O que é Teradata?

Teradata é um RDBMS (sistema de gerenciamento de banco de dados relacional) que inclui os seguintes recursos:

Ele é construído em uma arquitetura completamente paralela, o que significa que uma única tarefa será dividida em partes menores e computará simultaneamente, portanto, uma execução mais rápida.

O sistema Teradata é uma arquitetura sem compartilhamento, na qual cada nó é independente e autossuficiente. Além disso, cada processador lógico (AMP) é responsável apenas por sua própria parte do banco de dados.

Compatível com ANSI SQL padrão do setor para se comunicar com Teradata.

O banco de dados Teradata pode ser acessado por vários usuários simultâneos de diferentes aplicativos clientes por meio de conexão TCP/IP popular ou conexão de canal de mainframe IBM.

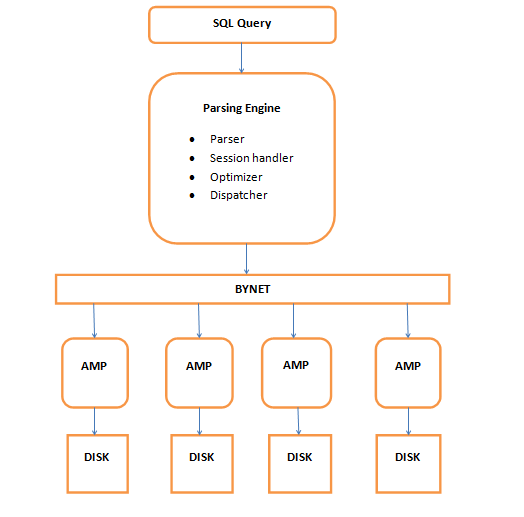
Por que usar Teradata?

Existem inúmeras razões pelas quais os clientes escolhem o Teradata em detrimento de outros bancos de dados.

* A escalabilidade linear ajuda a suportar mais complexidade de usuários/dados/consultas/consultas sem perder o desempenho. Quando a configuração do sistema aumenta, o desempenho aumenta linearmente.
* O sistema é construído em arquitetura aberta, portanto, sempre que qualquer chip e dispositivo mais rápido for disponibilizado, ele poderá ser incorporado à arquitetura já construída.
* Distribuição automática de dados em vários processadores (AMP) uniformemente. Os componentes dividem a tarefa em partes aproximadamente iguais para que todas as partes do sistema sejam mantidas ocupadas para realizar a tarefa mais rapidamente.
* Suporta mais de 50 petabytes de dados.
* Fornece um otimizador com reconhecimento paralelo que torna o ajuste de consulta desnecessário e o executa com eficiência.
* Visualização de operação única para um grande sistema de vários nós Teradata via SWS (Service Workstation). Isso é gerenciado principalmente pelo Teradata GSC.
* Ponto único de controle para o DBA gerenciar o banco de dados usando o ponto de vista Teradata.
* Compatível com um grande número de ferramentas de BI para buscar dados.

Arquitetura Teradata

Os principais componentes da Teradata Architecture são PE (Parsing Engine), BYNET, AMP (Access Module Processor), Virtual Disk. A seguir está a visão lógica da arquitetura:



Mecanismo de análise

Quando um usuário dispara uma consulta SQL, ele primeiro se conecta ao PE (Parsing Engine). Os processos como planejamento e distribuição dos dados para o AMPS são feitos aqui. Ele descobre o melhor plano ideal para a execução da consulta. Seguem os processos realizados pelo PE:

* Analisador: O analisador verifica a sintaxe e, se verdadeiro, encaminha a consulta para o manipulador de sessão.
* Session Handler: faz todas as verificações de segurança, como verificar as credenciais de login e se o usuário tem permissão para executar a consulta ou não.
* Otimizador: Encontra o melhor plano possível e otimizado para executar a consulta.
* Dispatcher: O Dispatcher encaminha a consulta para os AMPs.

BYNET

O BYNET atua como um canal entre PE e AMPs. Ele atua como um comunicador entre os dois. Existem dois BYNETs no Teradata 'BYNET 0' e 'BYNET 1'. Mas nos referimos a eles como sistema BYNET único. O motivo de ter 2 BYNETs é:

Se um BYNET falhar, o segundo pode tomar seu lugar.

Quando os dados são grandes, ambos os BYNETs podem se tornar funcionais, o que melhora a comunicação entre PE e AMPs, agilizando o processo.

AMP

O Access Module Processor é um processador virtual conectado ao PE via BYNET. Cada AMP possui seu próprio disco e pode ler e gravar em seu PRÓPRIO disco. Isso é chamado de 'ARQUITETURA DE NADA COMPARTILHADA'. Quando a consulta é disparada, o Teradata distribui as linhas da tabela em todos os AMPs e, quando solicita qualquer dado, todos os AMPs trabalham simultaneamente para devolver os dados. Isso se chama PARALELISMO. O AMP executa qualquer solicitação SQL em três etapas

* Tranque a mesa(table).
* Execute a operação solicitada.
* Termine a transação.
* Disco

A Teradata oferece um conjunto de discos virtuais para cada AMP. A área de armazenamento de cada AMP é chamada de Virtual Disk ou Vdisk. Os passos para a execução da consulta estão abaixo:

* O usuário dispara a consulta que é enviada ao PE.
* O PE faz as verificações de segurança e sintaxe e descobre o melhor plano ideal para executar a consulta.
* As linhas da tabela são distribuídas no AMP e os dados são recuperados do disco.
* O AMP envia de volta os dados por meio de BYNET para PE.
* PE retorna os dados para o usuário.

Guia de instalação do Teradata

Quer instalar o Teradata no seu PC e começar a praticar hoje? Aqui está o guia de instalação do Teradata com capturas de tela. Teradata lança imagem pré-instalada para Teradata Express para Vmware Workstation/Player. Ao instalá-lo, você pode acessar o teradata de seu sistema local usando o Teradata SQL Assistant, o Teradata Administrator ou qualquer outro utilitário como BTEQ, FASTLOAD, MLOAD etc. Você pode seguir algumas etapas simples, conforme abaixo:

Pré-requisito:

PC com capacidade de virtualização de 64 bits. [você pode verificar na configuração do BIOS]

Mínimo de 3 GB de RAM. [4GB é recomendado]

Estação de trabalho/Player VMware. [Estação de trabalho VMware é recomendada]

Teradata 15.00.01 ou qualquer outra versão. [Teradata Express para Vmware Workstation/Player]

td-ttu-15.00\_for\_Windows. [Ferramentas e utilitários Teradata]

Passo 1 :

Você pode baixar o Teradata Express e a estação de trabalho Vmware do site Teradata

(<https://downloads.teradata.com/download/database/teradata-express/vmware>)

e do site Vmware

(<https://customerconnect.vmware.com/downloads/details?downloadGroup=WKST-1113-WIN&productId=462&rPId=9527&irclickid=0W8wtfQJvxyNUVPR-0zaLzxkUkAWStzPOXAwW40&utm_source=affiliate&utm_medium=TEXT_LINK_&utm_campaign=VMware%2520Workstation%252016%2520Pro&utm_term=engine%253Aimpact%257Cpublisherid%253A95368%257Ccampaignid%253A11461&irpid=95368&irgwc=1>),

respectivamente. Uma vez baixado, você pode descompactar o Teradata Express em alguma pasta. Suponha que você o tenha descompactado em uma pasta de nome “virtual”. Você vai acabar com algo assim- ” F:\virtual\TDExpressXX\_Sles11\_40GB “. Instale a estação de trabalho VMware.

Passo 2 :

Abra a página de boas-vindas do VMware Workstation, escolha "Open a Virtual Machine" e navegue até a pasta Teradata express, procurando o arquivo "TDExpress15.00.01\_Sles11".

Passo 3 :

Depois de encontrar a imagem, abra-a clicando duas vezes. O VMware agora perguntará se você deseja copiar ou mover esta imagem. Certifique-se de escolher 'I MOVED IT'!

Passo 4 :

Agora clique em “Ligar esta máquina virtual”( “Power on this virtual machine”).

Passo 5 :

Faça login na VM SLES11 com o nome de usuário “root” e a senha “root”. Aguarde algumas vezes, ele inicializará o Teradata.

Passo 6 :

Quando estiver pronto, abra o terminal Gnome e você poderá verificar o status do servidor de banco de dados Teradata usando o comando abaixo

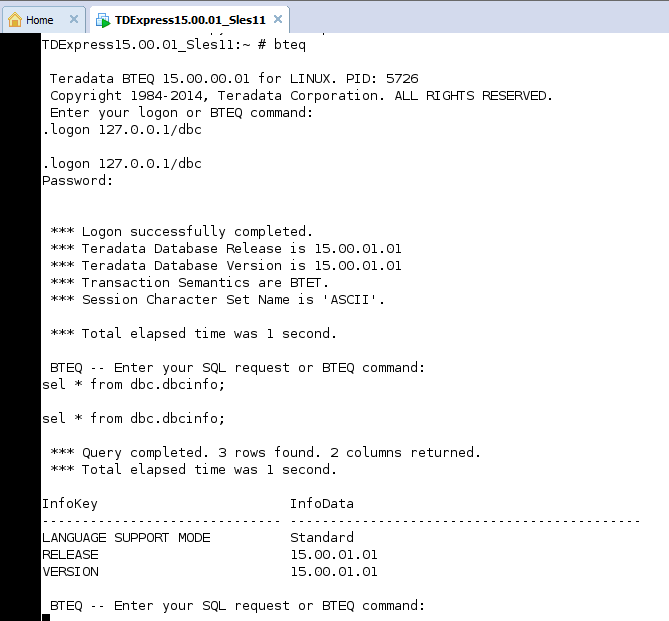
TDExpress15.00.01\_Sles11:~# pdestate -a

PDE state is RUN/STARTED.

DBS state is 4: Logons are enabled – Users are logged on

TDExpress15.00.01\_Sles11:~#

Agora você também pode testar a conectividade usando o bteq com o nome de usuário “dbc” e a senha “dbc”.



