# Avaliando a Influência da Modelagem Relacional e Colunar no Gerenciamento de Ambientes OLAP por meio do TPC-H

1

***Abstract.*** *Due to data volume growth and companies needs for efficient decision making, chosing the best DBMS to Data Warehouse management has been a challenge. Although traditional DBMS are yet applied to this purpose, NoSQL approach has emerged as an alternative. To evaluate which is the most appropriate, this research shows a comparative study between PostgreSQL and MonetDB, through TPC-H benchmark. While PostgreSQL achieved good results in a normalized environment, MonetDB excelled in general performance.*

***Resumo.*** *Em razão do crescimento do volume de dados e da necessidade de agilidade na tomada de decisão das empresas, escolher o SGBD mais adequado para o gerenciamento de um Data Warehouse tem sido um desafio. Embora os SGBD tradicionais ainda sejam utilizados para esse fim, a abordagem NoSQL tem surgido como alternativa. Para avaliar qual é a mais adequada, este trabalho apresenta um estudo comparativo entre os SGBD PostgreSQL e o MonetDB, realizado por meio do benchmark TPC-H. Embora o PostgreSQL tenha se destacado num ambiente normalizado, o MonetDB apresentou um desempenho geral superior.*

## 1. Introdução

*Data Warehouses* (DW) são utilizados para gerenciar o armazenamento dos dados de um ambiente de tomada de decisões, haja vista o aumento do volume de dados degradar o desempenho do acesso às informações em casos onde não há um gerenciador adequado. Além do armazenamento, aplicações OLAP (*Online Analytical Processing*) são utilizadas para recuperar e processar, eficientemente, esses dados, tal que estejam fortemente associadas ao DW.

Sistemas Gerenciadores de Bancos de Dados (SGBD) surgem como uma solução para esta questão, e duas classes de SGBD podem atuar como gerenciadores de DW, os relacionais e os NoSQL. Para avaliar a melhor abordagem, a aplicação de um *benchmark* torna-se pertinente, e um *benchmark* que se destaca por tratar de sistemas de suporte à decisão é o TPC-H [TPC-H 2017b].

Além do SGBD, outro fator que interfere no desempenho de um ambiente OLAP é a forma como este é modelado. Ambientes normalizados são utilizados pela facilidade de manutenção, e o fácil entendimento acerca do relacionamento entre as entidades, o que traz uma visão mais clara do sistema. Entretanto, modelos denormalizados surgem para trazer ganho no desempenho das consultas, por diminuir as junções entre tabelas devido ao menor número de entidades.

Dadas estas condições, o objetivo aqui é realizar um estudo comparativo entre SGBD relacional e NoSQL como gerenciadores de DW utilizando o TPC-H. Devido a seu

amplo uso e poucas modificações na linguagem SQL, foi definido o PostgreSQL como SGBD relacional e o MonetDB como NoSQL, por este também utilizar SQL, possuir interface simples e ser pioneiro entre os NoSQL colunares.

## 2. Recuperação e Persistência de Dados

Um DW é definido como uma coleção de dados não-volátil, orientado ao assunto da organização, integrado e variante no tempo [Inmon 2005]. É necessário que uma aplicação processe os dados contidos no DW e execute consultas analíticas para acessar um número massivo de registros, visando informações que dêem suporte à tomada de decisão. Devido a isso, histórico, taxa de vazão de consultas e o tempo de resposta são fatores importantes ao processar os dados de um DW.

Aplicações OLAP objetivam identificar tendências, padrões de comportamento e anomalias, e ainda relações entre dados [Codd et al. 1993] sob um grande volume de dados. Suas consultas são longas, consomem grande tempo de execução e estão interessadas em atributos específicos; e o processo de inserção de dados segue um processo conhecido como ETL (extração, transformação e carga), sendo mais complexo que pequenas transações [Zelen 2017]. É neste processo que a modelagem conceitual do ambiente é definida, que tem como base a modelagem dimensional, ou multidimensional, fazendo uma analogia com um cubo para a representação de dados, uma vez que uma informação pode ser vista por $n$ dimensões. Em sua concepção, a modelagem dimensional é composta por tabelas fato e tabelas dimensão.

Fatos representam regras ou métricas de negócio, e geralmente são descritos como atributos numéricos, relacionados a quantidades; ou aditivos, pois uma aplicação nunca irá retornar somente um fato, mas sim vários onde a operação mais recorrente é a soma de atributos. Atributos textuais não definem um fato, e na maioria das vezes apenas descrevem algo e são inseridos nas tabelas dimensão, que contem descrições a fim de tornar os fatos mais claros. Não é incomum encontrar tabelas dimensão com mais de 100 atributos e poucas tuplas, pois a intenção das dimensões é descrever as regras definidas por um fato. Nestas entidades são filtradas as consultas, compreendendo agrupamentos, padrões e ordenações, por exemplo.

