenas duas se interseccionam por vez.

Figura 3 – Teorema CAP e exemplos de bancos de dados com duas propriedades simultâneas

Fonte: Almeida e Dutra (2019).

Os sistemas CA, por não possuírem tolerância, não podem lidar com falha de rede. Portanto, os sistemas CA podem ser considerados como sistemas de armazenamento de servidor único centralizado, igualmente aos bancos de dados relacionais de escala pequena. Podem ser citados como exemplos deste tipo de sistema o MariaDB e o Microsoft SQL Server. Sistemas como esse (CA) apresentam uma única cópia de dados, tendo como consequência a garantia de consistência. Por possuírem essas características, "a maioria dos sistemas de armazenamento em larga escala são sistemas CP e sistemas AP" (ALMEIDA; DUTRA, 2019, p. 174).

Ainda considerando o Teorema CAP, pode-se afirmar que os sistemas CP mantêm réplicas, o que na prática significa a criação de várias cópias dos mesmos dados, cujo propósito é a garantia de nível mínimo de tolerância a falhas. A consistência é uma das propriedades deste tipo de sistema, o que indica que são produzidas cópias dos mesmos dados sempre que forem feitas solitações de consultas. O que impede a disponibilidade em sistemas CP é o alto custo para garantia da consistência. Por isso, são úteis em ambientes de grande número de solicitações e com nível rigoroso na precisão dos dados. Para exemplificar um sistema CP, podemos nos referir ao *BigTable*, o *Hbase*, o *MongoDB* e o *Redis*.

Os sistemas AP, como já visto, garantem a disponibilidade e a tolerância à partição. Portanto, sistemas como esse conseguem garantir uma consistência eventual, se diferenciando, principalmente, dos outros dois sistemas por essa característica. A consistência eventual implica que "os dados atualizados estarão consistentes em um momento eventual, podendo ser obtidos após um certo período de tempo" (ALMEIDA; DUTRA, 2019, p. 175). Dadas essas características, pode-se afirmar que os sistemas AP são perfeitamente aplicáveis aos cenários de frequentes requisições. Pode-se utilizar como exemplo as redes sociais, as quais são inseridas em contextos de intensa leitura simultânea de dados, onde, no entanto, permite-se leituras desatualizadas. Ainda exemplificando, podemos citar os bancos de dados *Dynamo*, *Voldemort*, *Cassandra*, *CouchDB* e *Riak* como sistemas AP.

O conteúdo de depuração de programas paralelos e sistemas de armazenamento de dados distribuídos é bastante amplo e não consegue se esgotar neste texto. Sugerimos que você busque por mais detalhes e informações sobre eles na literatura sugerida nas referências deste texto.

A proposta deste breve texto é de apresentar o vasto mundo dos sistemas paralelos e distribuídos de dados. Desejamos que ele tenha trazido conteúdo significativo para sua formação e que ele possa lhe trazer uma base mínima para o assunto abordado. Até breve!

TEORIA EM PRÁTICA

Suponha que você faça parte de uma equipe de tecnologia da informão (TI) de uma grande empresa, a qual possui várias filiais espalhadas por todo o Brasil. No entanto, na unidade em que você trabalha, estão centralizados os trabalhos de construção de sistemas de dados e de acompanhamento do desempenho deles. Portanto, você e seus colegas são responsáveis por esta importante atividade de trabalho, essencial para a troca de informações e dados da empresa.

Existe uma intenção de melhoria na rede de distribuição de dados para torná-la mais eficiente, segura e mais rápida na troca de informações, na consulta aos dados e na produção de relatórios de dados que possam auxiliar na tomada de decisões, tanto internas quanto externas. No entanto, para isto, é preciso realizar uma análise de desempenho do sistema atualmente instalado para buscar os pontos de deficiência e de fragilidade que, potencialmente, possam existir. Tal tarefa ajudará a convencer seus superiores a investirem na ideia de sua equipe.

Portanto, é preciso buscar as melhores técnicas de coleta e registro de dados, assim como a melhor técnica de análise de desempenho de sua rede de dados. É claro que, para escolher as técnicas apropriadas, é preciso conhecer as características fundamentais do sistema. Eis um grande desafio que vem pela frente. Vamos lá?

VERIFICAÇÃO DE LEITURA

1. Todo banco de dados para estar em ordem e em bom funcionamento precisa de cuidado e ser bem administrado. Por isso, é necessário possuir um gerenciador de banco de dados (SGBD). Na prática, o que é um SGBD?

   Assinale a alternativa CORRETA.

   a. Analista.

   b. Desenvolvedor.

   c. Software.

   d. Hardware.

   e. Equipe de TI.

2. O teorema CAP apresenta três características essenciais para sistemas distribuído de dados. No entanto sabemos na prática, que não se pode garantir todas de maneira simultânea nos sistemas existentes. Os sistemas AP garantem a disponibilidade e tolerância à partição. Qual dos sistemas a seguir pode ser considerado como um sistema AP?

Assinale a alternativa CORRETA.

a. Redis.

b. SQL Server.

c. BigTable.

d. Twitter.

e. Hbase.

3. É um conceito básico associado à análise de desempenho de aplicações paralelas. Estamos nos referindo a qual termo?

   Assinale a alternativa CORRETA.

   a. Efeito sonda.

   b. CPU.

   c. Hardware.

   d. Software.

   e. Estatística.

Referências bibliográficas

ALMEIDA, D. S.; DUTRA, D. B. Armazenamento de big data: uma abordagem didática big data storage – a didactic survey. **Perspectivas em Ciências Tecnológicas**, v. 8, n. 8, jun. 2019, p. 168-194. Disponível em: http://fatece.edu.br/arquivos/arquivos%20revistas/perspectiva/volume8/Dayse%20Silveira%20de%20Almeida;%20Danilo%20Borges%20Dutra.pdf. Acesso em: 21 out. 2019.

ELLER, N. A. **Estudo e implementação de um sistema de banco de dados distribuído**. 1997. 107 f. Dissertação (Mestrado em Ciência da Computação). Universidade Federal de Santa Catarina, Florianópolis, 1997. Disponível em: https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/158114/107003.pdf?sequence=1&isAllowed=y. Acesso em: 21 out. 2019.

RICARTE, I. L. M. **Sistemas paralelos de bancos de dados:** arquiteturas e algoritmos. Campinas: Faculdade de engenharia elétrica e de computação, 1996. Dissertação Disponível em: ftp://ftp.dca.fee.unicamp.br/pub/docs/ricarte/apostilas/spbdaa.pdf. Acesso em: 20 out. 2019.

SCHNORR, L. C. **Análise de desempenho de programas paralelos**. Porto Alegre: Instituto de Informática. Universidade Federal do Rio Grande do Sul. 2014. Disponível em: http://www.inf.ufrgs.br/~schnorr/download/talks/erad2014-minicurso-texto.pdf. Acesso em: 20 out. 2019.

STRINGHINI, D. **Depuração de programas paralelos:** projeto de uma interface intuitiva. 2003. 99 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2003. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/3639/000341170.pdf?sequence=1&isAllowed=y. Acesso em: 20 out. 2019.

VARAJÃO, F. **Sistemas distribuídos**. Apostila do curso de bacharelado em sistema de informação da FEUC, 2016. Disponível em: https://varajao.com.br/disciplinas/SD/SD%20-%20Apostila%20-%20SISTEMAS%20DISTRIBUIDOS.pdf. Acesso em: 21 out. 2019.

Gabarito

Questão 1 – Resposta C

Na prática, o sistema gerenciador de banco de dados trata-se de um software macro utilizado para gerenciar sistemas sob sua responsabilidade.

Questão 2 – Resposta D

Toda rede social, assim como o Twitter, pode ser classificada como um sistema AP de dados distribuídos.

Questão 3 – Resposta A

O efeito de sonda é um conceito básico associado à análise de desempenho de aplicações paralelas. Ele representa o tempo gasto no registro do comportamento da aplicação.

Sistemas *peer-to-peers* para processamento paralelo e distribuído

Autor: Marcelo Tavares de Lima

Objetivos

- Apresentar breve histórico dos sistemas *peer-to-peer*.
- Apresentar conceitos básicos de sistemas *peer-to-peer*.
- Apresentar as principais características de sistemas peer-to-peers para sistemas paralelos e distribuídos.

1. Introdução

Olá aluno! Neste texto serão apresentados os conceitos fundamentais de rede com enfoque para arquiteturas *peer-to-peer*. Será definido e explicado o funcionamento dessa arquitetura e, também, será realizada uma comparação com outros tipos de redes, para identificar as vantagens e desvantagens que possui.

Será apresentado um pouco da história da arquitetura *peer-to-peer* (P2P), com o intuito de contextualizar o assunto apresentado. Serão apresentados, também, os principais mecanismos de busca que possui, assim como- os sistemas de segurança de rede.

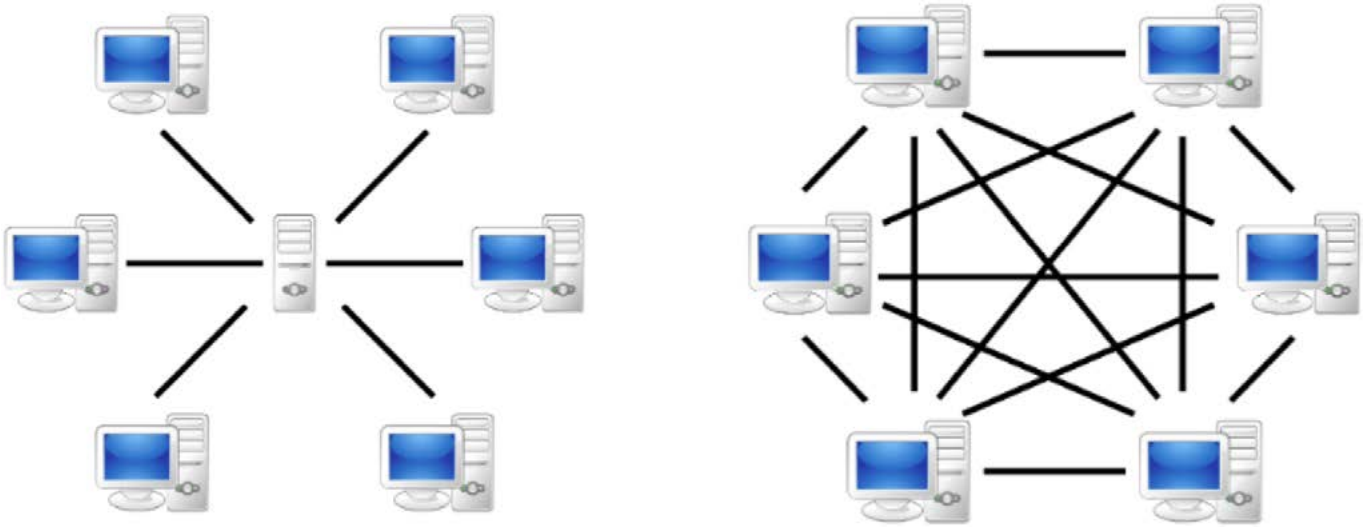
Desejamos que o conteúdo a ser desenvolvido neste texto possa trazer a você uma boa leitura, e que também possa somar no seu conhecimento, tanto como pessoa quanto como profissional da área de ciência de dados. Bons estudos!

2. Histórico e conceitos básicos dos sistemas *peer-to-peers*

Em tradução literal, *peer-to-peer* significa pares em pares. No entanto, os profissionais da área de ciência de dados utilizam o termo original em inglês ou utilizam a notação P2P para se referirem a esse tipo de arquitetura de dados.

O significado prático do termo está relacionado com uma rede de computadores interligados, no qual cada um deles “possui funções equivalentes, não havendo uma hierarquia entre eles (SCREMIN *et al*., 2007, p. 4). Essa característica torna todos os usuários do sistema em clientes e, também, servidores, fazendo com que funcionem de maneira independente e livre da necessidade de existir um servidor central como mostra a Figura 1, que faz uma comparação com um sistema baseado em um servidor.

Figura 1 – Sistemas de arquitetura cliente-servidor e P2P



Cliente-Servidor P2P

Fonte: Costa (2019).

Varajão (2016, p. 26) afirma que

> a definição do que é uma rede do tipo *peer-to-peer* (P2P), pode parecer obscura em um primeiro momento. Contudo, é clara a distinção entre o paradigma cliente-servidor e o paradigma P2P. No primeiro existe uma clara distinção entre a entidade que está provendo um serviço e o cliente que o está consumindo. Por outro lado, não existe distinção entre os elementos de uma rede P2P, todos os nós possuem funcionalidades equivalentes.

A comunicação e a troca de dados entre os diversos nós de uma rede P2P é feita diretamente entre eles, não havendo um nó intermediador para a realização da troca de informações, conforme dito anteriormente. A ausência de um nó intermediador ou servidor traz a característica de servidor e cliente, simultâneamente, para cada nó pertencente à rede P2P.

Em uma rede P2P, cada nó participante disponibiliza parte de seus recursos, enquanto que, ao mesmo tempo, utiliza recursos disponibilizados por outros nós participantes. Essa característica é contrastante com o modelo mais usual de arquitetura de redes, como por exemplo, a internet, que tem arquitetura de cliente-servidor, na qual cada elemento tem seu papel muito bem definido. O servidor ou servidores fornecem recursos para o consumo dos clientes.

Dentre os aspectos técnicos de sistemas P2P, pode-se afirmar que são sistemas distribuídos sem controle centralizado. Isso significa que o programa que estiver sendo executado em cada nó torna-se equivalente em funcionalidade. Por conta desse aspecto, há uma dependência da participação "voluntária dos pares, a fim de contribuir com recursos, os quais a infraestrutura é construída" (VARAJÃO, 2016, p. 26). Outros aspectos técnicos importantes de um sistema P2P referem-se a sua capacidade de auto-organização, adaptabilidade e escalabilidade (VARAJÃO, 2016).

Varajão (2016) declara que apesar da definição exata de redes P2P ser discutível, pode-se afirmar, com certeza, que não possuem infraestrutura centralizada. Scremin *et al*. (2007, p. 5) afirmam que a "vantagem de uma arquitetura de rede descentralizada é que ela é bem mais difícil de ser interrompida, pois não existe um ponto de falha".

Scremin *et al*. (2007) declaram que a arquitetura P2P sempre existiu, mas não era reconhecida como tal. Para afirmar o que disseram, apresentaram como exemplo a possibilidade de comunicação entre "servidores com endereço de IP fixos ou determináveis" (SCREMIN *et al*., 2007, p. 6), os quais tinham a capacidade de comunicação com outros servidores. Esse período prévio à oficialização da existência de redes P2P é conhecido como pré-P2P. Os autores exemplificam, ainda, com o uso de correios eletrônicos, estes com capacidade de oferecer serviço em rede distribuída.

No período pré-P2P, segundo Scremin *et al*. (2007), um aplicativo ganhou destaque, a USENET, que surgiu, segundo os autores, por volta de 1979, desenvolvido por estudantes universitários dos Estados Unidos. A USENET "permitia que dois computadores trocassem informação, nos primórdios, antes da conexão oferecida pela internet" (SCREMIN *et al*., 2007, p. 6).

O funcionamento da USENET, segundo Scremin *et al.* (2007, p. 6), "consistia na capacidade de um computador se ligar a outro, via modem, pesquisar documentos e transferir esses documentos para armazenamento local". Não existia autoridade centralizadora, portanto, a distribuição de documentos era gerada por cada usuário da rede e, o conteúdo era replicado para todos.

Outra aplicação que pode ser citada como exemplo de rede P2P é o ICQ. Scremin *et al.* (2007) afirmam que com seu surgimento, o correio eletrônico tradicional foi substituído, dadas as características de comunicação do ICQ, feita de maneira mais rápida, permitindo que seus usuários fossem notificados quando seus "amigos" se conectavam à rede. Assim, como a USENET, o ICQ também permitia a troca de arquivos. Apesar de ter sido classificada como uma rede P2P, o ICQ, de fato, era de arquitetura mista entre P2P e cliente/servidor, já que o serviço de notificação de novos usuários era centralizado em um servidor.

A estrutura P2P se popularizou por meio dos aplicativos de compartilhamento de arquivos, como o Napster (VARAJÃO, 2016). O aplicativo foi desenvolvido para compartilhamento de arquivos, principalmente de música e, por isso, a empresa criadora foi processada pela indústria fonográfica, as quais detinham direitos autorais das músicas que estavam sendo compartilhadas em arquivo eletrônico sem a devida permissão (VARAJÃO, 2016).

Para utilizar o Napster, o usuário precisava criar um registro no sistema, o qual identificava os arquivos que estavam sendo disponibilizados. Com a disponibilização, qualquer outro usuário poderia consultar o servidor, ou *cluster* de servidores, para buscar e para transferir arquivos (VARAJÃO, 2016).

Após o encerramento das operações do Napster, usuários desse sistema passaram a utilizar outro sistema, o Gnutella (VARAJÃO, 2016). Diferente do Napster, o Gnutella não tinha uma entidade central para realizar busca de arquivos, o que possibilitava operação direta entre seus usuários e, com isso, o Gnutella conseguiu com que a indústria da música não pudesse impedir o seu funcionamento (VARAJÃO, 2016).

Com a evolução de arquiteturas P2P, Varajão (2016, p. 27) afirma que

> com o aumento da capacidade dos computadores e dos links de Internet domésticos, outros usos de rede P2P existem, por exemplo, para projetos de computação distribuída, para distribuição de fluxos de mídia em tempo real (*streaming*), telecomunicações (telefonia, videoconferência...) e outros.

Um exemplo de rede descentralizada bastante noticiada nos últimos anos é o sistema econômico alternativo Bitcoin (BTC ou XBT), o qual, segundo define Melo (2019, p. 30) "é uma moeda digital do tipo criptomoeda descentralizada e, também, um sistema econômico alternativo (*peer-to-peer electronic cash system*)". O autor ainda afirma que "o Bitcoin permite transações financeiras sem intermediários, mas verificadas por todos os usuários da rede (nós da rede) Bitcoin, que são gravadas em um banco de dados distribuídos, chamado de blockchain" (MELO, 2019, p. 30).

Outros sistemas de rede P2P bastante conhecidos e utilizados são o eMule e o BitTorrent, os quais usam um algoritmo DHT (*distributed Hash Table*), ou tabela *hash* distribuída, em paralelo a suas redes não-estruturadas – detalhadas a seguir – originais com o intuito de descobrir pares para aumentar a eficiência da rede.

PARA SABER MAIS

Melo (2019) afirma que a tecnologia *blockchain* ("cadeia de blocos", em português) pode ser considerada como um banco de dados distribuídos com a função de livro-razão de contabilidade pública, onde são registradas todas as transações realizadas com bitcoin.

Segundo Varajão (2016), as redes P2P podem ser classificadas de duas maneiras: puras e híbridas. São consideradas puras as redes P2P, segundo o autor, que não carregam o conceito de cliente ou de servidor,

pois não existem nós de infraestrutura, pois cada par de nós pode funcionar como cliente e servidor sem distinção. Já as redes híbridas permitem a existência de nós especiais, também conhecidos como supernós (*supernodes*), os quais possuem mais recursos (em especial, banda) disponíveis.

As redes P2P também podem ser classificadas quanto ao controle de conexões e de fluxo de informações. Segundo Varajão (2016), uma rede P2P pode ser de controle distribuído ou centralizado. No controle de conexões e fluxo de informações distribuído as informações circulam entre os pares da rede, já no centralizado, existe um nó central que é responsável por indexar o conteúdo e pelo gerenciamento do processo de troca de dados que, apesar disso, são feitas diretamente entre os pares sem serem regidos por algum algoritmo particular.

Com relação aos enlaces existentes entre os nós, Varajão afirma que é possível classificar sistemas P2P em estruturados e não estruturados. Sistemas estruturados "são aqueles em que existe um protocolo que garante que cada nó possa rotear de forma eficiente uma busca por qualquer dado, mesmo que ele seja muito raro" (VARAJÃO, 2016, p. 27). O autor afirma ainda que grande parte das redes estruturadas utiliza tabelas *hash* distribuídas.

Já nos sistemas P2P não estruturados, "a distribuição dos nós é feita de forma arbitrária" (VARAJÃO, 2016, p. 27). São consideradas de elaboração mais simples, já que um nó que deseja participar da rede pode começar apenas copiando links de outros participantes até estabelecer suas próprias conexões. No entanto, na comparação com redes estruturadas, pode-se afirmar como desvantagem de uma rede não estruturada que mensagens de busca de conteúdo inundam a rede com o propósito de encontrar pares que estejam compartilhando dados de interesse, ocasionando congestionamento de tráfego de controle, além do que, buscas, também, podem não ser resolvidas, mesmo existindo nós compartilhando dados requisitados, com possibilidade de ocorrência dessa situação, crescente à medida que os dados sejam mais raros.

Oliveira e Rocha (2003 *apud* VARAJÃO, 2016) apresentam dois tipos de classificação para arquiteturas P2P, as quais são apresentadas com maiores detalhes a seguir.

2.1 Primeira classificação

A primeira classificação para arquiteturas P2P trazida por Oliveira e Rocha (2003 *apud* VARAJÃO, 2016) divide as redes P2P em centralizada, descentralizada e hierárquica. Os detalhes de cada classificação são descritos a seguir.

2.1.1 Modelo centralizado

Neste tipo de modelo, segundo Varajão (2016), existe uma unidade central ou servidor, ou ainda, um conjunto de servidores que contém identificadores de todos os participantes da rede, o que caracteriza um modelo de relação cliente/servidor entre os nós da rede com o servidor P2P. Um exemplo deste tipo de modelo é o aplicativo Napster, apresentado anteriormente.

2.1.2 Modelo descentralizado

Neste tipo de modelo não existe servidor ou servidores, o que caracteriza a rede como completamente distribuída (VARAJÃO, 2016). Exemplo desse tipo de arquitetura são os aplicativos Gnutella e Freenet.

2.1.3 Modelo hierárquico

Caracteriza-se por ser uma mistura dos dois modelos apresentados anteriormente. Por isso, em redes hierárquicas existe o que se conhece por supernó (*supernode* ou *superpeer*). São exemplos desse tipo de arquitetura o Kazaa e Morpheus.

