1: Teradata SQL > Default Database

Após a conclusão deste módulo, você deverá ser capaz de:

Descrever a estrutura de um Sistema de Gerenciamento de Banco de Dados Relacional (RDBMS).

Explique o papel da linguagem de consulta estruturada (SQL) no acesso a um banco de dados relacional.

Liste as três categorias de instruções SQL e descreva sua função.

Descrever os usos da instrução Teradata SQL SELECT.

Recupere dados de uma tabela relacional usando a instrução SELECT.

Use as opções SQL ORDER BY e DISTINCT.

Defina um banco de dados padrão usando o comando DATABASE.

Escreva instruções SQL usando as convenções de codificação recomendadas.

Nomeie os objetos do banco de dados de acordo com as regras Teradata SQL.

1 TeraData SQL > What is an RDBMS?

Os dados são organizados em tabelas em um Sistema de Gerenciamento de Banco de Dados Relacional (RDBMS). As linhas na tabela representam instâncias de uma entidade, neste caso, um funcionário ou um departamento. As colunas representam os campos de dados que compõem as linhas. As relações entre tabelas ocorrem quando uma coluna em uma tabela também aparece como uma coluna em outra.

Aqui está um exemplo de duas tabelas relacionadas:

Table

Description automatically generated

Table

Description automatically generated

Usando essas tabelas, você pode responder às perguntas abaixo?

Qual é o nome do departamento do funcionário 1004? [Responder]

Customer support

Quem é o gerente do funcionário 1004

Jane Trader

1: Teradata SQL > What is Sql?

A Linguagem de Consulta Estruturada (SQL) é a linguagem padrão da indústria para comunicação com Sistemas de Gerenciamento de Banco de Dados Relacional. O SQL é usado para pesquisar dados em tabelas relacionais, criar tabelas, inserir dados em tabelas, conceder permissões e muitas outras coisas.

A linguagem de consulta estruturada é usada para definir o conjunto de respostas que é retornado do banco de dados Teradata.

SQL é uma linguagem não procedural, o que significa que não contém instruções do tipo procedural como estas:

Go To -VÁ PARA

Perform – EXECUTAR

Do Loop - DO LOOP

Open File - ABRIR ARQUIVO

Close File - FECHAR ARQUIVO

End of File - FIM DO ARQUIVO

As instruções SQL são geralmente divididas em três categorias:

Data Definition Language (DDL) - Usado para definir e criar objetos de banco de dados, como tabelas, exibições, macros, bancos de dados e usuários.

Data Manipulation Language (DML) - Usado para trabalhar com os dados, incluindo tarefas como inserir linhas de dados em uma tabela, atualizar uma linha existente ou realizar consultas nos dados. O ponto focal deste curso será as instruções SQL nesta categoria.

Data Control Language (DCL) - Usado para tarefas administrativas, como conceder e revogar privilégios a objetos de banco de dados ou controlar a propriedade desses objetos.

Data Definition Language (DDL) - Usado para definir e criar objetos de banco de dados, como tabelas, exibições, macros, bancos de dados e usuários.

|  |  |
| --- | --- |
| Data Definition Language (DDL) | |
| SQL statement | Function |
| CREATE | Define a table, view, macro, index, trigger or stored procedure. |
| DROP | Remove a table, view, macro, index, trigger or stored procedure. |
| ALTER | Change table structure or protection definition. |

Data Manipulation Language (DML) - Usado para trabalhar com os dados, incluindo tarefas como inserir linhas de dados em uma tabela, atualizar uma linha existente ou realizar consultas nos dados. O ponto focal deste curso será as instruções SQL nesta categoria.

|  |  |
| --- | --- |
| Data Manipulation Language (DML) | |
| SQL statement | Function |
| SELECT | Select data from one or more tables. |
| INSERT | Place a new row into a table. |
| UPDATE | Change data values in one or more existing rows. |
| DELETE | Remove one or more rows from a table. |

Data Control Language (DCL) - Usado para tarefas administrativas, como conceder e revogar privilégios a objetos de banco de dados ou controlar a propriedade desses objetos. As instruções DCL não serão abordadas em detalhes neste curso. Consulte Referências na página de Ajuda para obter mais informações.

|  |  |
| --- | --- |
| Data Control Language (DCL) | |
| SQL statement | Function |
| GRANT | Give user privileges. |
| REVOKE | Remove user privileges. |
| GIVE | Transfer database ownership. |

A figura abaixo ilustra seis linhas de uma tabela muito maior, Employees.

Clique nas áreas nos quadrados vermelhos para ver alguns conceitos e termos comumente usados ​​com bancos de dados relacionais. Clique novamente em um quadrado vermelho para ocultar sua descrição.

Neste curso, vamos nos concentrar principalmente em DML. Veremos primeiro a instrução SELECT.

A instrução SELECT permite recuperar dados de uma ou mais tabelas. Em sua forma mais comum, você especifica certas linhas a serem retornadas conforme mostrado.

Select \*

From employee

Where department\_number = 401;

O asterisco, "\*", indica que desejamos ver todas as colunas da tabela.

A cláusula FROM especifica de qual tabela em nosso banco de dados recuperar as linhas.

A cláusula WHERE atua como um filtro que retorna apenas as linhas que atendem à condição especificada, neste caso, registros de funcionários do departamento 401.

E se não tivéssemos especificado uma cláusula WHERE?

Select \* From empregado;

Essa consulta retornaria todas as colunas e todas as linhas da tabela de funcionários.

Projeção significa escolher quais colunas (ou expressões) a consulta retornará.

Em vez de usar o símbolo de asterisco para especificar todas as colunas, poderíamos nomear colunas específicas separadas por vírgulas:

SELECT funcionário\_número

,data de contratação

,sobrenome

,primeiro nome

From empregado

WHERE número\_departamento = 401;

Os resultados voltam sem classificação, a menos que você especifique que deseja que eles sejam classificados de uma determinada maneira. Como recuperar resultados ordenados é abordado na próxima seção.

|  |  |  |  |
| --- | --- | --- | --- |
| employee\_number | hire\_date | last\_name | first\_name |
| 1004 | 2006-10-15 | Johnson | Darlene |
| 1003 | 2006-07-31 | Trader | Jane |
| 1013 | 2007-04-01 | Phillips | Charles |
| 1010 | 2007-03-01 | Rogers | Frank |
| 1022 | 2009-03-01 | Machado | Albert |
| 1001 | 2006-06-18 | Hoover | Wilma |
| 1002 | 2006-07-31 | Brown | Alan |

Use a cláusula ORDER BY para que seus resultados sejam exibidos em uma ordem de classificação. Sem a cláusula ORDER BY, as linhas de saída resultantes são exibidas em uma sequência aleatória.

SELECT funcionário\_número

,sobrenome

,primeiro nome

,data de contratação

FROM funcionário

WHERE número\_departamento = 401

ORDER BY data\_contratação;

Ordem de classificação

No exemplo acima, os resultados serão retornados em ordem crescente por data de contratação. A ordem crescente é a sequência de classificação padrão para uma cláusula ORDER BY. Para especificar explicitamente a ordem crescente ou decrescente, adicione ASC ou DESC ao final da cláusula ORDER BY, assim:

ORDER BY Hire\_date DESC;

Nomeando a coluna de classificação

Você pode indicar a coluna de ordenação nomeando-a diretamente (por exemplo, rental\_date) ou especificando sua posição na instrução SELECT. Como a data\_contratação é a quarta coluna na instrução SELECT, a cláusula ORDER BY a seguir é equivalente a dizer a data\_contratada ORDER BY.

ORDEM POR 4;

Uma cláusula ORDER BY pode especificar várias colunas. Nenhuma coluna única em uma cláusula ORDER BY deve exceder um comprimento de 4096 bytes, caso contrário, ela será truncada para fins de classificação.

A ordem na qual as colunas são listadas na cláusula ORDER BY é significativa. A coluna nomeada primeiro é a coluna de classificação principal. A segunda e as subsequentes são colunas de classificação secundária. No exemplo a seguir, os resultados são classificados por número de departamento em ordem crescente. Onde vários registros compartilham o mesmo número de departamento, essas linhas são classificadas por job\_code em ordem crescente. Aqui está um exemplo:

SELECT funcionário\_número

,department\_number

,Código de Trabalho

FROM funcionário

WHERE número\_departamento < 302

ORDER BY departamento\_número

,Código de Trabalho;

|  |  |  |
| --- | --- | --- |
| employee\_number | department\_number | job\_code |
| 801 | 100 | 111100 |
| 1025 | 201 | 211100 |
| 1021 | 201 | 222101 |
| 1019 | 301 | 311100 |
| 1006 | 301 | 312101 |
| 1008 | 301 | 312102 |

Cada coluna especificada na cláusula ORDER BY pode ter sua própria ordem de classificação, ascendente (ASC) ou descendente (DESC).

SELECT funcionário\_número

,department\_number

,Código de Trabalho

FROM funcionário

WHERE número\_departamento < 302

ORDER BY departamento\_número ASC

,job\_code DESC;

|  |  |  |  |
| --- | --- | --- | --- |
| employee\_number | department\_number | job\_code | |
| 801 | 100 | 111100 | |
| 1021 | 201 | 211101 | |
| 1025 | 201 | 211100 | |
| 1008 | 301 | 312102 | |
| 1006 | 301 | 312101 | |
| 1019 | 301 | 311100 | |
| Os funcionários 1025 e 1021 compartilham o departamento 201. Eles são classificados por código de cargo decrescente dentro desse departamento.  o operador DISTINCT consolidará linhas de saída duplicadas em uma única ocorrência. Por exemplo, duas pessoas no departamento 501 têm o mesmo código de trabalho (512101). Se nosso propósito é simplesmente descobrir quais códigos de trabalho existem no departamento 501, podemos usar DISTINCT para evitar a exibição de linhas duplicadas.  DISTINCT aparece diretamente após a palavra-chave SELECT e antes da primeira coluna nomeada. Pode parecer que se aplica apenas à primeira coluna, mas, na verdade, DISTINCT se aplica a todas as colunas nomeadas na consulta. Duas linhas em nosso conjunto de exemplos de resultados têm número\_de\_departamento 501. A combinação de número\_de\_departamento e código\_de\_trabalho são distintas, pois os códigos de trabalho são diferentes.  Exemplo Sem DISTINCT  SELECT departamento\_número  ,Código de Trabalho  FROM funcionário  WHERE número\_departamento = 501;  número\_departamento  Código de Trabalho  501  512101  501  512101  501  511100  Exemplo com DISTINCT  SELECT DISTINCT número\_departamento  ,Código de Trabalho  From funcionário  WHERE número\_departamento = 501;  número\_departamento  Código de Trabalho  501  512101  501  511100 | | |

**LECT    department\_number**

**,job\_code**

**FROM      employee**

**WHERE     department\_number = 501;**

|  |  |
| --- | --- |
| **department\_number** | **job\_code** |
| 501 | 512101 |
| 501 | 512101 |
| 501 | 511100 |

**Example With DISTINCT**

**SELECT    DISTINCT department\_number**

**,job\_code**

**FROM      employee**

**WHERE     department\_number = 501;**

|  |  |
| --- | --- |
| **department\_number** | **job\_code** |
| 501 | 512101 |
| 501 | 511100 |
|  |  |

Todos os objetos do banco de dados Teradata, como tabelas, devem receber um nome do usuário quando são criados.

