Estudos comparativos sobre diferentes planos de consulta no SAP HANA

Guilherme Balbinot <gbalbinot@gmail.com> *
Denise Bandeira <bandeira@unisinos.br> †

16 de dezembro de 2014

Resumo

Os sistemas analíticos são de extrema importância para tomada de decisões. O crescimento do volume de dados trouxe o desafio de lidar com um volume de dados cada vez maior, de maneira eficaz e veloz. Os usuários, por sua vez, desejam a informação em um tempo cada vez menor, em tempo real, ao invés de aguardar por relatórios demorados que são extraídos apenas em alguns períodos do mês. A utilização de bancos de dados em memoria oferece uma solução para estes problemas. A utilização do armazenamento de dados em memoria oferece ganhos de desempenho, visto que a manipulação de dados em memoria é significativamente mais rápida que a manipulação de dados em disco rígido. As tecnologias com armazenamento em colunas possibilitam a compressão de dados, aumentando mais ainda a velocidade de leitura de dados. Além disso, é possível executar consultas complexas através de funções e procedimentos para obter melhor modularização, simplificação, entendimento e desempenho . Este trabalho propõe conceituar as bases de dados em memória e pesquisar suas funcionalidades. Este trabalho ainda tem como objetivo estudar a plataforma SAP HANA e explorar algumas de suas funcionalidades, demonstrando os benefícios que tais ferramentas podem proporcionar para aumentar o desempenho na obtenção de dados.

Palavras-chaves: base de dados em memória. computação em memória.

Introdução

Segundo (GANTZ; REINSEL, 2011), no mundo globalizado dos negócios, estima-se que o volume de dados gerados em sistemas de Tecnologia da Informação (TI) dobre a cada 18 meses. Consequentemente, os setores envolvidos com TI tem um desafio de encontrar novas maneiras de lidar com maior volume de dados, maior volume de usuários e, ao mesmo tempo, manter os custos dentro do orçamento. Com tamanho volume de dados para manipular, as empresas lutam constantemente para manter os seus processos críticos dentro das janelas de tempo disponíveis. Além disso, é comum o fato de usuários sofrerem

*Aluno do curso de Sistemas de Informação da Universidade do Vale do Rio dos Sinos
†Profa. Msc. Denise Bandeira, coordenadora do curso de Sistemas de Informação da Universidade do Vale do Rio dos Sinos

com longos tempos de espera a cada execução de um relatório em grandes volumes de dados.

Para dar suporte a este constante crescimento de dados e usuários, uma nova geração de base de dados baseadas no conceito de armazenamento de dados em memória, está disponível. Este modelo de persistência viabiliza novas possibilidades que antes não eram possíveis com as tradicionais bases de dados em disco rígido, tais como: a computação em tempo real, na qual qualquer modificação nos dados que acontece no sistema se torna visível instantaneamente.

Tais ferramentas são baseadas no conceito de Base de Dados em Memória, do inglês In-memory Databases (IMDB) e tem como princípio armazenar a base de dados, ou parte dela, na memória principal (memória RAM) dos servidores, ao invés de armazená-la nos discos rígidos. De acordo com (ROSS, 2009), a leitura em memória é ordens de grandeza mais rápida do que a leitura em disco (microssegundos ao invés de milissegundos), e isso faz toda diferença.

A utilização de um IMDB tem o intuito de aumentar a produtividade dos usuários e, ao mesmo tempo, diminuir o custo total de uma solução quando comparada com outros meios, principalmente no quesito de aprimoramento do desempenho de pesquisas em grandes bases de dados. Se comparado às bases de dados tradicionais, não somente processos existentes são executados mais rapidamente do que antes, como também um número maior de processos pode ser realizado na mesma janela de tempo. Isso encoraja usuários a trabalharem mais livremente e extrair mais valor dos dados, o que significa uma vantagem competitiva para a empresa.

A gestão de banco de dados das aplicações de negócios, nos dias de hoje, é um dos temas mais desafiadores da indústria de software. A informação não apenas rege os negócios, mas como também serve como base para o desenvolvimento de novos negócios inovadores. A gestão de banco de dados, em todos seus escopos, é um ativo fundamental para as organizações. Além disso, ela tem ganhado atenção dos gestores, pois está sendo vista como a principal ferramenta para orientar e manter suas organizações. Do lado do sistema operacional, os cenários de gestão de dados estão cada vez mais complexos e complicados de gerenciar. Segundo (SIKKA et al., 2012), uma camada de gestão de dados eficiente, robusta e econômica é essencial nos dias de hoje .

Este trabalho propõe conceituar as bases de dados em memória e pesquisar suas funcionalidades. Este trabalho ainda tem como objetivo estudar a plataforma SAP HANA e explorar algumas de suas funcionalidades através da execução de experimentos.

1 Impacto da computação em memória

Os modelos de decisão de negócios estão cada vez mais sofisticados e dependem do acesso extremamente rápido e manipulação de quantidades enormes de dados. Realizar operações de negócios em tempo real exige volumes de dados que muitas vezes estão além das capacidades de sistemas tradicionais baseados em disco.

Conforme (PLATTNER; ZEIER, 2012), a obtenção da informação em tempo real, chamada de real-time, pode ser definida como “toda mudança que acontece dentro de uma empresa cuja visibilidade é imediata para os usuários tomadores de decisão”.

Alguns exemplos de sistemas de informação utilizados para a tomada de decisão são as bases de dados relacionais e os sistemas de Enterprise Resource Planning (ERP). Porém,

com o aumento do volume de dados que necessitam serem processados, as estratégias precisaram sofrer algumas alterações. Conforme (PLATTNER; ZEIER, 2012),

> atualmente, a maioria dos dados de uma empresa está distribuída ao longo de uma vasta gama de aplicações e estão armazenados em bases de dados diferentes. Criar uma visão unificada desses dados é um processo complicado e que consome tempo. Além disso, relatórios de análise de negócios tipicamente não são executados diretamente em dados operacionais, mas sim em esquemas de dados agregados de um Data Warehouse. Os dados operacionais são transferidos para Data Warehouses em processos que rodam durante a noite, o que torna os relatórios em tempo real uma tarefa impossível. Como consequência, os tomadores de decisão da companhia precisam tomar decisões baseadas nos dados que são ou incompletos ou desatualizados. Obviamente isso não é uma verdadeira solução em tempo real.

A arquitetura de hardware continua em rápida evolução desde a introdução dos microprocessadores. Na última década foram introduzidos processadores multi-nucleos e memoria de baixo custo, permitindo novas possibilidades na indústria de software, possibilitando o armazenamento de conjuntos de dados de empresas inteiras em memória. Segundo (PLATTNER; ZEIER, 2012), o discos rígido, o único dispositivo mecânico remanescente em um mundo de silício, em breve será apenas necessário para fazer backup de dados.

2 Principais funções de uma base de dados em memória

2.1 Armazenamento orientado em linhas e colunas

Bancos de dados relacionais organizam os dados em tabelas, que contêm registros de dados. A diferença entre armazenamento orientado a linhas e armazenamento orientado a colunas é a forma na qual as tabelas são armazenadas.