2.2 Segunda classificação

A segunda classificação de redes P2P apresentada por Oliveira e Rocha (2003 *apud* VARAJÃO, 2016) é detalhada a seguir.

2.2.1 CIA (*Centralized Indexing Architecture*)

Redes P2P com arquitetura CIA possuem servidor central ou cluster de servidores, responsáveis por coordenar pedidos de busca, assim como executar tarefas de manutenção de infraestrutura. Não são consideradas verdadeiras redes P2P por possuírem ponto de falha, pois, se o ponto central ficar inoperante, a rede toda não funciona. Apenas a transferência de arquivos é feita de forma distribuída. Pode ser considerado um exemplo deste tipo de rede o Napster.

2.2.2 DIFA (*Distributed Indexing with Flooding Architecture*)

Caracterizada pela completa descentralização de seu funcionamento. Mecanismos de busca e de manutenção da sua infraestrutura estão totalmente distribuídos pela rede. Cada nó participante da rede é responsável por manter sua listagem de arquivos e, também, por responder quando uma busca por arquivo for recebida. Para tanto, é utilizado o mecanismo de *flooding* ou de inundação. Como desvantagem, pode-se citar a existência de limitação ligada ao mecanismo de busca quanto ao uso de *flooding*. A busca por um arquivo, mesmo existente no sistema, pode falhar mesmo se estiver online, apresentando deficiência na performance da rede. O Gnutella é um exemplo deste tipo de modelo.

2.2.3 DIHA (*Distributed Indexing with Hashing Architecture*)

Assim como a arquitetura DIFA, é considerada uma rede totalmente descentralizada. O que a difere da DIFA trata-se do mecanismo de busca. Em sistemas com *flooding* cada par ou *peer* responde pelo espaço de índices relacionados aos arquivos que contêm ou administra.
Na arquitetura DIHA isso muda um pouco, cada nó pertencente à rede responde por um subconjunto do espaço total dos índices do sistema. Se um nó deixa de participar da rede, os arquivos sob sua responsabilidade são transferidos para outro nó. As buscas, diferentemente dos sistemas *flooding*, não são difundidas sem direção, mas são direcionadas para o nó correto. Os protocolos que implementam este tipo de arquitetura são mais complexos, por isso, são pouco utilizados (VARAJÃO, 2016).

2.3 Segurança de rede P2P

Sobre segurança de uma rede P2P, Scremin *et al*. (2007, p. 8) afirmam que

> a segurança é um componente essencial para qualquer sistema de computação e é essencialmente relevante para sistemas P2P. Navegar pelas redes P2P pode não ser muito seguro, pois existem várias ameaças dentro da rede, como vírus que vem com arquivos e outros. O P2P compartilha muitos problemas de segurança e soluções com o resto da rede e sistemas distribuídos.

Muitos mecanismos de segurança vêm sendo elaborados para tornar as redes P2P mais seguras e confiáveis. No entanto, ainda existem muitas ameaças que precisam ser combatidas para tornar essa arquitetura mais segura.

Uma lista de principais ameaças existentes para usuários de redes P2P é apresentada por Scremin *et al*. (2007) e replicada a seguir.

- Como já falado, a contaminação por vírus, além de *worms* e *malware* por meio da troca de arquivos.
- Invasão por conta de vulnerabilidade em aplicações com arquiteturas P2P.
- Publicação de informações sensíveis, de maneira acidental, por intermédio de compartilhamento de arquivos.
- Processos judiciais decorrentes de violação de direitos autorais por obtenção de software, filmes, músicas, livros etc.
- Furto de arquivos que podem causar prejuízo financeiro para pessoas ou empresas.
- *Bugs* para funcionamento de compartilhamento de arquivo.
- Quebra de criptografia e falta de confidencialidade.

Os autores afirmam ainda que “todos os sistemas de segurança atuais são baseados em criptografia que utiliza tanto chave simétrica e privada quanto chave assimétrica e pública ou, às vezes, uma combinação das duas” (SCREMIN *et al*., 2007, p. 9).

Dada a atenção que tem sido dispensada cada vez maior a redes P2P, os mecanismos de segurança têm sido cada vez mais aprimorados, como por exemplo, com o uso de biometria e de criptografia por chave quântica (SCREMIN *et al*., 2007).

ASSIMILE

Segundo Scremin *et al*. (2007, p. 12), o recurso de biometria para segurança de redes P2P “envolve o uso de uma característica pessoal para autenticar o usuário”.
As características utilizadas, em geral, são reconhecimento facial, uma assinatura, impressão digital ou padrão de retina.

3. Aplicações de rede P2P

Além das aplicações já descritas em seções anteriores deste texto, serão apresentadas algumas aplicações com base no texto de Pires (2007), tais como, compartilhamento de arquivo, troca de mensagens, sistemas distribuídos e trabalhos realizados em colaboração.

Diversos exemplos sobre compartilhamento de arquivos já foram apresentados, inclusive, já foram citados aplicativos que foram elaborados para esse propósito, como Napster, Gnutella, entre outros. No entanto, para reafirmar a aplicação de sistemas P2P para tal finalidade, Pires (2007) afirma que o uso de sistemas distribuídos para a finalidade de compartilhamento de arquivos tem alcançado o maior sucesso.

Aplicações de compartilhamento de arquivos por meio de redes P2P também armazenam índices, que são “estruturas utilizadas durante o processamento de consultas para mapear arquivos em ponto” (PIRES, 2007, p. 33). Esses índices são mantidos sob o formato de metadados e são definidos pelo sistema em uso. O conteúdo dos metadados criados são o nome e o tamanho dos arquivos, o total de upload/downloads, a largura da banda, dentre outras informações importantes para o compartilhamento.

É preciso ter um certo cuidado com os arquivos que circulam por sistemas distribuídos P2P. Há uma questão ética que envolve a permissão do compartilhamento de arquivos, o que, segundo Scremin *et al*. (2007, p. 13), “por si só, não constitui uma violação dos direitos autorais, desde que os arquivos compartilhados não estejam protegidos”.

As pesquisas realizadas em sistemas P2P para compartilhamento de arquivos são por meio de palavras-chave e, como dito anteriormente, aplicativos elaborados para tal finalidade são o Napster, Gnutella, Kazaa etc. As desvantagens de redes P2P para compartilhamento de arquivos, segundo Pires (2007, p. 34), em termos técnicos, são “segurança, consumo de largura de banda e capacidade de busca”.

Os mecanismos de busca, segundo afirmam Scremin *et al*. (2007, p. 14), “surgiram logo após o aparecimento da internet, com a intenção de prestar um serviço extremamente importante: a busca de qualquer informação na web, apresentando os resultados de uma forma organizada”.

A capacidade de troca de mensagens é outro tipo de aplicação de arquiteturas P2P. São considerados, principalmente, aplicativos de troca instantânea de mensagens como ICQ, MSN Messenger, Gtalk da Google, Yahoo! Messenger, dentre outros. Se popularizaram porque permitiram a entrega de mensagens em tempo real!

Outra aplicabilidade bastante importante de rede P2P trata-se do processamento distribuído, o qual, segundo Pires (2007, p. 35), "vislumbra a possibilidade de agregar e utilizar a capacidade de processamento e armazenamento que fica subutilizada em máquinas ociosas".

Scremin *et al*. (2007, p. 16) declaram que "a ideia de índices distribuídos é que cada nó da rede contém um índice de arquivos locais assim como índices de arquivos armazenados de alguns pares vizinhos". Os autores afirmam que sempre que um nó recebe uma consulta, a primeira verificação é a possibilidade da consulta na localidade pesquisada. Caso não seja possível, o nó utiliza o índice local para determinar o nó apropriado para destinar a consulta. O índice local de cada máquina do sistema não é estático (SCREMIN *et al*., 2007), muda de acordo com o movimento dos arquivos da rede.

A última aplicabilidade apresentado por Pires (2007) trata de sistemas de trabalho colaborativo, os quais, segundo o autor, "são projetados para melhorar a produtividade de indivíduos que possuem metas e interesses comuns" (PIRES, 2007, p. 36). São, também, conhecidos como *groupware*, cuja definição apresentada é de que se trata de "*software* que suporta colaboração, comunicação e coordenação de um grupo de usuários em uma rede" (PIRES, 2007, p. 36). São exemplos desse tipo de sistema as ferramentas de calendário, listas de discussão, correio eletrônico, as videoconferências, entre outras.

Com respeito a plataformas de desenvolvimento de sistemas P2P, em princípio, havia a necessidade de implementação de protocolo próprio de comunicação para que houvesse permissão da interoperabilidade entre os pontos de rede (PIRES, 2007, p. 36).

As várias formas de implementação de plataformas P2P, muitas vezes, impediam a comunicação entre as aplicações. Tal fato ocorria até mesmo entre aplicativos que ofereciam serviços equivalentes, como os

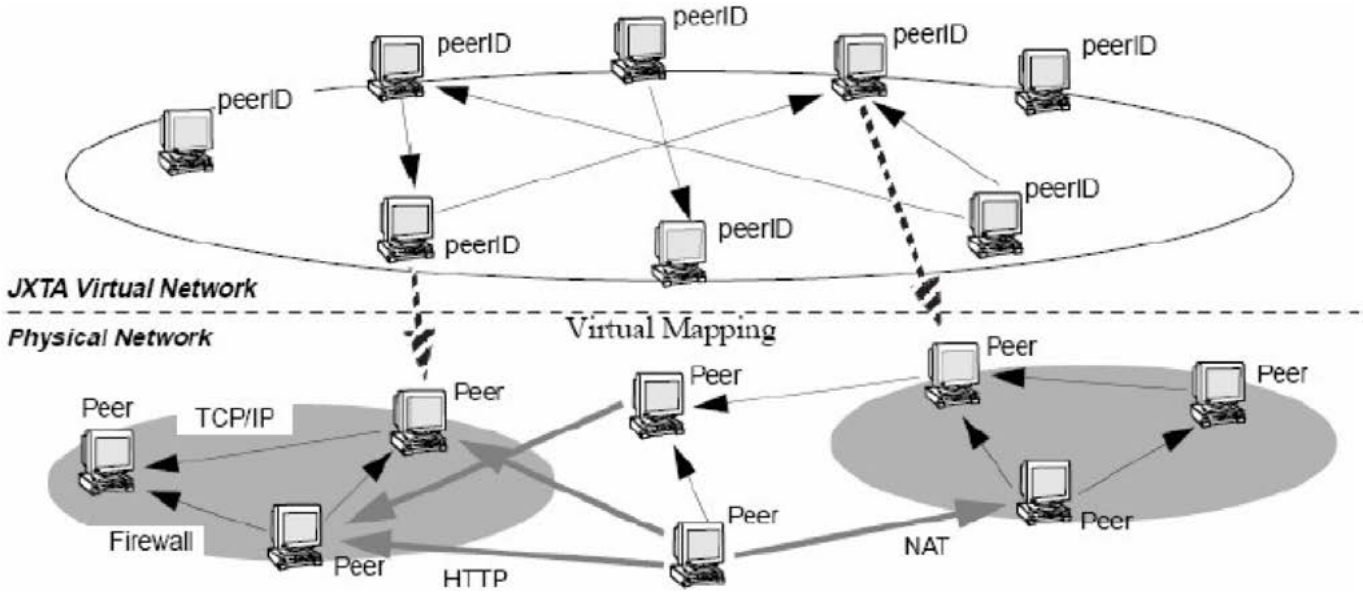
de compartilhamento de arquivos, como o Napster, Gnutella e Kazaa. Esses dois últimos, juntamente com o e-Mule, segundo Pires (2007, p. 36), “são acessíveis exclusivamente dentro de suas próprias redes, forçando os usuários a manterem instalados, em suas máquinas, clientes para cada um dos sistemas”.

Ao longo da popularização de aplicações P2P, houve um esforço na construção de plataformas para desenvolvimento de aplicações que pudessem se comunicar. Uma dessas plataformas que se destaca é a JXTA. A plataforma JXTA, segundo Pires (2007, p. 37), “é uma plataforma aberta, desenvolvida inicialmente pela Sun Microsystems, para construção de sistemas P2P”.

Pires (2007, p. 37) afirma que a plataforma JXTA “disponibiliza um conjunto de protocolos para comunicação, colaboração e compartilhamento de recursos, entre diferentes tipos de dispositivos”. O autor explicita o que chamou de diferentes dispositivos como aparelhos de celular, estações de trabalho, PDA (*personal digital assistants*) e, também, servidores.

A criação da plataforma JXTA teve como objetivos principais a interoperabilidade (operação entre diferentes sistemas P2P), a independência de plataforma (diversidade de linguagens de programação, de sistemas operacionais e, também, de redes), a generalidade (aplicável a qualquer tipo de dispositivo digital) e, também, a segurança. A Figura 2 apresenta uma ilustração de uma rede virtual JXTA.

Figura 2 – Rede virtual JXTA



Fonte: Pires (2007).

Pires (2007, p. 37) declara que

> Em uma rede virtual JXTA, qualquer ponto pode interagir com os demais, independente de sua localização, tipo de serviço ou ambiente operacional – mesmo quando alguns pontos e recursos estejam localizados atrás de *firewalls* ou utilizem diferentes tecnologias de transporte de rede.

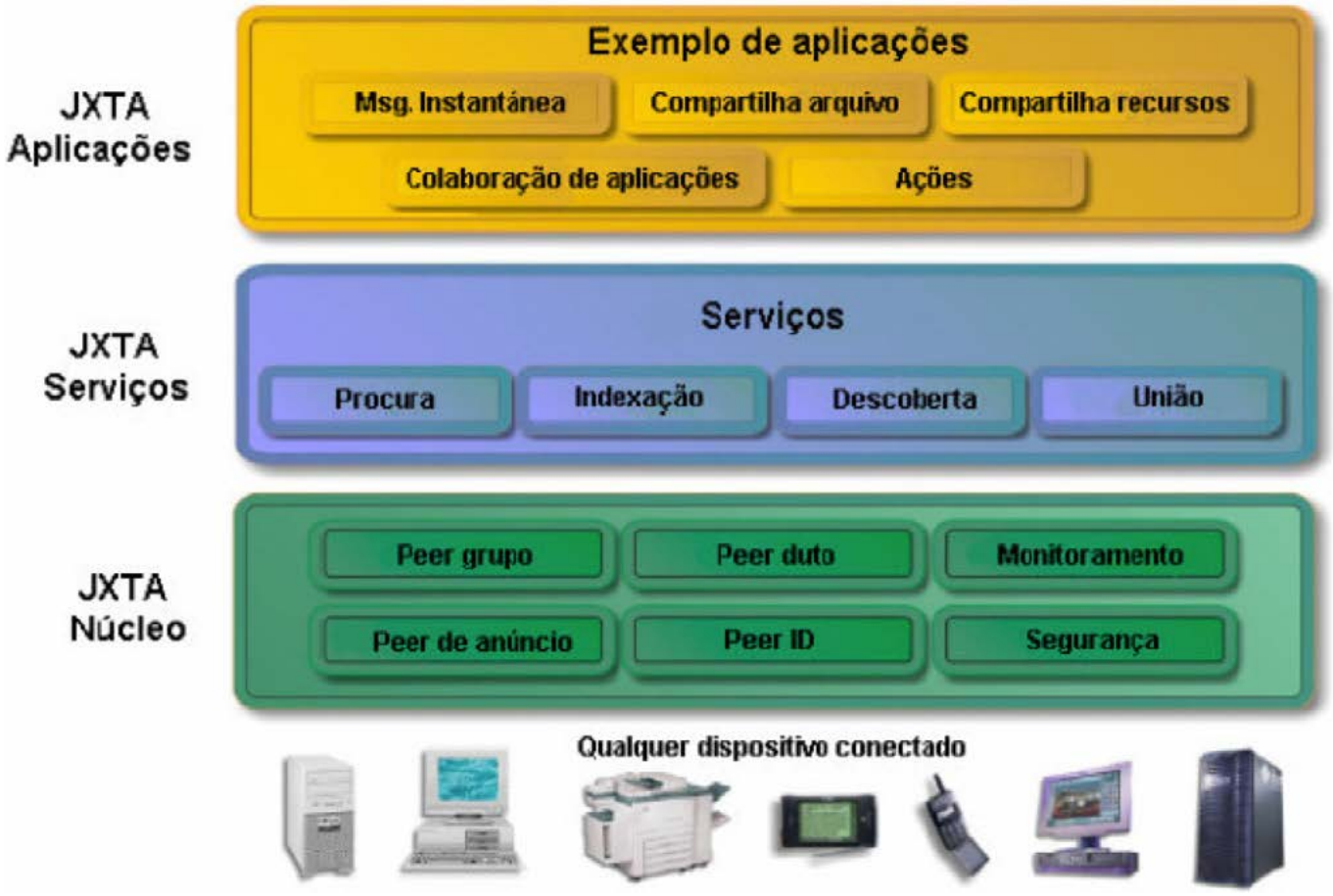
Uma importante característica da plataforma JXTA se refere a sua posição, ou seja, ela fica posicionada em pilha P2P, que é uma camada localizada acima do sistema operacional ou da máquina virtual e, também, localizada abaixo de aplicações e serviços P2P (PIRES, 2007). Conforme afirma o autor, "ela pode ser descrita simplesmente como uma tecnologia que permite a comunicação entre pontos" (PIRES, 2007, p. 37).

Os pontos da plataforma JXTA estão associados, um a um, a um único identificador, chamado de *peer ID*, assim como pertence a um ou mais grupos denominados *peergroups* (PIRES, 2007). Dentro de cada *peergroup* os pontos têm funções semelhantes abaixo de um conjunto munido de capacidades e de restrições.

Os protocolos disponibilizados por uma plataforma JXTA operam com funções básicas como a criação e o encontro de grupos, permissão para entrar e para sair de grupos, permissão para o monitoramento de grupos, capacidade de conversar com outros grupos e outros pontos e o compartilhamento de conteúdos e serviços.

Pires (2007, p. 38) afirma que "conceitualmente, cada ponto na plataforma JXTA abstrai três camadas: o núcleo, a camada de serviços e a camada de aplicação". O autor afirma ainda que "o núcleo é responsável por gerenciar o protocolo JXTA. Sua função é encapsular o conhecimento das operações P2P básicas". Isso quer dizer que o núcleo tem a responsabilidade de conter as funcionalidades, assim como a infraestrutura mínima necessária para que aplicações P2P possam ser desenvolvidas. A Figura 3 apresenta uma estrutura da plataforma JXTA.

Figura 3 – Arquitetura em camadas da plataforma JXTA



Fonte: Pires (2007).

Aplicações que utilizam arquiteturas P2P são diversas e cada vez mais ampliadas para uso em sistemas distribuídos. Existe uma vasta literatura que trata do assunto, aplicado tanto em ambientes acadêmicos quanto corporativos.

A proposta deste foi apresentar para o aluno conceitos básicos e introdutórios sobre redes P2P com a intenção de trazer uma reflexão sobre o assunto, assim como apresentar a sua aplicabilidade nos diversos ambientes. Com isso, espera-se que surja interesse no aprofundamento do assunto e que isso reflita em sua formação profissional de maneira satisfatória. Até breve!

TEORIA EM PRÁTICA

Na empresa que você trabalha está sendo implantado um projeto de planejamento coletivo de trabalho, ou seja, estão sendo implementadas boas práticas de informação e distribuição de atividades de conhecimento geral. A ideia é fazer com que todos saibam o trabalho desenvolvido por cada equipe, as etapas que estão sendo executadas e por quem estão, assim como os prazos determinados inicialmente e cumpridos.

A equipe de tecnologia da informação (TI) a qual você integra ficou responsável por criar um aplicativo de comum uso entre os colaboradores da empresa com o propósito de disseminação das informações acima especificadas. Portanto, a ideia é criar uma espécie de calendário de atividades no qual cada colaborador possa alimentar o sistema com informações atualizadas sobre as atividades cumpridas e a serem cumpridas, assim como o prazo estabelecido para cada uma.

Para isso, sua equipe pensou em criar uma rede interna P2P para a implementação dessa boa prática de trabalho. Você está, também, responsável por ajudar na criação do aplicativo. Portanto, o que você consegue apresentar de sugestão? Alguma ideia para a interface do aplicativo? Quais as ferramentas que ele poderá ter? Há alguma outra utilidade que possa ser implementada nesse aplicativo? Pense sobre isso e, nos momentos de encontro com sua equipe, apresente suas ideias.

VERIFICAÇÃO DE LEITURA

1. Uma rede de computadores interligados segundo uma arquitetura *peer-to-peer* tem atribuído a cada um dos participantes (nó) funções equivalentes, portanto, não havendo uma hierarquia entres eles. Qual o significado da tradução literal para o português do termo *peer-to-peer*?

   Assinale a alternativa CORRETA.

   a. Ponto a ponto.

   b. Par em par.

   c. Rede em rede.

   d. Núcleo a núcleo.

   e. Cliente a servidor.

2. Uma arquitetura *peer-to-peer* possui uma série de características técnicas que a distinguem de outros sistemas. No entanto, segundo Varajão (2016), existem três características técnicas importantes. Selecione a alternativa que contém uma das três características técnicas citadas pelo autor.

   Assinale a alternativa CORRETA.

   a. Velocidade.

   b. Desorganização.

   c. Escalabilidade.

   d. Imprecisão.

   e. Centralidade.

3. O Napster foi uma das aplicações que ajudou na disseminação de redes P2P para compartilhamento de arquivos. No entanto, ele foi encerrado por compartilhar arquivos com direitos autorais musicais. Apesar disso, usuários de rede P2P passaram a usar outro aplicativo que substitui o Napster. Qual o nome deste aplicativo?

   Assinale a alternativa CORRETA.

   a. ICQ.

   b. Gtalk.

   c. MSN Messenger.

   d. Gnutella.

   e. USENET.

Referências bibliográficas

COSTA, L. H. M. **Redes par-a-par (P2P)**. Rio de Janeiro: Universidade Federal do Rio de Janeiro. Notas de aula. 2019. Disponível em: https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/p2p/. Acesso em: 30 out. 2019.

