colloquium exactarum

AUMENTANDO DESEMPENHO DE ALGORITMOS DE MINERAÇÃO DE DADOS UTILIZANDO A PLATAFORMA CUDA

INCREASING PERFORMANCE OF DATA MINING ALGORITHMS USING CUDA PLATFORM

Matheus Varela Ferreira, Francisco Assis da Silva, Leandro Luiz de Almeida, Danillo Roberto Pereira

Faculdade de Informática – FIPP, Universidade do Oeste Paulista – UNOESTE
e-mail: varela@unoeste.edu.br, chico{chico, llalmeida, danilopereira}@unoeste.br

RESUMO – Com a necessidade cada vez maior de tomar decisões em curto prazo, a indústria (farmacêutica, petroquímica, aeronáutica e etc.) vem buscando novas formas de reduzir o tempo do processo de mineração de dados para obtenção do conhecimento. Nos últimos anos, muitos recursos tecnológicos estão sendo utilizados para amenizar essa necessidade, um exemplo é o CUDA. O CUDA é uma plataforma que possibilita o uso de GPUs GeForce em conjunto com a CPUs para processamento de dados, reduzindo de forma significativa o tempo de processamento. Este trabalho propõe realizar uma análise comparativa do tempo de processamento entre duas versões de alguns algoritmos de mineração de dados (Apriori, AprioriAll, Naïve Bayes e K-Means), uma executando somente na CPU e outra na CPU em conjunto com a GPU através da plataforma CUDA. Através dos experimentos realizados, observou-se que usando a plataforma CUDA é possível obter resultados satisfatórios.
Palavras-chave: algoritmos de mineração de dados; CUDA; processamento paralelo.

ABSTRACT - With the increasing need to make decisions in the short term, industry (pharmaceutical, petrochemical, aeronautics and etc.) has been seeking new ways to reduce the time of the data mining process to obtain knowledge. In recent years, many technological resources are being used to mitigate this need, an example is CUDA. CUDA is a platform that enables the use of GeForce GPUs in conjunction with CPUs for data processing, significantly reducing processing time. This work proposes to perform a comparative analysis of the processing time between two versions of some data mining algorithms (Apriori, AprioriAll, Naïve Bayes and K-Means), one running on CPU only and one on CPU in conjunction with GPU through platform CUDA. Through the experiments performed, it was observed that using the CUDA platform it is possible to obtain satisfactory results.
Keywords: data mining algorithms; CUDA; parallel processing.

Recebido em: 29/06/2017
Revisado em: 14/03/2018
Aprovado em: 17/10/2018

1. INTRODUÇÃO

Com os avanços tecnológicos, a sociedade teve um aumento substancial na geração, coleta e armazenagem de dados. Esse crescimento gerou a necessidade de criação de novas técnicas e ferramentas automatizadas que possibilitam transformar vastas quantidade de dados em informações e conhecimento uteis. Para suprir tais necessidades, ocorreu o surgimento da área multidisciplinar chamada mineração de dados (HAN; PEI; KAMBER, 2012).

Segundo Bortoleto, Giménez-Lugo e Silva (2015), devido algumas características dos algoritmos, o processo de mineração de dados em grandes bancos de dados pode levar rapidamente o esgotamento de recursos computacionais, tornando o processo menos eficiente.

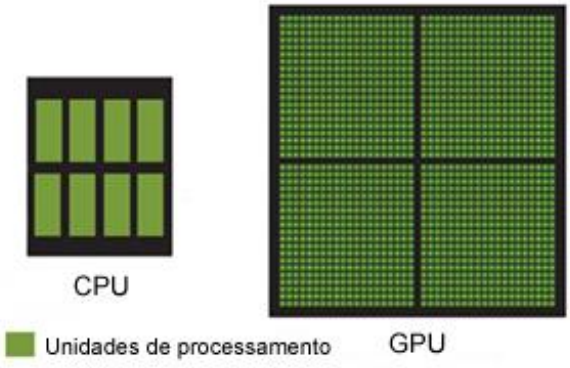
Quando possível, tarefas ou cálculos precisam ser executados de forma independente, aproveitando a capacidade de paralelismo dos processadores e seus respectivos núcleos (MORRISON, 2013).

De acordo com Silva, Gimenez-Lugo e Botoleto (2015), o processamento paralelo permite que mais de uma tarefa seja executada ao mesmo tempo, melhorando o desempenho e reduzindo o tempo de processamento.

Seguindo essa linha de raciocínio, o processamento em GPU[1] pode ser utilizado no âmbito de diminuir o tempo de execução dos algoritmos de mineração de dados, já que a tarefa é dividida entre centenas de unidades de processamento.

Segundo Rodrigues (2015), o alto grau de paralelismo das GPUs está relacionado à sua arquitetura. Diferente das CPUs, as GPUs dedicam maior parte de seus circuitos em unidades de processamento, em vez do controle. A Figura 1 mostra a grande diferença na quantidade de unidades de processamento de uma CPU e de uma GPU.

Figura 1. Diferença na quantidade de unidades de processamento de uma GPU e CPU.



A computação com GPU pode ser considerada como sendo o uso de uma unidade de processamento gráfico como um co-processador para CPU[2] para computação científica e de propósito geral (PAULA, 2013, p. 35).

Usando processamento em GPU uma equipe da NASA[3], conseguiu reduzir de dez minutos para três segundos o tempo de análise do Sistema de Espaço Aéreo Nacional. O sistema tem a função de gerenciar a coordenação do fluxo de tráfego aéreo dos Estados Unidos da América (NVIDIA, 2015b).