O armazenamento baseado em linhas armazena uma tabela em uma sequência de linhas, conforme visto na figura 1. A leitura de uma tupla em uma tabela orientada a linhas requer o mínimo de leituras possíveis. A figura 2 exemplifica como é feita a leitura nas tabelas orientadas a linha.

O armazenamento baseado em colunas armazena uma tabela em uma sequência de colunas, conforme visto na figura 3. A leitura de uma tupla em uma tabela orientada a linhas requer o mínimo de leituras possíveis. A figura 4 exemplifica como é feita a leitura nas tabelas orientadas a linha.

Figura 1 – Exemplo de armazenamento orientado a linhas. Fonte: elaborado pelo autor.

ID	NOME	TELEFONE	ID	NOME	TELEFONE	ID	NOME	TELEFONE
linha 1			linha 2			linha 3		

2.2 Vantagens e desvantagens de cada esquema

Para atender requisitos específicos, dois diferentes tipos de sistemas de bases de dados foram criados: os sistemas analíticos e transacionais. Os sistemas de bases de dados para carga de trabalho transacional armazenam e processam dados orientados em linha,

Figura 2 – Exemplo de leitura em uma tabelas orientada a linhas. Fonte: elaborado pelo autor.

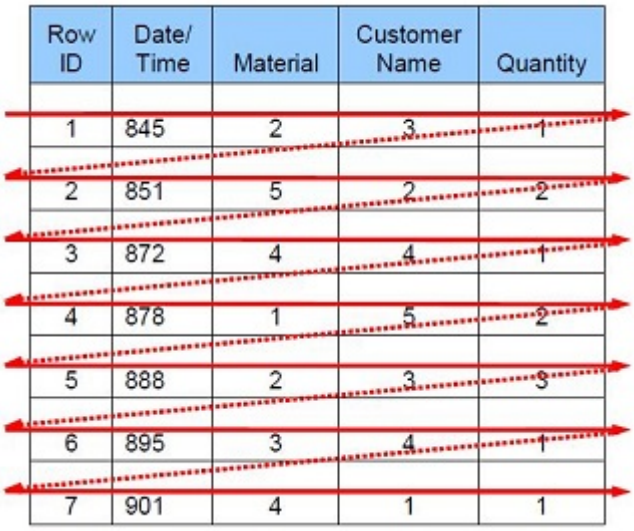


Figura 3 – Exemplo de armazenamento orientado a colunas. Fonte: elaborado pelo autor.

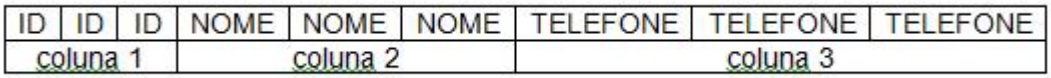
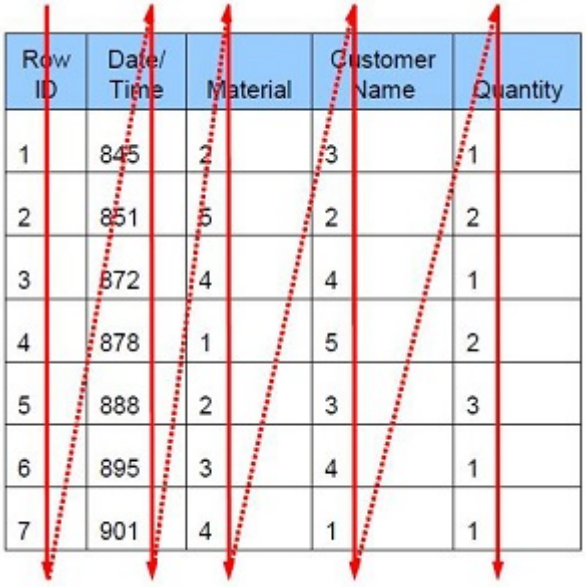


Figura 4 – Exemplo de leitura em uma tabelas orientada a linhas. Fonte: elaborado pelo autor.



onde os valores dos atributos de uma tabela são armazenados lado-a-lado. Os sistemas de bases de dados analíticos, por sua vez, analisam atributos selecionados de conjuntos de dados volumosos de maneira extremamente rápida. Se os dados completos de uma tupla necessitam ser acessados, o armazenamento orientado a linhas é mais vantajoso. Por exemplo, na comparação entre dados de dois clientes, todos os atributos desses dois clientes

necessitam ser carregados para comparação. Em contrapartida, bases de dados orientadas a coluna se beneficiam do seu formato de armazenamento de dados quando um subconjunto dos atributos necessita ser processados para todas, ou para um numeroso conjunto de entradas da tabela da base de dados. Por exemplo, a soma do número total de todos os produtos que passaram por um certo portão de leitura envolvem somente os atributos de data e locação e ignoram todos os outros como código de identificação do produto, status da produção, etc. Usar o armazenamento em linha para esse propósito resultaria no processamento de todos os atributos da tabela, mesmo quando somente dois atributos são requisitados. Segundo (PLATTNER; ZEIER, 2012), incorporar um armazenamento em coluna traz o benefício de acessar somente os dados relevantes e de usar menos operações de pulo na procura por identificadores de tupla, também chamada de search skipping operations.

2.3 Compressão de dados em tabelas orientadas a colunas

Intuitivamente, os dados armazenados em colunas são mais comprimíveis do que os dados armazenados em linhas. Segundo (ABADI; MADDEN; HACHEM, 2008),

> os algoritmos de compressão têm melhor desempenho em dados com informações de baixa entropia. Tomemos, por exemplo, uma tabela de banco de dados que contém informações sobre os clientes, tais como nome, número de telefone, endereço de e-mail, endereço. O armazenamento de dados em colunas permite que todos os nomes sejam armazenados em conjunto, assim como todos os números de telefone e assim por diante. Certamente os números de telefone são mais semelhantes entre si do que campos de texto ao redor, como endereços de e-mail ou nomes.

Um benefício adicional acoplado a compressão é o potencial para um desempenho aprimorado nas operações de consultas. Isso se aplica especialmente para o caso da compressão leve (light-weight compression) onde o custo do tempo utilizado pela CPU para a descompressão não ultrapassa o ganho de enviar uma maior quantidade de dados comprimidos entre a memória principal e a CPU. Segundo (PLATTNER; ZEIER, 2012), "atual tendência de desenvolvimento de hardware aponta para uma lacuna cada vez maior entre a largura de banda da memória e o desempenho da CPU, o que favoreceria a utilização de técnicas de compressão cada vez mais pesadas".

Segundo (ABADI; MADDEN; FERREIRA, 2006), a manipulação de dados que sofreram compressão utilizando algoritmos específicos para armazenamento em colunas apresentou melhora no desempenho em ordens de magnitude.

3 Arquitetura SAP HANA

3.1 SQLScript

O SQLScript é um recurso do SAP HANA que fornece uma extensão do padrão SQL ANSI. O SQLScript dispõe de extensões funcionais e processuais e possibilita definir variáveis locais para estruturas e tabelas, que são utilizados principalmente para a criação de procedimentos armazenados. O SQLScript também pode ser empregado para cálculos e para combinar views analíticas. Conceitualmente, o SQLScript é relacionado a um procedimento armazenado, conforme definição dos padrões SQL.