MELO, F. **Informática para concursos:** internet e redes. [s.l.]. 2019. Disponível em: https://www.grancursosonline.com.br/download-demonstrativo/download-aula-pdf-demo/codigo/gWodXP9P9EM%3D. Acesso em: 30 out. 2019.

PIRES, C. E. S. **Um sistema P2P de gerenciamento de dados com conectividade baseada em semântica**. 2007. 123 f. Exame de Qualificação e proposta de tese (Doutorado em Ciências da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2007. Disponível em: https://www.cin.ufpe.br/~speed/papers/Proposta_CESP_Banca.pdf. Acesso em: 30 out. 2019.

SCREMIN, A. M.; HERRERA, B. S.; PAIXÃO, B. C.; TAMAKI, D. **Sistemas de redes *peer to peer***. 2007. [s.l.]. Instituto de Computação: Universidade Federal Fluminense. Notas de aula. Disponível em: http://www.ic.uff.br/~otton/graduacao/informatica1/P2P.pdf. Acesso em: 30 out. 2019.

VARAJÃO, F. **Sistemas distribuídos**. Apostila do curso de bacharelado em sistema de informação da FEUC, 2016. Disponível em: https://varajao.com.br/disciplinas/SD/SD%20-%20Apostila%20-%20SISTEMAS%20DISTRIBUIDOS.pdf. Acesso em: 21 out. 2019.

Gabarito

Questão 1 – Resposta B

A tradução literal do termo *peer-to-peer* para o português é par em par ou pares em pares, que se refere à arquitetura de sistemas computacionais.

Questão 2 – Resposta C

Varajão (2016) cita três características técnicas importantes de uma arquitetura P2P, que são: organização, adaptabilidade e escalabilidade.

Questão 3 – Resposta D

Com o encerramento do Napster, usuários de redes P2P para compartilhamento de arquivos passaram a utilizar o Gnutella.

Construindo um *data warehouse* e ETL's

Autor: Marcelo Tavares de Lima

Objetivos

- Apresentar conceitos fundamentais de *data warehouse*.
- Descrever sobre a construção de um *data warehouse* a partir de indicadores.
- Descrever como construir ETL's para uso com *data warehouse*.

1. Introdução

Olá aluno! Seja bem-vindo a mais uma leitura de seu curso. Nesta, serão apresentados conceitos fundamentais de *data warehouse* e conceitos associados. Também serão apresentados exemplos de aplicação, assim como serão descritos métodos de elaboração de um *data warehouse*.

Pode ser que para você sejam conceitos novos, mas saiba que são conceitos construídos na década de 1990. É claro que vão sendo atualizados na medida em que as tecnologias vão se modernizando, também. Por isso, muitos conceitos quando surgiram pela primeira vez podem ter sido modificados. Aproveite esta leitura! Bons estudos!

2. Conceitos fundamentais de *data warehouse*

A informação é moeda de muito valor no mundo dos negócios, onde, inclusive, é uma das causas das vantagens em um mundo competitivo. Por isso, ao longo dos anos, com a tendência crescente do mercado e com a globalização, "a computação se tornou fundamental para processar grandes volumes de dados" (NOVAIS, 2012, p. 12).

Com o passar dos anos, as empresas passaram seus sistemas de informações da área operacional para áreas mais analíticas, com a intenção de realizar a tomada de decisão baseada em dados. Foi assim que detectou-se o grande gargalo a ser superado, relacionado os arcaicos meios de armazenamento de dados, dificultando a obtenção de informações a partir de grandes volumes de dados.

Dentre os muitos conceitos relacionados a um *data warehouse*, existe o conceito de *Business Intelligence* (BI), o qual, segundo Novais (2012, p. 13), "é um conjunto de conceitos e metodologias que, fazendo uso de acontecimentos (fatos) e sistemas baseados nos mesmos, apoia a tomada de decisões em negócios". Este conceito destaca a

existência de um grande desafio para os gestores quanto à análise dos acontecimentos (fatos) relacionados ao seu dever de detectar quaisquer informações úteis para alavancar os negócios e ganhar vantagem em relação à concorrência.

Novais (2012, p. 13) afirma que "o propósito do *Business Intelligence* é permitir a tomada de decisões proativas, ao gerar informações necessárias ao negócio e disponibilizá-los no momento certo". Para achar o momento certo é necessário estar muito bem aparelhado com ferramentas de sistemas de gestão de dados que possam extrair informação relevantes de grandes bases de dados, assim como realizar tal ação no momento apropriado sem perder a qualidade e a confiabilidade.

Ao longo dos anos percebe-se que há um aumento significativo na elaboração de sistemas de gestão, tendo como consequência o crescimento da produção de dados e de seu armazenamento e tratamento, também (GURA; BENCK, 2011). Diante deste cenário, surgiu o conceito de *data warehouse*, que também é conhecido como armazém de dados.

Apesar da tradução ser "armazém de dados", o conceito de *data warehouse* vai além disso, conforme afirma Machado (2004, p. 22 *apud* Gura e Benck, 2011, p. 21) que

> *data warehouse* é uma arquitetura, não uma tecnologia. Representa uma grande base de dados capaz de integrar, de forma concisa e confiável, as informações de interesse para a empresa, que se encontram espalhadas pelos sistemas operacionais e em fontes externas, para posterior utilização nos sistemas de apoio à decisão.

A tarefa de armazenamento de dados não é trivial, pois ela exige um rigor na sua organização e na seleção do que será guardado, já que nem todo dado necessita de armazenamento.

As tecnologias associadas ao conceito de BI têm sido utilizadas pelas grandes corporações já há um certo tempo. O seu uso, até então, era restrito a esse tipo de corporação por conta da necessidade de altos investimentos, o que acabou restringido o acesso de corporações de porte mediano e pequeno (NOVAIS, 2012). No entanto, esse cenário tem se modificado, pois, segundo o autor, "uma nova geração de produtos de BI tem surgido: alta performance, operações intuitivas e rápida implantação são características das novas soluções que atendem a empresas de todos os tamanhos" (NOVAIS, 2012, p. 15).

Apesar dessa "nova geração", as soluções que surgiram mantiveram o objetivo inicial do BI. No entanto, com novo foco, essas soluções possuem uma nova abordagem como, por exemplo, implantação de sistemas ágeis com obtenção de resultados cada vez mais rápidos e confiáveis.

Um dos pioneiros no estudo de *data warehouse* foi William H. Immon, o qual apresentou um dos primeiros conceitos de *data warehouse*, "é um conjunto de dados consolidados por assunto, não é volátil e está sempre em constante variação quanto ao tempo" (INMON, 1997 *apud* VIEIRA, 2009, p. 13).

Outra definição de *data warehouse*, apresentada por Verzola (s.d., p. 8), é que

> um *Data Warehouse* é um banco de dados contendo dados extraídos do ambiente de produção da empresa, que foram selecionados e depurados, tendo sido otimizados para processamento de consulta e não para processamento de transações.

Vieira (2009) apresenta uma listagem de características que um *data warehouse* deve possuir: (1) conter um conjunto de programas para extração de dados; (2) ter um banco de dados; (3) ter um sistema para recuperação e visualização de dados. Já em referência aos possíveis benefícios que pode trazer, um *data warehouse*, segundo Vieira (2009),

pode ser descrito: (1) capacidade de acesso fácil e rápido a dados; (2) capacidade de armazenamento de dados de forma consistente; (3) ser flexível; (4) ser detentor de ferramentas de consultas e de visualização de dados; (5) realizar armazenamento confiável de dados; (6) capaz de construção de dados específicos para direcionamento de negócios.

Machado (2004 *apud* Gura e Benck, 2011) apresenta uma lista com justificativas para uma empresa implementar um *data warehouse* em sua rotina, a qual é replicada a seguir:

- Existência de várias plataformas de hardware e de software.
- Frequentes alterações nos sistemas transacionais corporativos.
- Dificuldade na recuperação de dados históricos em períodos que antecedem o ano vigente de operações.
- Existência de sistemas de diferentes fornecedores.
- Existência de despadronização e de integração de dados dos diversos sistemas utilizados.
- Ausência de documentação e de segurança no armazenamento de dados.

Um *data warehouse* não contém, necessariamente, apenas dados resumidos, ele pode conter, também, dados originais ou brutos (primitivos), coletados diretamente de uma fonte. Verzola (s.d., p. 8) afirma que "é desejável prover ao usuário a capacidade de aprofundar-se num determinado tópico, investigando níveis de agregação menores ou mesmo dados primitivos". O autor afirma ainda que é necessário, também, permitir que sejam geradas novas agregações ou correlações com outras variáveis do banco de dados.

A necessidade de uma flexibilidade mínima para um *data warehouse* se faz necessária pois, se for assim, implicaria numa restrição aos usuários a realizarem apenas consultas e análises com motivações vigentes, apenas, sem sequer permitir que novos *insights* sejam implementados (VERZOLA, s.d.).

Gura e Benck (2011) afirmam existir dois tipos de bancos de dados e, consequentemente, dois sistemas distintos para o respectivo tratamento, o banco de dados operacionais e o banco de dados analíticos ou *data warehouse*. Ainda segundo as autoras, "os bancos de dados operacionais auxiliam em todas as operações de uma organização, sendo utilizado com frequência para registrar a situação da organização" (GURA; BENCK, 2011, p. 21). Bancos operacionais são, em geral, manipulados por desenvolvedores, analistas ou programadores, o que ocorre diferentemente em um *data warehouse*, onde a manipulação do banco pode ser realizada por qualquer profissional que tenha acesso ao sistema.

Além de todos os aspectos aqui apresentados sobre um *data warehouse*, pode-se afirmar que eles "registram a história da organização, dando suporte aos administradores e gerentes no momento de tomada de decisão" (GURA; BENCK, 2011, p. 22). O Quadro 1 a seguir, apresenta um comparativo entre um banco de dados operacional e um *data warehouse*.

Quadro 1 – Comparação entre banco de dados operacional e *data warehouse*

Característica	Banco de dados operacional	*Data warehouse*
Objetivo	Operações diárias do negócio	Analisar o negócio
Uso	Operacional	Informativo
Tipo de processamento	OLTP	OLAP
Unidade de trabalho	Inclusão, alteração, exclusão	Carga e consulta
Número de usuários	Milhares	Centenas
Tipo de usuário	Operadores	Comunidade gerencial
Interação do usuário	Somente pré-definida	Pré-definida e ad-hoc
Condições dos dados	Dados operacionais	Dados analíticos
Volume	Megabytes – gigabytes	Gigabytes – terabytes
Histórico	60 a 90 dias	5 a 10 anos
Granularidade	Detalhados	Detalhados e resumidos
Redundância	Não ocorre	Ocorre
Estrutura	Estática	Variável
Manutenção desejada	Mínima	Constante

Acesso a registros	Dezenas	Milhares
Atualização	Contínua (tempo real)	Periódica (em *batch*)
Integridade	Transação	A cada atualização
Número de índices	Poucos/simples	Muitos/complexos
Intenção dos índices	Localizar um registro	Aperfeiçoar consultas

Fonte: Verzola (s.d.).

Muitos são os aspectos listados no Quadro 1 que comparam ambos os bancos de dados. Neste texto não serão detalhados todos eles. No entanto, pretende-se apresentar com mais detalhes a construção de um *data warehouse* relacionado aos índices e ao acesso aos seus registros, por meio de ferramentas ETL (extração, transformação e carregamento).

2.1 Características de um *data warehouse*

Além da necessidade de compreender o que é um *data warehouse* (DW) e o que o diferencia dos demais bancos de dados, é importante conhecer algumas das principais características associadas a esta arquitetura de dados, as quais são descritas a seguir.

2.1.1 Orientado por assunto

O DW é construído e elaborado ao redor do principal assunto da organização, como produtos, atividades, contas, clientes ou algum processo importante, além do que, só utilizará dados importantes para o processamento de apoio à decisão. Para exemplificar, considere um processo que movimente algum negócio importante de uma empresa, ou seja, o assunto. Gura e Benck (2011, p. 23) afirmam que “é a escolha correta e precisa da necessidade essencial do negócio que garantirá ou não o sucesso do projeto”.

2.1.2 Variante no tempo

Um DW necessita de dados precisos no tempo, não atualizáveis. Os dados são considerados como uma foto do momento, instantânea, como registros estáticos, obtidos em um determinado momento do

tempo prévio. Segundo Verzola (s.d., p. 11) “o tratamento de séries temporais apresenta características específicas, que adicionam complexidade ao ambiente do *data warehouse*”.

2.1.3 Não volátil

Ser “não volátil”, segundo Gura e Benck (2011, p. 23), significa que “os dados carregados no DW não podem mais ser alterados”. Depois da inclusão dos dados em um DW, somente consultas, inclusões ou exclusões podem ser realizadas. Não é possível realizar qualquer tipo de manipulação deles. Verzola (s.d., p. 11) afirma ainda que “no ambiente operacional, ao contrário, os dados são, em geral, atualizados registro a registro, em múltiplas transações”.

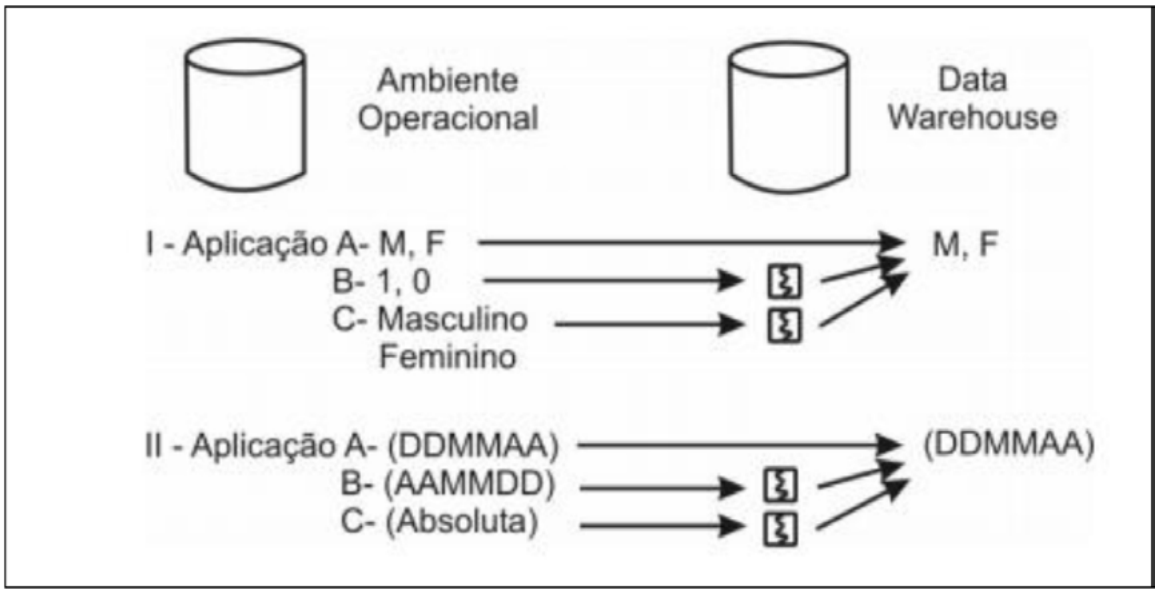
2.1.4 Granularidade

Sobre este aspecto Verzola (s.d., p. 12) declara que “diz respeito ao nível de detalhe ou de resumo contido nas unidades de dados existentes no *Data Warehouse*. Quanto maior o nível de detalhes, menor o nível de granularidade”. O autor afirma ainda que “o nível de granularidade afeta diretamente o volume de dados armazenado no *Data Warehouse* e ao mesmo tempo o tipo de consulta que pode ser respondida” (VERZOLA, s.d., p. 12).

2.1.5 Integração

Segundo Immon (1999 *apud* Gura e Benck, 2011), trata-se da mais importante característica de um sistema *data warehouse*. Os dados devem ser padronizados, ou seja, precisam ser armazenados de forma única no momento da migração do ambiente operacional para o ambiente analítico. Gura e Benck (2011, p. 24) afirmam que “a migração de dados de diferentes bases remotas para a base principal deverá depender de uma integração dos dados, convencionando nomes, variáveis de medidas e assim sucessivamente”. A Figura 1 exemplifica o processo de migração de um banco de dados operacional para um *data warehouse*.

Figura 1 – Exemplo de integração de dados



Fonte: Gura e Benck (2011).

Muitas outras características associadas a um *data warehouse* poderiam ser descritas. No entanto, as que foram apresentadas neste texto são consideradas pela maioria dos estudiosos as principais. Por isso, são descritas com mais detalhes na maioria dos textos que descrevem sobre *data warehouse*.

3. Construção de um *data warehouse*

Um *data warehouse*, para ser considerado útil, precisa ser capaz "de responder a consultas avançadas de maneira rápida, sem deixar de mostrar detalhes relevantes à resposta" (VERZOLA, s.d., p. 15). Por isso, a arquitetura necessária para sua construção exige que ele possa coletar, manipular e apresentar dados de maneira rápida e eficiente. Entende-se como eficiente, informações confiáveis que possam ser úteis no processo de tomada de decisão.

Gura e Benck (2011, p. 31) afirmam que "o processo de fazer *Data Warehouse* é denominado *Data Warehousing*". O desenvolvimento de um armazém de dados tem relação direta com a necessidade dos usuários do sistema e com a realidade de informações disponíveis para a sua integração e alimentação.

Sarkis (2001, p. 30) afirma que

> *Data warehousing*, refere-se a uma coleção de tecnologias que objetivam melhorar a tomada de decisões, desta forma, trata-se do processo e não produto. Este processo gerencia as informações para tomada de decisões das empresas através de: modelagem conceitual dos fatos do negócio, modelagem física, processo de extração de dados de sistemas existentes, limpeza, transformação e carga em um repositório de dados, o *Data Warehouse* (DW) – que tem como público alvo os tomadores de decisões gerenciais de alto nível e a longo prazo.

A partir das necessidades de um sistema de *data warehouse* surgiu o conceito de modelagem multidimensional, o qual Gura e Benck (2011, p. 27) afirmam que o modelo multidimensional "surgiu para atender sistemas de processamento analítico, com consultas para planejamento tático e estratégico da empresa", que só têm apresentado crescimento ao longo do passar do tempo desde que foram elaborados.

Segundo Machado (2004 *apud* GURA e BENCK, 2011, p. 27), "modelagem multidimensional é uma técnica em que a concepção e a visualização descrevem aspectos comuns de negócios". Essa técnica, segundo as autoras, pode ser dividida em três elementos, que são: fatos, dimensões e medidas.

Ainda é possível afirmar que um modelo multidimensional está relacionado com dois conceitos/tipos de tabelas de dados: tabela de dimensão e tabela de fatos (BARBOSA, 2016). Segundo a autora, "uma tabela de fatos é a principal tabela de um modelo dimensional" (BARBOSA, 2016, s.p.).

Em diversos ambientes, dados operacionais não integrados são complexos e muito difíceis de lidar. Para alcançar benefícios reais de um *data warehouse*, porém, é necessário passar pelo exercício doloroso, complexo e demorado de integração. Aplicativos de extração/transformação/carregamento (do inglês, *extract/transform/load*, ETL)

podem automatizar grande parte desse processo tedioso. Além disso, esse processo de integração deve ser feito apenas uma vez. Mas, em qualquer caso, é obrigatório que os dados que migram para o *data warehouse* sejam integrados, não apenas lançados no *data warehouse* a partir do ambiente operacional (INMON, 2005).

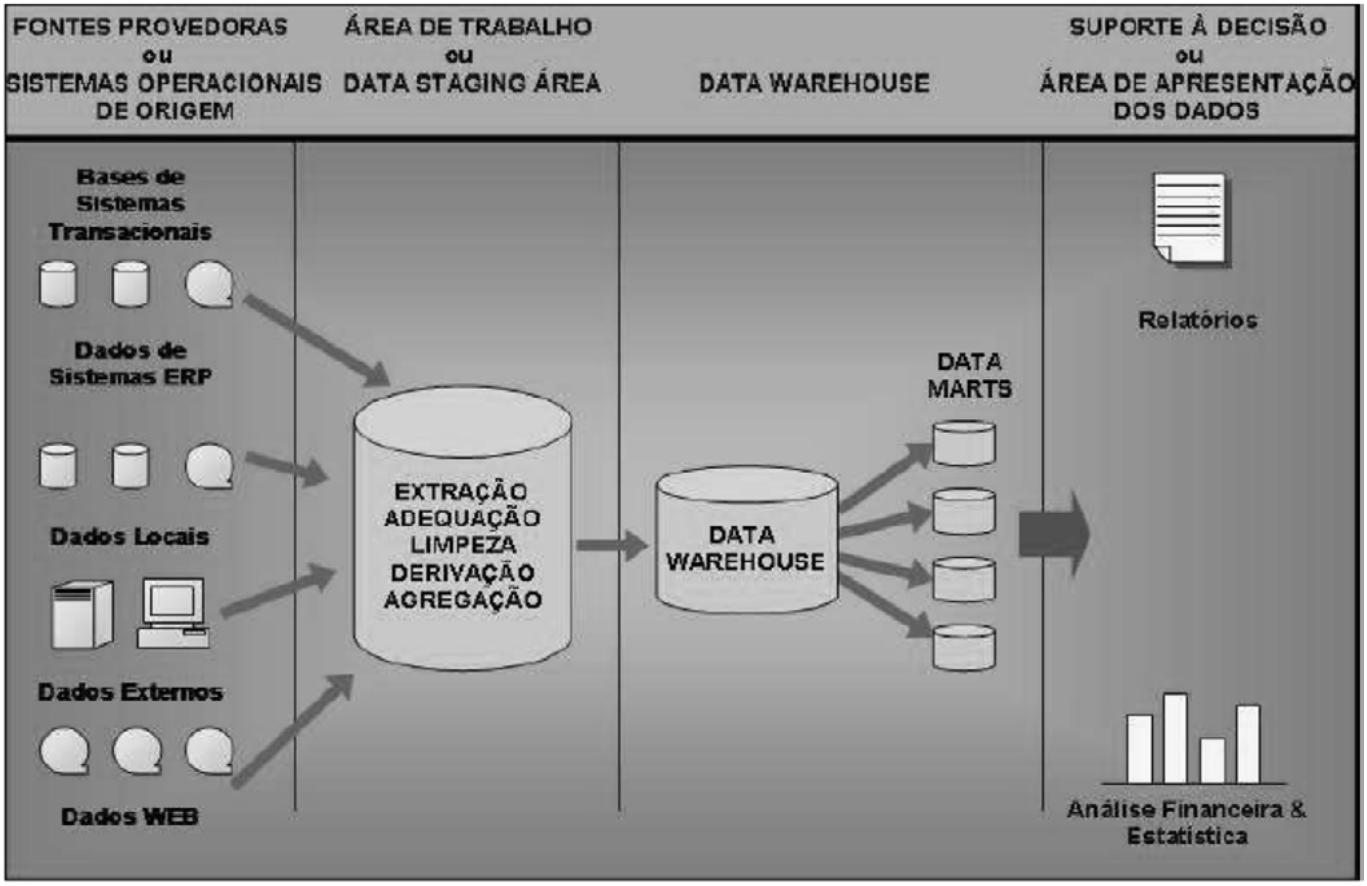
O objetivo de realizar um ETL em um *data warehouse*, segundo Novais (2012, p. 31), é "trabalhar com toda a parte de extração de dados de fontes externas, transformação para atender às necessidades de negócios e carga de dados dentro do *Data Warehouse*", visando a integração e a padronização dos dados que alimentarão o sistema.

