Monografia

Pós-Graduação

MIT Big Data

Greenplum Database

Autor: Mário Otávio Jr, Walanem Figueiredo, Erick Henrique

Orientador: Eduardo Morelli

Rio de Janeiro, Junho de 2016

Sumário

1 Introdução

O Greenplum Database se destaca no ramo de Big Data como sendo uma das tecnologias mais importantes no nicho de paralelismo em processamento de grandes volumes de dados.

Este trabalho tem por objetivo salientar e mostrar, de forma detalhada, desde o conceito até o funcionamento, arquitetura, particularidades e aplicações práticas de uso desta ferramenta.

2 Overview

2.1 O que é o Greenplum?

O Greenplum Database é uma solução de Data Warehouse avançada, com diversas features bastante poderosas, com capacidade de prover de forma extremamente rápida e eficiente ferramentas para analytics, em escalas de petabytes de dados.

Além disso, ele possui o conceito de "Massive Parallel Processing" (MPP), de arquitetura "Shared-nothing", termos que serão detalhados posteriormente neste trabalho.

2.2 História

Inicialmente desenvolvido através de uma fusão de duas empresas, Didera e Metapa, o Greenplum sempre teve como característica principal ser uma versão mais performática e robusta do PostgreSQL.

O produto foi desenvolvido como fruto da fusão de duas empresas: Didera e Metapa que juntas, formaram a Greenplum Software.

Posteriormente, em julho de 2010, a EMC adquiriu os direitos do produto após adquirir a Greenplum Software.

O produto sempre foi uma ferramenta paga, de alto valor. Até que em outubro de 2015, ele se tornou uma ferramenta gratuita e open-source.

3 Detalhamento teórico

3.1 Arquitetura

O Pivotal Greenplum Database é um servidor de banco de dados com processamento paralelo (MPP), com uma arquitetura especialmente projetada para gerenciar data warehouses de grande escala, workloads de dados analíticos e de inteligência de negócios (business intelligence).

MPP (também conhecido como uma arquitetura não compartilhada) refere-se à sistemas com dois ou mais processadores que atendem as requisições, cada um com seus recursos próprios de memória, sistema operacional e discos.

Greenplum usa essa arquitetura de sistema de alto desempenho para distribuir a carga de dados de data warehouses , e pode usar todos os recursos de um sistema em paralelo para processar uma consulta.

Greenplum Database é baseado na tecnologia de código-fonte aberto PostgreSQL. Na prática, essencialmente várias instâncias de banco de dados PostgreSQL agem em conjunto como um sistema de gerenciamento de banco de dados (DBMS).

Baseia-se no PostgreSQL 8.2.15, e na maioria dos casos é muito semelhante ao que diz respeito ao PostgreSQL no suporte SQL. Características, opções de configuração e funcionalidades do usuário final são relativamente semelhantes ao PostgreSQL DBMS, por exemplo.

Greenplum Database também inclui recursos projetados para otimizar o PostgreSQL para workloads de Business Intelligence (BI). Por exemplo, Greenplum permite a carga paralela de dados de tabelas externas (que apontam, por exemplo, para arquivo texto (csv, txt, outros)), possui diversos recursos de gerenciamento, otimizações de consulta e melhorias no armazenamento dos dados, que nativamente não são encontrados na versão regular do PostgreSQL. Muitos recursos e otimizações desenvolvidos para o Greenplum abrem caminho para o Comunidade PostgreSQL incluir essas features em versões futuras. Por exemplo,

particionamento de tabelas é exemplo de funcionalidade que foi primeiramente desenvolvida para o Greenplum, e agora é nativa no PostgreSQL.

Greenplum Database armazena e processa grandes quantidades de dados através da distribuição de dados e de processamento de workloads entre vários servidores ou hosts. Greenplum Database corresponde a uma variedade de bancos de dados individuais com base no PostgreSQL 8.2 trabalhando em conjunto para apresentar uma única instância de banco de dados. O mestre é o ponto de entrada para o Greenplum Database processar as requisições. É nele que os clientes se conectam e enviam instruções SQL. O mestre coordena as requisições com as outras instâncias de banco de dados no sistema, os chamados segmentos, que armazenam e processam os dados.