Segundo (HARPENSCHLAGER, 2014), o SQLScript é relativamente desconhecido e ainda encontra cepticismo dos desenvolvedores. Porém, ele é parte indispensável para

utilizar o potencial completo do SAP HANA. O SQLScript fornece elementos de linguagem para migrar operações pesadas e intensas do servidor de aplicação para o banco de dados.

Segundo (SAP, 2014b), que especifica o funcionamento do SQLScript, o SQL clássico requer que cada resultado seja enviado para o servidor de aplicação. Comando complexos, como aqueles que contêm JOINS, são frequentemente evitados, pois eles são difíceis de elaborar e compreender. Por esse motivo, frequentemente os resultados são quebrados em vários comandos SQL individuais. Isso não é eficiente, pois é necessário executar varias transações entre o servidor de aplicação e o banco de dados. Não raramente são selecionados conjuntos de dados desnecessários, enviando dados não essenciais do banco de dados para a camada de aplicação. Em contraste a isso, o SQLscript conecta essas transações em uma só, transformando-as em um único comando complexo. Adicionalmente, o compilador do SQLScript certifica-se que a execução seja paralelizada ao máximo (SAP, 2014b), aproveitando todo o potencial de processamento paralelo oferecido pela infra-estrutura do SAP HANA .

Os sistemas em ABAP, principal linguagem utilizada pela SAP, podem continuar rodando no SAP HANA sem quaisquer alterações em seu código. Contudo, a migração sem ajustes nas aplicações significa uma grande perda de potencial do SAP HANA. O SQLScript pode oferecer muitas vantagens, mas elas são melhor aproveitadas se houver um ajuste nos aplicativos para certificar-se que, idealmente, toda a logica de cálculo de dados rode no SAP HANA, transmitindo somente os resultados finais para o servidor de aplicação.

Os resultados de um SQL clássico, por sua vez, necessitam ser enviados diretamente para o servidor de aplicação ou capturados globalmente em views. O SQLScript remedia essa situação através da declaração de variareis para armazenar dados. Além disso, o SQLScript oferece procedimentos que são reutilizáveis, ao contrário das views em SQL. Segundo (BINNIG; MAY; MINDNICH, 2013), as consultas SQL declarativas clássicas não oferecem recursos para suportar lógicas de negócio complexas, quando comparado com procedimentos SQLScript. Apenas as chamadas a funções definidas pelo usuário podem oferecer apoio a isso, porém essas funções são implementadas utilizando SQL imperativo e, portanto, não podem ser optimizadas e paralelizadas eficientemente. A extensão funcional do SQLScript é declarativa e, portanto, suporta paralelização e optimização, conforme (SAP, 2014b).

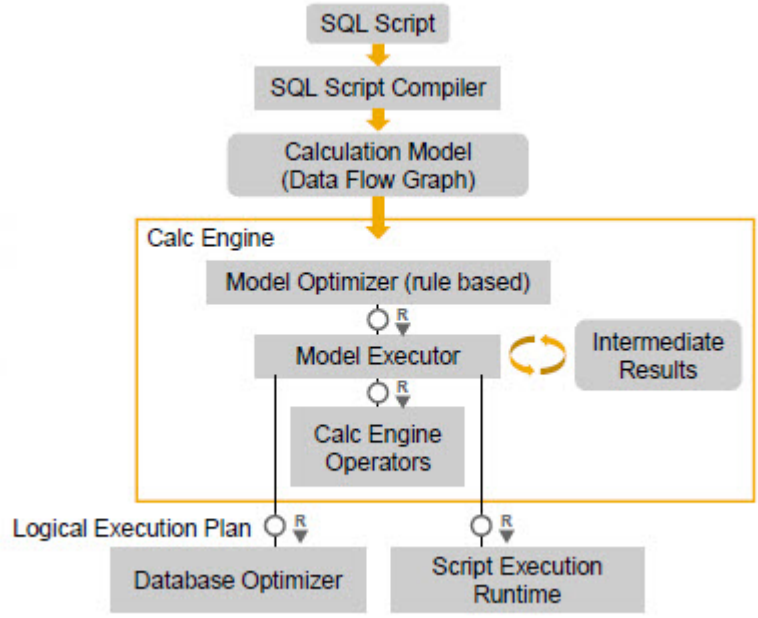
A execução de SQLScript ocorre da seguinte forma: o comando é analisado e desmembrado ao máximo para obter seu contexto e sua sintaxe e, logo após, os elementos são disponibilizados para o Calculation Engine. O Calculation Engine é o mecanismo de execução pra os SQLs, conforme visto na figura 5.

É possível utilizar consultas SQL e funções CE dentro de um SQLScript. O termo CE é abreviação de Calculation Engine Plan Operators, ou operadores de plano do mecanismo de cálculo. Elas podem conter parâmetros de entrada e saída, encapsulando operações de obtenção e transformação de dados.

As operações SQL também podem ser utilizadas dentro de um SQLScript, armazenando seu resultado dentro de variáveis. O otimizador de SQLs é o componente responsável por analisar e otimizar os SQLs para serem executados no melhor plano possivelmente calculado.

As SQLs, quando utilizados dentro de um procedimento SQLScript, exige uma mudança constante entre o Calculation Engine e o otimizador de SQLs . Para mitigar

Figura 5 – Arquiterura de execução de consultas do SAP HANA. Fonte: (SAP, 2014b)



esse problema, a arquitetura do SAP HANA foi desenvolvida para executar as funções CE diretamente no Calculation Engine, aumentando o desempenho na obtenção e transformação dos dados. Além disso, funções específicas para sistemas analíticos estão disponíveis para simplificar os procedimentos SQLScript.

4 Experimentos

4.1 Metodologia

O presente trabalho pode ser classificado como uma pesquisa do tipo quantitativa, que tem como objetivo analisar o desempenho de consultas na plataforma SAP HANA. Busca-se, através do estudo, demonstrar os recursos da mesma.

4.2 Especificações técnicas dos experimentos

O SAP HANA, acrônimo para High Performance Analytic Appliance, é um appliance. Um appliance é um sistema desenvolvido em conjunto com um hardware específico, combinando componentes de software otimizados pela SAP com hardwares homologados e providos por parceiros.

Foram utilizados dois appliances SAP HANA, ambos localizados na sede da SAP, na Alemanha. Eles fazem parte da infraestrutura de computação nas nuvens da SAP, chamado de SAP Hana Cloud. O SGDB utilizado para gerenciamento dos appliances foi o SAP HANA Studio.

Para o experimento número um, utilizou-se a versão do SAP HANA 1.00 SP06, revisão de número 60. O sistema operacional é o SUSE Linux Enterprise Server 11.1 de 64 bits. O servidor utilizado nos experimentos possui 4 CPUs Intel Xeon E7 – 4830 (2.13 Ghz), cada um com 8 núcleos, 24mb de cache e 49 GB de memoria RAM.

