**Capítulo 1**

Fundamentos da Consulta

Objetivos do exame neste capítulo:

■■ Trabalhar com dados

■■ dados de consulta usando instruções SELECT.

T ransact-SQL (T-SQL) é a principal língua utilizada para gerir e manipular dados no Microsoft SQL Server. Este capítulo estabelece as bases para consultar dados usando T-SQL. O capítulo descreve as raízes dessa linguagem, terminologia e a mentalidade, que você precisa adotar ao escrever um código T-SQL. Isto então se move para descrever um dos conceitos mais importantes que você precisa saber sobre a consulta de linguagem lógica em processamento.

Embora este capítulo não se destine diretamente ao exame específico da linguagem, outros também irão abordar e discutir o projeto da SELECT objetivos e declarações, a principal declaração T-SQL usada para consultar os dados, o restante dos capítulos deste kit de treinamento também irá o fazer. No entanto, a informação contida neste capítulo é fundamental, a fim de ajudar corretamente a compreender o restante do livro.

Lições neste capítulo:

■■ Lição 1: Compreender os Fundamentos da T-SQL

■■ Lição 2: compreensão lógica Processamento de consulta

**Antes de você começar**

Para concluir as lições deste capítulo, você deve ter:

■■ A compreensão dos conceitos básicos de banco de dados.

Experiência

 ■■ Trabalhar com SQL Server Management Studio (SSMS).

■■. Algum código T-SQL experiência de escrita.

■■ O acesso a um Server 2012 instância do SQL com o banco de dados exemplo TSQL2012 instalado.

(Por favor, veja a introdução do livro para obter detalhes sobre como criar o banco de dados de exemplo.)

***Você leu A página xxx?***

**Ela contém valiosas informações relativas as habilidades que você precisa para passar no exame!**

**Lição 1: Compreender os Fundamentos da T-SQL**

Muitos aspectos da computação, como linguagens de programação, evoluíram com base na intuição e da tendência atual. Sem bases sólidas, sua vida útil pode ser muito curta, e se o fizerem sobreviver, muitas vezes, as alterações são muito rápidas devido a alterações nas tendências. T-SQL é diferente, principalmente porque tem fortes fundamentos em matemática. Você não precisa ser um matemático para

Escrever uma boa SQL (embora certamente não seria ruim saber matemática), mas contanto que você entenda o que as funções são, e alguns de seus princípios fundamentais, você vai entender melhor a linguagem que você está lidando. Sem esses fundamentos, você ainda pode escrever um código T-SQL que vai ser executado com sucesso, mas será como comer sopa com um garfo! Então após esta lição, você será capaz de:

■■. Descrever as fundações que T-SQL se baseia.

■■. Descrever a importância do uso de T-SQL de uma forma relacional.

■■. Use a terminologia correta para descrever os elementos relacionados com a T-SQL.

Tempo de aula estimado: 40 minutos

Evolução do T-SQL

Como mencionado, ao contrário de muitos outros aspectos da computação, T-SQL tem forte base na matemática e em suas funções. Compreender alguns dos princípios-chave a partir dessas bases pode ajudar você a entender melhor a linguagem que você está lidando. Então você vai pensar em termos de T-SQL quando a codificação em T-SQL, ao contrário de codificação com T-SQL ao pensar em termos processuais.

Figura 1-1 ilustra a evolução da T-SQL a partir de suas bases matemáticas básicas.

T-SQL

SQL

Modelo relacional

Teoria dos Conjuntos Predicate Logic

**Figura 1-1** Evolução do T-SQL.

T-SQL é a principal língua utilizada para gerenciar e manipular dados em principal relacional do Microsoft

sistema de gerenciamento de banco de dados (RDBMS), SQL Server-se no local ou no nuvem (banco de dados Microsoft Windows Azure SQL). SQL Server também suporta outros idiomas, como o Microsoft Visual C # e Microsoft Visual Basic, mas T-SQL é geralmente o idioma preferido para gerenciamento de dados e manipulação.

T-SQL é um dialeto do SQL padrão. SQL é um padrão de tanto a Organização Internacional

de Padrões (ISO) e da American National Standards Institute (ANSI). Os dois padrões

para SQL são basicamente os mesmos. O padrão SQL continua a evoluir com o tempo. Segue-se uma lista

das grandes revisões do padrão até o momento:

■■ SQL-86

■■ SQL-89

■■ SQL-92

■■ SQL: 1999

■■ SQL: 2003

■■ SQL: 2006

■■ SQL: 2008

■■ SQL: 2011

Todos os fornecedores de banco de dados principais, incluindo Microsoft, implementar um dialeto do SQL como principal linguagem para gerenciar e manipular dados em suas plataformas de banco de dados. Por conseguinte, o núcleo elementos de linguagem têm a mesma aparência. No entanto, cada fornecedor decide quais recursos para implementar e que não. Além disso, o padrão de, por vezes, deixa alguns aspectos como uma implementação escolha. Cada fornecedor também geralmente implementa extensões para o padrão nos casos em que o fornecedor considera que uma importante característica não é coberta pelo padrão.

Escrevendo em uma forma padrão é considerado uma boa prática. Quando você faz isso, seu código é mais portátil. Seu conhecimento é mais portátil, também, porque é fácil para você começar a trabalhar com novas plataformas. Quando o dialeto você está trabalhando com suporta tanto um padrão e uma maneira fora do padrão para fazer alguma coisa, você deve sempre preferir a forma padrão como a sua escolha padrão. Você deve considerar uma opção fora do padrão somente quando ele tem alguns importantes benefício para você que não é coberto pela alternativa padrão.

Como um exemplo de quando a escolha da forma padrão, T-SQL suporta dois "não é igual a" operadores:

<> e =!. O primeiro é padrão e o último não é. Este caso deve ser um nobrainer: para ir a um padrão!

Como um exemplo de quando a escolha do padrão ou não padronizado depende das circunstâncias,

considere os seguintes:

T-SQL suporta múltiplas funções que convertem um valor de origem a um tipo de destino. Entre eles estão o CAST e CONVERT funções. O primeiro é padrão e esta não é. A função CONVERT fora do padrão tem um argumento de estilo que ELENCO não suporta. Porque ELENCO é padrão, você deve considerar que a sua escolha padrão para conversões. Você deve considerar o uso de converter somente quando você precisa contar com o estilo argumento.Ainda outro exemplo de escolher a forma padrão é no encerramento de instruções T-SQL.De acordo com o padrão SQL, você deve encerrar as instruções com um ponto e vírgula.T-SQL atualmente não tornar este um requisito para todas as declarações, apenas em casos onde há de outra forma seria ambiguidade de elementos de código, tais como na cláusula COM de um comum

expressão de tabela (CTE). Você ainda deve seguir o padrão e encerrar todas as suas declarações mesmo que de momento não é necessária. SQL padrão é baseado no *modelo relacional,* que é um modelo matemático para dados gestão e manipulação. O modelo relacional foi inicialmente criado e proposto por

Edgar F. Codd, em 1969. Desde então, tem sido explicado e desenvolvido por Chris Date, Hugh Darwen, e outros. Um equívoco comum é que o nome "relacional" tem a ver com as relações

entre tabelas (isto é, chaves estrangeiras). Na verdade, a verdadeira fonte para o nome do modelo é o

matemática conceito *de relação.* A relação no modelo relacional é o SQL chama uma *mesa.* Os dois não são sinônimos. Pode-se dizer que uma mesa é uma tentativa de SQL para representar uma relação (para além de uma relação variável, mas isso não é necessário para entrar aqui). Alguns poderiam dizer que não é muito tentativa bem sucedida. Embora SQL baseia-se no modelo relacional, afasta-se a partir dele numa

número de maneiras. Mas é importante notar que, como você entender os princípios do modelo, você pode usar SQL-ou mais precisamente, o dialeto você estiver usando-in uma maneira relacional. Mais sobre isso,

incluindo uma nova recomendação de leitura, é na próxima seção, "Usando T-SQL em um Relational Caminho." Voltando a uma relação, que é o SQL tenta representar com uma mesa: uma relação tem um título e um corpo. O título é um conjunto de atributos (o SQL tenta representar com colunas), cada um de um dado tipo. Um atributo é identificado pelo nome e nome do tipo. O corpo é um conjunto de tuplas (o SQL tenta representar com linhas). Cada posição do tuplo é a posição da relação. Cada valor do atributo de cada tupla é do respectivo tipo. Alguns dos princípios mais importantes para entender sobre haste T-SQL do relacional núcleo do modelo fundações-set teoria e lógica de predicados. Lembre-se que o título de uma relação é um conjunto de atributos, e o corpo de um conjunto de tuplas. Então, o que é um conjunto? De acordo com o criador da teoria dos conjuntos matemática, Georg Cantor, um *conjunto* é descrito como se segue:

*Por um "set" queremos dizer qualquer coleção M em um todo de definidos, objetos distintos m (que são chamados de "elementos" de M) de nossa percepção ou de nosso pensamento.* –

G Eorge C Antor, em "G EORG C Antor" por J oseph W. D auben

(P rinceton U niversidade P ress, 1990)

Há um certo número de princípios muito importante nesta definição que, se compreendidos, deve ter implicações diretas sobre suas práticas de codificação T-SQL. Por um lado, notar o termo *todo.* Um conjunto deve ser considerada como um todo. Isso significa que você não interagir com o elementos individuais do conjunto, em vez com o conjunto como um todo. Observe o termo *distinto* conjunto -a não tem duplicatas. Codd uma vez comentou sobre os sem duplicatas aspecto: "Se algo é verdadeiro, então dizendo que duas vezes não vai fazer com que seja mais verdadeiro." Por exemplo, o conjunto {a, b, c} é considerada igual ao conjunto {a, a, b, c, c, c}. Outro aspecto crítico de um conjunto não aparece explicitamente na definição acima mencionada por Cantor, mas está implícito-não há nenhuma relevância para a ordem dos elementos em um conjunto. Dentro contraste, uma sequência (que é um conjunto *ordenado),* por exemplo, tem uma ordem para os seus elementos. Combinando os sem duplicatas e nenhuma relevância para encomendar aspectos significa que o conjunto {a, b, c} é igual ao conjunto {b, A, C, C, A, C}. O outro ramo da matemática que o modelo relacional é baseada em é chamado de predicado lógica. Um *predicado* é uma expressão que, quando atribuída a algum objeto, faz uma proposição verdadeiro ou falso. Por exemplo, "salário maior que $ 50.000" é um predicado. Você pode avaliar este predicado para um funcionário específico, caso em que você tem uma proposição. Por exemplo, supor que para um determinado empregado, o salário é de R $ 60.000. Quando você avalia a proposição para que o empregado, você tem uma proposição verdadeira. Em outras palavras, um predicado é um parametrizada proposição. O modelo relacional usa predicados como um de seus elementos centrais. Você pode impor dados integridade usando predicados. Você pode filtrar os dados usando predicados. Você ainda pode usar predicados a definir o modelo em si os dados. Você primeiro identificar proposições que precisam ser armazenados na base de dados. Aqui está um exemplo proposição: uma ordem com 10248 fim ID foi colocad o em 12 de fevereiro de 2012 por parte do cliente com ID 7, e manipulados pelo empregado com ID 3. Você em seguida, criar predicados de proposições, removendo os dados e manter o título. Lembre-se, o título é um conjunto de atributos, cada um identificado por nome e nome do tipo. Nisso exemplo, você orderId INT, DATE orderdate, INT custid e INT empid.

**Checagem rápida**

1. Quais são os ramos matemáticos que o modelo relacional é baseado?

2. Qual é a diferença entre T-SQL e SQL?

**Quick Check Resposta**

1. A teoria dos conjuntos e lógica de predicados.

2. SQL é padrão; T-SQL é o dialeto de e extensão para SQL que Microsoft implementa em seu RDBMS-SQL Server.

Usando T-SQL em uma maneira relacional

Tal como mencionado na secção anterior, o t-SQL baseia-se na SQL, que por sua vez se baseia na modelo relacional. No entanto, há um número de maneiras nas quais o SQL-e, portanto, t-SQL- afasta-se o modelo relacional. Mas a linguagem fornece ferramentas suficientes para que se compreender o modelo relacional, você pode usar a linguagem de forma relacional e, portanto, escrever código mais correto.

*mais informações* **SQL e teoria relacional**

Para obter informações detalhadas sobre as diferenças entre SQL e o modelo relacional e como usar o SQL de uma forma relacional, consulte *SQL e relacional Theory,* Second Edition por CJ Data (O'Reilly Media, 2011). É um excelente livro que todos os praticantes de banco de dados deve ler.

Lembre-se que uma relação tem um título e um corpo. O título é um conjunto de atributos e o corpo é um conjunto de tuplas. Lembre-se da definição de um conjunto que um conjunto é suposto ser considerado como um todo. O que isto traduz-se em T-SQL é que você deveria escrever consultas que interagem com as tabelas como um todo. Você deve tentar evitar o uso de construções iterativos como cursores e loops que iterar através das linhas, uma de cada vez. Você também deve tentar evitar pensar em termos iterativos, porque esse tipo de pensamento é o que leva a soluções iterativos. Para as pessoas com um fundo de programação processual, a maneira natural de interagir com dados (em um arquivo, conjunto de registros, ou leitor de dados) é com iterações. Então, usando os cursores e outras iterativa construções em T-SQL é, de certa forma, uma extensão para o que já sabem. No entanto, a correcta caminho do ponto de vista do modelo relacional não é para interagir com as linhas de um de cada vez; em vez disso, use as operações relacionais e retornar um resultado relacional. Este, por sua T-SQL, se traduz em escrever consultas. Lembre-se também que um conjunto não tem duplicatas. T-SQL nem sempre aplicar esta regra. Por exemplo, você pode criar uma tabela sem uma chave. Em tal caso, você está autorizado a ter duplicado linhas na tabela. Para acompanhar teoria relacional, você precisa impor exclusividade em sua mesa por exemplo, usando uma chave primária ou uma restrição única. Mesmo quando a tabela não permite que as linhas duplicadas, uma consulta em relação a tabela ainda pode retornar linhas duplicadas em seu resultado. Você vai encontrar uma discussão mais aprofundada sobre duplicatas em capítulos subsequentes, mas aqui é um exemplo para fins de ilustração. Considere a seguinte consulta.

USE TSQL2012; SELECT country FROM HR.Employees;

A consulta é emitida contra o banco de dados exemplo TSQL2012. Ele retorna o atributo país dos empregados armazenados na tabela de HR.EMPLOYEES. De acordo com o modelo relacional, uma operação relacional contra uma relação deve retornar uma relação. Neste caso, isto deve traduzir para devolver o conjunto de países onde há empregados, com ênfase na definido, como em nenhuma duplicata. No entanto, T-SQL não tenta remover duplicatas por padrão. Aqui está a saída dessa consulta.

Country

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

USA

USA

USA

USA

UK

UK

UK

USA

UK

De fato, T-SQL é baseada mais na teoria do que na multiset teoria dos conjuntos. A *multiset* (também conhecido como um saco ou um super), em muitos aspectos, é semelhante ao de um conjunto, mas pode ter duplicados. Como mencionado, a linguagem T-SQL dá-lhe ferramentas suficientes para que, se você quiser seguir relacional teoria, você pode fazê-lo. Por exemplo, a linguagem fornece-lhe com uma cláusula

DISTINCT para remover duplicados. Aqui está a consulta revisto.

SELECT DISTINCT country FROM HR.Employees;

Aqui está a saída da consulta revisto.

Country

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

UK

USA

Outro aspecto fundamental de um conjunto é que não há nenhuma relevância para a ordem dos elementos.

Por esta razão, as linhas de uma tabela tem nenhuma ordem particular, conceptualmente. Então, quando você emitir uma consulta contra uma tabela e não indicam explicitamente que deseja retornar as linhas em

especial ordem de apresentação, o resultado é suposto ser relacional. Portanto, você não deve assume nenhuma ordem específica para as linhas no resultado, não importa o que você sabe sobre o representação física dos dados, por exemplo, quando os dados são indexados. Como um exemplo, considere o seguinte consulta.

