



Banco de Dados e T-SQL

Autor:

Tais Cristina Rodrigues

Começando com Banco de Dados e T-SQL

Resumo do SQL Server 2005

O que é uma arquitetura cliente/servidor?

A arquitetura cliente / servidor consiste de cliente, servidor e componentes da rede.

O cliente é o requerente dos serviços, enquanto que o servidor é o prestador de serviços de dados. Você pode projetar uma aplicação na arquitetura cliente / servidor e os usuários podem acessar o aplicativo concebido a partir de computadores clientes.

Para ajudar os usuários acessarem o pedido, é necessário instalar o aplicativo cliente no computador do cliente. Você também precisa instalar o servidor do lado da aplicação no servidor que tem de software do servidor, como a Microsoft SQL Server 2000 e SQL Server 2005.

O usuário pode acessar SQL Server 2005 usando Pipes ou Multi-protocolo, Net-Biblioteca e partilhar Windows NT. SQL Server 2005 concede acesso ao servidor baseado em Windows NT utilizador ou do grupo de segurança identificadores (SIDs).

No Microsoft Access existe apenas compartilhar arquivos que na arquitetura cliente / servidor há partilha de processos. Quando o cliente envia um pedido de um serviço ou dados para o servidor, o servidor responde ao pedido.

Você pode categorizar arquitetura cliente / servidor como aplicações multicamadas baseadas na lógica de como o negócio é distribuído e se o máximo de transformação tem lugar no cliente ou o servidor.

Você pode encontrar vários benefícios na utilização da arquitetura cliente / servidor. Em primeiro lugar, pode-se assegurar da integridade dos dados, pois os dados são armazenados em uma localização central.

Além disso, você pode definir negócio e regras de segurança no servidor e fazer cumprir essas regras equitativamente entre todos os utilizadores.

Além disso, é possível aperfeiçoar o tráfego de rede porque um SGBD realiza eficaz examinando e retorna somente os dados de que o pedido exige.

Portanto, a aplicação de trabalho é partilhada entre o cliente e o servidor, proporcionando assim um melhor desempenho.

Além disso, a arquitetura cliente / servidor permite uma melhor escalabilidade dos processos.

Quanto a escalonamento, você pode fazer escalonamento vertical e horizontal, ou seja, você pode aumentar as configurações do servidor para um melhor tratamento, ou aumentar o número de processadores distribuídos na rede para carregar melhor equilíbrio. Também se pode escalonar mais CPUs e servidores.

Introdução

Um banco de dados é uma coleção de dados relacionados e o dado em um banco é persistente.

Bancos de dados podem ser hierárquicos, relacionais e dimensionais. Esses bancos se diferem uns dos outros e na maneira como o dado é armazenado. O SQL Server é um banco de dados relacional, no qual armazena dado e mantém relacionamentos entre varias entidades ou tabelas.

O que é o SQL Server?

O SQL Server é um banco de dados relacional que pode ser usado para armazenar dados e manter relacionamento entre várias entidades (tabelas). Ele é organizado e acessado de acordo com os relacionamentos entre os valores dos dados. Portanto você pode coletar informação de varias tabelas através dos relacionamentos entre elas. Ajuda a reduzir a redundância de dados e melhora o desempenho.

O SQL Server é designado para assegurar alto desempenho, e um gerenciamento escalável, confiável e seguro. Portanto você pode manter o relacionamento entre os dados através das regras de integridade. O SQL Server usa o padrão ANSI para lidar com as aplicações cliente servidor.

O que é um Banco de dados Relacional?

Num banco de dados relacional os dados são armazenados em forma de tabelas.

Cada tabela é constituída por linhas horizontais chamadas tuplas e verticais, chamadas colunas ou atributos. Você pode identificar cada tabela, dando-lhes nomes únicos, e o banco de dados usa esses nomes para acessar as tabelas.

Os objetos vitais em um banco relacional são entidades, chaves e relacionamentos. Entidades que representam conceitos do mundo real que pode ser claramente identificados, tais como casas, objetos, eventos e pessoas. Atributos identificam ou descrevem uma entidade.

Atributos que descrevem uma entidade são chamados atributos não-principais e os atributos que ajudam na identificação de uma entidade são chamados de chaves. Chaves também podem referenciar outras entidades e, portanto, são utilizados para construir relações entre as entidades. Relacionamentos definem a associação entre as entidades e as vinculam.

As duas entidades em um relacionamento são freqüentemente chamadas entidades pai e filha. Os tipos de relações entre as duas entidades são implementadas em 'um-para-um', 'um-para-muitos', ou 'muitos para muitos'. A relação 'um-para-um' ocorre quando um exemplo de uma entidade-pai está relacionado com apenas um exemplo de uma entidade filha.

Uma relação 'um-para-muitos' ocorre quando um exemplo de uma entidade-pai está relacionado com múltiplas instâncias de uma entidade-filha. Uma relação 'muitos para muitos' ocorre quando um exemplo de uma entidade-pai está relacionado com muitos casos de uma entidade-filha e um exemplo de uma entidade-filha está relacionado com muitos exemplos de uma entidade pai.

Ao nível lógico, não pode haver qualquer número de entidades em um relacionamento fisicamente enquanto você definir as relações entre as duas tabelas.

O que é normalização?

No desenho do seu banco de dados, você precisará desenhar de forma eficiente a estrutura das tabelas do banco de dados para eliminar a redundância de dados e oferecer fácil acesso.

Normalização é o processo de eliminar a duplicação de dados através da definição de chaves, e novos relacionamentos e entidades. Durante a normalização, você precisará dividir tabelas em tabelas menores possíveis para que cada tabela descreva apenas um tipo de entidade, como uma pessoa, lugar, o cliente encomenda, produto ou item.

Existem seis níveis de normalização, mas a maioria dos bancos de dados está normalizada até a terceira forma normal (3NF). A tabela a seguir descreve normalização até a terceira forma normal e da forma normal Boyce-Codd

1ª Forma Normal:

Uma tabela relacional está na primeira forma normal se:

Tem uma chave primária.

Cada coluna é atômica. Ela contém apenas um valor que não pode ser quebrado mais em pedaços.

Não existe nenhum grupo repetição de colunas.

2ª Forma Normal:

Uma tabela relacional está na segunda forma normal se:

É na 1NF.

Cada coluna não-chave é completamente funcional dependente da chave primária.

3ª Forma Normal:

Uma tabela relacional está na terceira forma normal se:

É na 2NF.

Cada coluna não-chave é completamente funcional dependente da chave primária. Nenhuma coluna não-chave é dependente da outra não-chave coluna.

Forma Normal Boyce - Codd

Uma versão mais precisa do 3NF é a forma normal Boyce Codd que se baseia no conceito de determinantes. É suficiente que suas tabelas relacionais estão em 3NF porque garante que quase nenhuma redundância permanece em seu banco de dados.

Processo de Normalização

Você é o administrador do banco de dados pela Adventure Works.

Uma nova colaboradora, Linda, diz que lhe foi designada a tarefa de desenvolver o desenho de banco de dados. Ela está um pouco hesitante em enviar o seu desenho e pede pela sua ajuda para rever a estrutura das tabelas

Linda mostra a estruturas das tabelas. Ela tem as seguintes Entidades: Title (título), Author (Autor), Author Biography (Biografia Autor), ISBN, Price (Preço), Subject (Assunto), Numbers of Page (Nºs de Pagina), Publisher (Editor), Description (Descrição), Review (Resumo) e Reviewer Name (Nome do Escritor).

Qual a sua resposta a Linda?

A tabela tem dados redundantes e precisa ser normalizada.

Você identifica três problemas com a tabela criada, e chega às seguintes conclusões com a Linda:

A tabela não tem armazenamento eficiente. Por exemplo, se Lary Zhang e Laura Owen são escritores prolíficos e tem 500 livros escritos, então os seus nomes se repetiriam em 500 linhas. Isto são 12.500 bytes de espaço gasto desnecessariamente.

O desenho não tem integridade de dados. Se alguém tivesse que cadastrar os nomes 500 vezes é muito provável que um deles seja perdido pelo menos uma vez. O dado então se torna inconsistente e na procura de livros por autor vai faltar o resultado que está faltando.

A Tabela não é confiável. Você está limitado a um ou dois autores, e não vai prover os livros escritos por três ou mais autores. Além disso se todos os dados estão presentes em uma tabela e a entrada dos dados aumentarem rapidamente, então o tamanho da tabela também vai aumentar. O arquivo grande vai deixar lento o processo de busca do usuário.

Você faz esta sugestão a Linda e olha o desenho de novo.

A Linda volta e diz: Ok eu mudei a tabela, você acha que está legal agora?

Não, Linda eu não posso aprovar ainda?

Como não? Por favor, diga quais mudanças eu tenho que fazer.

Faça tabelas separadas para autor e livro

Linda volta: As minhas tabelas estão corretas?

Você responde: Ainda não é o bastante. Você precisa minimizar os dados redundantes. Também algumas entidades essenciais estão faltando. Pense nisso. Faça tabelas separadas de autores e assuntos

Linda: As tabelas estão corretas?

Você responde: Não é o bastante. Você precisa minimizar os dados redundantes.

Faça tabelas separadas para livro e assunto

Linda: As tabelas estão corretas?

Você responde: Não é o bastante. Você precisa minimizar os dados redundantes.

Tente tabelas separadas por autor, livro e assunto.

Linda: As tabelas estão corretas?

Agora sim, você pode enviar o desenho.

Linda diz: eu tenho boa notícias! A sua idéia foi aprovada. Obrigada. Você pode me ajudar a decidir as chaves primarias de cada tabela?

Você sugere a Linda: Author ID, e ISBN

Linda diz: Parece que está certo. Eu preciso definir o relacionamento entre livro e autor.

Como você auxilia Linda?

R: Muitos para muitos

Você ainda explica:

As tabelas livro e autor têm relacionamento muitos para muitos porque um livro pode ter mais de um autor e um autor pode escrever mais de um livro.

Linda pergunta: Agora, como eu represento um relacionamento muitos para muitos em um banco de dados?

Você sugere:*

Depois de ter criado as tabelas Linda volta e diz:

Olhe as minhas tabelas que define o relacionamento Livro-Autor e o relacionamento Livro-Assunto. Minha pergunta é: o SQL Server 2005 me permite inserir uma linha no Livro-Assunto se o livro e o assunto já existir na s tabelas livro e assunto? Eu vou ser capaz evitar e eliminação de livros da tabela de livros que tem filhos na tabela livro - assunto ou no livro-autor?

Sim, Linda, O SQL dá suporte à integridade desta referencia.

Linda diz: ' Eu disse que o meu banco de dados está na 1ª Forma normal. Contudo esta é uma forma muito básica de normalização e me ajudou a eliminar redundâncias nas linhas. Agora eu tenho que considerar a redundância de colunas. Na tabela de livro eu não repito o mesmo dado em duas colunas e cada coluna. Contudo na coluna editora eu tenho uma repetição em mais de uma linha. Qual a sua opinião?

Que opinião você dá a Linda?

Sim, Linda isso viola a 2FN Vamos trabalhar de novo na tabela.

Linda diz: eu concordo com você. Ela passa a levantar outra preocupação. Existe uma chance de erros de ortografia ocorrer desde que o usuário precisa digitar o nome do editor a cada vez. Este tipo de abordagem é também ineficiente no que se refere ao uso do armazenamento e, quanto mais dados a tabela tem, mais ações de entrada e saída são necessárias para mostrar a tabela, resultando em consultas mais lentas.

Qual a sua sugestão pra Linda?

Vamos dividir a tabela Livro de novo e Criar uma nova tabela de Editor

Eu dividi a tabela livros fiz uma nova tabela de Editor. Olhe. Agora qual o relacionamento entre a tabela Livro e a tabela Editor?

Você diz a ela: relacionamento um para muitos.

Linda diz: Sim você tem razão. Um livro só pode ser publicado por uma editora, pelo menos nesse modelo que estamos testando, e uma editora pode publicar vários livros. Quando nós temos um relacionamento um para muitos é uma boa idéia colocar a chave estrangeira na tabela muitos e apontar para a chave primaria na tabela um.

Linda continua: Aqui está a nova tabela Livro. Porque a tabela Livro tem o 'muitos' do relacionamento um para muitos, nós colocamos a chave primaria do 'um' na coluna Publisher_id como chave estrangeira.

Mas você vê que agora encontramos outro problema?

Você diz: Sim Parece que há um problema de chave. VC não pode ter um dado numa tabela com uma chave composta que não liga com o 'muitos' da chave. E se você precisasse adicionar uma tabela de resumo onde os usuários pudessem enviar o resumo dos livros como uma tabela de resumo?

Você ainda explica: Nesta tabela, a Combinação reviewer_id e ISBN forma uma chave - primaria composta e assegura que nenhum escritor escreve mais de um resumo para o mesmo livro. Contudo nome do revisor está relacionado com a ID e não com o ISBN e isso, portanto viola a 2FN.

O que você sugere para Linda?

Faça uma nova tabela Reviewer_id

Linda: Sim, eu poderia fazer isto. Minha tabela está agora em conformidade com a 2FN. E se eu tivesse um dado que não é totalmente dependente da chave primaria, mas dependente de outro valor na tabela. Por exemplo, na tabela de Escritor (reviewer) a rua e a cidade são dependentes do CEP e não da ID. O que eu posso fazer para deixar a tabela conforme a 3FN?

O que você sugere a Linda?

Faça uma nova tabela de CEP (PostalCode)

Depois de criar a tabela de CEP, Linda diz: Há um problema. Esta tabela viola a 2FN porque a rua e a cidade serão verticalmente redundante e precisam ser quebradas em tabelas separadas. Novamente terá de ser outra tabela que a tabela cidade vai referenciar como uma chave estrangeira. Você aconselharia prosseguir para a 3FN?

O que você aconselha a Linda?

Vamos fazer conforme a 3FN

Linda diz: Não. Eu não acho necessário, se continuarmos, os usuários terão que criar consultas com join o que iria diminuir o desempenho do banco.

Não Linda, vamos parar por aqui. Vá em frente e envie a tabela.

Linda envia o banco de dados completo para a Adventure Works
Parabéns!

Componentes de uma tabela do Banco de dados SQL Server

Antes de converter um banco de dados lógico em um físico, você precisa identificar as características que o banco de dados suporta. O desenho físico de um banco de dados varia baseando-se nas características e nos objetos do banco. O SQL Server suporta vários objetos de dados. A tabela a seguir mostra um resumo sobre os objetos que o SQL Server suporta:

Tabelas -> uma tabela consiste de linhas e colunas. É usada para armazenar dados em banco de dados relacional. Cada tabela armazena informações sobre os tipos de objetos modelados pelo banco de dados.

Views -> Uma View ajuda a encapsular uma consulta em uma entidade relacional e facilita a visão de um dado de uma ou mais tabelas em um banco de dados.

Restrições -> Uma restrição define regras a respeito de valores permitidos em colunas e é um mecanismo que executa a integridade de dados

Índices -> Num banco de dados relacional um índice oferece rápido acesso ao dado de uma tabela baseado no valor de chave. Índices também podem reforçar a unicidade nas linhas de uma tabela. SQL Server suporta índices clustered e non-clustered. A chave primária de uma tabela é automaticamente indexada. Na pesquisa de um texto completo, um índice de texto completo armazena informação sobre palavras significantes e a localização delas em uma dada coluna.

Funções -> Uma função é um pedaço de código que opera como uma única unidade lógica. Uma função pode retornar tanto uma tabela quanto um valor escalar. Qualquer código que deve executar a lógica incorporada em uma função pode chamar a função, em vez de repetir a função lógica.

Procedimentos (Procedures) -> Um procedimento é uma coleção de padrões T-SQL pré-compilados que são executados e retornam os parâmetros do usuário. Você pode usar uma procedure temporária ou permanente. E pode também executar uma procedure automaticamente sempre que o SQL Server for iniciado.

Gatilho (Trigger) -> Um gatilho é uma stored procedure (procedimento armazenado) que é executada quando um dado em uma tabela específica é modificado. Você pode criar triggers para impor integridade referencial entre os dados relacionados logicamente em diferentes tabelas.

Schemas -> Para organizar objetos de dados no espaço dos nomes, SQL Server 2005 introduz um novo recurso chamado schemas. Schemas ajudá-lo a gerenciar objeto propriedade e segurança. Segurança no SQL Server é organizado hierarquicamente. Você pode usar esquemas objeto permissões para gerir eficazmente. Você pode exigir que o grupo de segurança comum, permissões e conceder permissões sobre o esquema, em vez dos objetos individuais.

Como criar esquemas (schemas)?

A seguir a sintaxe parcial para criar um esquema:

```
CREATE SCHEMA schema_name_clause  
[<schema_element>[, ...n] ]
```

Você pode substituir o *schema_name_clause* pelas seguintes opções:

O nome do esquema

O nome e um dono específico usando a palavra chave AUTHORIZATION.

Um esquema sem nome com um dono específico usando a palavra chave AUTHORIZATION. Tais esquemas são definidos por compatibilidade interna.

Você também pode substituir *<schema_element>* por um *create table*, *create view* ou *Grant*. Portanto você pode criar um esquema, definir como objetos e atribuir permissões em uma única declaração, como mostrado no exemplo a seguir.

```
CREATE SCHEMA Marketing Data
```

```
CREATE TABLE Products (ID INT, Name NVARCHAR (10))
```

Vantagens de Esquemas

Você pode gerenciar as permissões dos objetos do banco de dados facilmente usando os esquemas. Você pode atribuir titularidade de um esquema para muitos usuários se eles pertencerem ao mesmo grupo ou tem um papel no do banco de dados na aplicação. Você pode também atribuir vários usuários ao mesmo esquema para uma resolução em comum. Porque o esquema é separado do usuário, você pode remover o nome ou o papel do usuário sem mudar o nome dos objetos no banco de dados criados pelo usuário.

Acrescentando, você precisa armazenar objetos compartilhados no esquema dbo. Ao invés disso, você pode armazenar objetos compartilhados num esquema padrão comum.

Esquemas requerem um nome totalmente qualificado.

No SQL Server 2005 cada objeto tem um nome totalmente qualificado, no qual toma a forma *<Server>.<Database>.<Schema>.<Object>*.

Se você não menciona os nomes do servidor e dos componentes de forma qualificada, o servidor corrente e os bancos de dados assumem. A menos que você explicitamente especificar um nome padrão para o usuário, sem reservas objeto nomes que existem por padrão em todos os dados são assumidos no dbo schema.

Como usar identificadores limitados

Você precisa usar identificadores limitados quando você usa palavras reservadas para nome de objeto e parte do nome, ou caracteres que não estão listados como identificadores qualificados. Sempre que qualquer identificador não está em conformidade com as regras de identificadores, você precisará usar os delimitados identificadores. Os dois tipos de identificadores delimitados que são usados em T-SQL são:

Colchetes, que são delimitados por usar parênteses ([]), como mostrado no exemplo a seguir.

```
SELECT * FROM ([Blanks in Table Name])
```

O identificador citado é válido apenas quando a opção QUOTED_IDENTIFIER é configurada como ON(ativado). Por padrão, o Microsoft Object Vinculação Embedded Database (OLE DB) Provider para SQL Server e do SQL Server Object Database Connectivity (ODBC) driver definem a opção de QUOTED_IDENTIFIER ON. Para garantir que a opção QUOTED_IDENTIFIER é executada, é sempre preferível utilizar os colchetes ([]).

Componentes de uma tabela de banco de dados SQL Server

Quando se armazena um dado em uma tabela, você precisa identificar os componentes da tabela e verificar que o dado que você está tentando armazenar é relevante a esses componentes.

Tabelas e Entidades: Entidades representam o conceito de mundo real como lugares, objetos, ou pessoas. No modelo do banco de dados relacional, as tabelas representam uma única entidade. Dados de uma entidade, quando dispostos em uma linha-coluna formato, formam uma tabela.

Campos, Atributos e Colunas: atributos são as características de uma entidade. No modelo do banco de dados relacional, os atributos correspondem a colunas ou campos do banco de dados.

Tipos de dados: Para definir os valores dos dados que pode ser armazenado em uma coluna, você precisa usar tipos de dados. O SQL Server fornece um conjunto de tipos de dados. No entanto, você também pode criar definido pelo usuário, tipos de dados.

Chaves- primarias e chaves estrangeiras: Chaves- primarias são um atributo ou um conjunto de atributos que identifica unicamente cada instancia de uma entidade. Chaves Estrangeiras identificam o lado 'filho' do relacionamento.

Chechagem de restrições (Check Constraints): Para limitar os possíveis valores de entrada em uma coluna, você precisa usar a checagem de restrições. Os valores da restrição devem ser avaliados para uma expressão booleana.

Classificando os objetos do banco de dados

Classificar os conceitos, o mais rapidamente possível para os seus associados guias clicando no segmento adequado. Você também pode utilizar os atalhos do teclado pressionando 1 para o balde esquerdo, 2 para o balde do meio, e 3 para balde direito.

Questões:

Como administrador de banco de dados da Adventure Works, você tem que normalizar o seu banco de dados relacional, no qual foi normalizado da 2FN para a 3FN. Quais das seguintes características significam estar na FN?

- *Não há grupo de colunas repetido
- *Cada coluna não chave é funcionalmente dependente da chave primaria.
- *Nenhuma coluna não chave é dependente de outra coluna não chave X
- *Cada coluna contem somente um valor que não pode ser quebrado em mais pedaços.

A Adventure Works tem o SQL Server instalado e você, como DBA, quer usar esquemas para a segurança do seu banco de dados. Quais dos seguintes fatos são verdadeiros sobre esquemas?

- *Você pode atribuir titularidade de um esquema a múltiplos usuários X
- *Você pode armazenar objetos compartilhados somente no esquema dbo.
- *Você pode usar esquemas para gerenciar titularidade e segurança de objetos. X
- *Você pode armazenar objetos compartilhados em um esquema padrão comum. X
- *Você não pode atribuir o mesmo esquema default a múltiplos usuários.
- *Você não pode separar o schema do usuário.

Você é o DBA da Adventure Works, e você armazena os dados da companhia no banco de dados SQL Server. Quais dos seguintes componentes do SQL Server são verdadeiros?

- *Os valores das expressões da checagem de restrição podem não avaliar para uma expressão booleana.
- *Características de uma entidade são chamadas de atributos. X
- *SQL Server não oferece tipos de dados definidos ao usuário.
- *Atributos correspondem a colunas ou campos de uma tabela em um banco de dados relacional. X
- *As restrições limitam os possíveis valores que podem ser inseridos em uma coluna. X
- *Chave estrangeira é uma atributo ou um conjunto de atributos que unicamente identificam uma instancia de cada entidade.

Resumo da Linguagem SQL

A linguagem T-SQL é comumente usada para escrever consultas e não está limitada a DML, DDL, e DCL. Ela consiste em vários elementos sintáticos como operadores, funções, variáveis, expressões e condições.

Os elementos sintáticos ajudam a lidar com as transações e consultas do banco.

Os scripts são consultas que podem ser armazenadas em um arquivo que pode ser usado no Query Analyzer ou no Management Studio. Essas ferramentas executam os scripts e mostram o resultado desejado. Os scripts são tipicamente guardados como cópia dos passos que são usados para criar e popular os bancos de dados ou como backup.

No SQL Server 2005 além das suas ferramentas você pode executar uma consulta pelo Access, pelo Excel e outras aplicações através do ODBC, do OLEDB e SQL Native Access Client.

ANSI x T-SQL

SQL é um grupo de linguagem de programação usada para consulta de dados relacionados. grupos baseados em linguagem são diferente de linguagens procedurais. Num grupo baseado em linguagem, você pode definir o grupo de dados onde você quer operar, e a operação atômica que você precisa aplicar para cada elemento do grupo. O processo do banco de dados determina o melhor método para coletar e aplicar o dado em suas operações. Contudo numa linguagem procedural, depois de definir o grupo de dados, você precisa planejar passos para manipular quais dados você quer coletar e atualizar os dados.

Vantagens da Linguagem baseada sobre a Linguagem procedural

A linguagem baseada é melhor do que a procedural por duas razões principais.

Você pode executar consultas na linguagem baseada em menos linhas de código do que na procedural.

Você pode facilmente lidar com problemas e é mais rápido em termos de processamento de operações de entrada e saída.

SQL tem sido uma linguagem de banco de dados dominante por mais de três décadas. E tem sido padronizado com a introdução de muitos SGBDs que contem a Sintaxe SQL e os dialetos SQL.

Portanto, Declarações SQL que executam com eficiência num SGBD em particular pode falhar quando executado em um produto diferente.

ANSI e T-SQL

ANSI define os padrões SQL. Conseqüentemente, vendedores que se deparam com problemas de compatibilidade com os produtos SGBD podem implementar padrões ANSI SQL para interoperabilidade. Você precisa escrever os scripts baseados nos padrões ANSI para garantir a compatibilidade e portabilidade do seu banco de dados. O ultimo padrão ANSI é o SQL 99 ou SQL 2003. Nenhum vendedor tem implementado o padrão completo. Antes disso, foram o padrão SQL 92. SQL Server 2000 e SQL Server 2005 são entrada de nível compatível com os padrões ANSI SQL 92. SQL Server 2005 também implementa muitas características do SQL 99. Transact-SQL (T-SQL) é a implementação da linguagem SQL pela Microsoft. É uma ferramenta do padrão ANSI SQL.

Classificação das Declarações SQL

Existem três grandes categorias do SQL 92, que ainda são comumente usados em todos os bancos de dados.

Data Definition language (DDL) - Linguagem de Definição de Dados: usada para a estrutura ou o esquema do banco de dados.

A declaração CREATE cria objetos.

A declaração ALTER altera a estrutura dos objetos no banco de dados.

A declaração DROP apaga os objetos do banco de dados.

Data Control Language (DCL) Linguagem de controle de dados: Usada para controlar o acesso aos dados e garantir a segurança. A seguir algumas declarações importantes:

GRANT - usada para atribuir privilégios de acesso aos objetos do banco para um usuário.

REVOKE - usada para retirar privilégios de acesso usando a declaração GRANT

DENY - usada para proibir determinados usuários de desempenhar certas tarefas.

Data Manipulation Language (DML): usada para gerenciar os dados através no esquema de objetos.

A seguir algumas importantes declarações:

INSERT - Para inserir dados na tabela

UPDATE - para modificar um dado já existente na tabela

DELETE - para deletar o que foi gravado na tabela

SELECT - para selecionar o que está gravado na tabela

Nota: SQL 2003 contém sete novas classes que substituirá o DDL, DML e DCL. Essas classes são declarações de conexão SQL, declarações de controle SQL, declarações de dados SQL, declarações de diagnóstico SQL, declarações de esquema SQL, declarações de sessão SQL e declarações de transação SQL.

Exemplo DDL

```
USE AdventureWorks;
GO
ALTER TABLE Production.TransactionHistoryArchive
DROP CONSTRAINT
PK_TransactionHistoryArchive_TransactionID;
GO
```

Exemplo DCL

```
USE AdventureWorks
GO
GRANT SELECT
ON Production.Product
TO public
GO
```

Exemplo DML

```
USE AdventureWorks
GO
SELECT*
FROM Production.Product
ORDER BY Name ASC
USE AdventureWorks
GO
INSERT INTO Production.UnitMeasure
VALUES (N'F2', N'Square Feet', GETDATE());
GO
```

Classificando as declarações SQL

Classifique as declarações, o mais rapidamente possível para os seus associados guias clicando no segmento adequado. Você também pode utilizar os atalhos do teclado pressionando 1 para a esquerda balde, 2 para o meio balde, e 3 para a direita balde.

Questões

Você é o DBA da Adventure Works e a sua organização mantém os detalhes do seu projeto nos bancos do SQL Server 2005. Para recuperar dados dos bancos, a linguagem baseada na programação. Identifique as declarações que são verdadeiras sobre esta linguagem.

* Na linguagem baseada, depois de definir o conjunto de dados você precisa gravar um serie de passos de maneira que você possa coletar e modificar os dados.

* Na linguagem baseada, você pode definir a operação atômica que precisa para aplicar a cada elemento do conjunto. X

* Na linguagem baseada, os processos são similares as da linguagem procedural.

- * Na linguagem baseada, você pode executar consultas com menos linhas de código do que a linguagem procedural. **X**
- * Na linguagem baseada, o processo do banco de dados determina o melhor método para coletar e aplicar os dados para a sua operação. **X**
- * Na linguagem baseada, operações de entrada e saída são mais lentas comparadas à linguagem procedural.

Sua companhia decidiu usar o SQL Server 2005 para armazenar e recuperar dados usando declarações DML. Qual das seguintes declarações é uma declaração DML?

- *ALTER
- *GRANT
- *UPDATE **X**
- *DROP

Sua companhia decidiu usar o SQL Server 2005 para armazenar e recuperar dados usando declarações DDL. Qual das seguintes declarações definem a função das declarações DDL?

- * Declarações DDL são usadas para gerenciar dados dentro da estrutura do banco de dados.
- * Declarações DDL dá privilégios de acesso aos usuários.
- * Declarações DDL são usadas para definir a estrutura do banco de dados. **X**
- * Declarações DDL são usadas para controlar dados.

Sintaxe dos Elementos T-SQL

T-SQL é comumente usado para escrever as consultas e não se limita apenas ao DML, DDL, DCL. T-SQL sintática consiste em vários elementos, como os operadores, funções, variáveis, expressões, e declarações condicionais. Estes elementos ajudam a lidar transações e consultas.

Operadores Aritméticos

Desempenham operações matemáticas em duas expressões com tipos de dados que pertencem a categoria de dados de tipo numérico. Os operadores são:

- + Adição
- Subtração
- * Multiplicação
- / Divisão
- % Porcentagem (modulo)

Operadores de Comparação

São usados para comparar duas expressões. Esses operadores podem ser usados em todas as expressões exceto text, ntext e image. A seguir, os operadores de comparação suportados pelo T-SQL.

- =: Igual a
- >: maior
- <: menor
- >=: maior ou igual
- <=: menor ou igual
- <>: Diferente

Nota: Os operadores !=, !<, !> devem ser evitados porque não são do padrão ANSI SQL – 92.

Operador de Concatenação de String

É um sinal de + que permite adicionar strings.

Operadores Lógicos

Testam condições de verdadeiro ou falso retornando um valor booleano, true ou false. A seguir, o operadores lógicos oferecidos pelo T-SQL.

ALL – Retorna True se todas as expressões da condição forem verdadeiras.

AND – Retorna True se ambas as expressões forem verdadeiras.

ANY – Retorna True se alguma expressão da condição for verdadeira.

BETWEEN – Retorna True se o operando estiver dentro de um determinado intervalo.

EXISTS – Retorna True se a subconsulta contem pelo menos uma linha.