Para o experimento número dois, utilizou-se a versão do SAP HANA 1.00 SP08, revisão de número 80. O sistema operacional dos appliances é o SUSE Linux Enterprise Server 11.3 de 64 bits. O servidor utilizado para os experimentos possui 2 CPUs Intel Xeon E7 – 4830 (2.13 Ghz), cada um com 8 núcleos, 24mb de cache e 32 GB de memoria RAM.

Apesar da localização do servidor, o tempo de resposta medido nos experimentos não será afetado, uma vez que os resultados dos testes são exibidos como tempo de resposta do servidor e desconsideram o tempo de tráfego de rede e de processamento do SAP HANA Studio.

De acordo com (FRIESS; GEISSLER, 2012) que especifica e estuda quais são as ferramentas para aferição do desempenho da execução de comandos no SAP HANA, o SAP HANA Studio é uma das ferramentas recomendadas.

5 Experimento 1: comparação de desempenho de consultas em tabelas orientadas a linha e a colunas

Baseado nos estudos deste trabalho foi apontado que a orientação por linhas e colunas possuem vantagens e desvantagens. O experimento visa analisar o desempenho da leitura em bases de dados transacionais e analíticas com tabelas orientadas a linhas e a colunas.

5.1 Especificações do experimento

Os experimentos foram executados utilizando três catálogos distintos no SAP HANA. Os dois primeiros catálogos, TCC_C e TCC_R, foram criados e preenchidos com dados aleatórios de testes com base no algoritmo de (ELLIOTT, 2013). Ambos contêm as mesmas tabelas, valores e atributos, sendo que um catálogo armazena as tabelas seguindo a orientação a linhas (TCC_R) e o outro catálogo com tabelas orientadas a coluna (TCC_C). O modelo das tabelas segue a modelagem dimensional, com o objetivo de simular um sistema analítico (OLAP). As especificações da tabela podem ser vistos na figura 7. Cada execução foi feita em um volume crescente de dados. O critério de decisão sobre o volume de dados foi baseado na capacidade do algoritmo de (ELLIOTT, 2013) em gerar dados em um tempo máximo de 4 horas.

O terceiro catálogo, chamado SFLIGHT, é um catálogo fornecido pela SAP para testes. O modelo das tabelas simula um sistema transacional (OLTP). Ele contêm tabelas com dados simulados de aeroportos, aeronaves, vendas de passagens, informações de voos e outras informações relacionadas. Para o experimento foi utilizada uma tabela orientada a colunas chamada SBOOK, que contêm 1.374.477 registros com informações sobre reservas de voos. O catálogo não foi gerado por um algoritmo gerador de dados de teste, pois o objetivo do experimento foi selecionar um conjunto específico de dados já existes na base de dados. Por esse motivo, não foi experimentado a execução em diferentes volumes de dados.

A tabela SBOOK, orientada a colunas, foi copiada para uma tabela orientada a linhas chamada SBOOK_R, mantendo os mesmos valores e atributos. O comando utilizado está ilustrado na figura 6.

Figura 6 – Comando de criação da tabela SBOOK_R. Fonte: elaborada pelo autor.

```
CREATE ROW TABLE "SFLIGHT"."SBOOK_R" as (SELECT * FROM "SFLIGHT"."SBOOK")
```

5.2 Execução do experimento

Foi escolhido uma consulta SQL para obter um conjunto de dados das tabelas contidas nos catálogos TCC_C e TCC_R. A consulta escolhida para o experimento consiste em uma pesquisa para obter o volume de vendas de um grupo de produtos em uma unidade de negócio. A consulta utilizada pode ser vista na figura 8. Em seguida, a consulta foi executada nos catálogos contendo tabelas orientadas a linha e orientadas a coluna. Foram executadas 3 rodadas de execuções com 100 repetições cada uma. Cada rodada de execuções foi efetuada no mesmo catálogo com diferentes volumes de dados, cuja especificação pode ser vista na figura 7.

Figura 7 – Informações das tabelas dos catálogos utilizadas no experimento. Fonte: elaborada pelo autor.

	Execução 1	Execução 2	Execução 3	Tipo de informação
BUSINESS_UNIT_D	8	8	8	unidades de negócio
MATERIAL_D	500.000	750.000	1.000.000	grupo de materiais
SUPPLIER_D	5.000	7.500	10.000	fornecedores
SALES_F	500.000	750.000	1.000.000	fatos das vendas

Figura 8 – Consulta realizada no experimento. Fonte: elaborada pelo autor.

```
SELECT sum(sa.unit_price*sa.quantity_sold), ma.sku, sa.calendar_day
FROM sales_f AS sa,
     material_d AS ma,
     supplier_d AS s,
     business_unit_d as bu
WHERE sa.material_id = ma.material_id
  AND sa.business_unit_id = bu.business_unit_id
  AND ma.material_group = 'Components'
  AND bu.business_unit_code = 'BU1'
  GROUP BY sa.calendar_day, ma.sku
  ORDER BY sa.calendar_day, ma.sku
```

Figura 9 – Comando de criação da tabela SBOOK_R. Fonte: elaborada pelo autor.

```
SELECT * FROM SBOOK_R
WHERE "MANDT" = 300 and "CARRID" = 'LH'
AND "CONNID" = 0401 and "FLDATE" = 20110928


SELECT * FROM SBOOK
WHERE "MANDT" = 300 and "CARRID" = 'LH'
AND "CONNID" = 0401 and "FLDATE" = 20110928
```

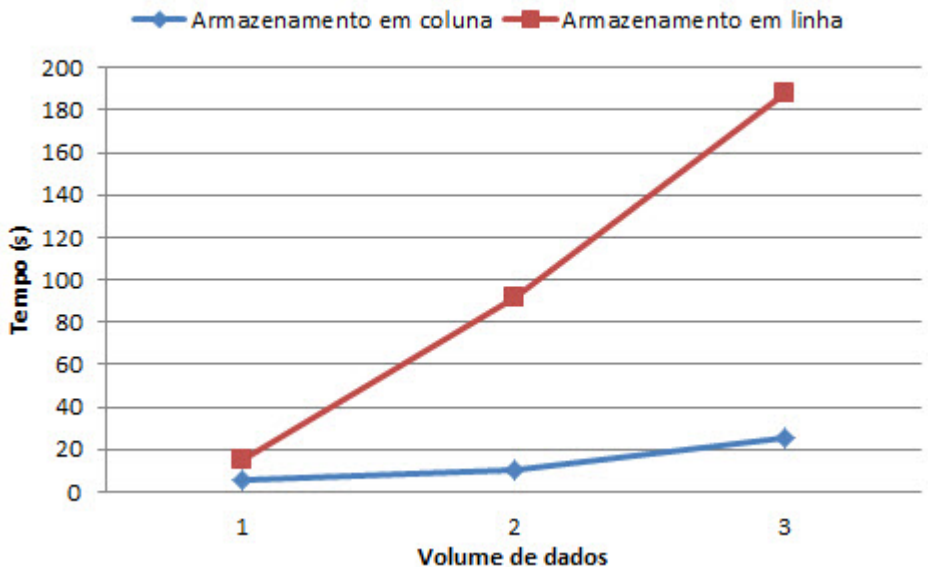
Foram executadas 100 execuções para obter dados dos voos, filtrando pela companhia aérea, data e código de conexão. As consultas, ilustradas na figura 9, foram executadas nas tabelas SBOOK e SBOOK_R.