Modelos dimensionais no escopo de um DW possuem apenas um fato com várias dimensões, fazendo com que sua estrutura se assemelhe a uma estrela, sendo conhecidos como modelos estrelas (*star*). Este modelo é simples e torna eficiente a recuperação de dados por ter um número menor de junções. Entretanto, com a afirmação de que modelos normalizados são mais fáceis para manutenção, em alguns modelos *star* podem ser criadas novas tabelas denominadas subdimensão. Estes modelos caracterizam modelagens relacionais, e são conhecidos como modelos *snowflake*.

Para comparar o desempenho entre modelagens pode-se utilizar SGBD como gerenciadores de DW, com vantagens como controle de redundância, restrições de acesso, inserção de índices, restrições de integridade, possibilidade de realizar *backup* e restauração de dados, e persistência de dados garantida. Um modelo de SGBD muito conhecido e utilizado é o relacional. Esta classe está atrelada a dados normalizados, fazendo com que se aproxime do modelo *snowflake* de DW, e preza pelos conceitos de ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Modelos assim definidos possuem, porém, algumas limitações quanto à escalabilidade, complexidade e o fato de todas as

informações serem mantidas e recuperadas sob a forma de tuplas [Leavitt 2010]. Tais limitações podem degradar o desempenho de um ambiente analítico.

Para suprir estes problemas SGBD NoSQL podem ser utilizados. Criados com o objetivo de satisfazer algumas deficiências de SGBD relacionais, trabalham com modelagens de dados mais denormalizadas e são classificados de diferentes formas de acordo com sua estruturação. NoSQL tendem a optar por disponibilidade e particionamento de acordo com o teorema CAP (Consistência, Disponibilidade e Partição) [Brewer 2000] para realizar operações de forma mais rápida e permitir escalabilidade. Uma classe de NoSQL que suporta linguagem SQL é o modelo colunar, cujo diferencial é o armazenamento de todas as instâncias de um mesmo atributo junto, e reduz o tempo de acesso e escrita em disco [Matei and Bank 2010, Abadi 2008], retornando apenas os atributos solicitados; e também torna possível a compressão de dados [Abadi et al. 2006] e a eficiência em operações matemáticas de agregação [Matei and Bank 2010].

## 3. TPC-H

O TPC-H é um *benchmark* de suporte à decisão de negócios, representando grandes volumes de dados e executando consultas com um alto grau de complexidade a fim de responder questões críticas de negócio. Ele propõe uma modelagem *snowflake* que pode ser encontrada no manual do TPC-H [TPC-H 2017a].

O TPC-H não apresenta uma modelagem denormalizada de dados similar à *star*. Para tanto, foi construído um ambiente denormalizado a partir do ambiente original do TPC-H para as devidas comparações. Essa construção se fundamentou na criação de um modelo com uma única entidade fato central, unindo atributos de diferentes fatos, descrita por dimensões, e na remoção de tabelas subdimensão.

Em cada modelagem foi executado o teste de desempenho, que é composto de duas execuções: o teste de força e o teste de vazão, representados pelas métricas *Power@Size* e *Throughput@Size*, onde *Size* representa o tamanho da base de dados. Estas duas execuções são utilizadas para calcular o desempenho, que utiliza a unidade de medida *QphH@Size*, que é a quantidade de consultas executada por hora para determinado tamanho de banco de dados.

O teste de força mede a taxa de consultas por hora do sistema com um único usuário ativo. Neste teste são executadas em uma única sessão três instruções: inserção de novos registros, consultas em SQL para recuperar informações, e remoção da mesma quantia de registros inseridos, nesta ordem. O segundo teste, o teste de vazão, mede a capacidade de processar a maior quantidade de consultas no menor intervalo de tempo simulando vários usuários ativos de maneira concorrente, e seu valor é definido pela razão do número total de consultas executadas pelo tempo total da execução do teste.

## 4. A Avaliação Experimental

O estudo foi feito para bases de dados de diferentes tamanhos, estes de 1GB, 10GB e 30GB, a fim de verificar o comportamento dos SGBD conforme o volume de dados. Previamente foi analisado também o tempo para realizar a carga de dados, notando-se que uma base de dados denormalizada leva mais tempo para o carregamento devido à redundância de dados de alguns atributos que antes eram tratados em entidades separadas,

aumentando o tamanho das entidades das quais eles agora fazem parte. Como mostra a Tabela 1, o MonetDB teve desempenho melhor que o PostgreSQL.