Você concluiu com êxito a instalação do servidor Teradata totalmente funcional. Se você deseja acessar este servidor de banco de dados Teradata do seu PC local, algumas etapas extras podem ser seguidas:

Passo 7 :

Baixe o Microsoft .NET 4

(<https://www.microsoft.com/en-in/download/details.aspx?id=17851>)

ou superior e o Teradata 15 TTU

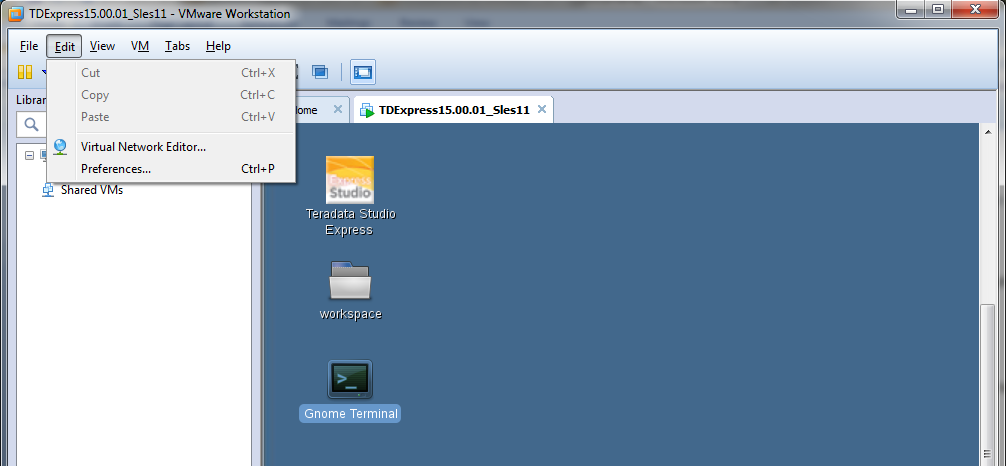
(<https://downloads.teradata.com/download/database/teradata-tools-and-utilities-13-10>)

para Windows. Você precisa instalar o .NET framework antes de instalar o Teradata Tools and Utilities (TTU).

Passo 8 :

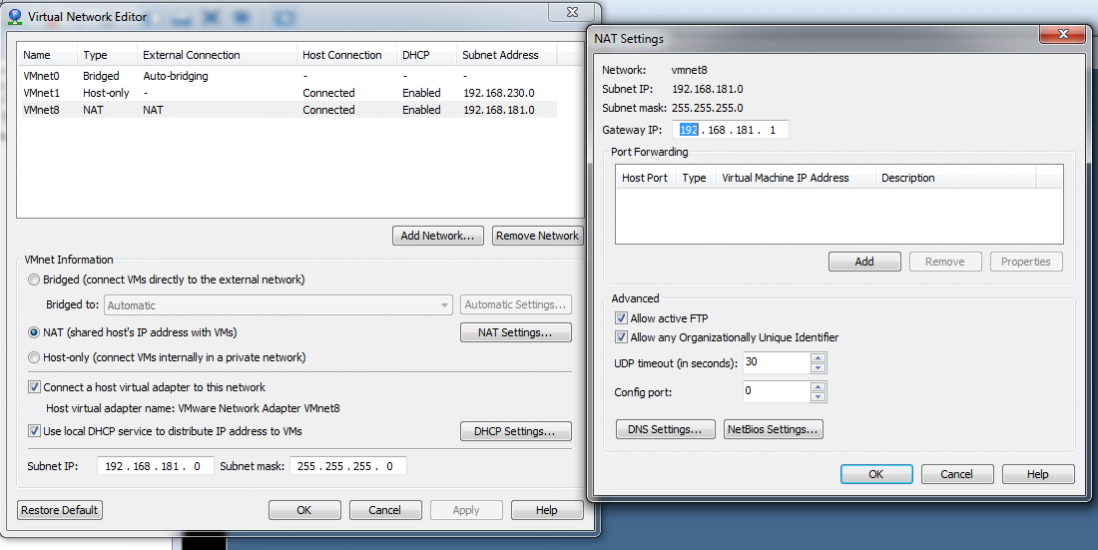
Após a conclusão da instalação do TTU, configure o IP para VM e SLES11 para acessá-lo de seu sistema local usando o Teradata SQL Assistant ou qualquer outro utilitário.

* Abra o Virtual Network Editor no menu EDIT da VM.



* Selecione a configuração VMnet8 NAT e configure o IP do gateway.

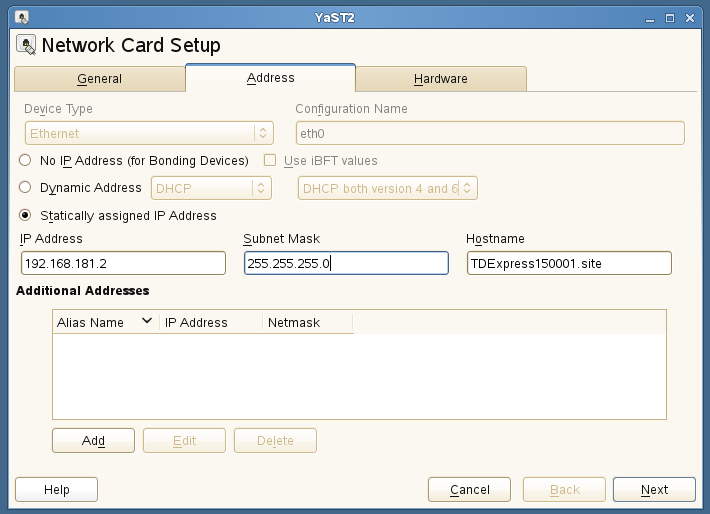
Suponha que seu endereço de sub-rede NAT seja 192.168.181.0, você pode fornecer o IP do gateway como 192.168.181.1. Clique OK.



* Agora abra Configurações de rede(Network Settings) no SLES11, ou seja, no sistema operacional dentro do VMware.

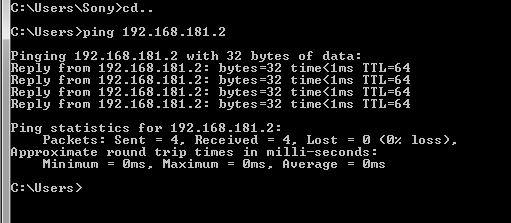
Computer -> Control Center -> Network Settings -> Select Ethernet Network Card->Edit.

Suponha que o IP do gateway seja 192.168.181.1 na configuração NAT, você deve inserir 192.168.181.2 aqui conforme abaixo:

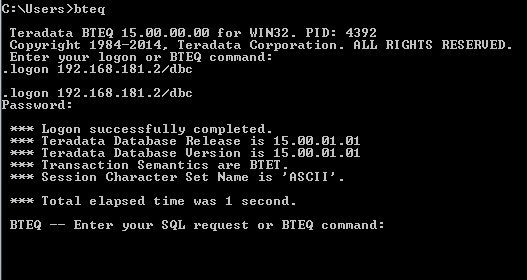


Click Next.

* Agora abra o prompt de comando no sistema local e faça ping neste endereço IP.



Tudo feito. Agora você pode usar este endereço IP (192.168.181.2) do seu driver ODBC local, Teradata SQL Assistant ou BTEQ para acessar o servidor de banco de dados Teradata.



No próximo tutorial, mostraremos como usar o Teradata SQL Assistant para conectar o Teradata Database usando o driver ODBC.

Como conectar o Teradata SQL Assistant ao Teradata

Neste tutorial, orientaremos você sobre como conectar o Teradata SQL Assistant ao banco de dados Teradata usando o Teradata ODBC Driver e outros métodos. Se você deseja instalar o Teradata Database no Windows, pode conferir nosso tutorial no Teradata Installation Guide.

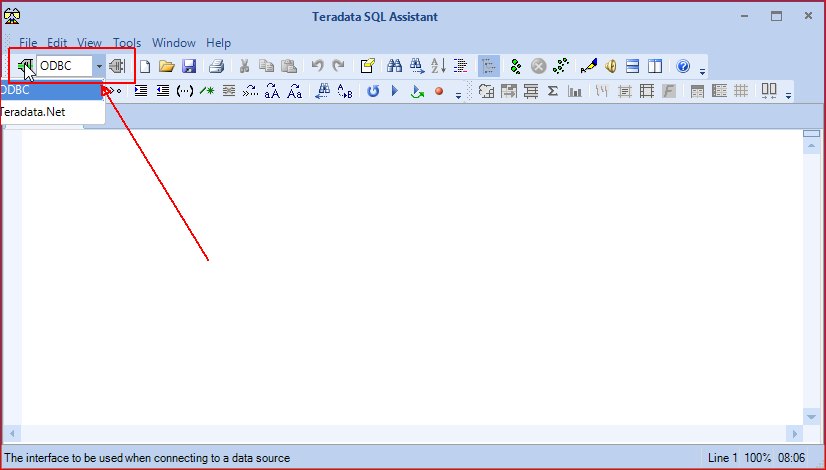
Abaixo estão as etapas para conectar o Teradata SQL Assistant ao Teradata RDBMS:

Etapa 1: você pode baixar o pacote Teradata Tools and Utilities no site oficial da Teradata, que inclui todas as ferramentas para Teradata, incluindo o Teradata SQL Assistant e o driver ODBC.

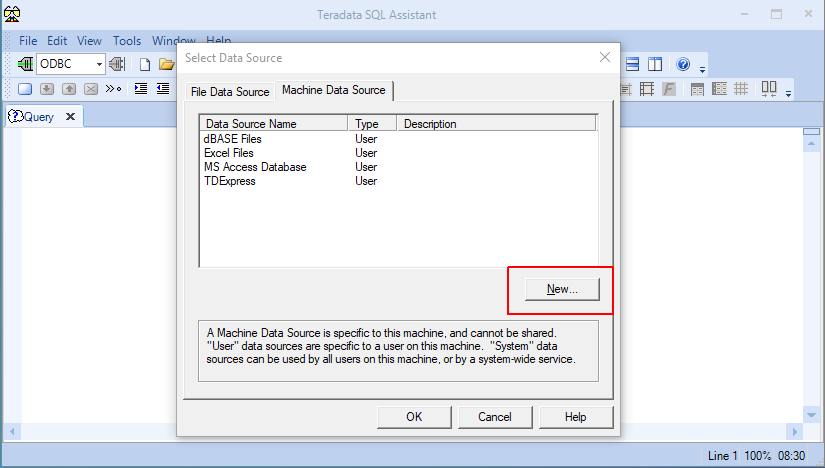
Como alternativa, você também pode baixar o Teradata SQL Assistant e o driver Teradata ODBC individualmente na página de download oficial do Teradata.

Etapa 2: Depois de baixar o pacote, basta extraí-lo e instalá-lo como qualquer outro software no Windows e almoçar o Teradata SQL Assistant Software.