IN – Retorna True se o operando tiver uma ligação.

NOT Retorna o valor oposto de outro operador booleano.

OR – Retorna True se uma ou outra expressão booleana for verdadeira.

SOME – Retorna True se algum do conjunto de comparação for verdadeira.

Nota: O + e - podem ser usados em operações com os valores datetime e smalldatetime

Os operadores são arranjados em uma ordem hierárquica e têm uma ordem de precedência. Pode-se escrever uma expressão complexa que tem vários operadores. Você vai achar que o operador de precedência determina a seqüência em que as operações são realizadas. A ordem de execução pode afetar significativamente o valor resultante. Você também pode escrever uma expressão com operadores que tenham o mesmo nível de precedência. Você vai achar que os operadores são avaliados da esquerda para a direita, baseada na sua posição na expressão. A tabela a seguir mostra a ordem de precedência dos operadores.

Níveis de Operadores

1	+ (Positive), - (Negativo), ~ (Bitwise NOT)
2	* (Múltiplo), / (Division), % (Modulo)
3	+ (Add), + Concatenate, - (Subtract), & (Bitwise AND)
4	=, >, <, >=, <=, <>, !=, !>, !< (Comparison operators)
5	^ (Bitwise Exclusive OR), (Bitwise OR)
6	NOT
7	AND
8	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
9	= (Atribuição)

Ordem de Precedência de Operadores

Quando uma expressão complexa tem vários operadores, o operador precedente determina a seqüência em que as operações são realizadas. A ordem de execução pode afetar significativamente o valor resultante.

Quando dois operadores em uma expressão têm o mesmo operador precedência nível, que são avaliadas da esquerda para a direita, baseada na sua posição na expressão. Por exemplo, na expressão utilizada no SET seguinte declaração, a subtração operador é avaliado antes da adição operador.

```
DECLARE @My Number int
```

```
SET @MyNumber = 2 * (4 + (5 - 3))
```

```
-- Evaluates to 2 * (4 + 2) which further evaluates to 2 * 6, and
```

```
-- yields an expression result of 12.
```

```
Print @MyNumber
```

```
-- First evaluated NOT (@MyNumber > 12 AND @MyNumber=0) which gives true
```

```
--and then evaluates @myNumber=12 which is also true. Hence the condition is met
```

```
IF @MyNumber=12 AND NOT (@MyNumber > 12 AND @MyNumber=0)
```

```
Begin
```

```
    Print 'First Condition Met'
```

```
END
```

```

ELSE
Begin
    Print 'First Condition Failed'
END
-- First evaluated @MyNumber =0 which is false AND then NOT @mynumber>12 which is true
--and then evaluates @myNumber=12 which is also true. As all have an And condition in between,
the final result is false
--This shows the significance of braces.
IF @MyNumber=12 AND NOT @MyNumber > 12 AND @MyNumber=0
Begin
    Print 'Second Condition Met'
END
ELSE
Begin
    Print 'Second Condition Failed'
END

```

Se a expressão tiver parênteses aninhados, o mais interno da expressão é avaliado primeiro.

Funções Nativas do SQL Server

Funções são elementos de sintaxe que aceitam zero, um ou mais valores de entrada e retornam um valor escalar ou um conjunto tabular de valores. No SQL Server 2005, funções são classificadas dentro de determinísticas e não determinísticas funções. Funções determinísticas sempre retornam o mesmo valor para um grupo de valores de entrada específico. Funções não determinísticas retornam diferentes valores quando são chamadas repetidamente com o mesmo grupo de valores de entrada. Funções SQL Server podem ser caracterizadas baseando-se no tipo de entrada que a função aceita.

Você pode usar para executar operações em caracteres e strings binárias.

SUBSTRING: Recupera uma parte do input string.

UPPER e LOWER: Converte um personagem dados minúsculas para maiúsculas e vice-versa.

STUFF: Apaga um determinado comprimento de caracteres e insere outro conjunto de caracteres em um determinado local.

LTRIM: Remove os principais espaços de uma string.

RTRIM: Remove os espaços a partir de uma string.

REPLACE: Substitui todas as ocorrências de uma determinada string em uma determinada string com outra string.

SOUNDEX: Devolução de quatro caracteres (SOUNDEX) código para avaliar a similaridade das duas cordas.

DIFERENÇA: Retorna um valor inteiro que indica a diferença entre os valores de dois caracteres

SOUNDEX expressões.

Exemplos:

```
USE AdventureWorks
```

```
GO
```

```
SELECT SUBSTRING (firstname,1,6) as 'First Name'
```

```
FROM Person.Contact
```

```
WHERE ContactID=17870
```

```
Christ
```

```
USE AdventureWorks
```

```
SELECT UPPER (firstname) as 'First Name'
```

```
FROM Person.Contact
```

```
WHERE ContactID=5276
```

```
HUNTER
```

```
USE AdventureWorks
```

```
SELECT REVERSE (firstname) as 'First Name'
```

```
FROM Person.Contact
```

```
WHERE ContactID=902
```

```
remoH
```

Funções DATETIME

Efetua uma operação em uma data e hora de entrada e retornar um valor string, numérico, ou valor data e hora.

GETDATE: função não determinística que retorna o atual sistema de data e hora.

DATEDIFF: Determinística função que retorna o número de data e hora que atravessaram fronteiras entre duas datas especificadas.

DATEADD: função determinística que retorna um valor datetime novo baseado em adicionar um intervalo para a data especificada.

DATEPART: função determinística exceto quando usado como DATEPART (dw, data). Dw, o dia datepart, depende do valor fixado pelo SET DATEFIRST, que define o primeiro dia da semana. Ele retorna um inteiro que representa a datepart especificado.

DATENAME: função não determinística que retorna uma seqüência de caracteres representando o especificado datepart. função não determinística que retorna uma seqüência de caracteres representando o especificado datepart.

DIA, MÊS, e ANO: Determinística funções que retorna um inteiro que representa o dia, mês e ano parte respectivamente.

GETUTCDATE: função não determinística que retorna o valor datetime representando o atual momento UTC (Coordinated Universal Time ou Tempo Médio de Greenwich).

Exemplos:

```
USE AdventureWorks
```

```
SELECT DATEDIFF (month, OrderDate, GETDATE ()) AS 'Number Of Months'
```

```
FROM Sales.SalesOrderHeader
```

```
WHERE SalesOrderId=43660
```

```
USE AdventureWorks
```

```
SELECT DATEADD (year, 3, hiredate) AS 'Date of 3 years Completion'
```

```
FROM HumanResources.Employee
```

```
WHERE EmployeeID=30
```

```
USE AdventureWorks
```

```
SELECT DATEPART (day, Birthdate) AS 'Birthday'
```

```
FROM HumanResources.Employee
```

```
WHERE EmployeeID=20
```

Funções Matemáticas

Realizar operações matemáticas em expressões numéricas e retornar o resultado da operação. Eles também incluem funções trigonométricas.

ABS: Retorna o absoluto, valor positivo de um número.

COS e SIN: Retorna o cosseno e seno do dado ângulo, respectivamente.

POWER: Retorna o valor de uma determinada expressão à potência especificada.

REDONDA: Retorna uma expressão numérica, arredondado para o comprimento especificado.

PISO e CEILING: Retorna o maior inteiro menor ou igual a, e o menor inteiro maior ou igual ao dado numérico expressão, respectivamente.

SQRT e SQUARE: Retorna a raiz quadrada e quadrada de um determinado número, respectivamente.

Exemplos

```
SELECT ABS (-1.0), ABS (0.0), ABS (1.0)
```

1.0

.0

1.0

```
SELECT CEILING (156.234), CEILING (-34.564)
```

157, -34

```
SELECT POWER(3,4), POWER(5,2)
```

81, 25

Funções de Conversão

Convertem uma expressão de um tipo de dados para outro tipo de dados.

As funções de conversão são CAST e CONVERT. As funções CAST e CONVERT explicitamente converter uma expressão de um tipo de dados para outro tipo de dados. A diferença entre CAST e CONVERT é que CAST é ANSI CONVERT que não é ANSI. Além disso, CONVERT tem estilo parâmetros opcionais.

Exemplos:

```
SELECT CAST ('19700926' AS SMALLDATETIME)
```

```
SELECT CONVERT (SMALLDATETIME,'19700926')
```

```
SELECT CONVERT (VARCHAR (MAX), GETDATE (), 3)
```

```
SELECT CONVERT (VARCHAR (MAX), GETDATE (), 103)
```

Funções de Sistema

Executam operações em funções e retornar informações sobre valores, objetos, e as configurações.

HOST_NAME: função não determinística que retorna o nome do trabalho e é bom para fins de auditoria.

XACT_STATE: função não determinística que os relatórios da operação estado de uma sessão, indicando se a sessão tem uma transação ativa, e se a operação é susceptível de ser cometidos.

SYSTEM_USER: função não determinística que permite que um sistema de valor fornecido para o atual login para ser inserido em uma tabela quando nenhum valor padrão é especificado.

CURRENT_TIMESTAMP: função não determinística que retorna a data e a hora atuais.

CURRENT_USER: função não determinística que retorna o nome do usuário atual como um sysname.

DATALENGTH: Determinística função que retorna o número de bytes usados para representar qualquer expressão.

SUSER_SNAME: função não determinística que retorna o login identificação nome de um usuário de segurança do número de identificação.

Exemplos:

```
SELECT 'The current user is: ' + convert (char (30), CURRENT_USER)
```

Funções Metadados

Retornam informação sobre o banco de dados e os seus objetos.

DB_NAME: Retorna o nome atual database.

OBJECT_ID: Retorna o objeto número de identificação do atual banco de dados.

OBJECT_NAME: Retorna o nome do banco de dados objeto.

Exemplos:

```
USE AdventureWorks;
GO
DECLARE @ObjID int;
SET @ObjID = (SELECT OBJECT_ID ('Sales.Customer',
    'U'));
SELECT name, object_id, type_desc
FROM sys.objects
WHERE name = OBJECT_NAME (@ObjID);
GO
```

Funções NULL

Desempenham operações com os valores NULL da sua tabela.

ISNULL: Substitui NULL com o valor especificado substituição.

NULLIF: Retorna NULL se as duas expressões são equivalentes especificados.

COALESCE: Retorna o primeira expressão não NULL entre os seus argumentos.

Exemplos:

```
USE AdventureWorks;
SELECT AVG (ISNULL (Weight, 50))
FROM Production.Product;
```

O que são variáveis?

Uma variável é um container para dados. Você pode declarar variáveis no corpo de um lote ou uma procedure usando a declaração DECLARE. Você pode atribuir valores a uma variável usando as declarações SET ou SELECT. Depois de declarar, todas as variáveis são inicializadas com NULL.

Variáveis Locais

Uma variável local é um objeto que pode suportar um único valor dado de um tipo específico. Em contrapartida, uma tabela variável é usada para armazenar um conjunto de registros.

A seguir a sintaxe para declarar uma variável local.

```
DECLARE @local_variable as <data_type>
```

Você pode usar o SET e o SELECT para atribuir valores. A diferença fundamental entre o SET e Select é que você pode usar o SELECT para atribuir valores para mais de uma variável ao mesmo tempo, onde o SET permite atribuir um valor por vez. Outra diferença é que o SET é o jeito ANSI de atribuir valores e o SELECT não.

Nota SE você usar o SELECT para atribuir valor, você precisa ter certeza de que retornará somente um valor. Contudo, quando a expressão é o nome de uma coluna, ela pode retornar múltiplos valores. Se O SELECT retornar vários valores, a variável é atribuída pelo ultimo valor retornado.

Variáveis de Tabelas

Foram introduzidas como uma alternativa às tabelas temporárias para armazenar um grupo de registros. Oferecem muitas vantagens quando comparadas às temporárias.

- * Podem ser usadas para armazenar um conjunto de resultados no SQL Server, diferente das temporárias que requerem a declaração de um código.

- * variáveis de tabela utilizam menos recursos do que uma tabela temporária devido ao seu escopo bem definido.

Variáveis de tabela podem ser usadas para transações porque durar apenas durante a vigência dos atualização na tabela variável. Portanto, há menos bloqueio e madeireiras gerais, quando comparados com tabelas temporárias.

Variáveis de tabela exigem menos recompilações quando comparadas às tabelas temporárias. Quando você armazenar tabelas temporárias dentro de um procedimento armazenado pode ser necessário realizar mais recompilações do procedimento armazenado.

A seguir os exemplos de como declarar variáveis locais e variáveis de tabelas.

```
/* Declaring variables */
DECLARE @Variable1 AS int, @Variable2 AS int
/* Initializing two variables at once */
SELECT @Variable1 = 1, @Variable2 = 2
/* The same can be done using SET, but two SET statements are required */
SET @Variable1 = 1
SET @Variable2 = 2
```

```
USE Adventure Works;
DECLARE @Product Totals TABLE
(
    Product int,
    Revenue money
)
INSERT INTO @Product Totals (Product, Revenue)
SELECT Product, SUM (Unit Price * OrderQty)
FROM Sales.SalesOrderDetail
GROUP BY ProductID
```

O que são expressões?

Uma expressão é uma combinação de identificadores, valores e operadores que o SQL Server avalia para gerar um resultado. Expressões podem ser parte dos dados que precisam ser retornados de uma consulta. Você pode usar expressões numa condição de pesquisa para pesquisar dados.

Uma expressão pode ser :

- Constante
- Função
- Nome de Coluna
- Variável
- Subconsulta
- Case , NULLIF ou COALESCE

Exemplos de como usar expressões

Você pode construir uma expressão combinando entidades com operadores. Por exemplo, no SELECT seguinte o caractere literal "B%" usado como padrão para a cláusula LIKE, devem estar dentro de aspas simples.

```
USE Adventure Works
SELECT LastName, FirstName
FROM Person.Contact
WHERE LastName LIKE 'B%'
```

Você pode incluir valores de datas dentro das aspas como no SELECT seguinte.

```
USE AdventureWorks
SELECT *
FROM HumanResources.Employee
WHERE BirthDate='1972 May 15'
```

A seguir, mais de uma expressão é usada na consulta.

```
USE AdventureWorks
SELECT ProductID, (UnitPrice * OrderQty) as 'Total Price',
```

(ReceivedQty +10) as 'Received Quantity'
FROM Purchasing.PurchaseOrderDetail
Here ProductID, (UnitPrice * OrderQty), (ReceivedQty +10) são todas expressões.

Conversões explícitas e implícitas

Tipos de dados definem os dados contidos pelos objetos, tais como colunas, variáveis, e parâmetros. Nos dois tipos de conversões de dados estão implícitas e explícitas as conversões. Quando você declarar um tipo de dados para um objeto, o compilador lhe dá as informações necessárias de que necessita para desempenhar a maioria das traduções. Você pode exigir essas traduções para trabalhar com os valores armazenados nos objetos. Quando o compilador executa essas traduções por si próprio, ele é chamado de um tipo implícito expresso ou implícito de conversão. Quando você instruir o compilador para executar uma dessas traduções, é chamado um explícito tipo cast explícito ou conversão. Evite implícito tipo de dados conversão sempre que possível.

Tipos de declarações Condicionais

Uma declaração condicional é uma expressão que precisa ser formulada para avaliar uma declaração específica, sendo os resultados Verdadeiros ou falsos. Então, dependendo do resultado, você pode decidir que ação precisa ser feita. T-SQL suporta declarações condicionais, e oferece extensões procedurais para que o SQL te ofereça várias estruturas de controle de fluxo.

BEGIN...END -> Inclui uma serie de declarações T-SQL que são executadas por bloco. As palavras-chave BEGIN e END são palavras-chave da linguagem de controle de fluxo.

IF...ELSE -> Impõe condições na execução de uma declaração T-SQL. A declaração ou o bloco seguinte do IF é executado somente se a condição é satisfeita e a expressão booleana é verdadeira. Caso contrário, a declaração ou bloco que segue a opção ELSE é executada. Mas é sempre melhor incluir o BEGIN e o END enquanto se executa o IF...ELSE.

WHILE -> Define uma condição para a execução repetida de uma declaração ou bloco, enquanto as condições especificadas forem verdadeiras. O BREAK e o CONTINUE no Loop ajuda a parar e reiniciar a execução do WHILE.

CASE -> Avalia uma lista de condições e executa um dos possíveis resultados.

Você pode usar declarações condicionais para gerenciar o controle de fluxo do seu programa quando executado. Exemplos:

```
BEGIN...END
BEGIN
{
    sql_statement | statement_block
}
END
```

Neste, o the sql_statement|statement_block pode ser substituído por qualquer declaração válida ou um grupo de declarações definido como bloco.

```
BEGIN_END
USE AdventureWorks;
GO
BEGIN TRANSACTION;
GO
IF @@TRANCOUNT = 0
```

```

BEGIN
SELECT * FROM Person.Contact
WHERE LastName = 'ADAMS';
ROLLBACK TRANSACTION
PRINT N'Rolling back the transaction two times would cause an error.'
END
ROLLBACK TRANSACTION
PRINT N'Rolled back the transaction.'
GO

```

```

IF...ELSE
IF Boolean_expression BEGIN
{sql_statement | statement_block}
END
[ELSE BEGIN
{sql_statement | statement_block}]
END

```

Neste, a expressão booleana é substituída por uma condição e sql_statement | statement_block pode ser substituído por uma declaração ou um grupo de declarações.

```

-- Simple Boolean comparison with ELSE clause.
IF (suser_name () = 'sa') BEGIN
PRINT 'Congratulations. You are SA on this system.'
END
ELSE BEGIN
PRINT 'You are not the SA on this system.'
END

```

WHILE... Statement

```

WHILE Boolean_expression
{sql_statement | statement_block}
[BREAK]
{sql_statement | statement_block}
[CONTINUE]

```

Neste, a expressão booleana é substituída por uma condição . O poderia ser substituído por qualquer declaração válida ou um bloco. O BREAK pára a execução do loop mais interno, O CONTINUE reinicia o loop, ignorando as declarações seguintes a ele.

```

WHILE...
DECLARE @month INT,
        @full_date VARCHAR (30),
        @line VARCHAR (255)
SELECT @month=0
WHILE (@month <= 12)
BEGIN
    -- increment a variable for the month
    SELECT @month = @month + 1
    -- build a variable for the full date
    SELECT @full_date = RTRIM (CONVERT (CHAR (2), @month)) + '/01/99'
    -- build the output line
    SELECT @line = 'Processing for date: ' + @full_date
    -- print the output line
    PRINT @line

```

END

CASE... Statement

Simple Case Function

```
CASE input_expression
  WHEN when_expression THEN result_expression
  [...]
[
  ELSE else_result_expression
]
END
Searched CASE Function
```

```
CASE
WHEN Boolean_expression THEN result_expression
  [...]
[
  ELSE else_result_expression
]
END
```

Neste exemplo, a expressão de entrada é substituída por uma condição. A *when_expression* e *else_result_expression* poderiam ser substituídas por qualquer declaração válida.

CASE...statement.

```
Use AdventureWorks;
SELECT EmployeeID, DepartmentID, ShiftName= CASE ShiftID
WHEN 1 THEN 'Day Shift'
WHEN 2 THEN 'Evening Shift'
  WHEN 3 THEN 'Night Shift'
END
FROM HumanResources.EmployeeDepartmentHistory
ORDER BY EmployeeID
```

Agrupando Funções com Funcionalidades Similares

Classifique as expressões tão rapidamente quanto possível em seus associados categorias clicando no segmento adequado. Você também pode utilizar os atalhos do teclado pressionando 1 para a esquerda balde, 2 para o meio balde, e 3 para a direita balde.

Questões

Seu gerente de marketing quer que você execute uma consulta no banco de dados Adventure Works e prepare um relatório com certas condições. Se o preço máximo de um livro for menor ou igual a 500, você precisa dobrar o preço. Esse processo deve continuar até que o preço máximo seja maior do que 500. Qual expressão Condicional, que avalia uma lista de condições e retorna um dos resultados possíveis expressões?

- * BEGIN END
- * IF aninhado com WHILE
- * IF...ELSE
- * CASE X

Você precisa executar uma consulta para descobrir o numero de dias restantes para entregar um pedido ao cliente. Qual função de data você irá usar?

- * GATEDATE

- *DATEADD
- *DATENAME
- *DATEIFF (retorna o numero de dias entre a data de entrega e a data atual) X

Você precisa atualizar os registros do banco. que variável você vai usar

- *SET
- *SELECT
- *TABLE X
- *CREATE TABLE

Você precisa determinar o numero de dias trabalhados no ano corrente. Qual expressão você vai usar para fazer o calculo?

- *CONSTANT
- *FUNCTION X (usa-se a função SUM para somar o total de dias trabalhados)
- *NOME DA COLUNA
- *VARIABEL

Trabalhando com Scripts T-SQL

T-SQL scripts são uma série de comandos T-SQL que podem ser armazenados em um arquivo, que pode ser usado como entrada para SQL Query Analyzer ou SQL Management Studio. Estes utilitários podem executar os scripts SQL para fornecer a saída desejada. A T-SQL script tem um ou mais lotes que controlam como SQL Server executa o script. Eles são normalmente guardados como uma cópia permanente de todos os passos que são usadas para criar e povoar as bases de dados em seu servidor ou como um mecanismo de backup.

Após concluir esta lição, você será capaz de:

Explicar o propósito de lote diretivo.

Explique como adicionar comentários a T-SQL.

Identificar as melhores práticas para escrever código T-SQL.

O que são lotes diretivos?

Um lote contem uma ou mais declarações SQL, separadas por ponto e vírgula (;), construída dentro de uma string. lotes podem ser mais eficientes do que enviar declarações separadamente porque o tráfego da rede é reduzido. O Código é executado em lotes. Portanto, você exige lotes diretivos para instruir no SQL quando analisar e executar todas as instruções dentro de determinado lote.

GO - Esta diretiva assinala o fim de um lote de declarações T-SQL para o SQL Server utilitários. Estes utilitários GO interpretam como um sinal para eles enviam para todo o lote de declarações de SQL Server para sua execução. GO não é uma T-SQL, mas é um comando reconhecido pelo SQL Management Studio SQLCMD e utilitários. O sqlcmd utilitários permitem que você entre T-SQL declarações, sistema de procedimentos, e arquivos script no aviso de comando.

EXEC - Esta diretiva executa um valor escalar definido pelo usuário, função, um sistema de procedimento, um procedimento definido pelo usuário, armazenados, ou de um procedimento armazenado estendido, T-SQL dentro de um lote. Ele também suporta a execução de uma dinâmica T-SQL construiu pela concatenação de strings literais e variáveis string. Ele também suporta a execução de uma seqüência de caracteres dentro de um lote.

Ponto e Vírgula - É um terminador opcional T-SQL. A declaração é considerada completa apenas quando um ponto-e-vírgula é digitado no final. Quando um comando SQL abrange muitas linhas de um script ou quando você digitá-la como uma linha de comando, você precisará usar um ponto e vírgula para significar que a sua declaração está completa.

SET - Esta diretiva altera a forma como as informações específicas da atual sessão são manuseadas. As declarações SET são agrupadas nas seguintes categorias:

- * date e time
- * locking
- * querying execution
- * configurações SQL 92
- * transactions
- * miscellaneous

Como acrescentar comentários aos códigos?

É importante documentar seu código adicionando comentários apropriados porque ajuda na revisão a entender o propósito do código. Ainda mais, quando o numero de linhas aumenta, você pode ter dificuldade de lembrar o propósito de cada linha de código. Em tais situações, comentários ajudam a identificar facilmente blocos críticos. É também importante manter um histórico de mudanças no código para auxiliar na solução de problemas. No SQL Server há dois jeitos de comentar:

(- -) - Comenta uma única linha, na qual pode ser inserida numa linha separada aninhada ao fim da linha de comando ou dentro de uma declaração.

Dois hífenes (--) são indicadores do padrão SQL 92. A seguir o exemplo de uma linha de comentário:

```
-- Choose all columns and all rows  
SELECT * FROM HumanResources.Employee
```

(/*...*/) Blocos de comentário são usados quando se quer inserir um bloco de declarações como comentário. A seguir, o exemplo:

```
/* Choose all columns and all rows  
From the HumanResource.Employee Table */  
SELECT * FROM HumanResources.Employee
```

Melhores práticas para escrever códigos T-SQL

Para fazer os eu código mais legível, você precisa seguir certas recomendações. Essas praticas vão facilitar o desempenho da sua aplicação

Formatação

Para formatar seu código, destaque as palavras reservadas e identifique as declarações. Garante que a seqüência do código e a definição do bloco estejam corretas. Exemplo:

```
DECLARE @var INT  
BEGIN  
SET @var=1  
END
```

Nome de objetos

Coloque o nome dos objetos entre colchetes. Ex [Costumers]

Reter o nome da tabela como na base de dados. EX: OrderDetails

Incluir esquemas nos nomes de objetos.Ex: Sales.Costumers

Exemplo de Nome de Objeto

```
USE AdventureWorks;  
SELECT SalesOrderID  
FROM Sales.SalesOrderDetail
```

Ponto e Vírgula

Indica o fim de uma declaração. Usando ponto e vírgula se torna mais fácil de executar os comandos na linha de comando. Além disso, muitos outros bancos de dados requerem ponto e vírgula. Portanto usar ponto e vírgula no SQL facilita uma migração futura de banco de dados. EX:

```
USE AdventureWorks;
SELECT SalesOrderID, ProductID, OrderQty
FROM AdventureWorks.Sales.SalesOrderDetail
WHERE SalesOrderID>50000;
```

Usando Padrões ANSI SQL

Muitos desenvolvedores usam o padrão ANSI SQL quando não há diferença de desempenho e funcionalidade porque eles são compatíveis com os outros bancos de dados. EX:

```
USE AdventureWorks;
DECLARE @var int;
SET @var = CAST((SELECT TaxRate FROM
[Sales].[SalesTaxRate]
WHERE SalesTaxRateID=23) AS int)
PRINT @var
```

As melhores práticas a seguir.

- *Não use SELECT *, sempre especifique o nome das colunas.
- * Use SQL avançada, views, ou tabelas SQL Server 2000 quando processar dados. Senão uma tabela temporária é criada durante o processo e envolve varias operações de entrada e saída.
- *Use SET NOCOUNT ON no começo dos lotes, stored procedures, e triggers. Garante que as mensagens padrão não desapareçam depois da execução.
- * Evite usar declarações dinâmicas sempre que possível porque durante a execução, o SQL Server deve gerar um plano de execução. Portanto, fica mais lento do que as declarações estáticas.
- * Use nomes apropriados para ter uniformidade e legibilidade de código, Por exemplo, ao nomear Stored Procedures você precisa usar 'sp' como prefixo. Do mesmo jeito ao nomear variáveis integer ou char, precisa-se dos prefixos 'i' e 'c' respectivamente.

Código de comentário

Adicione comentários apropriados ao seu código para que a revisão dele fique mais fácil.

```
/*Here we select all SalesOrderID greater than 50000
*/
```

```
SELECT SalesOrderID, ProductID, OrderQty
FROM AdventureWorks.Sales.SalesOrderDetail
WHERE SalesOrderID>50000;
```

Questões

Você é o DBA da Adventure Works, e sua organização tem o SQL Server 2005. Você precisa usar scripts para consultar o banco. Quais das seguintes afirmações sobre comentários são verdadeiras?

- *Comentários são avaliados pelo servidor.
- *Os dois tipos de comentários que podem ser inseridos no script são o comentário de linha e o comentário de bloco. **X**
- *Comentário de linha é usado quando múltiplas linhas precisam ser comentadas.
- *Comentário de linha pode ser aninhado ao fim de um comando SQL. **X**
- *Comentário de bloco pode ser usado somente em uma única linha que precisa ser comentada.
- *Comentário de linha pode ser inserido dentro de uma declaração. **X**

Seguindo a questão anterior. Agora os scripts são executados em lote. Qual das seguintes é a função do lote diretivo GO?

Significa o fim da declaração.

Altera o modo na qual a informação específica da seção corrente é manipulada.

É o sinal do fim do lote. X

Executa uma declaração dinâmica com a concatenação de uma string literal e uma string variável.

Quais das praticas abaixo você precisa ter para aumentar a legibilidade do seu código?

Use SELECT ao invés de SET sempre que possível

Adicione comentários apropriados ao seu código para facilitar revisão X

Use declarações dinâmicas ao invés de estáticas

Use ponto e vírgula para indicar o fim da declaração X

Destaque as palavras reservadas. X

Use SELECT * nas suas consultas

Usando Ferramentas de Consulta T- SQL

Em SQL Server 2005, você pode consultar bases de dados usando várias ferramentas como SQL Server Management Studio, Business Intelligence Desenvolvimento Studio, SQLCMD, e Visual Studio. NET Designers. Ao usar o SQL Server Management Studio, você pode criar uma solução de dados e executar uma consulta. Você também pode executar uma consulta SQL Server em outras aplicações cliente como o Microsoft Office Access ou Microsoft Office Excel. Além disso, para se conectar ao SQL Server, você pode executar consultas de outros clientes que utilizam aplicações Microsoft Open Database Connectivity (ODBC), com links e incorporação Object Database (OLEDB), e SQL Native Client Access (SNCLI).

SQL Server Management Studio

SQL Server Management Studio é o principal meio para o desenvolvimento do SQL Server 2005. Você pode usar SQL Server Management Studio para criar, executar e salvar scripts associados a um determinado banco de dados. Você também pode usá-lo para criar soluções de dados contendo todos os scripts e aplicativos de dados graficamente. Além disso, SQL Server Management Studio fornece uma Query Builder, que pode ser usado para criar e executar consultas.

Business Intelligence Development Studio

Business Intelligence Desenvolvimento Studio é um ambiente de desenvolvimento do SQL Server 2005. Você pode usar Business Intelligence Desenvolvimento Studio para criar e manter BI soluções. Business Intelligence Desenvolvimento Studio também é utilizado para desenvolver aplicações SSAS.

SQLCMD

SQLCMD é a nova ferramenta de linha de comando de consulta SQL Server 2005, que substitui o isql e utilitários osql. SQLCMD funciona como uma alternativa à interface SQL Server Management Studio. Você pode executar consultas interativamente na SQLCMD, do mesmo modo que você pode com isql e osql. Além disso, você pode usar para automatizar SQLCMD T-SQL scripts que são invocados por lote de arquivos ou outras aplicações que não necessitam de programa para o SQL Server API.

Visual Studio. NET Designers

SQL Server 2005 fornece muitos designers que alargar o Microsoft Visual Studio. NET. Isso facilita a construir SQL Server 2005 itens, tais como relatórios e objetos de dados.

Nota: Você pode usar o Access e o Excel para acessar os bancos do SQL Server e mostrar os resultados nas suas respectivas aplicações. Depois de executar consultas no SQL management Studio , copie e cole o código nas respectivas aplicações.