As regras para nomear objetos podem ser resumidas da seguinte forma:

Os nomes são compostos por:

a-z

A-Z

0-9

\_ (sublinhado)

$

#

Os nomes são limitados a 30 caracteres no Teradata Database 14.00. Esse limite foi aumentado para 128 caracteres a partir do lançamento do Teradata Database 14.10.

Os nomes não podem começar com um número.

Os nomes não diferenciam maiúsculas de minúsculas.

Exemplos de nomes válidos

contas

Contas\_2005

contas\_acima\_$2.000

Conta#

Observe que Contas e contas representam a mesma tabela. Caso não é considerado.

Os nomes de banco de dados e os nomes de usuário devem ser exclusivos no banco de dados Teradata.

Os nomes de tabela, exibição, macro, gatilho, índice e procedimento armazenado devem ser exclusivos em um banco de dados ou usuário.

Os nomes das colunas devem ser exclusivos dentro de uma tabela.

A sintaxe para qualificar totalmente um nome de coluna é:

Databasename.Tablename.Columnname

Hire\_date

Employee.hire\_date

Payroll.employee.hire\_date

O SQL é considerado uma linguagem de 'forma livre'. Ou seja, pode abranger várias linhas de código e não há restrição quanto à quantidade de 'espaço em branco' (ou seja, espaços em branco, tabulações, quebra de linha) pode ser incorporado na consulta. No entanto, SQL é sintaticamente uma linguagem muito precisa. Vírgulas, pontos e parênteses mal colocados sempre gerarão um erro de sintaxe.

As convenções de codificação a seguir são recomendadas porque facilitam a leitura, criação e alteração de instruções SQL.

Prática recomendada

SELECT sobrenome

,primeiro nome

,data de contratação

,valor\_salário

FROM funcionário

WHERE número\_departamento = 401

ORDER BY sobrenome;

Prática Não Recomendada

selecione sobrenome, nome\_primeiro, data\_contratação, valor\_salário

do funcionário onde departamento\_número = 401 ordem por último\_nome;

O primeiro exemplo é fácil de ler e solucionar problemas (se necessário). O segundo exemplo parece ser uma confusão de palavras. Ambos, no entanto, são instruções SQL válidas.

Como um usuário válido, você normalmente terá direitos de acesso ao seu próprio banco de dados de usuários e aos objetos que ele contém. Você também pode ter permissão para acessar objetos em outros bancos de dados.

O nome de usuário com o qual você faz logon geralmente é seu banco de dados padrão. (Isso depende de como você foi criado como usuário.)

Por exemplo, se esta for sua caixa de diálogo de logon:

Então td0001 seria seu banco de dados padrão.

As consultas feitas que não especificam o nome do banco de dados serão feitas no banco de dados padrão.

SELECT \*

FROM empregado; -- não há nome de banco de dados, então o sistema usa seu banco de dados padrão td0001

SELECT \*

FROM folha de pagamento.funcionário; -- o nome do banco de dados é fornecido, então o sistema usa o banco de dados da folha de pagamento

Se você deseja alterar seu banco de dados padrão durante uma sessão, use o comando DATABASE.

Exemplo

BANCO DE DADOS folha de pagamento;

Isso definirá seu banco de dados padrão para folha de pagamento. Consultas subseqüentes (supondo que os privilégios apropriados sejam mantidos) são feitas no banco de dados da folha de pagamento.

Se você deseja alterar seu banco de dados padrão para que seja sempre folha de pagamento toda vez que fizer login, use o comando MODIFY USER.

Exemplo

MODIFICAR USUÁRIO td0001 AS

BANCO DE DADOS PADRÃO = folha de pagamento;

Se você não configurou a conexão do servidor do laboratório, clique no botão Configuração do laboratório na parte inferior da página para obter instruções. Você precisará dessas instruções para fazer logon no banco de dados Teradata. Se você tiver problemas para se conectar ao servidor do laboratório, entre em contato com Training.Support@Teradata.com.

Para este conjunto de exercícios de laboratório, você pode precisar de informações do documento Database Info. Antes de fazer esses laboratórios, será útil definir seu banco de dados padrão para o banco de dados CustomerService (ou seja, DATABASE CustomerService;).

Clique no botão Avançar na parte inferior da página para ver as respostas.

1. Selecione todas as colunas para todos os departamentos da tabela de departamentos.

database CostumerService;

--SET SESSION DATABASE CustomerService;

-- Exec 1

SELECT \*

FROM CustomerService.department;

2. Usando a tabela de funcionários, gere um relatório com os sobrenomes e nomes dos funcionários e o salário de todos os funcionários do gerente 1019. Ordene o relatório em ordem crescente de sobrenome.

-- Exec 2

SELECT last\_name, first\_name, salary\_amount

FROM CustomerService.employee

WHERE manager\_employee\_number = 1019

ORDER BY last\_name;

3. Modifique a solicitação anterior para mostrar o número do departamento em vez do primeiro nome. Faça isso para os funcionários do gerente 801 em vez do gerente 1019.

-- Exec 3

SELECT last\_name, department\_number, salary\_amount

FROM CustomerService.employee

WHERE manager\_employee\_number = 801

ORDER BY last\_name;

4. Prepare um relatório dos números do departamento e dos números dos funcionários do gerente para todos na tabela de funcionários. Agora, como uma consulta separada, adicione a opção DISTINCT ao mesmo relatório.

-- Exec 4

SELECT department\_number, manager\_employee\_number

FROM CustomerService.employee;

SELECT Distinct department\_number, manager\_employee\_number

FROM CustomerService.employee;

MODULE 2: SIMPLE BTQE

Após a conclusão deste módulo, você deverá ser capaz de:

Use BTEQ para enviar SQL ao banco de dados Teradata.

Defina os parâmetros da sessão para habilitar:

Semântica de transação do Teradata.

O sinalizador SQL.

O que é BTEQ?

BTEQ é uma ferramenta de front-end para enviar consultas SQL. 'BTEQ' significa programa Basic Teradata Query.

BTEQ é um software cliente que reside em sua rede ou host conectado ao canal. Ele opera em todos os sistemas host e redes locais (LANs).

Depois de iniciar o BTEQ, você faria logon no banco de dados Teradata usando um TDP-id (Teradata Director Program id), seu ID de usuário e senha. O TDP-id identifica a instância do banco de dados Teradata que você vai acessar.

.LOGON tdpid/nome de usuário, logon

O TDP vem em duas variedades - o TDP padrão para clientes conectados ao canal e o Micro-TDP (ou MTDP) para clientes conectados à rede. O TDP está envolvido em fazer com que suas solicitações SQL sejam roteadas para um mecanismo de análise (PE), que valida sua solicitação e a passa para os processadores de módulo de acesso (AMPs). Os AMPs recuperam os conjuntos de respostas dos discos e enviam o conjunto de respostas de volta para o PE que, por sua vez, o encaminha para o TDP e, por fim, de volta para você em sua sessão.

Neste curso, não usaremos BTEQ para nossos laboratórios. Em vez disso, usaremos uma ferramenta GUI como Teradata Studio ou Teradata Studio Express. No entanto, é importante estar familiarizado com o BTEQ porque é a maneira mais simples de interagir com um banco de dados Teradata. Além disso, muitos clientes usam scripts BTEQ para realizar operações diárias em seus sistemas: coleta de estatísticas, tarefas ETL, etc.

Fatos sobre os comandos BTEQ:

Eles devem ser precedidos por um ponto (.) ou terminados por um ponto e vírgula ou ambos.

Exemplo: .LOGON

Eles fornecem uma listagem de saída (uma trilha de auditoria) do que ocorreu.

Existem comandos de formatação, ou seja, você pode fazer relatórios diretamente no BTEQ.

Eles não diferenciam maiúsculas de minúsculas.

Chamar o BTEQ varia um pouco dependendo da plataforma. Em muitos ambientes Linux®, basta digitar o comando 'bteq'.

Existem duas maneiras de enviar instruções SQL para BTEQ no modo interativo (em vez de scripts):

Digite instruções SQL diretamente no prompt BTEQ.

Abra outra janela com um editor de texto. Componha instruções SQL usando o editor de texto e, em seguida, recorte e cole o SQL no prompt BTEQ.

Parâmetro da Sessão: Semântica da Transação

Os parâmetros de sessão incluem a semântica da transação e o SQL Flagger.

A semântica da transação permite definir sua sessão para o modo ANSI ou Teradata (BTET). Todos os recursos do Teradata SQL funcionarão em qualquer modo, mas cada modo ativa diferentes padrões de conversão de dados e diferenciação de maiúsculas e minúsculas e, portanto, a mesma consulta pode retornar resultados diferentes em cada modo.

Para os fins deste curso, as sessões no modo Teradata serão assumidas, a menos que especificado de outra forma. Mais informações sobre esses parâmetros de sessão são abordadas em outro curso. Consulte Referências na página de Ajuda para obter mais informações.

Exemplo:

.SET SESSION TRANSACTION ANSI; /\* Sets ANSI mode \*/

.SET SESSION TRANSACTION BTET; /\* Sets Teradata mode \*/

Things to Notice:

All text between /\* and \*/ is treated as a comment by BTEQ.

The .SET SESSION command must be entered prior to logging on to the session.

Você também pode ativar o sinalizador ANSI, que sinaliza automaticamente a sintaxe não compatível com ANSI com um aviso, mas ainda retorna o conjunto de respostas esperado.

.SET SESSION SQLFLAC ENTRY; /\*Causes non-Entry level ANSI syntax to be flagged \*/

.LOGON…

SELECT DATE ; /\* Causes two warnings: 1. Keyword Date is not ANSI standard and 2, no FROM clause \*/

\*\*\*SQL warning 5821 Built-in values DATE and Time are not ANSI.