Outro ponto positivo de se utilizar as GPUs, é que elas têm menor custo se levar em consideração a sua capacidade de processamento. Um processador de quarta geração da Intel, o core i7 4770k, está na mesma faixa de preço que uma placa de vídeo com uma GPU GeForce[4] GTX770 e sua capacidade de processamento é de até 177 Gigaflops (177 bilhões de operações de ponto flutuante por segundo) e da placa de vídeo

1 GPU (Graphics Processing Unit, ou Unidade de Processamento Gráfico), conhecido também como VPU ou unidade de processamento visual, é o nome dado a um tipo de microprocessador especializado em processar gráficos em computadores pessoais, estações de trabalho ou videogames. (WIKIPÉDIA, 2017a)

2 A CPU - *Central Processing Unit* ou Unidade Central de Processamento (UCP) em português — é um circuito integrado que controla todas as operações e o funcionamento do computador, responsável pela execução de cálculos, decisões lógicas e instruções que resultam em todas as tarefas que um computador pode fazer. (TECMUNDO, 2017a)

3 Sigla em inglês de National Aeronautics and Space Administration – Administração Nacional da Aeronáutica e Espaço, é uma agência do Governo Federal dos Estados Unidos responsável pela pesquisa e desenvolvimento de tecnologias e programas de exploração espacial. (WIKIPÉDIA, 2017b)

4 GeForce é um modelo de aceleradores gráficos 3D para PCs desenvolvido pela NVIDIA. (WIKIPÉDIA, 2017g)

chegando até 3,2 Teraflops (3,2 trilhões de operações de ponto flutuante por segundo). Seria necessário em torno de 17 computadores equipados com a CPU i7 4770k interligados em rede, utilizando o conceito de processamento distribuído para ter o mesmo nível computacional.

Com os benefícios do processamento em GPU exposto, este trabalho de pesquisa implementou os algoritmos de mineração de dados Apriori, AprioriAll, Naïve Bayes e K-Means utilizando a plataforma de programação paralela CUDA, reduzindo o tempo de processamento quando comparado aos mesmos algoritmos sendo executados somente em CPU.

As demais seções deste trabalho estão organizadas da seguinte maneira: na Seção 2 são apresentadas as fundamentações; na Seção 3 são apresentados os trabalhos relacionados; na Seção 4 é apresentada a metodologia deste trabalho; na Seção 5 são apresentados os experimentos e resultados obtidos e, por fim, na Seção 6 são apresentadas as conclusões, considerações finais e trabalhos futuros.

2. FUNDAMENTAÇÃO

2.1 MINERAÇÃO DE DADOS

De acordo com Côrtes, Lifschitz e Porcaro (2002), a mineração de dados é uma etapa de um processo maior, chamado KDD (Knowledge Discovery in Databases – Descoberta de conhecimento em bancos de dados), o qual auxilia na preparação dos dados e facilita o entendimento dos padrões obtidos pela etapa de mineração. Mesmo sendo apenas uma etapa, a mineração de dados se tornou mais conhecida, pois é nela que são executados os algoritmos e são obtidos os padrões a serem interpretados.

Atualmente a mineração de dados dispõe de várias técnicas (algoritmos) para obtenção de padrões/conhecimentos, cada uma delas possui características próprias e podem produzir resultados diferentes, elas são categorizadas em tarefas.

A Tabela 1 mostra algumas tarefas e suas técnicas (algoritmos).

Tabela 1. Tarefas e suas técnicas.

Tarefa	Técnicas (algoritmos)
Regras de Associação	Apriori, FP-Growth, DCI, ECLAT, Closet.
Padrões Sequenciais	AprioriAll, GSP, PrefixSpan, BLAST.
Classificação	Naïve Bayes, Árvores de Decisão (ID3, C4.5, CART, CHAID), Redes Neurais, k-Nearest Neighbor, Support Vector Machines.
Análise de Agrupamentos	k-Médias, k-Medoides, Métodos Hierárquicos.

De acordo com Amo (2015a), a tarefa é a especificação do que se procura nos dados, que tipo de padrões ou regularidade se deseja encontrar nos dados. A técnica são os métodos (algoritmos) que garantem descobrir os padrões interessantes.

2.1.1 TAREFAS DE REGRAS DE ASSOCIAÇÃO

Consiste em encontrar transações de itens, que costumam ser adquiridos em conjunto (CARVALHO; VASCONCELOS, 2004). Segundo Pichiliani (2008) o algoritmo Apriori é um dos mais populares para extração de regras de associação. De acordo com Agrawal e Srikant (1994), o algoritmo retorna um conjunto de regras no formato A -> B (ocorrência de A implica na ocorrência de B), sendo A um conjunto de itens denominado antecedente da regra e B um conjunto denominado consequente. A geração dessas regras é baseada em dois parâmetros fornecidos pelo usuário. O primeiro é o suporte, que é o percentual mínimo que o conjunto A deve aparecer no número total de transações e o segundo é a confiança, sendo o percentual mínimo que o conjunto B deve aparecer no número total de ocorrência do conjunto A. Atingindo os mínimos de cada parâmetro, pode-se dizer que, ao adquirir o conjunto A em uma nova transação, adquiri-se também B. Saber trabalhar com esses parâmetros é fundamental para geração das

regras, já que eles determinam a quantidade e qualidades das regras geradas pelo algoritmo.

2.1.2 TAREFAS DE PADRÕES SEQUENCIAIS

Possibilitam descobrir itens que são obtidos em sequência, mas em transações diferentes (ALVARES, 2015). Segundo Amo (2015c), AprioriAll é um dos algoritmos usados para obtenção de padrões sequenciais. De acordo com Agrawal e Srikant (1995) o algoritmo é baseado em ideias semelhantes a do algoritmo Apriori (AprioriAll). Uma delas é o uso do parâmetro suporte, sendo neste caso o percentual mínimo que conjuntos de transações (ordenadas por data) deverão ser efetuados pelos clientes. Como resultado o algoritmo retorna padrões sequenciais no formato <{1,2,3},{4},{5,6}>, em que cada conjunto de itens ({...},{...}) na sequência (<...>) são transações e são ordenadas pela data da sua ocorrência. A partir dos padrões obtidos por essa técnica, pode-se prever possíveis itens a serem adquiridos em futuras transações.

2.1.3 TAREFAS DE AGRUPAMENTO

Tarefas de agrupamento têm a intenção de agrupar um conjunto de objetos com alguma similaridade. Segundo Prass (2013) o algoritmo K-Means é um dos algoritmos mais conhecidos e utilizados para análises de agrupamento. De acordo com J. MacQueen (1967), a quantidade de grupos gerada pelo algoritmo é definida inicialmente pelo usuário. Cada grupo possui um valor chamado de centroíde e o mesmo é usado para calcular a distância com cada elemento da tabela analisada. O elemento que estiver mais próximo, ou melhor, menor distância do grupo, será inserido nele. Em cada iteração, os centróides dos grupos são atualizados e a distância é calculada novamente e os elementos dependendo da distância trocam de grupo. Se na última iteração não houver mudanças nos grupos o algoritmo é finalizado.