Identificando os Componentes no SQLMS

Transcrição

Você pode usar SQL Server Management Studio para criar e gerenciar projetos de dados, que contém todas as conexões, consultas e outros objetos relacionados com a candidatura. Em SQL Server Management Studio, você também pode combinar vários Project os de dados sob uma solução, assim, gerir aplicações complexas.

Na amostra aplicação, que é utilizado na presente manifestação, o banco de dados projetos são armazenados no servidor Gen-Dev. Para ver todos os projetos no servidor, o que você precisa para se conectar ao servidor, fornecendo o nome do servidor e o Windows credenciais de autenticação no conectar-se ao Servidor caixa de diálogo. Após validação credenciais, o conteúdo do Object Explorer e da Síntese janela exibida. Esta janela mostra os projetos e as suas subpastas.

Object Explorer é uma ferramenta gráfica que é usado para localizar e gerenciamento de bancos de dados e objetos de dados. Agora, perto Object Explorer. Você também pode visualizar informações relacionadas com projetos no Resumo janela. Depois de visualizar as informações, feche a janela Síntese. Você pode agora abrir Object Explorer a partir de o menu Exibir.

Para adicionar um novo projeto no Solution Explorer, no menu Arquivo, aponte para Novo e, em seguida, clique em Project. A caixa de diálogo Novo Projeto aparece. Agora, especifique um nome para o novo projeto, e clique em OK. Repare que o novo projeto está agora disponível no Solution Explorer. Solution Explorer é uma ferramenta gráfica que lhe dá os detalhes do projeto soluções em uma árvore estrutura. Ela contém três pastas: Connection, consultas, e diversas pastas.

Depois de criar um novo projeto, o que você precisa para gerir o projeto e os servidores. Para visualizar uma lista de servidores que estejam matriculados em seu banco de dados, no menu Exibir, clique Registered Servers. O Registered Servidores janela aparece com uma lista de servidores registrados. Esta janela facilita a gestão eficiente servidor, fornecendo operações como parar, pausar, e iniciar servidores.

Para recuperar informações de um banco de dados para o seu projeto, você precisa criar uma consulta no Query janela. Esta janela é utilizada para escrever e executar consultas SQL. Para criar uma nova consulta, o botão direito do mouse a gen-dev pasta, aponte para Ligar e, em seguida, clique em New Query. Uma nova consulta arquivo aparece na janela de busca.

Você também pode abrir uma nova janela em consulta SQL Server Management Studio usando outros métodos. No menu Arquivo, você pode apontar para Novo e, em seguida, clique em Query Current Connection, ou você pode clicar no ícone Novo Query na barra de ferramentas. Neste caso, você deverá preencher as credenciais de login para se conectar ao banco de dados.

Para executar uma consulta sobre a base AdventureWorks, na nova consulta arquivo, digite USE AdventureWorks. Então, para executar a declaração, clique em Executar o ícone, na barra de ferramentas. Você também pode executar a consulta, pressionando a tecla de acesso, F5. As Mensagens guia exibe uma mensagem indicando que o comando foi concluído com êxito. Isto significa que você tenha sucesso ligado ao database AdventureWorks. Para obter os 10 melhores endereços da Morada tabela do AdventureWorks banco de dados, o tipo relevante selecione declaração. Então, para executar a declaração, clique em Executar o ícone na barra de ferramentas.

Para projetar e editar uma consulta ou declaração, você pode usar o Query Editor. Query Editor é uma ferramenta de fácil utilização que o ajuda a desenvolver consultas facilmente. Para invocar o Query Editor, o botão direito do mouse uma consulta declaração e, em seguida, clique em Design Query Editor. Endereço na tabela do Query Editor, limpar o ModifiedDate, rowguid, e AddressID caixas. Repare que a escolha declaração foi alterado para conter apenas os itens que foram selecionados no caixas da Morada tabela. Em seguida, clique em OK. Em seguida, clique em Executar o ícone na barra de ferramentas. Repare que os 10 melhores endereços da Morada tabela são

exibidos, sem os detalhes das colunas que você desmarcado.

Agora, esconder o Registered Servers, usando o recurso Auto Ocultar. Com este recurso, você pode ocultar automaticamente janelas que não estão em uso. Para ativar o Auto Hide recurso no Registrado Servidores janela, clique no botão Hide Automático. Para abrir a janela novamente, clique na guia Servidores Registrados. Você também pode fechar a janela Registrado Servidores usando o recurso Hide. Com esse recurso você pode simplesmente esconder uma janela. Para ativar o recurso no Hide Registrado Servidores janela, o botão direito do mouse a janela Posição botão e, em seguida, clique em Ocultar. Para mostrar uma janela oculta, no menu Exibir, clique na janela nome. Sem utilizar o Auto Hide Ocultar e recursos, você pode diretamente fechar qualquer janela, usando o botão Fechar. Nesta demonstração, que foram mostradas as funções dos diferentes componentes do SQL Server Management Studio.

Demonstrado procedimento

No menu Iniciar, aponte para Todos os programas, aponte para Microsoft SQL Server 2005 e, em seguida, clique em SQL Server Management Studio. Ligue para a caixa de diálogo Servidor de SQL Server Management Studio aparece.

Na caixa de diálogo conectar-se ao Servidor, clique em Connect para aceitar defaults. O conteúdo do Object Explorer e da Síntese janela exibida.

Fechar objeto Explorer.

Feche a janela Síntese.

No menu Ver, clique em Object Explorer. O Object Explorer aparece.

No menu Arquivo, aponte para Novo e, em seguida, clique em Project. A caixa de diálogo Novo Projeto aparece.

Na caixa Nome, selecione SQL Server Scripts1 e tipo SampleScript e, em seguida, clique em OK.

No menu Ver, clique Registered Servers. O Registered Servidores janela aparece.

Clique com o botão direito do gen-dev pasta, aponte para Ligar e, em seguida, clique em New Query. O GEN-DEV.master-SQLQuery1.sql arquivo aparece na janela do Query.

No GEN-DEV.master-SQLQuery1.sql arquivo, digite USE AdventureWorks.

Na barra de ferramentas, clique em Executar. A janela muda de nome GEN-DEV.master-SQLQuery1.sql para GEN-DEV.AdventureWorks-SQLQuery1.sql. Na Mensagem guia, aparece uma mensagem indicando que o comando foi concluído com êxito.

No GEN-DEV.AdventureWorks-SQLQuery1.sql arquivo, digite a seguinte instrução: `SELECT TOP 10 * FROM Person.Address`.

Na barra de ferramentas, clique em Executar. Os resultados da sua pesquisa são exibidos na guia Resultados.

No GEN-DEV.AdventureWorks-SQLQuery1.sql arquivo, selecione a declaração `SELECT TOP 10 * FROM Person.Address`, o botão direito do mouse e, em seguida, clique em Design Query Editor. O Query Editor aparecerá.

No Endereço (Pessoa) tabela, clique para limpar o ModifiedDate, rowguid, e AddressID caixas.

Clique em OK.

Na barra de ferramentas, clique em Executar. A consulta resultados são exibidos na guia Resultados.

No Registrado Servidores janela, na janela Posição lista, clique no botão Hide Automático.

Clique no Registrado Servidores janela.

No Registrado Servidores janela, o botão direito do mouse a janela Posição botão e, em seguida, clique em Ocultar.

Fechar SQL Server Management Studio.

O que é uma Solução SQL Server?

É uma coleção de um ou mais projetos que contem conexão com banco de dados e Scripts. Para criar uma Solução, você precisa criar um projeto especificando o seu tipo. Você pode fazer isso selecionando o tipo numa lista de modelos oferecidos pelo SQL Server. Esses modelos incluem scripts do SQL Server, do Analysis Server e do SQL Mobile. Após selecionar o tipo de projeto, a nova solução é criada e mostrada no Solution Explorer.

Funções da Pasta de uma Solução SQL

As pastas de uma solução tanto como conexão, consulta e diversos, com componentes apropriados, são mostradas no Solution Explorer. A seguir, as funções dos componentes presentes sob as pastas do projetos.

- *A pasta Connections consiste da definição dos arquivos que podem ser usados para configurar as credenciais para se conectar a nova solução.

- *A pasta Query consiste de diferentes tipos de arquivos de consultas criados para o projeto específico. Por exemplo, se o projeto é um Script SQL Server, todos os scripts SQL são armazenados nessa pasta. Você pode adicionar, modificar e deletar esses scripts.

- *A pasta Diversos consiste de todos os outros arquivos de suporte requeridos pela solução.

Uma Solução SQL Server é salva como arquivo .ssmssl. Quando você abre esse arquivo, pode ver todos os scripts e conexões salvas para a Solução.

Criando uma Solução SQL Server

Esta demonstração explica como criar uma solução no SQL Server Management Studio.

Você pode criar uma solução no Microsoft SQL Server SQL Server Management Studio. Abra SQL Server Management Studio. Em seguida, conecte ao servidor. Como um primeiro passo no sentido da criação de uma solução SQL Server, você precisará criar um novo projeto no SQL Server Management Studio. Para isso, no menu Arquivo, aponte para Novo e, em seguida, clique em Project. A caixa de diálogo Novo Projeto aparece.

Na caixa de diálogo, SQL Server Scripts é selecionado como o modelo padrão. Na caixa Nome, digite SampleScript-SolutionExp como o nome do projeto e, em seguida, clique em OK. Repare que este nome também é exibido na caixa Nome Solution. Clicar em OK, o projeto é criado e exibido na Solution Explorer. Em seguida, você precisa se conectar ao computador que está executando SQL Server, a fim de obter informações. Para fazer isso, o botão direito do mouse a pasta Conexões e, em seguida, clique em New Connection. Ligue para a caixa de diálogo Servidor aparece. Na caixa de diálogo conectar-se ao Servidor, verificar se GEN-DEV é selecionado na lista o nome do servidor e, em seguida, clique em OK.

Depois de estabelecer uma conexão com o GEN-DEV servidor, você precisa criar uma consulta para recuperar informações. Para fazer isso, o botão direito do mouse a pasta Consultas e, em seguida, clique em New Query. Ao adicionar uma consulta, a Connect para Database Engine caixa de diálogo aparecer. Nesta caixa de diálogo, você precisará especificar a conexão informações como nome do servidor e autenticação modalidade. Clique em Conectar-se a aceitar os padrões.

Agora, renomeie a consulta para GetAddress.sql. GetAddress.sql abre o arquivo na janela de busca. Para obter o endereço do top 10 pessoas desde o AdventureWorks banco de dados, digite os seguintes comandos SQL: `USE AdventureWorks SELECT TOP 10 * FROM PERSON.ADDRESS`. Depois de escrever a consulta, para salvá-lo e, em seguida, salve o projeto.

Para além das consultas, você também pode adicionar diversos arquivos tais como Ajuda arquivos, documentos do Microsoft Office Word, e outros projetos para o seu projeto. Por exemplo, para adicionar um arquivo Ajuda para o seu projeto, na caixa de diálogo Adicionar Item existente, navegue até o local onde o arquivo estiver presente, selecione Todos os arquivos da lista Arquivos do tipo e, em seguida, acrescentar o arquivo Ajuda, help.txt . O help.txt arquivo aparece na janela de busca. Agora, o arquivo Ajuda é adicionado à miscellaneous pasta. Finalmente, salve o SampleScript-SolutionExp projeto. Esta demonstração mostrou-lhe como criar uma solução em SQL Server SQL Server Management Studio.

Demonstrado procedimento Criando uma solução SQL Server

No menu Iniciar, aponte para Todos os programas e, em seguida, aponte para Microsoft SQL Server 2005 e clique em SQL Server Management Studio. O SQL Server Management Studio janela aparece.

Na caixa de diálogo conectar-se ao Servidor, verificar se o tipo de servidor, nome Server, a autenticação e caixas foram preenchidos corretamente e, em seguida, clique em Connect para aceitar as configurações padrão.

No menu Arquivo do SQL Server Management Studio, aponte para Novo e, em seguida, clique em Project. A caixa de diálogo Novo Projeto aparece.

Na caixa Nome, selecione SQL Server Scripts1 e tipo SampleScript-SolutionExp e, em seguida, clique em OK.

Em Solution Explorer, o botão direito do mouse a pasta Conexões do SampleScript-SolutionExp projeto e, em seguida, clique em New Connection. Ligue para a caixa de diálogo Servidor aparece.

Na caixa de diálogo conectar-se ao Servidor, clique em OK.

Em Solution Explorer, o botão direito do mouse a pasta Consultas e, em seguida, clique em New Query. Ligue para o Database Engine caixa de diálogo aparecer.

Na Ligue para Database Engine caixa de diálogo, clique em Connect.

Em Solution Explorer, o botão direito do mouse SQLQuery1.sql o arquivo, clique em Renomear e, em seguida, digite GetAddress.sql. GetAddress.sql abre o arquivo na janela de busca.

No GetAddress.sql arquivo, digite os seguintes comandos SQL. `USE AdventureWorks SELECT TOP 10 * FROM PERSON.ADDRESS`.

No menu Arquivo, clique em Salvar GetAddress.sql.

Em Solution Explorer, o botão direito do mouse o SampleScript-SolutionExp projeto e, em seguida, clique em Salvar SampleScript-SolutionExp.ssmssqlproj.

Em Solution Explorer, o botão direito do mouse o SampleScript-SolutionExp projeto, aponte para Adicionar e, em seguida, clique em Existente Item. A caixa de diálogo Adicionar Existente Item aparece.

Na lista Examinar, clique em Desktop.

Na lista Arquivos do tipo, clique em Todos os arquivos (*.*)

Sobre a guia Desktop, clique help.txt e, em seguida, clique em Adicionar. O help.txt arquivo aparece na janela de busca.

Em Solution Explorer, o botão direito do mouse o SampleScript-SolutionExp projeto e, em seguida, clique em Salvar SampleScript-SolutionExp.ssmssqlproj.

Executando uma consulta no SQL Management Studio

SQL Server Management Studio é uma ferramenta integrada com o SQL Server 2005 e pode ser usado para criar queries. Para design consultas, abra SQL Server Management Studio. Repare que o Microsoft SQL Server Management Studio janela com a conectar-se ao Servidor caixa de diálogo aparecer. Na amostra aplicação, que é utilizado na presente manifestação, o banco de dados projetos são armazenados no servidor Gen-Dev. Para ver todos os projetos no servidor, o que você precisa para se conectar ao servidor, fornecendo o nome do servidor e o Windows credenciais de autenticação no conectar-se ao Servidor caixa de diálogo. Após validação credenciais, SQL Server Management Studio aparece.

Para abrir uma nova janela em consulta SQL Server Management Studio, clique no ícone Nova Consulta na barra de ferramentas. A consulta arquivo abre na janela de busca. Você pode usar a consulta janela para escrever consultas SQL, executar procedimentos armazenados, e também executar operações. Para conectar à AdventureWorks base de dados que você precisa para executar consultas, o tipo USO declaração na consulta arquivo. Repare que a Mensagens guia exibe uma mensagem indicando que o comando foi concluído com êxito e uma ligação à base de dados está estabelecido AdventureWorks.

Agora, para obter o endereço do top 10 pessoas desde o endereço AdventureWorks tabela de banco de dados, o tipo relevante selecione declaração. Em seguida, na barra de ferramentas, clique no ícone análise para analisar a consulta antes de executá-la. Analisando uma consulta ajuda a identificar os erros. Então, você pode executar a consulta e executar a operação. Para executar uma consulta, clique em Executar o ícone na barra de ferramentas.

Após executar a consulta, você pode ver os resultados no separador Resultados. Mensagens sobre o separador Resultados do painel, você também pode ver o número de registros afetados pela consulta, bem como o tempo consumido para sua execução. Agora, feche a consulta arquivo. Uma caixa de diálogo, a fim de verificar se você quiser salvar as alterações feitas ao itens é exibida. Na caixa de mensagem, clique em Não se você não quiser salvar as alterações feitas ao itens mostrados.

Em seguida, a abrir um arquivo Sample.sql, no menu Arquivo, aponte para Abrir e, em seguida, clique em Arquivo. A caixa de diálogo Abrir ficheiro aparece. Abra o arquivo Sample.sql. Ligue para o Database Engine caixa de diálogo aparecer. Para fornecer as definições de ligação para o Sample.sql arquivo, na Ligue para Database Engine caixa de diálogo, clique em Connect. Após fornecer configurações de conexão para o arquivo de amostra, na janela de busca, selecione a declaração e, em seguida, clique no ícone Executar. Você pode ver a cor convenções na consulta selecionada. As palavras em azul são reservados palavras, as palavras em rosa são funções, e as palavras em verde

são comentários. Então, para executar a consulta selecionada, na barra de ferramentas, clique no ícone Executar. Você já foram mostrados como para executar uma consulta no SQL Server Management Studio.

Demonstrado procedimento

Executando uma consulta no SQL Server Management Studio

No menu Iniciar, aponte para Todos os programas e, em seguida, aponte para Microsoft SQL Server 2005 e, em seguida, clique em SQL Server Management Studio. O Microsoft SQL Server Management Studio janela e ao conectar-se ao Servidor caixa de diálogo aparecer.

Na caixa de diálogo conectar-se ao Servidor, clique em Connect. O Object Explorer e da Síntese janela exibida.

Na barra de ferramentas do SQL Server Management Studio, clique no ícone New Query. O Gen-Dev.master-SQLQuery1.sql arquivo abre na janela de busca.

Na Gen-Dev.master-SQLQuery1.sql arquivo, digite a seguinte declaração e, em seguida, clique no ícone Executar. USE AdventureWorks

Na Gen-Dev.master-SQLQuery1.sql arquivo, digite a seguinte declaração e, em seguida, clique em Executar. SELECT TOP (10) * FROM Pessoa. Endereço

Na barra de ferramentas, clique no ícone análise e, em seguida, clique no ícone Executar. O GEN-DEV.master-SQLQuery1.sql arquivo muda para GEN-DEV.Adventureworks-SQLQuery1.sql arquivo.

Resultados no painel, clique na guia Mensagens e, em seguida, clique na guia Resultados.

Fechar o GEN-DEV.Adventureworks-SQLQuery1.sql arquivo. Uma caixa de mensagem para verificar se você quiser salvar as alterações feitas ao itens é exibida.

Na caixa de mensagem, clique em Não.

No menu Arquivo, aponte para Abrir e, em seguida, clique em Arquivo. A caixa de diálogo Abrir ficheiro aparece.

No Desktop pasta, clique no arquivo Sample.sql e, em seguida, clique em Abrir. Sample.sql abre o arquivo na janela de busca.

Na Ligue para Database Engine caixa de diálogo, clique em Connect.

Na Gen-Dev.master-Sample.sql arquivo, selecione as declarações USO AdventureWorks SELECT AddressID, Cidade, Getdate () FROM Person.Address e, em seguida, na barra de ferramentas, clique em Executar.

Executando uma consulta no Microsoft Excel

Você pode AdventureWorks consulta ao banco de dados, e importar os dados que você exigir do banco de dados em Office Excel para análise. Para importar dados externos a partir de uma determinada fonte de dados em Office Excel, no menu Dados, aponte para Importar dados externos e, em seguida, clique em New Database Query. A caixa de diálogo Escolher Fonte Dados aparece. Verifique se o Use o Assistente de consulta para criar / editar consultas opção tiver sido selecionada. Em seguida, você precisa especificar o nome do banco de dados, e driver para se conectar a fonte de dados especificada. Para fazer isso, na guia Bancos de dados na caixa de diálogo Escolher Fonte de

dados, verificar que a Nova Fonte de dados foi selecionado e, em seguida, clique em OK. O Criar Nova Fonte de dados caixa de diálogo é exibida.

Em seguida, você precisa especificar o nome dos dados, no Que nome você deseja dar a sua fonte de dados? Caixa. Depois de nomear a fonte de dados, é necessário especificar o nome do driver baseado no tipo de banco de dados. Porque o AdventureWorks de dados SQL Server é um banco de dados, no Selecione um driver para o tipo de dados que você deseja acessar lista, selecione SQL Server. Para conectar à AdventureWorks banco de dados, clique em Connect. O SQL Server Login caixa de diálogo aparecer. Na lista Server, especificar o nome do servidor.

Em seguida, verificar se a caixa Use Trusted Connection foi selecionada. Em seguida, na área Opções, especifique a conexão detalhes. Na lista de banco de dados, clique AdventureWorks. Na lista de idiomas, clique em Inglês. Na caixa Nome do aplicativo, verificar que a Microsoft Office 2003 é exibida. No Workstation ID casa, verificar se GEN-DEV é exibido e, em seguida, clique em OK. O Criar Nova Fonte de dados caixa de diálogo é exibida.

Para selecionar uma tabela padrão para sua fonte de dados, clique em Endereço e clique em OK. Na caixa de diálogo Escolher Fonte de dados que é exibida, verifique se TestDSN está selecionado e clique em OK. O Query Wizard aparece. No endereço tabela, selecione a coluna Cidade. Em seguida, clique em Adicionar para adicionar à Cidade Colunas em sua consulta lista. Agora, clique em Avançar. O Filtro de Dados página aparece. Para executar uma consulta para mostrar os nomes de cidades diferentes do Newark, clique na condição não é igual e, em seguida, clique em Newark.

No Filter Dados página, clique em Avançar. A Classificar Ordem página aparece. Em seguida, na lista Ordenar por, clique em Cidade e, em seguida, clique em Avançar. O Query Wizard - Finish caixa de diálogo aparecer. Em seguida, para retornar os resultados para o Microsoft Office Excel, selecione o Retorno Dados para Microsoft Office Excel opção e, em seguida, clique em Concluir.

Agora, selecione a coluna em que você deseja que os resultados sejam exibidos. Por exemplo, você pode exigir os resultados a ser exibido a partir de Coluna A Linha 1. A coluna e a linha em que o cursor está atualmente colocado aparece por padrão no Existente planilha caixa Importar dados da caixa de diálogo. Para aceitar padrões, clique em OK. Agora, o Microsoft Office Excel é preenchida com os nomes de várias cidades, com exceção de que Newark. Esta demonstração mostrou-lhe como executar uma consulta no Microsoft Office Excel por importar dados de um database AdventureWorks.

Demonstrado procedimento
Executando uma Consulta no Microsoft Excel

Abra o Microsoft Office Excel 2003.

Dados sobre o menu, aponte para Importar dados externos e, em seguida, clique em New Database Query. A caixa de diálogo Escolher Fonte Dados aparece.

Bancos de dados sobre o guia, verificar que a Nova Fonte de dados foi selecionado e, em seguida, clique em OK. Criar Nova Fonte de dados a caixa de diálogo aparecer.

No Que nome você deseja dar a sua fonte de dados caixa, o tipo TestDSN.

No Selecione um driver para o tipo de dados que você deseja acessar lista, clique em SQL Server.

No Criar Nova Fonte de dados caixa de diálogo, clique em Connect. O SQL Server Login caixa de diálogo aparecer.

Na lista Server, o tipo GEN-DEV. Verifique se a caixa Use Trusted Connection foi selecionada.

Clique em Opções. As Opções área é exibida.

Na lista de banco de dados, clique AdventureWorks.

Na lista de idiomas, clique em Inglês.

Na caixa Nome do aplicativo, verificar que a Microsoft Office 2003 é exibida.

No Workstation ID casa, verificar se GEN-DEV é exibido e, em seguida, clique em OK.

No Selecione uma tabela padrão para sua fonte de dados (opcional) lista, clique em Endereço e clique em OK.

Na caixa de diálogo Escolher Fonte de dados que é exibida, verifique se TestDSN está selecionado e clique em OK. O Query Wizard aparece.

Disponível nas tabelas e colunas lista das colunas de dados que você deseja incluir em sua consulta página, clique para selecionar a cidade no Morada tabela. O endereço tabela é selecionada.

Clique em Adicionar (>) para adicionar à Cidade Colunas em sua consulta lista.

Na Escolha Colunas página do Query Wizard, clique em Avançar. O Filtro de Dados página aparece.

Clique para selecionar Cidade na Coluna de área Filtro. Na Cidade da lista Apenas incluem linhas onde seção, clique não iguais, e, em seguida, clique em Newark.

No Filter Dados página, clique em Avançar. A Classificar Ordem página aparece.

No Ordenar por lista, clique City, e clique em Avançar para aceitar as configurações padrão. O Query Wizard - Finish caixa de diálogo aparecer.

Na O que você gostaria de fazer próxima página, verifique se Retornar Dados para Microsoft Office Excel foi selecionada e, em seguida, clique em Concluir. Importar dados a caixa de diálogo aparecer.

Clique em OK para aceitar as configurações padrão. Os dados da Cidade coluna é importado para o Microsoft Office Excel sem a cidade Newark.

Questões

Como DBA da Adventure Works , você precisa escrever consultas interativas e obter os resultados do SQL Server 2005. Qual ferramenta de consulta você vai usar para consultar o banco de dados?

*SQL Server Management Studio

* Business Intelligence Development Studio

*SQLCMD X

*Visual Studio .NET Designers

Você é o DBA da Adventure Works e você trabalha com SQL Management Studio. Você é escolhido para gerenciar o banco de dados dos empregados. Qual ferramenta gráfica você irá usar?

*Object Explorer X

*Solution Explorer

*Registered Servers

* Query Editor

Você precisa importar dados do SQL Server 2005 para um aplicativo Microsoft, para então analisar e gerar um relatório. Qual aplicativo você propõe:

- *Word
- *Excel X
- *InfoPath
- *PowerPoint

Você precisa criar uma solução no SQL MS. Para isso precisa criar um projeto primeiro. Que tipos de projeto você vai selecionar para criar a solução?

- *SQL Server Script X
- *Connection
- *Analysis Server Script X
- *Queries

Lab.:Começando com Banco de Dados e T-SQL

Após concluir esta lição, você será capaz de:

Explore os componentes do SQL Server Management Studio.

Executar consultas em SQL Server Management Studio.

Examine diagrama em um banco de dados SQL Server Management Studio.

Gerar um relatório a partir de uma base em SQL Server 2005 usando o Microsoft Office Excel.

Você é o DBA da Adventure Works. O departamento de recursos humanos solicita que você gere vários relatórios usando o SQL Server 2005. Você tem que gerar relatórios com o endereço dos empregados ordenados por departamento ou uma lista de endereço dos empregados residentes nos Estados Unidos. Você também precisa analisar os detalhes dos novos empregados contratados usando um diagrama de banco de dados no SQL management Studio. Para isso você vai explorar e executar consultas, gerar relatórios do banco de dados e examinar o diagrama do banco de dados.

Exercícios

Testes no SQL Server 2005

Revisão

Neste Lab. você explorou o SQL management Studio , se conectou ao SQL Server 2005 para ver os vários componentes de um projeto.

Executou scripts e consultas, examinou um diagrama de banco de dados, criou um relatório e importou dados para o Excel.

Para conectar aos servidores de banco de dados registrado, você tem que clicar no projeto do Solution Explorer e clicar com o botão direito em Connections, new connection.Qual nome da caixa de dialogo aparece?

R: Conect to Server

Você quer rodar algumas consultas . Para esconder o Solution Explorer e o Object Explorer, que ocupam espaço na sua janela. Que botão você clica?

R: Auto Hide.

Você tem que rodar uma consulta no query builder. Que opção você escolhe no menu?

R: Design Query.

Você precisa gerar um relatório no Excel do SQL Server Management Studio. Que opção você clica primeiro no menu para abrir a caixa Create New Data Source?

R: Dados.

-----*****-----

Desempenhando Consultas Básicas no SQL Server 2005

Desempenhando consultas básicas

A declaração SELECT é usada para retornar dados de uma tabela. Os resultados de um SELECT são armazenados em uma tabela chamada do conjunto de resultados também chamada de conjunto de linhas. Usando o SELECT, você pode filtrar e formatar os conjuntos de resultados efetivamente. Você pode filtrar os dados usando operadores de comparação, operadores lógicos e strings de comparação. Você pode formatar os dados por combinação, por quebra, ordenação e convertendo os dados de acordo com os requisitos. Para escrever consultas eficientes, você precisa ser capaz de escrever argumentos de busca apropriados e de entender como o Query Optimizer no SQL Server processa as consultas.

Após completar esse módulo você será capaz de:

- *Retornar dados com o SELECT
- *Filtrar dados usando diferentes condições de busca.
- *Explicar como se trabalha com valores NULL
- *Formatar resultados.
- *Descrever as considerações de desempenho que afetam o retorno dos dados.

Usando a Declaração SELECT

O SELECT pode ser usado considerando três cláusulas: SELECT, FROM, WHERE. O SELECT especifica a lista de colunas a serem retornadas das tabelas escolhidas no FROM. Você pode colocar apelidos nas tabelas. O WHERE é opcional, para filtrar os dados.

Elementos da Declaração SELECT

Usando o SELECT, você pode retornar dados de uma tabela. Pode armazenar dados em um formato relacional conhecido como conjunto de resultados. O formato geral do SELECT é mostrado por você. Nesta declaração o SELECT é obrigatório.

A lista é usada por um particionamento vertical da tabela em termos de colunas. Ela descreve as colunas do resultado. Usando o asterisco (*) no SELECT significa que todas as colunas da tabela buscada devem ser retornadas.

O FROM contém uma lista das tabelas do qual o resultado é retornado. Quando existem várias tabelas no FROM, elas são separadas por vírgulas.

O WHERE é usado para um particionamento horizontal em termos de linha na tabela. Ele define as condições que deve encontrar em cada linha da tabela para qualificar o SELECT.

Você precisa especificar uma expressão lógica após essa cláusula, também conhecida como predicado, que avalia tanto o TRUE, FALSE e UNKNOWN.

A cláusula GROUP BY particiona os resultados em grupos baseados nos valores das colunas da expressão group by.

A cláusula HAVING é um filtro adicional que é aplicado ao resultado. A cláusula ORDER BY define a ordem que as linhas devem ser retornadas, se de forma ascendente ou descendente. Para obter o resultado correto, especifique as cláusulas na ordem apropriada.

Select [campo]

From [tabela]

Where [condição]

Group by [campos listados no select]

Having [condição]

Order by [ordem em que o resultado deve aparecer]

Como Retornar Colunas de uma Tabela

Você pode selecionar todas as colunas ou colunas específicas.

Todas as colunas: Use o asterisco (*). Por Exemplo: `SELECT * FROM PERSONS`. Retornarão todas as colunas da tabela PERSONS, Contudo, esta declaração causa um impacto no desempenho do SQL Server. Portanto é recomendável que se especifique as colunas.

Colunas Especificas: Para mostrar dados de colunas especificas de uma tabela, você precisa especificá-las separando por vírgula. O resultado aparecerá na mesma ordem em que as colunas estão na lista.