\*\*\*SQL warning 5804 a FROM clause is required in ANSI Query Specification.

\*\*\* QUERY completed. One row found. One column returned.

\*\*\* Total elapsed time was 1 second.

Date

15/08/15

.LOGOFF

.SET SESSION SQLFLAG NONE; /\*Disables ANSI Flagger\*/

.LOGGON …

SELECT DATE;

\*\*\*QUERY completed. One row found. One column returned.

\*\*\* Total elapsed time was 1 second.

Date

15/08/15

Things to Notice:

In both cases, the date was returned.

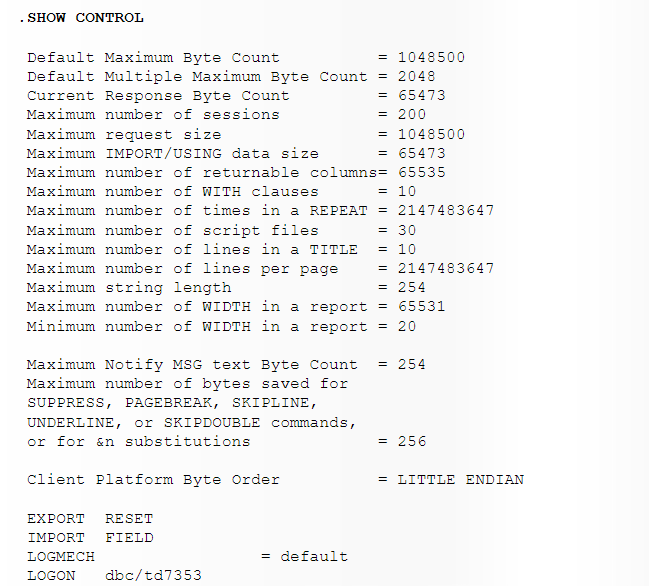
Teradata extensions cannot be disabled, but they can be flagged for warning using the ANSI flagger.

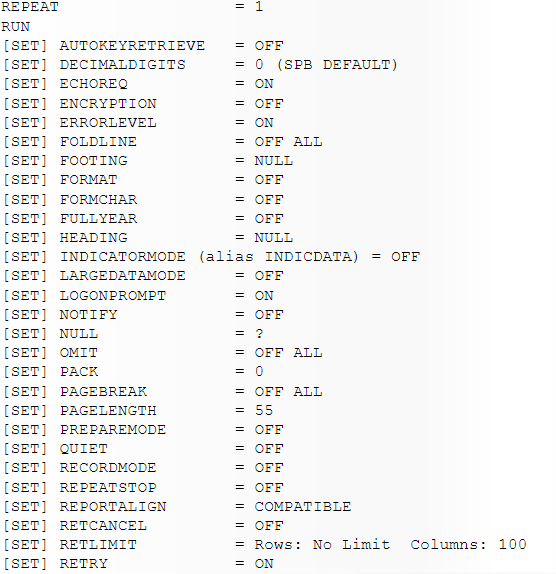
What syntax would ANSI prefer?

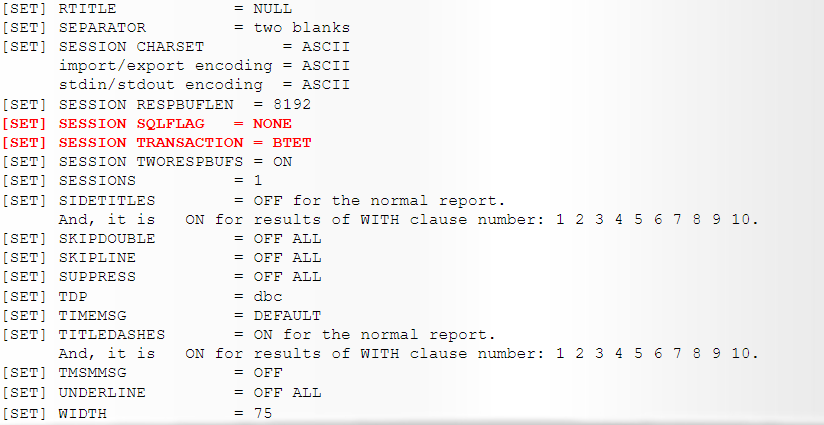
SELECT CURRENT\_DATE

BTEQ tem um comando SHOW, .SHOW CONTROL. Ele fornece informações sobre formatação e configurações de exibição para a sessão BTEQ atual.

Exemplo







Exemplo de Comentário BTEQ

/\* This is a BTEQ comment. The comment can span multiple lines. \*/

/\* Este é um comentário BTEQ. O comentário

pode abranger várias linhas. \*/

Exemplo de Comentário ANSI

-- Obtenha uma lista dos números e nomes dos departamentos

-- da mesa do departamento.

SELECT department\_number

,department\_name

FROM department;

Comentários estilo Teradata (/\* \*/) e estilo ANSI (--) podem ser incluídos em qualquer script de comando SQL.

SIMPLE BTEQ > BTEQ SCRIPTS

Um script BTEQ é um arquivo que contém comandos BTEQ e instruções SQL. Um script é construído para sequências de comandos que devem ser executados em mais de uma ocasião, ou seja, mensalmente, semanalmente, diariamente.

Como criar um script BTEQ

Para criar e editar um script BTEQ, use um editor em sua estação de trabalho cliente. Por exemplo, em uma estação de trabalho Linux®, use o vi ou o editor de texto.

Como enviar um script BTEQ

Inicie o BTEQ e insira o seguinte comando BTEQ para enviar um script BTEQ:

.RUN ARQUIVO = <scriptname>

Ou você pode enviar o script para o aplicativo a partir do prompt de comando:

bteq < <nome do script>

Veremos um exemplo na próxima página.

Vejamos um exemplo de script BTEQ. (Este script é denominado dept.btq )

.SET SESSION TRANSACTION ANSI

.LOGON wbtlabs.teradata.com/td0001,pwxxxx;

-- Obtain a list of the department numbers and names

-- from the department table.

SELECT department\_number

,department\_name

FROM department;

.QUIT

Para executar o script, você pode usar o comando .RUN dentro do BTEQ:

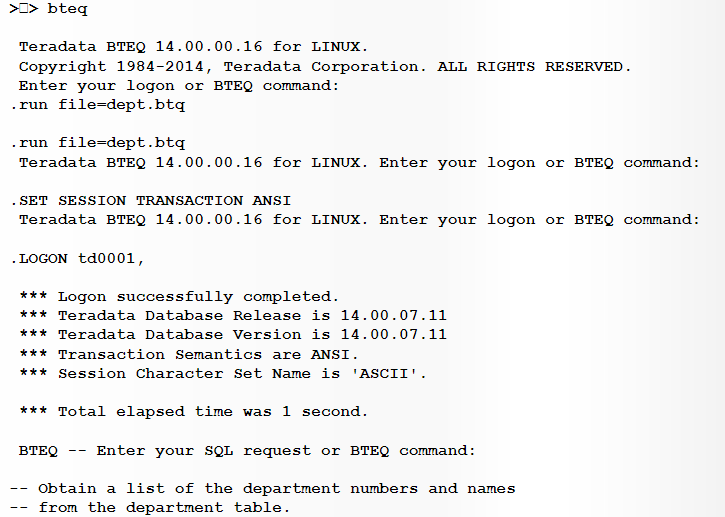
.RUN FILE=dept.btq

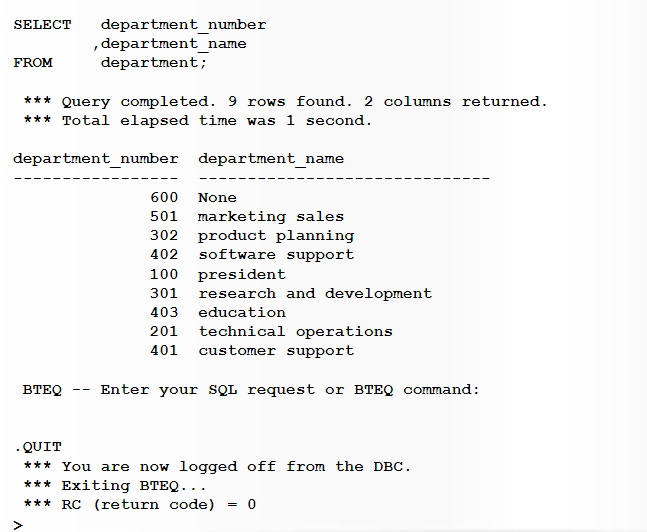
Ou você simplesmente chama o BTEQ na linha de comando (o < redireciona o script para o aplicativo BTEQ):

>> bteq < dept.btq

Veremos a saída na próxima página.

Aqui está a saída gerada pelo comando .RUN na página anterior:





Você pode querer capturar a saída de um script ou de uma sessão. Em uma estação de trabalho Linux®, simplesmente direcione a saída para um arquivo.

>> bteq < dept.btq > log.out

Você também pode usar o comando tee. Isso gravará a saída em um arquivo, mas também a mostrará a você na saída padrão.

>> bteq < dept.btq | tee log.out

Os resultados da saída BTEQ exibirão um "$" para indicar o local onde um erro foi detectado no script ou na solicitação.

Exemplo

BTEQ -- Insira sua solicitação SQL ou comando BTEQ:

SELECT departamento\_número

,DISTINCT nome\_do\_departamento

FROM department;

SELECT department\_number

,DISTINCT department\_name

FROM department;

,DISTINCT department\_name

$

\*\*\* Falha 3706 Erro de sintaxe: esperava algo entre ',' e a palavra-chave 'DISTINCT'.

O cifrão aponta para a palavra imediatamente após a palavra-chave DISTINCT. O texto do erro nos informa que não espera um DISTINCT nessa parte da consulta. O comando DISTINCT deve seguir diretamente o comando SELECT em sua solicitação.

Como não podemos acessar o servidor do laboratório com uma conexão SSH, não há laboratório BTEQ para este módulo. Mas você pode experimentar o BTEQ por conta própria.

Declaração nº 1, Informações = 46

\*\*\* O tempo total decorrido foi de 1 segundo.

Módulo 3: AJUDE, MOSTRAR, EXPLICAR

Após a conclusão deste módulo, você deverá ser capaz de:

Obtenha a definição de um banco de dados, tabela, exibição ou macro existente usando as instruções HELP e SHOW.