SELECT empid, lastname FROM HR.Employees;

Quando esta consulta foi executada em um sistema, ele retornou o seguinte resultado, que parece ele é classificado pelo sobrenome coluna.

empid lastname

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

5 Buck

8 Cameron

1 Davis

9 Dolgopyatova

2 Funk

7 King

3 Lew

4 Peled

6 Suurs

Mesmo se as linhas foram retornadas em uma ordem diferente, o resultado teria ainda sido considerada

corrigir. SQL Server pode escolher entre diferentes métodos de acesso físico para processar a consulta, sabendo que ele não precisa de garantir a ordem na sequência. Por exemplo, o SQL Servidor pode decidir paralelizar a consulta ou digitalizar os dados em ordem de arquivos (em oposição ao índice ordem). Se você precisa fazer para garantir uma ordem de apresentação específica para as linhas no resultado, você

precisa adicionar uma cláusula ORDER BY à consulta, como se segue.

SELECT empid, lastname FROM HR.Employees ORDER BY empid;

Desta vez, o resultado não é relacional-it é o padrão SQL chama um *cursor.* A ordem das linhas na saída é garantida com base no atributo empid. Aqui está a saída dessa consulta.

empid lastname

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

1 Davis

2 Funk

3 Lew

4 Peled

5 Buck

6 Suurs

7 King

8 Cameron

9 Dolgopyatova

O título de uma relação é um conjunto de atributos que devem ser identificados pelo nome e digite o nome. Não há nenhuma ordem para os atributos. Por outro lado, T-SQL faz manter o controle de posições ordinais das colunas com base em sua ordem de aparição na definição da tabela. Quando você emitir uma consulta com SELECT \*, você está garantido para obter as colunas no resultado com base em fim definição. Além disso, T-SQL permite referindo-se a posições ordinais das colunas do resultado a cláusula ORDER BY, como se segue.

SELECT empid, lastname FROM HR.Employees ORDER BY 1;

Além do fato de que esta prática não é relacional, pensar sobre o potencial de erro se a algum momento você alterar a lista SELECT e se esqueça de alterar a lista ORDER BY em conformidade. Portanto, a recomendação é sempre indicar os nomes dos atributos que você precisa de ordem por. T-SQL tem um outro desvio do modelo relacional na medida em que permite a definição de resultado colunas com base em uma expressão sem atribuir um nome à coluna de destino. Por exemplo, a consulta a seguir é válida em T-SQL.

SELECT empid, firstname + ' ' + lastname FROM HR.Employees;

Esta consulta gera a seguinte saída.

empid

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

1 Sara Davis

2 Don Funk

3 Judy Lew

4 Yael Peled

5 Sven Buck

6 Paul Suurs

7 Russell King

8 Maria Cameron

9 Zoya Dolgopyatova

9 Zoya Dolgopyatova

Mas de acordo com o modelo relacional, todos os atributos devem ter nomes. Para que o consulta a ser relacional, é necessário atribuir um alias para o atributo de destino. Você pode fazê-lo utilizando a cláusula AS, como se segue.

SELECT empid, firstname + ' ' + lastname AS fullname FROM HR.Employees;

Além disso, T-SQL permite uma consulta para retornar várias colunas de resultado com o mesmo nome. Para exemplo, considere a junção entre duas tabelas, T1 e T2, ambos com uma coluna chamada keycol. T-SQL permite que uma lista SELECT que se parece com o seguinte.

SELECT T1.keycol, T2.keycol ...

Para que o resultado seja relacional, todos os atributos devem ter nomes exclusivos, portanto, seria necessário para usar diferentes nomes alternativos para os atributos do resultado, como a seguir.

SELECT T1.keycol AS key1, T2.keycol AS key2 ...

Quanto aos predicados, seguindo a *lei do terceiro excluído* na lógica matemática, um predicado

pode ser avaliada como verdadeira ou falsa. Em outras palavras, predicados são supostamente para usar a lógica de dois valores. No entanto, Codd queria refletir a possibilidade de valores estar faltando em seu modelo. Ele a que se refere a dois tipos de valores em falta: falta, mas aplicável e em falta, mas inaplicável.

Tome um atributo de telemóvel de um empregado como um exemplo. Um valor em falta, mas o casoseria se um empregado tem um telefone celular, mas não quis fornecer essa informação, por exemplo, por razões de privacidade. Um valor em falta, mas inaplicável seria quando o empregado simplesmente não tem um telefone celular. De acordo com Codd, uma linguagem baseada em seu modelo deve fornecer duas marcas diferentes para os dois casos. T-SQL-novo, baseado no padrão SQL-implementos somente uma marca de propósito geral chamados NULL para qualquer tipo de valor em falta. Isto leva a lógica de predicados de três valores. Nomeadamente, quando um predicado compara dois valores, por exemplo, telemóvel = '(425) 555-0136', se ambos estiverem presentes, o resultado é avaliado como verdadeiro ou falso. Mas se um deles é NULL, o resultado é avaliada como um terço de valor lógico desconhecido. Note-se que alguns acreditam que um modelo relacional válida deve seguir a lógica de dois valores, e fortemente opor-se ao conceito de nulos em SQL. Mas como mencionado, o criador do relacional modelo acreditava na ideia de apoiar os valores em falta, e predicados que se estendem além da lógica de dois valores. O que é importante do ponto de vista de codificação com T-SQL é perceber que, se o banco de dados que você está consultando suportes nulos, o seu tratamento está longe de ser trivial. Ou seja, você precisa entender cuidadosamente o que acontece quando nulos são envolvidos nos dados que você está manipulando com várias construções de consulta, como filtragem, classificação, agrupamento, juntando, ou interseção. Assim, com cada pedaço de código que você escreve com T-SQL, você quer perguntar -se se nulos são possíveis nos dados que você está interagindo com. Se a resposta for sim, você quer ter certeza de que você compreende o tratamento de nulos em sua consulta e garantir que os testes de abordar o tratamento de nulos especificamente.

**Checagem rápida**

1. Nome dois aspectos em que T-SQL se desvia do modelo relacional.

2. Explicar como você pode abordar os dois itens em questão 1 e usar T-SQL em um maneira relacional.

**Quick Check Resposta**

1. Uma relação tem um corpo com um conjunto distinto de tuplas. A tabela não tem que ter

uma chave. T-SQL permite referindo-se a posições ordinais das colunas na ORDER BY cláusula.

2. Definir uma chave em cada tabela. Referem-se a nomes de atributos, não sua ordinal posições-na cláusula ORDER BY.

**Quick Check**

1. Name two aspects in which T-SQL deviates from the relational model.

2. Explain how you can address the two items in question 1 and use T-SQL in a

relational way.

**Quick Check Answer**

1. A relation has a body with a distinct set of tuples. A table doesn’t have to have

a key. T-SQL allows referring to ordinal positions of columns in the ORDER BY

clause.

2. Define a key in every table. Refer to attribute names—not their ordinal

positions—in the ORDER BY clause.

Usando terminologia correta

O uso da terminologia reflete sobre o seu conhecimento. Portanto, você deve fazer um esforço para compreender e usar a terminologia correta. Ao discutir T-relacionados com o SQL tópicos, as pessoas costumam usar termos incorretos. E se isso não é suficiente, mesmo quando você percebe o que a correta

termos são, você também precisa entender as diferenças entre os termos em T-SQL e aqueles no modelo relacional. Como exemplo de termos incorretos em T-SQL, as pessoas costumam usar os termos "campo" e "Record" para se referir ao que T-SQL chama de "coluna" e "linha", respectivamente. Campos e registros são físicos. Os campos são o que você tem em interfaces de usuário em aplicativos cliente, e os registros são o que você têm em arquivos e cursores. Tabelas são lógicas, e eles têm linhas e colunas lógicas. Outro exemplo de um termo incorreto é referindo-se a "valores nulos." A NULL é uma marca para um falta de valor não é um valor em si. Assim, o uso correto do termo é o "ponto de NULL" ou apenas "NULL". Além de usar a terminologia T-SQL correta, também é importante compreender as diferenças entre os termos T-SQL e os seus homólogos relacionais. Lembre-se da seção anterior que a T-SQL tenta representar uma relação com uma mesa, uma tupla com uma linha e um atributo com uma coluna; mas os conceitos de T-SQL e os seus homólogos relacionais diferem em vários de maneiras. Enquanto você está consciente dessas diferenças, você pode, e deve, se esforçam para usar T-SQL de uma forma relacional.