**Tabela 1. Tempo de carregamento em segundos para os cenários de *benchmark***

| SGBD | Base de Dados (Gb) | | | Ambiente |
|---|---|---|---|---|
| | 1 | 10 | 30 | |
| MonetDB | 44 | 409 | 1351 | *Snowflake* |
| PostgreSQL | 104 | 816 | 4979 | |
| MonetDB | 88 | 689 | 2216 | *Star* |
| PostgreSQL | 166 | 2365 | 7216 | |

Outra variável analisada foi o tamanho final de cada SGBD. A Tabela 2 mostra que mesmo para a mesma quantidade de dados o MonetDB tem o tamanho em bytes reduzido, e um fator que explica isso é a capacidade que um SGBD colunar tem de realizar compressão de dados de forma mais eficiente.

**Tabela 2. Tamanho do banco de dados em Mb para os cenários de *benchmark***

| SGBD | Base de Dados (Gb) | | | Ambiente |
|---|---|---|---|---|
| | 1 | 10 | 30 | |
| MonetDB | 771 | 8087 | 21352 | *Snowflake* |
| PostgreSQL | 1567 | 15510 | 44371 | |
| MonetDB | 952 | 9352 | 26271 | *Star* |
| PostgreSQL | 2565 | 25487 | 71586 | |

O processo do TPC-H foi desenvolvido utilizando o *driver JDBC*, cada qual disponível na página oficial dos SGBD, devido à simplificação que seu uso traz ao desenvolvimento de aplicações, e por dar suporte a diferentes SGBD. Os resultados dos testes de força e vazão para o ambiente normalizado podem ser observados na Tabela 3. No MonetDB o teste de força mostrou-se superior para todas as bases de dados, porém o teste de vazão apresenta comportamento diferente conforme a carga de dados aumenta, evidenciando um desempenho melhor do PostgreSQL em relação ao MonetDB.

**Tabela 3. Valores dos testes de força e vazão para os cenários normalizados**

| SGBD | Base de Dados (Gb) | | | Execução |
|---|---|---|---|---|
| | 1 | 10 | 30 | |
| MonetDB | 14043.395 | 11861.997 | 5457.325 | Power@Size |
| PostgreSQL | 1222.570 | 870.994 | 363.441 | |
| MonetDB | 1622.929 | 330.479 | 112.572 | Throughput@Size |
| PostgreSQL | 958.028 | 364.541 | 220.005 | |

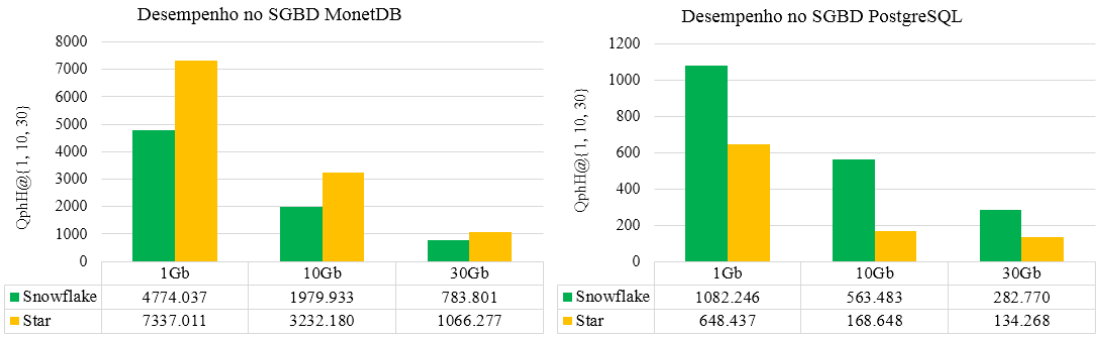
Ao contrário do cenário normalizado, no denormalizado o MonetDB obteve valores superiores ao PostgreSQL em ambos os testes. A Tabela 4 evidencia que o MonetDB em ambiente denormalizado consegue ter ganhos em relação ao normalizado para cenários com mais de um usuário ativo.

Os valores de execução dos testes de força e vazão foram aplicados no cálculo de desempenho final para o *benchmarking*. Ao avaliar de forma isolada, o MonetDB