Os resultados obtidos da execução da consulta feita nos catálogos TCC_C e TCC_R podem ser vistos na figura 10 e o gráfico na figura 11. A leitura e agregação de uma coluna foram, em média, cerca de 195% mais rápida na tabela orientada em colunas na execução 1, 762% na execução 2 e 636% na execução 3.

Figura 10 – Média de tempo de execução dos experimentos executados nos catálogos TCC_C e TCC_R . Fonte: elaborada pelo autor.

	Execução 1	Execução 2	Execução 3
Armazenamento em coluna	5,08 s	10,59 s	25,49 s
Armazenamento em linha	15,03 s	91,34 s	187,86 s

Figura 11 – Gráfico de resultados do experimento executados nos catálogos TCC_C e TCC_R. Fonte: elaborada pelo autor.



Foi experimentado o comportamento explanado no capitulo 2.2 sobre as vantagens e desvantagens de cada esquema de armazenamento. Era esperado que o catálogo com tabelas orientadas a coluna obtivesse um desempenho superior ao agregar e ler os dados de colunas inteiras. Os resultados deste experimento validam a ideia de que quando a aplicação necessita selecionar e agregar os dados de uma ou mais coluna, típico de sistemas analíticos (OLAP), a orientação das tabelas em coluna oferece vantagens. Por fim, conforme vimos no capítulo 2.3, os dados armazenados em colunas são mais comprimíveis do que os dados armazenados em colunas, o que pode oferecer ganhos de desempenho em ordens de magnitude.

Os resultados obtidos da execução da consulta feita no catálogo SFLIGHT podem ser vistos na figura 12. A consulta na tabela orientada a linhas foi cerca de 15 vezes mais rápida do que a consulta na tabela orientada a colunas. Os resultados validam a ideia de que as tabelas armazenadas em linhas levam vantagem quando a aplicação que as utiliza necessita selecionar registros únicos em cada pesquisa a base de dados, típico de sistemas analíticos (OLTP).

Muitos fatores podem exercer uma influência sobre os tempos de resposta de uma pesquisa como, por exemplo, como a mesma é estruturada, número de valores distintos em cada coluna, taxa de compressão da coluna, número de entradas na tabela e quantidade de atributos da tabela. O estudo de (SIKKA et al., 2012) também demonstra que o SAP HANA é capaz de lidar de maneira eficaz com ambientes analíticos e transacionais, através de uma análise detalhada da estrutura de processamento interna do SAP HANA.

Figura 12 – Resultados dos experimentos no catálogo SFLIGHT . Fonte: elaborado pelo autor.

	Tipo de Armazenamento	Média de Tempo (ms)
SBOOK	colunas	161,88 ms
SBOOK_R	linhas	9,81 ms

A habilidade de um sistema em suportar os dois tipos de armazenamento parece ser o mais indicado, pois traz mais flexibilidade ao desenvolvedor, que poderia criar tabelas tanto em linhas ou colunas, podendo se beneficiar das vantagens de um ou de outro modelo. De acordo com a finalidade e o tipo de leitura, é possível utilizar tabelas orientadas a linha ou a coluna, aproveitando assim as vantagens oferecidas em ambos cenários.

6 Experimento 2: comparação de desempenho de SQLs e procedimentos SQLScript

6.1 Especificações do experimento

Baseado nos estudos deste trabalho, foi apontado que o uso da linguagem SQLscript pode oferecer vantagens de desempenho e para execução de operações no SAP HANA. Este experimento visa demonstrar a diferença de desempenho entre o uso de três consultas distintas que retornam os mesmos dados e atributos. A primeira versão da consulta utiliza comando SQL, a segunda versão utiliza um procedimento SQLScript contendo comandos SQL e a terceira versão utiliza um procedimento SQLscript contendo funções CE.

Foi utilizado o conteúdo entregue pelo pacote educacional chamado SAP HANA Interactive Education (SHINE), que é um pacote educacional para implementar e desenvolver aplicações através do SAP HANA Studio. O conteúdo do SHINE é desenvolvido com base em um sistema EPM (Enterprise Procurement Model), incluindo todos os modelos de dados, tabelas, views, dashboards, a fim de fornecer um caso de uso real de uma organização. A instalação do SHINE no SAP HANA foi feita com base na publicação de (SAP, 2014a). O conteúdo do SHINE pode ser instalado através do download do pacote SAP HANA DEMO MODEL. No experimento foi utilizado o pacote versão 1.0 com o pacote de correções versão 8.4.

Um experimento semelhante foi executado por (HARPENSCHLAGER, 2014), que comparou a performance de SQLScripts contendo comandos SQL e SQLScripts contendo funções CE para obter o mesmo conjunto de dados contendo os mesmos atributos, também baseado em dados do EPM do SHINE. Foram analisadas as tabelas "businessPartner", "purchaseOrder", purchaseOrderItem"e "products". No estudo foram criados 20 milhões de registros utilizando o gerador de dados fornecido pelo pacote. Para o experimento foram utilizados dois procedimentos SQLScript para ler dados de duas views analíticas, juntando-as e realizando operações antes da exibição dos dados, um dos procedimentos contendo funções CE e o outro com SQLs. Através de uma execução, o autor mostrou que houve um maior desempenho na execução do procedimentos contendo funções CE. O estudo ainda conclui que o SQLScript oferece um grau menor de complexidade e um nível maior de transparência quando comparado as consultas SQL.

Contudo, com o objetivo de obter resultados comparativos mais amplos, neste experimento comparamos o desempenho não somente de um procedimento SQLScript com

SQL e funções CE, mas também de uma consulta SQL. Além disso, o experimento do presente estudo realiza testes com diferentes cargas de trabalho através da variação do volume de dados e com maior número de execuções.

6.2 Execução do experimento

Foram escolhidas as views AT_PURCHASE_ORDER_WORKLIST e AT_PO_ITEM, do catálogo _SYS_BIC. Através do uso do gerador de dados disponibilizado pelo pacote SAP HANA DEMO MODEL foram gerados registros de ordens de compra e ordens de venda. Foram executadas 3 rodadas de execuções com 100 repetições cada uma. Cada rodada de execuções foi efetuada no mesmo catálogo com diferentes volumes de dados, cuja especificação pode ser vista na figura 13. Em seguida, foram elaborados os procedimentos e a consulta SQL, sendo que todas retornam o mesmo conjunto de dados e os mesmos atributos. O comando SQL pode ser visto na figura 16. O procedimento SQLscript contendo comandos SQL pode ser vista na figura 17. O procedimento SQLscript contendo funções CE pode ser visto na figura 18.

Figura 13 – Número de registros em cada execução. Fonte: elaborado pelo autor.