Você pode também usar `$IDENTITY` para percorrer a coluna que tem essa propriedade e `%ROWGUID` para percorrer a coluna que tiver esta propriedade.

Exemplo:

USE AdventureWorks

SELECT * FROM HumanResources.Employee

WHERE EmployeeID < 10

A seguir o resultado parcial do resultado de uma consulta.

EmployeeID	NationalIDNumber	ContactID	LoginID	ManagerID
1	14417807	1209	Adventure-Works\guy1	16
2	253022876	1030	Adventure-Works\kevin0	6
3	509647174	1002	Adventure-Works\roberto0	12

USE AdventureWorks

SELECT ProductID, Name, ProductNumber, Color

FROM Production.Product

ProductID	Name	ProductNumber	Color
1	Adjustable Race	AR-5381	NULL
2	Bearing Ball	BA-8327	NULL
3	BB Ball Bearing	BE-2349	NULL

USE AdventureWorks

SELECT SalesPersonID, TerritoryID, SalesQuota, ModifiedDate

FROM Sales.SalesPerson

WHERE ModifiedDate < '2003-01-01'

SalesPersonID	TerritoryID	SalesQuota	ModifiedDate
268	NULL	NULL	2001-01-28 00:00:00.000
275	2	300000.00	2001-06-24 00:00:00.000
276	4	250000.00	2001-06-24 00:00:00.000

Há algumas desvantagens de usar o `SELECT *` e é preciso estar ciente. Ele retorna todas as colunas presentes independente se você precisa delas.

Pode tomar muito tempo para retornar os dados. E pode retornar de uma forma incorreta se a estrutura da tabela em uma data posterior.

O SQL Server pode procurar as tabelas listadas antes mesmo de executar a consulta causando problemas de desempenho em alguns casos.

Impede o uso dos índices e afeta o desempenho.

Não é recomendado para uma documentação porque não dá nenhuma informação sobre o dado retornado.

Como usar Apelidos para as Tabelas.

Você pode usar apelidos para melhorar a legibilidade das consultas. Ex:

```
FROM [FORCE] Table_List_Item [, ...] [[JoinType] JOIN  
DatabaseName!]Table [[AS] Local_Alias] [ON JoinCondition [AND | OR  
[JoinCondition] | FilterCondition] ...]
```

Nesta sintaxe temos um apelido para a tabela temporária. Você pode usar o apelido ao invés do nome da tabela através da consulta. Pode usar múltiplas tabelas usando um ou mais join.

Pode usar apelidos no WHERE, JOIN e subconsultas.

O apelido é usado para obter informação de duas tabelas separadas ao juntá-las. Você pode criar um apelido especificando logo na frente do nome no FROM. Você pode ler as consultas facilmente com os apelidos.

A seguir, os exemplos:

```
USE AdventureWorks  
SELECT a.EmployeeID, a.NationalIDNumber, a.Title  
FROM HumanResources.Employee AS a  
WHERE MaritalStatus = 'M'
```

EmployeeID	NationalIDNumber	Employee title
1	14417807	Production Technician - WC60
3	509647174	Engineering Manager
5	480168528	Tool Designer
8	690627818	Production Technician - WC10

```
USE AdventureWorks  
SELECT P.ProductID, P.Name, O.UnitPrice, O.DueDate  
FROM Purchasing.PurchaseOrderDetail as O,  
Production.Product as P  
WHERE P.ProductID = O.ProductID
```

ProductID	Name	UnitPrice	DueDate
1	Adjustable Race	50.26	2001-053100:00:00.000
359	Thin-Jam Hex Nut 9	45.12	2001-053100:00:00.000
360	Thin-Jam Hex Nut 10	45.5805	2001-053100:00:00.000
530	Seat Post	16.086	2001-053100:00:00.000

```
USE AdventureWorks  
SELECT E.EmployeeID, E.AddressID, A.AddressLine1  
FROM HumanResources.EmployeeAddress E,
```

Person.Address A
 WHERE E.AddressID = A.AddressID

EmployeeID	AddressID	AddressLine1
261	215	101 Candy Rd.
278	33	10203 Acorn Avenue
215	147	1061 Buskrik Avenue

Como Retornar Linhas Específicas da Tabela

Usando a cláusula WHERE. Esta cláusula opcional filtra as linhas das tabelas buscadas usadas no resultado. Você pode especificar uma série de condições e somente aquelas linhas que batem com os termos dessas condições aparecem no resultado.

Se você não usar o where, o SQL Server vai trazer as tabelas com todos as linhas. Frequentemente você não precisara de todos os dados. Além disso pode gastar muito processamento. Evita que o usuário modifique o dado que lido em tempo corrente que causa atraso.

Você pode com o Where usar índices para facilitar a execução. O índice ajuda a procurar um determinado registro.

Exemplo:

SELECT NationalIDNumber, Title
 FROM AdventureWorks.HumanResources.Employee
 WHERE EmployeeID=5

NationalIDNumber	Title
480168528	Tool Designer

SELECT FirstName, MiddleName, LastName, EmailAddress
 FROM AdventureWorks.Person.Contact
 WHERE ContactID <50

FirstName	MiddleName	LastName	EmailAddress
Gustavo	NULL	Achong	gustavo0@adventure-works.com
Catherine	R.	Abel	catherine0@adventure-works.com
Kim	NULL	Abercrombie	kim2@adventure-works.com

SELECT ProductID, ProductNumber
 FROM AdventureWorks.Production.Product
 WHERE Name='Blade'

ProductID	ProductNumber
-----------	---------------



Questões

Quais das seguintes afirmações são verdadeiras sobre a declaração SELECT?

A cláusula SELECT especifica as linhas a serem retornadas das tabelas listadas na cláusula FROM.

A cláusula SELECT é obrigatória. **X**

A cláusula SELECT consiste de nomes de colunas separados por vírgulas. **X**

A cláusula SELECT com um asterisco é usada para retornar todas as colunas de todas as tabelas listadas na cláusula FROM. **X**

A cláusula DISTINCT mostra todas as linhas retornadas pela cláusula SELECT.

A cláusula DISTINCT é o identificador padrão da cláusula SELECT.

Qual das seguintes declarações são verdadeiras sobre apelidos de tabela?

Apelidos para colunas ou tabelas são colocadas antes do nome.

Apelidos só podem ser usados na cláusula SELECT.

Apelidos podem ser usados para especificar múltiplas tabelas na cláusula JOIN. **X**

Apelidos são de três tipos: coluna, tabela e apelidos locais.

Filtrando Dados

Você pode minimizar os dados enviados pela rede filtrando-os. Para isso use os operadores lógicos e de comparação na cláusula WHERE na sua declaração SELECT. Usando a Cláusula WHERE, você pode selecionar uma gama ou uma lista de dados de uma tabela. E também desenvolver comparações de string e ligações padrão nos dados.

Como filtrar dados usando operadores de comparação

Operadores de comparação são usados para comparar duas expressões. Esses operadores podem ser usados em todas as expressões exceto as que envolvem text, ntext (National Text ou Unicode) , ou image.

A seguir a Tabela de operadores:

Operador	Descrição
=	igual a
>	Maior do que
<	Menor do que
>=	Maior ou igual a
<=	Menor ou igual a
<>	Diferente
!=	Diferente
!<	Não menor que
!>	Não maior que

Nota: os operadores !=, !<, e !> não são incluídos no padrão SQL-92

O resultado de um operador de comparação em duas expressões retorna um valor booleano. Portanto, as expressões são chamadas de expressões booleanas.

```
USE AdventureWorks
SELECT ProductID, NAME
FROM .Production.Product
WHERE Class ='H'
```

ProductID	Name
680	HL Road Frame - Black, 58
706	HL Road Frame - Red, 58
717	HL Road Frame - Red, 62

```
USE AdventureWorks
SELECT Name, ProductNumber, Color, StandardCost
FROM Production.Product
WHERE StandardCost >= 500
```

ProductName	ProductNumber	Color	StandardCost
HL Road Frame - Black, 58	FR-R92B-58	Black	1059.31
HL Road Frame - Red, 58	FR-R92R-58	Red	1059.31
HL Road Frame - Red, 62	FR-R92R-62	Red	868.6342

```
USE AdventureWorks
SELECT EmployeeID, NationalIDNumber, ContactID, ManagerID
FROM HumanResources.Employee
WHERE ManagerID <> 16
```

EmployeeID	NationalIDNumber	ContactID	ManagerID
2	253022876	1030	6
3	509647174	1002	12
4	112457891	1290	3

Como filtrar dados usando strings de comparação

Nas declarações SELECT, você pode usar o operador LIKE para determinar um dado

caracter corresponde ao que foi especificado. Esses caracteres podem ser o (?) ou o (*) que representa um ou mais caracteres em uma busca.

Caractere de curinga faz o operador LIKE mais flexível no lugar do igual (=) e operadores de comparação.

A Sintaxe do LIKE

match_expression* [NOT] LIKE pattern** [ESCAPE escape_character]***

*Uma expressão SQL Server válida do tipo character string.

**Uma expressão SQL Server válida do tipo character string

***É colocado na frente do caractere curinga que indica que deve ser interpretado como um caractere regular e não como curinga. Não tem um valor default e um só caractere deve ser avaliado.

A seguir, a tabela dos caracteres curinga usados com o LIKE.

% - Substitui qualquer string de zero ou mais caracteres.

_ (underline) – Substitui qualquer caractere único.

[] – Substitui qualquer caractere único de uma determinada gama.

[^] – Substitui qualquer carácter único fora de uma determinada gama.

Exemplo:

```
USE AdventureWorks
SELECT ProductID, Name
FROM .Production.Product
WHERE Name LIKE 'Ch%'
```

ProductID	Name
952	Chain
324	Chain Stays
322	Chainring

```
USE AdventureWorks
```

```
SELECT ContactID, FirstName, LastName, Phone
FROM Person.Contact
WHERE FirstName like 'Di_k'
```

ContactID	FirstName	LastName	Phone
132	Dick	Brummer	581-555-0189
136	Dirk	Bruno	817-555-0114
260	Dick	Dievendorff	1 (11) 500 555-0193

```
USE AdventureWorks
```

```
SELECT EmployeeId, FirstName, JobTitle
FROM HumanResources.vEmployee
WHERE EmployeeID like '2[0-3]7'
```

EmployeeID	FirstName	JobTitle
207	Kitti	Production Technician - WC50
217	Michael	Research and Development Manager
227	Mary	Production Technician - WC10
237	Prasanna	Production Technician - WC20

```
USE AdventureWorks
```

```
SELECT ContactID, FirstName, Phone
FROM Person.Contact
WHERE firstname like 'da[^n]%'
```

ContactID	FirstName	Phone
55	Darrell	816-555-0118
57	David	752-555-0115

Como filtrar dados usando operadores lógicos

Você pode filtrar dados usando os operadores lógicos AND, OR e NOT. Esses operadores pegam as expressões lógicas como operando e retornam um valor booleano, que pode ser verdadeiro, falso ou desconhecido. Nas expressões lógicas você pode mudar a ordem de avaliação usando parênteses.

AND

Combina dois operandos lógicos. Esses operandos podem ser expressões de comparação ou expressões lógicas. Tem o formato geral.

predicado_1 AND predicado_2

Retorna os seguintes resultados.

Verdadeiro – se os dois operandos avaliados forem verdadeiros.

Falso – se um dos operandos for falso.

Desconhecido – se um operando é verdadeiro e o outro for desconhecido ou se os dois forem desconhecidos.

OR

Combina dois operandos lógicos. Esses operandos podem ser expressões de comparação ou expressões lógicas. Tem o formato geral:

predicado_1 OR predicado_2

Retorna os seguintes valores:

Verdadeiro – se um dos operandos forem verdadeiros

Falso – se os dois operandos forem falsos

Desconhecido – se um operando é falso e o outro é desconhecido ou se ambos forem desconhecidos.

NOT

Reverte o resultado de uma expressão de comparação ou uma expressão lógica. Tem o seguinte formato geral.

NOT predicado_1

Retorna os seguintes resultados:

Verdadeiro – se o operando retorna falso

Falso – se o operando retorna verdadeiro

Desconhecido – se o operando retorna desconhecido

Exemplo:

```
SELECT <columns>
```

```
FROM <table>
```

```
WHERE [NOT] boolean_expression AND/OR/[NOT] boolean_expression
```

Quando mais de um operador lógico é usado em uma declaração, a ordem das operações ou a ordem de procedência é NOT, AND e por final OR. Operadores aritméticos e operadores Bitwise precedem operadores lógicos.

Exemplos com operadores lógicos.

AND

```
SELECT ContactID, EmailAddress
```

```
FROM AdventureWorks.Person.Contact
```

```
WHERE FirstName='Alexia' AND LastName='Watson'
```

ContactID	EmailAddress
-----------	--------------

7062	alexa0@adventure-
------	-------------------

works.com

```
SELECT AddressID, AddressLine1, city
FROM Adventureworks.Person.Address
WHERE city='Bothell' OR city='Seattle'
```

AddressID	AddressLine1	City
13079	081, boulevard du Montparnasse	Seattle
859	1050 Oak Street	Seattle
110	1064 Slow Creek Road	Seattle
113	1102 Ravenwood	Seattle
95	1220 Bradford Way	Seattle
5	1226 Shoe St.	Bothell

NOT

```
SELECT CustomerID, FirstName, LastName, EmailAddress, City
FROM AdventureWorks.Sales.vIndividualCustomer
WHERE NOT city='Ballard'
```

CustomerID	FirstName	LastName	EmailAddress	City
11377	David	Robinett	david22@adventure-works.com	Solingen
11913	Rebecca	Robinson	rebecca3@adventure-works.com	Seaford
11952	Dorothy	Robinson	dorothy3@adventure-works.com	Geelong

As tabelas a seguir sobre verdadeiro e falso para os operadores AND, OR e NOT.

Input 1	Input 2	AND Result
True	True	True
True	False	False
False	False	False
False	True	False
UNKNOWN	UNKNOWN	UNKNOWN
UNKNOWN	True	UNKNOWN
UNKNOWN	False	False
True	UNKNOWN	UNKNOWN
False	UNKNOWN	False

Input 1	Input 2	OR Result
True	True	True
True	False	True
False	False	False
False	True	True
UNKNOWN	UNKNOWN	UNKNOWN

UNKNOWN	True	True
UNKNOWN	False	UNKNOWN
True	UNKNOWN	True
False	UNKNOWN	UNKNOWN

☞ ☞

Input	NOT Result
True	False
False	True
UNKNOWN	UNKNOWN

☞ ☞

Como retornar um intervalo de dados

Você pode retornar uma gama de dados especifica em banco de dados usando a clausula BETWEEN.

Essa clausula especifica a gama de dados a ser testada. EX:

Expressão_teste [NOT] BETWEEN expressão inicial AND expressão final

A clausula BETWEEN sempre retorna um valor booleano. Retorna Verdadeiro se a expressão teste for maior ou igual à expressão inicial e menor e igual à expressão final. O NOT BETWEEN retorna o contrario.

Todas as expressão tem de ser do mesmo tipo de dados.

```
SELECT SalesPersonID, TerritoryID, SalesQuota, Bonus
FROM AdventureWorks.Sales.SalesPerson
WHERE Bonus BETWEEN 2000 AND 5000
```

SalesPersonID	TerritoryID	SalesQuota	Bônus
275	2	300000.00	4100.00
276	4	250000.00	2000.00
277	3	250000.00	2500.00

☞ ☞

```
SELECT ContactID, FirstName, LastName, EmailAddress
FROM AdventureWorks.Person.Contact
WHERE ContactID BETWEEN 500 AND 550
```

ContactID	FirstName	LastName	EmailAddress
500	Elsa	Leavitt	elsa0@adventure-works.com
501	Michael	Lee	michael18@adventure-works.com
502	Marjorie	Lee	marjorie0@adventure-works.com

☞ ☞

Como retornar uma lista de valores

Você pode retornar uma lista especifica de valore usando a clausula IN. Ela determina valores que ligam qualquer valor em uma subconsulta ou uma lista, É equivalente ao OR e pode substituí-lo.

test_expression [NOT] IN

```
(
    sub query
    | expression [,...n ]
```

)

Nota: é recomendado que você evite usar o NOT IN porque impede que o otimizador de usar o índice para desempenhar a busca.

Exemplos:

```
SELECT ProductID, Name, ProductNumber, Color
FROM [AdventureWorks].[Production].[Product]
WHERE color = 'Black' OR color = 'Red' OR color = 'Silver'
SELECT ProductID, Name, ProductNumber, Color
FROM AdventureWorks.Production.Product
WHERE color IN ('Black', 'Red', 'Silver')
```

ProductID	Name	ProductNumber	Color
317	LL Crankarm	CA-5965	Black
318	ML Crankarm	CA-6738	Black
319	HL Crankarm	CA-7457	Black
320	Chainring Bolts	CB-2903	Silver

```
SELECT AddressID, AddressLine1, city
FROM AdventureWorks.Person.Address
WHERE city = 'Seattle' OR city = 'Bothell'
```

```
SELECT AddressID, AddressLine1, city
FROM AdventureWorks.Person.Address
WHERE city IN ('Seattle', 'Bothell')
```

AddressID	AddressLine1	City
13079	081, boulevard du Montparnasse	Seattle
859	1050 Oak Street	Seattle
110	1064 Slow Creek Road	Seattle
113	1102 Ravenwood	Seattle
95	1220 Bradford Way	Seattle
5	1226 Shoe St.	Bothell

```
SELECT ContactID, FirstName, EmailAddress
FROM AdventureWorks.Person.Contact
WHERE ContactID = 508 OR ContactID = 4700 OR ContactID = 9228 OR ContactID = 6045 OR
ContactID = 19943
```

```
SELECT ContactID, FirstName, EmailAddress
FROM AdventureWorks.Person.Contact
WHERE ContactID IN (508, 4700, 9228, 6045, 19943)
```

ContactID	FirstName	EmailAddress
508	A.	a0@adventure-works.com
4700	Aaron	aaron13@adventure-works.com
6045	Cassidy	cassidy21@adventure-works.com



Identificando os predicados que retornam uma lista ou um intervalo de dados

Ordene os valores tão depressa quanto possível nas categorias associadas deles/delas clicando no balde apropriado. Você também pode usar atalhos de teclado apertando 1 para o balde esquerdo e 2 para o balde certo.

Questões

Como DBA da Adventure Works você precisa obter os registros de estoque de um mês em particular fazendo combinações. Qual das seguintes combinações é verdadeira sobre os caracteres curingas usados na cláusula LIKE?

O caractere % encontra qualquer caractere único especificado.

O caractere [^] encontra qualquer caractere de string fora da gama especificada.

O caractere _ encontra qualquer caractere único especificado depois do WHERE. X

O caractere [] encontra qualquer caractere de string da gama especificada.

Você precisa recobrar informação sobre citações dado por vários vendedores, enquanto

considerando fatores como quantia citou, parte de mercado, e anos de associação com o negócio.

Para fazer isto, você precisa usar os operadores lógicos em sua questão. Identifique a ordem de avaliação correta de operadores quando mais que um operador lógico é usado em uma declaração?

AND, OR e então NOT

OR, AND e então NOT

NOT, AND e então OR X

AND , NOT e então OR

O gerente de finanças nos Trabalhos de Aventura lhe pediu preparar uma lista de empregados que recebem salários mais que um limite específico, com a finalidade de cálculo de imposto. Você precisa usar os operadores de comparação para obter esta informação. Quais das declarações seguintes são VERDADES sobre operadores de comparação?

Operadores de comparação podem ser usados somente em expressões de text,ntext, ou image.

Operadores de comparação testam se duas expressões são iguais ou não.X

Operadores de comparação retornam o resultado que tem o tipo image.

Operadores de comparação retornam valores booleanos.X

Operadores de comparação incluem operadores != e <>.X

Operadores de Comparação devem ser especificados no FROM.

Trabalhando com valores NULL

Linhas em uma tabela podem conter valores nulos. Você precisa localizar esses valores nulos em seu banco de dados. Precisa saber como esses valores trabalham para garantir que as consultas que você executa oferecem resultados exatos. Sistemas de Banco de Dados se diferem ao lidar com valores NULL e termos de ordenar dados com esses valores. Para lidar eficientemente com esses valores numa tabela você pode usar as funções ISNULL, NULLIF e COALESCE.

Considerações para se trabalhar com valores NULL

NULL indica que o dado é desconhecido, não aplicado o deve ser adicionado mais tarde. É diferente de um valor vazio ou zero. Comparações entre dois valores NULL ou entre um valor NULL e outro valor, retornará um valor desconhecido.

Comparando valores NULL

Quando se compara colunas com valores NULL em uma consulta, ela não retorna erro. Mas pode também não retornar algumas linhas se, a sua conexão ou seu banco de dados tiver a opção ANSI_NULL habilitada. Essa opção é previamente habilitada por conexões e bancos de dados. E quando ela está habilitada o padrão ANSI SQL-92 define que qualquer (=) ou (<>) deve calcular como falso quando comparado a um valor NULL. Quando ela está desabilitada o (=) e o (<>) não seguem o padrão SQL-92 e vai retornar os valores NULL e os valores não NULL. É melhor usar funções ao invés de confiar na opções fixas.

Valores NULL nas expressões

Quando você multiplica ou soma qualquer número com NULL, a função sempre retorna NULL. Em tais situações, você não irá receber o resultado esperado, como mostra o exemplo:

NULL + \$100000

NULL * 100

Ambos retornarão NULL.

Algumas Diretrizes

Para evitar erros nos resultados, você pode usar a função ISNULL, NULLIF e COALESCE. Para testar valores NULL numa consulta você pode usar o ISNULL ou IS NOT NULL no WHERE. Também você pode checar valores NULL nas consultas para realizar cálculos somente em linhas com dados válidos. A presença de valores NULL evita que as linhas sejam usadas como chave-primária ou chave-estrangeira.

USE AdventureWorks

SELECT Bonus, CommissionPct, SalesQuota

FROM Sales.SalesPerson

WHERE SalesQuota = NULL

USE AdventureWorks

SELECT Bonus, CommissionPct, SalesQuota

FROM Sales.SalesPerson

WHERE SalesQuota IS NULL

Bônus	CommissionPct	SalesQuota
0.00	0.00	NULL
0.00	0.00	NULL
0.00	0.00	NULL

Identificando valores NULL nas expressões

Ordene as expressões tão depressa quanto possível nas categorias associadas deles/delas clicando em o balde apropriado. Você também pode usar atalhos de teclado apertando 1 para o balde esquerdo e 2 para o balde certo.

Funções para trabalhar com valores NULL

ISNULL – substitui valores NULL com o valor especificado.EX:

ISNULL (check_expression , replacement_value)

Retorna o mesmo tipo de dado que o tipo de dado da expressão especificada.

NULLIF - Retorna o valor NULL de duas expressões equivalentes

NULLIF (expression , expression)

A primeira expressão pode ser uma constante, uma coluna, uma função, uma Subconsulta, uma combinação aritmética, Bitwise, ou operador string.

Retorna o mesmo tipo de dado da primeira expressão.

COALESCE – retorna a primeira expressão não nula entre os argumentos

COALESCE (expression [,...n])

A expressão de argumento é de qualquer tipo de dados. A expressão n é uma variável que significa que varias expressões podem ser especificadas, mas todas tem de ser do mesmo tipo.

Retorna o mesmo tipo de dado da expressão e se todos os argumentos forem NULL então retorna NULL.

Exemplo:

```
USE AdventureWorks;
SELECT AVG(ISNULL(Weight, 50))
FROM Production.Product
59.79 (1 row(s) affected)
```

Neste exemplo, a função de ISNULL é usada para achar a média do peso de todos os produtos. O ISNULL funciona para substituir o valor 50 por todas as entradas NULAS na coluna de Peso da mesa de Produto.

O exemplo de código seguinte recobra o StandardCost comum de produtos que pertencem à Classe 'L.'

```
SET NOCOUNT ON
USE AdventureWorks;
SELECT AVG (StandardCost) AS AvgCost
FROM Production.Product
WHERE Class='L'
Average Cost
-----
```

220.4349

Esta média inclui algumas filas que têm valores NULOS na coluna de StandardCost. Então, o valor de custo médio que é exibido no jogo de resultado está incorreto. Porém, a mesma questão pode ser reescrita com ajuda da função de NULLIF para assegurar que os produtos com NULO como StandardCost não são incluídos no cálculo.

```
AdventureWorks;
SELECT AVG(NULLIF(StandardCost, 0)) AS AvgCost
FROM Production.Product
WHERE Class='L'
```

AvgCost

240.2493

No banco de dados de AdventureWorks, considere a tabela CustomerPhone que contém as filas seguintes.

Customer_Name	Home_Phone	Office_Phone	Mobile_Phone
Thomas	0225872342	NULL	9870414716
Gary	NULL	NULL	982034452
Janet	NULL	099345522	NULL
Steve	232323452	334435454	NULL
Michelle	NULL	NULL	NULL

A anterior mesa contém vários números de telefone para um determinado cliente. Alguns clientes podem não ter todos os três números de telefone. O exemplo de código seguinte usa o FUNDA função para devolver o primeiro número de telefone não-nulo para um cliente particular e para clientes sem números de telefone válidos, o fio 'Nenhum Número Listado' é exibido.

```
SET NOCOUNT OFF
USE AdventureWorks
SELECT Customer_Name, COALESCE(Mobile_Phone, Office_Phone, Home_Phone, 'No Listed
Number') as Telephonefrom CustomerPhone
```





Customer_Name	Telephone
Thomas	9870414716
Gary	982034452
Janet	099345522
Steve	334435454
Michelle	No Listed Number



Questões

A Adventure Works vende produtos numa base de venda ou retorno. O gerente de vendas quer que você faça um relatório das vendas de um ano, incluindo vendas feitas nesta base. Portanto você precisa usar valores NULL para descrever vendas pendentes na base. Quais das seguintes afirmações são verdadeiras sobre valor NULL?

Valores NULL servem para distinguir uma linha da outra numa tabela.

Valores NULL podem ser usados para realizar cálculos somente em linhas com valores válidos. X

Um valor NULL multiplicado ou somado com qualquer número retorna NULL.X

Valores NULL indica que o dado é desconhecido, não aplicado, ou será adicionado mais tarde. X

NULL é igual a zero ou a vazio.

NULL comparado a (=) ou (<>) deve ser igual a verdadeiro quando o ANSI_NULL está habilitado

Qual das seguintes declarações são verdadeiras sobre a função ISNULL?

O ISNULL substitui valor NULL com zero.

O argumento de substituição do ISNULL não precisa ser do mesmo tipo de dados da primeira expressão.

O argumento da primeira expressão é expressão a ser checada se é NULL.X

A sintaxe do ISNULL é: ISNULL (replacement_value, expression).

Como DBA da Adventure Works você quer executar uma consulta que contém a descrição, contagem, porcentagem, qtde mínima, qtde máxima, para ofertas especiais. Para isso, você precisa da tabela Sales.SpecialOffer . O seu Select é:

```
SELECT Description, DiscountPct, MinQty, ISNULL(MaxQty, 0.00)
```

Qual é o resultado?

Se a quantidade de máximo para uma determinada oferta especial for NULA, o DiscountPct no jogo de resultado é 0.00.

Se a quantidade de máximo para uma determinada oferta especial for NULA, o MinQty no jogo de resultado é 0.00.

Se a quantidade de máximo para uma determinada oferta especial for NULA, o MaxQty no jogo de resultado é 0.00. X

Se a quantidade de máximo para uma determinada oferta especial for NULA, o MaxQty no jogo de resultado é 100.00.

Formatando Resultados

Os dados em suas tabelas poderiam não estar no formato que você requer e pode não ser útil para informar. Então, quando você quiser exibir dados em uma ordem específica, você pode formatar o resultado fixado combinando, dividindo, reordenando e convertendo os dados de acordo com suas exigências. Além disso, você pode formatar o resultado dos dados isto em ordem ascendente ou ordem descendente, elimine filas duplicadas em jogos de resultado e etiquete as colunas nos jogos de resultado.

Como Ordenar Dados

Ao recuperar dados do SQL Server, não há garantias de ordenação. Portanto, para ordenar os dados do resultados, você pode usar a cláusula ORDER BY, ordenando de forma ascendente ou descendente. A ordem padrão é ascendente e os valores NULL são os valores mais baixos possíveis. Na cláusula ORDER BY, você pode prescrever como você gostaria que a saída fosse ordenada especificando os nomes de uma ou mais colunas. O nome poderia ser um nome de coluna ou apelido de coluna, o que pode ser qualificado por uma tabela ou exibir nome, uma expressão, ou um número inteiro positivo que representa a posição do nome, apelido, ou expressão na lista seletione. Você também pode especificar múltiplas classificar colunas. A sequência do tipo colunas na cláusula ORDER BY define a organização do ordenado conjunto de resultados. O que se segue é a sintaxe da cláusula ORDER BY:

[ORDER BY { order_by_expression [ASC | DESC] } [,...n]]

Classificando dados potencialmente pode desacelerar a execução da consulta SQL Server, pois necessita de realizar uma etapa distinta para ordenar os dados. Portanto, é melhor usar o tipo função apenas quando necessário. Para otimizar o desempenho do SQL Server 2005, você pode usar índices para classificar os dados.

Como Eliminar Linhas Duplicadas

A palavra-chave DISTINCT elimina linhas duplicadas do resultado de um SELECT. Se você não usar o DISTINCT todas alinhas da tabela são retornadas.

Quando você inclui o DISTINCT em um SELECT somente um NULL é retornado independentemente de quantos valores NULL são encontrados. Porque valores NULL são considerados replica um do outro pelo DISTINCT.

Como rotular colunas no Resultado

Rótulos podem ser adicionados depois de cada expressão do SELECT. Um rótulo é um identificador SQL que se torna o nome da coluna no resultado. Eles ajudam a organizar a visualização. Se a consulta envolve muitas operações aritméticas, então o cabeçalho da coluna pode ficar complicado e pode retornar valores nulos ou desconhecidos.

SELECT <col1> AS 'AliasName'

SELECT 'AliasName' = <col1> - ANSI-SQL

Rótulos melhoram a legibilidade do resultado e são comumente usados por expressões nas views. O tamanho máximo que pode ter é de 128 caracteres.

Como usar strings (letras)

Para representar um determinado dado, você pode usar símbolos tanto como letras, valores constantes ou valores escalares. O formato de uma constante depende do tipo de dado que o valor representa. Você pode usar letras para gerar relatórios significativos e concatenar dados com mensagens amigáveis ao usuário. Também pode usar letras para criar declarações SQL dinâmicas no tempo de execução.