2.1.4 TAREFAS DE CLASSIFICAÇÃO

Em tarefas de classificação, cria-se um modelo a partir de objetos pré-classificados de um banco de dados, e esse modelo passa a ter a capacidade de classificar novos objetos. Segundo Pichiliani (2006b) o algoritmo Naïve Bayes é um dos algoritmos que podem ser utilizados para realizar tarefas de classificação. De acordo com Mitchell (1997), esse algoritmo possui esse nome, por ser baseado no teorema de probabilidade de Bayes. Antes de iniciar o processo de classificação, é necessário que um conjunto de linhas já esteja previamente separado em classes, esse conjunto de dados é conhecido como amostras de treinamento. Para que uma nova amostra (desconhecida) possa ser classificada, cada classe da amostra de treinamento, precisa ter a sua probabilidade calculada, e a probabilidade de cada valor da amostra desconhecida, calculada para cada classe possível. Em seguida, as probabilidades de cada valor da amostra desconhecida de mesma classe, são multiplicadas, e o resultante multiplicado pela probabilidade da mesma classe. A classe que possuir maior probabilidade para a amostra desconhecida, será retornada pelo algoritmo.

2.2 CUDA

Conforme as GPUs foram evoluindo, muitas linguagens foram criadas para usufruir dos seus recursos e facilitar o seu uso. As duas mais conhecidas são o CUDA[5] e OPENCL[6] (PAULA, 2013).

CUDA é uma plataforma de computação paralela que visa aproveitar o poder das GPUs. Fornece aos programadores, abstrações simples para o gerenciamento de

5 CUDA™ é uma plataforma de computação paralela e um modelo de programação inventados pela NVIDIA. Ela permite aumentos significativos de performance computacional ao aproveitar a potência da unidade de processamento gráfico (GPU).

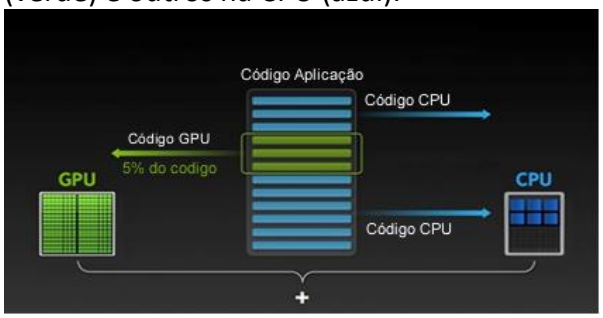
6 Open Computing Language é uma arquitetura para escrever programas que funcionam em plataformas heterogêneas, consistindo em CPUs, GPUs e outros processadores. (WIKIPÉDIA, 2017c)

threads[7], memória e outros recursos (RODRIGUES, 2015).

Segundo Paula (2013), CUDA foi criada pela NVIDIA[8] em 2006 e foi a primeira API[9] a permitir que a GPU pudesse ser utilizada para uma ampla variedade de aplicações.

Segundo Paula (2013), trechos de códigos em CUDA são executados separadamente na CPU e na GPU. Essa separação é realizada antecipadamente pelo compilador durante o processo de compilação. A Figura 2 mostra uma representação de código de uma aplicação com um trecho a ser executado na GPU e outros dois na CPU.

Figura 2. Demonstração de aplicação com um trecho de código a ser executados na GPU (verde) e outros na CPU (azul).



A popularização da plataforma CUDA no campo cientifico, biomédico, computação, análise de risco e da engenharia é devido às características das aplicações nesses campos, as quais são altamente paralelizáveis (VASCONCELLOS, 2009).

3. TRABALHOS RELACIONADOS

Lorenzoni, Rolim e Roque (2015), implementaram uma versão do algoritmo C4.5 em CUDA. Este algoritmo gera como produto, uma árvore de decisão, que posteriormente é usada nas tarefas de classificação na mineração de dados. O objetivo do estudo, foi a realização de testes, comparando a velocidade de execução do algoritmo em CPU e GPU através da plataforma CUDA. Nos três testes realizados, variando o tamanho da base de dados, 687, 797 e 803 Quilobytes, as versões em CUDA, obteve melhores resultados, e percebeu-se também, conforme aumentava a base de dados, melhores resultados eram obtidos no algoritmo em CUDA.

Artero, Meneguette Jr e Ywata (2017), implementaram os algoritmos de pré-processamento de dados e de coordenadas paralelas em CUDA, com o objetivo de torna-lo mais rápido do que executado na CPU. O algoritmo de coordenadas paralelas, possibilita que resultados obtidos em algoritmos de mineração de dados, sejam apresentados de forma mais compreensiva, a fim de aproveitar a capacidade humana em percepções de padrões. Para realização dos testes, foram utilizados arquivos com 5.000 a 5.000.000 de registros de dados contendo 5 atributos. Com o processamento dos arquivos de 5.000 até 400.000 registros, a versão executada somente em CPU, obteve menor tempo de execução, chegando a ser até 3 ordens de grandeza mais rápida em relação a versão com CUDA. Com arquivos de 500.000 a 5.000.000 de registros, o tempo de execução em CUDA foi ligeiramente menor, chegando apenas a décimos de milissegundos mais rápido. Segundo os autores, a pequena melhoria, está relacionado a baixa capacidade de processamento da GPU utilizada nos testes, a baixa velocidade de transferência de dados da memória RAM[10] do computador para a memória da placa de vídeo e o inverso, e também, possíveis melhorias no código CUDA poderiam ser realizadas para torna-lo mais rápido.

7 Thread: permitem que um programa simples possa executar várias tarefas diferentes ao mesmo tempo, independentemente umas das outras (ALCÂNTARA, 1996, p 1).