	Execução 1	Execução 2	Execução 3
Ordens de Compra	5.880	593.880	1.181.880
Ordens de Venda	6.046	594.046	1.182.046

Figura 14 – Chamada da função SQLScript contendo funções CE. Fonte: elaborado pelo autor.

```
CALL "_SYS_BIC"."Z_SQLSCRIPT_PROCEDURE_CE"  (9, av_proj => ?)
```

Figura 15 – Chamada da função SQLScript contendo SQLs. Fonte: elaborado pelo autor.

```
CALL "_SYS_BIC"."Z_SQLSCRIPT_PROCEDURE_SQL"  (9, av_proj => ?)
```

Figura 16 – Consulta SQL.

```
SELECT
A."PARTNERID",
A."COMPANYNAME",
A."PURCHASEORDERID",
A."ORDERINGDESC",
B."GROSSAMOUNT",
B."QUANTITY",
B."CURRENCY",
(B."GROSSAMOUNT" + (B."GROSSAMOUNT" * 9) / 100 ) as "GROSSPLANNED"
FROM "_SYS_BIC"."sap.hana.democontent.epm.models/AT_PURCHASE_ORDER_WORKLIST" A,
     "_SYS_BIC"."sap.hana.democontent.epm.models/AT_PO_ITEM" B
WHERE A."PURCHASEORDERID" = B."PURCHASEORDERID"
AND B."GROSSAMOUNT" > 100
GROUP BY B."GROSSAMOUNT", A."CURRENCY", A."PARTNERID", A."COMPANYNAME",
         A."PURCHASEORDERID", A."ORDERINGDESC", A."QUANTITY", B."QUANTITY", B."CURRENCY",
         A."GROSSAMOUNT", B."PRODUCTID", B."DELIVERYDATE", B."PURCHASEORDERID";
```

Figura 17 – Procedimento SQLScript utilizando comandos SQL. Fonte: elaborado pelo autor.

```
CREATE PROCEDURE Z_SQLSCRIPT_PROCEDURE_SQL
(in percentage decimal(10,2), out av_proj tt_data_sql )
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER
READS SQL DATA AS
BEGIN

av1 = SELECT "PARTNERID", "PURCHASEORDERID", "COMPANYNAME", "ORDERINGDESC", "CURRENCY"
      FROM  "_SYS_BIC"."sap.hana.democontent.epm.models/AT_PURCHASE_ORDER_WORKLIST"
      GROUP BY "CURRENCY", "PARTNERID", "COMPANYNAME",
               "PURCHASEORDERID", "ORDERINGDESC", "QUANTITY";

av2 = SELECT "DELIVERYDATE", "PURCHASEORDERID", "QUANTITY", "GROSSAMOUNT","PRODUCTID"
      FROM "_SYS_BIC"."sap.hana.democontent.epm.models/AT_PO_ITEM"
      GROUP BY "PRODUCTID", "DELIVERYDATE", "PURCHASEORDERID", "QUANTITY", "GROSSAMOUNT"

av_join = SELECT O."PARTNERID", O."COMPANYNAME", O."PURCHASEORDERID", O."ORDERINGDESC",
                 I."GROSSAMOUNT", I."DELIVERYDATE", O."CURRENCY"
          FROM :av1 as O, :av2 as I
          WHERE O."PURCHASEORDERID" = I."PURCHASEORDERID";

av_proj = SELECT "PARTNERID", "COMPANYNAME", "PURCHASEORDERID", "ORDERINGDESC",
                 "GROSSAMOUNT", "CURRENCY",
                 ("GROSSAMOUNT" + ("GROSSAMOUNT" * :percentage) / 100 ) as GROSSPLANNED,
                 "DELIVERYDATE"
          FROM :av_join
          WHERE "GROSSAMOUNT" > '100';

END;
```

Figura 18 – Procedimento SQLScript utilizando comandos CE. Fonte: elaborado pelo autor.

```
CREATE PROCEDURE Z_SQLSCRIPT_PROCEDURE_CE
(in percentage decimal(10,2), out av_proj tt_data2 )
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER
READS SQL DATA AS
BEGIN

av1 = CE_OLAP_VIEW("_SYS_BIC"."sap.hana.democontent.epm.models/AT_PURCHASE_ORDER_WORKLIST",
["PARTNERID",
"PURCHASEORDERID",
"COMPANYNAME",
"ORDERINGDESC",
"CURRENCY"]);

av2 = CE_OLAP_VIEW("_SYS_BIC"."sap.hana.democontent.epm.models/AT_PO_ITEM",
["DELIVERYDATE",
"PURCHASEORDERID",
"QUANTITY",
"GROSSAMOUNT",
"PRODUCTID"
]);

av_join = CE_JOIN(:av1,:av2,["PURCHASEORDERID"]);

av_proj = CE_PROJECTION(:av_join,
["PARTNERID",
"COMPANYNAME",
"PURCHASEORDERID",
"ORDERINGDESC",
"GROSSAMOUNT",
"CURRENCY",
CE_CALC('"GROSSAMOUNT" + ("GROSSAMOUNT" * :percentage) / 100', decimal(10,2)) as GROSSPLANNED,
"DELIVERYDATE"],
'"GROSSAMOUNT" > 100');

END;
```

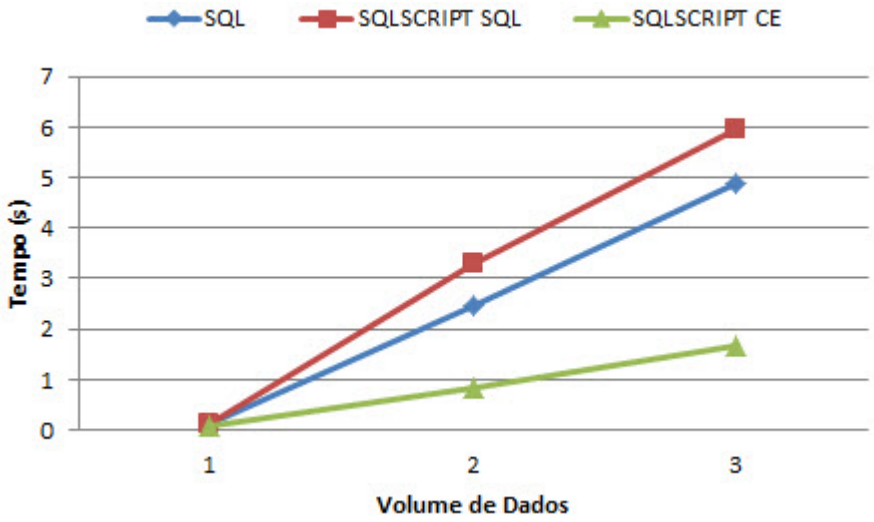
Figura 19 – Média de tempo de execução dos experimentos. Fonte: elaborado pelo autor.

	Execução 1	Execução 2	Execução 3
SQL	0,138 s	2,46 s	4,89 s
SQLScript SQL	0,122 s	3,28 s	5,94 s
SQLScript CE	0,075 s	0,831 s	1,65 s

O experimento consistiu em executar as consultas e obter a média de tempo entre as execuções. A execução dos procedimentos SQLScript foi feita através dos comandos ilustrados nas figuras 14 e 15. A execução do SQL foi feita através do comando ilustrado na figura 16.