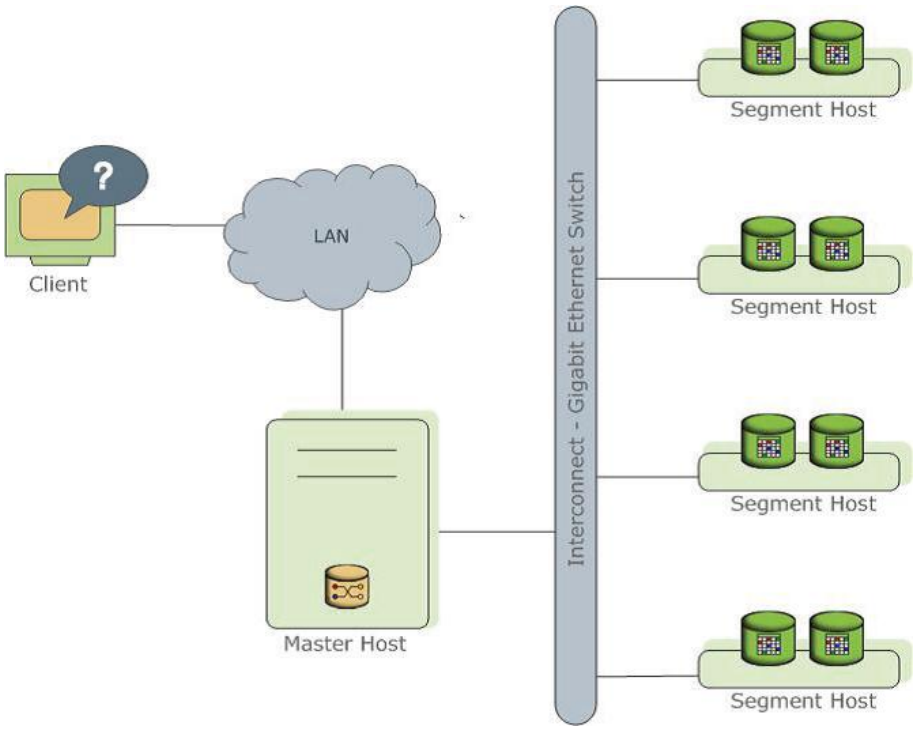

Figura 1: Arquitetura Greenplum Database alto nível.

3.2 Nó mestre

O nó mestre do Greenplum Database é o orquestrador do ambiente, recebendo as conexões dos usuários e instruções SQL, distribuindo o workload para os segmentos.

Os usuários do Greenplum Database se conectam ao banco de dados usando programa cliente (client program), como PLSQL ou interfaces de programação de aplicativos

(APIs), tais como JDBC ou ODBC.

O mestre contém o catálogo do conjunto de metadados do próprio sistema Greenplum Database. O mestre não contém todos os dados dos usuários; os dados residem apenas nos segmentos. O mestre autentica as conexões de cliente, processa as instruções SQL recebidas, distribui os workloads entre os segmentos, coordena os resultados retornados por cada segmento e apresenta os resultados finais para o programa cliente.

3.3 Segmentos

Os segmentos do Greenplum casos são bancos de dados PostgreSQL independentes que contém porções dos dados e executam a maioria do processamento de consultas.

Quando um usuário se conecta ao banco de dados através do nó mestre e realiza uma consulta, os processos são criados em segmento para processar a consulta.

As tabelas definidas pelo usuário e seus índices são distribuídos entre os segmentos disponíveis no ambiente; cada segmento contém uma porção distinta de dados. Os usuários interagem com segmentos apenas através do mestre.

Os segmentos são executados em servidores chamados segment hosts. Esses normalmente executam de dois a oito segmentos, dependendo da quantidade de núcleos de CPU, RAM, armazenamento, interfaces de rede e workloads.

É desejável que os segment hosts possuam as mesmas configurações pois essa é a chave para obter o melhor desempenho e distribuir dados e workloads uniformemente os segmentos, que consequentemente processam as requisições simultaneamente e concluem as requisições no mesmo tempo.

3.4 Interconexão

A interconexão é a camada de rede da arquitetura responsável pela comunicação entre processos, segmentos e a infraestrutura de rede em que esta comunicação se baseia. Essa camada utiliza a interface de rede 10-Gigabit Ethernet.