Aplicativos de ETL, segundo Inmon (2005), podem se apresentar em duas variações: softwares que produzem códigos e softwares que produzem módulo de tempo de execução (*run-time*) parametrizado. A produção de código para aplicativos de produção de códigos é muito mais poderosa que para aplicativos/softwares de tempo de execução. Aplicativos produtores de código podem acessar dados em seu formato original, já aplicativos de tempo de execução, geralmente, exigem que os dados sejam padronizados.

De qualquer forma, sistemas de ETL automatizam o processo de conversão, reformatação e de integração de dados de várias fontes operacionais herdadas. Apenas em circunstâncias muito incomuns, tentar construir e manter a interface operacional ou de um *data warehouse* manual faz algum sentido.

Como dito antes, arquiteturas de *data warehouse* podem consolidar dados de fontes diversas, podendo grande parte ser oriunda de bancos de dados relacionais, por exemplo. No entanto, podem ser originários de outros tipos. O sistema ETL precisa ser capaz de fazer comunicação com todas as bases de dados de uma organização, assim como ser capaz de fazer a leitura de dados armazenados em diversos formatos (NOVAIS, 2012). A Figura 2 apresenta, de forma ilustrativa, um processo de construção de um *data warehouse*, ou seja, um *data warehousing*.

Figura 2 – Processo de *data warehousing* com sistemas ETL



Fonte: Gura e Benck (2011).

A extração de dados pode ser realizada com ferramentas específicas, destancando-se as ferramentas OLAP (*Online Analytical Processing*), que são ferramentas de tratamento de dados multidimensionais, que surgiram nos anos 1960. O conceito OLAP aborda a ideia de múltiplas dimensões hierárquicas e "proporcionam ao usuário grande capacidade de manipulação dos dados e análise crítica dos resultados obtidos" (SARKIS, 2001, p. 35).

Segundo Sarkis (2001), existe um consenso entre os estudiosos do assunto de que o ambiente de um *data warehouse* envolve três grandes componentes: *back end*, o repositório *data warehouse* e *front end*. Uma breve descrição sobre cada um desses componentes é apresentada a seguir.

3.1 Ferramentas *back-end*

Segundo Sarkis (2001, p. 35), "o *back end* compõe a arquitetura de dados onde o processo de transformação de dados ocorre". São ferramentas elaboradas com o objetivo de agregar e de conduzir dados de maneira correta de um ambiente para outro com o uso de ferramentas de extração, limpeza, resumo e de carga de dados.

A limpeza dos dados se justifica porque o processo de tomada de decisão baseado em grandes volumes de dados precisa de informações acuradas, ou seja, precisa estar baseado em dados confiáveis. Para tanto, é necessário que os dados que compõem um *data warehouse* tenham poucos erros ou nenhum erro, preferencialmente que tenham alto nível de consistência e sejam íntegros.

Quanto à carga dos dados, deseja-se que após a extração, limpeza e transformação, os dados sejam carregados de forma apropriada para o ambiente do *data warehouse*. Outra ferramenta considerada como *back-end* trata-se do *refresh*, que "consiste em propagar as atualizações ocorridas nos bancos de dados operacionais para o banco de dados derivado do *Data Warehouse*" (VERZOLA, s.d., p. 18).

3.2 Ferramentas *front-end*

Trata-se da interface do sistema. É o que o usuário final visualiza após realizar consultas no sistema. Segundo Moraes (1998 *apud* VERZOLA, s.d., p. 18), "o componente *front end* de um sistema de *Data Warehouse* é o responsável por fornecer uma solução de acesso aos dados que atenda as necessidades por informações dos trabalhadores do conhecimento".

São ferramentas úteis para a análise das consultas, ajudam a interpretar as informações e a tomar decisões futuras. Para isso, podem executar uma série de atividades, como: (1) seleção de dados; (2) cálculos e outros tratamentos de dados; (3) elaboração de relatórios de consultas.

Verzola (s.d., p. 18) afirma que "os geradores de consultas e relatórios são considerados a primeira geração de ferramentas para acesso a dados, as quais permitem a realização de consultas *ad-hoc*". O autor afirma ainda que as ferramentas OLAP são consideradas a segunda geração de ferramentas para acesso a dados em um *Data Warehouse*.

As ferramentas OLAP não são um software específico, mas sim um conceito no qual vários programas computacionais podem ser enquadrados como OLAP, tanto pela capacidade de manipulação de dados quanto pela capacidade de apresentação visual. É uma ferramenta que permite ao usuário final de um *Data Warehouse* usufruir da análise multidimensional (MDA), ou seja, manipular dados que estejam armazenados em diversas dimensões. Também compõe as ferramentas *front-end* o conceito de *data mining*, o qual representa um conjunto de ferramentas apropriadas para a identificação de correlação entre dados com o uso de ferramentas estatísticas, matemáticas e computacionais.

ASSIMILE

O processo de *data mining* permite a extração de conhecimento, que pode estar oculto em uma base de dados ou *Data Warehouse*. Para tanto, um conjunto de ferramentas analíticas apropriadas para a busca por estas informações foi elaborado (VERZOLA, s.d.).

3.3 Repositório *Data Warehouse*

O repositório *Data Warehouse* é o sistema em si. É o que abrange todos os componentes até aqui descritos. Segundo Sarkis (2001, p. 36), "o *Data Warehouse* é o alicerce do processamento analítico de verificar dados. No ambiente analítico, os dados destinados para a análise do usuário

final são obtidos a partir de uma fonte única (DW ou *Data Mart* (DM))". A autora complementa ainda que "o DW é um banco de dados analítico projetado especificamente para suporte à decisão, onde os dados desnormalizados permitem melhor desempenho na recuperação de informações" (SARKIS, 2001, p. 36).

PARA SABER MAIS

Segundo Verzola (s.d., p. 20), "um *Data Mart* é um sistema de suporte à decisão que incorpora um subconjunto de dados da empresa focalizado em funções ou atividades específicas da organização". É uma espécie de restrição de um *Data Warehouse*.

Há uma infinidade de conceitos e ferramentas associados ao tema *Data Warehouse*. O que fora apresentado neste texto não esgota nem de perto o assunto. Portanto, espera-se que você, por meio desta leitura, possa ficar motivado por buscar mais conhecimento sobre o tema, para, assim, se tornar um profissional capacitado desse conhecimento.

Prezado aluno! Chegamos ao final desta leitura. Que você possa ter aproveitado todo o conteúdo aqui discorrido, e que ele possa agregar de maneira significativa na sua formação!

TEORIA EM PRÁTICA

Imagine que você seja o gerente do departamento de compras de uma empresa de pequeno porte e, por ser uma empresa pequena, possui sistemas de dados mais simples e menos analíticos. A partir de um levantamento realizado por sua equipe, você percebeu que está havendo uma tendência de crescimento no ramo em que atua a empresa.

A partir deste cenário, você, como responsável por um dos principais departamentos da empresa, decide investir em sistemas de armazenamento e tratamento de dados mais avançados e robustos. Portanto, faz uma consulta ao departamento de informática para verificar a possibilidade de por em prática seus planos. A equipe de informática sugere que seja elaborado um sistema de *Data Warehouse* em seu departamento, para, assim, permitir a você e sua equipe consultar os dados que alimentam os diversos sistemas do departamento de compras, lembrando também que há um interesse em realizar consultas para avaliar as tomadas de decisões dos processos de trabalho. Pense na proposta. Como você acha que deveria ser esse sistema? Que tipo de dado seria interessante alimentar o sistema? Muitas outras perguntas e dúvidas poderão surgir com a proposta do departamento de informática. Avalie!

VERIFICAÇÃO DE LEITURA

1. Novais (2012, p. 13) apresentou um conceito associado a *Data Warehouse* cujo propósito "é permitir a tomada de decisões proativas ao gerar informações necessárias ao negócio". Qual o nome deste conceito?

   Assinale a alternativa CORRETA.

   a. *Businnes Analytics*.

   b. *Businnes Intelligence*.

   c. *Data Warehouse*.

   d. Acontecimentos.

   e. ETL.

2. São bancos de dados, em geral, manipulados por desenvolvedores, analistas ou por programadores. Qual o nome do tipo de banco de dados manipulado por estes profissionais?

   Assinale a alternativa CORRETA.

   a. Operacionais.

   b. Analíticos.

   c. Numéricos.

   d. Classificatórios.

   e. Textuais.

3. Segundo Sarkis (2001, p. 35), “compõe a arquitetura de dados onde o processo de transformação de dados ocorre”. São ferramentas elaboradas com o objetivo de agregar e de conduzir dados de maneira correta de um ambiente para outro com o uso de ferramentas de extração, limpeza, resumo e de carga de dados. Qual o nome do elemento que a autora está se referindo?

   Assinale a alternativa CORRETA.

   a. *Middle-End*.

   b. *Front-End*.

   c. *Analytics*.

   d. *Back-End*.

   e. *Data Warehouse*.

Referências bibliográficas

BARBOSA, E. M. **Integração de dados de redes sociais a armazém de dados**. 2016. 63 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Ciências Exatas, Universidade Federal de Minas Gerais, Belo Horizonte, 2016. Disponível em: https://repositorio.ufmg.br/bitstream/1843/ESBF-AKUNG3/1/elainemunizbarbosa.pdf. Acesso em: 10 nov. 2019.

GURA, E. F.; BENCK, L. L. N. **Construção de um *data warehouse*, aliado a uma ferramenta *open source ireport* na geração de informações para tomada de decisão**. 2011. 89 f. Trabalho de conclusão de curso (Curso de Tecnologia em análise e desenvolvimento de sistemas) – Coordenação de Informática, Universidade Tecnológica Federal do Paraná, Ponta Grosa, 2011. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/6449/1/PG_COADS_2011_2_05.pdf. Acesso em: 7 nov. 2019.

INMON, W. H. **Bulding the data warehouse**. 4. ed. Indianopolis: Wiley Publishing, Inc. 2005.

NOVAIS, R. R. C. **Modelagem dimensional**. 2012. 65 f. Trabalho de conclusão de curso (Curso de Tecnologia em processamento de dados) – Faculdade de Tecnologia de São Paulo, São Paulo, 2012. Disponível em: http://www.fatecsp.br/dti/tcc/tcc00071.pdf. Acesso em: 13 nov. 2019.

SARKIS, L. C. ***Data Warehouse:*** o processo de migração de dados. 2001. 146 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, 2001. Disponível em: https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/80047/227423.pdf?sequence=1&isAllowed=y. Acesso em: 10 nov. 2019.

VERZOLA, I. **Data warehouse**. [s.d.]. Barueri: Pontes Computadores e Serviços Ltda. Disponível em: http://www.pontes.inf.br/docs/datawarehouse.pdf. Acesso em: 7 nov. 2019.

VIEIRA, E. **Tecnologia olap**. 2009. 38 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Instituto Municipal do Ensino Superior de Assis, Assis, 2009. Disponível em: https://cepein.femanet.com.br/BDigital/arqTccs/0411150200.pdf. Acesso em: 7 nov. 2019.

Gabarito

Questão 1 – Resposta B

Novais (2012, p. 13) apresentou um conceito associado a *Data Warehouse* cujo propósito "é permitir a tomada de decisões proativas ao gerar informações necessárias ao negócio", cujo nome é *Businness Intelligence*.

Questão 2 – Resposta A

Bancos manipulados exclusivamente por desenvolvedores, analistas e/ou programadores são os bancos operacionais.

Questão 3 – Resposta D

Ferramentas *back-end* "compõem a arquitetura de dados onde o processo de transformação ocorre" (SARKIS, 2001, p. 35).

Definindo um projeto de *Big Data* e as bases e arquivos a considerar

Autor: Marcelo Tavares de Lima

Objetivos

- Apresentar conceitos fundamentais de *Big Data*.
- Descrever sobre tratamento de dados em linguagem *Python*.
- Apresentar aplicações de *Big Data* em linguagem *Python*.

1. Introdução

A produção de dados e de informações a partir destes tem crescido cada vez mais rapidamente à medida que os recursos tecnológicos e computacionais evoluem. Como reflexo, é possível perceber que existe uma disputa acirrada para o armazenamento de grandes volumes de dados, conhecidos como *Big Data*, e não só é disputado o armazenamento eficiente, mas, também, o tratamento e a divulgação adequada para este tipo de informação.

Este texto apresenta conceitos básicos e fundamentais associados ao termo *Big Data*, o qual não se refere apenas ao grande volume de produção, mas ao conjunto de conceitos e recursos tecnológicos que tratam desde o armazenamento eficiente de grandes massas de dados até ferramentas computacionais apropriadas para a realização de análise de dados.

2. Conceitos fundamentais de *Big Data*

Big data, no sentido literal da palavra, significa grande volume de dados. No entanto, o sentindo prático e real do termo não se limita somente a isso, pois é muito mais. Além de se referir a grandes volumes de dados, também se refere à grande variedade de dados manipulados das diversas fontes disponíveis e, também, se refere à velocidade em que os dados são tratados.

Taurion (2013) acrescenta ao conceito de *Big Data* mais duas variáveis importantes, a veracidade dos dados (os dados têm significância ou são apenas sujeira?) e o valor para o negócio. E, por último, um elemento que tem estado no centro das atenções, a questão da privacidade dos dados, tema complexo e controverso, segundo o autor. Uma regulamentação sobre proteção de dados pessoais foi criada em 2018, a Lei nº 13.709/2018 – Lei geral de proteção de dados pessoais – que tem gerado investimentos nas empresas para proteger e garantir proteção a dados pessoais.

Não existe um consenso quando se trata do conceito básico de *Big Data*. Para exemplificar, Taurion (2013, n.p.) apresenta a definição dada pela McKinsey Global Institute como "o termo *Big Data* refere-se a este conjunto de dados cujo crescimento é exponencial e cuja dimensão está além da habilidade das ferramentas típicas de capturar, gerenciar e analisar dados".

Taurion (2013, n.p.) ainda apresenta a definição dada pelo Gartner, declarando que define como *Big Data* "o termo adotado pelo mercado para descrever problemas no gerenciamento e processamento de informações extremas as quais excedem a capacidade das tecnologias de informações tradicionais ao longo de uma ou várias dimensões".

Uma analogia feita por Taurion (2013) com respeito à *Big Data* e medicina afirma que *Big Data* é um microscópio, o qual permitiu que se vissem coisas que já existiam, como bactérias e vírus, mas que não se tinha conhecimento. É a tão conhecida descoberta do conhecimento por meio dos dados.

A grande massa de dados, invisíveis até então, agora pode ser tratada como *Big Data* e receber o tratamento analítico apropriado para, então, se tornar informações úteis na tomada de decisão de negócios. Então, em tempos de B*ig Data*, com o "microscópio" descoberto, percebeu-se que os dados surgem de fontes diversas. Eles surgem dos milhares de web sites existentes, dos mais de cem mil tuítes publicados por minuto, dos compartilhamentos de bilhões de usuários do Facebook e outras redes sociais, dos sensores e câmeras espalhados pelas cidades e, também, não podiam deixar de ser citados, os bilhões de celulares existentes em funcionamento ao redor do mundo.

O espaço ocupado por esses "novos" dados, que passaram a ser vistos e enxergados como fonte de informação para os negócios, é muito maior, e a sua produção em massa requer espaços amplos para o seu armazenamento. Por exemplo, o uso de imagens e vídeos divulgados

nas redes sociais requer muito mais espaço em memória do que um simples dado numérico. Um vídeo em alta definição ocupa muito mais espaço para armazenamento em comparação a uma página de texto, e assim por diante.

Amaral (2016, n.p.) apresenta uma definição para *Big Data*, o autor afirma que "*Big Data* é o fenômeno em que dados são produzidos em vários formatos e armazenados por uma grande quantidade de dispositivos e equipamentos". As causas do "fenômeno" *Big Data*, basicamente, são associadas aos investimentos feitos em tecnologia, como por exemplo, investimentos em unidade central de processamento (CPU), memórias e unidades de armazenamento, dentre outros equipamentos, tornando-os cada vez mais baratos.

Com o barateamento dos insumos e a sua miniaturização, assim como o aumento de suas capacidades de processamento, tem-se como consequência o aumento do acesso a mais equipamentos, dispositivos e processos com maiores capacidades de produção e armazenamento de dados. A computação em nuvem e a internet, por exemplo, também fazem parte desse pacote de consequências, o que para Amaral (2016) trata-se do que é conhecido como *Big Data*.

A velocidade com que o mundo tem se tornado digital é demasiadamente veloz. Por exemplo, na primeira década dos anos 2000 apenas metade dos dados produzidos no mundo estavam armazenados em formato digital, enquanto que, no final da segunda década (2019), quase 100% dos dados se encontravam neste formato.

Em notação matemática, Taurion (2013) afirma que em uma equação, "*Big Data* = volume + variedade + velocidade + veracidade", tudo agregando valor. É o que se conhece como conjunto de cinco V's do *Big Data*. De todos os componentes desta equação matemática, apenas o 'agregar valor' não foi discutido em detalhes até aqui. No entanto, uma descrição sobre este componente é apresentada a seguir.

Agregar valor à implantação de *Big Data* nada mais é do que o retorno esperado de todo o investimento realizado em tecnologia e mão de obra para equipar o negócio. Outro conceito que não pode deixar de ser citado quando se fala de B*ig Data* é o de internet das coisas (IoT). Taurion (2013, n.p.) afirma que "a Internet das Coisas vai aglutinar o mundo digital com o mundo físico, permitindo que os objetos façam parte dos sistemas de informação".

Com os objetos componentes da estrutura física que nos rodeia gerando dados a todo instante, tem-se mais do que nunca um impulsionador de grande capacidade para a produção de *Big Data*. Por exemplo, quase todos os componentes de um avião, quando em funcionamento, podem gerar dados diversos, permitindo que sejam utilizados para realizar previsões meteorológicas e, também, com respeito à manutenção, evitar que a aeronave fique parada desnecessariamente, gerando custos sem ganhos. Outro exemplo é a automação residencial ou domótica, a qual realiza integração entre os equipamentos de uma residência, tornando-a uma casa inteligente.

Com relação à disseminação de equipamentos e processos citados por Amaral (2016), o entendimento do fenômeno *Big Data* pode ser realizado com um breve levantamento histórico, lembrando que há poucas décadas tinha-se a produção de dados centralizadas em *mainframes* (computadores gigantescos e centralizados) e em computadores pessoais.

Na era *Big Data*, os dados passaram a ser produzidos por diversas fontes, tais como "redes sociais, comunidades virtuais, blogs, dispositivos médicos, TVs digitais, cartões inteligentes, sensores em carros, trens e aviões, leitores de código de barra, [...], celulares, sistemas informatizados, satélites, entre outros" (AMARAL, 2016, n.p.). Os dados produzidos por essas fontes diversas têm formatos diversos também, e diferentes velocidades em sua produção, além de serem de volumes variados.

Um termo bastante associado ao conceito de *Big Data* é o de *analytics*, o que é entendido como um termo amplo que engloba processos e tecnologias. O *analytics* é o processo de extração e criação de informações a partir de dados brutos por filtragem, processamento, categorização, condensação e contextualização dos dados (BAHGA; MADISETTI, 2019). *Analytics* é o tratamento aplicado em dados para transformá-los em informação útil para ser utilizada nos processos de tomada de decisão e para divulgação de resultados.

A união do conceito de *Big Data* com *analytics* gerou o conceito *Big Data Analytics*, cujo termo pode ser definido como "o estudo e interpretação de grandes quantitades de dados armazenados com a finalidade de extrair padrões de comportamento" (BAHGA; MADISETTI, 2019, p. 25, tradução nossa). O *Big Data Analytics* se utiliza de uma combinação de sistemas de programas computacionais matemáticos e estatísticos de alta tecnologia que juntos são capazes de tratar dados estruturados e não estruturados, realizar análise desses e extrair valor de grande importância na tomada de decisão dos negócios (BAHGA; MADISETTI, 2019).

Imagine como uma empresa mantinha disponibilizadas informações sobre seus funcionários na década de 1990? Por exemplo, elas tinham currículos e formulários preenchidos em seus processos seletivos; outras informações em sistemas de folhas de pagamento e, em geral, com difícil acesso; possuíam, também, dados de desempenho individual, coletados de forma esporádica por um superior hierárquico. E no século XXI? Como será que as empresas guardam as informações de seus colaboradores? Especificamente na segunda década dos anos 2000, podia-se buscar informações sobre um desses colaboradores em redes sociais, seu histórico de uso de internet, seus e-mails; imagens e vídeos elaborados por eles, dentre outras fontes. Todos estes fenômenos, em sua maioria, sempre ocorreram, a diferença é que, de um certo tempo em diante, passaram a ser registrados digitalmente.