**Checagem rápida**

1. Por que os termos "campo" e "Record" incorreto quando referentes a coluna e

linha?

2. Porque é que o termo "valor nulo" incorreta?

**Quick Check Resposta**

1. Porque "campo" e "record" descrever as coisas físicas, enquanto as colunas e

linhas são elementos lógicos de uma mesa.

2. Porque NULL não é um valor; mas sim, é uma marca para um valor em falta.

**Quick Check**

1. Why are the terms “field” and “record” incorrect when referring to column and

row?

2. Why is the term “NULL value” incorrect?

**Quick Check Answer**

1. Because “field” and “record” describe physical things, whereas columns and

rows are logical elements of a table.

2. Because NULL isn’t a value; rather, it’s a mark for a missing value.

Prática

**Usando T-SQL em uma maneira relacional**

Nesta prática, você exercer o seu conhecimento do uso de T-SQL de uma forma relacional. Se você encontrar um problema de completar um exercício, você pode instalar os projectos concluídos a partir da pasta Solution que é fornecido com o conteúdo complementar para este capítulo e lição.

Exercício 1

Identificar os elementos não-relacionais em uma consulta Neste exercício, você terá uma consulta. Sua tarefa é identificar os elementos não-relacionais ema consulta. Estúdio de gestão de Servidor  1. Abra SQL (SSMS) e se conectar ao banco de dados de exemplo TSQL2012. (Veja a introdução do livro para obter instruções sobre como criar o exemplo banco de dados e como trabalhar com SSMS). 2. Digite a seguinte consulta na janela de consulta e executá-lo.

SELECT custid, YEAR(orderdate) FROM Sales.Orders ORDER BY 1, 2;

Você obtém o seguinte resultado mostrado aqui de forma abreviada.

custid

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

1, 2007

1, 2007

1, 2007

1, 2008

1, 2008

1, 2008

2, 2006

2, 2007

2, 2007

2, 2008

...

3. Examine o código e sua saída. A consulta deve retornar para cada cliente e fim do ano o ID do cliente (custid) e ano fim (ANO (orderdate)). Observe que não há nenhuma exigência de apresentação de pedidos a partir da consulta. Você pode identificar o que os aspectos não-relacionais da consulta são? Resposta: A consulta não alias o ANO expressão (orderdate), então não há nenhum nome para o atributo resultado. A consulta pode retornar duplicatas. A consulta força certa apresentação ordenando ao resultado e usa posições ordinais na cláusula ORDER BY. Exercício 2 Faça o não-relacional de consulta relacional Neste exercício, você trabalha com a consulta fornecida no Exercício 1 como seu ponto de partida. Depois de identificar os elementos não-relacionais na consulta, é necessário aplicar as revisões apropriadas para torná-lo relacional.

■■ Na etapa 3 do Exercício 1, você identificou os elementos não-relacionais na última consulta. Aplicar revisões da consulta para torná-lo relacional. Uma série de revisões são obrigados a fazer a relacional consulta.

■■ Defina um nome de atributo, atribuindo um alias para o ANO expressão (orderdate).

■■ Adicionar uma cláusula DISTINCT para remover duplicatas.

■■ Além disso, remova a cláusula ORDER BY para retornar um resultado relacional.

■■ Mesmo se houvesse uma exigência de apresentação de pedidos (não neste caso), que deveria

Não use posições ordinais; em vez disso, use nomes de atributos. Seu código deve ser semelhante

Os seguintes.

SELECT DISTINCT custid, YEAR(orderdate) AS orderyear FROM Sales.Orders;

Resumo da lição

■■ T-SQL é baseada em fortes fundamentos matemáticos. Ele é baseado no padrão SQL, que por sua vez é baseada no modelo relacional, que por sua vez é baseada na teoria e conjunto lógica de predicados.

■■ É importante entender o modelo relacional e aplicar seus princípios ao escrever

Código T-SQL.

■■ Ao descrever conceitos em T-SQL, você deve usar a terminologia correta porque

reflete sobre o seu conhecimento.

Examine a lição

Responda as seguintes perguntas para testar seu conhecimento da informação nesta lição. Vocês podem encontrar as respostas para estas perguntas e explicações de por que cada opção de resposta está correta

ou incorreta na secção "Respostas" no final deste capítulo.

1. Por que é importante usar o código SQL padrão quando possível e saber o que é padrão e o que, não é? (Escolha tudo o que se aplicam).

A. Não é importante para código usando SQL padrão. Código

B. padrão SQL é mais portátil entre plataformas. Código

C. padrão SQL é mais eficiente.

D. Saber o código SQL padrão faz o seu conhecimento ser mais agil.

2. Qual das seguintes não é uma violação do modelo relacional?

A. Usando posições ordinais para colunas

B. Retornando linhas duplicadas

C. Não definindo uma chave em uma tabela

D. Assegurar que todos os atributos no resultado de uma consulta têm nomes

3. Qual é a relação entre SQL e T-SQL?

A. T-SQL é a linguagem padrão e SQL é o dialeto no Microsoft SQL Server.

B. SQL é a linguagem padrão e T-SQL é o dialeto no Microsoft SQL Server.

C. Ambos SQL e T-SQL são linguagens padrão.

D. Ambos SQL e T-SQL são dialetos em Microsoft SQL Server.

**Lição 2: compreensão lógica Processamento de consulta**

T-SQL tem lados tanto lógicos e físicos a ele. O lado lógico é a interpretação conceitual da consulta que explica o que o resultado correto da consulta é. O lado físico é o processamento da consulta pelo mecanismo de banco de dados. Transformação física deve produzir o resultado definido pela lógica de processamento de consultas. Para atingir este objetivo, o motor de banco de dados pode aplicar a otimização. Optimization pode reorganizar passos do processamento de consultas lógica ou remover etapas por completo, mas apenas desde que o resultado continua a ser aquele definido pela lógica de processamento de consultas. O foco desta lição é *o processamento de consultas lógica* -a interpretação conceitual do consulta que define o resultado correto. Após esta lição, você será capaz de

: ■■ compreender o raciocínio para o desenho de T-SQL.

■■ Descrever as principais fases de processamento de consultas lógicas.

■■ explicar as razões para algumas das restrições em T-SQL.

Tempo de aula estimado: 40 minutos

T-SQL Como declarativa Inglês-Like Idioma

T-SQL, sendo baseada no padrão SQL, é uma linguagem de Inglês-like declarativa. Nesta língua, declarativa significa que você definir *o que* você quer, ao contrário do *imperativo* línguas que definem Também *como* conseguir o que deseja. Padrão SQL descreve a interpretação lógica da pedido declarativa ( "o que" parte), mas é responsabilidade do motor de banco de dados para descobrir como processar fisicamente o pedido (o "como" parte). Por esta razão, é importante não para tirar quaisquer conclusões relacionadas com o desempenho de o que você aprender sobre o processamento de consultas lógico. Isso porque o processamento de consultas lógica única define a correção da consulta. Ao abordar aspectos de desempenho da consulta, você precisa entender como funciona a otimização. Como mencionado, a optimização pode ser bastante diferente de processamento de consulta lógico, porque é permitido para mudar as coisas, desde que o resultado alcançado é o definido pelo processamento de consultas lógico.