Por padrão, a interconexão usa User Datagram Protocol (UDP) para enviar mensagens através da rede.

O software Greenplum executa a verificação de pacotes fornecidos pela UDP (de maneira equivamente ao Transmission Control Protocol (TCP)) , porém com desempenho e escalabilidade superiores. Se a interconexão utilizasse o protocolo TCP, o Greenplum teria um limite de escalabilidade de 1000 instâncias de segmento. Com UDP como o protocolo padrão para a interconexão, este limite não é aplicável.

3.5 Gerenciamento e utilitários de monitoramento

O Greenplum Database fornece utilitários de linha de comando padrão para a realização de tarefas de monitoramento e administração.

Os utilitários de linha de comando estão localizados no diretório $GP_HOME/bin e são executados no nó mestre. O Greenplum fornece utilitários para as seguintes tarefas de administração:

- Instalação do Greenplum Database em um cluster;
- Inicialização de um sistema Greenplum Database;
- Start ou stopping do Greenplum Database;
- Adição ou remoção de um nó;
- Expansão do cluster e redistribuição das tabelas entre os novos segmentos
- Gerenciamento de recuperação de falhas em nós de segmentos
- Gerenciamento de redundância (failover) e recuperação de falhas em um nó mestre
- Backup e Restauração de um banco de dados (em paralelo)
- Carga de dados em paralelo
- Transferência de dados entre Greenplum Databases
- Relatórios de estado do sistema

Greenplum fornece uma ferramenta de monitoramento do sistema opcional e de gerenciamento que os administradores podem instalar e habilitar. Greenplum Command

Center usa coleção de dados de segment hosts para coletar e armazenar as métricas do sistema em um banco de dados dedicado. Os agentes de coleta de dados dos segment hosts enviam informações para o nó mestre em intervalos regulares (normalmente a cada 15 segundos).

Os usuários podem consultar o Greenplum Command Center para analisar métricas de consulta e de sistema. Essa ferramenta possui uma interface de usuário gráfica baseada na web e os administradores podem instalar separadamente. Atualmente como o Greenplum Database tornou-se um produto de código-fonte aberto, o recurso nativo para essa gestão é o *Ambari*.