Um exemplo muito prático trata-se do comparativo entre celulares e computadores da década de 1980, feito por Amaral (2016, n.p.), no qual concluiu que "*big data* fica ainda mais compreensível quando falamos em números: um *smartphone* de hoje tem maior capacidade que o melhor computador de 1985".

Amaral (2016) apresenta alguns números para dar uma ideia da dimensão do volume de dados atualmente existente, tal como a quantidade de pessoas com aparelho celular em todo o mundo, que contam mais de seis bilhões, e são em torno de dois bilhões de pessoas utilizando rede social; cerca de três milhões de e-mails são enviados por segundo no mundo, quinhentos milhões de tuítes por dia e por aí vai.

É importante, também, depois de todo um esforço para definir *Big Data*, saber e definir o que não é *Big Data*. Ficar restrito unicamente em processos de geração de grande volume de dados não é *Big Data*, pois, além disso, o seu conceito também inclui "mudança social, cultural, é uma nova fase da revolução industrial" (AMARAL, 2016, n.p.). Como já dito, *Big Data* é um fenômeno, e não uma tecnologia.

Por ser considerado um fenômeno, *Big Data* "envolve o uso de diversos tipos de conceitos e tecnologias, como computação nas nuvens, virtualização, internet, estatística, infraestrutura, armazenamento, processamento, governança e gestão de projetos" (AMARAL, 2016, n.p.).

O armazenamento indiscriminado de dados oriundos de diversas fontes é uma característica muito associada ao conceito de *Big Data*. Em tempos passados, armazenamento tinha um custo monetário alto a ser dispensado. Por isso, armazenava-se apenas o dado que era visto possuir algum valor imediato. Em tempos de *Big Data* esse cenário tem se modificado continuamente.

3. Tratamento de dados com *Python*

A linguagem *Python* foi criada em 1989 pelo pesquisador Guido van Rossum, do *National Research Institute for Mathematics and Computer Science in Amsterdam* – CWI (SANTOS, 2018). O nome *Python* foi dado à

linguagem por conta de um seriado de comédia que existia na época, cujo nome era Tropa *Monty Python*. O pesquisador criou a linguagem baseado em sua necessidade de uma linguagem de alto nível para desenvolver suas pesquisas. Naquele ano não existiam linguagens de programação que atendessem as necessidades de van Rossum. Foi então que ele tomou a decisão de desenvolver sua própria linguagem de programação. A linguagem *Python* foi desenvolvida em código aberto (*Open Source*), e a sua primeira versão foi disponibilizada em 1991.

Rossum, na realidade, tinha habilidades em desenvolver programação paradigma funcional estrutural, no entanto, sabia que o futuro da programação tendia para a programação orientada a objetivos (SANTOS, 2018). Foi então que teve a ideia de criar uma linguagem multiparadigma, a linguagem *Python*. Sua intenção era fazer comunicação com as máquinas de uma maneira mais simplificada possível e menos verbosa que as linguagens existentes na época, retirando as chaves e parênteses excessivos que as linguagens de programação da época continham.

O ambiente de desenvolvimento da linguagem *Python* ou IDE (*Integrated Development Environment*), assim como para a linguagem R, é bastante utilizado por conta de uma série de facilidades que trazem para o uso da linguagem. Em número, também são vários, pois, além de versões distintas, diferentes IDEs e diferentes plataformas exigem algumas condições para que possam ser instaladas. Também existem versões gratuitas e versões pagas. As interfaces (IDE) que serão utilizadas para o desenvolvimento de linguagem de programação Python serão a *Anaconda-Spyder* e *Jupiter Notebook*, que podem ser encontradas facilmente na internet.

Santos (2018, n.p.) afirma que

> O mercado de trabalho para programadores demanda *Python* como uma das principais linguagens de programação, isso porque o seu uso para programação nas áreas de ciência de dados, análise de dados e inteligência artificial como um todo faz uso principalmente dessa linguagem.

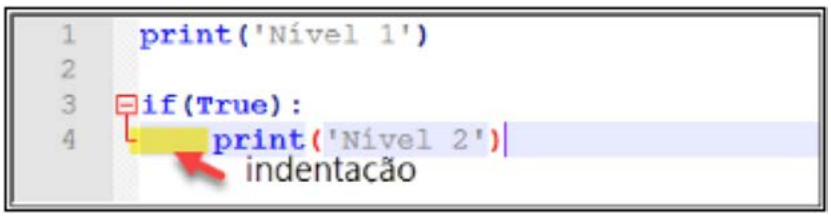
Dentre os tipos de linguagens existentes, classificadas como compiladas e interpretadas, a linguagem *Python* é considerada uma linguagem interpretada simples (SANTOS, 2018). É considerada uma linguagem interpretada simples porque para sua compilação precisa de um interpretador interno à maquina onde é inserida. A função do interpretador é traduzir a linguagem *Python* para a linguagem de máquina. Seu código fonte é convertido para *bytecode*, que tem formato binário e tem instruções para o interpretador.

Uma importante característica do *bytecode* é que ele é um multiplataforma, podendo ser executado em diversos sistemas operacionais como Linux, Windows e Mac. Santos (2018) descreve que é possível executar códigos em linguagem *Python* de três maneiras: (1) por meio do *shell* (execução de comandos linha a linha); (2) por meio de *scripts* (arquivos de texto com extensão .py) e; (3) por modo interativo, a partir de um *browser* (*Jupiter Notebook*).

Dado o intuito em eliminar o excesso de parênteses e chaves, a linguagem *Python* utiliza o recurso de indentação (recuo) para estruturar seus comandos. A intenção do uso desse recurso é de tornar os códigos mais claros e mais simplificados. No entanto, é necessário um grau de atenção maior na hora de estruturar uma série de linhas de comandos.

Santos (2018, n.p.) afirma que “a indentação serve para definir ou destacar a estrutura de um algoritmo a partir de recuos” e, para realizar um recuo ou indentação, utiliza-se a tecla “tab” do computador ou aperta-se a tecla de espaço quatro vezes. Dado isto, percebe-se que a indentação é parte da sintaxe da linguagem *Python* e deve ser respeitada. A Figura 1 apresenta um exemplo de comando em linguagem *Python* com uso de indentação.

Figura 1 – Exemplo de código em Python com indentação



Fonte: Santos (2018).

É possível documentar *scripts* (conjuntos de linhas de comandos) por meio de comentários, sem que este seja entendido como uma linha de comando para o interpretador de linguagem *Python*. Para isso, utiliza-se caracteres especiais que fazem com que comentários não sejam compilados. Santos (2018) descreve duas principais maneiras de se fazer um comentário em um *script*: (1) parágrafos entre três aspas simples (‘’’ ... ‘’’); (2) parágrafos após o símbolo de cerquilha (#).
O Quadro 1 apresenta algumas linhas de comando em linguagem *Python* incluindo linhas de comentários.

Quadro 1 – Exemplo de programação em *Python* com comentários

```
import numpy as np
import pandas as pd
from math import ceil

''' ESTE É UM COMENTÁRIO '''

# ESTE É OUTRO COMENTÁRIO

populacao = 150
amostra = 15
```

Fonte: elaborado pelo autor.

No mundo da ciência de dados, os dados brutos podem surgir em vários formatos e tamanhos. No entanto, existe uma infinidade de informação que pode ser extraída desses dados brutos. Para dar um exemplo, Madhavan (2015) cita a empresa Amazon, a qual coleta dados do fluxo de cliques de pessoas que visitam a página de internet da empresa.

Os dados podem ser utilizados para entender se um usuário é um potencial cliente sensível ao preço ou alguém que tenha preferência por produtos.

O primeiro passo para uma análise de dados seria avaliar os dados brutos que pode envolver, segundo Madhavan (2015), as seguintes etapas: (1) extração dos dados de uma fonte e, (2) limpeza dos dados.

PARA SABER MAIS

Para trabalhar com a linguagem *Python* pode-se utilizar uma série de bibliotecas. Elas ajudam a dar o tratamento devido aos dados por meio da maneira que se deseja analisá-los. Por exemplo, a biblioteca NumPy é uma biblioteca básica apropriada à execução de cálculos científicos básicos com dados.

A extração de dados brutos de uma fonte de dados pode coletar registros em diversos formatos, tais como dados de planilhas como a MS-Excel, dados em CSV (valores separados por vírgulas), dados em JSON (*JavaScript Object Notation*), dentre outros. A linguagem *Python* facilita a leitura de dados dessas fontes com a ajuda de alguns pacotes úteis.

Depois de obtidos, é preciso limpar os dados adequadamente para que possam ser utilizados para análise. Você pode ter um conjunto de dados sobre os alunos de uma turma e, por exemplo, desejar coletar dados sobre altura e peso. É possível que no conjunto de dados não haja alguma informação de peso e altura para algum aluno da base. Dependendo da análise que está sendo realizada, essas linhas com valores ausentes podem ser ignoradas ou substituídas pela altura ou peso médio. É preciso analisar o impacto que esses ajustem trarão para análises posteriores.

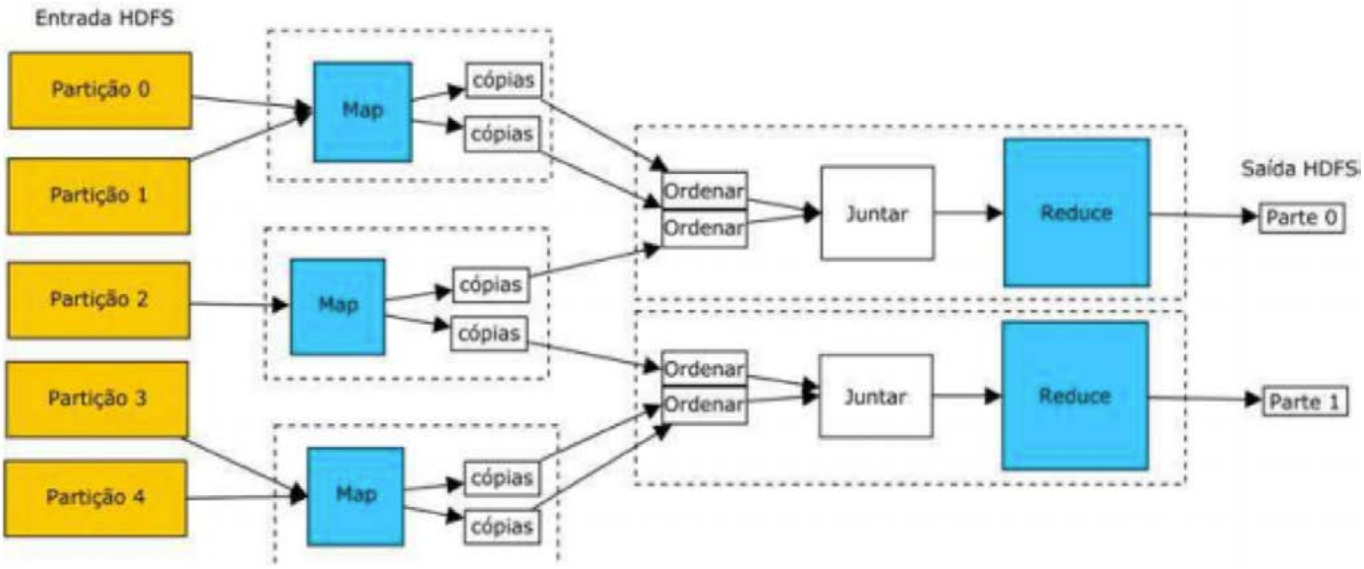
O tratamento de grandes volumes de dados por intermédio da linguagem *Python* pode ser feito pela plataforma Hadoop, que é de computação distribuída, com alta escalabilidade, de grande confiabilidade e bastante tolerante a falhas. De acordo com o site do Apache Hadoop, o Hadoop armazena dados em um. Foi projetado para ser dimensionado de maneira fácil para qualquer quantidade de máquinas com a ajuda do poder computacional e de armazenamento. A Hadoop foi criada por Doug Cutting e Mike Cafarella no ano de 2005 (MADHAVAN, 2015). O nome Hadoop era o nome do elefante de brinquedo do filho de Doug Cutting.

O Hadoop é um paradigma de programação que lida com computação distribuída, como uma sequência de operações distribuídas em grandes conjuntos de dados de pares de valores-chave. A estrutura *MapReduce*, proposta pela Google (Araújo e Montini, 2016), utiliza *cluster* de máquinas e executa trabalhos de *MapReduce* nessas máquinas. Há duas fases no *MapReduce* – uma fase de mapeamento e uma fase de redução. Os dados de entrada para o *MapReduce* são pares de dados de valor-chave (MADHAVAN, 2015).

O *MapReduce* é um conjunto de bibliotecas que permite realizar processamento em paralelo, de grandes quantidades de dados, usando todo o hardware disponível em um *cluster* Hadoop. Ele divide o processamento em duas etapas principais: (1) mapeamento e validação dos dados (chamado MAP); (2) os dados validados na etapa MAP são tratados, gerando como resultado os valores finais do processo.

Durante a fase de redução, o redutor recebe o par intermediário de valor-chave e chama uma função definida pelo usuário, que gera um par de valores-chave de saída. O Hadoop distribui os redutores pelas máquinas e atribui um conjunto de valores-chave pares para cada um dos redutores (MADHAVAN, 2015). A Figura 2 apresenta um esquema do modelo *MapReduce*.

Figura 2 – Exemplo de uma estrutura de *MapReduce*



Fonte: Araújo e Montini (2016).

A implementação do *MapReduce* é feita pelo Apache Hadoop, o qual é *open source* e fornece estrutura para executar as aplicações de um modelo de *MapReduce* (ARAÚJO e MONTINI, 2016). Ele também "permite o armazenamento confiável e distribuído de dados, utilizando dispositivos de armazenamento baratos (*commodity hardware*)" (ARAÚJO; MONTINI, 2016, p. 92).

Os dados a serem analisados são armazenados de forma distribuída por meio do sistema de arquivos distribuídos *Hadoop Distributed File System* (HDFS), o qual é uma implementação de código aberto (*open source*) da *Google File System* (ARAÚJO; MONTINI, 2016).

Segundo Araújo e Montini (2016, p. 92)

> O HDFS é um sistema de arquivos distribuídos que realiza atualizações e cargas simultâneas; diante disso, torna-se possível armazenar grandes bancos de dados em grandes conglomerados de máquinas (*clusters*). O arquivo de dados é compartilhado em aglomerados computacionais formados por diversos "nós". O modelo *MapReduce* possibilita o processamento dos dados armazenados nos nós distribuídos com grande eficiência e escalabilidade.

A plataforma Hadoop pode ser obtida no site da empresa (HADOOP, 2019). Para executar códigos em linguagem *Python*, pode-se utilizar a interface de programação de aplicação (API) de *streaming* do Hadoop, a qual auxilia no uso de programação que possua uma entrada e saída padrão, tal como um programa *MapReduce*.

O *framework* responsabiliza-se por copiar o programa e processá-lo para o computador, onde estão armazenados, além de armazenar todo o resultado obtido (ARAÚJO; MONTINI, 2016).

Um exemplo de implementação encontra-se em Araújo e Montini (2016), cujos autores utilizam dados de negociação *tick by tick* de ações cotadas na Bovespa. Segundo os autores

> os dados foram obtidos na BMF&Bovespa, por meio de cuja base de dados podem-se analisar não só séries de tempo evento a evento de todas as ações e de todos os contratos de opções e derivativos negociados (preço, volume, corretora), mas também as séries de ofertas de compra e venda (ARAÚJO; MONTINI, 2016, p. 96).

Os dados analisados correspondem ao período de 2012 a 2014 e possuem tamanho em cerca de 80 GB e, mesmo sendo possível armazenar essa quantidade de dados em um único disco, Araújo e Montini (2016) afirmam que o tratamento analítico é complicado.

A análise dos dados foi realizada em duas etapas. Na primeira etapa, Araújo e Montini (2016) desenvolveram funções para o tratamento analítico dos dados e estimaram medidas de risco por meio de linguagem *Python*. Na segunda etapa, os autores utilizaram o Apache Hadoop e o *MapReduce* (por intermédio do Hadoop *streaming*) para realização de cálculo distribuído para a estimação do modelo que desejam obter, um modelo econométrico conhecido como modelo de volatilidade. Mais detalhes metodológicos utilizados podem ser encontrados em Araújo e Montini (2016).

Foram utilizadas as bibliotecas de análise de dados Pandas e NumPy, assim como uma biblioteca de análise estatística de dados chamada Statmodels. Mais detalhes sobre estas bibliotecas podem ser encontrados em Madhavan (2015).

Araújo e Montini (2016) disponibilizaram trechos da programação em linguagem *Python* utilizada para a realização de sua pesquisa, os quais são replicados na Figura 3.

Figura 3 – Código em Python utilizado para mapeamento de dados

```
def mapper(arquivo):
    def _agrupador(indice):
        # chave emitida pelo map é (simbolo, ano, mes)
        chave = (indice[0], indice[1].year, indice[1].month)
        return(chave)

    bmf = dados_bmf()
    for chave, tabela in bmf.dados.groupby(_agrupador):
        dados = _df_para_string(tabela)
        print("{}\t{}".format(chave,dados))
```

Fonte: Araújo e Montini (2016).

Os comandos apresentados na Figura 3 foram utilizados na etapa de *mapping* da execução, na qual os dados brutos de cada transação realizada na bolsa de valores foram analisados e agrupados por símbolo do instrumento, ano e, também, por mês, apresentados na linha do comando chave (quarta linha de comando – Fase *Map*).

Também são apresentados comandos em linguagem *Python* para a fase *Reduce* do *MapReduce*. Nesta última, os autores obtiveram as séries de volatilidades percebidas (medida econométrica) para períodos diários, semanais e mensais (ARAÚJO; MONTINI, 2016). Os comandos utilizados pelos autores são apresentados nas Figuras 4 e 5.

Figura 4 – Código em *Python* utilizado para redução de dados

```
def reducer():
    def _read_mapper_output(file, separator='\t'):
        for line in file:
            yield(line.rstrip().split(separator, maxsplit=1))

    # dados_codificados é a tupla gerada pelo mapper (chave, frame)
    for chave, dados_codificados
        in groupby(_read_mapper_output(sys.stdin), itemgetter(0)):
        simbolo, ano, mes = eval(chave)
        bmf = dados_bmf(dados=pd.concat((_string_para_df(frame[1]).reset_index()
    for frame in dados_codificados)))
        RV = bmf.RVd()
        print(_df_para_string(RV))
```

Fonte: Araújo e Montini (2016).

ASSIMILE

Um algoritmo computacional é algo semelhante, por analogia, a uma receita de bolo, na qual tem-se os ingredientes (variáveis) e, também, as ações (métodos e funções) que devem ser realizadas com eles. Tudo isso, é claro, deve seguir uma estrutura lógica, uma ordenação de funcionamento e etapas bem definidas.

As demais linhas de comando são apresentadas na Figura 5, como dito anteriormente. Os resultados da execução da programação apresentada podem ser encontrados em Araújo e Montini (2016) com mais detalhes, inclusive com a interpretação e análise das informações encontradas.

Figura 5 – Código em *Python* utilizado para redução de dados (continuação)

```
def RVd(dados):
    precos = dados[['preco','qde']]
    media_precos = precos.resample('5min',
                                   how = lambda x:
                                   np.ma.average(x['preco'], weights=x['qde']))

    media_precos['preco'].interpolate(method='nearest', inplace=True)
    p = np.log(media_precos['preco'])
    r = np.square(p - p.shift())

    RV = pd.DataFrame()
    RV['dia'] = r.resample(feriados.bmf, how=lambda x: np.sqrt(x.sum()))
    return(RV)

def RVx(dados):
    dados['semana'] = pd.rolling_mean(dados['dia'], window=5)
    dados['mes']    = pd.rolling_mean(dados['dia'], window=20)
    return(dados.dropna())

def combine(RVd):
    bmf = dados_bmf(dados=RVd)
    volatilidade = bmf.dados.groupby(level=0).apply(RVx)
```

Fonte: Araújo e Montini (2016).

Analisar dados em tempos de alta produção de informações é um grande desafio para qualquer profissional. O desafio se torna maior ainda quando se deseja produzir resultados relevantes para uso em análise de negócios e tomada de decisão empresarial.

Os conceitos de *Big Data*, assim como os recursos computacionais que a linguagem *Python* fornece para lidarmos com essa gigante massa de dados produzidas a cada momento, não se esgotam neste texto. Ele, basicamente, é um mero aperitivo de muito ferramental que pode ser utilizado e conhecido com mais detalhes. Portanto, não deixe de buscar mais sobre o conteúdo apresentado.

TEORIA EM PRÁTICA

Na empresa que você trabalha são produzidos muitos dados sobre uma série de medidas e registros. No seu departamento, por exemplo, o departamento de marketing, você e seus pares produzem dados sobre campanhas e sobre intenção de consumo de clientes de vários ramos de produtos. A grande massa de dados produzida traz para vocês um grande desafio, não só para armazenamento, mas, também, para tratamento e análise. Você recebeu um treinamento recente sobre linguagem *Python*, o que fez com que tivesse a ideia de utilizar esse recurso computacional para realizar o tratamento e a análise dessa grande massa de dados que sua empresa produz. Você acha que será possível implementar a sua ideia? Como faria? Eis o desafio a postos!

VERIFICAÇÃO DE LEITURA

1. Segundo Taurion (2013), verificar se um conjunto de dados tem alguma significância real, recebe um determinado nome. Qual o nome deste termo?

   Assinale a alternativa CORRETA.

   a. Valor de negócio.

   b. Veracidade dos dados.

   c. Privacidade dos dados.

d. Análise de dados.

e. *Big Data*.

2. Em qual ano foi criada a linguagem *Python*?

   Assinale a alternativa CORRETA.

   a. 1988.

   b. 1991.

   c. 1989.

   d. 1992.

   e. 1990.

3. Para o seu bom aproveitamento, a linguagem Python recorre a um conjunto de bibliotecas elaboradas por diversos estudiosos do assunto. Qual o nome da biblioteca que realiza cálculos científicos básicos?

   Assinale a alternativa CORRETA.

   a. Python.

   b. IPython.

   c. MapReduce.

   d. Hadoop.

   e. NumPy.