8 NVIDIA (não é abreviatura, embora fosse grafada como nVidia, anteriormente) é uma empresa multinacional de tecnologia com sede em Santa Clara, Califórnia que fabrica peças de computador, e é mais popularmente conhecida por sua série de placas de vídeo GeForce. (WIKIPÉDIA, 2017d)

9 API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A sigla API refere-se ao termo em inglês "Application Programming Interface" que significa em tradução para o português "Interface de Programação de Aplicativos". (CANALTECH, 2017b)

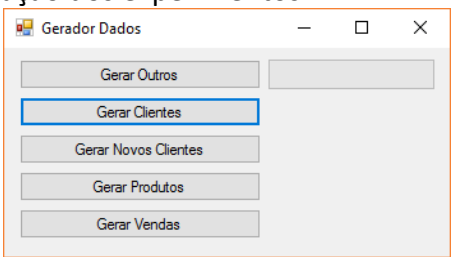
10 A memória RAM (*Random Access Memory* - Memória de Acesso Aleatório) é um hardware de armazenamento randômico e volátil de memória. Isto significa que esta peça armazena dados de programas em execução enquanto o computador está ligado. (INFOESCOLA, 2017)

Mamani (2011) realizou estudos e desenvolveu soluções baseados em busca por similaridade aproximada em CUDA, com o objetivo de reduzir o custo computacional de tarefas de mineração de dados que utilizam a "busca pelo vizinho mais próximo" como procedimento mais importante. A busca aproximada é indicada, quando a resposta exata não é um requisito fundamental, e sim o tempo de resposta. É uma alternativa para reduzir o efeito da "maldição da alta dimensionalidade" que acaba elevando o custo computacional na busca exata. Para realização dos estudos, foram utilizados 10 conjuntos de dados complexos, com muitas dimensões, e de acordo com o autor, a solução utilizando CUDA, resultou em um aumento de desempenho de até 7 vezes.

4. METODOLOGIA

A solução criada neste trabalho, para realizar comparações nos algoritmos de mineração de dados, executados em CPU e CPU em conjunto com a GPU, por meio da plataforma CUDA, foi o desenvolvimento de duas aplicações na linguagem C#[11], usando o modelo Windows Form em .NET[12], através do ambiente Visual Studio 2013 Community e o SQL Server 2016 para armazenagem dos dados. Como a linguagem C# da plataforma .NET não é suportada "nativamente" na plataforma CUDA, foi usada uma biblioteca chamada CUDAFY. Essa biblioteca é disponibilizada juntamente com vários exemplos no site Codeplex, ela possibilita o uso da plataforma .NET em conjunto com a plataforma CUDA. Uma das aplicações desenvolvidas (Figura 3) tem como propósito gerar dados fictícios, simulando transações de um supermercado, e carregá-los no banco de dados. A quantidade de dados gerada por essa aplicação, possibilitou a realização de três testes por algoritmo, usando 100, 500 e 1024 Megabytes de dados.

Figura 3. Gerador de dados fictícios para realização dos experimentos.

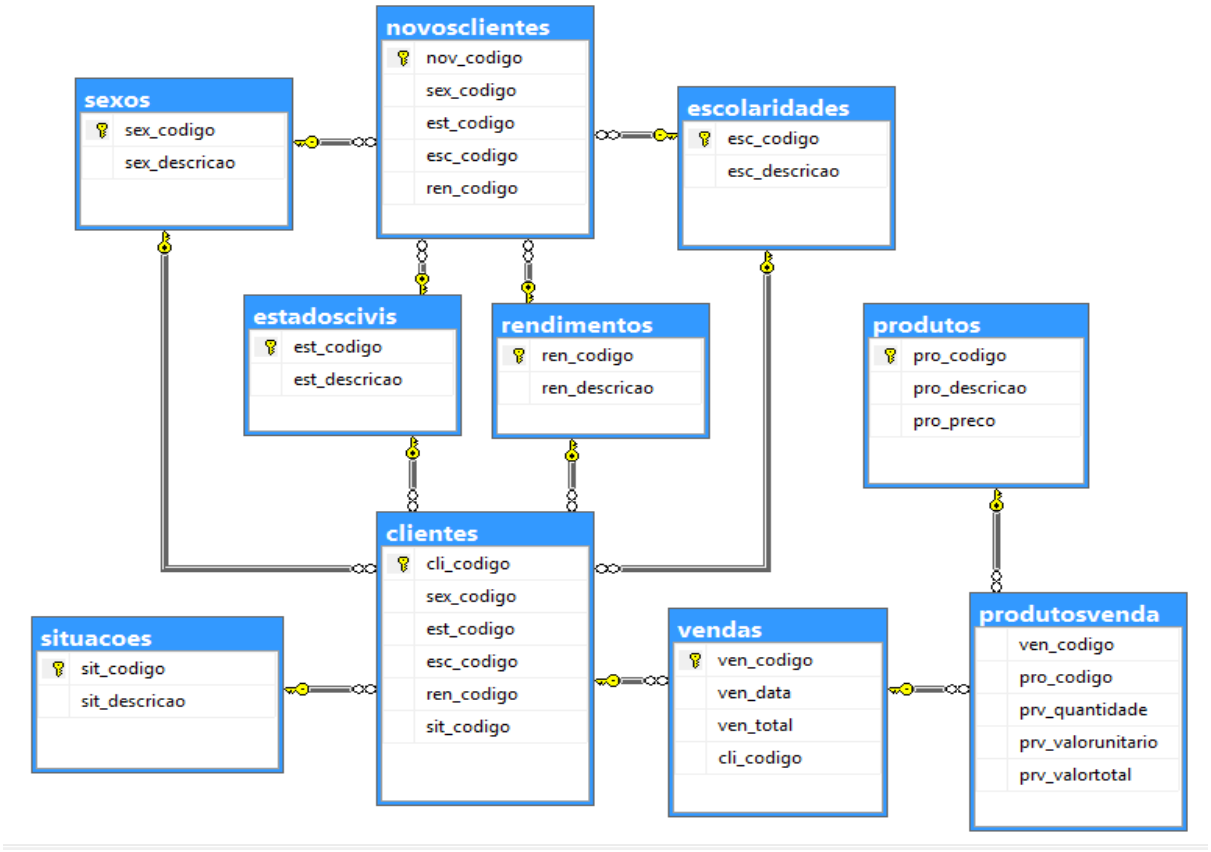


Para armazenar os dados gerados pela aplicação e posteriormente extrai-los para os algoritmos, foi criado um banco de dados representado pelo modelo de dados da Figura 4.