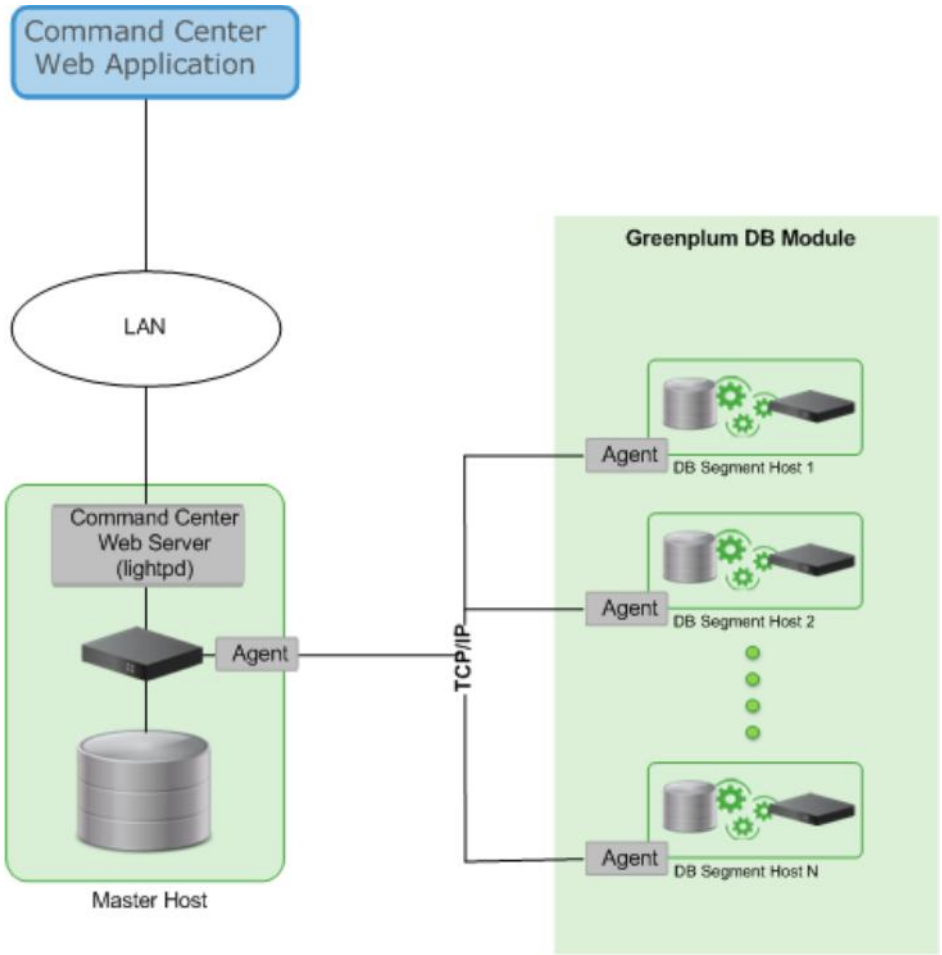

Figura 2: Ilustração do Command Center e interação dos agentes (master e demais segmentos)

3.6 Carga paralela de dados

Em grande escala, um data warehouse de vários terabytes, grandes quantidades de

dados devem ser carregados dentro de uma janela de manutenção relativamente pequena. Greenplum suporta carregamento rápido de dados em paralelo com o recurso de tabela externa.

Os administradores também podem carregar tabelas externas no modo isolado para filtrar separamente linhas com bugs, continuando a carregar linhas formatadas corretamente. Os administradores podem especificar um limite de erro para uma operação de carga para controlar quantas linhas formatadas incorretamente devem abortar a operação.

Ao usar tabelas externas em conjunto com o servidor de arquivos paralelo do Greenplum (gpfdist), os administradores podem atingir o máximo de paralelismo e largura de banda durante a carga.

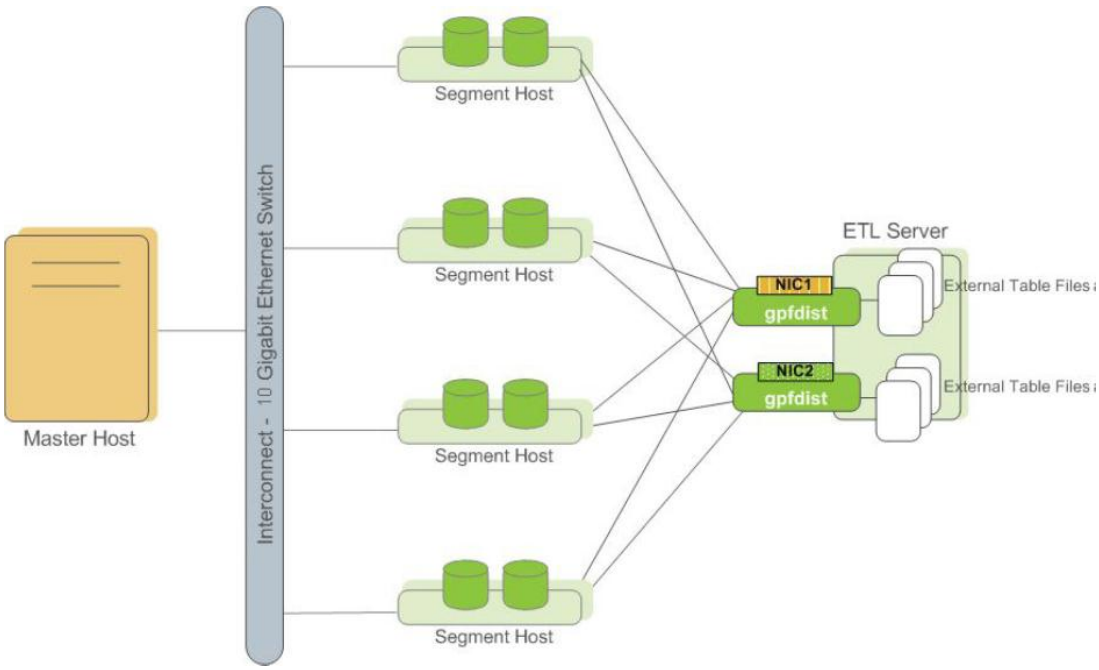

Figura 3: Utilizando Greenplum Parallel File Server (gpfdist) para carregar tabelas externas.