Referências bibliográficas

AMARAL, F. **Introdução à ciência de dados:** mineração de dados e Big Data. Rio de Janeiro: Alta Books, 2016. KINDLE. Não paginado.

ARAÚJO, A. C.; MONTINI, A. A. Técnicas de *Big Data* e projeção de risco de mercado utilizando dados em alta frequência. **Future Studies Research Journal**. São Paulo, v. 8, n. 3, p. 83-108, set./dez. 2016. Disponível em: https://revistafuture.org/FSRJ/article/view/219/375. Acesso em: 17 nov. 2019.

BAHGA, A.; MADISETTI, V. **Big data science & analytics:** a hands-on approach. Arshdeep Bahga & Vijay Madisetti, 2019. ISBN: 978-1-949978-00-1.

HADOOP. **Apache™ Hadoop®**. 2019. Disponível em: https://hadoop.apache.org/. Acesso em: 17 nov. 2019.

MADHAVAN, S. **Mastering Python for Data Science**. Brimingham, UK: Packt Publishing, 2015. Disponível em: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1058787&lang=pt-br&site=ehost-live. Acesso em: 17 nov. 2019.

SANTOS, R. F. V. C. **Python:** guia prático do básico ao avançado. Série cientista de dados. 2018. E-BOOK KINDLE. Não paginado.

TAURION, C. **Big Data**. Rio de Janeiro: Brasport, 2013. EPUB. Não paginado. Disponível em: https://bv4.digitalpages.com.br/#/legacy/epub/160676. Acesso em: 9 jul. 2019.

Gabarito

Questão 1 – Resposta B

Para verificar se os dados possuem alguma significância, Taurion (2013) associa ao termo de veracidade dos dados.

Questão 2 – Resposta C

A linguagem *Python* foi criada no ano de 1989.

Questão 3 – Resposta E

A biblioteca da linguagem *Python* que realiza cálculos científicos básicos é a NumPy.

Criando e tratando processos utilizando o R Programming. Criando e definindo *Data Lakes* a partir das bases disponibilizadas para tratativas

Autor: Marcelo Tavares de Lima

Objetivos

- Apresentar procedimentos para tratamentos de processos e dados com o R *Programming*.
- Apresentar conceitos fundamentais de *Data Lake*.
- Apresentar aplicações de *Data Lake* com R *programming*.

1. Introdução

São diversas as ferramentas computacionais para o tratamento de grandes volumes de dados. No entanto, a escolha de uma boa ferramenta para a tratativa de processos analíticos é importante para a obtenção de resultados importantes e significativos para a tomada de decisão em cada contexto.

Uma boa ferramenta computacional é aquela que atende as necessidades do processo onde a grande massa de dados está inserida, ou seja, não existe uma única ferramenta computacional adequada que atenda todas as demandas. Portanto, o profissional de ciência de dados precisa compreender a dinâmica da produção dos dados e a exigência que eles requerem para apresentar resultados.

Existem muitas ferramentas computacionais para análise de grandes volumes de dados, principalmente dos *Big Data*. Há um avanço significativo na produção de ferramental computacional para a análise avançada e eficiente de dados oriundos de diversas áreas, como, ciências sociais, educação, agricultura, ciências da saúde, engenharias etc.

A proposta deste texto é apresentar a linguagem R como ferramenta computacional para o tratamento de grandes volumes de dados. Serão apresentados alguns conceitos básicos e conceitos de *Data Lakes*, que são arquiteturas de armazenamento de massas de dados semelhantes a um *Data Warehouse*. Aproveite esta leitura e tenha um bom momento de estudo!

2. Processos com o R *Programming*

O programa R é um software para tratamento e análise de dados, é gratuito e de código aberto (*open source*). Ele é resultante de um projeto elaborado por diversos colaboradores ao redor do mundo, composto

por implementações para análises estatísticas desde as mais simples até a mais complexas, além de poder manipular grandes volumes de dados (SIQUEIRA; TIBÚRCIO, 2011).

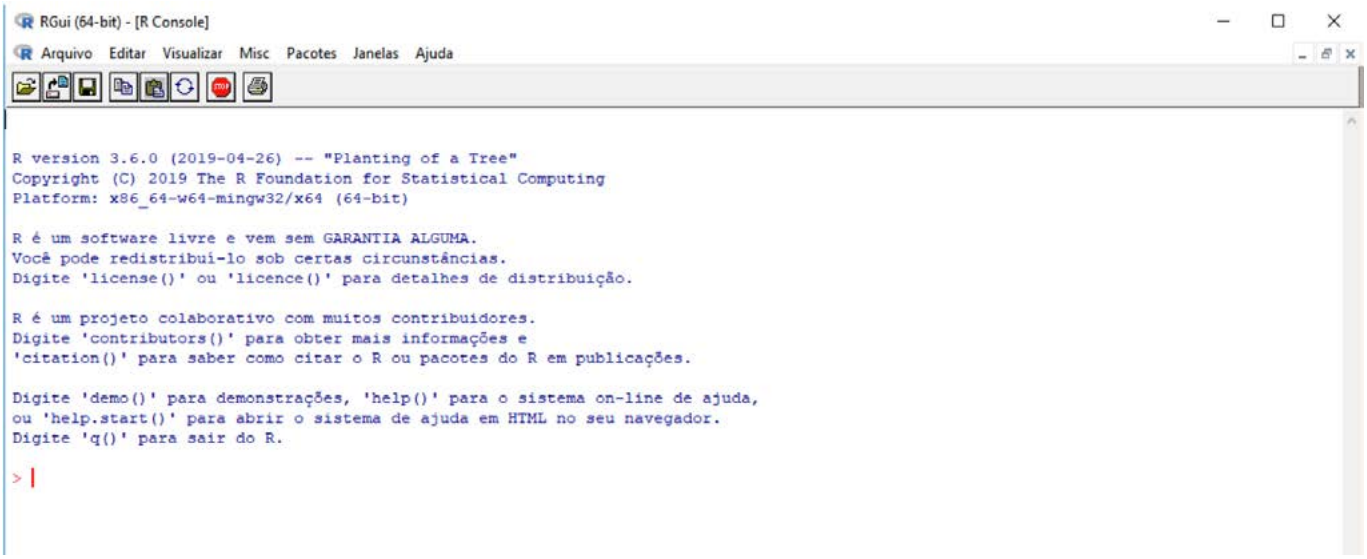
A linguagem R é semelhante à linguagem S desenvolvida no Bell *Laboratories* (antiga AT&T e, agora, Lucent *Technologies*) por John Chambers e colaboradores. Ela pode ser considerada como uma implementação diferenciada da linguagem S, pois existem diferenças significativas entre elas, no entanto, o código escrito para S não se alterou na mudança para a linguagem R.

O programa R é composto por vários pacotes, os quais também contêm funções, sendo estas organizadas por linhas de comando. A linguagem de programação R, conforme definem Oliveira, Guerra e McDonnell (2018, p. 10), pode ser entendida como "um conjunto de pacotes e ferramentas estatísticas, munido de funções que facilitam sua utilização, desde a criação de simples rotinas até análises de dados complexas".

A linguagem de programação R pode dialogar com outras linguagens como C, C++, FORTAN, dentre outras. De certa forma, tal característica é uma vantagem, pois dados armazenados em diferentes sistemas podem estar em formatos diversos, tal como os *Big Data*.

A interface do programa R é bastante simples e baseia-se, basicamente, em linhas de comando. No entanto, para executar suas linhas de programação, recomenda-se a utilização de um editor de texto ou de uma IDE (*Integrated Development Environment*), ou seja, uma interface amigável para facilitar a sua utilização. A Figura 1 mostra a interface básica do programa R.

Figura 1 – Tela de interface do programa R



Fonte: elaborada pelo autor com o programa R.

"Inúmeras organizações atualmente têm a necessidade de realizar o processamento e análise de uma grande quantidade de dados em tempo computacional hábil" (HÖLBIG; MAZZONETTO; PAVAN, 2017, p. 27). Por isso, muitas empresas, ao longo dos anos, passaram a utilizar o programa R como um de seus recursos computacionais por conta de suas características, conforme algumas citadas anteriormente. É possível mencionar algumas dessas empresas, conforme apresentado no Quadro 1.

Quadro 1 – Empresas que utilizam programação R em suas atividades

Empresa	Finalidade do uso do R
Google	Para descobrir o retorno sobre o investimento em publicidade.
Facebook	Análise de atualizações de status do Facebook.
Microsoft	Serviço *Xbox*, estatístico com estrutura do Azure *Machine Learning*.
Ford Motor	Para análise estatística e decisão orientada por dados.
John Deere	Modelagem de séries temporais e análise geoespacial.
Lloyds	Desenvolvimento de gráficos de movimento para fornecer análises para investidores.

Fonte: Shinde, Oza e Kamat (2017).

A linguagem de programação R, por ser baseada em linhas de comando, pode, em princípio, não ser muito atrativa. No entanto, existem interfaces mais amigáveis como dito anteriormente, as quais podem ajudar na superação de uma dificuldade inicial. Outra vantagem existente no seu uso trata-se da vasta documentação disponível para consulta na internet e em outros meios.

A IDE mais popular para o programa R é o *RStudio*, a qual é disponibilizada em diversas versões para os vários sistemas operacionais existentes. Além de existirem versões gratuitas e pagas que se diferenciam nas suas funcionalidades que facilitam a utilização de programação R.

Segundo Hölbig, Mazzonetto e Pavan (2017, p. 28) "a análise e a visualização de dados são, provavelmente, um dos maiores pontos fortes da linguagem R". Os autores afirmam ainda que a concepção inicial de sua criação foi exatamente essa. No entanto, com o passar do tempo, muitos pacotes foram criados agregando outras funções para o programa, expandindo as suas funcionalidades.

Considerando a gama de funcionalidades que a linguagem de programação R possui, "a visualização de dados e interação *on-line* com os usuários na *web* por meio do R é outra característica muito utilizada pelos usuários do R" (HÖLBIG; MAZZONETTO; PAVAN, 2017, p. 29). Os autores afirmam ainda que tal aspecto vale tanto para simples usuários individuais do programa como para empresas, universidades etc.

Existe um pacote chamado *Shiny*, desenvolvido pela mesma empresa, que desenvolveu a IDE *RStudio*, que é, também, de código aberto (*open source*), e que disponibiliza para o usuário um *framework web* para que sejam construídas aplicações com a linguagem R e, também, possibilitar a integração desta com tecnologias HTML, JavaScript e CSS (*Cascading Style Sheets* – Folha de estilo em cascata) (HÖLBIG; MAZZONETTO; PAVAN, 2017).

Diferentemente de outras linguagens de programação, a linguagem R "possui uma otimização de processos durante a própria escrita da aplicação" (HÖLBIG; MAZZONETTO; PAVAN, 2017, p. 30). Isto significa que, enquanto os *scripts* são elaborados, a otimização deles ocorre de forma simultânea, a qual pode ser feita a partir de vetorização de funções, como por exemplo, com o uso da família de funções "*apply*", com a pré-alocação de memória, com o uso de estruturas de dados mais simplificados, utilização de funções internas, utilização de código compilado.

Ainda falando de otimização da linguagem R em processos, estudiosos do assunto afirmam que esta é uma questão crítica, pois em muitas situações os programas podem demorar um longo período de tempo para realizar a execução de algumas tarefas (HÖLBIG; MAZZONETTO; PAVAN, 2017). Por isso, dá-se à otimização uma ampla atenção com a intenção de produzir *scripts* apropriados, que possam otimizar o desempenho computacional do processo.

Em muitas situações, não é possível obter otimização apropriada para um trabalho desenvolvido em um processo. Em um caso como este, não existe alternativa melhor do que tentar implementar o trabalho, ou parte dele, em outra linguagem de programação, para uma posterior integração com a linguagem R (HÖLBIG; MAZZONETTO; PAVAN, 2017).

É possível, também, utilizar programação em linguagem R em ambientes paralelos que trabalham em *clusters*, *grids*, processadores multicore e GPU (*Graphics Processing Unit* – Unidade de processamento gráfico). Existem diversos pacotes elaborados para ambientes com arquitetura paralela, os quais são descritos com mais detalhes no Quadro 2 para *clusters* de computadores.

Quadro 2 – Principais pacotes para uso do R em *clusters* de computadores

Pacote	Dependência	Descrição	Plataforma
Rmpi	R (≥ 2.15.1)	Uma interface para MPI (*Message Passing Interface*).	Linux, Windows.
snow	R (≥ 2.13.1), snow	Suporte para simples computações paralelas em R.	Linux, Windows, OS X Mavericks.
snowfall	R (≥ 2.10)	Interface para facilitar o desenvolvimento de programas paralelos em R.	Linux, Windows, OS X Mavericks.
foreach	R (≥ 2.5.0)	Suporte para o uso do construtor de *looping foreach* e para a exceção de laços em paralelo.	Linux, Windows, OS X Mavericks.
doMC	R ((≥ 2.14.0), foreach ((≥ 1.2.0), iterators ((≥ 1.0.0), parallel	Fornece um *backend* paralelo para a função %dopar% usando a funcionalidade multicore do pacote parallel.	Linux e OS X Mavericks.
doSNOW	R (≥ 2.5.0), foreach (≥ 1.2.0), iterators (≥ 1.0.0), snow (≥ 0.3.0), utils	Fornece um *backend* paralelo para a função %dopar% usando o pacote snow.	Linux, Windows, OS X Mavericks.
doMPI	R (≥ 2.14.0), foreach (≥ 1.3.0), iterators (≥ 1.0.0), Rmpi (≥ 0.5-7)	Fornece um *backend* paralelo para a função %dopar% usando o pacote Rmpi.	Linux e Windows.
Rdsm	bigmemory (≥ 4.0.0), parallel	Fornece um ambiente de programação com suporte a *threads* em R. Possibilita trabalhar de maneira eficiente com matrizes de grande porte.	Linux e OS X Mavericks.

Fonte: Hölbig, Mazzonetto e Pavan (2017).

Para sistemas *multicore*, ou seja, processadores que possuem mais de um núcleo, também, existe um conjunto de pacotes em linguagem R criados para serem executados nesses ambientes. O Quadro 3 apresenta os pacotes criados para esses ambientes.

Quadro 3 – Principais pacotes para uso do R em processadores *multicore*

Pacote	Dependência	Descrição	Plataforma
fork	nenhuma	Funções em R para manipulação de múltiplos processos.	Unix, Windows, OS X Mavericks
multicore	R (≥ 2.14.0)	Código para processamento paralelo do R em máquinas com múltiplos cores ou CPU's.	Linux, Windows, OS X Mavericks

Fonte: Hölbig, Mazzonetto e Pavan (2017).

Em ambientes GPU's, também existe uma crescente produção de pacotes em linguagem R, principalmente para uso em GPU's da NVIDIA®. O Quadro 4 apresenta os principais pacotes em linguagem R.

Quadro 4 – Principais pacotes para uso do R em processadores *multicore*

Pacote	Dependência	Descrição	Plataforma
gputools	R (≥ 3.1.2)	Disponibiliza uma interface em R para um completo conjunto de funções que são implementadas utilizando o NVIDIA CUDA *toolkit*.	Linux
OpenCL	R (≥ 2.0.0)	Disponibiliza uma interface em R para o OpenCL *toolkit*.	Linux
HiPLARM	R (≥ 2.14), Matrix, methods	Pacote para computação de alto desempenho que oferece suporte ao pacote Matrix do R.	Unix
gmatrix	methods, stats	Pacote para uso do R em GPU's da NVIDIA.	Unix
gpuR	R (≥ 3.0.2), methods, utils	Fornece acesso às funcionalidades das GPU's aos objetos do R.	Linux

Fonte: Hölbig, Mazzonetto e Pavan (2017).

Para arquiteturas paralelas em *grid*, Hölbig, Mazzonetto e Pavan (2017, p. 33) afirmam que "não há um desenvolvimento significativo de pacotes para este tipo de ambiente". Os autores descrevem, também, que o repositório (CRAN) do R possui dois pacotes disponíveis, denominados de "*multiR*" e "*biocep-distrib*".

Um exemplo de aplicação em ambientes com arquitetura paralela, Hölbig, Mazzonetto e Pavan (2017) apresentam rotinas em linguagem de programação R com o uso de alguns pacotes para realizar cálculo sequencial da soma entre duas matrizes de 2000 elementos cada uma (linha 12 da programação). O Quadro 5 apresenta os comandos utilizados pelos autores. O programa apresenta, em comentários, os tempos de execução. A soma é construída com um objeto "*array*", que é vetorizado pelo ambiente R.

Quadro 5 – Programa em linguagem R com uso de funções otimizadas

```
1  ordem = 2000
2  A = array (0, dim = c(ordem, ordem))
3  B = array (0, dim = c(ordem, ordem))
4  S1 = array (0, dim = c(ordem, ordem))
5
6  for(i in 1:ordem)
7    for(j in 1:ordem)
8    {
9      A[i,j] = round(runif(1)*10);
10     B[i,j] = round(runif(1)*10);
11   }
12 system.time(S<-A+B)
13 # usuario sistema decorrido
14 #    0.02    0.00       0.01
15
16 system.time(
17   for(i in 1:ordem)
18     for(j in 1:ordem)
19     {
20       S1[i,j]<-A[i,j] + B[i,j];
21     })
22 # usuario sistema decorrido
23 #    9.39    0.03       9.43
24
25 system.time(X<-sum(A))
26 # usuario sistema decorrido
27 #    0.02    0.00       0.01
28
29 X<-0
30 system.time(
31   for(i in 1:ordem)
32     for(j in 1:ordem)
33     {
34       X<-A[i,j]+X;
35     })
36 # usuario sistema decorrido
37 #    3.11    0.00       3.13
```

Fonte: Hölbig, Mazzonetto e Pavan (2017).

Depois de executados os comandos e obtidos os tempos de execução, os autores analisaram que se a operação da soma de matrizes tivesse sido executada de maneira tradicional, ou seja, por laços de repetição

(linhas 17 a 21), o tempo de execução seria de 9,43s. O segundo laço realiza a soma dos elementos de uma matriz (linha 25) fazendo uso de uma função vetorial "*sum*" com o tempo de execução de 0,01s. A mesma operação é feita com o uso de dois laços de repetição (linhas 31 a 35), com tempo de execução observado de 3,13s. Os resultados demonstram a importância de se utilizar funções vetorizáveis, no caso do exemplo, função "*sum*", que é um objeto do tipo "*array*".

Um outro exemplo é apresentado onde são utilizados alguns pacotes elaborados para sistemas paralelos. O Quadro 6 apresenta as linhas de comando em linguagem R.

Quadro 6 – Programa em linguagem R com uso dos pacotes foreach e *doSNOW*

```
1  library(foreach)
2  require(doSNOW)
3  cl<-makeCluster(4) # numero de cores
4  registerDoSNOW(cl)
5
6  # cria uma funcao para rodar em cada iteracao do loop
7  check<-function(n) {
8    for(i in 1:1000)
9    {
10     sme<- matrix(rnorm(100),10,10)
11     solve(sme)
12   }
13 }
14 times <- 100 # numero de vezes para rodar o loop
15 system.time(x<-foreach(j=1:times) %dopar% check(j))
16 # usuario sistema decorrido
17 #    0.08    0.02      1.84
18
19 system.time(for(j in 1:times) x <- check(j))
20 # usuario sistema decorrido
21 #    3.80    0.00      3.82
22 stopCluster(cl)
```

Fonte: Hölbig, Mazzonetto e Pavan (2017).

O programa apresentado no Quadro 6 realiza a chamada da função do usuário "*check*" em 100 vezes. É possível observar o tempo utilizado para a execução de 1,84s (linha 17) obtido com a utilização do laço "*foreach*" e da opção "%dopar%" (linha 15), que foram executadas em processamento paralelo com o uso de quatro processadores do computador (linha 3). Com a utilização do laço "*for*" simples (linha 19), o tempo de execução aumentou para 3,82s (execução sequencial).

Voltando a falar de *Big Data*, em se tratando de linguagem R, Hölbig, Mazzonetto e Pavan (2017) declaram que há uma dificuldade encontrada nesse ambiente de programação. Os autores afirmam que tal dificuldade está ligada com a necessidade de manter os dados na memória no ambiente R. "Mesmo para os computadores modernos com grande capacidade de armazenamento, isto pode se apresentar como um desafio significativo" (HÖLBIG; MAZZONETTO; PAVAN, 2017, p. 34).

PARA SABER MAIS

Existem ferramentas em linguagem R com aplicabilidade mais flexíveis e que podem ser utilizadas em *clusters* e/ou processadores *multicore*. Dentre as existentes, podem ser citadas a *RHadoop* e a *parallel*.

Os principais pacotes em linguagem R capazes de manipular grandes quantidades de dados e fazer gerenciamento do seu uso em memória são denominados "*RevoScaleR*", "*parallel*" e "*RHadoop*". Mais detalhes podem ser encontrados em Hölbig, Mazzonetto e Pavan (2017).

3. Conceitos fundamentais de *Data Lake*

No ano de 2010 foi apresentado um novo conceito, o conceito de *Data Lake* ou *Data Hubs*, o qual foi introduzido por James Dixon. Este conceito inicialmente foi menosprezado, por ter sido entendido como um rótulo de marketing para um produto que fosse compatível com o *Hadoop* (MILOSLAVSKAYA; TOLSTOY, 2016).

Data Lake, em sua essência, é uma estratégia de armazenamento de dados semelhante a um *Data Warehouse*. "O *Data Lake* pode armazenar dados estruturados e não estruturados em seu formato bruto e são projetados para o consumo de dados, apoiando a descoberta de novas perguntas" (DATAPREV, 2019, p. 11).

Os *Data Lake*, geralmente, incluem um banco de dados semântico, um modelo conceitual que utiliza os mesmos padrões e tecnologias usados para criar *hiperlinks* da internet e adicionar uma camada de contexto sobre os dados, o que define o significado dos dados e suas inter-relações com outros dados. As estratégias de *Data Lake* podem combinar abordagens de banco de dados SQL e NoSQL e processamento analítico on-line (OLAP) e recursos de processamento de transações on-line (OLTP) (MILOSLAVSKAYA; TOLSTOY, 2016).