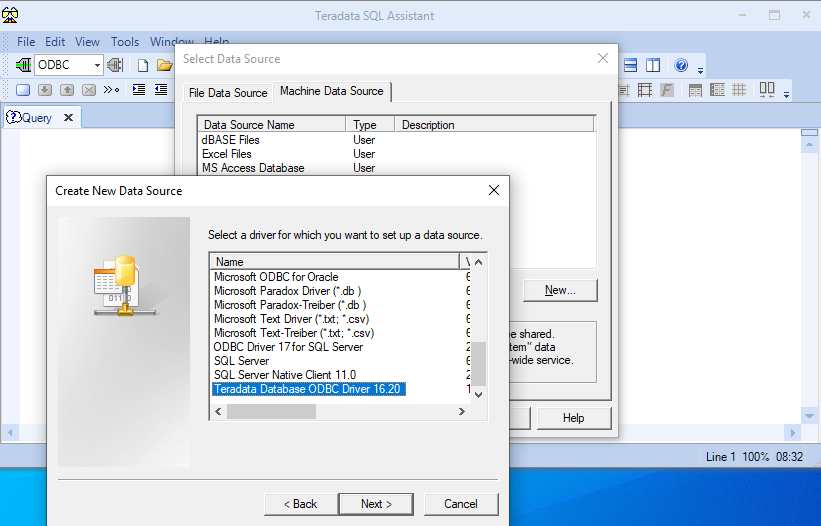
Etapa 3: Depois de abrir o Teradata SQL Assistant, você encontrará a opção de conexão no canto esquerdo logo abaixo da opção de arquivo. Selecione a opção ODBC e clique em conectar.



Etapa 4: Na próxima janela de Selecionar fonte de dados(Select Data Source), selecione Fonte de dados da máquina(Machine Data Source) e clique em Novo(New).



Etapa 5: Clique em Avançar(Next) e selecione o driver ODBC do banco de dados Teradata no último extremo. Depois clique e finalize(Finish).



Etapa 6: Na próxima janela, forneça as informações abaixo e clique em OK.

**Name** – Teradata ODBC ou qualquer nome que você desejar.

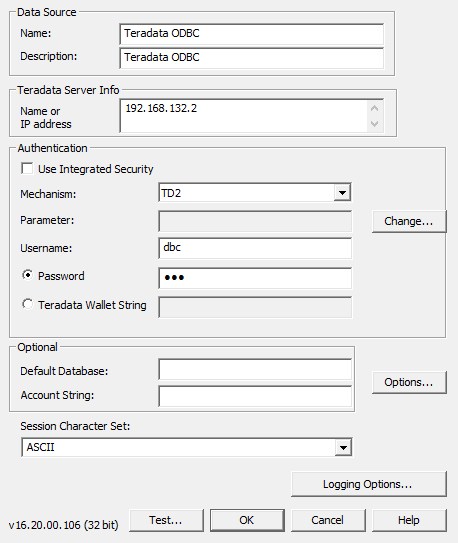
**Description** – Teradata ODBC ou qualquer descrição que você desejar.

**Name or IP address** – Nome ou endereço IP do servidor Teradata. Nesse caso, estou usando meu Teradata instalado localmente.

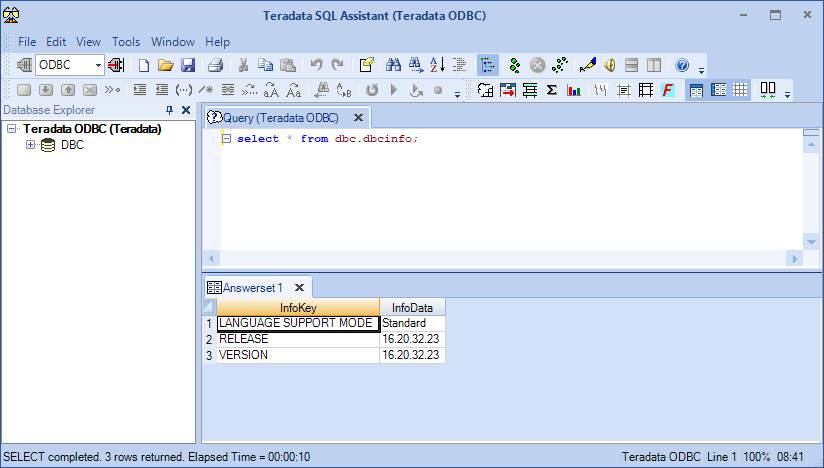
**Mechanism** - TD2

**Username** – dbc (o usuário padrão veio após a nova instalação do Teradata. Podemos criar outros usuários posteriormente.)

**Password** – dbc (senha padrão)



Etapa 7: Na próxima etapa, depois de clicar em ok duas vezes, o driver ODBC conectará o Teradata SQL Assistant ao sistema Teradata.



Etapa 8: Uma vez conectado ao Sistema Teradata, você pode verificar a versão do banco de dados Teradata usando a seguinte Consulta.

select \* from dbc.dbcinfo;

Configuração Adicional do Teradata SQL Assistant:

Abaixo estão algumas das configurações adicionais que você pode fazer no Teradata SQL Assistant.

1. Janela de respostas separada para a consulta.

Para configurar uma janela de folha de respostas separada para cada consulta, você pode seguir as etapas abaixo.

a) Clique em Tools.

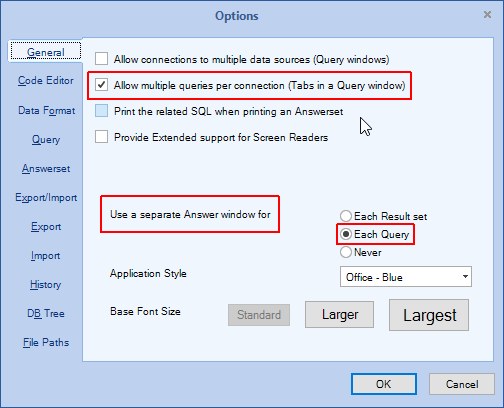
b) Clique nas options.

c) Clique na guia ‘General’.

d) Selecione ‘Allow multiple queries per connection’.

e) E para “Use a separate Window for” selecione “Each Query”.

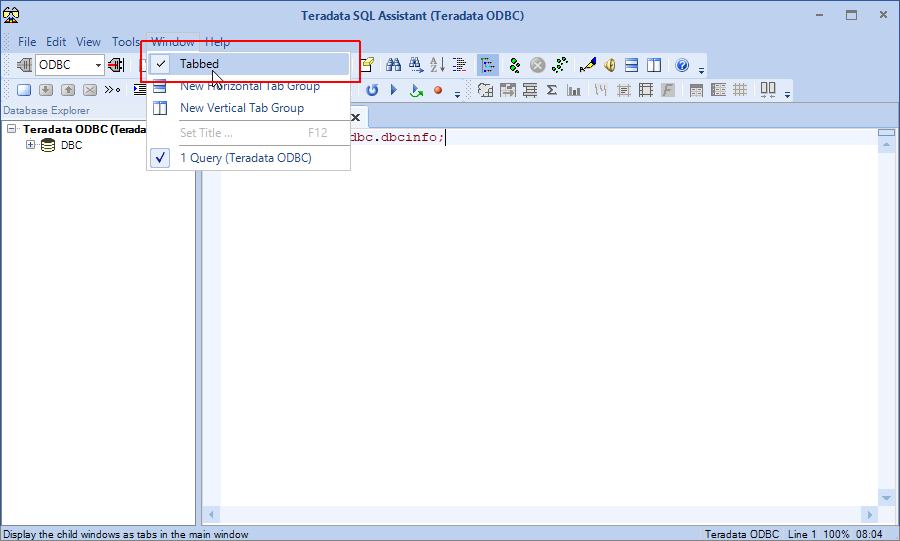
f) Agora clique na guia Consulta e desmarque a opção “Close Answers windows before submitting a new query”.



2. Folha de respostas com guias do Teradata SQL Assistant.

a) Vá para Window.

b) Agora selecione Tabbed.



Espero que você tenha se conectado com sucesso ao banco de dados Teradata usando o SQL Assistant usando o guia a seguir. Ótimo! agora verifique os nºs de AMPs, PEs em seu sistema Teradata consultando nosso artigo Conheça a configuração do seu sistema Teradata.

Conheça a configuração do seu sistema Teradata  
Pode vir à sua mente quantos **Nodes, Amps, Pes** existem em seu sistema Teradata de desenvolvimento e produção no qual você está trabalhando diariamente. Conheça a configuração do seu sistema Teradata conforme abaixo. Para verificar a configuração do sistema Teradata, você pode usar a consulta abaixo:

SELECT DISTINCT

NodeID

,NodeType

,VProcType1||’: ‘||TRIM(VProc1) AS AMPs

,VProcType2||’: ‘||TRIM(VProc2) AS PEs

,VProcType3||’: ‘||VProc3 AS GTW

,VProcType4||’: ‘||VProc4 AS RSG

,VProcType5||’: ‘||VProc5 AS TVS

FROM DBC.ResUsageSpma

ORDER BY NodeID

Para descobrir a contagem de AMPs no sistema Teradata:

SELECT HASHAMP( 0 ) +1;

AMP:

Processadores do módulo de acesso responsáveis ​​por todas as funções relacionadas ao banco de dados, como executar consultas ao banco de dados, bloquear tabelas/linhas/bancos de dados, classificação, junção de linhas de dados, agregação, gerenciamento de espaço em disco.

PE:

Os mecanismos de análise executam controle de sessão, análise de consulta, verificação da existência de objetos, validação de segurança, otimização, plano de preparação, envio de consulta.

GTW:

O Gateway fornece uma interface de soquete entre a consulta do usuário e o mecanismo de análise do banco de dados Teradata.

RSG:

O Relay Services Gateway fornece uma interface de soquete para retransmitir as alterações do dicionário para o utilitário Teradata MDS (Meta Data Service).

TVS:

O Teradata Virtual Storage gerencia o armazenamento em disco. Ele fornece uma parte do armazenamento em disco para cada AMP funcionar de forma independente.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **NodeID** | **NodeType** | **AMPs** | **PEs** | **GTW** | **RSG** | **TVS** |
| 101 | XXXX | AMP : 24 | PE : 2 | GTW : 1 | RSG : 0 | TVS : 1 |
| 102 | XXXX | AMP : 24 | PE : 2 | GTW : 1 | RSG : 0 | TVS : 1 |
| 103 | XXXX | AMP : 24 | PE : 2 | GTW : 1 | RSG : 0 | TVS : 1 |
| 104 | XXXX | AMP : 24 | PE : 2 | GTW : 1 | RSG : 0 | TVS : 1 |

No exemplo acima, você pode ver que este é um sistema MPP (Massively Parallel Processing) com 4 nodes, cada um com 24 AMPs, 2 PEs, 1 GTW, no RSG and 1 TVS.