Determine como o Teradata RDBMS processará uma solicitação SQL usando a instrução EXPLAIN.

Vários comandos em nosso software são específicos do Teradata. Eles não são compatíveis com ANSI e não funcionarão necessariamente com outros sistemas de banco de dados.

Neste módulo, focamos nas seguintes funções:

EXPLICAR

AJUDA

MOSTRAR

O comando HELP é usado para exibir informações sobre qualquer objeto de banco de dados, como (mas não limitado a):

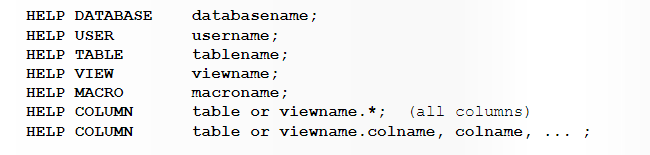
Bancos de dados e usuários

Tabelas

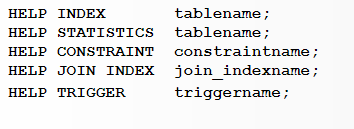
Visualizações

Macros

HELP recupera informações sobre esses objetos do Dicionário de Dados. Abaixo estão as opções sintáticas para várias formas do comando HELP:

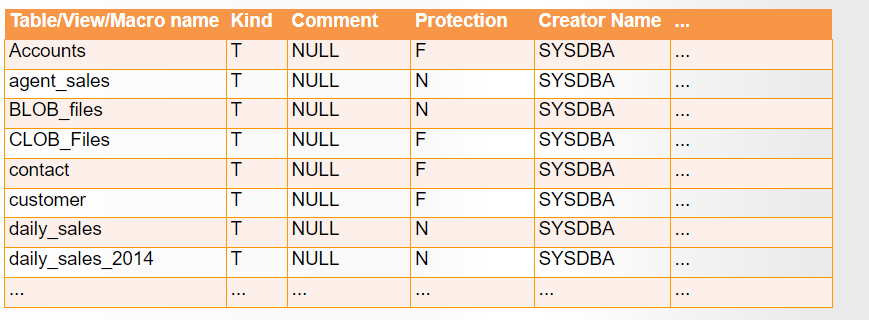


Alguns outros comandos HELP incluem:



O comando HELP DATABASE mostra todos os objetos no banco de dados ou usuário.

Exemplo

HELP DATABASE CustomerService;

Itens a serem observados:

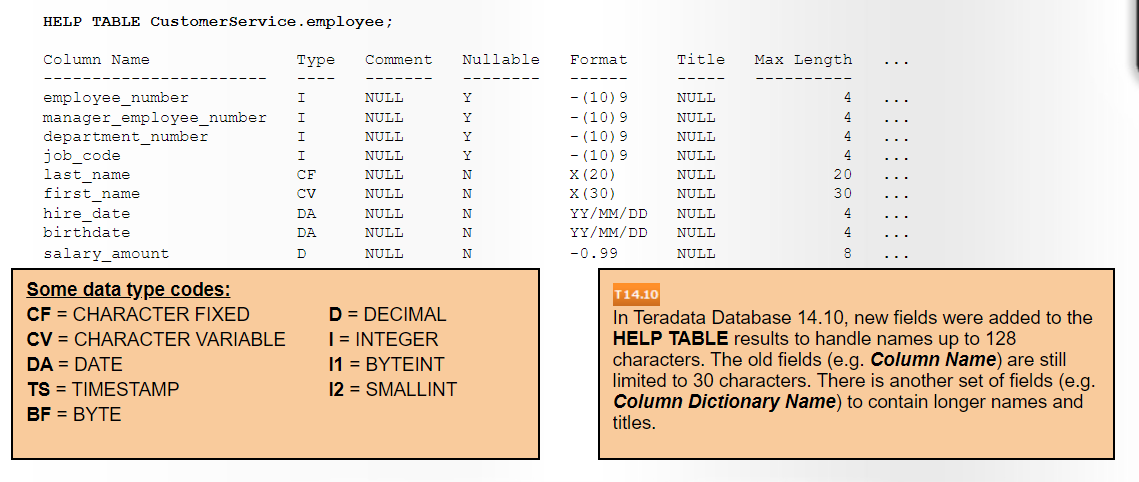
Os objetos podem ser reconhecidos por sua designação de tipo. Por exemplo, T = Table, V = View, M = Macro, I = Join Index, G = Trigger, S = Stored Procedure, etc. Todos os objetos neste banco de dados são tabelas.

Para retornar informações HELP sobre um objeto, um usuário deve possuir o objeto ou ter pelo menos um privilégio sobre ele.

O comando HELP TABLE mostra o nome, tipo de dados, comentário (se aplicável) e muitas outras informações para todas as colunas na tabela especificada. No exemplo abaixo, não vemos todas as colunas porque ficaria muito largo para esta tela.

Exemplo

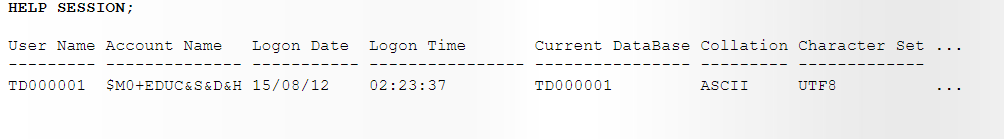
HELP TABLE CustomerService.employee;



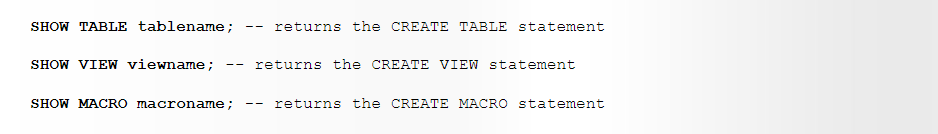
Use o comando HELP SESSION para ver informações específicas sobre sua sessão SQL atual. A saída são informações de sessão para incluir o nome do usuário, data e hora de logon, banco de dados padrão e outras informações relacionadas à sessão atual.

Abaixo está uma pequena parte dos resultados para um comando HELP SESSION. Observe que, com novos lançamentos de banco de dados, geralmente há campos adicionados.

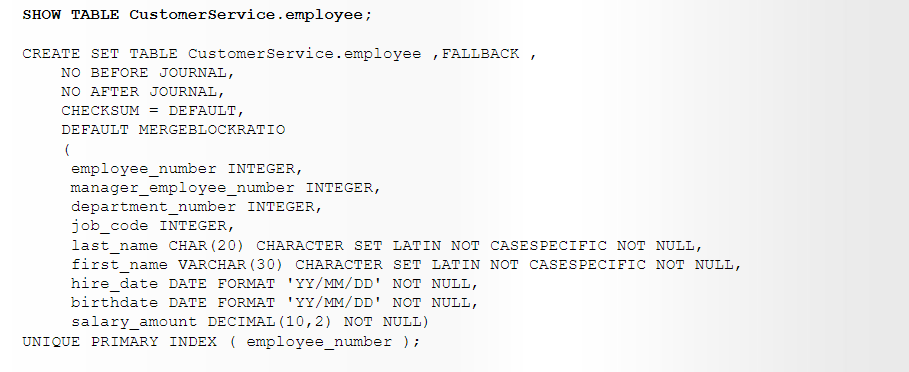
Exemplo



O comando SHOW exibe a linguagem de definição de dados (DDL) atual de um objeto de banco de dados (por exemplo, tabela, exibição, macro, gatilho, índice de junção, procedimento armazenado, etc.). O comando SHOW é usado principalmente para ver como um objeto foi criado.



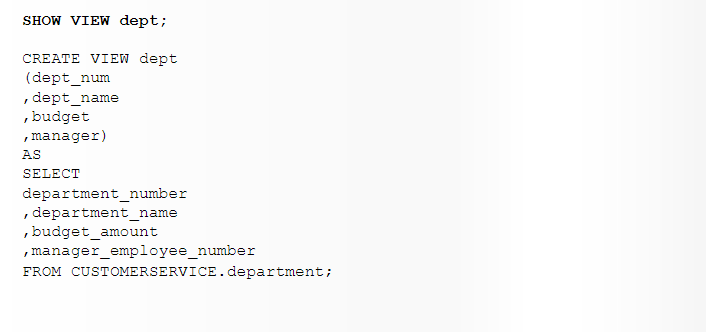
BTEQ também possui um comando SHOW, .SHOW CONTROL, que vimos no último módulo. É diferente do comando SQL SHOW. Ele fornece informações sobre formatação e configurações de exibição para a sessão BTEQ atual.

O comando SHOW TABLE visto aqui retorna a instrução CREATE TABLE que foi usada para criar a tabela de funcionários. Para executar um SHOW TABLE, o usuário deve ter privilégio na própria tabela ou no banco de dados que a contém.

O comando SHOW sempre refletirá a definição mais atual. Se uma tabela for CRIADA e depois ALTERada, o comando SHOW mostrará a definição com quaisquer alterações.

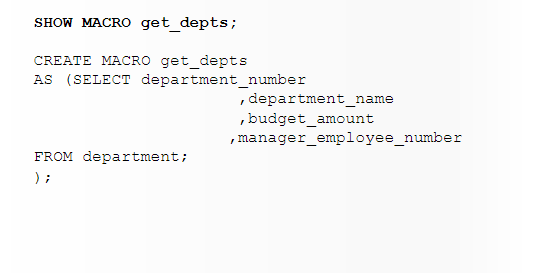
O comando SHOW VIEW mostra a instrução CREATE VIEW usada para criar uma exibição.

Exemplo



O comando SHOW MACRO mostra a instrução usada para criar uma macro.

Exemplo



A função EXPLAIN analisa uma solicitação SQL e responde em inglês como o otimizador planeja executá-la. Na verdade, ele não executa a instrução SQL. No entanto, é uma boa maneira de ver quais recursos do banco de dados serão usados ​​no processamento de sua solicitação.

Por exemplo, se você perceber que sua solicitação forçará uma varredura de tabela completa em uma tabela muito grande ou causará uma junção de produto cartesiano, poderá decidir reescrever uma solicitação para que seja executada com mais eficiência.

EXPLAIN fornece uma riqueza de informações, incluindo o seguinte:

Quais índices, se houver, serão usados ​​na consulta.