Os tempos de execução podem ser visto na figura 19. Apesar das três consultas retornarem os mesmos resultados, elas variam significativamente no seu tempo de execução. As consultas SQL levaram, em média, 0,138 segundos, 2,46 segundos e 4,89 segundos. O SQLScript com comandos SQL levou, em média, 0,122 segundos, 3,28 segundos e 5,94

Figura 20 – Grafico dos resultados do experimento. Fonte: elaborado pelo autor.



segundos. Por fim, o SQLScript com funções CE levou, em média, 0,075 segundos, 0,831 segundos e 1,65 segundos.

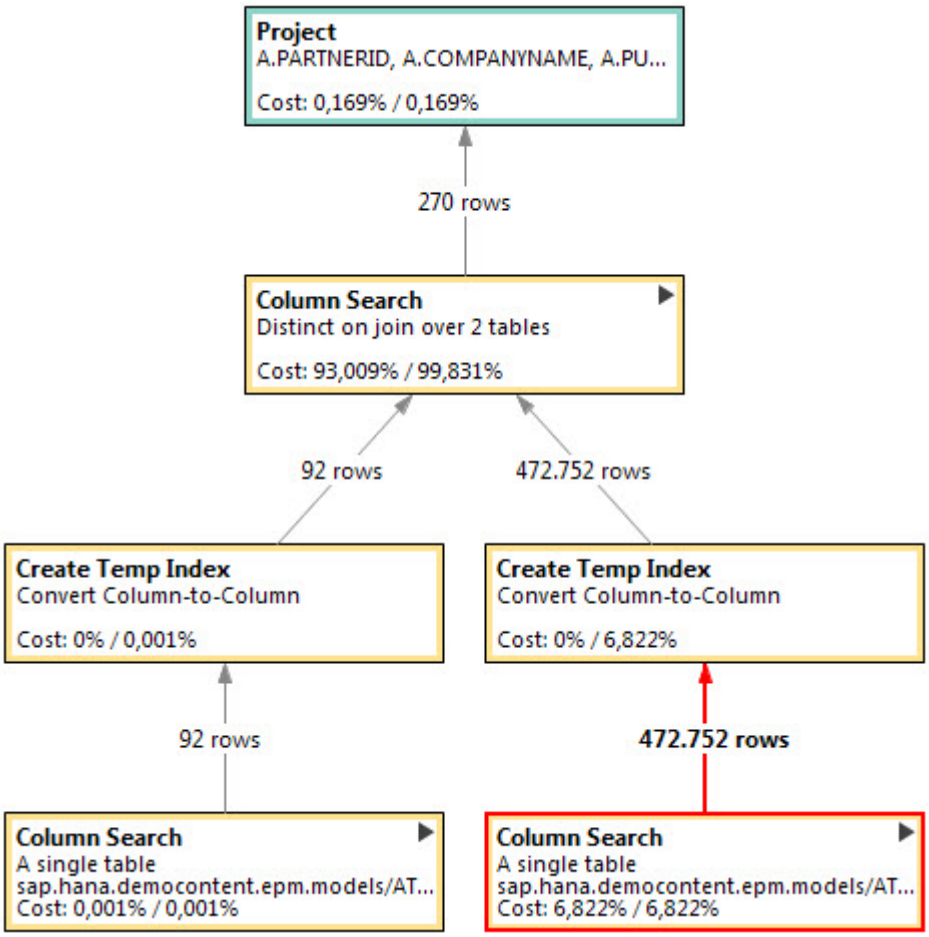
Conforme visto no capítulo 3.1, as consultas SQLs dentro de um SQLScript exige uma mudança constante entre o otimizador de SQLs, que é um componente responsável por analisar e otimizar os SQLs para serem executados no melhor plano possivelmente calculado.

Quando comparamos o procedimento SQLScript contendo SQLs e a consulta SQL , temos duas grandes diferenças. A primeira é um custo maior de execução pois o procedimento deve ser chamado e preparado e, logo após lançada em uma tabela temporária. Essa tabela, chamada de table type, pode ser vista na declaração da função na figura 10. As table types armazenam dados temporários que não são persistidos no banco de dados. A consulta SQL, por sua vez, executa diretamente sem nenhum custo adicional além do custo do otimizador SQL. A segunda diferença é que as operações SQLs dentro do procedimento SQLScript selecionam um conjunto inteiro de dados e, logo após, realiza uma operação de JOIN, conforme visto na figura 9 na variável av_join é realizada a junção de duas variáveis, av1 e av2, onde ambas contem um comando de seleção completo, pois não contém nenhum critério de seleção (clausula WHERE). Apesar do otimizador SQL atuar no SQLScript, fazendo com que não obrigatoriamente todos os dados sejam selecionados, não é garantia que a execução seja mais rápida que a consulta SQL. A consulta SQL é, por sua vez, executada de maneira otimizada, pois o otimizador escolhe o melhor plano possivelmente calculado.

O plano calculado e escolhido para o SQL pode ser visto na figura 21. Foi feita uma busca nas colunas da tabela AT_PURCHASE_ORDER_WORKLIST e AT_PO_ITEM, retornando 92 e 472,752 registros, respectivamente. Em seguida foram criados dois índices temporários e duas tabelas temporárias contendo o mesmo número de registros. Foi realizado um INNER JOIN das tabelas, resultando em 284 registros e, logo após, foram selecionados os dados distintos conforme os critérios de agrupamento, resultando em 270 registros. Com base no resultado, foi criado uma projeção e o resultado foi retornado.

Por fim, quando comparamos os tempos de execução do procedimento SQLScript contendo SQLs e a execução do procedimento SQLScript contendo funções CE, o último

Figura 21 – Plano de execução calculado pelo otimizador SQL na execução 3. Fonte: elaborado pelo autor.



executa entre 0,62 vezes mais rápido na primeira execução, 3,9 vezes mais rápido na segunda execução e 3,6 mais rápido na terceira execução, como foi esperado. Conforme visto no capítulo 3.1, as funções CE são diretamente integradas no Calculation Engine, evitando assim a troca de contexto entre o otimizador SQL e o Calculation engine. Além disso, segundo as especificações do SAP HANA (SAP, 2014b), o uso de funções CE torna o procedimento SQLScript passível de sofrer paralelização.

Neste experimento podemos ver todos os beneficios que o SQLScript oferece em relação ao SQL. Vemos também quais são as vantagens de utilizar as funções CE quando comparadas com SQLs dentro de procedimentos SQLSCript. Por fim, conforme mencionado por (HARPENSCHLAGER, 2014), concluímos que as consultas têm um nível de complexidade menor e são mais transparentes e, ao mesmo tempo, têm maior performance. Sendo assim, quanto maior a complexidade e quanto maior o volume de dados, é recomendado utilizar procedimentos SQLScript.