Ciclo de vida de uma consulta Teradata Pós-navegação

Os usuários do Teradata ou quaisquer outros aplicativos enviam uma consulta na forma de SQL (Structure Query Language) e recebem a resposta. Mas, nos bastidores, ele passa por muitos testes que determinam seu destino.

Abaixo estão as etapas que descrevem o ciclo de vida de uma consulta Teradata:

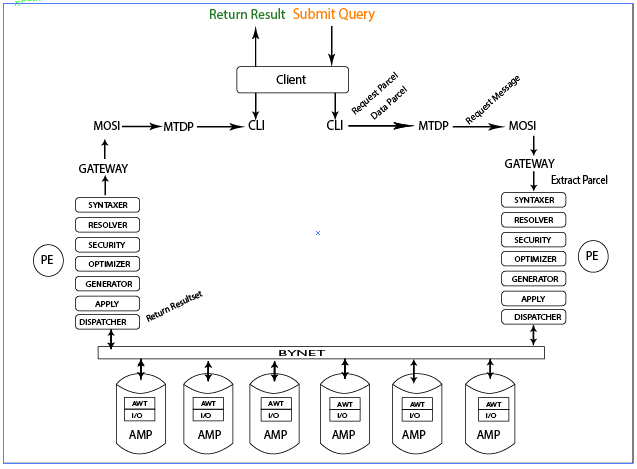
Envio de Consulta:

O ponto inicial de uma consulta do Teradata começa quando o usuário envia uma consulta. Uma interface de nível de chamada (CLI) prepara o pacote dessa consulta SQL. As encomendas podem ser de dois tipos –

* Pacote de solicitação – contém instrução SQL.
* Pacote de dados – contém quaisquer valores de parâmetro que podem ser fornecidos.

Conectar ao banco de dados:

Em seguida, as encomendas são enviadas para o Micro Teradata Director Program (MTDP), que estabelece uma sessão com o banco de dados Teradata. O MTDP prepara uma mensagem de solicitação contendo o pacote para transmissão ao banco de dados Teradata via rede. Um Micro Operating System Interface (MOSI) é usado para aceitar esta mensagem de solicitação para o sistema operacional e protocolo de rede como TCP/IP.



Extrair Parcela:

O PE recebe a mensagem de solicitação via gateway e o PE extrairá o pacote de solicitação e o pacote de dados. O PE é composto por vários módulos: sintaxador, resolvedor, segurança, otimizador, gerador e despachante.

Passa pela segurança:-

* Sintaxe: verifique a consulta SQL para obter a sintaxe adequada.
* Resolvedor: procura a existência de objetos no dicionário de dados.
* Segurança: Determine se o usuário ou banco de dados tem direitos de acesso apropriados.
* Otimizador: cria o plano ideal para uma consulta.
* Geração: Crie etapas para o plano vencedor.
* Dispatcher: Despacha as etapas para AMP via bynet.

Execute a consulta:

Cada AMP possui uma porção de dados e gerencia seus próprios conjuntos de bloqueio e possui seus próprios conjuntos de dicionário de dados. Cada um funciona independentemente dos outros AMPs. As etapas de execução são recebidas em paralelo por todos os AMPs participantes da etapa, e uma tarefa de trabalho AMP (AWT) em cada AMP é atribuída para executar a etapa.

O AWT pode classificar, agregar e redistribuir os dados conforme necessário para produzir um conjunto de respostas.

Produzir conjuntos de resultados:

Cada AMP participante retornará sua parte do resultado ao Bynet, que mesclará todo o conjunto de resultados, preparará o resultado final e o passará ao despachante.

Retornar Resposta ao usuário:

O Dispatcher agora empacota o resultado em um pacote de resposta e o despacha para o Gateway para entregar pela conexão de rede como mensagem de resposta. A parcela de resposta é então extraída da mensagem de resposta e o conjunto de respostas é devolvido ao usuário.

Tipos de Dados Teradata

Cada coluna em uma tabela está associada a um tipo de dados. Os tipos de dados simplesmente denotam o tipo de dados que podemos armazenar em qualquer coluna específica. Por exemplo, para armazenar dados de string como “James” em uma coluna, precisamos definir uma coluna como char ou varchar.

Sempre que estiver criando uma tabela no Teradata, você precisa definir os tipos de dados de cada coluna. Você pode escolher tipos de dados conforme sua necessidade.

Tipos de Dados Teradata

A seguir estão alguns dos tipos de dados usados ​​com frequência no Teradata.

|  |  |  |
| --- | --- | --- |
| **Data Types** | **Length (Bytes)** | **Range of values** |
| BYTEINT | 1 | -128 to +127 |
| SMALLINT | 2 | -32768 to +32767 |
| INTEGER | 4 | -2,147,483,648 to +2147,483,647 |
| BIGINT | 8 | -9,233,372,036,854,775,80 8 to +9,233,372,036,854,775,8 07 |
| DECIMAL | 1-16 |  |
| NUMERIC | 1-16 |  |
| FLOAT | 8 | IEEE format |
| CHAR | Fixed Format | 1-64,000 |
| VARCHAR | Variable | 1-64,000 |
| DATE | 4 | YYYYYMMDD |
| TIME | 6 or 8 | HHMMSS.nnnnnn or HHMMSS.nnnnnn+HHMM |
| TIMESTAMP | 10 or 12 | YYMMDDHHMMSS.nnnnnn or YYMMDDHHMMSS.nnnnnn +HHMM |

Primary Index no Teradata

Primary Index é o mecanismo físico para armazenar e recuperar linhas de dados no Teradata AMP. Cada tabela no Teradata deve ter pelo menos uma coluna como Índice Primário. É definido no momento da criação da tabela. Se alguma alteração no índice primário precisar ser implementada, é necessário descartar a tabela e recriá-la. PI não pode ser alterado ou modificado.

É o índice mais preferido e importante pelas razões abaixo:

Distribuição de Dados

Caminho de acesso conhecido

Melhora o desempenho da junção

Existem dois tipos de índices primários:

* Unique Primary Index (UPI).
* Non-Unique Primary Index(NUPI)

Vamos agora entender o que exatamente acontece quando você define um PI na tabela.

Unique Primary Index (UPI)

Como o nome sugere, um UPI permite valores exclusivos, ou seja, nenhuma duplicata é permitida. É uma operação de um AMP e a distribuição de dados é uniforme. Ele pode conter um valor nulo.

Sintaxe da Unique Primary Index (UPI):

CREATE TABLE sample\_1

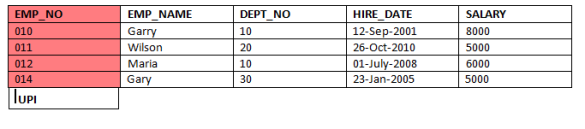
(col\_a INT,

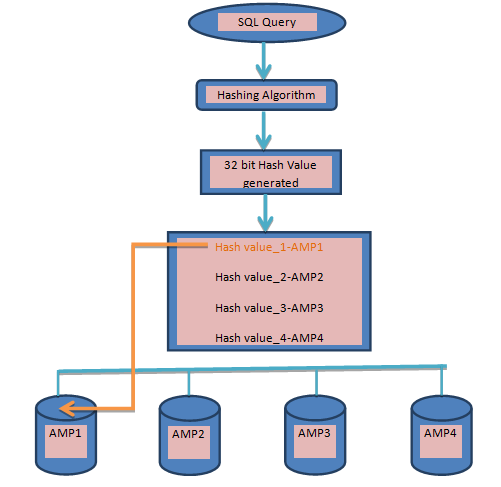
col\_b VARCHAR(20),

col\_c CHAR(4))

UNIQUE PRIMARY INDEX (col\_a);

Por exemplo: Temos uma tabela Employee onde EMP\_NO é o índice primário (escolhemos isso porque EMP\_NO é exclusivo para todos).

Data Distribution usando UPI:

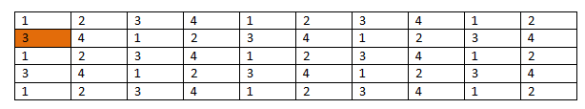


Exemplo de consulta:

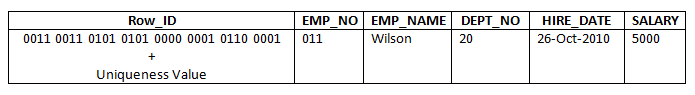
INSERT INTO DBNAME.EMPLOYEE VALUES (011,'Wilson',20,'2010-10-26',5000);

Quando um usuário envia uma consulta de inserção para uma tabela com índice primário, ocorrem os seguintes processos:

* O valor do índice passa por um algoritmo de hash e fornece um valor de hash de linha de 32 bits, algo como este 0011 0011 0101 0101 0000 0001 0110 0001 para EMP\_NO 011.
* Os primeiros 20 bits desse valor de hash de linha de 32 bits determinam o AMP no qual a linha residirá. Isso é decidido a partir do mapa Hash, que contém 1 milhão de baldes de hash. O mapa de hash se parece com isso para o sistema de 4 AMP.



* Portanto, agora o valor do hash apontará para o amplificador específico do HASH MAP. por exemplo: Nosso valor 0011 0101 0101 0000 0001 0110 0001 aponta para a 2ª linha, 1ª coluna, ou seja, AMP 3. Agora temos AMP 3 onde a linha residirá.
* O PE enviará a linha para o AMP com o valor de hash anexado a ele, algo assim:



* Um valor de exclusividade é definido para cada linha. Como EMP\_NO é único para todos, no caso de UPI o valor de exclusividade será 1 para todos. Isso pode ser bem compreendido enquanto estudarmos o NUPI. Então, é assim que a linha pode ser distribuída para o AMP. O mesmo é o processo de recuperação.

Non-Unique Primary Index(NUPI):

Um NUPI pode permitir duplicatas. Pode ter n número de valores nulos.

Sintaxe para NUPI:

CREATE TABLE sample\_1

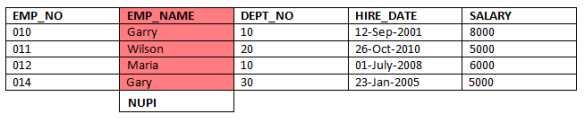
(col\_a INT,

col\_b VARCHAR(20),

col\_c CHAR(4))

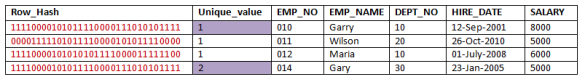
PRIMARY INDEX (col\_a);

Vamos pegar o mesmo exemplo para entender o NUPI.