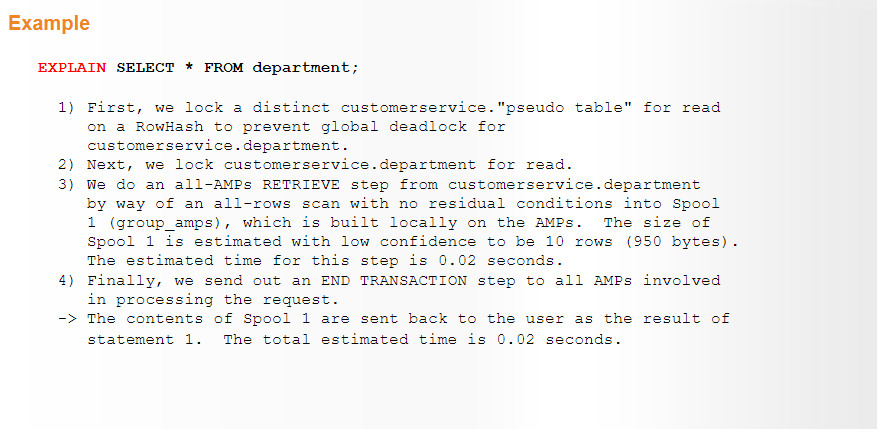
Se as etapas individuais dentro da consulta podem ser executadas simultaneamente (ou seja, etapas paralelas).

Uma estimativa do número de linhas que serão processadas.

Uma estimativa do custo da consulta (em incrementos de tempo).

O conteúdo do EXPLAIN mudará com o número de linhas na(s) tabela(s) e a configuração do sistema.

O recurso EXPLAIN é sua primeira ferramenta para ajustar uma consulta.



**Lab: Page 1**

Se você não configurou a conexão do servidor do laboratório, clique no botão Configuração do laboratório na parte inferior da página para obter instruções. Você precisará dessas instruções para fazer logon no banco de dados Teradata. Se você tiver problemas para se conectar ao servidor do laboratório, entre em contato com Training.Support@Teradata.com.

Para este conjunto de exercícios de laboratório, você pode precisar de informações do documento Database Info.

1.) Use o comando HELP DATABASE para localizar todas as tabelas, exibições e nomes de macros no banco de dados CSViews. Que tipo de objetos você encontra lá?

**HELP** **DATABASE** CSViews**;**

Execute um comando HELP semelhante no banco de dados CustomerService. Que tipo de objetos você encontra lá?

**Help** **Database** CustomerService**;**

**Lab: Page 2**

2.) Para ver os nomes das colunas na tabela de departamentos, use o comando AJUDA apropriado. (Como a tabela está no banco de dados CustomerService, não no seu banco de dados padrão, você pode ter que qualificar o nome da tabela com o nome do banco de dados.) Este é o comando que você pode querer usar no futuro para pesquisar nomes de dados.

**HELP** **TABLE** CustomerService.department**;**

3.) No Laboratório 1 da página anterior, você deve ter notado que o banco de dados CSViews inclui uma visão chamada emp. Show essa visão. Observe a lista de nomes curtos que a exibição emp usa no lugar dos nomes completos das colunas. Para economizar digitação, você pode usar a visualização emp com os nomes abreviados no lugar da tabela de funcionários em qualquer laboratório deste curso. (As soluções de laboratório usarão a tabela de funcionários e não a exibição.)

**SHOW VIEW CSViews.emp;**

**Lab: Page 3**

4.) Modifique o Laboratório 1 do primeiro módulo para fazer com que o conjunto de respostas apareça na sequência do nome do departamento. Para descobrir como o banco de dados Teradata planeja lidar com essa solicitação, envie-a com EXPLAIN na frente.

EXPLAIN

SELECT \*

FROM **CustomerService.department**

ORDER BY department\_name;

5.) Use o comando SHOW apropriado para ver a definição da tabela employee\_phone no banco de dados CustomerService.

**SHOW** **TABLE** CustomerService.employee\_phone**;**

**Lab: Page 4**

6.) Use o comando HELP apropriado para ver quais tipos de índices existem na tabela de clientes. Para obter informações sobre sua sessão, use outro comando HELP.

**HELP** **INDEX** CustomerService.customer**;**

**HELP** **SESSION;**

7.) Altere sua configuração de banco de dados atual para CustomerService usando o comando DATABASE. Tente fazer um SELECT de todas as colunas e linhas na visualização emp. Funciona? Se não, como você pode fazer isso funcionar?

**DATABASE** CustomerService**;**

**SELECT** \* **FROM** emp**;**

*-- Dá erro object não existe*

*--Comando que funciona*

**SELECT** \* **FROM** CSViews.emp**;**

MODULO 4: Data Types and Conversions

Após a conclusão deste módulo, você deverá ser capaz de:

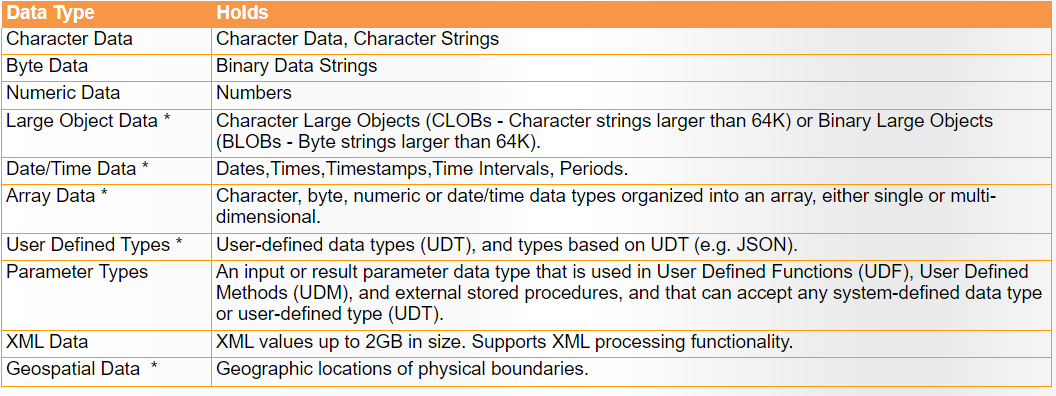
Defina o tipo de dados para uma coluna em uma tabela.

Calcule valores usando funções aritméticas e operadores.

Converter dados de um tipo para outro.

Manipule o tipo de dados DATE.

Cada coluna em uma linha está associada a um tipo de dados que define o tipo de valor que ela aceita. Esses tipos de dados são associados aos campos quando a tabela é criada. Os tipos de dados se enquadram em uma das categorias mostradas abaixo:



Os tipos listados com um asterisco (\*) são abordados com mais detalhes em outro curso. Consulte Referências na página de Ajuda para obter mais informações.

Existem dois tipos de dados para armazenar dados de caracteres:

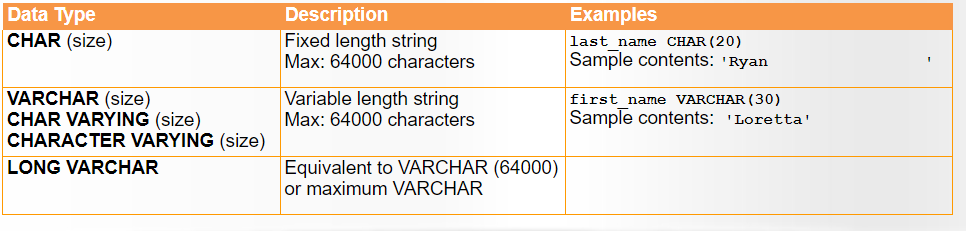
CHAR - tem cadeias de caracteres de comprimento fixo.

VARCHAR - possui cadeias de caracteres de comprimento variável.

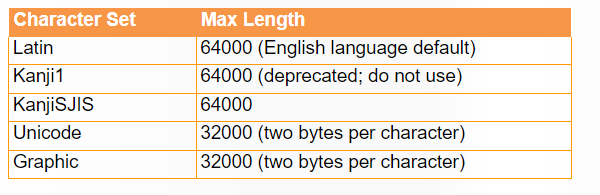
Em ambos os casos, o comprimento máximo da string é de 64.000 caracteres. O tamanho da string é especificado entre parênteses após a palavra-chave CHAR ou VARCHAR como em CHAR(20) ou VARCHAR(20). Existem diferenças sutis entre os dois.

No caso de CHAR, se você inserir uma cadeia de caracteres menor que o tamanho especificado, o sistema acrescentará e armazenará espaços em branco à direita conforme necessário para expandir a cadeia para o comprimento especificado. O tamanho especificado em um campo VARCHAR representa um comprimento máximo. O sistema não preencherá o conteúdo com espaços em branco à direita. No entanto, ele usará dois bytes extras para armazenar o comprimento internamente.

Quando você sabe que os dados sempre terão um comprimento específico (por exemplo, uma parada de correio com quatro caracteres), use CHAR, porque você salva os dois bytes que o VARCHAR usaria para armazenar o comprimento. Quando os dados (especialmente strings mais longas) não têm um comprimento fixo (por exemplo, sobrenomes), VARCHAR faz uso mais eficiente do espaço, porque não preenche valores com espaços em branco à direita.



As cadeias de caracteres podem ser definidas usando qualquer um dos seguintes conjuntos de caracteres:



Exemplos:

first\_name VARCHAR(30) CHARACTER SET LATIN

image\_abc CHAR(32000) CHARACTER SET GRAFIC

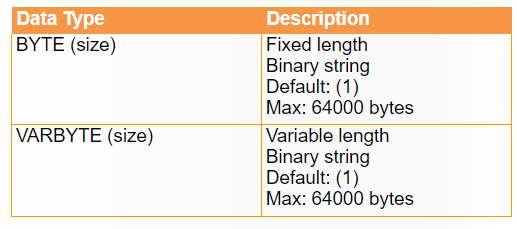
O banco de dados Teradata oferece suporte a dois tipos de dados para armazenar dados binários:

BYTE para strings binárias de comprimento fixo.

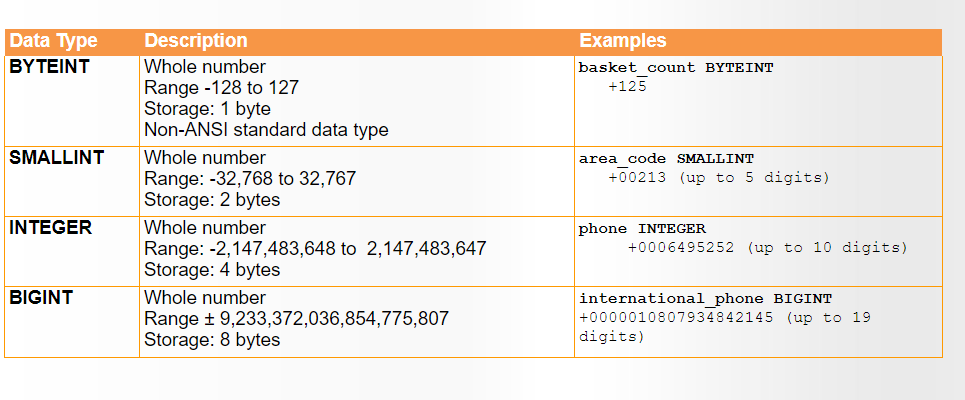
VARBYTE para strings binárias de comprimento variável.