Considerações finais

Atualmente muitas empresas sofrem com sistemas ineficientes, incapazes de fornecerem informações em tempo útil. A tendência é de que essa situação se agrave, pois

estamos em uma era na qual a quantidade de dados de negócio tende a dobrar a cada dois anos. Na grande maioria das empresas, o principal gargalo entre a solicitação de dados e a entrega dos mesmos em forma de informação aos usuários é justamente a leitura de dados no disco rígido.

Enquanto isso, as novas tendências de hardware apontam cada vez mais para servidores com uma crescente capacidade de processamento e endereçamento de memória RAM com um custo cada vez menor. As bases de dados em memória utilizam essa tendência em seu favor, utilizando grandes quantidades de memória e o aproveitando ao máximo os recursos de processamento paralelo.

Sendo assim, as IMDBs visam acabar com o gargalo da leitura em disco rígido, pois armazenam os dados na memória principal, que é ordens de grandeza mais rápida do que a leitura de dados no disco rígido. Ainda fazem melhor uso da capacidade de processamento paralelo ao conseguir particionar verticalmente colunas das tabelas com armazenamento orientado em colunas.

Ao fornecer informações de maneira mais rápida, as IMDBs aumentam a produtividade dos usuários finais e tomadores de decisão. Ainda tendem a diminuir o custo total da solução quando comparada com outros meios, principalmente pelo fato de eliminar, ou diminuir, a necessidade de aplicar ajustes no sistema para melhora de desempenho.

Na pesquisa sobre o SAP HANA e suas principais funcionalidades, vimos que que algumas funções são de extrema importância e potencializam ainda mais o poder das mesmas. Nos experimentos demonstramos que o SAP HANA reúne muitas funções e características que auxiliam a lidar com carga de trabalhos mistas, através do uso de armazenamento orientado a linhas ou colunas e a utilização de procedimentos SQLScript para ganho de desempenho, redução de complexidade e aumento da paralelização.

O presente estudo de reforçou e expandiu os estudos sobre o SQLScript já efetuados por (HARPENSCHLAGER, 2014) e mostrou que a migração da lógica do negócio para o banco de dados pode oferecer maior desempenho na obtenção dos dados, maior modularização, aproveitamento e entendimento dos fluxos de obtenção de dados. Através do uso de recursos como o SQLScript, é possível explorar ao máximo o potencial do SAP HANA e, ao mesmo tempo, reduzir a complexidade das aplicações.

Finalmente, de acordo com o material pesquisado e relatado neste artigo, pode-se afirmar que a evolução de hardware, em conjunto com novas técnicas de codificação e tecnologias de armazenamento de dados, está revolucionando a maneira como as empresas usufruem das vantagens proporcionadas pela TI. Com base nisso, pode-se esperar que as IMDBs tornem-se cada vez mais comuns no meio empresarial e no futuro irão tornar-se bases de dados padrão para toda e qualquer empresa.

Com o objetivo de continuar os estudos sobre as vantagens do uso do armazenamento orientado a colunas e a linhas, assim como o uso de procedimentos SQLScript, recomendamos estudos futuros sobre o ganho de desempenho de aplicações reais e, também, estudos que sobre como realizar a otimização de aplicações para aproveitar o potencial oferecido pelo SAP HANA.

Comparative studies of different query plans in SAP HANA

Guilherme Balbinot <gbalbinot@gmail.com> *
Denise Bandeira <bandeira@unisinos.br> †

16 de dezembro de 2014

Abstract

Analytical systems are of utmost importance in the decision making processes. The data volume growth brought the challenge to deal with increasing data volumes, efficiently and fast. The users want the get the information faster and in real time, rather than waiting for lengthy reports that are extracted only at certain times of the month. The use of in-memory databases can solve such problems. The use of in-memory data storage enables performance gains, as the manipulation of in-memory data is significantly faster than the manipulation of hard disk data. Features such as column storage enables data compression, further increasing the data read speed. In addition, it's possible to perform complex queries using functions and procedures to achieve better modularization, paralelization, simplification and performance. This paper aims to study in-memory databases and its functionalities. This work also aims at studying the SAP HANA platform to study and explore some of its functionalities, demonstrating the benefits that such tools can provide to improve performance in data retrieval.

Key-words: in-memory databases. in-memory computing.

Referências

ABADI, D. J.; MADDEN, S. R.; FERREIRA, M. C. Integrating compression and execution in column-oriented database systems. p. 671–682, 2006. Citado na página 5.

ABADI, D. J.; MADDEN, S. R.; HACHEM, N. Column-stores vs. row-stores: How different are they really? In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2008. (SIGMOD '08), p. 967–980. ISBN 978-1-60558-102-6. Disponível em: <http://doi.acm.org/10.1145/1376616.1376712>. Citado na página 5.

*Aluno do curso de Sistemas de Informação da Universidade do Vale do Rio dos Sinos
†Profa. Msc. Denise Bandeira, coordenadora do curso de Sistemas de Informação da Universidade do Vale do Rio dos Sinos

BINNIG, C.; MAY, N.; MINDNICH, T. Sqlscript: Efficiently analyzing big enterprise data in SAP HANA. In: *Datenbanksysteme für Business, Technologie und Web (BTW), 15. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme"(DBIS), 11.-15.3.2013 in Magdeburg, Germany. Proceedings*. [s.n.], 2013. p. 363–382. Disponível em: <http://www.btw-2013.de/proceedings/SQLScript%20Efficiently%20Analyzing%20Big%20Enterprise%20Data%20in%20SAP%20HANA.pdf>. Citado na página 6.

ELLIOTT, G. *HANA POC - Generating big data*. 2013. <http://scn.sap.com/thread/3286683>. Acessado em 07/11/2014. Citado na página 8.

FRIESS, S.; GEISSLER, S. *Testing SAP HANA*. [S.l.]: SAP, 2012. 10–12 p. Citado na página 8.

GANTZ, J.; REINSEL, D. *Extracting value from chaos*. [S.l.]: IDC, 2011. Citado na página 1.

HARPENSCHLAGER, M. *SAP HANA speaks SQLScript*. Contrimo Labs, 2014. Disponível em: <http://www.contrimo.com>. Citado 4 vezes nas páginas 5, 11, 16 e 17.

PLATTNER, H.; ZEIER, A. *In-Memory Data Management*. [S.l.]: Springer, 2012. Citado 3 vezes nas páginas 2, 3 e 5.

ROSS, A. *SAP Netweaver BI Accelerator*. [S.l.]: Galileu Press, 2009. Citado na página 2.

SAP. *SAP HANA Interactive Education (SHINE) SPS 08*. [S.l.]: SAP, 2014. Citado na página 11.

SAP. *SAP HANA SQLScript Reference*. [S.l.]: SAP SE, 2014. Citado 3 vezes nas páginas 6, 7 e 16.

SIKKA, V. et al. Efficient transaction processing in sap hana database: The end of a column store myth. In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2012. (SIGMOD '12), p. 731–742. ISBN 978-1-4503-1247-9. Disponível em: <http://doi.acm.org/10.1145/2213836.2213946>. Citado 2 vezes nas páginas 2 e 10.