Tomamos EMP\_NAME como a coluna NUPI, pois pode conter registros duplicados. De acordo com nossa tabela, o nome do funcionário Gary apareceu duas vezes, o que significa que é um valor duplicado. Agora vamos ver o que acontece quando o PE recebe um valor duplicado.

* O mesmo processo de geração do valor Row-Hash é seguido.
* Para diferenciar dentro das duplicatas, um valor de exclusividade é adicionado com o valor de hash, algo como abaixo:



* O valor de exclusividade é adicionado para tornar a linha exclusiva de todas as duplicatas. Se tivéssemos mais um funcionário como Gary, o valor de exclusividade seria 3 para ele.
* Como o AMP é selecionado com a ajuda do valor Hash, todos os valores duplicados irão para o mesmo AMP.



As duplicatas residem no mesmo AMP, portanto, isso leva a uma distribuição desigual de dados e pode prejudicar o desempenho.

Secondary Index no Teradata

Um índice secundário (SI) oferece um caminho alternativo para acessar os dados. Ao contrário do Índice Primário, que só pode ser definido no momento da criação da tabela, um Índice Secundário também pode ser criado/eliminado após a criação da tabela. Existem dois tipos de Índice Secundário:

* Unique Secondary Index (USI).
* Non-Unique Secondary Index (NUSI).

Assim que você define o índice secundário, o próximo movimento vem de Teradata criando uma subtabela em cada AMP. Isso é verdade tanto para a USI quanto para a NUSI. As subtabelas exigem espaço em disco adicional e manutenção; eles devem ser usados ​​apenas em consultas que são executadas repetidas vezes.

O objetivo de criar uma subtabela é apontar de volta para a linha real na tabela base por meio do Row-ID.

Antes de prosseguir com o índice secundário, é recomendável aprender sobre o índice primário no Teradata.

Agora você aprendeu sobre o índice primário que fornece o caminho mais rápido para acessar os dados e evitar a verificação completa da tabela. **Então, por que queremos definir o índice secundário em uma tabela?**

Como sabemos, podemos definir um e apenas um índice primário para qualquer tabela específica e acessar essa tabela será extremamente rápido somente se usarmos coluna(s) de índice primário na cláusula where. Mas e se acessarmos a tabela com alguma outra coluna na cláusula where? Nesse caso, ele irá para a varredura completa da tabela e algum pensamento alternativo vem em mente.

Pontos-chave sobre o Índice Secundário:

* Podemos definir até 32 índices secundários.
* Um índice secundário tem um limite de 64 colunas.
* O Teradata mantém uma subtabela separada para cada índice secundário.
* Pode ser definido/descartado dinamicamente.

Índice Secundário Único (USI)

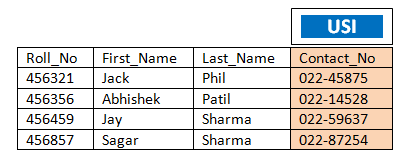
Como o nome sugere, USI impõe exclusividade em uma coluna ou grupo de valores de coluna. O Teradata criará uma subtabela em cada amplificador assim que você criar o USI em uma coluna ou grupo de colunas. A subtabela conterá abaixo três informações:-

* O valor do índice secundário
* O ID da linha do índice secundário
* O ID da linha da tabela base

Sintaxe para para criação do USI:

***CREATE UNIQUE INDEX (column\_name) on dbname.tablename;***

Criação da subtable:  
Considere a tabela do aluno abaixo como um exemplo para melhor compreensão:



Suponha que temos uma tabela chamada aluno com quatro colunas Roll\_No, First\_Name, Last\_Name, Contact\_No. A coluna Roll\_No foi definida como índice Primário. Portanto, a tabela será distribuída de acordo com Roll\_No no AMP.

Agora você definiu um índice secundário exclusivo na coluna Contact\_No conforme abaixo:

***CREATE UNIQUE INDEX (Contact\_No) on Mydb.student;***

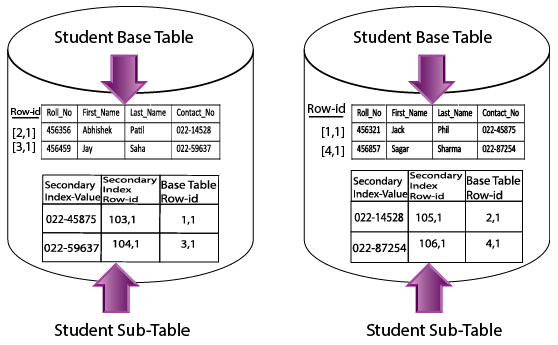
A Teradata executará as etapas abaixo para manter a USI:

* Assim que você definir USI, o Teradata criará uma subtabela para Contact\_no em cada AMP.
* Os valores da coluna USI (neste caso Contact\_No) serão escolhidos um a um e enviados para PE para hashing.
* Depois de fazer o hash do valor usando o algoritmo de hash e o mapa de hash, ele encontrará o amplificador de destino para qualquer valor específico.
* No AMP de destino, o valor do índice será armazenado junto com o id da linha do índice secundário e o id da linha base(row\_id).

Dessa forma, o Teradata cria e mantém a subtabela. Como a coluna USI contém apenas valores exclusivos, nenhum valor duplicado existe em qualquer subtabela, ou seja, quando você envia qualquer consulta usando USI, apenas uma linha exclusiva será retornada.

Acesso USI:

Agora vou explicar se você enviar qualquer consulta USI, como ela será processada e muitos AMP envolverão e por quê? É verdade universal que acessar dados usando USI é sempre 2 operações AMP. Você vai conhecer um pouco mais tarde.



Suponha que você tenha a consulta USI abaixo:

***SELECT \* FROM mydb.student where Contact\_No = 022-45875;***

Depois que a consulta acima passar por toda a verificação de sintaxe e segurança e o otimizador Teradata encontrar a coluna associada à cláusula where é uma coluna USI, o Teradata buscará o registro conforme abaixo:

* O algoritmo de hash faz o hash do valor USI, ou seja, 022-45875 e gera o valor de hash.
* Usando o mapa de hash, ele encontrará o número do amplificador onde o valor SI está armazenado.
* Agora ele irá para a subtabela do funcionário e buscará o id da linha da tabela base do funcionário.
* Assim que o ID da linha for recebido, ele buscará o número do amplificador onde a linha da tabela base reside usando o mapa de hash.
* Assim, a linha agora é encontrada na tabela base.

A partir das etapas acima, agora está claro que a recuperação de dados usando USI é uma operação sempre de 2 AMPS e retorna apenas uma linha. O primeiro AMP é usado para buscar o ID da linha base da subtabela e o segundo é para o valor real da tabela base.

Non-Unique Secondary Index (NUSI):

Um índice secundário não exclusivo (NUSI) é projetado para impedir a varredura completa da tabela (FTS) e geralmente contém valores duplicados.

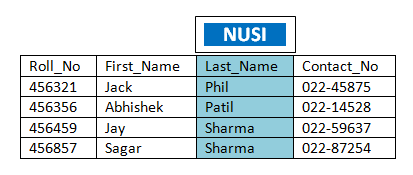
Depois de criar um índice secundário, uma subtabela é criada em cada AMP.

**A principal diferença entre USI e NUSI é que as linhas da subtabela USI são hash e as linhas da subtabela NUSI são AMP-Local. Mas por que NUSI é AMP Local e o que isso significa?**

É óbvio que o valor NUSI é duplicado e pode haver um grande número de valores duplicados nele. Portanto, o Teradata adota uma estratégia diferente para manter a subtabela neste caso. Cada linha da subtabela rastreia apenas as linhas base no mesmo AMP. **Isso é o que significa AMP Local.**

Sintaxe para criar USI:

***CREATE INDEX (column\_name) on dbname.tablename;***



Agora aprendemos que o valor NUSI não é usado HASH MAP para encontrar o AMP de destino. Depois de definir Last\_Name como um NUSI, o Teradata fará o hash do valor Last\_Name e armazenará o ID da linha junto com o valor do índice na subtabela do mesmo AMP. Portanto, a subtabela NUSI é AMP local. Outra diferença é que apenas uma linha de subtabela é mantida, embora possa haver várias linhas para um valor NUSI no mesmo AMP. Nesta base múltipla, o id de linha será mantido em uma única linha.

Recuperação de dados usando NUSI:

Considere a consulta abaixo

***SELECT \* FROM mydb.student where Last\_Name = ‘Sharma’;***

* Depois que o Parsing Engine descobre que Last\_Name está definido como NUSI, ele usa o algoritmo de hash para gerar o valor de hash que circula para todos os AMP. (No nosso caso, o valor NUSI é Sharma)
* Cada Amp agora começa a corresponder ao valor de hash em sua subtabela de aluno.
* O AMP que não possui esse valor de hash não participará mais dessa operação.
* Outros AMPs participantes encontram o id de linha base em seu aluno e as linhas reais são buscadas na tabela base e retornadas ao cliente.

Join Index no Teradata

Um índice de junção é uma estrutura de indexação que contém colunas de várias tabelas, especificamente as colunas resultantes de uma ou mais tabelas. Em vez de ter que juntar uma tabela individual sempre que a operação de junção for necessária, a consulta pode ser resolvida por meio de um índice de junção e, na maioria dos casos, melhorar drasticamente o desempenho.

* É usado para definir uma tabela de pré-junção na coluna de junção frequente.
* Você pode definir o índice de junção no Teradata em tabelas únicas ou múltiplas.
* Pode ser definido para criar uma replicação parcial ou completa da tabela base com um índice primário diferente.
* O Join Index é armazenado no espaço permanente e mantido pelo sistema automaticamente.
* Você não pode disparar nenhuma consulta diretamente no índice de junção.

Lembre-se de que a definição do índice de junção no Teradata não implica que o mecanismo de análise sempre usará o índice de junção. Cabe ao mecanismo de análise Teradata usar o índice de junção ou acessar a partir da tabela base. Existem diferentes tipos de índice de junção no Teradata:

Single Table Join Index(STJI):

Um único índice de junção de tabela é usado para criar uma estrutura de indexação para uma única tabela, mas com índice primário diferente. Isso melhora o desempenho das junções, pois nenhuma distribuição ou duplicação é necessária. O usuário consultará a tabela base, mas o PE decidirá se deseja acessar a tabela base ou o índice de junção de tabela única.

Sintaxe do índice de junção de tabela única:

CREATE JOIN INDEX STUDENT\_IX

AS

SELECT ROLL\_NO,STUDENT\_NAME, DEPT\_NO, PH\_NO

FROM STUDENT\_BASE\_TABLE

PRIMARY INDEX(DEPT\_NO);

Aqui, a coluna DEPT\_NO é definida como índice primário de STJI. Portanto, durante o processamento de junção com a tabela DEPARTMENT em DEPT\_NO, não há necessidade de redistribuir a tabela STUDENT\_BASE\_TABLE usando a coluna DEPT\_NO, pois já definimos STUDENT\_IX com o primário a partir de DEPT\_NO.