É interessante notar que a linguagem SQL padrão não foi originalmente chamado assim; em vez, ele foi chamado SEQUEL; um acrônimo para "linguagem estruturada consulta Inglês." Mas, em seguida, devido a uma disputa de marca com uma companhia aérea, a linguagem foi renomeado para SQL, para "estruturado

linguagem de consulta. "Ainda assim, o ponto é que você fornecer suas instruções de forma Inglês-like. Por exemplo, considere a instrução, "Traga-me um refrigerante da geladeira." Observar que na instrução em Inglês, o objeto vem antes da localização. Considere o seguinte solicitação na T-SQL.

SELECT shipperid, phone, companyname

FROM Sales.Shippers;

Observe a semelhança de ordem com chave-in da consulta para Inglês. A consulta primeira indica o SELECT lista com os atributos que deseja retornar e, em seguida cláusula FROM com a tabela que você deseja consultar. Agora tente pensar a ordem em que o pedido tem de ser logicamente interpretada. Para

exemplo, como você definiria as instruções para um robô, em vez de um ser humano? O original Instrução de Inglês para pegar um refrigerante da geladeira, provavelmente precisa ser revisto para algo como: "Vá para a geladeira; abra a porta; pegar um refrigerante; traga isso para mim." Da mesma forma, o processamento lógico de uma consulta deve primeiro saber qual tabela está sendo consultado antes que ele possa saber quais atributos podem ser devolvidos a partir dessa tabela. Portanto, ao contrário do a ordem digitadas da consulta anterior, a lógica de processamento de consulta tem que ser como se segue.

FROM Sales.Shippers  
SELECT shipperid, phone, companyname

Este é um exemplo básico com apenas duas cláusulas de consulta. Claro, as coisas podem ficar mais complexa. Se você entender o conceito de processamento de consulta lógica bem, você vai ser capaz de explicar muitas coisas sobre a forma como a linguagem se comporta-coisas que são muito difícil de explicar

de outra forma.

Fases de Consulta de processamento lógico

Esta seção aborda o processamento de consultas lógico e as fases envolvidas. Não se preocupe se alguns dos os conceitos discutidos aqui não são claras ainda. Próximos capítulos neste kit de treinamento fornecem mais detalhes, e depois de passar por cima daqueles, este tópico deve fazer mais sentido. Para ter a certeza você realmente entender esses conceitos, fazer uma primeira passagem sobre o tópico agora e, em seguida, voltar a ele mais tarde, depois de passar os capítulos 2 a 5. A declaração principal usado para recuperar dados em T-SQL é a instrução SELECT. Seguem-se as principais cláusulas de consulta especificadas na ordem em que você deve para digitá-los (conhecido como"Keyed-in order"):

1. SELECT

2. FROM

3. WHERE

4. GROUP BY

5. HAVING

6. ORDER BY

Mas como mencionado, a ordem de processamento de consulta lógica, que é a interpretação conceitual fim, é diferente. Ele começa com a cláusula FROM. Aqui está o processamento de consultas lógica ordem das seis cláusulas de consulta principais:

1. FROM

2.WHERE  
3.GROUP BY  
4.HAVING  
5.SELECT  
6.ORDER BY

Cada fase opera em um ou mais tabelas como entradas e retorna uma tabela virtual como saída.A tabela de uma fase de saída é considerada a entrada para a próxima fase. Isso está de acordo com operações sobre as relações que produzem uma relação. Observe que se um ORDER BY é especificada, o sultadonão é relacional. Este fato tem implicações que são discutidos mais adiante neste kit de treinamento, no

Capítulo 3, "Filtrando e classificando dados," e Capítulo 4, "Conjuntos de combinação."

Considere a seguinte consulta como um exemplo.

SELECT country, YEAR(hiredate) AS yearhired, COUNT(\*) AS numemployees

FROM HR.Employees

WHERE hiredate >= '20030101'

GROUP BY country, YEAR(hiredate)

HAVING COUNT(\*) > 1

ORDER BY country , yearhired DESC;

Esta consulta é emitida contra a mesa de HR.EMPLOYEES. Ele filtra apenas os funcionários que estavam

contratado em ou após o ano de 2003. Os grupos de TI os demais funcionários por país e o aluguer ano. Ele mantém apenas grupos com mais de um empregado. Para cada grupo de qualificação, o consulta retorna o ano de aluguer e contagem de funcionários, classificado por país e contratar ano, em ordem decrescente

ordem.

As seguintes seções fornecem uma breve descrição do que acontece em cada fase de acordo para o processamento de consultas lógico.

1. Avaliar a cláusula FROM

Na primeira fase, a cláusula FROM é avaliada. É aí que você indicar as tabelas que você quer para consultar e operadores de mesa como se junta se for o caso. Se precisar de consultar apenas uma mesa, você indicar o nome da tabela como a tabela de entrada nesta cláusula. Em seguida, a saída desta fase é um resultado tabela com todas as linhas da tabela de entrada. Esse é o caso da seguinte consulta: a entrada é a tabela de HR.EMPLOYEES (nove linhas), ea saída é um resultado tabela com todas as nove linhas (apenas um subconjunto dos atributos são mostrados).

empid hiredate country

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

1 2002-05-01 USA

2 2002-08-14 USA

3 2002-04-01 USA

4 2003-05-03 USA

5 2003-10-17 UK

6 2003-10-17 UK

7 2004-01-02 UK

8 2004-03-05 USA

9 2004-11-15 UK

2. Linhas filtro com base em cláusula WHERE

As linhas segundo filtros de fase com base no predicado na cláusula WHERE. Somente as linhas para

que o predicado é avaliada como verdadeira são devolvidos.

*Dica exame*

Linhas para as quais o predicado avaliar para false, ou é avaliada como um estado desconhecido, não são devolvidas.

Nesta consulta, o que a filtragem de filtros de fase somente as linhas para funcionários admitidos a partir

1º de janeiro de 2003. Seis linhas são retornadas a partir desta fase e são fornecidos como entrada para a próxima um. Aqui está o resultado desta fase.

empid hiredate country

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

4 2003-05-03 USA

5 2003-10-17 UK

6 2003-10-17 UK

7 2004-01-02 UK

8 2004-03-05 USA

9 2004-11-15 UK

Um erro típico feito por pessoas que não entendem o processamento de consultas lógica está tentando para se referir na cláusula WHERE para um alias de coluna definido na cláusula SELECT. Este não é permitido porque a cláusula WHERE é avaliada antes da cláusula SELECT. Como um exemplo, considere a seguinte consulta.

SELECT country, YEAR(hiredate) AS yearhired

FROM HR.Employees

WHERE yearhired >= 2003;

Esta consulta falha com o seguinte erro.

Msg 207, Level 16, State 1, Line 3

Invalid column name 'yearhired'

Se você entender que a cláusula WHERE é avaliada antes de a cláusula SELECT, você percebe que esta tentativa é errado, porque nesta fase, o atributo yearhired ainda não existe.Você pode indicar o ano de expressão (HireDate)> = 2003 na cláusula WHERE. Melhor ainda, por razões de otimização que são discutidos no Capítulo 3 e Capítulo 15 ", índices de execução e Estatística ", use o formulário HireDate> =" 20030101 ", como feito na consulta original.

3. Linhas grupo com base na cláusula GROUP BY

Esta fase define um grupo para cada combinação distinta de valores nos elementos agrupados a partir da tabela de entrada. Em seguida, ele associa cada linha de entrada para o seu respectivo grupo. a consulta

você tem trabalhado com grupos as linhas por país e ano (orderdate). Dentro das seis linhas na tabela de entrada, este passo identifica quatro grupos. Aqui estão os grupos e as linhas de detalhes que estão associados com eles (informação redundante removido para fins de ilustração).

group group detail detail detail

country YEAR(hiredate) empid country hiredate

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

UK 2003 5 UK 2003-10-17

6 UK 2003-10-17

UK 2004 7 UK 2004-01-02

9 UK 2004-11-15

Como você pode ver, o grupo britânico de 2003 tem dois associados linhas de detalhes com os funcionários e 5 6; o grupo para o Reino Unido de 2004 também tem duas linhas de detalhes associados com os empregados 7 e 9; a grupo para EUA, 2003 tem uma linha de detalhes associado com empregado 4; o grupo para os EUA de 2004 também tem uma linha de detalhes associado com empregado 8. O resultado final desta consulta tem uma linha representando cada grupo (a não ser filtrados). Portanto, expressões em todas as fases que ocorrem após a fase de grupo atual são um pouco limitado. Todas as expressões processados ​​nas fases subsequentes devem garantir um único valor por grupo. Se você se refere a um elemento da GROUP BY lista (por exemplo, país), você já tem essa garantia, então essa referência é permitido. No entanto, se você quiser referem-se a um elemento que não faz parte da sua GROUP BY lista (por exemplo, empid), deve ser contido dentro de uma função agregada como MAX ou SUM. Isso porque vários valores são possível no elemento dentro de um único grupo, e que a única maneira de garantir que apenas uma vontade ser devolvido é agregar os valores. Para mais detalhes sobre consultas agrupadas, consulte o Capítulo 5, "Agrupamento e de janelas."