Para construir consultas com letras, você precisa concatenar strings e variáveis para oferecer relatórios significativos. No SQL Server 2005 você precisa usar as aspas simples (")

*caracteres alfanuméricos tanto de : a-z, A-Z, e 0 a 9.

*Caracteres especiais: (!), (@),(#)

String vazias são representadas por aspas duplas ""("" ") com nada dentro. Contudo, se um caractere string incluído em aspas simples contem aspas, então você precisa representar as aspas dentro de outras aspas.

A seguir os exemplos:

'Sydney'

'O' 'Hara'

"O'Hara"

'Process X is 50% complete.'

Como usar expressões

Você pode usar expressões como busca no WHERE do SELECT. Expressões devem retornar um valor escalar único se usado na lista do SELECT.

Manipulação de String

Você pode manipular expressões string quando quiser concatenar strings ou extrair parte de uma string. Ex:

USE AdventureWorks

SELECT UPPER(RTRIM(LastName)) + ', ' + FirstName AS Name

FROM Person.Contact

ORDER BY LastName;

Operação Matemática

Você pode fazer cálculos como adição, multiplicação, divisão, e modulo usando expressões matemáticas. Ex:

USE AdventureWorks

SELECT Name, ListPrice, SQUARE(ListPrice)

AS 'Square Value'

FROM Production.Product

Valor de sistema

Você pode usar funções de valor de sistema para comparar dados com valores de sistema.

Tanto de data corrente como de usuários logados. Ex:

USE AdventureWorks

SELECT LoginID FROM HumanResources.Employee

WHERE BirthDate < GETDATE()

Construções especiais

Você pode usar essas funções como CASE, CAST e COALESCE que ajudam a gerar o resultado esperado baseado nas condições oferecidas. EX:

USE AdventureWorks

SELECT ProductID, Name, COALESCE(color, size, CONVERT(nvarchar,weight),

'NA') AS Feature

FROM Production.Product

Construindo uma consulta que filtra e formata dados

Você é o DBA da Adventure Works e. O gerente de marketing, Kevin, pede sua ajuda para preparar alguns relatórios que o ajudarão a tomar algumas decisões sobre o preço dos produtos. Além do mais expressa interesse em aprender SQL para poder gerar relatórios no futuro. Você o ajuda.

Kevin: Obrigado. Como eu executo uma consulta na tabela de produção do banco de dados Adventure Works e o que é um esquema?

Você: Você pode usar a clausula FROM para executar consultas pela janela de consulta. O esquema para Products é Production.

Qual clausula FROM você usa?

From AdventureWorks. Product

Você: Eu esqueci algo. O SQL Server está apontando um erro.

From AdventureWorks. Production

Você: Eu esqueci algo. O SQL Server está apontando um erro.

From AdventureWorks.Production.Product

Você: eu devia ter usado ambas as tabelas, assim o SQL Server não apontaria erro.

Kevin: Bom. Nós temos o nome do banco, o nome do esquema e o nome da tabela. Agora, na primeira coluna, eu quero o ProductID de todos os produtos. Na segunda, eu quero os dados atuais do ListPrice. Porém eu quero renomear a coluna ListPrice, que contem a lista de preços atual como "OldPrice". Então eu quero adicionar \$10 a cada item na coluna ListPrice e mostrar o resultado numa terceira coluna chamada NewPrice.

Você: Sim, aqui está o SELECT necessário antes do FROM:

Qual você escolhe?

SELECT ProductID, ListPrice, (ListPrice + \$10) as New Price

SELECT ProductID, ListPrice as Old Price, (ListPrice + \$10) as New Price X

SELECT ProductID, ListPrice as Old Price, New Price

Kevin: É isso mesmo o formato que eu quero. Porém eu gostaria de saber a cor desses produtos.

Você: Sem problema. É só adicionar a coluna de Cor no nosso SELECT.

SELECT ProductID, ListPrice as Old Price, (ListPrice + \$10) as New Price, Color
FROM[AdventureWorks].[Production].[Product]

Kevin: Agora eu só quero a informação sobre os produtos de cores vermelhas e azuis.

Você: Claro, é só adiciona o WHERE

Qual você escolhe?

WHERE Color <> 'Green'

WHERE Color IN (Red, Blue)

WHERE Color IN ('Red', 'Blue') X

Kevin: Ótimo! Era isso que eu queria. Só preciso de mais uma coisa. Quero mostrar a coluna Weight também. Quero ordenar os resultados por peso com o maior peso no topo.

Você: Claro! É só incluir a coluna Weight no final do SELECT, assim:

SELECT ProductID, ListPrice as Old Price, (ListPrice + \$10) as New Price, Color, and Weight
FROM[AdventureWorks].[Production].[Product]
WHERE Color IN ('Red', 'Blue')

E então adicionar a clausula ORDER BY depois do WHERE.

Qual você escolhe?

ORDER BY Weight

ORDER BY Weight DESC X

Kevin: Ótimo! É o que eu queria. Você pode me dar uma copia dos detalhes?

Você: Claro.

Kevin: Muito obrigado por ajudar. É ótimo ter alguém como você na equipe.

Parabéns! Você completou a atividade!

Questões

Qual das seguintes afirmações são verdadeiras sobre o ORDER BY?

A palavra-chave DESC especifica que os valores são ordenados na coluna de forma ascendente

A expressão do order by é a linha que indica quais dados tem que ser ordenados. A palavra-chave ASC especifica que os valores são ordenados na coluna de forma ascendente.X

A ordem padrão do resultado é a descendente.

Qual das seguintes afirmações são verdadeiras sobre o DISTINCT?

O DISTINCT elimina linhas duplicadas do resultado do SELECT e mostra todas as linhas.

O DISTINCT considera valores NULL distintos uns dos outros.

O DISTINCT cria uma tabela de trabalho para ordenar dados e executar uma consulta.X

O DISTINCT deve sempre ser incluído numa consulta.

Qual das seguintes afirmações são verdadeiras sobre apelidos de colunas?

Ajudam a organizar a entrada.

Reduzem a legibilidade da saída e são comumente usadas por expressões nas views.

O tamanho máximo que podem ter é 128 caracteres.X
Devem ser especificados com o AS.

Considerações de desempenho para escrever consultas

Para executar consultas eficientes, você precisa saber como o SQL as processa. Ele seleciona um caminho de acesso e prepara um plano de execução. O database engine usa um otimizador para automaticamente otimizar as consultas adquirindo informações sobre tamanho, a estrutura dos tipos, índices, a porcentagem dos dados retornados.

Baseado nestes dados e a estrutura de armazenamento subjacente, o otimizador baseado no custo nomeia um custo calculado em termos de dados devolvidos e I/O de disco físico para cada operação relacionada. Além disso, você pode usar índices para melhorar o desempenho de questões.

Como o SQL Server processa as Consultas T-SQL

Sempre que você executa uma consulta na linguagem DML tanto como T-SQL, a consulta passa por diferentes estágios como analisar a gramática, solucionar, aperfeiçoar compilar e executar.

Parse: A primeira fase que uma declaração de T-SQL sofre está analisando gramaticalmente.

Durante analisar gramaticalmente, SQL Servidor 2005 valida a Declaração de T-SQL. O processo de análise gramatical consiste em duas funções. Primeiro, o processo de análise gramatical é usado para conferir a questão entrante para sintaxe correta. Qualquer erro como erros ortográficos ou uso de palavras chaves incorretas está fora pontudo para o usuário na vidraça de Mensagens. Segundo, é usado para demolir a sintaxe em partes de componente que podem ser usadas pelo RDBMS. Então, estas partes de componente são armazenadas dentro uma estrutura interna como uma árvore.

Resolve: Na fase solucionando, SQL Servidor 2005 rearranja a questão na árvore de questão em um formato mais unificado. Servidor de SQL pode aceitar e pode solucionar várias combinações de declarações de T-SQL de usuários quando eles submeterem questões para recobrar dados.

Solucionar é o processo de unificar e transformar estas questões em um formato útil para otimização. O processo de padronização aplica um jogo de regras de manipulação de árvore a árvore de questão produzida usando o processo de análise gramatical. Porque estas regras são independentes dos valores de dados subjacentes, eles são satisfatórios para todos os operações de banco de dados.

Optimize: Durante a fase de otimização, SQL Servidor 2005 executa alguns otimizações sintaxes-baseadas como fazer exterior direito une equivalente a esquerda exterior une ou convertendo questões de substituto para fazê-los equivalente para unir.

Compile: Na fase de compilação, Servidor de SQL produz o melhor possível plano de execução de forma que a questão executa melhor. Porém, este plano de execução poderia não ser o plano ótimo. Servidor de SQL implementa o modelo baseado no custo para gerar o plano de execução. Primeiro, o Servidor de SQL valida se um plano trivial já estiver disponível. Se disponível, executa este plano. Caso contrário, Servidor de SQL provê uma questão mais simplificada, de forma que um plano trivial fica disponível. Se a questão não pode ser simplificada, então Servidor de SQL avalia planos mais complexos e escolhe o mais eficiente entre eles. Finalmente, Servidor de SQL executa uma otimização cheia quando todos os planos disponíveis é comparado durante um tempo estipulado. Então, provê o melhor possível plano achado durante o intervalo de tempo estipulado.

Execute: Depois que o plano de execução é compilado, o SQL Server executa a consulta nesse plano, Ele continua a usar o mesmo plano a menos que haja mudanças na consulta ou nos objetos dos quais a consulta é dependente.

Como o otimizador de consultas processa os argumentos de busca

SQL Server utiliza um motor de base de custo baseada query optimizer para otimizar automaticamente dados manipulação consultas que são criados e submetidos usando T-SQL. Esta otimização é realizado em três fases.

A primeira fase é a fase em que a consulta análise otimizador olha para cada cláusula presentes na consulta e determina se ele pode ser mais otimizado. Se possível, também se reparte a consulta

declarações em cláusulas otimizadas.

A próxima fase é o índice seleção fase em que, para cada otimizada cláusula, o otimizador verifica sistema de bases de dados mesas para ver se existe um índice associado que pode ser usado para acessar os dados.

A terceira fase é a aderir seleção fase em que o otimizador encontra uma forma eficiente para combinar a forma como as cláusulas aderir podem ser acessados.

Para realizar estes três passos, o otimizador precisa da pesquisa argumentos presentes na consulta e juntar os predicados. Baseado na pesquisa argumentos, o otimizador realiza a consulta e análise baseia-se no juntar predicados, o otimizador determina o melhor possível juntar condição.

Para ajudar a melhorar o desempenho do otimizador, você deve escrever correto pesquisa argumentos e junte as condições e os quadros índice adequadamente. Como desenvolvedores de dados, você precisa criar os índices direito à direita colunas.

O Database Tuning Advisor utiliza as mesmas informações e estatísticas que o otimizador usa para criar um plano de execução. Você pode usar esta ferramenta para obter orientações e dicas sobre como índice opções

Guias para escrever eficientes argumentos de busca

Ao conceber consultas, é importante para escrever adequado pesquisa argumentos, porque a consulta otimizador utiliza a pesquisa argumentos para determinar o melhor plano de consulta. Ao escrever pesquisa argumentos, você precisará seguir algumas orientações que lhe dará o melhor desempenho.

Curingas: Evite usar curingas, como líder LIKE '% um '. Se você usa a palavra-chave LIKE na cláusula WHERE, então você deve usar um ou mais caracteres líder na cláusula WHERE, por exemplo LIKE '% m'. Quando há um personagem líder na cláusula WHERE, a consulta otimizador utiliza um índice para realizar a consulta, assim, acelerar o desempenho e reduzir a carga sobre SQL Server. No entanto, se o principal personagem é um curinga, em seguida, a consulta otimizador não será capaz de usar um índice, e uma mesa varredura deve ser realizada, reduzindo assim o desempenho.

Condições: Use positivo condições, em vez de procurar condições negativas. Negativo pesquisa condições como NOT IN e NOT LIKE não são os ideais, pois impede o otimizador de consulta usando um índice para realizar uma pesquisa. Por exemplo, não use negativo pesquisa condições tais como, NOT IN ('Califórnia').

Expressões: Evite expressões que incluem nomes coluna em uma pesquisa argumento ou uma função em uma coluna. Além disso, evitar expressões que têm a mesma coluna em ambos os lados do operador, ou comparar contra uma coluna que não é uma constante. Estas expressões não são otimizadas e eles abrandar o desempenho da consulta. Por exemplo evitar expressões como, ONDE (Preço + \$ 10)> \$ 20.

Clausula HAVING: Não use a cláusula HAVING para filtrar os registros que podem ser filtradas utilizando a cláusula WHERE. No seguinte código exemplo, a cláusula HAVING filtra os registros que podem ser filtrada pela cláusula WHERE.

```
SELECT * FROM Production.Product  
GROUP BY Product Line  
HAVING ProductID >10
```

Se o SELECT declaração contém uma cláusula HAVING, em seguida, escreva a sua consulta de tal forma que a cláusula WHERE filtra a maioria dos registros. Isso porque a cláusula WHERE é executado em primeiro lugar, seguido da cláusula GROUP BY e, em seguida, finalmente, a cláusula HAVING. Se a cláusula WHERE é utilizada para eliminar o maior número possível linhas indesejadas e, em seguida, a cláusula HAVING terão menos trabalho a executar, aumentando assim o desempenho da consulta.

Identificando argumentos de busca eficientes

Classificar a pesquisa argumentos, o mais rapidamente possível para os seus associados categorias clicando no segmento adequado. Você também pode utilizar os atalhos do teclado pressionando 1 para a esquerda balde e 2 para a direita balde.

Questões

O gerente financeiro quer que você prepare um relatório sobre o custo de materiais baseado na tabela Stocks. RawMaterials. Você precisa fazer uma consulta usando o SQL Server. Qual o procedimento no qual o SQL Server processa a consulta?

No estágio de resolução, o SQL Server rearranja a consulta na árvore de consulta num formato mais padronizado. 2

No estágio de compilação, SQL Server produz a melhor execução possível para desempenhar melhor a consulta.4

No estágio de análise de gramática, o SQL Server checa a sintaxe e faz quebras. 1

No estágio de otimização, o SQL Server desempenha melhoras baseado na sintaxe.3

No estágio de execução, o SQL Server executa a consulta. 5

O gerente de RH pede para fazer um relatório. Sobre os salários pagos no ano anterior. Você precisa da tabela Employees.Salary para executar a consulta.Precisa também do otimizador para executar essa consulta de forma eficiente. Qual das seguintes afirmações são verdadeiras sobre o otimizador?

O Otimizador é usado pelo SQL Server para otimizar automaticamente as consultas DML

O Otimizador constrói as declarações dentro de componentes otimizados na fase de análise da consulta.

O Otimizador checa as tabelas de sistema para ver se há índices associados úteis para acessar os dados de cada cláusula, na fase de seleção de índice. X

O Otimizador desempenha a função join antes da seleção de índice ser completada.

O gerente de marketing pede para você preparar um relatório sobre o lucro dos novos produtos introduzidos durante o ano anterior. Qual das seguintes GUIDELINES vai te orientar enquanto você escreve a consulta.

Use o curinga seguido de um caractere da cláusula LIKE.

Use expressões que envolvam a mesma coluna em ambos os lados do operador.

Use argumentos de busca apropriados e condições de join na consulta depois de indicar as tabelas. X

Use o HAVING em uma consulta ao invés do WHERE.

O gerente de estoque precisa de um relatório os estoques que tenham atingido o nível de reorder. Você precisa executar uma consulta para preparar o relatório. Qual dos seguintes ARGUMENTOS de PESQUISA é o mais eficiente?

* LIKE '%ab'

* LIKE 'm ke' X

* SELECT * FROM Product HAVING Price >100

* SELECT*FROM Product WHERE (Price + 10) > 20

Desempenhando consultas básicas

Você é o administrador de dados em Adventure Works. O departamento de Recursos Humanos lida com o recrutamento, orientação, formação e bem-estar dos empregados da Adventure Works. O departamento de Recursos Humanos tem decidido enviar um cartão por e-mail para todos os contatos da empresa. Eles pediram-lhe para gerar um relatório que contém uma lista de todos os contatos da empresa e dos seus associados informações. O departamento de Recursos Humanos tem também planejado para combinarem um park-and-ride facilidade para os trabalhadores. Portanto, eles têm pediu-lhe para gerar um relatório que contém uma lista de todas as cidades onde os empregados viagens, agrupados de acordo com os serviços para os quais os funcionários trabalham. Para fazer isso, você precisa usar a pesquisa diferentes condições e SELECT declaração para filtrar e recuperar dados. Neste processo, você irá usar as funções para trabalhar com valores NULL e também formatar o resultado obtido conjuntos.

Além disso, um executivo de vendas requer sua ajuda para executar consultas e obtendo informações para clientes que necessitam de um determinado produto em uma determinada quantidade. Um programador júnior na sua organização necessita da sua ajuda na refinação queries. Portanto, para melhorar o desempenho do SQL Server processos, o que você precisa para reescrever

as consultas com base em orientações específicas. Finalmente, você irá obter os resultados pretendidos em um formato padronizado e gerar um relatório.

Lab. Revisão

Neste laboratório, que usou a declaração SELECT para recuperar dados da base AdventureWorks. Você usou a pesquisa diferentes condições de filtrar os dados recuperados. Você também trabalhou com valores NULL usando as funções NULL. Você usou a diferentes declarações ao formato do conjunto de resultados obtidos a partir da base de dados. Então, você reescreveu as consultas para melhorar o desempenho dos processos em SQL Server e, assim, reduzir o tempo levado pelo SQL Server para obter o conjunto de resultados.

Qual a cláusula que você vai usar para especificar o nome da tabela a partir da qual os dados têm de ser recuperados?

Você pode especificar o nome da tabela na cláusula FROM.

Quais operadores que você vai usar para selecionar produtos com um preço específico e um nome específico?

Você precisa usar o operador LIKE com o E cláusula.

Que operador que você vai usar para substituir os valores NULL em uma coluna de uma tabela com um valor determinado?

Você pode usar o operador ISNULL para substituir os valores NULL com um valor especificado.

Qual o operador e a cláusula que você vai usar na declaração SELECT para ordenar um resultado fixado em ordem crescente?

Você precisa usar o ASC operador com a cláusula ORDER BY para ordenar um resultado fixado em ordem crescente.

Qual o operador e a cláusula que você vai usar na declaração SELECT para ordenar um resultado fixado em ordem crescente?

Você pode usar a palavra-chave AS no SELECT para especificar uma tabela alias.

-----*****----- Agrupando e Resumindo Dados No SQL Server

Você pode querer agrupar ou resumir os dados quando você os consulta em um banco de dados. Para isso você pode usar as funções de agregação oferecidas pelo SQL Server 2005. Você pode criar funções de agregação customizadas na linguagem .NET mas não na T-SQL. Você pode usar as funções SUM, AVG, COUNT, MAX e MIN calcular os dados. Também pode usar as clausulas GROUP BY e HAVING e então usar as cláusulas ROLLUP e CUBE com a função GROUPING para agrupar dados e calcular valores para esses grupos . Além disso, você pode usar as funções de categoria para retornar valores das categorias de cada linha em uma partição. T-SQL oferece quatro funções de categorias: RANK, DENSE_RANK, NTILE e ROW_NUMBER. Os operadores PIVOT e UNPIVOT ajudam a somar os dados e criar relatórios de tabela.

Resumindo dados usando funções de agregação

O SQL Server oferece varias funções e também permite criar funções definidas por usuários. T-SQL oferece quatro tipos de função: rowset, aggregate, ranking e scalar. As funções agregadas, como SUM, AVG, COUNT, MAX, MIN e ajudá-lo a resumir os dados. Para além das funções built-in, você pode criar escalar e mesa-valorizada funções usando T-SQL declarações. Estas funções ajudam você a parametrizar vista ou melhorar a funcionalidade de um índice ver, definir uma coluna em uma tabela, definir uma restrição em uma coluna, ou substituir um procedimento armazenado.

Como usar as funções de agregação do SQL Server

AVG	Retorna a media de um grupo de valores AVG ([ALL DISTINCT] expression) A palavra ALL inclui todos os valores da coluna para cálculo, a palavra
-----	--

	DISTINCT inclui valores únicos e EXPRESSION pode ser um valor exato ou aproximado, exceto pelo tipo de dado BIT. O parâmetro ALL é padrão e a expressão não pode conter funções de agregação ou subconsultas. O tipo retornado é determinado pelo tipo do resultado avaliado pela expressão.
COUNT (expression)	Retorna o numero de itens de um grupo. COUNT ({ [ALL DISTINCT] [expression] * }) COUNT(ALL expression) avalia a expressão para cada linha de um grupo e retorna o numero de valores não-nulos. A expressão pode ser de qualquer tipo, exceto unique identifier, text, image, ou ntext e não pode ser uma função de agregação ou uma subconsulta. O tipo retornado é INT.
COUNT(*)	Retorna o numero de itens de um grupo incluindo valores nulos e duplicados. Tem a mesma sintaxe do COUNT acima, mas não aceita qualquer parâmetro e não pode ser usado com o DISTINCT. Conta cada linha separadamente, incluindo as linhas que contém valores nulos. Retorna valores do tipo INT.
MAX	Retorna o valor máximo de uma expressão. MAX ([ALL DISTINCT] expression) O DISTINCT é irrelevante com a função MAX. Uma expressão é uma constante, uma coluna ou uma função, e qualquer combinação aritmética, Bitwise e string. Para colunas de caracteres, a função MAX encontra o valor mais elevado na sequência. O retorno deste tipo de função é o mesmo que o da expressão.
MIN	É o contrario do MAX. MIN ([ALL DISTINCT] expression)
SUM	Retorna a soma de todos os valores, ou somente os valores do Distinct numa expressão. Pode ser usada somente com colunas numéricas. ALL inclui todos os valores e é padrão. DISTINCT especifica a soma de valores únicos. Retorna a soma dos valores da expressão no tipo de dado mais preciso.

Como usar funções de agregação com valores NULL

O Efeito dos valores NULL nas Funções de Agregação

Valores nulos podem fazer com que funções de agregação produzam resultados inesperados ou incorretos, porque estas funções ignoram esses valores no cálculo. Ex: se a coluna Weight de uma tabela tem 504 registros, nos quais 299 tem valores nulos, a media dessa coluna calculada pela função AVG pode retornar valor incorreto. Exemplo:

[USE AdventureWorks](#)

[SELECT AVG\(Weight\) AS 'AvgWeight'](#)

[FROM Production.Product](#)

[AvgWeight](#)

[74.069219](#)

O resultado acima é mostrado porque ao calcular a media, a função AVG considera apenas as 205 linhas que contem valores não nulos. Porém é esperado que ela considere todos os registros.

A função ISNULL

Você pode corrigir este problema usando a função ISNULL, como no exemplo.

[USE AdventureWorks](#)

[SELECT AVG\(ISNULL\(Weight,0\)\) AS 'AvgWeight'](#)

[FROM Production.Product](#)

AvgWeight

30.127361

Efeito dos valores NULL na função COUNT(*)

Esta função é uma exceção para este problema porque retorna o numero total de registros em uma tabela, independente dos valores nulos. Ex:

USE AdventureWorks

SELECT COUNT(*) AS 'COUNT'

FROM Production.Product

COUNT

504

Como implementar Funções de Agregação Customizadas

Essa demonstração mostra como implementar funções de agregação customizadas. O departamento de RH requer informação dos dias que todos os empregados trabalharam em cada turno. Isso requer a criação e execução de uma função de agregação customizada. Esta demonstração mostra-lhe como criar e executar um costume agregado função. Abra o Microsoft Visual Studio 2005. Como um primeiro passo no sentido de implementar funções agregadas personalizado, você precisará criar um novo projeto em Microsoft Visual Studio. Para isso, no menu Arquivo, aponte para Novo e, em seguida, clique em Project. A caixa de diálogo Novo Projeto aparece. Na caixa de diálogo, selecione o tipo de projeto como Visual C #.

Você também pode criar custo agregado funções em Visual Base. Neste caso, você precisará selecionar o Visual Base opção no Projeto tipos área. Em seguida, selecione o modelo. Especificar o nome do projeto como MyAgg e clique em OK. O MyAgg projeto é apresentado no Solution Explorer. Para criar funções definidas pelo usuário, você precisa primeiro criar um agregado personalizado usando o. NET code. Para isso, abra o arquivo Class1.cs no código vista. Suprimir o padrão código no arquivo Class1.cs.

Para criar uma custom agregadas usando existentes. NET Framework código, abra o SQL Server 2005 Books Online. No menu Ver, aponte para Web Browser, e, em seguida, clique em Procurar. Na caixa URL, digite o link para abrir o Invocar Cor-User Defined Functions Agregado página. O código exemplo na página de um usuário-definidos agregado que concatenates função de um conjunto de valores string tomada de uma coluna em uma tabela. Selecione o código relacionado com C # e colá-lo no código janela. Agora, você tem sido demonstrado como implementar o agregado função desta classe arquivo. Em seguida, renomeie o arquivo Class1.cs para AggregateFunction.cs.

Para gerar o. NET assembly, selecione Release na lista Solução Configurações na barra de ferramentas. Então, para compilar a assembléia, o botão direito do mouse em MyAgg Solution Explorer e, em seguida, clique em Recriar. Navegue para o bin / Release pasta no Solution Explorer e verificar se a montagem MyAgg.dll é criado. Abra SQL Server Management Studio para registrar o criou assembléia. Em seguida, conectar ao servidor.

Na barra de ferramentas, clique no ícone New Query. Tipo da declaração de usar o database AdventureWorks. Criar um conjunto chamado MyAgg. Execute o CREATE ASSEMBLÉIA consulta na janela de busca para criar o MyAgg assembléia. Agora, criar uma função agregada usando a declaração CREATE AGREGADA. Nesta declaração, perceberá que você precisará usar o registrado assembléia. Executa a declaração. Um agregado função é criada. Executa a declaração SELECT para usar a função agregada personalizado que você criou para o cálculo do agregado de dias de trabalho em todos os turnos. Essa demonstração revelou que a forma de implementar um costume agregado função.

No menu Iniciar, aponte para Todos os programas e, em seguida, aponte para Microsoft Visual Studio 2005, e clique em Microsoft Visual Studio 2005. O Microsoft Visual Studio 2005 aparecerá.

No menu Arquivo do Microsoft Visual Studio, aponte para Novo e, em seguida, clique em Project. A caixa de diálogo Novo Projeto aparece.

No Projeto tipos de área, selecione Visual C #.

No Visual Studio instalado templates área, selecione Class Library.

Na caixa Nome, selecione o nome padrão ClassLibrary1, prima DELETE, tipo MyAgg e, em seguida, clique em OK. O MyAgg projecto é apresentado no Solution Explorer. O Class1.cs arquivo é exibido no código vista.

Suprimir o padrão o código-fonte do arquivo Class1.cs. Clique no botão Minimizar.

Clique no botão Iniciar na área de trabalho, aponte para Todos os programas, aponte para Microsoft SQL Server 2005, ponto de Documentação e Tutoriais, clique em SQL Server Books Online. O SQL Server Books Online janela se abre.

No menu Ver, aponte para Web Browser, e, em seguida, clique em Procurar.

Na caixa URL, tipo ms-help: / / MS.SQLCC.v9/MS.SQLSVR.v9.en/denet9/html/5a188b50-7170-4069-acad-5de5c915f65d.htm, e pressione ENTER.

Selecione o código relacionado com C #, pressione CTRL + C para copiar o código do link. Clique no MyAgg - Microsoft Visual Studio janela. Em seguida, pressione CTRL + V para colar o código no arquivo Class1.cs.

Em Solution Explorer, o botão direito do mouse Class1.cs e, em seguida, clique em Renomear e digite AggregateFunction.cs. O arquivo Class1.cs é rebatizado como AggregateFunction.cs.

Na lista Solução Configurações na barra de ferramentas, clique em Release. Em Solution Explorer, o botão direito do mouse MyAgg o projeto e, em seguida, clique em Recriar. O MyAgg projeto é reconstruído.

Em Solution Explorer, clique no ícone Mostrar Todos os arquivos. Navegue para o bin / Release pasta. Clique no botão Minimizar.

Abra o Microsoft SQL Server Management Studio.

Na caixa de diálogo conectar-se ao Servidor, verificar se o tipo de servidor, nome Server, a autenticação e caixas foram preenchidos corretamente e, em seguida, clique em Connect para aceitar as configurações padrão.

Na barra de ferramentas, clique no ícone New Query. A nova consulta janela é exibida.

Na janela de busca, digite USE AdventureWorks

Na janela de busca, digite CREATE ASSEMBLÉIA MyAgg FROM 'C: \ Documents and Settings \ Estudante \ My Documents \ Visual Studio 2005 \ Projetos \ MyAgg \ MyAgg \ bin \ Release \ Myagg.dll'

Selecione as declarações, e, na barra de ferramentas, clique em Executar. O MyAgg montagem é criado.

Na janela de busca, digite CREATE AGREGADA MyAgg (@ input nvarchar (200)) RETURNS nvarchar (max) EXTERNO NOME MyAgg.Concatenate

Em seguida, selecione a declaração e, na barra de ferramentas, clique em Executar.

Na janela Consulta, o tipo `SELECT ShiftID, dbo.MyAgg (EmployeeID) AS 'DaysWorked' FROM HumanResources.EmployeeDepartmentHistory GROUP BY ShiftID` Em seguida, selecione a declaração e, na barra de ferramentas, clique em Executar.

Questões

O gerente de RH pede para fazer um relatório de fim de ano com a média dos salários pagos, número de empregados até a última data do mês, os salários mínimo e máximo pagos e a quantia de salários pagos durante o ano, da tabela `Employees.Salaries`. Qual das seguintes informações são verdadeiras sobre a consulta?

Você precisa incluir o `distinct` na função `AVG` para obter o valor correto da média de salário.

Você não precisa incluir o `distinct` na função `MAX` para obter o valor correto do Máximo

salário pago.X

Você não precisa incluir o `distinct` na função `COUNT(employeeID)` para obter o total correto de empregados.

Você não precisa incluir o `distinct` na função `SUM` para obter o valor total dos salários. X

Você precisa incluir o `distinct` na função `MIN` para obter o salário mínimo pago.

Qual das seguintes afirmações são verdadeiras sobre funções de agregação com valores nulos?

A função `AVG` leva em consideração os valores `NULL`.

A função `COUNT` é uma função que ignora os valores `NULL`.

A função `MAX` retorna o valor Máximo de valores não nulos em uma expressão.X

A Função `COUNT(*)` retorna o número total de registros excluindo os registros `NULL`.

Sua companhia FEA uma recente expansão. Então o gerente financeiro pede para fazer uma função de agregação customizada para facilitar o relatório anual de empregados. Qual a ordem correta dos passos para criar uma função de agregação customizada?