Multi Table Join Index(MTJI):

Um índice de junção de várias tabelas é usado para manter o conjunto de resultados pré-junção de duas ou mais colunas. Portanto, durante o processamento de junção, o PE pode decidir acessar os dados do MTJI em vez de unir novamente as tabelas base subjacentes. Precisamos lembrar que devemos definir MTJI após muita análise com base na frequência e custo de adesão.

Sintaxe do índice de junção de várias tabelas:

CREATE JOIN INDEX STUDENT\_DEPT

AS

SELECT ROLL\_NO, STUDENT\_NAME ,DEPT\_NO, ADDRESS, PH\_NO

FROM STUDENT\_BASE\_TABLE A

INNER JOIN

ON A. DEPT\_NO = B. DEPT\_NO

UNIQUE  PRIMARY INDEX(ROLL\_NO);

Aggregate Join Index:

Índice de junção agregada:

Os índices de junção agregados oferecem um método extremamente eficiente de resolver consultas que frequentemente especificam as mesmas operações agregadas na mesma coluna ou colunas. Quando os índices de junção agregados estão disponíveis, o sistema não precisa repetir os cálculos agregados para cada consulta. Um índice de junção agregado pode ser definido em duas ou mais tabelas ou em uma única tabela.

Um índice de junção agregado pode ser criado usando:

* função SOMA
* função CONTAGEM
* Cláusula GROUP BY

Exemplo de índice de junção agregada:

CREATE JOIN INDEX SALES\_INX STORE\_NO, SUM(QTY\_SOLD)

FROM SALES\_BASE\_TABLE

GROUP BY 1, ORDER BY 2;

Sparse Join Index:

Sparse Join indexa uma parte da tabela usando predicados da cláusula WHERE para limitar as linhas indexadas. Quando as tabelas base são grandes, esse recurso pode ser usado para reduzir o conteúdo do índice de junção apenas para a parte da tabela que é usada com frequência se a consulta típica fizer referência apenas a uma parte das linhas. Esta capacidade:

Reduza o requisito de espaço para um índice de junção.

Reduz o custo para manter o índice de junção.

Sintaxe do índice de junção esparsa:

CREATE JOIN INDEX SPARSE\_BILL\_INX

AS

SEL ACC\_NO, CUST\_NAME, BILL\_DATE, BILL\_AMT

FROM BILL\_DETAILS

WHERE BILL\_YEAR=2015

UNIQUE PRIMARY INDEX(ACC\_NO);

Limitação do índice de junção:-

* Os índices Join não são suportados pelos utilitários Fastload e Multiload. Eles devem ser eliminados e recriados após o carregamento da tabela.
* Durante a restauração de uma tabela base ou índice de junção de banco de dados é marcado como inválido.
* Máximo de 64 colunas por tabela por índice de junção.
* As subtabelas de índice de junção não podem ser protegidas contra fallback.

Hash Index in Teradata

O índice de hash no Teradata foi projetado para melhorar o desempenho da consulta do Teradata de maneira semelhante ao índice de junção de tabela única, que restringe o mecanismo de análise para não acessar ou redistribuir a tabela base. Ele pode atuar como um índice secundário, fornecendo um caminho de acesso alternativo para a tabela base.

Diferença entre o índice de hash e o índice de junção de tabela única:

* A principal diferença externa entre o Hash Index e o Single table join Index está na sintaxe para criá-los. É mais fácil criar um índice de hash do que um índice de junção de tabela única funcionalmente comparável.
* O índice de hash deve ter um índice primário, mas o índice de junção de tabela única pode ou não ter.
* Não podemos definir um índice primário particionado em um índice de hash, mas podemos defini-lo em um índice de junção de tabela única.
* Não podemos especificar índice secundário não exclusivo, função de agregação, gatilho no índice de hash, mas tudo isso podemos definir no índice de junção de tabela única.

Nota: Tanto o índice de hash quanto o índice de junção de tabela única são definidos para melhorar o desempenho da consulta. Um índice de hash é uma forma restrita de um índice de junção de tabela única, portanto, o desempenho de ambos deve ser o mesmo. Considere abaixo exemplos do índice de hash para entender o processo de criação:

Example table:

CREATE BASE TABLE EMPLOYEE (

EMP\_ID INTERGER NOT NULL,

EMP\_SSN\_NO INTEGER NOT NULL,

EMP\_NAME VARCHAR(100),

MANAGER\_ID INTEGER NOT NULL,

MANAGER\_NAME VARCHAR(100),

DEPT VARCHAR(30))

UNIQUE PRIMARY INDEX(EMP\_ID);

Join Index Syntax:

CREATE HASH INDEX EMP\_HI (

EMP\_SSN\_NO,

DEPT

)

ON EMPLOYEE;

OR

CREATE HASH INDEX EMP\_HI (

EMP\_ID,

EMP\_SSN\_NO,

DEPT

)

ON EMPLOYEE

BY (EMP\_ID)

ORDER BY HASH (EMP\_ID);

A partir do exemplo acima, as características do índice de hash podem ser observadas como:

* Não há necessidade de incluir a coluna EMP\_ID, pois é o índice principal da tabela base, portanto, será incluída automaticamente no índice de hash.
* As linhas da subtabela do índice de hash serão AMP local para a tabela base e as sequências de hash de linha estão na sequência de hash com base no índice primário da tabela base.

Example 2:

CREATE HASH INDEX EMP\_HI (

EMP\_ID,

EMP\_SSN\_NO,

DEPT

)

ON EMPLOYEE

BY (EMP\_ID)

ORDER BY VALUES (EMP\_SSN\_NO);

Neste exemplo:

Como antes, a subtabela do índice de hash será AMP local, pois os valores serão redistribuídos com base no valor do índice primário da tabela base.

Mas, neste caso, as linhas de índice de hash serão armazenadas na sequência EMP\_SSN\_NO em vez da sequência de hash de linha.

Esse tipo de índice de hash é útil para consultas de intervalo.

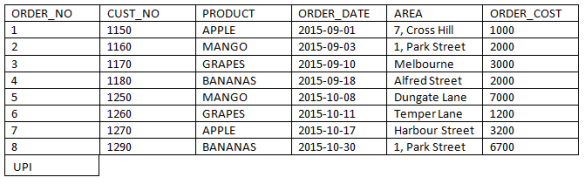
Partitioned Primary Index in Teradata

O índice primário particionado (PPI) é um dos recursos poderosos do Teradata que permite que um usuário acesse uma parte da tabela em vez do acesso total à tabela. Isso ajuda a melhorar o desempenho, pois a verificação completa da tabela é eliminada. O PPI funciona da mesma forma que o Índice Primário para a distribuição de dados, mas cria partições de acordo com o intervalo ou caso especificado na tabela.

Existem 4 tipos de IBP:

1. Particionamento de caso.
2. Particionamento baseado em intervalo.
3. Particionamento em vários níveis.
4. Particionamento baseado em caracteres.

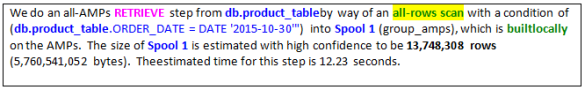
Vamos pegar um exemplo de tabela de produtos (uma parte da tabela é tomada aqui) e tentar entender todos os métodos acima de porcionamento de uma tabela:



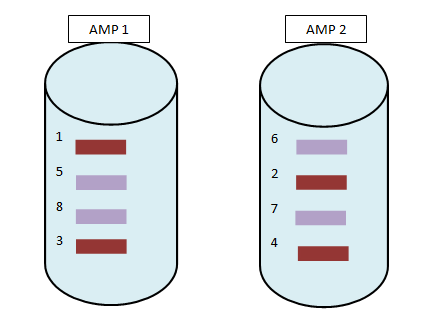
Você executa a consulta abaixo:

SEL \* FROM DB.PRODUCT\_TABLE WHERE ORDER\_DATE = '2015-10-30';

Explicar o plano para isso será mais ou menos assim:



Agora, quando os dados forem distribuídos, eles ficarão mais ou menos assim. Nesse caso, as linhas serão classificadas em cada AMP com base no ID da linha.



Isso levará a uma verificação completa da tabela. Para evitar isso, podemos usar o índice primário da partição.

RANGE PARTITIONING:

CREATE TABLE PRODUCT\_TABLE (

ORDER\_NO INTEGER NOT NULL,

CUST\_NO INTERGER,

PRODUCT VARCHAR(20),

ORDER\_DATE DATE,

AREA VARCHAR(50),

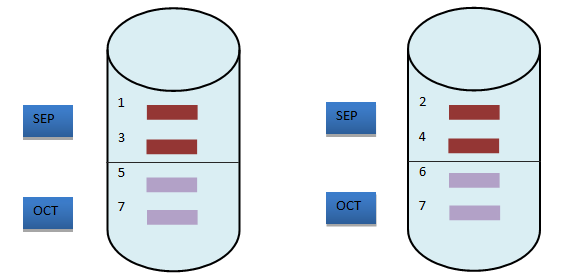
ORDER\_COSTDECIMAL(10,2) )

PRIMARY INDEX(ORDER\_NO)

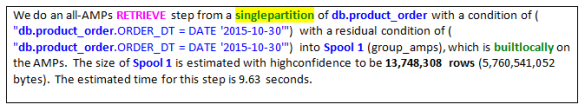
PARTITION BY RANGE\_N (ORDER\_DATE BETWEEN DATE '2015-01-01'

AND DATE '2015-12-31' EACH INTERVAL '7' DAY);

Quando criamos uma tabela usando isso, a distribuição será algo como abaixo:



Como podemos ver acima, agora as linhas foram particionadas de acordo com ORDER\_DATE. O plano de explicação será semelhante a:



Agora, a partir do plano de explicação acima, podemos ver que a recuperação de dados está demorando menos e não indo para a varredura completa da tabela, portanto, melhorará o desempenho da consulta.

CASE PARTITIONING:

CREATE TABLE PRODUCT\_TABLE

(

ORDER\_NO INTEGER NOT NULL,

CUST\_NO INTERGER,

PRODUCT VARCHAR(20),

ORDER\_DATE DATE,

AREA VARCHAR(50),

ORDER\_COSTDECIMAL(10,2)

)

PRIMARY INDEX(ORDER\_NO)

PARTITION BY CASE\_N(ORDER\_COST< 1000,

ORDER\_ COST< 2000,

ORDER\_ COST< 5000,

NO CASE, UNKNOWN);

A condição NO CASE é aplicada para os valores que não atendem aos critérios de caso e UNKNOWN para valores NULL, se houver. Isso funcionará da mesma forma que o particionamento de intervalo, apenas a diferença estaria no valor do particionamento. Aqui os dados serão particionados de acordo com o ORDER\_COST.