4. Filtrar linhas com base na cláusula HA VING

Esta fase também é responsável por filtrar os dados com base num predicado, mas é avaliada depois os dados foram agrupados; Por isso, é avaliada por grupos de grupo e filtros como um todo. Como é usual em T-SQL, o predicado de filtragem pode ser avaliada como verdadeira, falsa ou desconhecida. apenas os grupos para o qual o predicado avaliar para true são devolvidos a partir desta fase. Neste caso, o TENDO

cláusula usa a COUNT predicado (\*)> 1, ou seja, filtro único país e contratar grupos ano que ter mais de um empregado. Se você olhar para o número de linhas que foram associados com cada grupo na etapa anterior, você vai notar que apenas a grupos Reino Unido, 2003 e Reino Unido de 2004 qualificar. Por isso, o resultado desta fase tem os seguintes grupos restantes, mostrado aqui com suas linhas de detalhes associados.

grupo grupo Detalhe Detalhe Detalhe

País Ano (HireDate) HireDate país empid

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

Reino Unido 2003 5 UK 2003-10-17

6 UK 2003-10-17

UK 2004 7 UK 2004/01/02

9 UK 2004-11-15

**Checagem rápida**

■■ Qual é a diferença entre o WHERE e HAVING?

**Quick Check Resposta**

■■ A cláusula WHERE é avaliada antes filas são agrupados e, portanto, é avaliada

por linha. A cláusula HAVING é avaliada após filas são agrupados e, portanto,

é avaliada por grupo.

5. Processo de cláusula SELECT

A quinta fase é a única responsável pelo processamento da cláusula SELECT. O que é interessante sobre o que é o ponto no processamento de consultas lógica onde fica avaliada, quase durar. Isso é interessante, considerando o fato de que a cláusula SELECT aparece em primeiro lugar na consulta. Esta fase compreende duas etapas principais. O primeiro passo é avaliar as expressões em SELECT listar e produzir o resultado atributos. Isto inclui a atribuição de atributos com nomes se eles são derivados a partir de expressões. Lembre-se que se uma consulta é uma consulta agrupados, cada grupo é representada por uma única linha no resultado. Na consulta, dois grupos permanecem após o processamento do filtro que tem. Portanto, este passo gera duas filas. Neste caso, a lista SELECT retornos para cada grupo país e ano a fim de uma fila com os seguintes atributos: país, ANO (HireDate) alias como yearhired, e COUNT (\*) alias como numemployees. A segunda etapa desta fase é aplicável se você indicar a cláusula DISTINCT, em que caso este passo remove duplicados. Lembre-se que T-SQL é baseado na teoria multiset mais do que na teoria dos conjuntos, e, portanto, se duplicatas são possíveis no resultado, é sua responsabilidade para remover aqueles com a cláusula DISTINCT. No caso desta consulta, esta etapa é inaplicável. Aqui está o resultado desta fase na consulta.

country yearhired numemployees

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

UK 2003 2

UK 2004 2

Se você precisar de um lembrete de que a consulta se parece, aqui está ele novamente.

SELECT country, YEAR(hiredate) AS yearhired, COUNT(\*) AS numemployees

FROM HR.Employees

WHERE hiredate >= '20030101'

GROUP BY country, YEAR(hiredate)

HAVING COUNT(\*) > 1

ORDER BY country , yearhired DESC;

A quinta fase retorna um resultado relacional. Portanto, não é garantida a ordem das linhas. No caso desta consulta, há uma cláusula ORDER BY que garante a ordem no resultado, mas isso será discutido quando a fase seguinte é descrita. O que é importante notar é que o resultado da fase que processa a cláusula SELECT ainda é relacional. Além disso, lembre-se que esta fase atribui apelidos de coluna, como yearhired e numemployees. Isto significa que apelidos de coluna recém-criados não são visíveis para cláusulas tratados no anterior fases, como FROM, WHERE, GROUP BY e HAVING. Note-se que um alias criado pela fase SELECT não é ainda visível para outras expressões que aparecem na mesma lista SELECT. Por exemplo, a seguinte consulta não é válida.

SELECT empid, country, YEAR(hiredate) AS yearhired, yearhired - 1 AS prevyear

FROM HR.Employees;

Esta consulta gera o seguinte erro.

Msg 207, Level 16, State 1, Line 1

Invalid column name 'yearhired'.

A razão pela qual isso não é permitido é que, conceitualmente, T-SQL avalia todas as expressões que aparecem na mesma fase de processamento da consulta lógica de uma maneira toda de uma vez. Observe o uso de a palavra *conceitualmente.* SQL Server não será necessariamente processar fisicamente todos expressões no mesmo ponto no tempo, mas tem que produzir um resultado como se fez. Este comportamento é diferente muitas outras linguagens de programação em que as expressões geralmente são avaliados em um-toright esquerda fim, fazendo um resultado produzido em uma expressão visível ao que parece sua certo. Mas T-SQL é diferente.

**Checagem rápida**

1. Por que você não tem permissão para se referir a uma coluna alias definido pelo SELECIONAR

cláusula na cláusula WHERE?

2. Por que você não tem permissão para se referir a uma coluna alias definido pelo SELECIONAR

cláusula na mesma cláusula SELECT?

**Quick Check Resposta**

1. Uma vez que a cláusula WHERE é logicamente avaliada numa fase anterior à uma

que avalia a cláusula SELECT.

2. Porque todas as expressões que aparecem na mesma fase de processamento de consulta lógica

são avaliados conceptualmente no mesmo ponto no tempo.

**Quick Check**

1. Why are you not allowed to refer to a column alias defined by the SELECT

clause in the WHERE clause?

2. Why are you not allowed to refer to a column alias defined by the SELECT

clause in the same SELECT clause?

**Quick Check Answer**

1. Because the WHERE clause is logically evaluated in a phase earlier to the one

that evaluates the SELECT clause.

2. Because all expressions that appear in the same logical query processing phase

are evaluated conceptually at the same point in time.

6. Handle Apresentação Ordenação

A sexta fase é aplicável se a consulta tem uma cláusula ORDER BY. Esta fase é responsável para devolver o resultado em uma ordem de apresentação específica de acordo com as expressões que aparecem na lista ORDER BY. A consulta indica que as linhas do resultado devem ser ordenados pela primeira vez por

país (em ordem ascendente por padrão), e depois por numemployees, descendente, produzindo o seguinte

saída.

country yearhired numemployees

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

UK 2004 2

UK 2003 2

Observe que a cláusula ORDER BY é a primeira e única cláusula que é permitido para se referir a coluna

aliases definidos na cláusula SELECT. Isso porque a cláusula ORDER BY é o único a serem avaliadas após a cláusula SELECT.

Ao contrário nas fases anteriores, onde o resultado foi relacional, a saída desta fase não é relacional porque tem uma ordem garantida. O resultado desta fase é o padrão SQL chama um cursor. Note-se que a utilização do cursor termo aqui é conceitual. T-SQL também suporta um objeto chamado um cursor que é definida com base no resultado de uma consulta, e que permite atraente linhas, uma de cada vez em uma ordem especificada.