Esses tipos de dados são usados ​​para armazenar objetos binários, como imagens digitais, objetos executáveis, arquivos simples, etc.

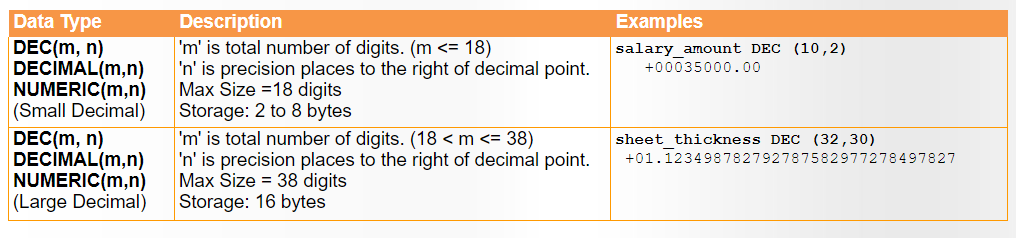
Eles não são conversíveis em outros tipos de dados.



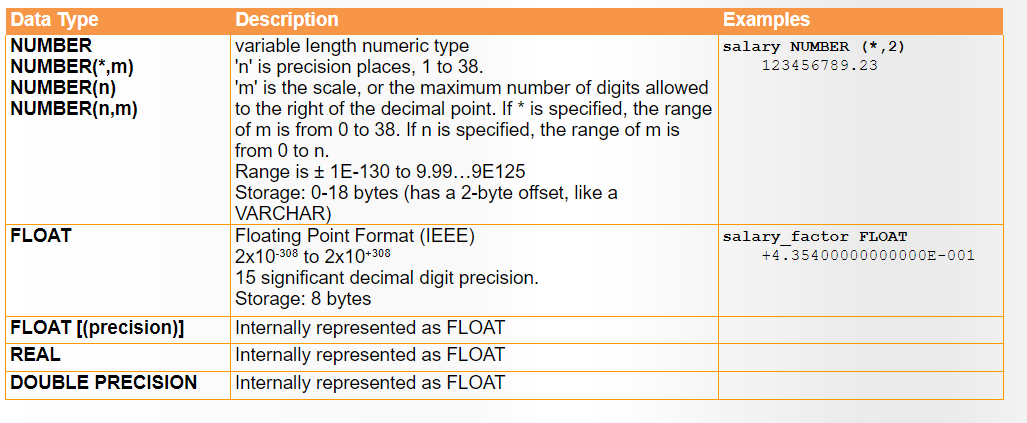
Os seguintes tipos de dados numéricos inteiros estão disponíveis:



Os seguintes tipos de dados decimais (ou numéricos) estão disponíveis



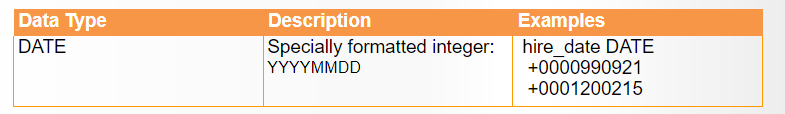
Aqui estão alguns outros tipos de dados numéricos disponíveis:



DATE representa uma data do calendário. Ele simplifica o manuseio e a formatação de datas comuns a aplicativos de negócios. DATE é armazenado como um número inteiro, usando a seguinte fórmula:

(ano - 1900) \* 10000 + (mês \* 100) + dia

O gráfico a seguir mostra como o tipo de dados DATE é armazenado internamente como um número inteiro pelo banco de dados Teradata.

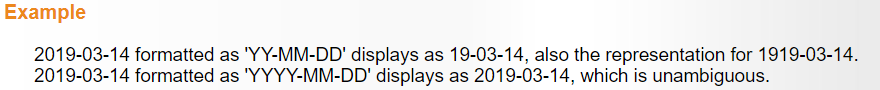


No primeiro exemplo, a data representada é 21 de setembro de 1999.

No segundo exemplo, a data representada é 15 de fevereiro de 2020.

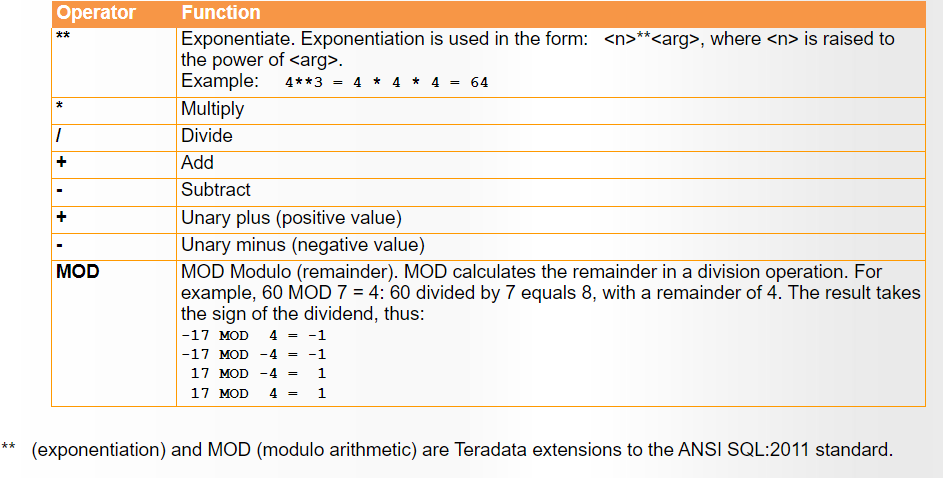
Observe que o ano é representado como um deslocamento do ano 1900 em ambos os casos. É assim que a data é armazenada internamente. A forma como você vê a data em um relatório depende de como a data é formatada.

O formato de exibição recomendado para valores de DATA é ‘AAAA-MM-DD’. Este formato evitará problemas que podem ocorrer em formatos que especificam apenas dois dígitos para o ano



Observação: as técnicas de formatação de datas são discutidas em outros cursos. Consulte Referências na página de Ajuda para obter informações

O banco de dados Teradata fornece 8 operadores aritméticos:



O banco de dados Teradata oferece suporte a várias funções aritméticas que fornecem recursos adicionais além daqueles disponíveis em ANSI SQL. Aqui estão alguns (onde arg é qualquer argumento numérico constante ou variável):

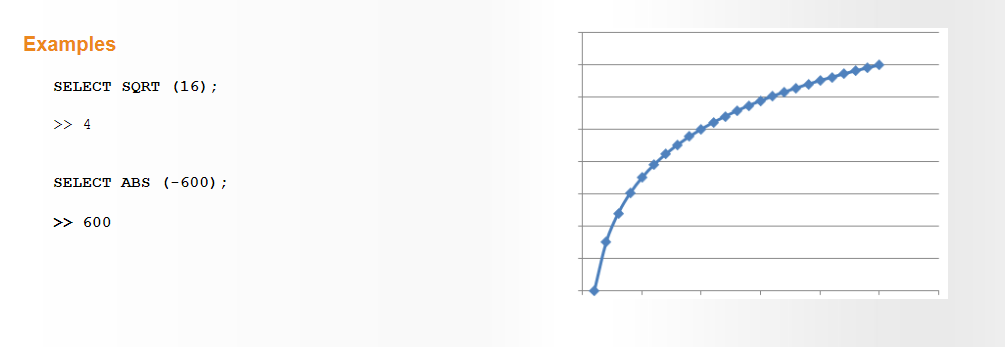
ABS (arg) - Valor absoluto. É equivalente a um arg positivo, independentemente de o arg real ser positivo ou negativo.

EXP (arg) - Eleva e à potência de arg (e \*\* arg)

LOG (arg) - Logaritmo de base 10

LN (arg) - Base e logaritmo (natural)

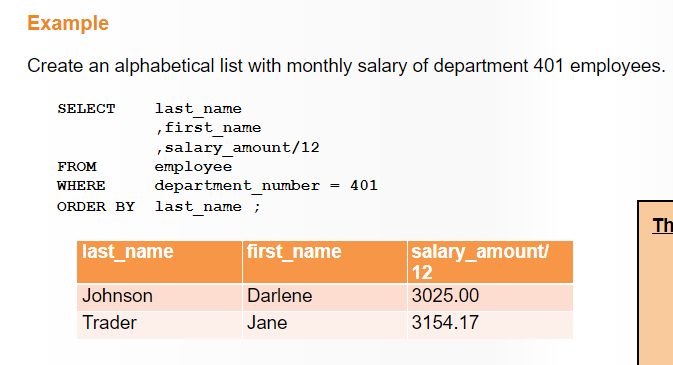
SQRT (arg) - Raiz quadrada



Você pode usar expressões aritméticas em seu SELECT para realizar cálculos nos dados da tabela. As expressões aritméticas permitirão que você faça duas coisas:

Re-rotule o cabeçalho da coluna para fins de exibição da saída.

Execute o cálculo nos valores de dados nessa coluna.



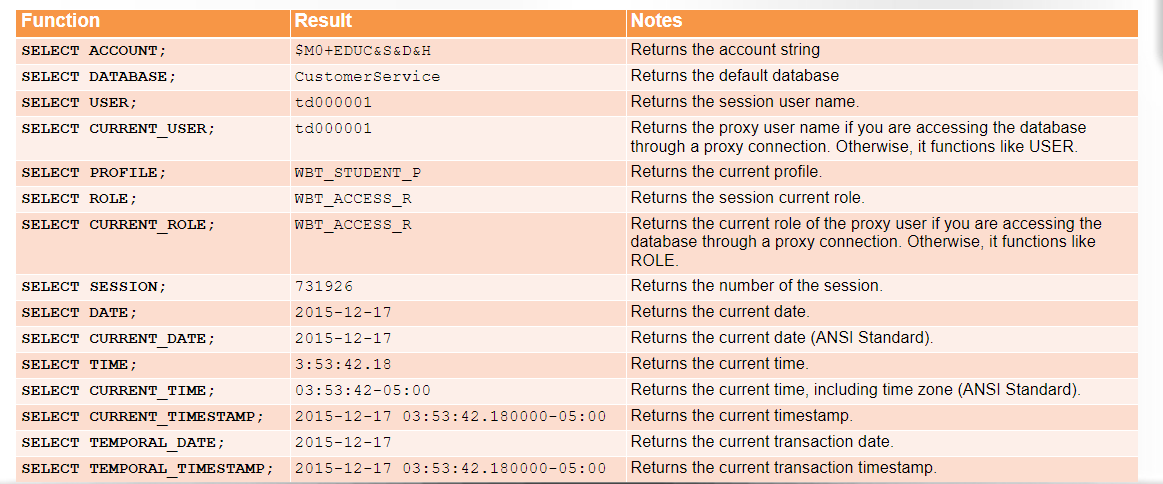
Itens a serem observados:

salário\_amount/12 é considerado uma 'coluna computada', porque uma operação aritmética é executada no valor subjacente.