**Tabela 4. Valores dos testes de força e vazão para os cenários denormalizados**

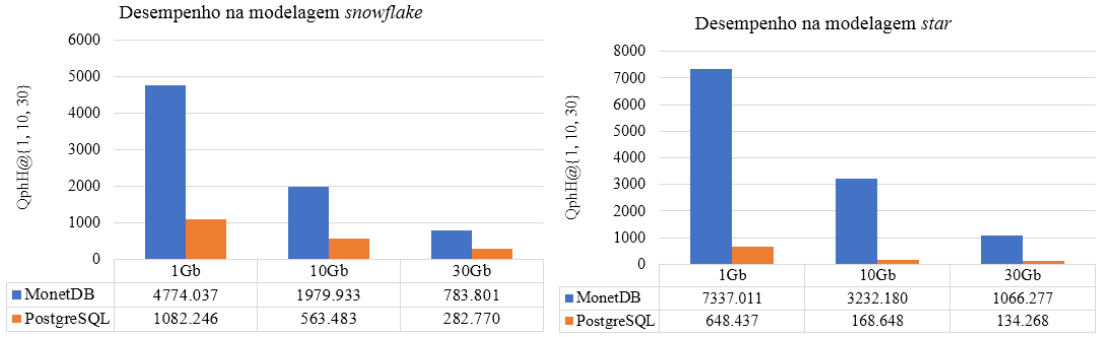
| SGBD | Base de Dados (Gb) 1 | 10 | 30 | Execução |
|---|---|---|---|---|
| MonetDB | 21811.705 | 17902.155 | 6873.122 | Power@Size |
| PostgreSQL | 659.758 | 190.307 | 149.514 | |
| MonetDB | 2468.02 | 583.560 | 165.419 | Throughput@Size |
| PostgreSQL | 637.310 | 149.455 | 120.578 | |

destacou-se na modelagem *star* por ser mais denormalizada, com ganhos de 54%, 63% e 36%. O PostgreSQL, por ser relacional, teve perdas de 40%, 70% e 53% no ambiente *star* em relação ao *snowflake*. As Figuras 1(a) e 1(b) ilustram os resultados em conjunto de seus valores para o MonetDB e o PostgreSQL, respectivamente.




(a) (b)

**Figura 1. Ilustração dos resultados entre os SGBD**




(a) (b)

**Figura 2. Ilustração dos resultados entre os ambientes**

Dados os ambientes e comparando os SGBD, o MonetDB obteve ganhos sob o PostgreSQL em todos os cenários. Este resultado era esperado se considerado o armazenamento e leitura aprimorados do MonetDB, apesar dos resultados apresentados pelo teste de força. Os ganhos para as bases de dados foram de 341%, 251% e 177% no ambiente normalizado e 1031%, 1817% e 694% no ambiente denormalizado, como ilustrado nas Figuras 2(a) e 2(b).

## 5. Conclusão e Trabalhos Futuros

Com todos os ganhos superiores a 100% e com seu melhor valor sendo em torno de dezenove vezes melhor que o do PostgreSQL, corrobora-se a ideia de que um SGBD NoSQL pode trazer vantagens para ambientes OLAP bases de dados pequenas e grandes; tanto à leitura quanto carregamento e armazenamento de dados, por adaptar-se melhor à modelagem denormalizada comumente utilizada em DW e por realizar compressão de dados. Com mais de um usuário ativo o PostgreSQL mostrou que atuando sobre uma base de dados normalizada um SGBD relacional pode obter resultados melhores que um NoSQL.

Como continuidade da pesquisa, propõe-se a inclusão de mais SGBD NoSQL de diferentes classes para analisar se a mudança na estruturação de SGBD que não utilizam SQL para manipulação do banco tem impacto direto em ambientes empresariais. Ainda, a aplicação dos resultados do estudo em empresas que utilizam uma modelagem relacional juntamente com SGBD relacionais, para de verificar se a mudança para um NoSQL sob modelagem denormalizada trará melhorias no desempenho de suas aplicações.

## Referências


Abadi, D., Madden, S., and Ferreira, M. (2006). Integrating compression and execution in column-oriented database systems. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 671–682. ACM.

Abadi, D. J. (2008). *Query execution in column-oriented database systems*. PhD thesis, Massachusetts Institute of Technology.

Brewer, E. A. (2000). Towards robust distributed systems. In *PODC*, volume 7.

Codd, E., Codd, S., and Salley, C. (1993). Providing olap (on-line analytical processing) to user-analysis: An it mandate. *White paper*.

Inmon, W. H. (2005). *Building the data warehouse*. John wiley & sons.

Leavitt, N. (2010). Will nosql databases live up to their promise? *Computer*, 43(2).

Matei, G. and Bank, R. C. (2010). Column-oriented databases, an alternative for analytical environment. *Database Systems Journal*, 1(2):3–16.

TPC-H (2017a). Tpc benchmark h standard specification. revision 2.17.2. `https://goo.gl/uaRRcv`. Acessado em: 12 de maio de 2017.

TPC-H (2017b). Tpc-h homepage. `http://www.tpc.org/tpch/`. Acessado em: 19 de maio de 2017.

Zelen, A. (2017). Oltp vs. olap — what's the difference? `https://academy.vertabelo.com/blog/oltp-vs-olap-whats-difference/`. Acessado em: 12 de abril de 2018.