Outro utilitário, o gpload, executa uma tarefa de carga que pode ser especificada em um arquivo de controle em formato YAML (esse é um formato de serialização (codificação de dados) de dados legíveis por humanos inspirado em linguagens como XML, C, Python,

Perl, assim como o formato de correio eletrônico especificado pela RFC 2822. YAML foi proposto por Clark Evans em 2001 em conjunto com Ingy döt Net e Oren Ben-Kiki).

Para isso, é necessário descrever os locais de fontes de dados, formato, as transformações necessárias, hosts participantes, os destinos de banco de dados e outros elementos contidos no arquivo de controle e o utilitário executa a carga. Isso permite para descrever uma tarefa complexa e executá-lo de uma forma controlada e se necessário, cíclica.

3.7 Redundância e failover

É possível implementar o Greenplum sem um único ponto de falha através do espelhamento de componentes.

É de extrema importância uma estratégia de replicação eficiente, para prever casos em que queries que tentem executar em segmentos com falha não possam se tornar um problema no fluxo de uma aplicação. Para isso, este tipo de estratégia deve levar em conta que hajam hosts suficientes para replicação de componentes, os quais são denominados de "mirrors".

3.8 Failover de segmento e recuperação

Quando o espelhamento está habilitado, o sistema irá falhar automaticamente para a cópia espelho se uma cópia principal ficar indisponível. O sistema Greenplum pode permanecer operacional se um segment host ou segmento ou estiver indisponível, desde que todos os dados estejam disponíveis nos demais segmentos ativos restantes.

Se o mestre não puder se conectar a um segmento, o sistema atualiza o status desse segmento no catálogo de metadados e automaticamente é ativado o segmento espelho em seu lugar. Um segmento que apresente falha permanecerá fora de operação até que um administrador tome medidas para torná-lo online novamente.

Um administrador pode recuperar um segmento com falha enquanto o sistema está em funcionamento. O processo de recuperação copia as alterações que foram perdidas enquanto o segmento estava fora de operação.

Se o espelhamento não estiver ativado, o sistema será desligado automaticamente se um segmento apresentar falha. Será necessário recuperar todos os segmentos com falha antes de retomar o ambiente.

3.9 Espelhamento de segmento

Ao implantar o Greenplum, o espelhamento de segmento poderá ser configurado opcionalmente.

Esse recurso permite que consultas de banco de dados sejam direcionadas para um segmento de backup se o segmento primário estiver indisponível (failover). Para configurar o espelhamento, é necessária uma quantidade suficiente de segment hosts para que o espelhamento do segmento secundário esteja sempre em um host diferente do seu segmento primário.

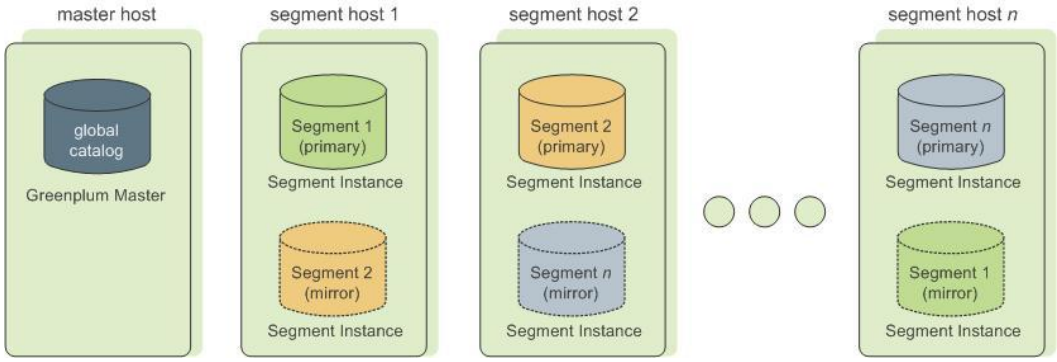

Figura 4: É possível verificar como os dados da tabela são distribuídos entre os segmentos quando o espelhamento é configurado.