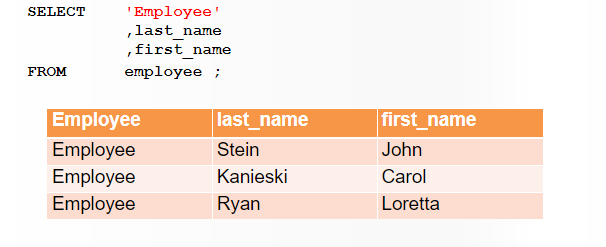
O título da coluna reflete o cálculo realizado.

O banco de dados Teradata possui várias funções integradas. Essas funções não possuem argumentos e retornam várias informações sobre o sistema. As funções incorporadas às vezes são chamadas de registradores especiais.

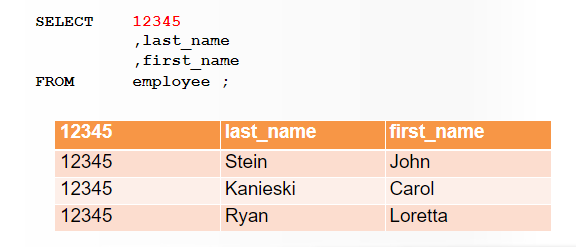
As funções internas podem ser usadas em qualquer lugar onde um literal possa aparecer. Se sua instrução SELECT fizer referência a um nome de tabela e a uma função integrada, o resultado da consulta conterá uma linha para cada linha da tabela que satisfaça a condição de pesquisa.



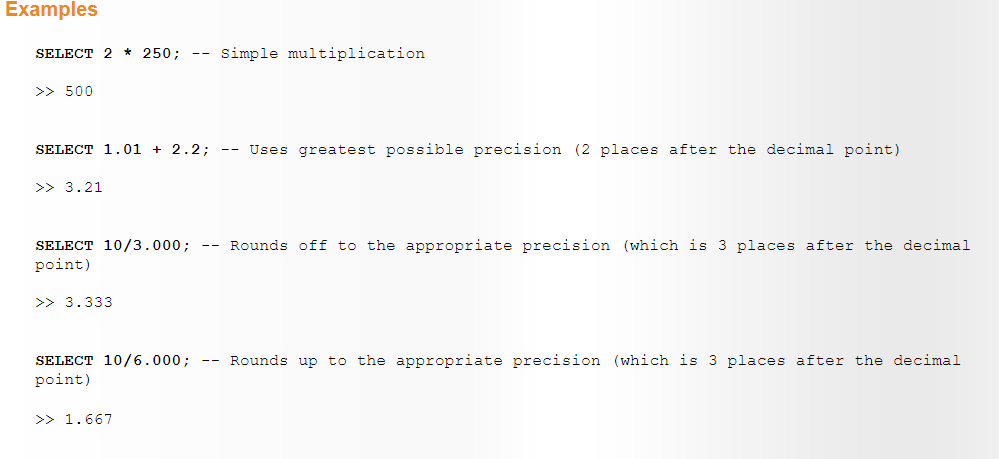
Você pode adicionar literais de caracteres à sua instrução SELECT. Você usa aspas simples para a string:



Você também pode adicionar constantes numéricas à sua instrução SELECT:



Você pode usar expressões numéricas para fazer cálculos.



O módulo, ou operador de resto, é uma extensão do Teradata que calcula o resto em uma operação de divisão.

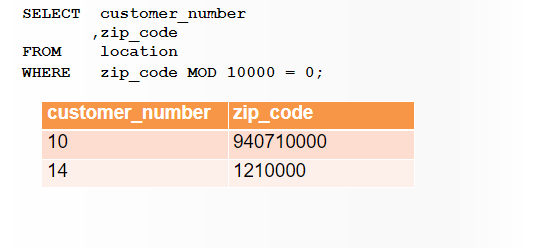
Como funciona? Ele divide os números e pega o restante. Por exemplo, 7/4 = 1 com resto 3. Portanto,

7 MOD 4 = 3

Exemplo

Identifique os códigos postais na tabela de localização que termina em quatro zeros. Você poderia fazer essa pesquisa com comparações de strings, mas é mais eficiente usar MOD porque nenhuma conversão de caractere é necessária.

Observe que os zeros à esquerda são suprimidos por padrão.



O tipo de dados Teradata DATE é usado para armazenar datas do calendário representando ano, mês e dia. Ele é armazenado internamente como um inteiro de quatro bytes. O banco de dados Teradata executa funções básicas de manipulação de DATE, incluindo:

Conversões mês a mês.

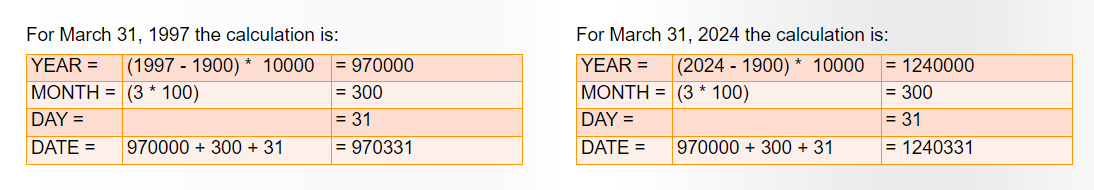
Conversões ano a ano.

Conversões de anos bissextos.

Datas anteriores a 1900.

O banco de dados trata o tipo de dados DATE como um valor INTEGER, mas não permite datas de calendário inválidas. O sistema codifica datas usando a fórmula:

(((YEAR - 1900) \* 10000 ) + MONTH \* 100 ) + DAY



Para 1º de janeiro de 1900, o resultado seria 000101.

Para datas anteriores a 1º de janeiro de 1900, o cálculo de número inteiro retorna um número negativo. Lembre-se, é assim que a data é armazenada, não como é representada em uma saída de consulta.

O SQL nos permite realizar cálculos aritméticos em DATE. Quando a palavra-chave CURRENT\_DATE é selecionada, ela é uma variável do sistema que representa a data atual. (DATA também pode ser usado).

Exemplos

Encontre a data 30 dias a partir de hoje.

SELECT CURRENT\_DATE + 30;

Como dias é a unidade básica, precisamos expressar um ano em termos do número de dias que ele contém. CURRENT\_DATE + 365 (ou em um ano bissexto, CURRENT\_DATE + 366), nos dá a data daqui a um ano.

Encontre a data daqui a 1 ano.

SELECT CURRENT\_DATE + 365;

Como você encontraria a idade de uma pessoa (em anos arredondados)?

SELECT (DATE-birthdate)/365

FROM employee

WHERE last\_name = ‘Stein’;

>> 61

Este exemplo pega a data de hoje (codificada como um número inteiro) e "subtrai" a data de nascimento de John Stein (também codificada como um número inteiro) e divide pelo número de dias em um ano (365).

Suponha que a data de hoje seja 1150817 e a data de nascimento de John seja 531015. Se você fizer isso como uma subtração inteira, descobrirá que 980826 - 531015 = 619802 - não é uma data legal. Mas como esses são tipos de dados DATE, não INTEGERS, o operador de subtração realmente retorna o número de dias entre os dois operandos em vez de subtrair os INTEGERs.

(DATA - data de nascimento) nos fornece o número de dias decorridos entre uma data de nascimento e hoje. Dividindo esse valor pelo número de dias em um ano (365) nos dá a idade de John em anos. Para contabilizar o ano bissexto, devemos dividir por 365,25 para obter um número mais preciso de anos.

Você também pode incluir operações usando DATE na cláusula WHERE, como no exemplo abaixo.

Exemplo

Encontre funcionários com mais de 30 anos de serviço.

SELECT last\_name

,hire\_date

FROM employee

WHERE (DATE-hire\_date)/365 > 30;

A função ADD\_MONTHS permite a adição de um número especificado de meses a uma data existente, resultando em uma nova data. ADD\_MONTHS contabiliza o calendário gregoriano e sabe quantos dias tem cada mês, incluindo anos bissextos. Devido ao número variável de dias em um mês, ADD\_MONTHS fornece um grau muito maior de precisão ao projetar datas com base em incrementos de mês.

O formato da função é:

ADD\_MONTHS (date, n)

onde n é um número ou uma expressão que representa o número de meses a serem adicionados à data.

Exemplos

SELECT CURRENT\_DATE

,ADD\_MONTHS(CURRENT\_DATE, 2)

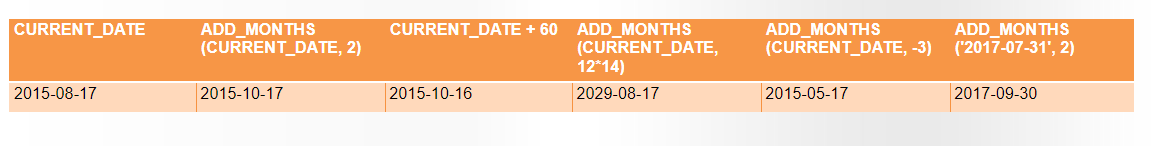
,CURRENT\_DATE + 60

,ADD\_MONTHS (CURRENT\_DATE, 12\*14)

--uma expressão

, ADD\_MONTHS (CURRENT\_DATE, -3) – numero negativo e subtraido de meses

,ADD\_MONTHS (‘2017-07-31’, 2) –usa a data literal



O banco de dados Teradata oferece suporte a dois tipos de dados para armazenar dados relacionados ao tempo:

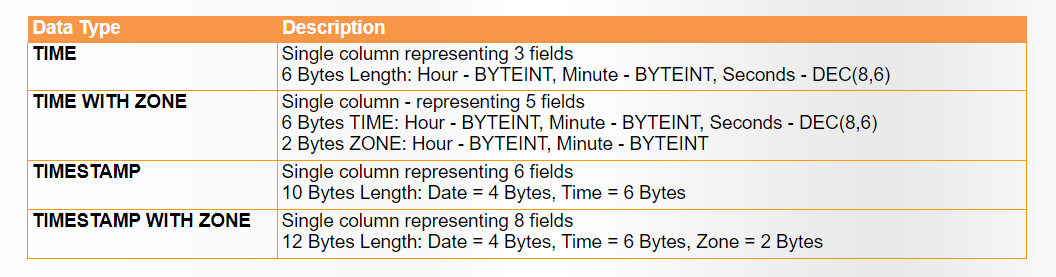
HORÁRIO (COM ZONA)

TIMESTAMP (COM ZONA)

O tempo é representado como um tipo de dados TIME que, na realidade, carrega três campos diferentes de informação: horas, minutos e segundos. Ele também possui 'inteligência de relógio' em sua implementação, assim como DATE possui inteligência de calendário, possibilitando assim cálculos complexos.