MULTILEVEL PARTITIONING:

O número máximo de partições que você pode definir é 65.535.

CREATE TABLE PRODUCT\_TABLE

(

ORDER\_NO INTEGER NOT NULL,

CUST\_NO INTERGER,

PRODUCT VARCHAR(20),

ORDER\_DATE DATE,

AREA VARCHAR(50),

ORDER\_COSTDECIMAL(10,2)

)

PRIMARY INDEX(ORDER\_NO)

PARTITION BY (RANGE\_N(ORDER\_DATE BETWEEN DATE '2015-01-01' AND DATE '2015-12-31'

EACH INTERVAL '1' DAY)

CASE\_N (ORDER\_TOTAL < 1000,

ORDER\_TOTAL < 2000,

ORDER\_TOTAL < 3000,

NO CASE, UNKNOWN));

Character based partitioning:

CREATE TABLE EMPLOYEE (

EMP\_ID INTEGER,

LAST\_NAME VARCHAR (30) CHARACTER,

FIRST\_NAME VARCHAR(30),

CITY VARCHAR(50))

PRIMARY INDEX (EMP\_ID)

PARTITION BY CASE\_N (LAST\_NAME LIKE 'A%', LAST\_NAME LIKE 'B%',

LAST\_NAME LIKE 'C%', LAST\_NAME LIKE 'D%',

LAST\_NAME LIKE 'E%', LAST\_NAME LIKE 'F%',

LAST\_NAME LIKE 'G%', LAST\_NAME LIKE 'H%',

LAST\_NAME LIKE 'I%', LAST\_NAME LIKE 'J%',

LAST\_NAME LIKE 'K%', LAST\_NAME LIKE 'L%',

LAST\_NAME LIKE 'M%', LAST\_NAME LIKE 'N%',

LAST\_NAME LIKE 'O%', LAST\_NAME LIKE 'P%',

LAST\_NAME LIKE 'Q%', LAST\_NAME LIKE 'R%',

LAST\_NAME LIKE 'S%', LAST\_NAME LIKE 'T%',

LAST\_NAME LIKE 'U%', LAST\_NAME LIKE 'V%',

LAST\_NAME LIKE 'W%', LAST\_NAME LIKE 'X%',

LAST\_NAME LIKE 'Y%', LAST\_NAME LIKE 'Z%',

NO CASE, UNKNOWN);

O exemplo acima criará partições com base no sobrenome.

Vantagens do PPI:

1. Evita a verificação completa da tabela e apenas a parte relevante da tabela é verificada.
2. Inserção rápida, operações de exclusão.
3. Pode ser criado em tabela temporária global, tabela volátil, índices de junção não compactados.
4. Para consultas baseadas em intervalo, podemos remover SIs e usar PPI, o que reduzirá o espaço de uma subtabela SI de sobrecarga.

Desvantagens do PPI:

1. Um adicional de 2 bytes é adicionado a cada linha e, portanto, aumenta o espaço perm.
2. Ao ingressar em uma tabela PPI com uma tabela não particionada, pode levar muito tempo.
3. O acesso diferente da coluna PPI pode levar mais tempo.

Teradata Space

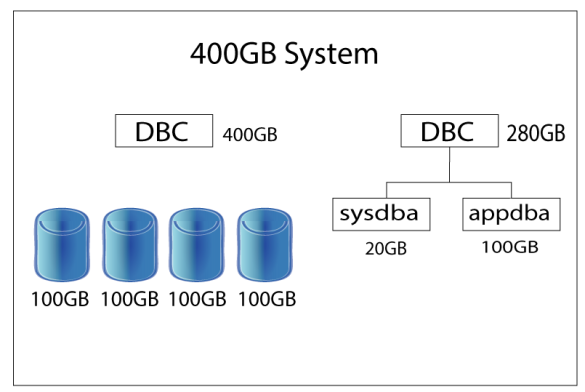
Existem basicamente três tipos de espaços no Teradata. Estes são:

1. Espaço Permanente.
2. Espaço Carretel.
3. Espaço Temporário.

Vamos entender brevemente sobre eles.

ESPAÇO PERMANENTE (Perm Space):

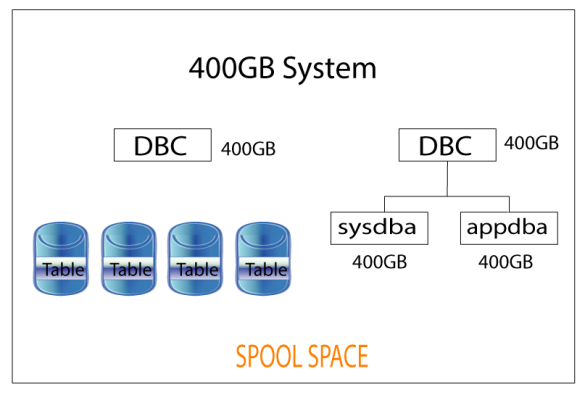
Este é o espaço onde objetos como bancos de dados, usuários, tabelas, visualizações, etc., são criados e armazenados. O espaço permanente é liberado quando os dados são excluídos ou quando os objetos são removidos do banco de dados. O espaço permanente define o limite superior de espaço para um usuário ou banco de dados. Suponha que você tenha um usuário com acesso ao Teradata. Quando você cria qualquer tabela, procedimento de armazenamento, etc., o espaço ocupado será do espaço Perm. O espaço permanente alocado para um usuário ou um banco de dados que não está sendo usado no momento está disponível para uso como spool ou espaço temporário.



Suponha que tenhamos um sistema Teradata com 400 GB de espaço permanente. Inicialmente, o DBC possui 100% do espaço permanente de um novo sistema. Como no Teradata tudo é paralelo, o espaço perm será distribuído uniformemente por todos os AMPs. Sistema Teradata com 400 GB de espaço permanente e 4 AMPs, cada AMP terá 100 GB de espaço permanente para executar a consulta em paralelo. Se o DBC criar dois usuários como sysdba e appdba com 20 GB e 100 GB de espaço perm, respectivamente, o espaço será retirado do pai, ou seja, aqui DBC. Portanto, agora o DBC tem 280 GB de espaço permanente.

SPOOL SPACE in Teradata:

Este espaço é literalmente um espaço permanente não utilizado que é usado para armazenar o resultado intermediário da consulta disparada pelo usuário final. Sempre que um usuário final executa uma consulta, a saída armazenada temporariamente em algum espaço é o espaço de spool. Isso não deve exceder o limite de espaço de spool fornecido, pois a consulta é interrompida imediatamente e você obtém um problema de spool out. Se o espaço de spool não for especificado para o usuário/banco de dados, ele herdará o espaço de seu pai. O espaço de spool é retirado do espaço Perm e é revertido assim que a janela de saída é fechada pelo usuário.



Suponha que 10 GB de espaço permanente tenham sido alocados para um banco de dados. No entanto, o espaço perm real ocupado pelas tabelas ocupou 50% do espaço perm, ou seja, 5 GB. Portanto, o espaço restante de 5 GB está disponível em todo o sistema para o spool. O espaço de spool para o filho não será subtraído de seus pais imediatos, mas o espaço de spool do banco de dados filho pode ser tanto quanto o pai imediato. O espaço de spool para sysdba e appdba é igual ao DBC.

TEMPORARY SPACE:

Este espaço é usado principalmente nas tabelas temporárias globais. O espaço Perm não utilizado é usado como espaço temporário para armazenar a tabela e os dados nela contidos. Se você estiver especificando algum valor para o espaço Temp, ele não deve exceder o valor do banco de dados/usuário pai. Se você não especificar um valor, o valor máximo será herdado do pai.

SET VS. MULTISET TABLE in Teradata

Temos dois tipos de tabelas no Teradata, SET e tabela Multiset e a principal diferença dessas tabelas é que a tabela SET não permite linha duplicada enquanto a tabela Multiset permite, conforme abaixo:

Uma tabela definida(SET TABLE) não permite nenhuma linha duplicada nela. Por exemplo:

Insert into set\_table values (1,2,3);  
Insert into set\_table values (2,3,4);  
Insert into set\_table values (1,2,3); – Não permitido (duplicado)

Uma tabela Multiset(MULTISET TABLE) permite linhas duplicadas nela. Por exemplo:

Insert into multiset\_table values (1,2,3);  
Insert into multiset\_table values (2,3,4);  
Insert into multiset\_table values (1,2,3); – Permitido(duplicado)

Há impacto no desempenho envolvido com a tabela SET. Uma tabela SET não permite linha duplicada, portanto, para cada nova linha inserida ou atualizada na tabela, o Teradata verifica a violação da restrição de exclusividade para cada nova linha (inserida ou atualizada). causar sérios problemas de desempenho se o número de registros for grande.

Para evitar o problema de desempenho acima, é aconselhável definir UPI, USI ou qualquer outra restrição de exclusividade em qualquer coluna na tabela SET. restrição de exclusividade, portanto, o desempenho melhorará.

Nota: DUPLICATE ROW CHECKS cresce exponencialmente com o número de registros por valores de Índice Primário (PI).

Se inserirmos dados usando “Inserir na tabela1, selecione \* da tabela2”, a tabela SET filtrará registros duplicados automaticamente e não haverá erro de linha duplicada.

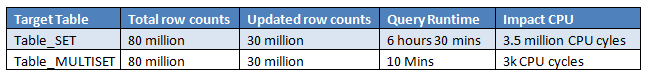
Caso insiramos dados usando a cláusula “insert into table2 values”, ele verificará registros duplicados e se houver algum registro duplicado.

Aqui está um exemplo que encontrei com SET table-

Uma tabela de destino SET foi definida com NUPI (índice primário não exclusivo) e 80 milhões de registros e a consulta estava tentando atualizar 30 milhões de linhas nela. Como é definido como tabela SET, para cada um dos 30 milhões de linhas, o Teradata estava executando DUPLICATE ROW CHECK na tabela de destino, imagine qual seria o impacto no sistema em tal cenário.

Se você alterar a estrutura da tabela como MULTISET e incorporar a cláusula group by na tabela de origem, o impacto no desempenho poderá ser evitado, pois o Teradata ignorará as verificações de linha duplicada.