3.10 Espelhamento do Mestre

É possível implantar opcionalmente um backup ou espelhamento da instância mestre em um segmento separado do nó mestre. O segmento principal de backup serve como warm standby (método de redundância em que o sistema secundário funciona no lugar do sistema primário) no caso d o host mestre primário tornar-se inativo. O mestre de espera é mantido up to date por um processo de replicação do log de transações, que é executado no host mestre de

espera e sincroniza os dados entre os hosts mestre primário e de espera.

Se o mestre primário falhar, o processo de replicação de registo para, e o mestre de espera pode ser ativado em seu lugar. Após a ativação do mestre de espera, os logs replicados são usados para reconstruir o estado do host principal no momento da última transação confirmada (commit) com êxito. O mestre de espera ativado torna-se efetivamente o mestre do sistema, aceitando conexões de clientes na porta mestre (que deve ser definida para o mesmo número de porta no host mestre principal).

Desde que o mestre não contenha nenhum dado de usuários, apenas as tabelas de catálogo do sistema precisam ser sincronizadas entre as cópias primárias e de backup. Quando estas tabelas são atualizadas, as alterações são automaticamente copiadas para o mestre de espera para garantir a sincronização com o mestre primário.

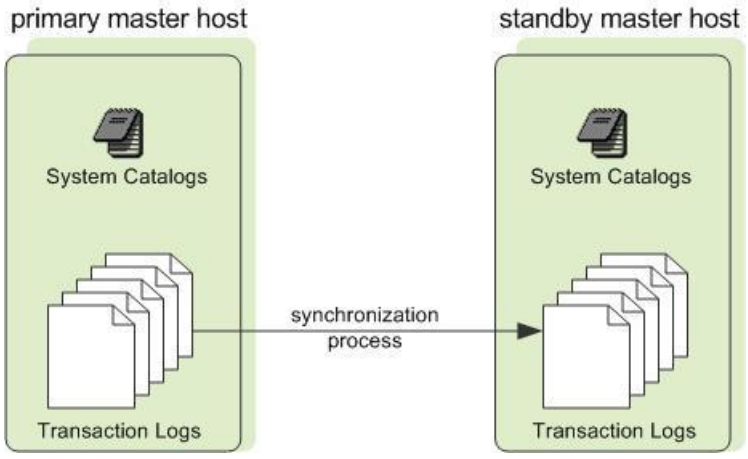

Figura 4: Master mirrroring.

3.11 Redundância de Interconexão

A interconexão refere-se a comunicação entre processos entre os segmentos e a infraestrutura de rede em que essa comunicação se baseia. É possível obter uma interconexão de alta disponibilidade usando duas interfaces 10-Gigabit Ethernet nas conexões entre os servidores mestre e demais segmentos.

3.12 Start e Stop

Em um banco de dados Greenplum DBMS, as instâncias de servidor (o mestre e todos os segmentos) são iniciados ou parados em todos os hosts do sistema de tal forma que eles possam trabalhar juntos como um SGBD unificado.

Um sistema Greenplum é distribuído através de muitas máquinas, o processo para iniciar e parar o sistema é diferente do processo para iniciar e parar um PostgreSQL regular DBMS.

Os comandos gpstart e gpstop são os utilitários para iniciar e parar o banco, respectivamente. Estes utilitários estão localizados no diretório $GP_HOME/bin do host mestre.

Caso seja necessário reiniciar o sistema, deve ser utilizado o utilitário gpstop, através do comando gpstop –r.

O sistema possui dois arquivos básicos de configuração (postgresql.conf e pg_hba.conf) que podem ser alterados a qualquer momento em caso de necessidade. O Greenplum permite que as configurações sejam atualizadas de maneira on-line, sem a necessidade de reiniciar o banco. Para esse caso deve ser utilizado o comando gpstop –u.

3.13 Modo de manutenção

Pode ser necessário realizar alguma manutenção no sistema, nesse caso é possível que o ambiente seja configurado para modo de manutenção. Dessa maneira, apenas o nó mestre fica disponível e todas as tarefas e demais requisições serão processadas unicamente por ele.