Criar uma agregação usando o código `.NET Framework`. 2

Registrar o assembly criado no `SQL Server Management Studio`.4

Criar um novo projeto no `Microsoft Visual Studio`.1

Gerar o assembly `.NET Framework` selecionando `release` no `Solution Configurations` na barra de ferramentas.3

Executar o `select` para usar a função criada.6

Criar uma função de agregação usando `CREATE AGGREGATE`.5

Resumindo dados agrupados

Quando você precisa gerar relatórios de resumo para dados agrupados, você pode usar o `GROUP BY`. Você pode dividir a tabela de acordo com as colunas especificadas, depois você pode aplicar as funções de agregação para cada coluna separadamente. Você pode resumir várias combinações usando os operadores `CUBE` e `ROLLUP`. O `CUBE` oferece agregações para todas as combinações de valores nas colunas selecionadas enquanto o `ROLLUP` oferece agregações para uma hierarquia de valores nas colunas selecionadas. Você também pode a ver o `COMPUTE BY`, que oferece compatibilidade com a versão anterior, para visualizar ambos os detalhes e resumos em um único `SELECT`.

Como usar o GROUP BY

Ele especifica os grupos nos quais as linhas de saída devem ser organizadas. Se funções de agregação são incluídas no `SELECT`, o `GROUP BY` calcula um valor para cada grupo. Quando o `GROUP BY` é usado, cada coluna fora da agregação da lista deve ser incluída no `GROUP BY`. A

expressão do GROUP BY deve conter exatamente a mesma quantidade e ordem da lista do SELECT.

[GROUP BY [ALL] group_by_expression [,...n] [WITH { CUBE | ROLLUP }]]

O GROUP BY aceita os seguintes argumentos:

ALL	Inclui todos os grupos e resultados, inclusive aqueles que não estão em nenhuma condição especificada na condição WHERE. Quando você não usa o ALL, o resultado não mostra os grupos para os quais as linhas não se qualificam. Também, quando o ALL é colocado valores NULL são retornados para o resumo das colunas dos grupos que não se encontram nas condições. Você não pode usar o ALL nos operadores CUBE e ROLLUP. O ALL só pode ser usado com a condição WHERE.
Group by expression	É uma expressão na qual o agrupamento é feito. Pode ser uma coluna ou uma expressão não agregada que se refere a uma coluna devolvida especificada no FROM.
ROLLUP	Especifica para além dos habituais linhas fornecidas pela cláusula GROUP BY, o resumo de linhas é introduzido no conjunto de resultados. Grupos estão resumidos numa ordem hierárquica, a partir do nível mais baixo no grupo ao mais alto.
CUBE	Retorna, além das linhas habituais do GROUP BY, uma linha para cada combinação possível de grupo e subgrupo.

Se você não colocar o ORDER BY, os grupos retornados pelo GROUP BY não ficam em uma ordem em particular. Porém, é recomendado que sempre se use o ORDER BY para especificar a ordem dos dados.

Limitações do GROUP BY

SQL Server utiliza estruturas temporárias para construir agregados para a cláusula GROUP BY. Ele armazena estas estruturas temporárias tempdb em um banco de dados. Porque este intermédio worktable é necessária para realizar consulta resultados intermédios, cada GROUP BY cláusula é limitado a 8060 bytes.

SQL Server utiliza estruturas temporárias para construir agregados para a cláusula GROUP BY. Ele armazena estas estruturas temporárias tempdb em um banco de dados. Porque este intermédio worktable é necessária para realizar consulta resultados intermédios, cada GROUP BY cláusula é limitado a 8060 bytes.

As colunas em uma cláusula GROUP BY são limitadas ao número de bytes por cláusula GROUP BY.

O número de colunas ou expressões em um GROUP BY ou GROUP BY COM CUBO COM ROLLUP declaração é limitado a 10.



O seguinte exemplo usa o código cláusula GROUP BY e ao AVG função de encontrar a média ListPrice de produtos de cada tamanho no Produto tabela. Você pode usar a cláusula ORDER BY para ter a média ListPrice exibidas no fim de tamanhos, que vão desde o menor tamanho para o mais alto. Se você não especificar a cláusula ORDER BY, o grupo retornou ao utilizar a cláusula GROUP BY não são devolvidos em nenhuma ordem particular.

USE AdventureWorks

SELECT Size, AVG(ListPrice) AS 'Average ListPrice'

FROM Production.Product



GROUP BY Size
ORDER BY Size



Size	Average ListPrice
NULL	31.5548
38	1531.8091
40	690.6781
42	1351.518
44	1181.5758
46	1495.77
48	1284.5664

O seguinte exemplo usa o código cláusula GROUP BY e da função COUNT para encontrar o número de empregados em cada departamento da EmployeeDepartment tabela. Você pode usar a cláusula ORDER BY para obter o número de empregados em cada departamento, na ordem de serviços.

USE AdventureWorks
SELECT COUNT(EmployeeID) AS 'Employee Count', Department
FROM HumanResources.vEmployeeDepartment
GROUP BY Department
ORDER BY Department




EmployeeCount	Department
5	Document Control
6	Engineering
2	Executive
7	Facilities and Maintenance

O seguinte exemplo usa o código cláusula GROUP BY para agrupar Cliente IDs de acordo com tipo de cliente e, em seguida, utiliza a função COUNT para contar o número de clientes em cada tipo de cliente. Você pode usar o operador como para especificar os nomes das colunas no resultado.

USE AdventureWorks
SELECT COUNT(CustomerID) AS 'CustomerID Count', CustomerType
FROM Sales.Customer
GROUP BY CustomerType

CustomerType	CustomerID Count
S	701
I	18484



O seguinte exemplo usa o código cláusula GROUP BY para agrupar Cliente IDs de acordo com tipo de cliente e, em seguida, utiliza a função COUNT para contar o número de clientes em cada tipo de cliente. Você pode usar o operador como para especificar os nomes das colunas no resultado.

USE AdventureWorks
SELECT Color,AVG(ListPrice) AS 'Listprice'
FROM Production.Product

```
WHERE ProductNumber = 'FR-R72R-58'  
GROUP BY ALL COLOR
```

```
Color ListPrice  
NULL NULL  
Black NULL  
Blue NULL  
Grey NULL  
Multi NULL  
Red 594.83
```

Como filtrar grupo de dados usando a Clausula HAVING

Você pode criar uma consulta para agrupar dados usando o GROUP BY e filtrá-los usando o HAVING, que é similar ao WHERE.

[HAVING < search_condition >]

No argumento search_condition dessa sintaxe você precisa especificar a condição de busca para o grupo ou função de agregação. O HAVING pode ser usado para buscar vários itens no banco de dados. Você também vai encontrar que o HAVING divide o resultado horizontalmente.

Para executar consultas eficientes, você deve usar WHERE, GROUP BY, HAVING na seguinte sequência:

O WHERE é usado para filtrar as linhas que resultam das operações que são especificadas no FROM.

O GROUP BY é usado para agrupar a saída do WHERE.

O HAVING é usado para filtrar as linhas do resultado do grupo.

Comparações entre HAVING e WHERE

O HAVING traz as condições do GROUP BY, enquanto o WHERE traz as condições do SELECT.

O HAVING é aplicado depois que a operação de agrupamento ocorre, enquanto o WHERE é aplicado antes da operação de agrupamento.

A sintaxe do HAVING pode conter funções de agregação, mas a sintaxe do WHERE não.

Tanto o WHERE quanto o HAVING podem se referenciar a qualquer coluna ou expressão dentro da consulta.

USE AdventureWorks

```
SELECT ProductID, SUM(OrderQty) 'Order Quantity'  
FROM [Purchasing].[PurchaseOrderDetail]  
GROUP BY ProductID  
HAVING SUM(OrderQty) < 50
```

ProductID OrderQuantity

```
464      45  
473      42  
461      33  
470      42
```

USE AdventureWorks

```
SELECT COUNT(ProductSubcategoryID) AS Count, ProductCategoryID  
FROM [Production].[ProductSubcategory]  
GROUP BY ProductCategoryID  
HAVING COUNT(ProductSubcategoryID) = 14
```

Total Quantity	LocationID
186	3
958	30
110	4

Como os operadores ROLLUP e CUBE trabalham

O operador CUBE gera um resultado em um cubo multidimensional. Um cubo multidimensional é uma expansão dos dados, ou os dados que registram eventos individuais. Essa expansão é baseada em colunas que usuário quer analisar. Essas colunas são chamadas dimensões. Um cubo é um conjunto de resultados que contem uma junção de tabelas de todas as combinações possíveis das dimensões. O operador CUBE é especificado no GROUP BY de um SELECT. A lista deste SELECT contem as colunas de dimensão e as expressões das funções de agregação. O GROUP BY especifica as colunas de dimensão usando as palavras WITH CUBE.

O resultado contem todas as combinações possíveis dos valores nas colunas, junto com os valores agregados nas linhas que ligam essas combinações de valores de dimensão.

O operador ROLLUP é útil para gerar relatórios que contem valores agregados. O ROLLUP gera um resultado similar ao do CUBE. Porém, a diferença entre eles é que o CUBE gera um resultado que mostra os valores agregados para todas as combinações das colunas selecionadas. Em contraste, o ROLLUP retorna somente o resultado de uma combinação específica.

O ROLLUP gera um resultado que mostra as agregações por uma hierarquia de valores nas colunas selecionadas. Também oferece somente um nível de resumo, por exemplo, a soma cumulativa numa tabela.

Como usar os operadores ROLLUP e CUBE

GROUPING: No resultado de uma consulta, valores NULL podem estar presentes nos dados ou podem ser gerados pelo CUBE e pelo ROLLUP. Você pode usar o GROUPING para diferenciar os valores nulos gerados por esses operadores e os valores nulos padrões. O GROUPING retorna 0 se o valor NULL está presente no dado atual e retorna 1 se o valor NULL distingue entre valores de detalhe e de resumo no resultado. A seguir, a sintaxe parcial: GROUPING (column_name) <OVER Clause>

Você pode escrever a declaração SELECT com a função GROUPING para substituir a string ALL no lugar de qualquer NULL gerado. O SELECT com o GROUPING podem ser escritos para retornar a string UNKNOWN (desconhecido) no lugar de qualquer valor NULL.

[USE AdventureWorks](#)

```
SELECT ProductID,Shelf,SUM(Quantity) AS QtySum
FROM Production.ProductInventory
WHERE ProductID<6
GROUP BY ProductID,Shelf WITH ROLLUP
```

ProductID	Shelf	QtySum
-----------	-------	--------


1	A	761
1	B	324
1	NULL	1085

[USE AdventureWorks](#)

```
SELECT ProductID,Shelf,SUM(Quantity) AS QtySum
FROM Production.ProductInventory
WHERE ProductID<6
GROUP BY ProductID, Shelf WITH CUBE
```

ProductID	Shelf	QtySum
1	A	761
1	B	324
1	NULL	1085
2	A	791
2	B	318
2	NULL	1109
3	A	909

```
USE AdventureWorks
SELECT ProductID,Shelf,SUM(Quantity) AS QtySum
  Grouping(Shelf) AS 'Shelf Grouping'
FROM Production.ProductInventory
WHERE ProductID<6
GROUP BY ProductID, Shelf WITH CUBE
```



ProductID	Shelf	QtySum	Shelf Grouping
1	A	761	0
1	B	324	0
1	NULL	1085	1
2	A	791	0
2	B	318	0
2	NULL	1109	1

Como usar a Cláusula COMPUTE

Você pode usar o COMPUTE para gerar um resumo adicional de linhas num formato não relacional. Essa clausula não é um padrão ANSI e é compatível com versões anteriores.

Segue a sintaxe: `[COMPUTE{ { AVG | COUNT | MAX | MIN | SUM } (expression) } [,...n] [BY expression [,...n]]]`

Os argumentos AVG,COUNT, MAX, MIN e SUM especificam a agregação a ser desempenhada. A expressão de argumento se refere a uma expressão, tanto como o nome de uma coluna onde é feito o calculo. A expressão BY gera os subtotais e os controles de quebra em um resultado.

Os resultados de uma consulta que incluem o COMPUTE são similares a um controle de quebra de relatório. Nesses relatórios, grupos ou quebras que você especifique controlam os valores de resumo. Usando o COMPUTE, você pode produzir valores de resumo por grupos e pode calcular mais de uma função de agregação para o mesmo grupo.

Resultados do COMPUTE

Quando o COMPUTE é usado, há dois resultados para cada grupo. O primeiro mostra para cada grupo as linhas de detalhe do select daquele grupo. O segundo mostra uma linha contendo o subtotal das funções de agregação especificadas.

Usando a palavra BY com a Cláusula COMPUTE

Um COMPUTE BY permite que você veja os detalhes e o resumo das linhas com um SELECT. Ele calcula a linha especificada para cada coluna. As colunas listadas depois do COMPUTE BY devem ser idênticas ou devem ser um subconjunto indo à esquerda ou a direita do ORDER BY.Se você usar o COMPUTE BY tem também que usar o ORDER BY. E deve configurar para que o resultado apareça no modo texto para testar.


```
USE AdventureWorks
SELECT SalesOrderID, UnitPrice, UnitPriceDiscount
FROM Sales.SalesOrderDetail
ORDER BY SalesOrderID
COMPUTE SUM(UnitPrice), SUM(UnitPriceDiscount)
```

SalesOrderID	UnitPrice	UnitPriceDiscount
53569	334.0575	0.15
53569	334.0575	0.15
53569	54.942	0.00
53569	200.052	0.00
...
SUM	SUM	
7543846.2688	84.32	

```
USE AdventureWorks
SELECT PurchaseOrderID, OrderQty, ProductID, UnitPrice
FROM Purchasing.PurchaseOrderDetail
COMPUTE AVG(UnitPrice)
```

PurchaseOrderID	OrderQty	ProductID	UnitPrice
1	4	1	50.26
2	3	359	45.12
2	3	360	45.5805
...

AVG
34.8293

```
USE AdventureWorks
SELECT ProductID, Color, ListPrice
FROM Production.Product
COMPUTE SUM(ListPrice)
```

ProductID	Color	ListPrice
832	Black	348.76
833	Yellow	594.83
834	Yellow	594.83
...

SUM
221087.79

USE AdventureWorks
SELECT ProductID, Color, ListPrice
FROM Production.Product
ORDER BY Color
COMPUTE SUM(ListPrice) BY Color

ProductID	Color	ListPrice
923	NULL	4.99
946	NULL	46.09
947	NULL	91.57
994	NULL	53.99

SUM
4182.32

ProductID	Color	ListPrice
999	Black	539.99
989	Black	539.99
990	Black	539.99

SUM
67436.26

ProductID	Color	ListPrice
866	Blue	63.50
895	Blue	333.42
896	Blue	333.42

SUM
24015.66

Construindo uma consulta para resumo de Grupos de Dados

Você é o DBA da Adventure Works. Os dados na sua organização foram agrupados. Você precisa uma consulta para resumir os dados agrupados. Você vai construir dinamicamente um SELECT com um GROUP BY.

Você gostaria de filtrar os dados agrupados? SIM

Você quer resumir os dados em um nível? SIM

Você precisa saber quais linhas resumidas foram criadas pelo ROLLUP? SIM

Você gostaria de filtrar os dados agregados? SIM

Você gostaria de ordenar o resultado? SIM

Use a seguinte sintaxe: SELECT [GROUPING()]FROM...[WHERE...]GROUP BY ...[WITH ROLLUP]
[HAVING...][ORDER BY...]

Classificando dados agrupados

Para analisar dados efetivamente, você pode classificar os registros de uma tabela e oferecer valores classificados para o resultado de uma consulta. O SQL Server 2005 oferece quatro novas funções de classificação, chamadas RANK, DENSE_RANK, ROW_NUMBER e NTILE. Essas funções não somente ajudam a classificar os dados mas também facilitam bastante a execução de subconsultas e relatórios.

O que é classificar?

Você pode precisar classificar os dados de uma coluna para efeitos de apresentação, paginação, pontuando, ou fornecendo histogramas. Por exemplo, você pode precisar de classificação de um conjunto de alunos com base em suas marcas, classificar uma lista de empresas com base em suas ações valores, ou atribuir ao resultado seqüencial inteiros linhas para melhor apresentação fins. Para realizar estas tarefas, você pode usar o ranking funções no SQL Server 2005.

SQL Server 2005 prevê quatro funções ranking. O ROW_NUMBER função, o NTILE função, a função RANK e a DENSE_RANK função. Funções ranking são não determinísticas.

`<function_name>() OVER(
[PARTITION BY <partition_by_list>]
ORDER BY <order_by_list>)`

Na sintaxe, o PARTITION BY cláusula determina como as linhas serão agrupadas por ranking. A cláusula ORDER BY determina a ordem das linhas dentro de cada partição. Se você não usar o PARTITION BY cláusula, SQL Server aplica o ranking função para definir o resultado como um todo. O OVER cláusula prevê uma forma de partição e condenar a linhas de um conjunto de resultados.

O ranking função pode ser especificado apenas em duas cláusulas de uma consulta, a saber, as cláusulas SELECT e ORDER BY.

O ranking funções de lhe proporcionar um método eficaz para analisar os dados. Além disso, o SQL Server otimizador precisa para digitalizar os dados da tabela apenas uma vez para calcular os valores. O otimizador varre a tabela especificada usando uma varredura de um índice colocado no classificar colunas. Alternadamente, o otimizador digitaliza os dados uma vez e, em seguida, ordena os dados se um índice adequado não é criado.

Como classificar os dados usando as funções de classificação

O RANK função retorna a classificação de cada linha dentro de uma determinada partição de um conjunto de resultados. Quando várias linhas são do mesmo nível, que são atribuídas ao mesmo valor. Posteriormente fileiras na mesma janela são objeto de uma classificação que reflete o número de linhas com uma maior classificação nessa janela.

`RANK () OVER ([< partition_by_clause >] < order_by_clause >)`

Nesta sintaxe, o partition_by_clause divide o resultado produzido pela cláusula FROM nas partições a que o RANK função é aplicada. O order_by_clause determina a ordem em que os RANK valores são aplicados ao linhas de uma partição.

Quando utilizar a função RANK, é possível produzir rankings não-consecutivos posições. Por exemplo, se duas linhas de uma janela do retorno do mesmo ranking valor e são atribuídos a classificação valor 1, o próximo ranking fila será atribuído o valor 3.

A função DENSE_RANK executa a mesma classificação como tarefa a função RANK, mas atribui valores consecutivos. Portanto, se duas linhas são atribuídas à classificação valor 1, o próximo ranking fila será atribuído o valor 2.

DENSE_RANK () OVER ([< partition_by_clause >] < order_by_clause >)

Nesta sintaxe, o `partition_by_clause` divide o resultado produzido pela cláusula FROM nas partições a que a função DENSE_RANK é aplicada. O `order_by_clause` determina a ordem em que os valores são aplicados DENSE_RANK para as linhas em uma partição.

Você pode usar a função ROW_NUMBER para voltar à posição ordinal de cada linha em um agrupamento dentro de um conjunto de resultados.

ROW_NUMBER () OVER ([<partition_by_clause>] <order_by_clause>)

Nesta sintaxe, o 'partition_by_clause' divide o resultado produzido pela cláusula FROM nas partições a que o ROW_NUMBER função é aplicada. O 'order_by_clause' determina a ordem em que o ROW_NUMBER valor é atribuído às linhas de uma partição.

A função NTILE divide as linhas de cada partição de um resultado em uma classificação baseada num dado valor. As linhas serão distribuídas igualmente nas partições. Porém, se o numero de linhas do resultado não for dividido igualmente, as linhas serão distribuídas em dois grupos de tamanho diferente com o maior grupo aparecendo primeiro.

Nesta sintaxe, a integer_expression é um numero inteiro positivo constante que especifica o numero de partes em que cada partição deve ser dividida. Esse numero pode ser do tipo bigint. O 'partition by' divide o resultado produzido pelo From em partes em que o rank é aplicado. O 'order by' determinar a ordem dos valores nas linhas da partição.

RANK

USE AdventureWorks

```
SELECT P.Name Product, P.ListPrice, PSC.Name Category,  
RANK() OVER(PARTITION BY PSC.Name  
ORDER BY P.ListPrice DESC)  
AS PriceRank  
FROM Production.Product P  
JOIN Production.ProductSubCategory PSC  
ON P.ProductSubCategoryID = PSC.ProductSubCategoryID
```

Product	ListPrice	Category	PriceRank
HL Mountain Handlebars	120.27	Handlebars	1
HL Road Handlebars	120.27	Handlebars	1
HL Touring Handlebars	91.57	Handlebars	3
ML Mountain Handlebars	61.92	Handlebars	4
HL Headset	124.73	Headsets	1
ML Headset	102.29	Headsets	2
LL Headset	34.20	Headsets	3

DENSE_RANK

USE AdventureWorks

```
SELECT P.Name Product, P.ListPrice, PSC.Name Category,  
DENSE_RANK()  
OVER(PARTITION BY PSC.Name ORDER BY P.ListPrice DESC)
```

AS PriceRank
 FROM Production.Product P
 JOIN Production.ProductSubCategory PSC
 ON P.ProductSubCategoryID = PSC.ProductSubCategoryID

Product	ListPrice	Category	PriceRank
HL Mountain Handlebars	120.27	Handlebars	1
HL Road Handlebars	120.27	Handlebars	1
HL Touring Handlebars	91.57	Handlebars	2
ML Mountain Handlebars	61.92	Handlebars	3
HL Headset	124.73	Headsets	1
ML Headset	102.29	Headsets	2
LL Headset	34.20	Headsets	3

ROW_NUMBER
 USE AdventureWorks
 SELECT ROW_NUMBER()
 OVER(PARTITION BY PC.Name ORDER BY ListPrice)
 AS Row, PC.Name Category, P.Name Product, P.ListPrice
 FROM Production.Product P
 JOIN Production.ProductSubCategory PSC
 ON P.ProductSubCategoryID = PSC.ProductSubCategoryID
 JOIN Production.ProductCategory PC
 ON PSC.ProductCategoryID = PC.ProductCategoryID

Row	Category	Product	ListPrice
1	Accessories	Patch Kit/8 Patches	2.29
2	Accessories	Road Tire Tube	3.99
1	Bikes	Road-750 Black, 44	539.99
2	Bikes	Road-750 Black, 44	539.99

NTILE
 USE AdventureWorks
 SELECT NTILE(3) OVER(PARTITION BY PC.Name ORDER BY ListPrice)
 AS PriceBand, PC.Name Category, P.Name Product, P.ListPrice
 FROM Production.Product P
 JOIN Production.ProductSubCategory PSC
 ON P.ProductSubCategoryID = PSC.ProductSubCategoryID
 JOIN Production.ProductCategory PC
 ON PSC.ProductCategoryID = PC.ProductCategoryID

PriceBand	Category	Product	ListPrice
1	Accessories	Patch Kit/8 Patches	2.29
1	Accessories	Road Tire Tube	3.99
2	Accessories	LL Road Tire	21.49

2	Accessories	ML Road Tire	24.99
3	Accessories	Sport-100 Helmet, Blue	34.99
3	Accessories	HL Mountain Tire	35.00
1	Bikes	Road-750 Black, 44	539.99
1	Bikes	Road-650 Red, 48	782.99
2	Bikes	Road-650 Red, 52	782.99
2	Bikes	Mountain-200 Black, 46	2294.99
3	Bikes	Mountain-200 Silver, 38	2319.99
3	Bikes	Road-150 Red, 56	3578.27



Funções de Classificação e suas Funcionalidades

Ordene as expressões tão depressa quanto possível nas categorias associadas deles/delas clicando no balde apropriado. Você também pode usar atalhos de teclado apertando 1 para o balde esquerdo, 2 para o balde mediano, e 3 para o balde certo.

Questões

Você é o DBA da Adventure Works. O gerente de produção pede para preparar um relatório das cotas dadas a varias companhias, classificando as cotas do menor custo para o maior. Para preparar esse relatório, você precisa usar os recursos do SQL Server. Qual das seguintes afirmações é verdadeira sobre classificação?

O ROW_NUMBER retorna a posição cardinal da linha no resultado.

O WHERE oferece como será a partição e a ordenação das linhas no resultado.

O NTILE divide linhas de uma tabela em partições. X

O PARTITION BY determina como as linhas são particionadas para classificação

Você foi encarregado de preparar um relatório de nível de desempenho dos empregados, baseado na ultima avaliação anual. Para isso, você precisa classificar os empregados do maior nível ao menor, Qual das seguintes afirmações é verdadeira sobre as funções de classificação?

Quando múltiplas linhas são iguais, elas são atribuídas a valores diferentes no resultado.

Você pode usar o ROW_NUMBER para retornar o numero de linhas em um resultado.

O DENSE_RANK produz uma classificação de posições consecutivas. X

Quando o numero de linhas no resultado não divide exatamente no número de partições, as linhas são distribuídas em três grupos de tamanho igual.

Criando Consultas de Matriz

O SQL Server oferece consultas de matriz novas que ajudam a criar relatórios para requerimentos de negócios comuns. Você pode usar essas consultas para resumir os resultados e mostrá-los numa planilha do Microsoft Excel. Você pode criar essas consultas usando as cláusulas PIVOT e UNPIVOT que são similares a declaração TRANSFORM do Access.

Como os operadores PIVOT e UNPIVOT trabalham

Ao usar os operadores PIVOT e UNPIVOT, você pode manipular uma expressão de valor de tabela em outra tabela.

O PIVOT roda uma expressão de valor de tabela tornando os valores únicos de uma coluna na expressão de múltiplas colunas na saída. Ele também pode realizar agregações sobre os restantes valores de uma coluna desejada no final da saída.

O UNPIVOT executa a operação inversa ao PIVOT pelas colunas rodadas de uma expressão de valor de tabela em valores de coluna. No entanto, UNPIVOT não é exatamente o inverso do PIVOT e não reproduz a expressão de tabela de valor original no resultado quando as linhas da tabela fonte são invertidas.

Como usar o PIVOT

Você pode gerar consultas de matriz nos quais os valores das linhas são convertidos em colunas.

SELECT select_list FROM table_expression

PIVOT aggregate_function (value_column)

FOR pivot_column

IN (<column_list>)

O aggregate_function pode ser usado para gerar os dados ou valores coluna no resultado. O Value_column é a coluna de table_expression que pode ser utilizado como argumento para aggregate_function. Ao usar pivot_column pode rodar as colunas do conjunto de resultados. Você também pode especificar explicitamente os valores em <column_list> à lista de valores de pivot_column a apresentar resultado colunas seguidas pela cláusula IN.

PurchaseOrderID	EmployeeID	VendorID
7	233	84
8	238	78
9	261	74
10	274	13
11	244	51
12	231	80
13	241	47
14	266	81
15	164	46
16	223	21
17	233	27
18	238	86
19	261	45
20	264	93
21	274	64
22	231	4
23	241	241
24	266	12
25	164	44
26	223	30
27	233	8
28	238	73
29	261	76
30	264	77

USE AdventureWorks

```
SELECT VendorID, [164] AS Emp1, [198] AS Emp2, [223] AS Emp3, [231] AS Emp4, [233] AS Emp5
FROM (SELECT PurchaseOrderID, EmployeeID, VendorID FROM Purchasing.PurchaseOrderHeader) p
PIVOT ( COUNT (PurchaseOrderID)
FOR EmployeeID IN ( [164], [198], [223], [231], [233] ) ) AS pvt
ORDER BY VendorID
```

VendorID	Emp1	Emp2	Emp3	Emp4	Emp5
1	4	3	5	4	4
2	4	1	5	5	5

3	4	3	5	5	4
4	4	2	5	5	4
5	5	1	5	5	5

USE AdventureWorks

```
SELECT EmployeeID, [2001] AS Y2001, [2002] AS Y2002, [2003] AS Y2003, [2004] AS Y2004
FROM (SELECT EmployeeID, YEAR(OrderDate) AS OrderYear, Freight
FROM Purchasing.PurchaseOrderHeader) AS ORD
PIVOT
( SUM(Freight)
FOR OrderYear IN([2001], [2002], [2003], [2004])
) AS PVT
```

EmployeeID	Y2001	Y2002	Y2003	Y2004
261	NULL	10483.27	32854.9343	73992.8431
238	17.3345	9745.1001	37836.583	100106.3678
264	NULL	10337.3207	37170.1957	82406.4474
241	221.1825	6865.7299	35559.3883	114430.983
198	NULL	5344.4771	14963.0595	45020.9178
244	5.026	5689.4571	35449.316	74690.3755
233	1467.1388	9590.7355	32988.0643	98603.745
164	509.9325	14032.0215	34605.3459	105087.7428
231	6.8025	9603.0502	37604.3258	75435.8619
274	NULL	1877.2665	13708.9336	41011.3821
223	365.7019	12496.0776	37489.2896	117599.4156
266	4.2769	9588.8228	38533.9582	

Como usar o UNPIVOT

Ele desempenha o inverso do PIVOT enquanto roda as colunas em linhas. Porém os valores nulos não aparecem na saída.

```
SELECT select_list
FROM table_expression
UNPIVOT ( value_column
FOR unpivot_column IN ( <column_list> ) )
```

Value_column é criado no resultado gerado pelo operador UNPIVOT que pode segurar os valores no unpivot_column. The unpivot_column é a coluna de table_expression cujos valores que pretende rodar em resultado fileiras. Column_list é a lista de valores de unpivot_column que pretende converter em linhas seguidas pela cláusula IN.