TIMESTAMP é um tipo de dados que combina DATE e TIME em um único tipo de dados.

Ambos TIME e TIMESTAMP têm uma opção WITH ZONE, que permite o armazenamento dependente do fuso horário e o processamento de informações relacionadas ao horário.



Os tipos de dados de intervalo representam um deslocamento entre dois pontos no tempo. Os intervalos são armazenados internamente como um ou vários campos numéricos combinados em um único tipo de dados.

Existem duas categorias gerais de tipos de dados INTERVAL:

\*\* YEAR

\*\* YEAR TO MONTH

\*\* MONTH

Intervalos diurnos

\*\*DAY

\*\*DAY TO HOUR

\*\*DAY TO MINUTE

\*\*DAY TO SECOND

\*\*HOUR

\*\*HOUR TO MINUTE

\*\*HOUR TO SECOND

\*\*MINUTE

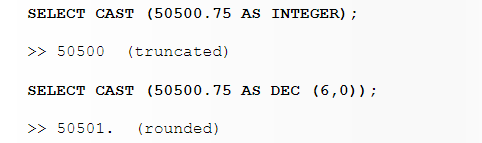
\*\*MINUTE TO SECOND

\*\*SECOND

Esses tipos de dados são abordados com mais detalhes em outro curso. Consulte Referências na página de Ajuda para obter mais informações.

A função CAST permite converter um valor ou expressão de um tipo de dados para outro.

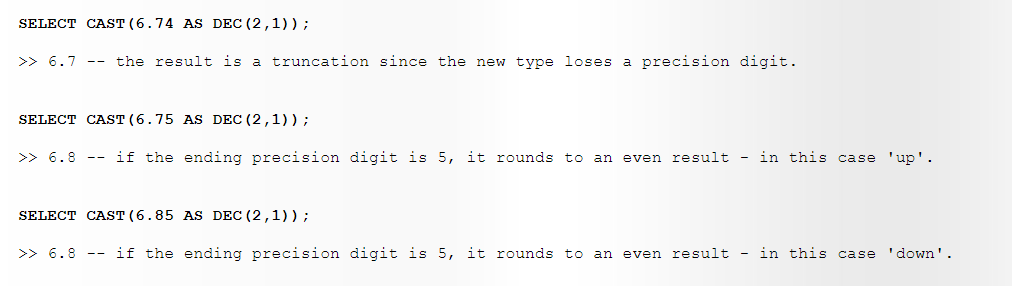
Exemplos



Ao converter de um decimal para um inteiro, o sistema descarta tudo à direita do ponto decimal. Em nosso exemplo, embora a parte decimal seja 0,75, tornando o número mais próximo de 50501 do que de 50500, o resultado da conversão ainda é 50500.

CASTing de um decimal em um nível de precisão para um decimal em um nível de precisão menor faz com que o sistema arredonde o resultado para o valor mais próximo exibido no novo nível de precisão.

Exemplos



O arredondamento é baseado no modo 'Arredondar para o mais próximo', seguindo o processo abaixo:

Deixe B representar o resultado real (por exemplo, B = 8,35).

Deixe A e C representar os valores de colchetes mais próximos que podem ser representados, de modo que A < B < C. (por exemplo, A = 8,3 e C = 8,4)

A determinação se A ou C é o resultado representado é feita da seguinte forma:

Quando possível, o resultado é o valor mais próximo de B.

Se A e C são equidistantes (como quando a parte fracionária é exatamente 0,5), o resultado é o número par. (por exemplo, 8.4)

Como você pode ver no exemplo abaixo, CAST pode pegar o campo de dados CHAR(20) last\_name da tabela de funcionários e convertê-lo em um tipo de dados CHAR(5). Os valores exibidos como resultado dessa consulta são truncados para cinco caracteres ou preenchidos com espaços em branco à direita de até cinco caracteres (se houver menos de cinco caracteres para começar).

Exemplo

SELECT CAST(last\_name AS CHAR (5))

FROM employee

WHERE department\_number = 401;

last\_name

------------

Johns

Trade

Você também pode fazer uma conversão sem a função CAST conforme mostrado abaixo. Esse tipo de sintaxe de conversão é considerado abreviado e não é padrão ANSI.

Exemplo

SELECT last\_name (CHAR (5)) FROM employee;

Além do CASTing simples, o banco de dados Teradata permite que você especifique atributos de dados, bem como tipos de dados em seu CAST.

Exemplo

SELECT CAST(last\_name AS CHAR(5) UPPERCASE)

FROM employee

WHERE department\_number = 401;

last\_name

------------

JOHNS

TRADE

Itens a serem observados:

Esse SQL retornaria um erro em uma sessão do modo ANSI porque o truncamento de sequências de caracteres não é permitido em uma expressão CAST do modo ANSI.

É importante usar o tipo de dados apropriado ao projetar o banco de dados físico.

Escolher o tipo de dados apropriado ajudará a evitar conversões dispendiosas (por exemplo, comparar uma coluna numérica com uma coluna de caracteres) e manter as necessidades de armazenamento minimizadas (por exemplo, usar um BYTEINT em vez de um INTEGER economizará 3 bytes por linha no mesa).

LAB:

Se você não configurou a conexão do servidor do laboratório, clique no botão Configuração do laboratório na parte inferior da página para obter instruções. Você precisará dessas instruções para fazer logon no banco de dados Teradata. Se você tiver problemas para se conectar ao servidor do laboratório, entre em contato com Training.Support@Teradata.com.

Para este conjunto de exercícios de laboratório, você pode precisar de informações do documento Database Info. Antes de fazer esses laboratórios, será útil redefinir seu banco de dados padrão para o banco de dados CustomerService (ou seja, DATABASE CustomerService;).

Clique no botão Avançar na parte inferior da página para ver as respostas.

1.) Em uma única instrução SELECT, exiba a DATA, HORA e USUÁRIO atuais.

**SELECT** Date

, Time

, **User**

**;**

2.) Exibir a data 365 dias a partir de hoje.

**SELECT** CURRENT\_DATE + 365**;**

3.) Crie um relatório que forneça o valor literal "Função matemática" como sua primeira coluna, o valor numérico 1 como sua segunda coluna e o cálculo 2\*3+4\*5 como sua terceira coluna. Observe os resultados.

**Select** 'Mat Function'

,1

,((2\*3)+(4\*5))

**;**

4.) Modifique a solicitação nº 3 para incluir parênteses em torno de 3+4. Reenvie a solicitação. Observe a diferença no resultado.

**Select** 'Mat Function'

,1

,2\*(3+4)\*5

**;**

5.) Alguns funcionários estão recebendo aumentos. O gerente do departamento 401 está interessado em saber quais seriam os salários se cada funcionário recebesse um aumento de 10%. Liste os 10 primeiros caracteres do sobrenome, salário atual e salário após o aumento para os funcionários do departamento 401. Classifique-os em ordem decrescente de salário.

**Select** Cast (last\_name **as** CHAR(10))

,salary\_amount

,salary\_amount \* 1.1

**FROM** employee

**Where** department\_number = 401

**Order** **By** salary\_amount **DESC**

**;**

Envie novamente usando EXPLAIN para ver como o banco de dados Teradata processará essa consulta.

EXPLAIN

SELECT \*

FROM **CustomerService.employee**;

1) First, **we lock** a distinct CustomerService."pseudo table" for **read**

on a **RowHash** to prevent global deadlock for

**CustomerService.employee**.

2) Next, **we lock** **CustomerService.employee** for **read**.

3) We do an **all-AMPs** **RETRIEVE** step from **CustomerService.employee** by

way of an **all-rows scan** with no residual conditions into **Spool 1**

(group\_amps), which is **built** **locally** on the AMPs. The size of

**Spool 1** is estimated with low confidence to be **40 rows** (3,880

bytes). The estimated time for this step is 0.07 seconds.

4) Finally, we send out an **END TRANSACTION** step to all AMPs involved

in processing the request.

-> The contents of **Spool 1** are sent back to the user as the result of

statement 1. The total estimated time is 0.07 seconds.

6.) Os resultados do exercício #5 deixaram a gestão um pouco abalada. Em vez disso, eles estão considerando um aumento fixo de $ 500,00 por funcionário. Modifique a consulta para incluir esse novo salário estimado, bem como o aumento de 10% para comparação. Inclua uma coluna adicional que informe a diferença líquida em dólares entre as duas opções para cada funcionário. Renomeie esta coluna calculada inserindo as duas palavras AS Difference após o cálculo. Retire a coluna do salário atual do relatório.

**Select** Cast (last\_name **as** CHAR(10))

,salary\_amount

,salary\_amount \* 1.1

,salary\_amount + 500

, (salary\_amount\*1.1)- (salary\_amount+500) AS Difference

**FROM** employee

**Where** department\_number = 401

**Order** **By** salary\_amount **DESC**

**;**

7.) O governo ordenou um estudo especial de demografia do emprego. Gere uma lista de todos os funcionários com menos de 60 anos. Inclua o sobrenome (CAST apenas os dez primeiros caracteres), o número do departamento e a idade calculada em anos. Classifique para que os funcionários mais jovens sejam listados primeiro.

**Select** Cast (first\_name **as** CHAR(10))

,department\_number

,(DATE-birthdate)/365

**FROM** employee

**Where** (DATE-birthdate)/365 < 60

**Order** **By** birthdate **DESC**

**;**