Segue roteiro para essa implementação:

1. Execute gpstart usando a opção -m:

$ Gpstart -m

2. Execute as instruções necessárias (ex: manutenção no catálogo):

$ PGOPTIONS =- 'c gp_session_role = utility’ template1 psql

3. Após as demais instruções de manutenção, o serviço deve ser reiniciado em modo de produção:

$ Gpstop MR

3.14 Aplicações client

O Greenplum vem instalado com uma variedade de aplicações client que estão localizadas no caminho $GP_HOME/bin do nó mestre. Seguem abaixo as instruções mais utilizadas:

Nome	Função
CreateDB	Cria um novo banco de dados
Createlang	Define uma nova linguagem procedural
createuser	Define uma nova role no banco de dados
DROPDB	Apaga um banco de dados
Droplang	Remove uma linguagem procedural
Dropuser	Remove uma função
Psql	Terminal interativo do PostgreSQL
Reindexdb	Reindexa um banco de dados
Vacuumdb	Realiza garbage collection e análise do banco de dados

3.15 Conexão via psql

Pode ser necessário, em certos casos, acessar o ambiente via psql. Os passos a seguir mostram como isto pode ser realizado.

$ Psql -d gpdatabase -h MASTER_HOST -p 5432 -U gpadmin

$ Gpdatabase psql

$ psql

Se o banco de dados definido pelo usuário ainda não tiver sido criado, é possível realizar o acesso ao sistema através do banco padrão template1. Exemplo:

$ Template1 psql

Depois de se conectar a um banco de dados, o psql disponibiliza um prompt com o nome do banco de dados conectado, seguido pela cadeia de caracteres => (ou = # se você é o root do banco de dados). Por exemplo:

gpdatabase → No prompt, você pode digitar comandos SQL. Um comando SQL deve terminar com um ‘;’ (ponto e vírgula). Exemplo:

SELECT * FROM minha_tabela;.

3.16 Carga de dados

Os dados podem ser carregados, transformados e formatados no Greenplum usando utilitários e ferramentas built-in. Existem opções que carregam dados em paralelo ou forma sequencial. Seguem algumas diferentes maneiras de carregar dados no Greenplum:

• INSERT: é um comando SQL padrão que é usado para a carga de dados em tabelas de banco de dados linha por linha. Esta opção deve não ser utilizada para a carga de muitas colunas. Nesta opção, os dados são encaminhados através o nó mestre e pode vir a ser um gargalo no caso de grandes volumes.

Este comando é comumente usado em conexão baseada em ODBC / JDBC.

Sintaxe:

INSERT INTO <<nome_tabela>> (<<nome de colunas separados por vírgula>>) VALUES (<<valores correspondentes>>);

Exemplo:

INSERT INTO funcionario (id, nome, sobrenome) VALUES (001, 'Mario', 'Junior');

• COPY: o comando é uma das formas iniciais de carga de dados. Não é executado

em paralelo, mas normalmente é usado para carga de grandes volumes de dados e é possível executar vários comandos COPY simultaneamente. Ele facilita a cópia de dados de STDIN (entrada padrão) ou STDOUT (saída padrão) usando a conexão entre o nó mestre e o cliente.

Exemplo:

```
COPY funcionario FROM '/usr/home/funcionario_historico.dat' WITH
DELIMITER '|';
```

• EXTERNAL TABLE: as tabelas externas são exclusivas no Greenplum e são tipicamente usadas para cargas massivas com alta performance (parallel e bulk load). O acesso às tabelas externas é baseado em arquivo, através dos protocolos file:// ou gpfdist://. Fontes dinâmicas podem ser acessadas através do protocolo http: //.

• gpload: é um utilitário para carga de dados baseado em arquivo de controle YAML.

3.17 Suporte a Arquivos JSON

Inicialmente o Greenplum não possui suporte para utilização de JavaScript Object Notation, JSON a única maneira que posria ser utilizada era a utilização de técnicas de conversão para arquivos delimitados. Para esta conversão era utilizadas técnicas em Phyton.

Exemplo:

```
import sys, json
for line in sys.stdin:
try:
j = json.loads( line )
except ValueError:
continue

vals = map( lambda x: str( j[x] ), ['id', 'type'] )
print '|'.join( vals ).encode('utf-8')
```

O arquivo delimitado gerado pelo processo de converção poderia ser carregado utilizando a técnica de criação de EXTERNAL TABLES.