ASSIMILE

No mundo dinâmico atual, os dados das empresas estão crescendo muito rápido. Como o fluxo de dados de redes é muito grande, tornou-se vital para as empresas identificar quais dados são sensíveis ao tempo e devem agir imediatamente e vice-versa, quais dados podem ficar em um banco de dados ou *data lake* até que haja uma razão para explorá-los.

Ao contrário de um *Data Warehouse* hierárquico com armazenamento de arquivos ou pastas, o *Data Lake* utiliza uma arquitetura simples, em que cada elemento de dados tem um identificador exclusivo e um conjunto de *tags* de metadados estendidas. Ele não requer um esquema rígido ou manipulação de dados de todas as formas e tamanhos, mas requer manter a ordem de chegada dos dados. Pode ser imaginado como um grande conjunto de dados que traz registros históricos acumulados e novos (estruturados, não estruturados e semiestruturados) em tempo quase real em um único local, no qual o esquema e os requisitos de dados não são definidos até que sejam consultados (MILOSLAVSKAYA; TOLSTOY, 2016).

Se necessário, o *Data Lake* pode ser dividido em três camadas separadas: uma para dados brutos, outra para conjuntos de dados alimentados diariamente e outro para informações de terceiros. Outra abordagem possível é dividir o *Data Lake* em três partições de acordo com sua vida útil: dados com menos de seis meses; dados mais antigos, mas ainda ativos, e dados arquivados não mais usados, mas que precisam ser retidos (esses dados obsoletos podem ser removidos para outras mídias) (MILOSLAVSKAYA; TOLSTOY, 2016).

Um "*Data Lake* serve como um local econômico para realizar análises preliminares dos dados, enquanto a estruturação de dados flexível e orientada a tarefas é implementada apenas onde e para o que for necessário" (STEIN, 2014 *apud* MILOSLAVSKAYA; TOLSTOY, 2016, p. 303, tradução nossa).

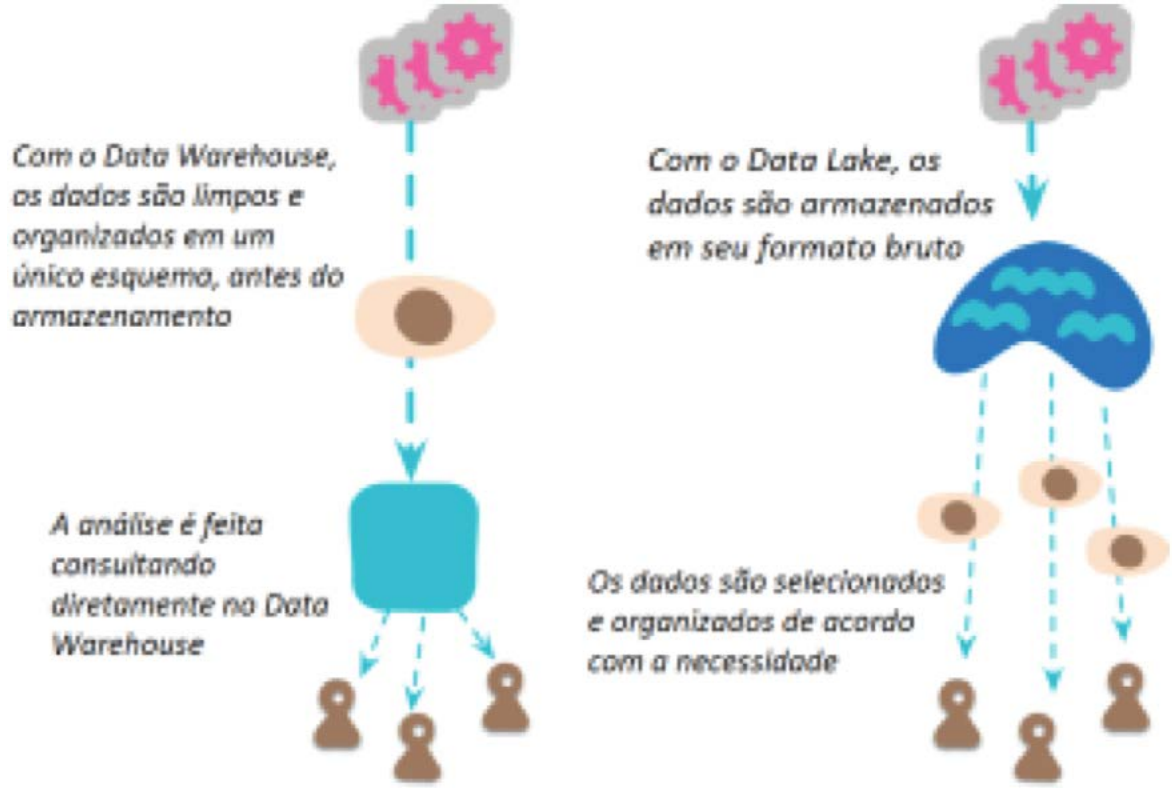
A característica distinta de um *Data Lake* é que ele atrai mais atenção dos campos de negócios do que dos campos de pesquisa acadêmica. O conceito de *Data Lake* é um conceito relativamente novo, mesmo para o domínio de *Big Data*. Portanto, suas definições e características, arquitetura, criação (implementação) e uso são muito mais prevalentes em estudos na web e blogs do que em estudos acadêmicos (KHINE; WANG, 2018).

Matos (2015, n.p.) afirma que

> Quando se ouve falar sobre um ponto único para reunir todos os dados que uma organização deseja analisar, imediatamente se imagina a noção de *Data Warehouse* e *Data Mart*. Mas há uma distinção fundamental entre *Data Lake* e *Data Warehouse*. O *Data Lake* armazena dados brutos, sob qualquer forma do jeito que foram coletados na fonte de dados. Não há suposições sobre o esquema dos dados e cada fonte de dados pode usar qualquer esquema. Cabe aqueles que vão analisar os dados, dar sentido a esses dados para o propósito ao qual a análise se destina.

O autor apresenta uma ilustração para mostrar as principais diferenças entre um *Data Warehouse* e um *Data Lake*. A Figura 2 replica a ilustração de Matos (2015).

Figura 2 – Principais diferenças entre *Data Warehouse* e *Data Lake*.



Fonte: Matos (2015).

A utilização de linguagem R para tratamento de dados armazenados em *Data Lake* pode ser realizada com o *software Spark*. Para sua utilização em ambiente R é necessário utilizar o pacote "*SparkR*". Existe uma interface para R que utiliza *Apache Spark* para manipulação de grandes volumes de dados. Para a sua utilização, basta acessar o repositório do R (CRAN) e fazer o *download* e instalação. Ela funciona como se fosse um pacote qualquer do ambiente R, o que significa que, após a sua instalação é necessário ativar o pacote com o comando "*require*( )" ou "*library*( )". O Quadro 7 apresenta comandos de instalação, ativação do pacote e de base de dados em ambiente R.

Quadro 7 – Exemplo de programação R com Spark.

```
# Para verificar a versão do Java que está instalada.
system("java -version")

# Instalação do pacote sparklyr
install.packages("sparklyr")
library(sparklyr)

# Para verificar a versão do sparklyr que foi instalada
packageVersion("sparklyr")

# Verifica as versões existentes do Spark
spark_available_versions()

# Instala a versão mais recente do Spark
spark_install("2.4")

# Verifica a versão que foi instalada
spark_installed_versions()

# Um cluster local é útil para começar, testar o código e

# solucionar problemas com facilidade.
# para conectar execute a linha abaixo.
sc <- spark_connect(master = "local", version = "2.4")

# Agora que estamos conectados, podemos executar um
# simples comando.com uma base de dados do R.
# conectar o banco de dados com o cluster

cars <- copy_to(sc, mtcars)

# faz a visualização do banco de dados
cars

# Ativa a biblioteca DBI para utilizar SQL
library(DBI)

# conta o número de registros na base de dados
dbGetQuery(sc, "SELECT count(*) FROM mtcars")
```

Fonte: adaptado de Ruiz, Kuo, Luraschi (2019).

Para que uma empresa possa se manter competitiva no mercado, é necessário estar atualizada com ferramental tecnológico atualizado. Para tanto, é necessário investir em equipamentos, sistemas e material humano qualificado e preparado para o bom manuseio das ferramentas.

A linguagem R é uma das ferramentas utilizadas para lidar com grandes volumes de dados, associada a outros programas computacionais, ela auxilia na tomada de decisões a partir do conhecimento que os dados podem trazer para quem os manipula. É uma corrida contínua para estar na frente em todos os quesitos.

Existe um amplo material disponível na internet sobre o assunto. Este texto, de longe, esgota o assunto. Portanto, não deixe de buscar outras fontes de pesquisa para agregar o conhecimento aqui trazido, pois, será importante para a sua formação profissional.

TEORIA EM PRÁTICA

Considere que a empresa que você trabalha está passando por uma série de reformas, inclusive, na área de tecnologia de informação (TI). Você é usuário de alguns sistemas da empresa e, por conta dessas mudanças, muitos sistemas de dados serão substituídos por sistemas mais modernos e mais atuais. Para que você possa continuar utilizando os sistemas de forma satisfatória, precisará realizar alguns treinamentos relacionados ao manuseio dos novos sistemas existentes. Para isso, você precisa priorizar os treinamentos em sistemas que mais utiliza e que são prioritários para o bom andamento do seu trabalho. Em um desses sistemas, você precisará utilizar a linguagem R para ter acesso aos dados. Como você daria início ao estudo dessa linguagem de programação? Supondo que você tem pouco conhecimento sobre o assunto e precisa utilizar sistemas que lidam com grandes massas de dados.

VERIFICAÇÃO DE LEITURA

1. Existe um pacote elaborado em linguagem R o qual disponibiliza para o usuário um *framework* web para que sejam construídas aplicações em R. Qual o nome deste pacote?

   Assinale a alternativa CORRETA.

   a. RStudio.

   b. RBase.

   c. Shiny.

   d. RHadoop.

   e. HTML.

2. O conceito de *Data Lake* foi apresentado para a comunidade científica em qual ano?

   Assinale a alternativa CORRETA.

   a. 2001.

   b. 2010.

   c. 2000.

   d. 2006.

   e. 2009.

3. Um *Data Lake*, se necessário, pode ser dividido em uma quantidade de camadas separadas. Em quantas camadas pode-se dividir um *Data Lake*?

   Assinale a alternativa CORRETA.

   a. Duas.

   b. Quatro.

   c. Cinco.

   d. Três.

   e. Sete.

Referências bibliográficas

BELETI JUNIOR, C.R. **Paralelização de aplicações na plataforma R**. 2013. 95f. Dissertação (Mestrado em Ciência da Computação) – Departamento de Informática, Universidade Estadual de Maringá, Maringá, 2013. Disponível em: http://www.din.uem.br/~mestrado/diss/2013/beletijr.pdf. Acesso em: 29 nov. 2019.

DATAPREV. **Administração de dados: padrões para construção de modelos de dados**. [s.l.]. 2019. Disponível em: https://portal.dataprev.gov.br/sites/default/files/arquivos/instrumentos_normativos/n_ad_001_05.pdf. Acesso em: 29 nov. 2019.

HÖLBIG, C.A.; MAZZONETTO, A.; PAVAN, WILLINGTHON. Computação paralela com a linguagem R: técnicas, ferramentas e aplicações. Minicurso. **17ª Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul**. Anais, p. 25-42. Ijuí: RS, 2017. Disponível em: http://www.lbd.dcc.ufmg.br/colecoes/erad/2017/003.pdf. Acesso: 29 nov. 2019.

KHINE, P.; WANG, Z. Data lake: a new ideology in big data era. **ITM Web of Conferences**. 17, 03025 (2018). DOI: 17. 03025. 10.1051/itmconf/20181703025. Acesso em: 29 nov. 2019.

MATOS, D. **Data Lake, a fonte do Big Data**. [s.l.]. 2015. Disponível em: http://www.cienciaedados.com/data-lake-a-fonte-do-big-data/. Acesso em: 29 nov. 2019.

MILOSLAVSKAYA, N.; TOLSTOY, A. Big data, fast data and data lake concepts. **Procedia Engineering**, 88(2016), 300–305. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050916316957. Acesso em: 29 nov. 2019.

OLIVEIRA, P.F.; GUERRA, S.; McDONNELL, R. **Ciência de dados com R:** introdução. Brasília: IBPAD. 2018. Disponível em: https://www.ibpad.com.br/o-que-fazemos/publicacoes/introducao-ciencia-de-dados-com-r#download. Acesso em: 17 jul. 2019.

RUIZ, E.; KUO, K.; LURASCHI, J. **Mastering spark with R**. O'Reilly Media, Inc. 2019. Disponível em: https://learning.oreilly.com/library/view/mastering-spark-with/9781492046363/. Acesso em: 29 nov. 2019.

SIQUEIRA, Arminda. L., TIBÚRCIO, Jacqueline. D. **Estatística na área da saúde:** conceitos, metodologia, aplicações e prática computacional. Belo Horizonte: Coopmed, 2011. 520 p.

SHINDE, P.P.; OZA, K.S.; KAMAT, R.K. Big Data predictive analysis: using R analytical tools. 2017. **International Conference on I-SMAC** (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, 2017, pp. 839-842. doi: 10.1109/I-SMAC.2017.8058297. Disponível em: https://ieeexplore.ieee.org/document/8058297. Acesso em: 19 nov. 2019.

Gabarito

Questão 1 – Resposta C

O pacote *Shiny* foi elaborado em linguagem R e disponibilizado em *framework web* para o usuário com o intuito de permitir a construção de aplicações em R.

Questão 2 – Resposta B

O conceito de *Data Lake* foi apresentado para a comunidade científica no ano de 2010.

Questão 3 – Resposta D

Um *Data Lake* pode ser dividido em até três camadas.

Unificando e integrando os dados. Concluindo o projeto e exibindo os *Dashboards* em ferramentas OLAP

Autor: Marcelo Tavares de Lima

Objetivos

- Descrever os principais tipos de *dashboards*.
- Apresentar os principais conceitos de OLAP (Processo Analítico On-line).
- Mostrar aplicações de integração de dados com ferramentas OLAP e *Dashboard*.

1. Introdução

Todo trabalho realizado para ser conhecido precisa ser apresentado por meio de alguma ferramenta de comunicação. É uma forma de divulgação do esforço realizado para a produção de subsídios para a tomada de decisão. É, também, uma forma de mostrar aos interessados resultados que poderão ser importantes e que podem impactar de alguma forma o fluxo de trabalho ou de suas rotinas.

Dentre as muitas ferramentas de comunicação ou de divulgação de resultados estão os *dashboards*. Eles são painéis de visualização compostos por resultados obtidos por meio do tratamento analítico dispensado aos dados utilizados em alguma análise.

Os dados utilizados para a produção de *dashboards*, em geral, para um bom uso, são armazenados em ambientes construídos com ferramentas OLAP (Processo analítico on-line), cujo conceito é muito maior que um ambiente de armazenamento de dados, mas, também, é um ambiente de tratamento analítico e de consulta a dados armazenados em um *data warehouse* ou *data lake*.

Este texto apresenta os principais conceitos e tipos de dashboards, processos e procedimentos integrados de dados com o uso de ferramentas OLAP. Que você possa ter um bom momento de leitura e que ela possa agregar conhecimento para a sua formação profissional e acadêmica.

2. Introdução a *dashboard*

Não há um consenso na definição formal de *dashboard*. No entanto, sabe-se que é uma ferramenta de visualização de dados que apresenta resultados de forma rápida e dinâmica e que é utilizada para apresentar resultados importantes para o acompanhamento de alguma atividade corporativa, seja de negócios ou acadêmica.

Wexler, Shafer e Cotgreane (2017, n.p., tradução nossa) definem *dashboard* como "um painel ou uma visualização de dados utilizado para monitorar condições e/ou facilitar a comunicação". A definição de *dashboard* apresentada é um conceito, de certa forma, bastante ampla e, por isso, ao longo deste texto, exemplos práticos de aplicação do uso de *dashboard* serão apresentados.

Um dos principais objetivos de um *dashboard* é a divulgação rápida e clara de resultados por meio de indicadores e métricas diversas, que possam ser compreendidos por todos os envolvidos em um processo, desde estagiários até os executivos. Métricas, segundo Malik (2005, p. 13, tradução nossa), "são medidas de avaliação de performance dentro de um contexto temporal, geográfico e de agregação".

As informações que devem constar em um *dashboard* estão diretamente relacionadas com os objetivos de uma empresa e, portanto, variam de forma diversa. Portanto, para identificar as informações que devem constar, é necessária a construção de perguntas essenciais que possam direcionar para o que, de fato, é informação importante para ser divulgada e acompanhada periodicamente, para uma correta tomada de decisões associada aos negócios.

As perguntas essenciais e certas para serem acompanhadas via *dashboard* são identificadas quando se conhece as necessidades da empresa de maneira clara e eficiente. Com esta informação, é possível, também, definir as métricas e os indicadores que ajudarão a acompanhar as respostas e os objetivos dos negócios.

Em um instante inicial, pode ser que a(s) pergunta(s) essencial(is) não seja(m) identificada(s). No entanto, não se pode deixar de elaborá-las, mesmo que em um momento posterior sejam identificadas como inapropriadas ou imperfeitas. A dinâmica dos negócios também faz com que as perguntas essenciais se modifiquem de tempos em tempos. É natural que adaptações sejam realizadas ao longo do tempo.

Especialistas no assunto classificam *dashboard* em vários tipos, a depender do tipo de informação que o compõe. De maneira geral, eles apresentam a saúde da empresa. Um *dashboard* que contém informações técnicas, como informações sobre a infraestrutura da empresa, auxilia na análise de desempenho e de disponibilidade de tecnologias diversas associadas aos processos da empresa.

Quando associado à gestão de negócios, um *dashboard* é composto por um conjunto de indicadores de performance geral da empresa em alguma área específica. Uma das muitas vantagens existentes com a utilização de um *dashboard* é a "libertação" dos relatórios abarrotados de tabelas de dados que tornam cansativo e de difícil interpretação os números contidos neles. Além de simplificar os resultados, um *dashboard* apresenta exatamente o dado que realmente interessa.

Qualquer tipo de dado pode ser manipulado para ser apresentado em *dashboard*. O importante é que sejam dados importantes para a gestão e tomada de decisões, como por exemplo, dados de estoque, produção por período de tempo, total de vendas de um período etc.

É comum que os *dashboards* sejam exibidos em grandes telas espalhadas pelos setores envolvidos na produção dos dados. Muitos especialistas da área denominam o processo de divulgação e transparência de dados via *dashboard* de "gestão à vista", pois um *dashboard* ideal deve conter somente uma página de visualização. O conceito de gestão à vista está diretamente relacionado com os tipos de métricas e indicadores que serão apresentados no *dashboard*, assim como os padrões ou mídias visuais escolhidas para a sua elaboração.

Nesse momento, em que se está colocando na prática o conceito de gestão à vista, é preciso ter muito cuidado com a seleção de informações para serem utilizadas na construção do *dashboard*. Não se pode selecionar informação demais e nem de menos. É preciso ponderar adequadamente para não poluir o painel de informações.

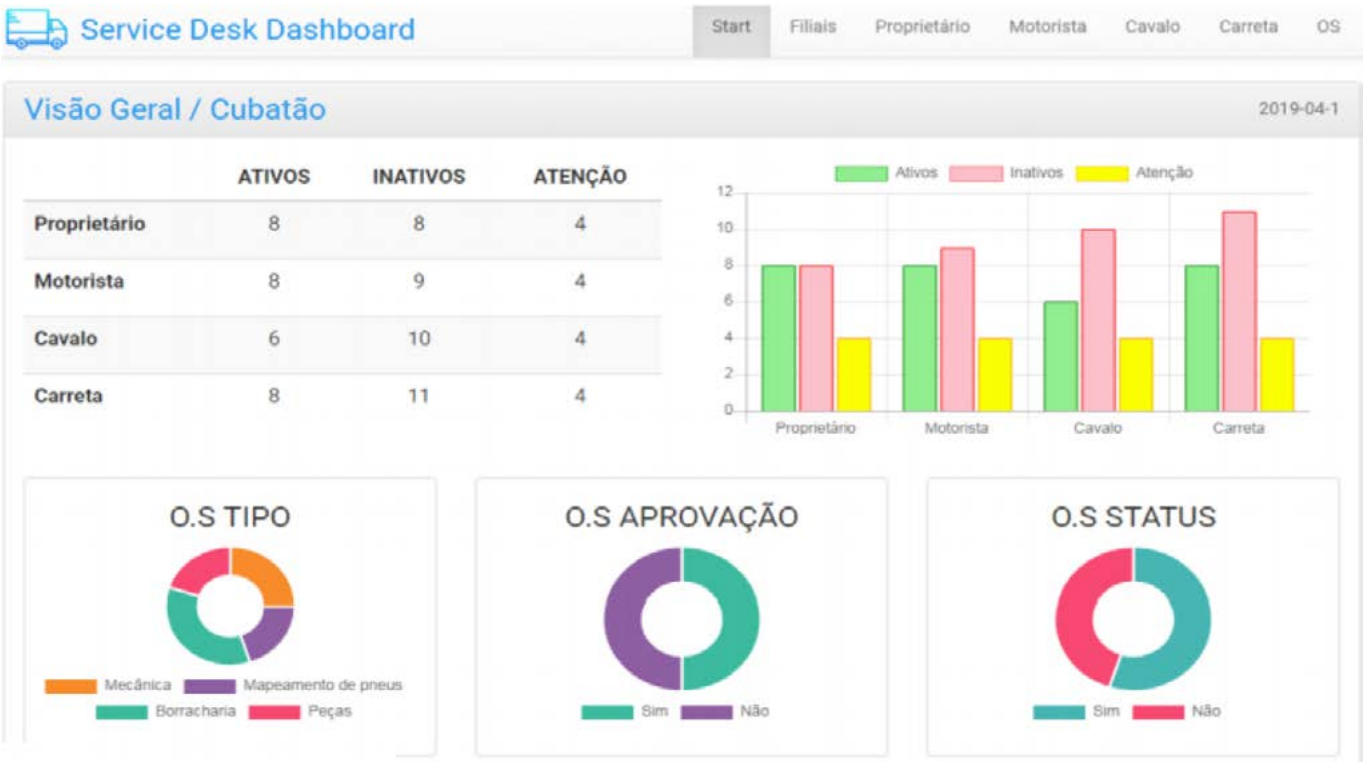
Kerzner (2017, p. 255, tradução nossa) afirma que "embora os *dashboards* sejam muito comuns em indústrias, a sua presença pode estar em diversos ambientes". Para exemplificar, um *dashboard* pode ser elaborado

para gerenciar o uso de leitos em um hospital. Podem ser instalados em museus, cassinos, consultórios médicos, dentre outros. Alguns fatos associados a *dashboard* são elencados a seguir, segundo Kerzner (2017):

- *Dashboards* são ferramentas de comunicação.
- *Dashboards* fornecem aos seus usuários o significado atual e futuro de uma informação.
- Quando elaborados apropriadamente, os dashboards fornecem informações de *Business Intelligence* (BI).
- Os *dashboards* podem ser considerados relatórios detalhados.