Abaixo está o resultado de desempenho para a declaração de atualização:



A partir do exemplo acima, sabemos como a escolha errada da estrutura da tabela pode afetar o sistema.

É altamente recomendável definir a tabela de destino como tabela Multiset se estivermos usando qualquer cláusula GROUP BY ou QUALIFY na tabela de origem.

NOPI (No primary index) Table in Teradata

Como o nome sugere, Nenhuma tabela de índice primário significa que não haverá nenhum índice primário definido na tabela Teradata. Este recurso foi introduzido na versão V2R13. O principal objetivo desse recurso que nos permite criar tabelas sem índice primário no Teradata é melhorar o desempenho das operações de carregamento de dados de inserção de array FastLoad e Tpump.

Podemos usar esse tipo de tabela na área de preparação do data warehouse.

Sintaxe para tabela NOPI – CREATE Table:

1. **CREATE MULTISET TABLE**teradatapoint  
           (Column1 INTEGER **NOT NULL**,  
            Column2 DATE FORMAT ‘MM/DD/YYYY’ **NOT NULL**,  
            Column3 INTEGER)  
   **NO PRIMARY INDEX;**

      2.  **CREATE MULTISET TABLE**teradatapoint\_tmp **AS**  
           (**SELECT**\* **FROM**teradatapoint ) **WITH DATA NO PRIMARY INDEX;**

Nesse caso, a tabela teradatapoint atuará como uma tabela de origem e pode ser uma tabela PI ou uma tabela NOPI.

Distribuição de dados:

Como sabemos que não há índice primário envolvido na tabela NOPI, suas linhas não são hash para AMP com base no valor do índice primário. Banco de dados Teradata, hashes no ID da consulta ou use algum algoritmo diferente para atribuir as linhas ao seu AMP de destino.

Quando a linha chega ao AMP de destino, o Teradata gera um ROWID para ela selecionando um balde de hash aleatório que é AMP local.

Ao contrário do FastLoad na tabela PI normal, no caso da tabela NOPI, nenhum hash, redistribuição e classificação são necessários. Na tabela NOPI, as linhas serão anexadas ao bloco de dados da tabela, portanto, a sobrecarga de classificação de linhas com base em ROWID será evitada aqui. Além disso, como as linhas são atribuídas aleatoriamente aos AMPs, os dados sempre serão distribuídos uniformemente em todos os AMPs e não ocorrerá distorção.

Changing NOPI table to PI table:

Depois que a tabela NOPI é criada, podemos alterá-la para tabela PI de duas maneiras:

1.   **CREATE MULTISET TABLE**teradatapoint\_tmp **AS**  
     (**SELECT**\* **FROM**teradatapoint )  
**WITH PRIMARY INDEX (**COLUMN\_NAME**);**

Aqui a tabela teradatapoint é uma tabela NOPI.

1. Crie uma nova tabela com PI e execute inserir-selecionar da tabela NOPI para a tabela PI:
2. **CREATE MULTISET TABLE**teradatapoint\_tmp (  
           Column1 INTEGER **NOT NULL**,  
           Column2 DATE FORMAT ‘MM/DD/YYYY’ **NOT NULL**,  
           Column3 INTEGER)  
   **PRIMARY INDEX (**Column1**);**
3. **Insert into**teradatapoint\_tmp **Select**\* **from**teradatapoint**;**

Limitação da tabela NOPI:

* Não é possível criar uma tabela NOPI como uma tabela SET.
* MultiLoad não é suportado pela tabela NOPI.
* Nenhum diário permanente possível.
* Não é possível definir PPI, índice de hash.
* Não é possível usar como fila e tabela de erros.
* Atualização e upsert não são suportados.

Permanent Journal in Teradata

O diário permanente no Teradata fornece a capacidade de capturar o instantâneo da(s) tabela(s) antes e depois de aplicar alterações usando instruções DML como inserir, excluir e atualizar. Ao contrário do diário temporário, que captura o instantâneo da tabela antes de aplicar as alterações e descartar na confirmação, ele armazena na tabela do diário permanentemente e requer espaço permanente do Teradata até que você o descarte explicitamente.

Vantagens do jornal permanente:

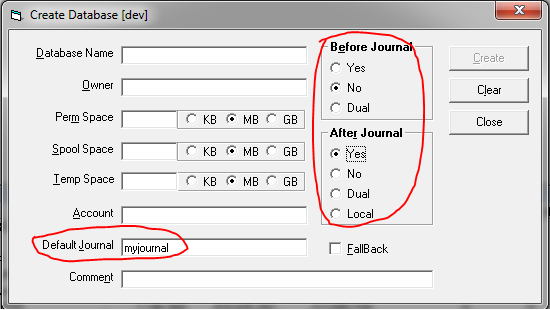
O diário permanente protege os dados do usuário contra abaixo:

* perda de dados devido a falha de disco em uma tabela que não é fallback ou protegida por RAID.
* perda de dados devido à falha de 2 ou mais AMPs no mesmo cluster.
* Processamento incorreto de lote ou aplicativo.

Você pode definir diário permanente em nível de banco de dados ou nível de tabela. A única tabela de diário será mantida em um banco de dados ou espaço de usuário. Depois de definir o diário permanente para tabela/tabelas e começar a fazer operações como inserir, excluir e atualizar na tabela, o diário permanente começará a crescer e ocupará mais espaço. Portanto, é responsabilidade do DBA manter o diário permanente, excluindo, arquivando e assim por diante. Durante a criação do diário permanente da tabela, você pode especificar para capturar o instantâneo da tabela conforme abaixo-

**Before journal / before image:** O instantâneo será capturado antes de aplicar qualquer alteração na tabela, usado para reverter ou desfazer as alterações. Você pode escolher o instantâneo da tabela como cópia única/dupla ou nenhuma cópia.

**After journal/after image:** O instantâneo será capturado após a aplicação de quaisquer alterações na tabela, usado para rolar ou refazer as alterações. Você pode escolher o instantâneo da tabela como cópia única/cópia dupla/sem cópia ou cópia local que será criada no mesmo AMP. Primeiro, você precisa criar uma tabela de diário no mesmo banco de dados ou em qualquer outro banco de dados ou espaço do usuário, conforme abaixo:



CREATE DATABASE "Mydb" FROM "TESTDB"

AS PERM=10737418240

NO FALLBACK

NO BEFORE JOURNAL

AFTER JOURNAL

DEFAULT JOURNAL TABLE="Mydb"."Myjournal";

Aqui optamos pelo after journal. Você pode substituir isso no nível da tabela especificando “NO AFTER JOURNAL”.

Sintaxe para definir diário permanente no nível da tabela:

Create Set Table "MyDB"."My\_Table"

, No Fallback

, No Before Journal

, After Journal

, With Journal Table = "MyDB"."journals"

("COLUMN1" CHAR

,"COLUMN2? CHAR

) INDEX ("COLUMN1") INDEX ("COLUMN1");

No exemplo acima, podemos ver que a imagem posterior de “MyDB”.”My\_Table” será capturada na tabela de diário “MyDB”.”Myjournals”.

Exclua a sintaxe da tabela de diário permanente:

Se você acha que o diário permanente não é mais necessário, pode eliminá-lo depois de modificar as propriedades da tabela e do banco de dados.

ALTER TABLE Mydb.Tablename

,NO BEFORE JOURNAL

,NO AFTER JOURNAL;

MODIFY USER Mydb.Tablename AS

no after journal;

MODIFY USER Mydb.Tablename AS

DROP DEFAULT JOURNAL table= Mydb.journals;

Você pode verificar tabelas de diário em todo o sistema usando a consulta abaixo:

SELECT \* FROM DBC.JOURNALS;

Recovery Journal in Teradata

Diário de Recuperação Down-AMP:

O diário de recuperação fornece a capacidade de continuar a operação normal de gravação, como inserir, atualizar, criar e excluir em uma tabela, mesmo que um AMP lógico fique inativo no mesmo cluster.

Suponha que tenhamos ativado a opção de fallback para uma tabela e um AMP seja desativado em um cluster. Assim que o AMP diminuir, o diário de recuperação de down-amp será ativado e capturará todas as alterações feitas na tabela protegida contra fallback. Isso significa que, durante o tempo de inatividade, o diário de recuperação de down-amp é o representante do down-amp até que o AMP volte.

Quando o AMP voltar, todas as alterações serão aplicadas de volta à tabela original. O diário de recuperação é descartado quando o AMP inativo é recuperado e totalmente funcional.

Diário Transitório:

Diário transitório é o mecanismo para capturar antes da imagem dos dados durante o processamento da transação. Isso ajuda a reverter para o estado anterior dos dados se a transação falhar por qualquer motivo.

Um diário transitório é usado durante o processamento da transação para manter o instantâneo anterior das linhas alteradas. Isso ajuda a restaurar o estado anterior dos dados se uma transação falhar. No ponto inicial de qualquer transação, o software Teradata ativa o diário transitório e começa a capturar as linhas afetadas para uma transação.

Depois que a transação for bem-sucedida, as imagens anteriores serão descartadas para a transação específica do log de diário transitório.

Teradata Locks

Todos os objetos no banco de dados são compartilhados entre vários usuários que acessam os dados em paralelo. Por exemplo, um usuário está atualizando uma tabela e se outro usuário tentar acessar essa tabela ao mesmo tempo, o segundo usuário obterá informações inconsistentes e incorretas dessa tabela. Para evitar esse tipo de inconsistência de dados, foi inventado um mecanismo de bloqueio.

A Teradata possui um gerenciador de bloqueio dedicado para bloquear automaticamente no banco de dados, tabela e nível de hash de linha.

Bloqueio do banco de dados(**Database Lock**): o bloqueio será aplicado a todas as tabelas e exibições.

Bloqueio de tabela(**Table Lock**) Bloqueio aplicado a todas as linhas na tabela/exibição.

Bloqueio de hash de linha(**Row hash Lock**): Uma ou várias linhas em uma tabela serão bloqueadas.

A Teradata usa quatro tipos de bloqueios para bloquear seus objetos:

O bloqueio exclusivo:

Quando alguém tenta modificar a estrutura de qualquer objeto, como uma tabela ou exibição, o Teradata aplica um bloqueio exclusivo a ele. Isso significa que nenhum outro usuário pode acessar ou qualquer tipo de operação até que o Teradata libere o bloqueio desse objeto.

O bloqueio exclusivo pode ser aplicado apenas no nível do banco de dados ou no nível da tabela. Este é o nível de bloqueio mais restritivo no Teradata.

Por exemplo, se enviarmos qualquer instrução CREATE TABLE, o Teradata aplicará um bloqueio exclusivo na tabela.