Você pode se preocupam com retornando o resultado de uma consulta em uma ordem específica para a apresentação finalidades ou se o chamador tem de consumir o resultado em que a forma por meio de algum cursor mecanismo que vai buscar a uma linha de cada vez. Mas lembre-se que tal tratamento não é relacional. Se você precisa para processar o resultado da consulta em um exemplo relacional forma-para, definir uma expressão de tabela como uma visão baseada na consulta (detalhes mais tarde no Capítulo 4) -o resultado será precisa ser relacional. Além disso, os dados de triagem podem adicionar custo para o processamento de consultas. Se não o fizer se preocupam com a ordem em que as linhas de resultados são retornadas, você pode evitar isso desnecessário custo, não adicionando uma cláusula ORDER BY.

Uma consulta pode especificar o TOP ou compensar-busca opções de filtragem. Se isso acontecer, o mesmo ORDER BY cláusula que normalmente é usado para definir ordenação apresentação também define quais linhas para filtrar estas opções. É importante notar que esse filtro é processado após o Fase Escolher avalia todas as expressões e remove duplicatas (no caso de uma cláusula DISTINCT foi Especificadas). Você pode até considerar o TOP e compensar-fetch filtros como sendo processada em seu próprio número fase 7. A consulta não indica tal filtro e, portanto, esta fase é inaplicáveis ​​neste caso.

Praticar **o processamento de consultas logicas**

Nesta prática, você exercer o seu conhecimento de processamento de consulta lógico. Se você encontrar um problema de completar um exercício, você pode instalar os projectos concluídos a partir da pasta Solution que é fornecido com o conteúdo complementar para este capítulo e lição.

Exercício 1 corrigir um problema com Agrupamento Neste exercício, você é apresentado com uma consulta agrupados isso falhar ao tentar executá-lo. Está equipado com instruções sobre como corrigir a consulta. 1. SSMS aberto e conectar-se à TSQL2012 base de dados exemplo. 2. Digite a seguinte consulta na janela de consulta e executá-lo.

SELECT custid, orderid

FROM Sales.Orders

GROUP BY custid;

A consulta deveria retornar para cada cliente o ID do cliente eo máximo pedir ID para esse cliente, mas em vez disso ele falhar. Tente descobrir por que a consulta falhou e o que precisa ser revisto para que ele iria retornar o resultado desejado. 3. A consulta falhou porque orderid nem aparece na lista GROUP BY nem dentro de uma função de agregação. Existem vários valores NúmeroDoPedido possíveis por cliente. Para corrigir o consulta, é preciso aplicar uma função agregada ao atributo orderid. A tarefa é devolver o valor máximo orderid por cliente. Portanto, a função de agregação deve ser no máximo. Sua consulta deve se parecer com o seguinte.

SELECT custid, MAX(orderid) AS maxorderid

FROM Sales.Orders

GROUP BY custid;

Exercício 2 corrigir um problema com Aliasing Neste exercício, você é apresentado com outra consulta agrupados isso falhar, desta vez por causa de um problema de aliasing. Como no primeiro exercício, que são fornecidos com instruções sobre como corrigir a consulta.

1. Limpar a janela de consulta, digite a seguinte consulta, e executá-lo.

SELECT shipperid, SUM(freight) AS totalfreight

FROM Sales.Orders

WHERE freight > 20000.00

GROUP BY shipperid;

A consulta deveria retornar apenas carregadores para quem o valor total do frete é superior a 20.000, mas em vez disso, retorna um conjunto vazio. Tente identificar o problema em a consulta. 2. Lembre-se que a cláusula de filtragem WHERE é avaliada por linha, não por grupo. O consulta filtra as ordens individuais com um valor de carga maior do que 20.000, e há Nenhum. Para corrigir a consulta, é necessário aplicar o filtro por cada expedidor grupo não por cada encomenda. Você precisa filtrar o total de todos os valores de frete por remetente. Isto pode ser obtida utilizando o filtro que tem. Você tentar corrigir o problema usando a seguinte consulta.

SELECT shipperid, SUM(freight) AS totalfreight

FROM Sales.Orders

GROUP BY shipperid

HAVING totalfreight > 20000.00;

Mas esta consulta também falha. Tente identificar o que deu errado eo que precisa ser revisto para atingir o resultado desejado.

3. O problema agora é que a consulta tenta se referem na cláusula HAVING para o alias totalfreight, o qual é definido na cláusula SELECT. A cláusula HAVING é avaliada antes da cláusula SELECT, e, por conseguinte, o alias coluna não é visível a ele. Para corrigir o problema, você precisa para se referir à soma de expressão (frete) na cláusula HAVING, como segue.

SELECT shipperid, SUM(freight) AS totalfreight

FROM Sales.Orders

GROUP BY shipperid

HAVING SUM(freight) > 20000.00;

Resumo da lição

■■ T-SQL foi concebido como uma linguagem declarativa, onde as instruções são fornecidas em de forma Inglês-like. Portanto, a ordem digitado-in das cláusulas de consulta começa com a cláusula SELECT. ■■ processamento de consultas Logical é a interpretação conceitual da consulta que define a resultado correto, e ao contrário do digitadas ordem das cláusulas de consulta, ele começa pela avaliação cláusula FROM.

■■ Compreender o processamento de consultas lógica é crucial para a correta compreensão da T-SQL.

Examine a lição

Responda as seguintes perguntas para testar seu conhecimento da informação nesta lição. Vocês podem encontrar as respostas para estas perguntas e explicações de por que cada opção de resposta está correta

ou incorreta na secção "Respostas" no final deste capítulo.

1. Qual dos seguintes corretamente representa a ordem de processamento de consulta lógica do

várias cláusulas de consulta?

A. SELECT > FROM > WHERE > GROUP BY > HAVING > ORDER BY

B. FROM > WHERE > GROUP BY > HAVING > SELECT > ORDER BY

C. FROM > WHERE > GROUP BY > HAVING > ORDER BY > SELECT

D. SELECT > ORDER BY > FROM > WHERE > GROUP BY > HAVING

2. Qual das seguintes é inválido? (Escolha tudo o que se aplicam).

A. Referindo-se a um atributo que você agrupar por na cláusula WHERE

B. Referindo-se a uma expressão na cláusula GROUP BY; por exemplo, GROUP BY

ANO (orderdate)

C. Em uma consulta agrupada, referindo-se na lista SELECT para um atributo que não faz parte do

GROUP BY lista e não dentro de uma função de agregação

D. Referindo-se a um alias definido na cláusula SELECT na cláusula HAVING

3. O que é verdade sobre o resultado de uma consulta sem uma cláusula ORDER BY?

A. É relacional, enquanto outros requisitos relacionais são cumpridos.

B. Não pode ter duplicatas.

C. A ordem das linhas na saída é garantida para ser a mesma que a inserção

ordem.

D. A ordem das linhas na saída é garantida para ser a mesma que a do

índice de cluster.

**Cenários de caso**

Nos seguintes cenários, você aplicar o que você aprendeu sobre a consulta T-SQL. Você pode encontrar as respostas a estas perguntas na seção "Respostas" no final deste capítulo.

Cenário caso 1: Importância da Teoria

Você e um colega em sua equipe entrar em uma discussão sobre a importância da compreensão os fundamentos teóricos da T-SQL. Seu colega argumenta que não há nenhum ponto na compreensão as fundações, e que é apenas o suficiente para aprender os aspectos técnicos da T-SQL para ser um bom programador e escrever código correto. Responda as seguintes perguntas feitas pelo seu colega:

1. Você pode dar um exemplo de um elemento da teoria dos conjuntos que podem melhorar a sua compreensão de T-SQL?

2. Você pode explicar porque a compreensão do modelo relacional é importante para as pessoas que escrever o código T-SQL?

Caso Cenário 2: entrevistando para uma posição Código Avaliador

Você está entrevistado para uma posição como um revisor de código para ajudar a melhorar a qualidade do código. Da organização aplicação tem consultas escritas por pessoas não treinadas. As consultas têm numerosos problemas, incluindo erros lógicos. Seu entrevistador coloca uma série de perguntas e pede uma resposta concisa de algumas frases para cada pergunta. Responda as seguintes perguntas dirigidas

pelo seu entrevistador:

1. É importante a utilização de normas padrão quando possível, e por quê?