11 É uma linguagem de programação orientada a objetos desenvolvida pela Microsoft, que roda sobre o .NET Framework. (DEVMEDIA, 2017)

12 O .NET Framework (pronuncia-se: dotNet) é uma iniciativa da empresa Microsoft, que visa uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código gerado para .NET pode ser executado em qualquer dispositivo que possua um framework de tal plataforma. (WIKIPÉDIA, 2017e)

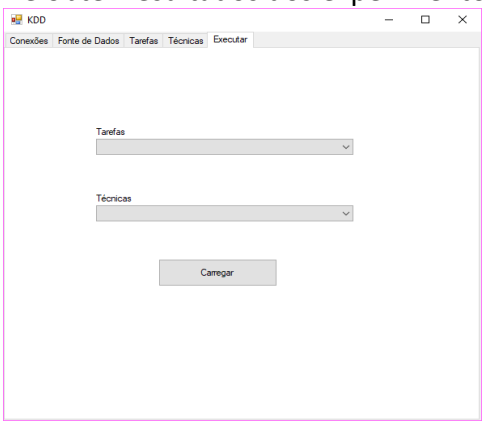
Figura 4. Modelo de dados usado para armazenar dados dos experimentos.



As entidades “sexos”, “escolaridades”, “estadoscivis’, “rendimentos” e “situações”, caracterizam os registros das entidades “clientes” e “novosclientes” (exceto situações). Os registros da entidade “novosclientes”, indicam clientes que vão passar por um processo de aprovação de crédito. Na entidade “clientes” possui registros das pessoas que realizaram compras no supermercado. Em “produtos” fornece dados sobre produtos e é utilizado na realização das compras. A entidade “produtosvenda” registra produtos comprados na venda. Em “vendas” registra as vendas do supermercado.

Para realizar os testes e comparações entre versões do mesmo algoritmo, foi desenvolvida uma outra aplicação (Figura 5), que possibilitou a conexão com banco de dados, seleção dos dados, escolha e configuração (parâmetros) dos algoritmos e obtenção do tempo de execução.

Figura 5. Aplicação para configurar e obter resultados dos experimentos.



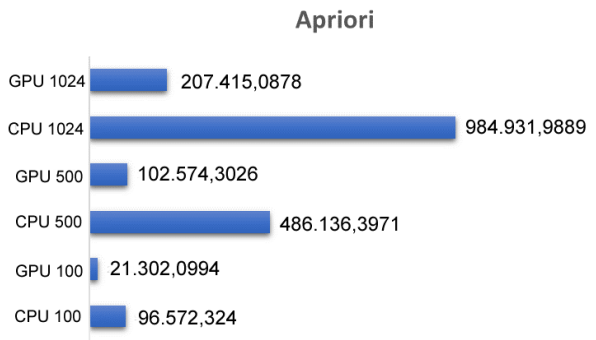
Os testes foram realizados em um computador com o sistema operacional Windows 10 Professional, uma CPU Intel Core i7-4770k (3.5 Gigahertz), 16 Gigabytes de Memória RAM e uma placa de vídeo com a GPU GTX770 (1536 núcleos CUDA) contendo 2 Gigabytes de memória GDDR5[13].

5. EXPERIMENTOS E RESULTADOS

Foram realizados três experimentos nas duas versões de cada algoritmo (Apriori, AprioriAll, K-Means e Naïve Bayes), variando a quantidades de dados. O primeiro experimento foi usado 100 Megabytes de dados, o segundo 500 Megabytes e o terceiro 1024 Megabytes.

No algoritmo Apriori, usando 100 Megabytes, resultou em um experimento de 13.107.200 registros, 500 Megabytes em 65.536.000 registros e 1024 Megabytes em 134.217.728 registros, sendo que em cada experimento, cada registro continha 2 atributos. Como o algoritmo necessita de parâmetros de entrada, suporte e confiança, foram usados valores 20% e 60% respectivamente. Os resultados dos experimentos são apresentados graficamente na Figura 6.

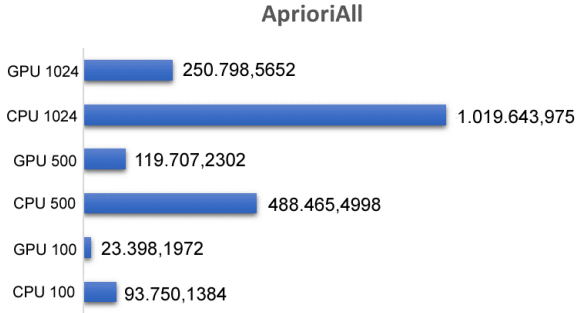
Figura 6. Gráfico do tempo de execução do algoritmo Apriori (Milissegundos).



No experimento de 100 Megabytes (CPU 100, GPU 100), a versão usando CUDA foi 4,53 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 21.302,0994 milissegundos (21 segundos e 302 milissegundos), enquanto a versão usando somente a CPU obteve 96.572,324 milissegundos (1 minuto, 36 segundos e 572 milissegundos). Com 500 Megabytes (CPU 500, GPU 500) a versão usando CUDA foi 4,74 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 102.574,3026 milissegundos (1 minuto, 42 segundos e 574 milissegundos), enquanto a versão usando somente a CPU obteve 486.136,3971 milissegundos (8 minutos, 6 segundos e 136 milissegundos). No experimento de 1024 Megabytes (CPU 1024, GPU 1024), a versão usando CUDA foi 4,75 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 207.415,0878 milissegundos (3 minutos, 27 segundos e 415 milissegundos), enquanto a versão usando somente a CPU obteve 984.931,9889 milissegundos (16 minutos, 24 segundos e 932 milissegundos).

No algoritmo AprioriAll, usando 100 Megabytes, resultou em um experimento de 5.242.880 registros, 500 Megabytes em 26.214.400 registros e 1024 Megabytes em 53.687.092 registros, sendo que em cada experimento, cada registro continha 4 atributos. Como o algoritmo necessita de parâmetro de entrada, suporte, foi usado valor 75%. Os resultados dos experimentos são apresentados graficamente na Figura 7.