USE AdventureWorks

```
CREATE TABLE pvt (VendorID int, Emp1 int, Emp2 int, Emp3 int, Emp4 int, Emp5 int)
INSERT INTO pvt VALUES (1,4,3,5,4,4) INSERT INTO pvt VALUES (2,4,1,5,5,5)
INSERT INTO pvt VALUES (3,4,3,5,4,4) INSERT INTO pvt VALUES (4,4,2,5,5,4)
INSERT INTO pvt VALUES (5,5,1,5,5,5)
SELECT VendorID, Employee, Orders
FROM (SELECT VendorID, Emp1, Emp2, Emp3, Emp4, Emp5 FROM pvt) p UNPIVOT (Orders FOR
Employee IN (Emp1, Emp2, Emp3, Emp4, Emp5) )AS unpvt
```


VendorID Employee Orders

1	Emp2	3
1	Emp3	5
1	Emp4	4
1	Emp5	4
1	Emp1	4
2	Emp1	4
2	Emp2	1
2	Emp3	5
2	Emp4	5
2	Emp5	5

USE AdventureWorks

SELECT SalesPersonID, Fullname, [2002], [2003], [2004] FROM Sales.vSalesPersonSalesByFiscalYears

Sales PersonID	FullName	2002	2003	2004
275	Michael G Blythe	1951086.8256	4743906.8935	4557045.0459
276	Linda C Mitchell	2800029.1538	4647225.4431	5200475.2311
277	Jillian Carson	3308895.8507	4991867.7074	3857163.6331
278	Garrett R Vargas	1135639.2632	1480136.0065	1764938.9857
279	Tsvi Michael Reiter	3242697.0127	2661156.2418	2811012.715
280	Pamela O Ansman Wolfe	1473076.9138	900368.5797	1656492.8626
281	Shu K Ito	2040118.6229	2870320.8578	3018725.4858
282	José Edvaldo Saraiva	2532500.9127	1488793.3386	3189356.2465
283	David R Campbell	1243580.7691	1377431.3288	1930885.5631
285	Jae B Pak	NULL	5287044.3125	5015682.3751
286	Ranjit R Varkey Chudukatil	NULL	1677652.4369	3827950.2378
287	Tete A Mensa Annan	NULL	883338.7107	1931620.1835
289	Rachel B Valdez	NULL	NULL	2241204.0424
290	Lynn N Tsoflias	NULL	NULL	1758385.926

USE AdventureWorks

SELECT * FROM

(SELECT SalesPersonID, [2002], [2003], [2004] FROM Sales.vSalesPersonSalesByFiscalYears) AS d
UNPIVOT

(
Sales FOR SalesYear IN ([2002],[2003],[2004])
) AS unpvt

SalesPersonID Sales

SalesYear

275	1951086.8256	2002
275	4743906.8935	2003
275	4557045.0459	2004
276	2800029.1538	2002
276	4647225.4431	2003
276	5200475.2311	2004
285	5287044.3125	2003
285	5015682.3751	2004
286	1677652.4369	2004
286	3827950.2378	2004
287	883338.7107	2003
287	1931620.1835	2004
289	2241204.0424	2004

Questões

O gerente financeiro pede para preparar um relatório sobre o numero de pedidos de compra no período de um ano. Para isso, você precisa de um relatório de matriz com a ajuda da tabela.

PurchaseOrderHeader. Você escreve a seguinte consulta usando o operador PIVOT.

```
SELECT VendorID, [164] AS Emp1, [198] AS Emp2, [223] AS Emp3, [231] AS Emp4, [233] AS Emp5 FROM (SELECT PurchaseOrderID, EmployeeID, VendorID FROM Purchasing.PurchaseOrderHeader) p PIVOT ( COUNT (PurchaseOrderID) FOR EmployeeID IN ( [164], [198], [223], [231], [233] ) ) AS pvt ORDER BY VendorID
```

Qual das seguintes afirmações é verdadeira no resultado da consulta?

Os VENDORIDs se tornam os nomes de coluna no resultado.

Os resultados são agrupados na coluna VendorID.

Se uma operação UNPIVOT é usada nessa consulta, irá retornar a tabela original.

EmployeeIDs se tornam nomes de coluna no resultado. X

O gerente de vendas pede para preparar um relatório de numero de pedidos de cliente por produto durante o mês atual. Para isso você precisa usar o operador PIVOT. Qual das seguintes sentenças é a sintaxe correta do operador PIVOT?

SELECT * FROM <table_source>FOR <pivot_column> IN <pivot_list>PIVOT

SELECT * FROM <table_source>PIVOT IN <pivot_list>FOR <pivot_column>

SELECT *FROM <table_source>PIVOT <aggregate function (value_column)>FOR

<pivot_column>IN <pivot_list> X

SELECT * FROM <table_source>PIVOT FOR <pivot_column>IN <pivot_list>

Lab. Agrupando e Resumindo Dados

Você é administrador de dados do Adventure Works. O departamento de RH quer que você prepare um relatório sobre o número de horas de folga dos vice-presidentes da empresa, e outro relatório sobre a lista de empregados com endereços completos. O gerente de estoque requer três relatórios sobre o número médio de horas de trabalho necessário para fabricar um produto, a quantidade final e a lista de preço de cada produto, bem como o nível de estoque de capacetes de segurança vermelho, azul, preto. Além desses pedidos, o gerente de marketing quer que você prepare um relatório sobre os dias por ano de venda de vendedores e classificar os seus nomes baseado no seu desempenho.

Exercícios

Inicie o ambiente de máquina virtual e complete os exercícios de laboratório seguintes:

Exercício: Resumindo dados agregados Usando Funções.

Neste exercício, você irá criar consultas para obter as informações sobre o total de horas de uso por deixar a vice-presidentes da empresa e também gerar o endereço da lista de dados AdventureWorks.

Exercício: Resumindo Dados Agrupados.

Neste exercício, você vai escrever as consultas para calcular e encontrar a média de dados dado. Você também vai distinguir entre os valores NULL gerada pelos operadores de resumo e os valores NULL em dados reais, utilizando a função GROUPING.

Exercício: Classificando dados agrupados

Neste exercício, você irá usar funções para classificar os dados agrupados.

Exercício: Criando consultas de matriz usando os operadores PIVOT e UNPIVOT

Neste exercício, você irá criar consultas de matriz usando as cláusulas PIVOT e UNPIVOT.

Lab. Revisão

Neste laboratório, você resumiu os dados, utilizando as várias funções agregadas. Você recuperadas férias, o número de horas de Vice-Presidentes da sua empresa. Você também gerou um relatório que constou do número total de empregados cujos endereços completos estão disponíveis utilizando o COUNT, COUNT (*), e COUNT (DISTINCT) funções. Em seguida, você resumiu dados agrupados usando o agrupamento de várias funções, tais como GROUP BY e GROUP BY ALL. Você também analisou como usar a cláusula HAVING, o CUBO operador e o operador ROLL UP com o agrupamento função. Após isto, você classificou os dados agrupados por usar o ranking funções como RANK, DENSE_RANK, ROW_NUMBER, e NTILE. Por último, que usou o PIVOT e UNPIVOT cláusulas para criar consultas crosstab.

Que uma das cláusulas você vai usar se você deseja listar apenas as únicas linhas de uma tabela?

Você irá usar a cláusula DISTINCT lista única para as linhas de uma tabela.

Operador que você vai usar com a cláusula GROUP BY se pretende resumir os dados em todos os níveis?

Você irá usar o CUBO operador para resumir os dados em todos os níveis.

Que uma das funções que você irá usar para classificar uma lista de produtos consecutivamente?

Você usará a função DENSE_RANK para classificar os produtos consecutivamente.

Você deseja criar uma consulta crosstab . É o operador PIVOT especificado juntamente com a cláusula GROUP BY?

Não, o operador PIVOT não é especificado, juntamente com a cláusula GROUP BY.

-----*****-----

Unindo dados de múltiplas tabelas no SQL Server 2005

Consultando múltiplas tabelas usando joins

Quando você executa consultas no SQL Server 2005, você muitas vezes precisa recuperar os dados de várias tabelas e bancos de dados. Para fazer isso, você pode usar os diversos tipos de join, como inner join, outer join, e cross join. Quando usar cross join você precisa estar consciente do impacto potencial do produto cartesiano das tabelas envolvidas na cross join. Além disso, você deve ser capaz de classificar declarações T-SQL baseadas em aderir tipos.

O que são Joins?

Joins são utilizados para se obter os dados de duas ou mais tabelas baseadas na lógica das relações entre as tabelas. Ao usar join você pode especificar como o Database Engine SQL Server 2005 deve utilizar os dados de uma tabela para selecionar as linhas de outra tabela. Você pode executar

diversos tipos de join em SQL Server e juntar tabelas. Para melhorar a legibilidade das consultas quando utilizar join, você deve seguir as regras e as melhores práticas. Você também precisa se assegurar que as normas ANSI SQL são respeitadas quando se utiliza join. O que se segue é a sintaxe para uma simplificada cláusula join SQL-92 FROM.

FROM first_table join_type second_table [ON (join_condition)]

Nessa sintaxe a expressão join_type especifica a maneira que é feito o join, se é inner join, outer join ou cross join. A expressão join_condition define o predicado a ser avaliado para cada par de linhas que são juntadas.

Realizando JOINS

Você pode especificar a aderir condição que define a maneira pela qual duas tabelas estão relacionadas, utilizando dois métodos. Em primeiro lugar, é necessário especificar a coluna de cada tabela que tem de ser utilizada no join.

Então, você pode especificar um operador lógico igual (=) ao operador ou um não igual ao operador a ser utilizado na comparação dos valores das colunas. Na maioria dos casos, você precisará especificar uma chave estrangeira de uma tabela e seus associados chave primária na outra tabela.

No entanto, se você tiver uma chave composta, então você deve lembrar-se de incluir todas as colunas na tabela.

Tipos de JOIN

Os joins podem ser classificados em inner join, outer join, e cross join. Quando você especifica a condição como um inner join, os padrões ANSI SQL recomendam que você especifique a condição na cláusula FROM ao invés de especificar na cláusula WHERE da instrução SELECT.

Isso ajuda a separar a condição de join de qualquer outra condição pesquisa, que pode ser especificada na cláusula WHERE. Inner join combina linhas de duas tabelas baseadas nos valores em comum entre elas.

Quando você especifica a condição de join em um outer join, no entanto, você pode especificar a condição só na cláusula FROM.

Neste caso, ao aderir condições especificadas na cláusula FROM são combinadas com as condições WHERE e HAVING para limitar as linhas que são selecionadas a partir das tabelas referenciadas baseadas na instrução SELECT.

Ao invés de inner e outer join, cross join retornam todas as linhas da tabela esquerda, isto é, cada linha da tabela à esquerda é combinada com todas as linhas da tabela à direita, e formam produto cartesiano.

Regras e melhores práticas

Quando você usa um join, você deve garantir que as duas colunas listadas na condição de join contem dados compatíveis, ou tipos de dados que o SQL Server pode converter implicitamente. Porém, os nomes, ou os tipos de dados de ambas não precisam ser o mesmo.

Por exemplo, se a tabela Contact contem a coluna ContactID e a tabela Orders contem a coluna CustomerID, então a tabela Contact e a tabela Orders podem ser juntadas. Isso porque as colunas contem o mesmo valor, mesmo tendo nomes diferentes.

Quando você lista condições de join, para fazer com que o código fique mais legível, você pode usar apelidos de tabela. Eles também ajudam a diferenciar colunas ambíguas que são comuns em join. Muitas declarações T-SQL que incluem subconsultas podem ser formatadas no modo de joins. Em T-SQL, normalmente não há diferença entre o desempenho uma declaração que contém uma subconsulta e uma declaração equivalente que contém um join.

Contudo, em alguns casos, onde há existência deve ser verificado, um join de desempenho do que uma subconsulta. Se você usar uma subconsulta, então qualquer consulta aninhada dentro de outra consulta deve ser tratada para cada resultado da consulta externa de garantir a eliminação das duplicatas.

Em tais situações, uma abordagem join dará melhores resultados. Além disso, você pode melhorar o

desempenho das consultas através da criação de índices sobre as colunas que fazem parte da condição join.

Como juntar dados de múltiplas tabelas usando INNER JOIN

Inner Join usa um operador de comparação para juntar linhas de duas tabelas baseadas nos valores comuns em colunas de cada quadro. Porque um inner join tipicamente retorna apenas as linhas para as quais existe um valor igual na coluna, o inner join é também chamado de um equi-join. O inner join é o tipo padrão no SQL Server 2005. Se um tipo de join explícito não é especificado no SELECT, então é assumido ser um inner join. O inner join é por vezes referido como um join natural. No entanto, não é estritamente um join natural porque ANSI SQL fornece um operador separado chamado NATURAL JOIN. Em um inner join, a cláusula WHERE contém tanto uma join e uma restrição e ambas as cláusulas usando o operador "=". No entanto, com o operador NATURAL JOIN, a informação de join não está presente na cláusula WHERE.

Restrições ao usar o INNER JOIN

De acordo com o padrão ANSI SQL-92, INNER JOIN pode ser especificado na declaração SELECT, no FROM ou na cláusula WHERE. No entanto, você deve evitar especificar o inner join na cláusula WHERE, pois não prevê uma distinção clara entre uma condição join e uma condição limitante ou pesquisa argumento (SARG).

Além disso, para um inner join, não se pode trabalhar com colunas que têm valores NULL. Isto porque se existem valores NULL nas colunas dos quadros sendo join e, em seguida, os valores NULL não correspondem uns aos outros. Portanto, os dados de uma coluna com valores NULL de uma das tabelas se juntaram podem ser recuperadas apenas utilizando um outer join.

Cláusula FROM e a Sintaxe INNER JOIN

FROM first_table INNER JOIN second_table ON (join_condition)

Nesta sintaxe, o join_condition define o predicado a ser avaliado para cada par de linhas juntadas.

USE AdventureWorks

```
SELECT e.EmployeeID, e.ManagerID, c.FirstName + ' ' + UPPER(c.LastName) AS Name, e.Title
FROM HumanResources.Employee AS e
INNER JOIN Person.Contact AS c ON e.ContactID = c.ContactID
```

EmployeeID	ManagerID	Name	Title
1	16	Guy GILBERT	Production Technician - WC60
2	6	Kevin BROWN	Marketing Assistant
3	12	Roberto TAMBURELLO	Engineering Manager
4	3	Rob WALTERS	Senior Tool Designer

USE AdventureWorks

```
SELECT S.Name AS STORE_NAME, SO.SalesOrderNumber, SO.OrderDate, SO.TotalDue
FROM Sales.Store AS S
JOIN Sales.SalesOrderHeader AS SO ON S.CustomerID = SO.CustomerID
ORDER BY S.Name, SO.OrderDate
```

STORE_NAME	SalesOrderNumber	OrderDate	TotalDue
A Bike Store	SO43860	2001-08-01 00:00:00.000	14603.7393

A Bike Store	SO44501	2001-11-01 00:00:00.000 26128.8674
A Bike Store	SO45283	2002-02-01 00:00:00.000 37643.1378
A Bike Store	SO46042	2002-05-01 00:00:00.000 34722.9906
A Great Bicycle Company	SO44125	2001-09-01 00:00:00.000 3450.9847

Como juntar múltiplas tabelas usando OUTER JOIN

LEFT OUTER JOIN

Inner join retorna registros somente quando há, pelo menos, uma linha de ambas as tabelas que corresponde à condição join. Inner join elimina as linhas que não correspondem com uma linha da outra tabela. Outer join, no entanto, retorna todas as linhas de pelo menos uma das tabelas ou views mencionadas na cláusula FROM, desde que essas linhas cumpram qualquer outra condição mencionada na cláusula WHERE ou cláusula HAVING. Existem três tipos de outer join.

USE AdventureWorks

SELECT st.Name AS Territory, sp.SalesPersonID

FROM Sales.SalesPerson sp

LEFT OUTER JOIN Sales.SalesTerritory st

ON sp.TerritoryID = st.TerritoryID

USE AdventureWorks

SELECT p.Name, pr.ProductReviewID

FROM Production.Product AS p

LEFT OUTER JOIN Production.ProductReview AS

pr

ON p.ProductID = pr.ProductID

ORDER BY pr.ProductReviewID DESC

Q1'

Territory	SalesPersonID
NULL	268
Northeast	275
Southwest	276
Central	277
Canada	278
Southeast	279
Northwest	280
Southwest	281
Canada	282
Northwest	283

Name	ProductReviewID
Road-550-W Yellow, 40	4
HL Mountain Pedal	3
HL Mountain Pedal	2
Mountain Bike Socks,	1

M

Mountain Bottle Cage	NULL
Mountain End Caps	NULL
Mountain Pump	NULL



RIGHT OUTER JOIN

Um right outer join retorna todas as linhas da tabela da direita especificada na cláusula RIGHT OUTER e junta apenas as linhas correspondentes da tabela direita. Valores NULL são devolvidos para as colunas da tabela esquerda, sempre que o resultado não tenha uma linha correspondente na tabela a esquerda.

USE AdventureWorks

```
SELECT st.Name AS Territory, sp.SalesPersonID
FROM Sales.SalesTerritory st
RIGHT OUTER JOIN Sales.SalesPerson sp
ON st.TerritoryID = sp.TerritoryID
```

Territory	SalesPersonID
-----------	---------------

NULL	268
------	-----

Northeast	275
-----------	-----

Southwest	276
-----------	-----

Central	277
---------	-----

Canada	278
--------	-----

Southeast	279
-----------	-----

Northwest	280
-----------	-----

FULL OUTER JOIN

O full outer join retorna todas as linhas em ambas as tabelas da esquerda e da direita. Qualquer momento uma linha não tem linha correspondente na outra tabela, a lista das colunas selecionadas na outra tabela contém valores NULL. Se existe uma correspondência entre as tabelas e, em seguida, o resultado das linhas contém os dados de ambas as tabelas base.

USE AdventureWorks

```
SELECT p.Name, sod.SalesOrderID
FROM Production.Product p
FULL OUTER JOIN Sales.SalesOrderDetail sod
ON p.ProductID = sod.ProductID
```

Name	SalesOrderID
------	--------------

Tension Pulley	NULL
----------------	------

Rear Derailleur Cage	NULL
----------------------	------

HL Road Frame - Black, 58	NULL
---------------------------	------

HL Road Frame - Red, 58	NULL
-------------------------	------

Sport-100 Helmet, Red	43665
-----------------------	-------

Como juntar dados de múltiplas tabelas usando o CROSS JOIN

Um cross join, sem uma condição de join, produz o produto cartesiano das tabelas envolvidas no join. Um produto cartesiano é uma coleção de cada linha da tabela sobre o lado esquerdo, a

participar, juntamente com todas as linhas da tabela sobre o lado direito do join. O tamanho de um conjunto de resultado cross join é o número de linhas na primeira tabela multiplicado pelo número de linhas na segunda tabela.

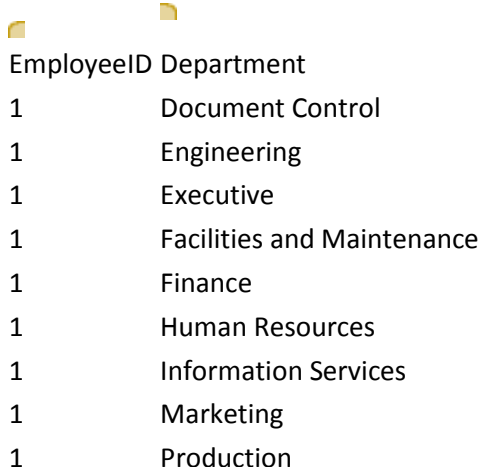
Um cross join ou um join irrestrito normalmente gera um grande conjunto de resultado, que é em grande parte inútil. Além disso, para manter o grande conjunto resultado que é gerado na execução cross join, o SQL Server 2005 precisaria criar estruturas temporárias no banco TempDB. Isso aumenta disco I / O e degrada o desempenho global do SQL Server 2005.

Ao usar cross join, você pode gerar dados de teste ou amostra. Por exemplo, para gerar cerca de 10000 nomes para uma tabela de amostra Cliente, você pode construir três pequenos quadros: FirstName, MiddleName, e LastName, com 25 linhas cada. Ao se associar estas três tabelas utilizando o operador CROSS JOIN, o conjunto resultado contém 15625 nomes únicos que você pode inserir na tabela Customer.

Você também pode usar cross join para gerar estruturas de apoio a outras consultas. Por exemplo, para descobrir quantos pedidos foram colocados por produtos todos os dias do ano de 2004, você pode usar um cross join para gerar uma tabela temporária com o intuito de obter informações para cada dia do ano, e, em seguida, fundi-la com as vendas. A tabela SalesOrder com um outer join, obtém uma contagem do número de encomendas em cada dia do ano.

USE AdventureWorks

```
SELECT e.EmployeeID, d.Name AS Department
FROM HumanResources.Employee e
CROSS JOIN HumanResources.Department d
ORDER BY e.EmployeeID, d.Name
```



EmployeeID	Department
1	Document Control
1	Engineering
1	Executive
1	Facilities and Maintenance
1	Finance
1	Human Resources
1	Information Services
1	Marketing
1	Production

Identificando o impacto potencial do produto cartesiano

Use os controles deslizantes para indicar o número de linhas em tabelas que você deseja aderir cruz, e depois ver a quantidade de espaço necessário para o produto cartesiano resultado fixado em tempdb. Mover uma alça deslizante para a esquerda ou direita (ou use o + e - keys) a alterar a cada variável e ver o resultado correspondente.

Classificando Declarações por Tipo de Join

Classifique as tarefas, o mais rapidamente possível para as suas categorias associadas clicando no segmento adequado. Você também pode utilizar os atalhos do teclado pressionando 1 para a esquerda balde, 2 para o meio balde, e 3 para a direita balde.

Aplicando Join para necessidades típicas de Relatórios

Para além dos tipos básicos join, existem certos tipos de join que são utilizados em complexos requisitos de informação. Você pode executar um join em várias tabelas se precisar de informações relacionadas à lista de todos eles. Você também pode executar um auto join. ou juntar uma tabela consigo mesma, fornecendo nomes alias para as instâncias da tabela. Além disso, você também pode realizar não-equi join a valores em duas colunas que não são iguais. Você pode usar os operadores, like, <, >, e OR ao juntar tabelas. Além disso, você pode juntar-se a uma linha de uma tabela para uma função definida por usuário.

Como juntar mais de duas tabelas

Você pode executar um join em várias tabelas para obter informações relacionadas a partir deles. Embora uma cláusula JOIN tipicamente junta apenas duas tabelas, uma cláusula FROM podem conter múltiplas condições de join. Assim, é possível associar qualquer número de tabelas que contêm os dados que precisa ser incluído no conjunto de resultados. No entanto, é preciso estar ciente de que une muitas tabelas em uma única consulta pode resultar na degradação do desempenho do SQL Server 2005.

USE AdventureWorks

```
SELECT P.Name AS Product, L.Name AS [Inventory Location],  
SUM(PI.Quantity) AS [Qty Available] FROM Production.Product AS P  
JOIN Production.ProductInventory AS PI ON P.ProductID = PI.ProductID  
JOIN Production.Location AS L ON PI.LocationID = L.LocationID  
GROUP BY P.Name, L.Name
```

Product	Inventory Location	Qty Available
HL Mountain Frame - Black, 38	Debur and Polish	148
HL Mountain Frame - Silver, 38	Debur and Polish	132
LL Mountain Frame - Black, 40	Debur and Polish	129
LL Mountain Frame - Black, 40	Debur and Polish	123
ML Mountain Frame - Black, 38	Debur and Polish	137

Questões

Você precisa preparar um relatório dos salários pagos aos empregados durante o ano anterior, pegando dados de conta das tabelas Humam.Resource.Employees e HumamResource.Salaries. A coluna EmployeeID é comum em ambas as tabelas. Você precisar usar o join. Quais das seguintes afirmações são verdadeiras sobre a condição join que você vai usar para obter o resultado desejado?

Você precisa fazer um join da coluna EmployeeID da tabela Salaries com a coluna EmployeeName da Tabela Employees

Você precisa fazer um join da coluna EmployeeID que é comum em ambas as tabelas. X

Você não pode especificar uma comparação de igualdade na condição join para descobrir os nomes dos empregados e seus respectivos salários.

Você pode só pode juntar as tabelas pela coluna EmployeeID se elas forem de tipos de dados compatíveis.X

Você pode melhorar o desempenho das consultas criando um índice na coluna EmployeeID. X

Você precisa especificar apelidos de coluna para as coluna EmployeeName e EmployeeID na condição de join.

O gerente de estoque pede para preparar um relatório um relatório da lista de preço e a quantidade de materiais solicitados para produção. Essa informação pode ser obtida das tabelas

RawMaterials.Quantity e RawMaterials.ListPrice. RawMaterial Name é a coluna que é comum entre ambas as tabelas. Você precisa fazer um Inner Join. Qual das seguintes afirmações é verdadeira sobre o inner join que você vai usar?

Você pode usar operadores lógicos para comparar o nome dos materiais nas tabelas.

Você pode recuperar todos os materiais que tem a mesma lista de preço na tabela ListPrice.

Você pode recuperar todas as linhas de ambas as tabelas que tem um valor de combinação para a coluna Name. X

Você pode recuperar todas as linhas de ambas as tabelas que tem valores NULL.

O gerente de RH pede um relatório baseado do nível de desempenho dos trabalhadores do turno da noite levando em conta as tabelas HumanResource.Employees e Employees.NightShift. Você tem que escrever um outer join que retornará todos os nomes dos empregados que trabalharam no turno da noite de 25 a 40 anos da tabela HumanResource.Employess e somente os nomes daqueles que trabalharam à noite. Qual das seguintes afirmações é verdadeira sobre o outer join que você vai usar para obter a informação?

Você vai fazer um LEFT OUTER JOIN e listar Employees.NightShift como tabela esquerda.

Você vai fazer um RIGHT OUTER JOIN e listar Employees.NightShift como tabela DIREITA.

Você vai fazer um LEFT OUTER JOIN e listar HumanResources.Employess como tabela esquerda. X

Você vai fazer um RIGHT OUTER JOIN e listar HumanResources.Employess como tabela direita.

Você tem que preparar um relatório de balanços do banco de todas as divisões da companhia. Esse será usado como dado modelo. Para preparar esse relatório você tem que recuperar todas as linhas da tabela direita para cada divisão sua companhia na tabela esquerda. Qual TIPO DE JOIN você vai fazer para obter o resultado?

Left outer join

Inner join

Cross join X

Full outer join

Como juntar uma tabela com ela mesma

Para recuperar registros de dados que estão relacionados a outros dados na mesma tabela, você pode criar uma auto-join ou juntar-se a uma mesa para si. Você pode usar auto join para comparar valores de colunas nas várias linhas da mesma tabela. Quando você usa o auto-join, você deve especificar a tabela aliases referência a várias instâncias da mesma tabela. Há dois cenários básicos envolvendo auto-join, ou seja, unir diferentes colunas da mesma tabela e ingressar na mesma coluna para si. Ao se associar a mesma coluna para si mesmo na mesma tabela, cada linha corresponde própria e pares são repetidos, resultando em duplicado e espelhados linhas. Em tais situações, você deve utilizar a cláusula WHERE para eliminar estas linhas duplicadas e espelhadas.

Em colunas diferentes

Você pode realizar uma auto-join em diferentes colunas. Por exemplo, a seguinte consulta utiliza uma auto-join para listar os gestores e os empregados dos relatórios que lhes HumanResources.Employee tabela. A auto-join é realizada no ManagerID e EmployeeID colunas.

USE AdventureWorks

```
SELECT A.EMPLOYEEID, A.LOGINID, B.EMPLOYEEID as ManagerID, B.LOGINID as  
ManagerLoginID
```

```
FROM HUMANRESOURCES.EMPLOYEE A
```

```
INNER JOIN HUMANRESOURCES.EMPLOYEE B
```

```
ON A.MANAGERID = B.EMPLOYEEID
```

```
ORDER BY A.EMPLOYEEID
```

Na mesma coluna

Você também pode realizar uma auto-join na mesma coluna. Por exemplo, a seguinte consulta utiliza a Person.StateProvince tabela de dar uma lista dos StateProvinces que têm o mesmo TerritoryID.

USE AdventureWorks

```
SELECT s.Name, s.TerritoryID, p.Name
```

```
FROM Person.StateProvince s
```

```
INNER JOIN Person.StateProvince p
```

ON s.TerritoryID= p.TerritoryID

WHERE s.Name<p.Name

Exemplos:

SELECT a.CustomerID, a.OrderDate, a.SalesOrderID, b.CustomerID, b.SalesOrderID

FROM Sales.SalesOrderHeader a

INNER JOIN Sales.SalesOrderHeader b

ON a.OrderDate = b.Orderdate

WHERE a.OrderDate='2001-07-01'

CustomerID	OrderDate	SalesOrderID	CustomerID	SalesOrderID
676	2001-07-01 00:00:00.000	43659	676	43659
676	2001-07-01 00:00:00.000	43659	117	43660
676	2001-07-01 00:00:00.000	43659	442	43661
676	2001-07-01 00:00:00.000	43659	227	43662
676	2001-07-01 00:00:00.000	43659	510	43663
676	2001-07-01 00:00:00.000	43659	397	43664
676	2001-07-01 00:00:00.000	43659	146	43665

USE AdventureWorks

SELECT a.CustomerID, a.OrderDate, a.SalesOrderID, b.CustomerID, b.SalesOrderID

FROM Sales.SalesOrderHeader a

INNER JOIN Sales.SalesOrderHeader b

ON a.OrderDate = b.Orderdate

WHERE a.OrderDate='2001-07-01' AND a.CustomerID < b.CustomerID

CustomerID	OrderDate	SalesOrderID	CustomerID	SalesOrderID
676	2001-07-01 00:00:00.000	43659	679	43677
676	2001-07-01 00:00:00.000	43659	21768	43697
676	2001-07-01 00:00:00.000	43659	28389	43698
676	2001-07-01 00:00:00.000	43659	25863	43699
676	2001-07-01 00:00:00.000	43659	14501	43700
676	2001-07-01 00:00:00.000	43659	11003	43701
117	2001-07-01 00:00:00.000	43660	676	43659
117	2001-07-01 00:00:00.000	43660	442	43661
117	2001-07-01 00:00:00.000	43660	227	43662
117	2001-07-01 00:00:00.000	43660	510	43663

USE AdventureWorks

SELECT DISTINCT pv1.ProductID, pv1.VendorID

FROM Purchasing.ProductVendor pv1

INNER JOIN Purchasing.ProductVendor pv2

ON pv1.ProductID = pv2.ProductID

AND pv1.VendorID < pv2.VendorID

ORDER BY pv1.ProductID

ProductID	VendorID
317	50
318	50
319	40
319	50
320	13
320	47
321	13
321	47
322	47
322	13

```
USE AdventureWorks
SELECT DISTINCT A.EmployeeID, A.Title
FROM HumanResources.Employee A
INNER JOIN HumanResources.Employee B
ON A.Title = B.Title
WHERE a.EmployeeID < b.EmployeeID
ORDER BY A.Title
```

EmployeeID	Title
178	Accountant
166	Accounts Payable Specialist
59	Accounts Receivable Specialist
94	Accounts Receivable Specialist
66	Application Specialist
102	Application Specialist
149	Application Specialist
164	Buyer
198	Buyer
223	Buyer

Como juntar tabelas usando Non-Equi Joins

A maioria das consultas envolvendo uma condição join usa um operador de igualdade entre duas tabelas. Portanto, elas são também referidas como equi-join. No entanto, é um equívoco comum que join entre dois quadros só pode ser estabelecido com a ajuda do operador de igualdade (=). Você também pode juntar duas colunas com a ajuda de outros operadores. Esses tipos de consultas são conhecidos como não-equi join. Pode escrever consultas que utilizam não-equi join, como comparação, a desigualdade, gama, de expressão, e de conversão. Além disso, os mesmos operadores e predicados utilizados para inner join podem ser utilizados para fins não-equi join.