2. Nós temos muitas consultas que usam posições ordinais na cláusula ORDER BY. É que um mau praticar, e se assim por quê?

3. Se uma consulta não tem uma cláusula ORDER BY, o que é a ordem em que os registros são devolvidos?

4. Você recomendaria colocar uma cláusula DISTINCT em cada consulta?

**Práticas sugeridas**

Para ajudar você a dominar com sucesso os objetivos do exame apresentados neste capítulo, preencha as seguintes tarefas.

Visite T-SQL newsgroups públicos e revisão de código

Para praticar o seu conhecimento do uso de T-SQL de forma relacional, você deve revisar o código amostras escritos por outras pessoas.

■■ **Prática 1** Lista como muitos exemplos que puder para aspectos de codificação T-SQL que não são relacionais.

■■ **Prática 2** Depois de criar a lista na Prática 1, visite o fórum público da Microsoft para T-SQL em *http://social.msdn.microsoft.com/Forums/en/transactsql/threads.* Código de avaliação amostras nos tópicos T-SQL. Tente identificar casos em que elementos não-relacionais são usava; se você encontrar tais casos, identificar o que precisa ser revisto para torná-los relacional.

Descrever o processamento de consultas logicas

Para entender melhor o processamento de consultas lógico, recomendamos que você preencha as seguintes tarefas:

■■ **Prática 1** Criar um documento com uma lista numerada das fases envolvido com lógica processamento de consultas na ordem correta. Fornece um breve parágrafo resumindo o que ocorre em cada passo.

■■ **Prática 2** criar um diagrama de fluxo gráfico que representa o fluxo lógico da consulta fases de processamento, usando uma ferramenta como o Microsoft Visio, Microsoft PowerPoint, ou Microsoft Word.

**Respostas**

Esta seção contém as respostas para as perguntas de revisão lição e soluções para o caso cenários neste capítulo.

Lição 1

1. **Corrigir respostas: B e D**

A. **incorreta:** É importante usar o código padrão.

B. **correto: O** uso de código padrão torna mais fácil portar o código entre plataformas porque menos revisões são necessárias.

C. **incorreto:** Não há nenhuma garantia de que o código padrão será mais eficiente.

D. **correto:** Ao usar o código padrão, você pode se adaptar a um novo ambiente mais facilmente porque os elementos de código padrão semelhante nas diferentes plataformas.

2. **Resposta correta: D**

A. **incorreto:** Uma relação tem um cabeçalho com um conjunto de atributos e duplas da relação têm a mesma posição. Um jogo não tem fim, então as posições ordinais não têm sentido e constituem uma violação do modelo relacional. Você deve consultar a atributos pelo seu nome.

B. **incorreto:** Uma consulta deve retornar uma relação. Uma relação tem um corpo com um conjunto de tuplas. Um conjunto não tem duplicatas. Retornando linhas duplicadas é uma violação damodelo relacional.

C. **incorreto:** não definir uma chave na tabela permite que linhas duplicadas na tabela, e como a resposta para B, que é uma violação do modelo relacional.

D. **correto:** Porque os atributos devem ser identificados pelo nome, assegurando que todos os atributos têm nomes é relacional e, portanto, não é uma violação da relacional modelo.

3. **Resposta correta: B**

A. **incorreta:** T-SQL não é padrão e SQL não é um dialeto no Microsoft SQL Server.

B. **correto:** SQL é padrão e T-SQL é um dialeto no Microsoft SQL Server.

C. **incorreto:** T-SQL não é padrão.

D. **incorreto:** SQL não é um dialeto no Microsoft SQL Server.

Lição 2

1. **Resposta correta: B**

**Incorreto A.:** Lógica de processamento de consulta não começa com a cláusula SELECT.

B. **correto:** o processamento de consultas logicas começa com a cláusula FROM, e depois segue em frente a WHERE, GROUP BY, HAVING, SELECT e ORDER BY.

C. **incorreto:** A cláusula ORDER BY não é avaliado antes de a cláusula SELECT.

**Incorreto D.:** Lógica de processamento de consulta não começa com a cláusula SELECT.

2. **Resposta correta: C e D**

A. **incorreta:** T-SQL permite que você para se referir a um atributo que você agrupar pôr no Cláusula WHERE.

B. **incorreto:** T-SQL permite agrupamento por uma expressão.

C. **correto:** Se a consulta é uma consulta agrupada, em fases processadas após o GROUP BY fase, cada atributo que você se refere, deve figurar ou na lista GROUP BY ou dentro de uma função de agregação.

D. **correto:** Porque a cláusula HAVING é avaliado antes de a cláusula SELECT, referindo para um alias definido na cláusula SELECT na cláusula HAVING é inválida.

3. **Resposta correta: A**

A. **correta:** Uma consulta com uma cláusula ORDER BY não retorna um resultado relacional. Para o resultado a ser relacional, a consulta deve satisfazer uma série de requisitos, incluindo o seguinte: a consulta não deve ter uma cláusula ORDER BY, todos os atributos devem ter nomes, todos os nomes de atributo devem ser exclusivos e duplicatas não devem aparecem no resultado.

B. **incorreto:** Uma consulta sem uma cláusula DISTINCT na cláusula SELECT pode retornar duplicatas.

C. **incorreto:** Uma consulta sem uma cláusula ORDER BY não garante a ordem de linhas na saída.

D. **incorreto:** Uma consulta sem uma cláusula ORDER BY não garante a ordem de linhas na saída.

Case Scenario 1

1. Um dos erros mais comuns que os desenvolvedores T-SQL fazem é assumir que uma consulta sem uma cláusula ORDER BY sempre retorna os dados em um determinado exemplo de injunção, ordem de índice de cluster. Mas se você entender que a teoria em conjunto, um conjunto não tem especial para seus elementos, você sabe que você não deve fazer tais suposições. O único forma como SQL para garantir que as linhas serão devolvidos em uma determinada ordem é para adicionar um ORDER BY. Isso é apenas um dos muitos exemplos de aspectos da T-SQL que podem ser melhor compreendida se você compreender os fundamentos da linguagem.

2. Mesmo que o t-SQL se baseia no modelo relacional, afasta-se a partir dele em um número de maneiras. Mas dá-lhe ferramentas suficientes de que se você entender o modelo relacional, você pode escrever de uma forma relacional. Seguindo o modelo relacional ajuda a escrever código mais corretamente. Aqui estão alguns exemplos:

■■ você não deve confiar em forma de colunas ou linhas.

■■ você sempre deve nomear colunas de resultados.

■■ você deve eliminar duplicatas se eles são possíveis no resultado de sua consulta.

Cenário caso 2

1. É importante usar o código SQL padrão. Dessa forma, tanto o código e conhecimento das pessoas é mais portátil. Especialmente nos casos em que existe padrão e não padrão formulários para um elemento de linguagem, é recomendado usar o formulário padrão.

2. Usando as posições ordinais na cláusula ORDER BY é uma prática ruim. A partir de uma perspectiva relacional, você é suposto para se referir aos atributos pelo nome, e não pela posição ordinal. Além disso, que se a lista de seleção é revista no futuro e que o desenvolvedor se esquece de rever ORDER BY lista em conformidade?

3. Quando a consulta não tem uma cláusula ORDER BY, não há garantias para qualquer ordem particular no resultado. A ordem deve ser considerada arbitrária. Você também observará que o entrevistador usou o *registro* incorreto prazo em vez de *linha.* Você pode querer falar alguma coisa sobre isso, porque o entrevistador pode ter feito isso de propósito para testá-lo.

4. A partir de uma perspectiva relacional pura, isso realmente poderia ser válido, e talvez até recomendado. Mas a partir de uma perspectiva prática, há a chance de que o SQL Server vai tentar remover duplicatas, mesmo quando não há nenhuma, e isso vai incorrer em custos adicionais. Portanto, é recomendável que você adicionar a cláusula DISTINCT apenas quando duplicatas são possíveis no resultado e você não deveria retornar as duplicatas.