Figura 7. Gráfico do tempo de execução do algoritmo AprioriAll (Milissegundos).



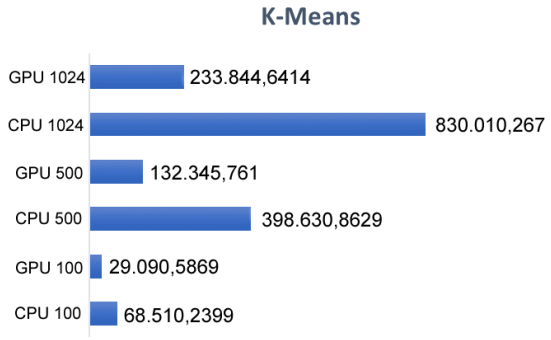
No experimento de 100 Megabytes (CPU 100, GPU 100), a versão usando CUDA foi 4,01 vezes mais rápida que a executada somente em CPU, obtendo o tempo de

13 Desenvolvida para a utilização em placas gráficas, a GDDR é uma sigla em inglês que significa Graphics Double Data Rate, ou, em uma tradução simples, "velocidade de transmissão de dados gráficos dobrada". (CANALTECH, 2017c)

execução de 23.398,1972 milissegundos (23 segundos e 398 milissegundos), enquanto a versão usando somente a CPU obteve 93.750,1384 milissegundos (1 minuto, 33 segundos e 750 milissegundos). Com 500 Megabytes (CPU 500, GPU 500) a versão usando CUDA foi 4,08 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 119.707,2302 milissegundos (1 minuto, 59 segundos e 707 milissegundos), enquanto a versão usando somente a CPU obteve 488.465,4998 milissegundos (8 minutos, 8 segundos e 465 milissegundos). No experimento de 1024 Megabytes (CPU 1024, GPU 1024), a versão usando CUDA foi 4,07 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 250.798,5652 milissegundos (4 minutos, 10 segundos e 799 milissegundos), enquanto a versão usando somente a CPU obteve 1.019.643,975 milissegundos (16 minutos, 59 segundos e 644 milissegundos).

No algoritmo K-Means, usando 100 Megabytes, resultou em um experimento de 8.738.134 registros, 500 Megabytes em 43.690.667 registros e 1024 Megabytes em 89.478.486 registros, sendo que, em cada experimento, cada registro continha 2 atributos. Como o algoritmo necessita de parâmetro de entrada, foi usado valor 3 representando a quantidade de grupos. Os resultados dos experimentos são apresentados graficamente na Figura 8.

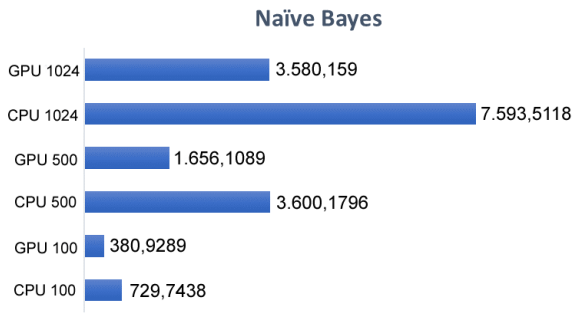
Figura 8. Gráfico do tempo de execução do algoritmo K-Means (Milissegundos).



No experimento de 100 Megabytes (CPU 100, GPU 100), a versão usando CUDA foi 2,36 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 29.090,5869 milissegundos (29 segundos e 91 milissegundos), enquanto a versão usando somente a CPU obteve 68.510,2399 milissegundos (1 minuto, 8 segundos e 510 milissegundos). Com 500 Megabytes (CPU 500, GPU 500) a versão usando CUDA foi 3,01 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 132.345,761 milissegundos (2 minutos, 12 segundos e 346 milissegundos), enquanto a versão usando somente a CPU obteve 398.630,8629 milissegundos (6 minutos, 38 segundos e 631 milissegundos). No experimento de 1024 Megabytes (CPU 1024, GPU 1024), a versão usando CUDA foi 3,55 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 233.844,6414 milissegundos (3 minutos, 53 segundos e 845 milissegundos), enquanto a versão usando somente a CPU obteve 830.010,267 milissegundos (13 minutos, 50 segundos e 10 milissegundos).

No algoritmo Naïve Bayes, no experimento de 100 Megabytes de dados, foi usado como amostra desconhecida 50 Megabytes e na amostra de treinamento 50 Megabytes, resultando em um experimento de 3.276.800 registros para amostras desconhecidas e 2.621.440 registros para amostras de treinamento. Com o experimento de 500 Megabytes, foi usado na amostra desconhecida 250 Megabytes e na amostra de treinamento 250 Megabytes, resultando em um experimento de 16.384.000 registros para amostras desconhecidas e 13.107.200 registros para amostras de treinamento. No experimento de 1024 Megabytes, foi usado como amostra desconhecida 512 Megabytes e na amostra de treinamento 512 Megabytes, resultando em um experimento de 33.554.432 de registros para amostras desconhecidas e 26.843.546 registros para amostras de treinamento. Os resultados dos experimentos são apresentados graficamente na Figura 9.

Figura 9. Gráfico do tempo de execução do algoritmo Naïve Bayes (Milissegundos).



No experimento de 100 Megabytes (CPU 100, GPU 100), a versão usando CUDA foi 1,92 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 380,9289 milissegundos, enquanto a versão usando somente a CPU obteve 729,7438 milissegundos. Com 500 Megabytes (CPU 500, GPU 500) a versão usando CUDA foi 2,17 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 1.656,1089 milissegundos (1 segundo e 656 milissegundos), enquanto a versão usando somente a CPU obteve 3.600,1796 milissegundos (3 segundos e 600 milissegundos). No experimento de 1024 Megabytes (CPU 1024, GPU 1024), a versão usando CUDA foi 2,12 vezes mais rápida que a executada somente em CPU, obtendo o tempo de execução de 3.580,159 milissegundos (3 segundos e 580 milissegundos), enquanto a versão usando somente a CPU obteve 7.593,5118 milissegundos (7 segundos e 594 milissegundos).

6. CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi realizar uma análise comparativa do tempo de processamento entre duas versões de cada algoritmo de mineração de dados: Apriori, AprioriAll, Naïve Bayes e K-Means. Uma versão sendo executada somente na CPU e outra na CPU em conjunto com a GPU através da plataforma CUDA. Em todos os experimentos realizados, 100, 500 e 1024 Megabytes de dados, as versões usando CUDA, demonstraram melhorias significativas nos tempos de execução em relação às versões executadas somente em CPU. Desta forma, conclui-se que o objetivo foi alcançado, pois o aumento de desempenho de alguns algoritmos foi mais de 4 vezes usando a plataforma CUDA. Em relação a trabalhos futuros tem se a possibilidade de otimizar os algoritmos CUDA e quantificar a melhora. Havendo possibilidade de usar técnica de sobreposição de dados e usar memórias mais próximas da GPU (registradores e memoria compartilhada) ou até mudando a forma como é acessada a memória global, o algoritmo poderá ter uma melhora considerável.

REFERÊNCIAS

AGRAWAL R.; SRIKANT R. **Mining Sequential Patterns.** 1994. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.1559&rep=rep1&type=pdf>. Acesso em: 13 jan. 2018.

AGRAWAL R.; SRIKANT R. **Fast Algorithms for Mining Association Rules.** 1995. Disponível em: <http://cs.stanford.edu/people/chrismre/cs345/rl/ar-mining.pdf>. Acesso em: 09 jan. 2018.

ALCÂNTARA, A. A. **O que são threads.** 1996. Disponível em: <http://www.di.ufpe.br/~java/verao/aula8/definicao.html>. Acesso em: 14 jan. 2016.

ALCHIERI, E. **Processos e threads.** 2016. Disponível em: <http://www.cic.unb.br/~alchieri/disciplinas/graduacao/pc/processos.pdf>. Acesso em: 14 jan. 2016.

ALVARES, L. O. **Associação:** algoritmos (parte 2). 2015. Disponível em: <http://www.inf.ufsc.br/~alvares/INE5644/associacao2.ppt>. Acessado em: 20 Mar. 2016.

AMO, S. **Técnicas de Mineração de Dados.** 2015a**.** Disponível em:

<http://www.deamo.prof.ufu.br/arquivos/JAI-cap5.pdf>. Acesso em: 28 nov. 2015.

AMO, S. **Curso de Data Mining -** Aula 2 - Mineração de Regras de Associação - O algoritmo APRIORI. 2015b. Disponível em: <http://www.deamo.prof.ufu.br/arquivos/Aula2.pdf>. Acessado em: 21 mar. 2016.

AMO, S. **Curso de Data Mining -** Aula 4 - Mineração de Sequências - Algoritmos AprioriALL e GSP. 2015c. Disponível em: <http://www.deamo.prof.ufu.br/arquivos/Aula4.pdf>. Acesso em: 21 mar. 2016.

ANHEMBI. **Tarefas e Técnicas de Mineração de Dados.** 2015. Disponível em: <http://conteudo.anhembi.br/ead/conteudo/pool_online/6800_banco_de_dados_e_inteligencia_empresarial/pdf/6800_(8_2).pdf>. Acesso em: 28 nov. 2015.

ARTERO, A. O.; MENEGUETTE JR, M.; YWATA, R. S. **Paralelização da técnica coordenadas paralelas utilizando a plataforma de computação paralela CUDA.** 2017. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwil2aCOrN7TAhWIiZAKHcCbATwQFggjMAA&url=http%3A%2F%2Fprope.unesp.br%2Fcic%2Fadmin%2Fver_resumo.php%3Farea%3D100081%26subarea%3D24302%26congresso%3D36%26CPF%3D31290141878&usg=AFQjCNFNQMQmNxI2keSX4Ba2PLmC33yFlA&sig2=c6mmvT2xkNISfbcCU30a1A>. Acesso em: 7 mai. 2017.

BORTOLETO, S.; GIMÉNEZ-LUGO, G. A.; SILVA, F. R. B. **Aplicação de um sistema para mineração de dados de vendas.** 2015. Disponível em: <http://www.aedb.br/seget/arquivos/artigos07/1405_SetsMiner_V8B.pdf>. Acesso em: 28 nov. 2015.

CAMILO, C. O.; SILVA, J. C. **Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas.** 2009. Disponível em: <http://www.inf.ufg.br/sites/default/files/uploads/relatorios-tecnicos/RT-INF_001-09.pdf>. Acesso em: 28 nov. 2015.

CANALTECH. **O que é a CUDA?**. 2017a. Disponível em: <https://corporate.canaltech.com.br/o-que-e/placa-de-video/O-que-e-a-CUDA/>. Acesso em: 6 mai. 2017.

CANALTECH. **O que é API?**. 2017b. Disponível em: <https://canaltech.com.br/o-que-e/software/o-que-e-api/>. Acesso em: 7 mai. 2017.

CANALTECH. **O que é GDDR5?**. 2017c. Disponível em: <https://canaltech.com.br/o-que-e/o-que-e/O-que-e-GDDR5/>. Acesso em: 8 mai. 2017.

CARVALHO, C. L.; VASCONCELOS, L. M. R. **Aplicação de Regras de Associação para Mineração de Dados na Web.** 2004. Disponível em: <http://www.inf.ufg.br/sites/default/files/uploads/relatorios-tecnicos/RT-INF_004-04.pdf>. Acesso em: 20 jan. 2016.

CÔRTES, S. C.; LIFSCHITZ, S.; PORCARO, R. M. **Mineração de Dados - Funcionalidades, Técnicas e Abordagens.** 2002. Disponível em: <ftp://ftp.inf.puc-rio.br/pub/docs/techreports/02_10_cortes.pdf>. Acessado em: 28 nov. 2015.

DEVMEDIA. **O que é C#?.** 2017. Disponível em: <http://www.devmedia.com.br/curso/o-que-e-csharp/1983>. Acesso em: 7 mai. 2017.

HAN, J.; PEI, J.; KAMBER, M. **Data mining:** concepts and techniques. 3. ed. 2012.

INFOESCOLA. 2017. **Memória RAM**. Disponível em: <http://www.infoescola.com/informatica/memoria-ram/>. Acesso em: 7 mai. 2017.