Exemplo:

```
CREATE EXTERNAL TABLE data_ext (
id int,
type text
) LOCATION (
'gpfdist://localhost:8080/data/json_data.csv'
) FORMAT 'CSV' (
DELIMITER AS '|' );
```

Após a criação do objeto externo poderiamo realizar uma extrução SQL para efetuarmos a carga dos dados para um obejto interno.

Exemplo:

```
INSERT INTO data SELECT * FROM data_ext;
```

Com o aumento da necessidade de suportar o formato JSON, e com a transfomação do Greenplum em Software Livre, Código Aberto, que ajudou no crescimento da comunidade, facilitou a criação de algumas bibliotecas que suportam a utilização deste formato sem a necessidade de convertê-los antes da "importação"dos dados. Hoje a bibilioteca mais utilizada é a Greenplum JSON Formatter – CDF.

Através dela podemos criar objetos externos baseados em arquivos JSON somente leitura.

Exemplo:

```
CREATE EXTERNAL TABLE data_ext (
```

```
id int,
type text,
"address.city" text,
"address.state" text
) LOCATION (
'gpfdist://localhost:8081/data/sample.json.'
) FORMAT 'custom' (
formatter=json_formatter_read
);
```

Ou podemos criá-la com a opção de escrita:

Exemplo:

```
CREATE WRITABLE EXTERNAL TABLE writable_ext (
id int,
type text,
"address.city" text,
"address.state" text
) LOCATION (
'gpfdist://localhost:8081/out/phone.json'
) FORMAT 'custom' (
formatter=json_formatter_write
);
```

Desta forma, podemos utilizar instruções SQL para realizarmos consultas ou atualizações no arquivo JSON, neste caso é interessanto, pois caso executemos uma instrução de insert o arquivo JSON é atualizado.

Exemplo:

```
select * from writable_ext;
insert into writable_ext select * from data;
```

Outro ponto que podemos abordar com relação a EXTERNAL TABLE é que podemos carregar os dados de saída de uma instrução PYTHON, acrescentando a opção WEB na instrução de criação de tabela externa.

Exemplo:

```
CREATE EXTERNAL WEB TABLE json_data_web_ext (
   id int,
   type text
) EXECUTE 'json.py' ON MASTER
FORMAT 'CSV' (
   DELIMITER AS '|' );
```

4 Conclusão

Como pôde ser analisado neste trabalho, o Greenplum se trata de uma ferramenta extremamente poderosa, que através do paradigma MPP, permite processar quantidades massivas de dados, da ordem de petabytes, de forma simples e eficiente.

Além de possuir uma série de utilitários bastante eficazes e performáticos, o Greenplum ainda conta com integrações para outras soluções externas de Big Data, potencializando ainda mais seu portifólio.

Sem dúvida alguma, se trata de uma das soluções mais eficientes e robustas disponíveis no mercado para processamento de dados. Além disso, por ser open-source, ainda pode contar com constantes melhoras, através de feedbacks da comunidade.

5 Bibliografia

GOLLAPUDI, Sunila. **GETTING STARTED WITH GREENPLUM FOR BIG DATA.** Estados Unidos da America: Lightning Source, 2013. 172 p. (1782177043)

OJHA, Sachchida N. **GREENPLUM DATABASE DBA HANDBOOK.** Estados Unidos da América: N/A

EMC. **Greenplum Command Center 2.2.0 Release Notes.** 2016. Disponível em: <http://gpcc.docs.pivotal.io/220/gpcc/relnotes/GPCC-220-release-notes.html>. Acesso em: 01 maio 2016

PIVOTAL, Emc. **Greenplum Database Reference Guide.** Disponível em: <http://gpdb.docs.pivotal.io/4382/pdf/GPDB43RefGuide.pdf>. Acesso em: 01 maio 2016

6 Conceito Final e Feedback

6.1 Conceito Final

Nome do Bloco	Conceito	Validado no TCC
Greenplum Database	0 - 100	( ) Sim ( ) Não

Conceito Final do TCC		

6.2 Feedback do TCC

(Espaço destinado ao feedback do orientador, a respeito do TCC)