Existem diversos programas que elaboram *dashboards*, dentre os quais podem ser citados o MS Excel, versão 2007 em diante, R *Programming*, Python, Qlick Sense, dentre outros. É possível encontrar muitos manuais de elaboração de *dashboards* na internet e no menu de ajuda dos programas específicos para elaboração dessa ferramenta. A Figura 1 apresenta um exemplo de *dashboard*.

Figura 1 – Exemplo de *dashboard*



Fonte: Brito *et al*. (2019).

Não existe um consenso sobre os tipos de *dashboard* existentes, principalmente quanto à nomenclatura criada para a tipologia. No entanto, são aproximadamente semelhantes e, por isso, não há problemas quanto às diferenças existentes.

Neste texto serão considerados existentes três tipos de *dashboards*, os quais são denominados como operacional, tático e estratégico, que são os tipos mais gerais. Em termos estéticos pouco diferem entre si. A diferença, de fato, ocorre principalmente quanto ao público-alvo para o qual a ferramenta é elaborada.

Dashboards operacionais são os que possuem métricas que devem ser acompanhadas para um bom desenvolvimento de uma atividade operacional. São úteis para auxiliar os analistas a corrigir erros e falhas possíveis nos processos de trabalho, os quais poderão ser identificados com maior rapidez por meio de seu acompanhamento. São painéis com público-alvo, principalmente, os operadores envolvidos nos processos de trabalho. Portanto, é importante aplicar treinamento adequado em toda a equipe envolvida. Isso acelera o processo de tomada de decisão com as informações disponibilizadas.

Exemplos de um *dashboard* operacional podem ser considerados os painéis sobre indicadores de produção de uma indústria, painéis com indicadores de acompanhamento de entrega de mercadorias, para acompanhar os atrasos.

Dashboards táticos são painéis compostos por informações que conseguem permitir que os gestores direcionem recursos para que os objetivos previamente estabelecidos possam ser alcançados em médio prazo. Seu público-alvo principal são as gerências departamentais dos negócios de uma empresa.

Em termos de complexidade, os *dasboards* táticos são considerados em maior nível que os *dashboards* operacionais. Por isso, podem influenciar na tomada de decisão para mudanças operacionais de uma empresa.

Cada gestor de área pode identificar, por exemplo, gargalos que estejam atrapalhando o processo, e a partir dessa identificação realizar decisões operacionais.

Os *dashboards* estratégicos são endereçados à alta direção das empresas. Apresentam informações que permitem direcionar recursos para que os objetivos, previamente elaborados, sejam alcançados em longo prazo. Apresentam indicadores que permitem análises comparativas, seja por períodos, regiões ou outras unidades, com a intenção de avaliar a evolução dos trabalhos ou processos, enfim, servem para decisões estratégicas dos negócios.

Apesar de serem direcionados para a alta direção, vale a pena compartilhar, também, com os demais colaboradores da empresa. A intenção é promover maior engajamento nas atividades e aumentar o compromisso com a empresa. Portanto, é uma ferramenta apropriada para ser utilizada na comunicação interna das empresas.

3. Conceitos fundamentais de OLAP (Processo Analítico On-line)

As ferramentas de visualização de dados multidimensionais, também conhecidas como OLAP (*online analytical process*), são um tipo dentre muitos ferramentais disponíveis para a visualização adequada para se obter um bom subsídio na tomada de decisões. Em princípio, funcionam como interfaces de visualização de dados, no entanto, vão muito além disso.

O conceito OLAP não é um conceito novo, o seu desenvolvimento, segundo Vieira (2009), se deu no início de 1962, com a publicação de um livro denominado *A programming language*, de autoria de Kenneth Iverson, professor de matemática nascido no Canadá.

OLAP (*Online analytical process*), conhecido em português como processo analítico on-line ou sistema de informações multidimensionais, na prática, é a interface entre a grande massa de dados complexos armazenada em um banco de dados e o seu usuário.

A respeito de outro conceito divulgado sobre OLAP, Inmon (1999 *apud* Vieira, 2009, p. 16) afirma que "Olap é uma tecnologia de *software* que possibilita uma variedade de visualização das informações que antes era de uma coleção de dados referentes ao empreendimento". Por conta dessa ampla definição, segundo Thomsen (2002), pode-se falar em conceito OLAP, em linguagem OLAP e produtos OLAP.

Os conceitos OLAP incluem a noção ou ideia de múltiplas dimensões hierárquicas e podem ser usados para se pensar mais claramente sobre a estrutura dos dados analisados.

PARA SABER MAIS

Duas informações mudam de resultado quando se realiza combinações de suas dimensões: mudança nos eixos e em suas vizinhanças. A primeira coisa que muda é a forma de visualização dos dados visíveis. É possível apresentar resultados com maior rigor de detalhes ou menos rigor a partir dessas combinações. Uma coisa importante não muda durante o processo. Não importa quantas dimensões são combinadas e, independentemente da organização dos dados, elas fazem as mesmas declarações, reivindicações ou proposições sobre o mundo cuja verdade ou falsidade é uma função do mesmo critério.

As linguagens formais OLAP incluem linguagem de definição de dados (DDL), linguagem de manipulação de dados (DML), linguagem de representação de dados (DRL) e analisadores associados (e

compiladores opcionais), os quais podem ser usados para qualquer modelagem descritiva, seja transacional ou de suporte a produtos OLAP completos que precisam incluir um compilador e métodos de armazenamento e acesso de dados.

Linguagem de definição de dados (DDL) é a linguagem de computador utilizada para a definição de dados, ou seja, para criar estrutura de dados, remover estrutura de dados e realizar qualquer alteração em estrutura de dados. Linguagem de manipulação de dados é um conjunto de linguagens de computador utilizada para a realização de tarefas, como a recuperação, inclusão, remoção e modificação dos dados.

OLAP não se trata de um software específico, mas de um conceito, no qual vários programas computacionais podem ser enquadrados como OLAP, tanto pela capacidade de manipulação de dados quanto pela capacidade de apresentação visual. É uma ferramenta que permite ao usuário usufruir de uma análise multidimensional (MDA), ou seja, de poder manipular dados que estejam armazenados em dimensões diversas.

Como função básica de uma ferramenta OLAP, pode-se enumerar: (1) visualização muldimensional de dados e; (2) exploração de dados. A etapa do armazenamento de dados está mais vinculada a um outro conceito conhecido como *data warehouse*, que, em uma tradução literal, significa depósito de dados digitais. Na prática, *data warehouse* é o ambiente onde são armazenados dados digitalmente, no intuito de disponibilizá-los para produção de relatórios.

Os conceitos de OLAP e de *data warehouse* caminham juntos quando se aborda o assunto de manipulação e visualização de dados complexos. Caminham juntos porque não se pode falar de um sem abordar o outro.

O conceito de *data warehouse*, de forma simples e genérica, significa depósito de dados orientado por assunto, por período, dentre outros agrupamentos para dar suporte à tomada de decisões de empresas,

negócios etc., subsidiando o controle de processos e de determinações de padrões. Inmon (1997 *apud* VIEIRA, 2009, p. 13) afirma que *data warehouse* "é um conjunto de dados consolidados por assunto, não é volátil e está sempre em constante variação quanto ao tempo".

Um *data warehouse*, segundo Loh (2014), possui uma estrutura principal de dados (fatos) e, também, estruturas auxiliares (dimensões). Como exemplo, pode ser considerado um banco de dados com informações de vendas de uma empresa em que os dados sobre as vendas seriam os fatos (nota fiscal, códigos de produtos, código de clientes, data, valor pago, forma de pagamento, código da loja, código do vendedor etc.) e quaisquer outros dados relacionados a vendas seriam as dimensões. Loh (2014, n.p.) afirma ainda que "as dimensões normalmente possuem uma estrutura particular e separada". No exemplo das vendas, as dimensões podem ser consideradas dados sobre a descrição do produto vendido, o seu preço, dados dos clientes (nome, endereço, idade) etc.

Uma das vantagens de se utilizar dimensões é que "os dados podem ser vistos sob diferentes perspectivas". Ainda considerando o exemplo das vendas, é possível apresentar o total de vendas por produto, por cliente, por loja ou por vendedor. É possível apresentar os dados de maneira cruzada, ou seja, apresentar dados de vendas das lojas segundo os vendedores, neste caso, os dados estão apresentados em duas dimensões.

O formato multidimensional dos dados tem a vantagem de ser mais compacto e de ajudar em operações de análises diversas para a tomada de decisão. O exemplo apresentado das vendas com duas dimensões pode ser modificado acrescentando mais dimensões para apresentação dos resultados, como por exemplo, pode-se acrescentar a dimensão do tipo de produto vendido por cada vendedor em cada loja.

A organização e a apresentação de dados multidimensionais pode ser pensada como um formato de cubo, o que muitos autores e estudiosos do assunto denominam dados multidimensionais como dados cúbicos

(LOH, 2014). Dentre as vantagens conhecidas de dados cúbicos ou multidimensionais, podem ser citadas a aceleração de consultas e de análises de bancos de dados.

A noção de hipercubo, um cubo com mais de três dimensões, é fundamental para compreender e trabalhar com softwares de análise multidimensional de dados, que usam dados de planilhas eletrônicas e banco de dados baseado em tabelas. Estes softwares permitem a navegação, geração de relatórios e análises avançadas de dados. Rodrigues *et al*. (2012, p. 2) afirmam que "a multidimensionalidade se dá pelo fato de que os dados podem ser visualizados em diversas faces, causando uma ideia de cubo. Cada uma das faces apresenta uma significação, delimitando o assunto que se deseja analisar".

É sabido que a civilização está vivendo em uma era de grandes volumes de informações, em que o volume pode ser medido em exabytes, ou em escalas, como bit, byte, kylobyte, megaybte, gigabyte, terabyte, petabyte, exabyte, zettabyte, yottabyte etc. (LOH, 2014).

ASSIMILE

Embora não exista uma exibição certa ou errada, existem algumas regras que se deve ter em mente ao analisar dados multidimensionais em tabelas. Aninhar dimensões entre linhas e colunas consome muitas telas e recursos relativos à colocação de dimensões nas páginas da tela.

Há uma infinidade de confirmações sobre o crescimento da produção de dados e, consequentemente, do aumento do volume de produção de informações. Loh (2014) apresenta uma lista dessas justificativas, a qual é replicada a seguir:

- O armazenamento de dados torna-se cada vez mais barato (discos rígidos e DVDs) ou mesmo de graça (serviços de hospedagem gratuito).
- Aumento da familiaridade com as tecnologias, o que tem como consequência maior geração e armazenamento de dados.
- Mais possibilidades de serviços para publicação e difusão de informações (blogs, twitter, e-mail etc.).

Somado a esta lista, pode-se dizer que o aumento contínuo da possibilidade de realização de análises cada vez mais complexas a partir do desenvolvimento de softwares com funções de inteligência artificial tem potencializado o aumento da produção de dados no mundo. Em contrapartida, a infinidade de possibilidades de análises exige que a escolha das informações relevantes para determinada tomada de decisão seja feita com um rigor cada vez maior, pois a escolha inadequada dos dados pode implicar em uma tomada de decisão equivocada e, consequentemente, trazer prejuízo para os negócios.

4. Integração de dados

O uso de ferramentas de armazenamento e de tratamento de dados permite que a integralização de dados e de informações relevantes para a tomada de decisão possa ser realizada de forma satisfatória em diversas situações. Para isso, é necessário o uso de ferramentas apropriadas para a realização das diversas etapas de trabalho de um processo de integração de dados.

No geral, os processos analíticos de tratamento de dados são realizados sobre bases de dados armazenadas em locais distintos e não sobre uma única base de dados transacional (OLTP), onde, em geral, são realizadas as operações diárias de um trabalho. Como justificativa para a execução de tratamentos analíticos em bases distintas daquelas utilizadas no dia a dia, a desoneração dos servidores ou qualquer outro trabalho que possa atrapalhar operações rotineiras pode ser usado como justificativa.

Uma solução que vem sendo utilizada cada vez mais pelas empresas é a produção de bases específicas para o tratamento analítico, ou seja, bases chamadas de OLAP (LOH, 2014). As bases OLAP estão relacionadas com arquiteturas de dados conhecidas como *data warehouse*, que são bases de dados centralizadas, compostas por dados copiados de outras bases de dados (LOH, 2014).

As bases de dados de um *data warehouse* possuem fins específicos. Por exemplo, existem bases de dados para serem acessadas em um nível operacional de uma empresa, onde é possível realizar aplicações transacionais como inclusão, exclusão, alteração e consultas simplificadas de registros e valores. Outras bases dentro do mesmo ambiente são utilizadas especificamente para análise, onde os dados são não voláteis, permitindo unicamente a inclusão e, com isso, prestando apoio a decisões estratégicas (LOH, 2014).

Em tempos de grande produção de dados, os conceitos de *business intelligence* (BI) são primordiais para a organização e a seleção apropriada dos dados para produção de informações que possam agregar e subsidiar a tomada de decisões adequada.

As ferramentas de BI auxiliam na busca por padrões, os quais auxiliam na estimação de eventos futuros e na sua possibilidade de ocorrência. Loh (2014, n.p.) afirma que “a identificação de padrões é parte da nossa vida. A descoberta de padrões iniciou há milhares de anos atrás”.

Loh (2014, n.p.) afirma que “BI tem a ver com descobrir conhecimento, para poder gerar inteligência e resolver problemas. [...] O objetivo final então é poder gerar conhecimento novo e útil”. É um processo cuja entrada trata-se de um banco de dados e a saída é um conjunto de conhecimentos.

Tem-se como etapa principal, segundo Loh (2014), a mineração ou análise de dados (*data mining*), na qual esta raramente é realizada sobre todos os dados, mas sim realizada com maior frequência sobre amostras representativas do conjunto total.

Para a obtenção de um produto final ou da integração total dos dados, é necessário cumprir uma série de etapas no trabalho da descoberta do conhecimento por meio dos dados. Inicialmente, os dados necessitam de uma limpeza e da realização de uma crítica severa para que possam se tornar um conjunto de dados confiável para a extração de classificação e de padrões apropriados. Em seguida, a etapa de análise deve ser executada. É nessa etapa que a amostra confiável é tratada e recebe os métodos analíticos apropriados para a obtenção de resultados, como identificação de padrões ou tendências.

Para Loh (2014, n.p.) "o processo da descoberta do conhecimento é iterativo e interativo". O autor explica que ser iterativo (ou cíclico) indica que, em muitas situações, o processo pode precisar ser executado diversas vezes, com amostras diferentes ou, até mesmo, com técnicas distintas. O uso do termo interativo, ainda segundo o autor, justifica-se devido à necessidade de intervenção humana ao longo de todo o processo.

O processo de integração de dados precisa satisfazer algumas condições para que possa fornecer resultados confiáveis e satisfatórios. Inicialmente, é necessário ter uma claridade do objetivo do trabalho ou processo de BI, o qual pode ser obtido de duas maneiras, reativa ou proativa. Para ser obtido de forma reativa, o objetivo deve ser muito bem definido e deve ser capaz de identificar ou monitorar indicadores quantitativos (LOH, 2014). Já no caso de o objetivo ser identificado de forma proativa, não há uma clareza no que se busca, pois está ligado diretamente com a exploração, uma busca por algo ainda não identificado (LOH, 2014).

Outra premissa ou condição para uma boa integração de dados está relacionada com a coleta apropriada dos dados. Para que os objetivos sejam alcançados faz-se necessário que os dados sejam coletados com qualidade e confiança. Não basta coletar uma quantidade grande de dados, pois quantidade nem sempre significa que sejam apropriados para os objetivos elaborados (LOH, 2014).

O formato dos dados para a execução do tratamento analítico é um item importante no processo de integração e unificação. Por exemplo, dados armazenados de forma numérica ou quantitativa são mais fáceis de serem analisados estatisticamente (LOH, 2014). No entanto, é possível aplicar tratamento analítico a dados não numéricos com o uso de técnicas de mineração de texto, por exemplo.

Por fim, o uso das técnicas apropriadas para o tratamento analítico é um passo extremamente fundamental no processo de integração e unificação de dados para a produção de resultados, no qual a descoberta do conhecimento é completada.

O processo de integração e de unificação de dados é um assunto amplo e complexo que não consegue ser esgotado em umas poucas páginas de texto como este. Portanto, sugere-se que o aluno busque mais informações e referências que abordem do tema.

O uso de ferramentas para divulgação de resultados como *dashboards* são eficientes em diversos contextos, mas precisam ser utilizadas com cuidado para não transmitirem informações inapropriadas. Para evitar tal situação, pode-se recorrer às arquiteturas de *data warehouse* e fazer uso de ferramentas OLAP para a seleção apropriada dos dados, assim como do seu correto tratamento analítico.

TEORIA EM PRÁTICA

Suponha que você trabalha na equipe de TI de uma grande empresa. Uma de suas atividades é avaliar os sistemas de armazenamento e disponibilização de dados para os usuários de diversos departamentos. Frequentemente sua equipe de trabalho avalia a possibilidade de realizar atualizações nos sistemas de gerenciamento de dados. Você foi informado que o departamento de compras

está com um sistema apresentando muitas falhas. Então, você leva o problema para a sua equipe e avaliam o problema. Descobrem que está havendo um problema de comunicação entre os sistemas de armazenamento e tratamento com o programa de geração de informativos e *dashboard*. Como você pretende encarar esse problema? O que vai sugerir? Pense sobre isso!

VERIFICAÇÃO DE LEITURA

1. *Dashboards* são ferramentas para apresentação de resultados obtidos com tratamento analítico aplicados em dados. Esta ferramenta também recebe outro nome. Qual o outro nome aplicado a um *dashboard*?

   Assinale a alternativa CORRETA.

   a. Gráfico.

   b. Tabela.

   c. Painel.

   d. Quadro.

   e. Desenho.

2. Muitos estudiosos de *dashboard* atribuem um termo ao processo de divulgação e transparência de dados via *dashboard*. Qual é este termo?

   Assinale a alternativa CORRETA.

   a. *Dashboard* operacional.

b. Gestão à vista.

c. Gestão de *dashboards*.

d. *Dashboard* ideal.

e. Elaboração de *dashboards*.

3. São *dashboards* que possuem métricas que devem ser acompanhadas para um bom desenvolvimento de atividades gerais. São úteis para auxiliar analistas a corrigir erros e falhas possíveis nos processos de trabalho. Estamos nos referindo a qual tipo de *dashboard*?

   Assinale a alternativa CORRETA.

   a. Técnicos.

   b. Específicos.

   c. Táticos.

   d. Operacionais.

   e. Estratégicos.

Referências bibliográficas

BRITO, M. F.; TAVARES, P. F. R.; MALTA, R. F.; MENDONÇA, R. H. Planejamento logístico: dashboard para apoio à tomada de decisão relacionada à escolha de frota – estudo de caso. **X FATECLOG**. Guarulhos: SP. 2019. Disponível em: http://fateclog.com.br/anais/2019/PLANEJAMENTO%20LOG%C3%8DSTICO%20DASHBOARD%20PARA%20APOIO%20A%20TOMADA%20DE%20DECIS%C3%83O%20RELACIONADA%20A%20ESCOLHA%20DE%20FROTA%20%E2%80%93%20Estudo%20de%20Caso..pdf. Acesso em: 7 dez. 2019.

KERZNER, H. **Project management metrics, KPIs, and dashboards:** a guide to measuring and monitoring project performance. 3. ed. New Jersey: Wiley, 2017.

LOH, S. **BI na era do big data para cientistas de dados:** indo além de cubos e dashboards na busca pelos porquês, explicações e padrões. Porto Alegre. Edição do Kindle. 2014. Não paginado.

MALIK, S. **Enterprise Dashboards:** design and best practices for IT. New York, NY, EUA: John Wiley & Sons, Inc, 2005.

RODRIGUES, C. H. M.; ALMEIDA, C. C. O.; ROCHA, E. D.; COSTA, E. J. A. OLAP: uma perspectiva estratégica de análise de dados. **Revista Clique**. v. 1, n. 1, ago. 2012.

THOMSEN, E. **OLAP solutions:** building multidimensional information systems. 2. ed. New York: John Wiley & Sons, Inc. 2002.

VIEIRA, E. **Tecnologia olap**. 2009. 38 f. Trabalho de conclusão de curso (Graduação em Ciência da Computação). Instituto Municipal do Ensino Superior de Assis, Assis, 2009. Disponível em: https://cepein.femanet.com.br/BDigital/arqTccs/0411150200.pdf. Acesso em: 22 jul. 2019.

WEXLER, S.; SHAFFER, J.; COTGREAVE, A. **The big book of dashboards:** visualizing your data using real-world business scenarios. New Jersey: John Wiley & Sons, Inc., 2017.

Gabarito

Questão 1 – Resposta C

Os *dashboards* também são conhecidos como painéis.

Questão 2 – Resposta B

O processo de divulgação e transparência de dados via *dashboards* é conhecido como “gestão à vista”.

Questão 3 – Resposta D

Os *dashboards* úteis para auxiliar analistas a corrigirem erros e falhas possíveis nos processos de trabalho são os *dashboards* operacionais.

Bons estudos!