MACQUEEN, J. **Some methods for classification and analysis of multivariate**

observations. 1967. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.8619&rep=rep1&type=pdf>. Acesso em: 22 jan. 2018.

LORENZONI, A. M.; ROLIM, C. O.; ROQUE, A. S. **Paralelização do Algoritmo C4.5 para Mineração de Dados usando Unidades Gráficas de Processamento GPUs.** 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/erad-rs/2014/0049.pdf>. Acesso em: 28 nov. 2015.

MAMANI, A. V. O. **Paralelização da técnica coordenadas paralelas utilizando a plataforma de computação paralela CUDA.** 2011. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-22112011-132339/publico/dissertacao_aocsa.pdf>. Acesso em: 7 mai. 2017.

MARTINS, W. S. et al. **Aplicação de Processamento Paralelo em método iterativo para solução de sistemas lineares.** 2013. Disponível em: <http://www.academia.edu/5356091/Aplica%C3%A7%C3%A3o_de_Processamento_Paralelo_em_M%C3%A9todo_Iterativo_para_Solu%C3%A7%C3%A3o_de_Sistemas_Lineares>. Acesso em: 28 nov. 2015.

MITCHELL, T. M. **MachineLearning**. WCB McGraw - Hill, 1997. Cap. 6.

MORRISON, R. S. **Cluster Computing -** architectures, operating systems, parallel processing & programming languages. Sydney: University of Technology, 2003.

NETO, F. P. **Implementação de convolução em GPU.** 2010. Disponível em: <http://adessowiki.fee.unicamp.br/adesso/wiki/courseIA366F2S2010/fp100356P1/view/>. Acesso em: 28 nov. 2015.

NVIDIA. **O QUE É COMPUTAÇÃO COM ACELERAÇÃO DE GPU?.** 2015a. Disponível em: <http://www.nvidia.com.br/object/what-is-gpu-computing-br.html>. Acesso em: 28 nov. 2015.

NVIDIA. **O que é CUDA?.** 2015b. Disponível em: <http://www.nvidia.com.br/object/cuda_home_new_br.html>. Acesso em: 21 nov. 2015.

PAULA, L. C. M. **CUDA VS. OPENCL:** uma comparação teórica e tecnológica**.** 2013. Disponível em: <http://formiga.ifmg.edu.br/forscience/index.php/forscience/article/view/53/59>. Acesso em: 28 nov. 2015.

PICHILIANI, M. **Data Mining na Prática: Regras de Associação.** 2008. Disponível em: <http://imasters.com.br/artigo/7853/sql-server/data-mining-na-pratica-regras-de-associacao>. Acesso em: 21 mar. 2016.

PICHILIANI, M. **Data Mining na prática:** algoritmo K-Means**.** 2006a. Disponível em: <http://imasters.com.br/artigo/4709/sql-server/data-mining-na-pratica-algoritmo-k-means>. Acesso em: 20 mar. 2016.

PICHILIANI, M. **Data Mining Data Mining na Prática: Classificação Bayesiana.** 2006b. Disponível em: <http://imasters.com.br/artigo/4926/sql-server/data-mining-na-pratica-classificacao-bayesiana>. Acesso em: 20 mar. 2016.

PRASS, F. **Algoritmo de k-means.** 2013. Disponível em: <http://fp2.com.br/blog/index.php/2013/algoritmo-de-k-means/>. Acesso em: 20 Mar. 2016.

RODRIGUES, E. C. S. **GPU sob o ponto de vista de arquiteturas paralelas, organização interna e utilização em sistemas de paralelismo massivo.** 2015. Disponível em: <http://www.ic.unicamp.br/~cortes/mo601/trabalho_mo601/elisa_rodrigues_gpu/ra098329_relatorio.pdf>. Acesso em: 28 nov. 2015.

TECMUNDO. **Você sabe o que é uma CPU?**. 2017a. Disponível em: <https://www.tecmundo.com.br/intel/209-voce-sabe-o-que-e-uma-cpu-.htm>. Acesso em: 6 mai. 2017.

TECMUNDO. **O que é um arquivo com extensão DLL?**. 2017b. Disponível em: <https://www.tecmundo.com.br/o-que-e/44592-o-que-e-um-arquivo-com-extensao-dll-.htm>. Acesso em: 8 mai. 2017.

VASCONCELLOS, F. B. **Programando com GPUs:** paralelizando o Método Lattice-Boltzmann com CUDA. 2009. Disponível em: <http://www.inf.ufrgs.br/~nicolas/pdf/tcc_vasconcellos.pdf>. Acessado em: 04 set. 2018.

W3SCHOOLS. **Tutorial do SQL.** 2017. Disponível em: <https://www.w3schools.com/sql/DEfaULT.asP>. Acesso em: 6 mai. 2017.

WIKIPÉDIA. **Unidade de processamento gráfico**. 2017a. Disponível em: <https://pt.wikipedia.org/wiki/Unidade_de_processamento_gr%C3%A1fico>. Acesso em: 6 mai. 2017.

WIKIPÉDIA. **NASA.** 2017b. Disponível em: <https://pt.wikipedia.org/wiki/NASA>. Acessado em: 7 mai. 2017.

WIKIPÉDIA. **OpenCL**. 2017c. Disponível em: <https://pt.wikipedia.org/wiki/OpenCL>. Acesso em: 7 mai. 2017.

WIKIPÉDIA. **NVIDIA**. 2017d. Disponível em: <https://pt.wikipedia.org/wiki/NVIDIA>. Acesso em: 7 mai. 2017.

WIKIPÉDIA. **Microsoft .NET**. 2017e. Disponível em: <https://pt.wikipedia.org/wiki/Microsoft_.NET>. Acesso em: 7 mai. 2017.

WIKIPÉDIA. **IBM**. 2017f. Disponível em: <https://pt.wikipedia.org/wiki/IBM>. Acesso em: 7 mai. 2017.

WIKIPÉDIA. **GeForce**. 2017g. Disponível em: <https://pt.wikipedia.org/wiki/GeForce>. Acesso em: 7 mai. 2017.

WIKIPÉDIA. **ODBC**. 2017h. Disponível em: <https://pt.wikipedia.org/wiki/ODBC>. Acesso em: 3 jun. 2017.