Comparação

Os operadores de comparação podem ser utilizados para comparar os dados entre as duas tabelas. O seguinte exemplo usa um join de menor que (<) para encontrar ordens em que um determinado produto foi vendido a um UnitPrice superior a 2000.

```
USE AdventureWorks
SELECT DISTINCT p.ProductID, p.Name, p.ListPrice, sd.UnitPrice
FROM Sales.SalesOrderDetail AS sd
JOIN Production.Product AS p
ON sd.ProductID < p.ProductID
WHERE sd.UnitPrice > 2000.00
```

Desigualdade

O Non-Equi join (<>) é raramente utilizado. O seguinte exemplo usa um não-equi join combinado com um auto-join para encontrar todas as colunas na tabela ProductVendor em que duas ou mais linhas não têm o mesmo ProductID e VendorID números.

```
USE AdventureWorks
SELECT DISTINCT p1.VendorID, p1.ProductID
FROM Purchasing.ProductVendor p1
INNER JOIN Purchasing.ProductVendor p2
ON p1.ProductID <> p2.ProductID
WHERE p1.VendorID <> p2.VendorID
```

Range (Intervalo)

Intervalo é uma entidade sem fins equi join onde um intervalo é usado para juntar duas tabelas. O exemplo mostra como as colunas na tabela são Sales.SalesOrderHeader juntam-se à Production.Product tabela quando o DueDate no Sales.SalesOrderHeader tabela encontra-se no intervalo dos SellStartDate e SellEndDate no Production.Product tabela.

```
USE AdventureWorks
SELECT SalesOrderID, OnlineOrderFlag, DueDate
FROM Sales.SalesOrderHeader AS s
JOIN Production.Product p
ON s.DueDate BETWEEN p.SellStartDate AND p.SellEndDate
```

Expressão

Expressão não é uma equi join, é uma expressão que é avaliada e, em seguida, com base no que é feito o join. O seguinte é um exemplo da expressão não-equi join.

```
USE AdventureWorks
SELECT DISTINCT p.ProductID, p.Name, p.ListPrice, sd.UnitPrice AS 'Selling Price'
FROM Sales.SalesOrderDetail AS sd JOIN Production.Product AS p
ON (sd.ProductID = p.ProductID AND sd.UnitPrice < p.ListPrice)
WHERE p.ProductID = 718
```

Conversão

Conversão é uma entidade sem fins equi join onde uma conversão é feita para coincidir com o join de colunas. O que se segue é um exemplo de conversão não-equi join.

```
USE AdventureWorks
SELECT E.FirstName, E.LastName, V.FirstName AS VendorName
FROM HumanResources.vEmployee E
```

JOIN Purchasing.vVendor V
ON V.FirstName LIKE SUBSTRING(E.Firstname, 1,3)

Exemplos

USE AdventureWorks

```
SELECT p1.ProductID, p1.Name,
p2.ProductID, p2.Name,
p1.ListPrice, p2.ListPrice
FROM Production.Product p1
INNER JOIN Production.Product p2
ON p1.ListPrice > p2.ListPrice
WHERE p2.ListPrice > 0
ORDER BY p1.ListPrice, p1.ProductID, p2.ListPrice
```

ProductID	Name	ProductID	Name	ListPrice	ListPrice
922	Road Tire Tube	873	Patch Kit/8 Patches	3.99	2.29
870	Water Bottle - 30 oz.	873	Patch Kit/8 Patches	4.99	2.29
870	Water Bottle - 30 oz.	922	Road Tire Tube	4.99	3.99
921	Mountain Tire Tube	873	Patch Kit/8 Patches	4.99	2.29
921	Mountain Tire Tube	922	Road Tire Tube	4.99	3.99

USE AdventureWorks

```
SELECT soh.SalesOrderID, soh.TotalDue, cont.FirstName, cont.LastName
FROM Sales.SalesOrderHeader soh
INNER JOIN HumanResources.Employee emp
ON soh.SalesPersonID = emp.EmployeeID
INNER JOIN Person.Contact cont
ON emp.ContactID < cont.ContactID
WHERE soh.TotalDue > 200000
```

SalesOrderID	TotalDue	FirstName	LastName
46616	207058.3754	Michael	Blythe
46616	207058.3754	Tete	Mensa-Annan
46616	207058.3754	David	Bradley
46616	207058.3754	Wanida	Benshoof
46616	207058.3754	Kevin	Brown

USE AdventureWorks

```
SELECT SalesOrderID, OrderDate, TotalDue
FROM Sales.SalesOrderHeader soh
INNER JOIN Sales.Customer cust
ON soh.CustomerID = cust.CustomerID
WHERE CustomerType <> 'S'
```

SalesOrderID	OrderDate	TotalDue
--------------	-----------	----------

43697	2001-07-01 00:00:00.000 3953.9884
43698	2001-07-01 00:00:00.000 3756.989
43699	2001-07-01 00:00:00.000 3756.989
43700	2001-07-01 00:00:00.000 772.5036

Como usar Funções Definidas pelo Usuário

A entrada direita é avaliada para cada linha de entrada à esquerda e as linhas produzidas são combinadas para a saída final. A lista de colunas produzidas pelo operador APPLY é o conjunto de colunas da entrada esquerda e, em seguida, a entrada direita.

Os dois tipos de operador APPLY são CROSS APPLY e OUTER APPLY. O operador CROSS APPLY invoca uma tabela de valores de função de cada linha em uma tabela expressão exterior.

No entanto, se a tabela de valores de função retorna um conjunto vazio para uma determinada linha exterior e, em seguida, que a linha não é retornada no resultado exterior.

Em contrapartida, o operador OUTER APPLY também retorna os registros da tabela para o exterior, que a tabela de valores de função retorna um conjunto vazio. Valores NULL da coluna são retornados como valores que correspondem às colunas da tabela de valores de função.

Questões

Você é o DBA da AdventureWorks e precisa gerar um relatório de todos os empregados que são gerentes e também têm a mesma idade. Você precisa comparar os detalhes de cada empregado da tabela HumanResource.Employee para fazê-lo. Para também garantir que não serão retornadas as linhas duplicadas. Qual das seguintes opções você escolheria para obter o resultado esperado?

*Você precisa criar outra tabela, copiar dados do HumanResource.Employee tabela para a nova tabela, e então comparar as duas tabelas.

* Você precisa realizar uma auto-join HumanResource.Employee sobre a tabela em várias colunas e especificar as condições em limitar a cláusula WHERE para eliminar as linhas duplicadas. X

* Você precisa executar uma não-equi join à HumanResource.Employee tabela para recuperar linhas que não são duplicados.

*Você precisa realizar uma auto-join à HumanResource.Employee tabela na mesma coluna.

Você precisa gerar um relatório que contenha as vendas mais recentes. E está usando a tabela UDF em suas consultas. Para isso você precisa do operador APLLY. Quais das seguintes afirmações são verdadeiras sobre o operador APPLY?

Valores NULL são retornados quando é usado o CROSS APPLY.

CROSS APPLY e OUTER APLLY são duas formas de operadores APPLY.X

CROSS APPLY retorna linhas da tabela exterior para que a mesa-valorizada função retornou um conjunto vazio.

Ao usar o operador APPLY, você pode invocar uma mesa-valorizada função para cada linha retornada por uma tabela exterior expressão de uma consulta. X

O CROSS APPLY operador retorna um resultado unificado conjunto de todos os resultados retornados pelo indivíduo invocações da mesa-valorizada função. X

O OUTER APPLY operador não retornar uma linha se a mesa-valorizada função retorna um conjunto vazio para essa linha.

O gerente de RH de sua empresa quer que você se prepare um relatório sobre os trabalhadores recrutados entre os meses de Janeiro e Julho do ano passado e que obteve uma ordem de vendas de

US \$ 20.000. Para fazer isso, você precisa obter os dados da tabela HumanResource.Employee e os Sales.SalesOrderHeader tabela. Você precisa juntar as duas tabelas em um intervalo de valores. Qual das seguintes JOINS você vai usar para obter o resultado exigido?

Self – join

Inner – join

Non –equi join X

Natural Join

Combinando e Limitando Resultados

Você pode usar o operador UNION, EXCEPT, ou INTERSECT de combinar ou limitar os conjuntos de resultados de múltiplas consultas. O operador UNION combina os resultados de duas ou mais declarações SELECT em um único conjunto de resultados. O operador EXCEPT retorna quaisquer distintos valores da consulta à esquerda do operando EXCEPT que não são igualmente o direito de regressar de consulta, que o operador INTERSECT retorna distintas linhas que aparecem em ambos os conjuntos de resultados. É necessário identificar a ordem de precedência dos INTERSECT, EXCEPT, e UNION operadores. Você também pode limitar os conjuntos de resultados usando os operadores TOP e TABLESAMPLE.

Como Combinar Resultados usando o UNION

O que se segue é a sintaxe parcial do operador UNION.

Select_statement UNION [ALL] select_statement

Por padrão, o operador UNIÃO remove todas as linhas de duplicar o resultado conjunto. Esta função implícita DISTINCT pode abrandar o desempenho da consulta porque SQL Server tem de realizar uma segunda passagem sobre o resultado combinado definido para remover duplicatas.

O resultado de um conjunto UNIÃO coluna contém os mesmos nomes que foram especificados na primeira SELECT declaração na UNIÃO e ignora os nomes das colunas na segunda SELECT. Portanto, para se referir a uma coluna no resultado de um novo nome, a coluna devem ser referidos na mesma forma no primeiro SELECT.

Usando o UNION ALL

Quando você usa a palavra-chave ALL, o resultado contém todas as linhas, incluindo duplicatas. Se você tiver a certeza de que não existem duplicatas ou se a presença de duplicatas não afeta a sua finalidade, considere o uso da palavra-chave ALL, porque ela conduz a um melhor desempenho, evitando um segundo passo.

Ordem de Avaliação do UNION

A T-SQL pode ter qualquer número de operadores UNION. Por padrão, o SQL Server 2005 avalia uma declaração que contém UNIÃO operadores da esquerda para a direita. Então, você tem que usar parênteses quando você deseja especificar certa forma de avaliação.

Conversão de dados com o UNIÃO Operador

Todas as colunas e expressões do conjunto de resultados que estão sendo combinados com o operador UNIÃO deve ter uma implícita dados conversão possível entre os tipos de dados ou ter um compromisso explícito conversão oferecidos.

Por exemplo, um UNION entre uma coluna do tipo de dados datetime e um binário do tipo de dados não irá funcionar a menos que expressamente uma conversão é fornecido. No entanto, um UNIÃO irá funcionar entre uma coluna de dinheiro e um tipo de dados int tipo de dados, porque eles podem ser convertidos implicitamente.

Quando vários tipos de dados são combinados em um UNION operação, que são convertidos usando as regras do tipo de dados precedência.

Regras para usar o UNION

A seguir estão as regras para o uso do operador UNION.

O resultado estabelece que seja combinados usando o operador deve UNIÃO todos têm o mesmo número e a ordem das colunas. O resultado correspondente fixado colunas deve ter tipos de dados compatíveis.

A declaração SELECT que tem a UNIÃO cláusula não pode ter o seu próprio ou COMPUTE cláusula ORDER BY. Não pode ser apenas um ORDER BY ou COMPUTE cláusula após o último SELECT declaração e é aplicada ao final, combinado conjunto de resultados.

Você pode especificar as cláusulas GROUP BY e HAVING apenas no indivíduo SELECT declarações.

Exemplos

```
USE AdventureWorks
(SELECT ProductID, ListPrice AS Price
FROM Production.Product
UNION ALL
SELECT ProductID, UnitPrice
FROM Purchasing.PurchaseOrderDetail)
UNION
SELECT ProductID, UnitPrice
FROM Sales.SalesOrderDetail
```

Como limitar os resultados usando os operadores EXCEPT e INTERSECT

Você pode limitar os conjuntos de resultados usando o EXCETO e CRUZE operador, que são definidas pelos padrões ANSI SQL. Ao contrário do SQL Server 2000, SQL Server 2005 fornece uma cláusula implícita DISTINCT com o EXCETO e CRUZE operadores. Você também pode usar o NOT IN ou NOT EXISTS operador, em vez dos EXCETO e CRUZE operadores para produzir resultados similares.

`<query_expression> (EXCETO | CRUZE) <query_expression>`

As regras a seguir e as melhores práticas deveriam ser aplicados ao EXCETO e CRUZE operadores.

A primeira consulta determina os nomes das colunas no resultado.

O número e a ordem das colunas deve ser a mesma em ambas as consultas.

As colunas correspondentes devem ser compatíveis pela conversão implícita.

Não deve ser apenas uma cláusula ORDER BY que se aplica ao conjunto resultado da operação. A cláusula ORDER BY não pode ser especificada com consultas individuais.

Operador EXCEPT

O operador EXCEPT retorna quaisquer distintos valores da consulta sobre o lado esquerdo do EXCETO operador que não sejam devolvidos no lado direito. Diferentemente da UNIÃO operador, o operador EXCETO é assimétrica. Por exemplo, UNIÃO U V é logicamente equivalente ao V UNIÃO U. Contudo, EXCETO U V não é uma lógica equivalente do V EXCETO U.

Operador INTERSECT

O operador INTERSECT retorna distintas linhas que aparecem nas duas séries. O operador INTERSECT é semelhante ao INNER JOIN, que também retorna um resultado conjunto de linhas que aparecem

em ambas as tabelas especificadas na condição JOIN. Similar ao operador EXCEPT, o operador INTERSECT também retorna apenas linhas distintas. No entanto, o operador INTERSECT é um operador simétrico.

```
USE AdventureWorks
SELECT ProductID
FROM Production.Product
EXCEPT
SELECT ProductID
FROM Purchasing.PurchaseOrderDetail
USE AdventureWorks
SELECT ProductID
FROM Production.Product
INTERSECT
SELECT ProductID
FROM Purchasing.PurchaseOrderDetail
```

Identificando a ordem de precedência do UNION, EXCEPT e INTERSEPT

INTERSECT Avaliado primeiro. EXCEPT Avaliado por segundo. UNION Avaliado por ultimo.

Como Limitar Resultados Usando os operadores TOP e TABLESAMPLE

Quando você recupera resultados de múltiplas tabelas, você pode usar o operador TOP para retornar o primeiro conjunto de linhas de uma consulta. Esse resultado pode ser tanto um número ou uma porcentagem de linhas. De modo alternativo, você pode usar a cláusula TABLESAMPLE, que limita o número de linhas retornadas de uma tabela na cláusula FROM para retornar um número ou uma porcentagem de linhas modelo. Você tem que determinar quando usar os operadores TOP e TABLESAMPLE.

TOP: Permite retornar somente um número fixo de linhas do resultado. Pode ser um número ou uma porcentagem. A seguir a sua sintaxe parcial:

```
[
TOP (expression) [PERCENT]
[ WITH TIES ]
]
```

O TOP pode incluir a palavra chave WITH TIES, que especifica que linhas adicionais podem ser retornadas com base no resultado com o mesmo valor na coluna mostrada no ORDER BY como a última do TOP n(percentual) linhas. Essa palavra chave pode ser listada somente no SELECT e somente se existe um ORDER BY.

Diferente das versões anteriores do SQL Server onde o valor oferecido para o TOP tinha que ser uma constante, o SQL Server 2005 permite que você especifique qualquer expressão numérica que retorne um número. Você também pode optar por especificar variáveis e subconsultas. Além disso, o SQL Server 2005 permite que o TOP seja usado no SELECT, INSERT, UPDATE ou DELETE.

A expressão numérica tem que ser escrita entre parênteses. As constantes sem parênteses é aceita apenas pelo SELECT em compatibilidade com versões anteriores. A expressão tem que ser independente se você usar um subconsulta e a expressão não pode referenciar colunas da tabela exterior. Se você não especificar a opção PERCENT, será implicitamente convertido para o tipo bigint, se especificar, a expressão será convertida para float na gama de 0 a 100. A opção SET ROWCOUNT é usada para limitar o número de linhas afetadas pela consulta. Mesmo se essa opção oferecer os mesmos resultados quando aplicada as tabelas, é preferível usar TOP com SELECT. O otimizador de consulta pode usar o valor da expressão do TOP como parte do SELECT quando realiza a execução de uma consulta. Porque o SET

ROWCOUNT PE usado fora da declaração e o seu valor não pode ser usado para gerar um plano de execução para uma consulta.

Também é importante frisar que o uso do Set ROWCOUNT não afetará o DELETE, INSERT e UPDATE no próximo lançamento do SQL Server. Portanto evite usar o SET ROWCOUNT com o DELETE, INSERT e UPDATE para modificar aplicações que você usa atualmente. Modifique usando o operador TOP.

TABLESAMPLE

Limita o numero de linhas o retornadas de uma tabela na clausula FROM para um modelo de numero ou percentual de linhas. Não pode ser usado nas views. A seguir a sintaxe.

**TABLESAMPLE [SYSTEM] (sample_number [PERCENT | ROWS])
[REPEATABLE (repeat_seed)]**

É usado somente para teste, para retornar uma tabela modelo. Pode ser aplicado a qualquer tabela listada no FROM de uma consulta, exceto tabelas derivadas , tabelas de servidores de link e tabelas derivadas de tabelas de valores de função. Retorna uma porcentagem aproximada de linhas.

Você podem usar a opção SYSTEM junto com o TABLESAMPLE na implementação de métodos ANSI SQL. E só deve ser usado com algum conhecimento de modelagem. Por exemplo, juntar duas tabelas para retornar as linhas em comum. Contudo se o TABLESAMPLE for usado para qualquer uma das tabelas e em seguida algumas linhas retornadas da tabela não modelada não são suscetíveis de ter uma correspondência nas linhas da tabela modelo. Esta duplicação poderia levar você a suspeitar de uma consistência de dados problema nas tabelas subjacentes quando os dados são realmente válidos. Do mesmo modo, se TABLESAMPLE é especificado para ambas as tabelas que são unidas, a percepção do problema é susceptível de confundir o usuário.

Exemplos:

USE AdventureWorks

```
SELECT top 10 SalesOrderID, CustomerID, TotalDue  
FROM Sales.SalesOrderHeader
```

USE AdventureWorks

```
SELECT top 10 SalesOrderID, CustomerID, TotalDue  
FROM Sales.SalesOrderHeader  
ORDER BY TotalDue DESC
```

USE AdventureWorks

```
SELECT top 10 with TIES SalesOrderID, CustomerID, TotalDue  
FROM Sales.SalesOrderHeader  
ORDER BY TotalDue
```

USE AdventureWorks

```
SELECT SalesOrderID, CustomerID, TotalDue  
FROM Sales.SalesOrderHeader  
TABLESAMPLE (10 PERCENT)
```

Classificando Declarações que Limitam Resultados

Classifique as tarefas, o mais rapidamente possível para as suas categorias associadas clicando no segmento adequado. Você também pode utilizar os atalhos do teclado pressionando 1 para a esquerda balde, 2 para o meio balde, e 3 para a direita balde

Questões

O gerente de produção pediu um relatório sobre o retorno de aquisição do ano corrente. Você tem que limitar a quantia desse retorno para 10% ara facilitar a previsão. E precisa

retornar somente as 5 primeiras linhas do resultado. Qual das seguintes opções você vai usar?

O operador UNION com o operador PERCENT.

O operador EXCEPT com a opção SET ROWCOUNT.

O operador TOP com o operador PERCENT. X

A opção TABLESAMPLE com a opção SET ROWCOUNT.

Quais das seguintes declarações são verdadeiras sobre o EXCEPT e o INTERSECT?

O INTERSECT operador é semelhante a um OUTER JOIN.

O operador EXCEPT retorna distintos valores da consulta.

U EXCETO V é logicamente equivalente ao V EXCETO U. X

U Diferentemente da UNION operador, o operador EXCETO é assimétrica. X

O operador INTERSECT retorna todas as linhas que aparecem em ambos os conjuntos.

O INTERSECT operador é um operador simétrico. X

O gerente de RH pede um relatório de número de empregados por departamento, com os números da employee ID. Para isso você precisa combinar o resultado da Employee ID com a Department ID. E também terá que usar uma consulta que leve o menor tempo. Qual dos seguintes METODOS você vai usar?

Você irá usar o operador UNION e especificar explicitamente a cláusula DISTINCT na consulta.

Você irá usar o operador INTERSECT com a palavra-chave ALL na consulta.

Você irá usar o operador UNION ALL com a palavra-chave da consulta. X

Você irá usar o operador UNION com o padrão DISTINCT cláusula na consulta.

Como administrador de banco de dados Adventure Works, o que você precisa para preparar um relatório sobre os produtos produzidos no ano anterior, mas ainda não foi vendido. Para preparar este relatório, o que você precisa para comparar os Purchasing.PurchaseOrderDetail tabela com o Sales.SalesOrderDetail tabela. Você precisa obter todas as informações sobre os produtos que estão no Purchasing.PurchaseOrderDetail mas não são encontradas no Sales.SalesOrderDetail tabela. Você também precisará classificar os resultados pela quantidade do produto adquirido. Qual dos seguintes OPÇÕES que você escolher para obter o resultado exigido?

Você precisa usar o operador UNION com a cláusula ORDER BY.

Você precisa usar o operador INETRSECT com a cláusula ORDER BY.

Você precisa usar o operador com o EXCEPT cláusula ORDER BY. X

Você precisa usar o operador com o EXCEPT cláusula GROUP BY.

Exercícios Lab.

Você é o DBA da Adventure Works. Você foi solicitado pelos vários gestores de preparar os relatórios seguintes.

Lista de funcionários baseados em TerritoryID.

Lista de empregados contratados em 2001, os gestores para quem relatório, e as datas em que estes gestores foram contratados.

Lista das lojas manipuladas por cada vendedor.

Lista dos empregados que aderiram em 2001.

Lista de mulheres empregados que têm mais de 50 férias horas.

Lista das duas principais produtos ordenados no ano de 2002.

Lista dos top 10 por cento das vendas despachos do total de vendas encomendas recebidas durante o ano.

Lista das encomendas feitas pelos clientes, utilizando-se uma amostragem de 30 por cento dos dados no database AdventureWorks

Exercício: examinando várias tabelas utilizando junta

Neste exercício, você:

Executa uma consulta utilizando o ANSI JOIN e os T-JOIN.

Executa uma consulta usando o exterior esquerda e juntar-se o direito exterior join.

Exercício: combinando os conjuntos de resultados

Neste exercício, você:

Consulta várias tabelas utilizando INNER JOIN.

Consulta tabelas utilizando auto join e à cláusula DISTINCT.

Consulta tabelas utilizando-se um não-equi-join com um operador de igualdade e, em seguida, um operador de não-igualdade.

Consulta tabelas utilizando o não-equi join, com o operador IN e uma expressão

Exercício: limitar os conjuntos de resultados

Combine o resultado conjuntos de duas consultas usando o operador UNION ALL.

Limitar o resultado conjuntos com a cláusula EXCEPT, a cláusula INTERSECT, o operador TOP, e o operador TABLESAMPLE.

Revisão

Neste laboratório, que criou um relatório sobre o emprego relacionado com detalhes dos vendedores da Adventure Works. Você fez isso ao executar uma consulta em várias tabelas utilizando interior junta, junta exterior esquerda, direita exterior junta, e cruzar junta. Em seguida, é aplicada à junta vários quadros do AdventureWorks database. Você fez isso ao executar SELECT declarações utilizando-se uma não-equi join com uma comparação, com uma desigualdade comparação, com um intervalo operador, com uma expressão, e com uma conversão. Então você executa SELECT usando as declarações CROSS APPLY e OUTER APPLY. Você utilizadas várias técnicas para combinar e limitar os conjuntos de resultados. Você usou o operador UNION para combinar duas tabelas compatíveis. Então você gerados relatórios usando o INTERSECT e EXCETO cláusulas. Finalmente, você executa consultas usando o TOP e TABLESAMPLE cláusulas.

Com o qual você pode usar cláusula OUTER JOIN?

[Cláusula FROM.](#)

APLICAR operador que retorna linhas da tabela exterior para que a mesa-valorizada função retorna um conjunto vazio?

[Operador CROSS APPLY.](#)

Operador, que limita o resultado conjuntos, especificando um valor numérico?

[Operador TOP.](#)

-----*****-----

Trabalhando com Subconsultas no Microsoft SQL Server 2005

Uma subconsulta é um SELECT aninhado dentro de um SELECT, INSERT, UPDATE ou DELETE. Pode ser usada para simplificar consultas complexas. Dependendo do tipo de resultado retornado, uma subconsulta pode ser usada ao invés de uma expressão ou uma tabela derivada. Há dois tipos de subconsultas, nominalmente, consultas escalares e tabulares. Uma subconsulta é introduzida ao usar operadores de comparação que podem ser modificados. Você também precisa ter cuidado com as linhas de orientação para escrever subconsultas eficientes.

Como as subconsultas funcionam

Uma subconsulta é um SELECT aninhado dentro de um SELECT, INSERT, UPDATE ou DELETE. Você pode reescrever varias subconsultas como joins. Também pode usá-las no lugar de expressões. Por exemplo, essa consulta requer uma lista sw todos os produtos da listPrice que são maiores do que a média dos mesmos. Nesse exemplo, o SELECT calcula a media de todos os produtos da ListPrice que é chamado na consulta de dentro. Esse Select é chamado de consulta externa.

A consulta de dentro será avaliada primeiro e depois o valor da media que é calculado seria retornado na consulta externa por substituição na clausula WHERE.

Você pode ter subconsultas aninhadas em 32 níveis embora o limite se baseie na memória disponível e na complexidade de outras expressões na consulta.

Outro exemplo demonstra três subconsultas aninhadas que retornam uma lista de Produtos cuja quantidade pedida é maior que a media de todos os pedidos. O Select do nível mais interno retorna a media da quantidade de todos os pedidos.

Então o valor da media é usado no Select do meio para recuperar os produtos cuja quantidade é maior que a media.

Por final, a lista dos produtos recuperada é usada no select externo que mostra os nomes dos produtos cuja quantidade é maior do que é media da quantidade total. Esse exemplo esclarece a seqüência natural de execução das subconsultas. A consulta mais interna é executada, o seu resultado é usado na consulta de nível acima e assim por diante.

Como utilizar Subconsultas como Expressões e Tabelas Derivadas

Uma subconsulta pode ser substituída em qualquer declaração onde uma expressão é permitida. A subconsulta deve avaliar um valor escalar, ou uma única lista de colunas. Subconsultas que retornam uma lista de valores substituem uma expressão na clausula WHERE que contenha a palavra-chave IN. Os dois tipos de subconsulta são escalar e tabular.

Subconsulta Escalar

Pode substituir qualquer expressão valida usada. Retorna somente um único valor escalar. Os valores retornados pela subconsulta podem ser substituídos como uma das colunas do select , ou no WHERE, HAVING ou FROM como uma tabela 1x1. Como no exemplo seguinte em que os valores retornados são substituídos como uma das colunas.

```
USE AdventureWorks
SELECT Name, ListPrice,
(SELECT AVG(ListPrice)
FROM Production.Product)
AS Average, ListPrice –
(SELECT AVG(ListPrice)
FROM Production.Product)
AS Difference
FROM Production.Product
WHERE ProductSubcategoryID = 1
```

Subconsultas Tabulares

Retorna uma lista de valores. Portanto é introduzida na clausula WHERE usando o operador IN. Podem também ser usadas na clausula FROM. Retorna múltiplos valores para uma coluna. Uma subconsulta de tabela aninhada ou uma tabela derivada retorna múltiplas colunas com múltiplos valores. A seguir, um exemplo.

```
USE AdventureWorks
SELECT Name FROM Production.Product
```

```
WHERE ProductID IN  
(SELECT ProductID FROM Sales.SalesOrderDetail WHERE OrderQty > 30)
```

Mais exemplos:

```
USE AdventureWorks  
SELECT SalesTaxRateID, TaxType, Name  
FROM Sales.SalesTaxRate  
WHERE TaxRate=  
(SELECT MAX(TaxRate)  
FROM Sales.SalesTaxRate)
```

```
┌───┐ ┌───┐  
SalesTaxRateID TaxType Name  
30           3      France Output Tax
```

```
┌───┐  
USE AdventureWorks  
SELECT SalesOrderID, ProductID, OrderQty  
FROM Sales.SalesOrderDetail  
WHERE UnitPrice >  
(SELECT AVG(UnitPrice)  
FROM Sales.SalesOrderDetail)
```

```
┌───┐ ┌───┐  
SalesOrderID ProductID OrderQty  
43659      776      1  
43659      777      3  
43659      778      1  
43659      771      1  
43659      772      1
```

Como usar os operadores ANY, ALL e SOME

Operadores de comparação que introduzem uma subconsulta podem ser modificados pelas palavras-chaves ALL ou ANY. A palavra-chave SOME é um padrão SQL-92 equivalente ao ANY. A seguir, a sintaxe:

```
SELECT ...  
WHERE expression comparison {ALL | ANY | SOME} (Sub query)
```

As subconsultas introduzidas com esses operadores retornam uma lista de 0 ou mais valores e podem incluir as cláusulas GROUP BY ou HAVING.

As palavras-chave ALL e ANY

ALL significa maior do que todo valor. Por exemplo, ALL(1,2,3), significa maior do que 3. Se um SELECT tem uma subconsulta com ALL, então o valor na coluna que traz a subconsulta deve ser maior que cada valor da lista retornada.

De modo similar a expressão ANY significa maior do que pelo menos um valor. Então ANY(1,2,3) significa maior que 1. Se um SELECT tem uma subconsulta com ANY o valor na coluna que tem a subconsulta deve ser maior que pelo menos um dos valores da lista retornada. O ANY é equivalente ao IN.

Exemplos:

```
USE AdventureWorks
SELECT Name,ListPrice
FROM Production.Product
WHERE ListPrice >= ANY (
SELECT MAX (ListPrice)
FROM Production.Product
GROUP BY ProductSubcategoryID)
```

```
USE AdventureWorks
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice >= SOME(
SELECT MAX (ListPrice)
FROM Production.Product
GROUP BY ProductSubcategoryID)
```

Name	ListPrice
LL Mountain Seat Assembly	133.34
ML Mountain Seat Assembly	147.14
HL Mountain Seat Assembly	196.92
LL Road Seat Assembly	133.34
ML Road Seat Assembly	147.14

```
USE AdventureWorks
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice >= ALL(SELECT MAX (ListPrice)
FROM Production.Product
GROUP BY ProductSubcategoryID)
```

Name	ListPrice
Road-150 Red, 62	3578.27
Road-150 Red, 44	3578.27
Road-150 Red, 48	3578.27
Road-150 Red, 52	3578.27
Road-150 Red, 56	3578.27

Considerações para se escrever Subconsultas

Você pode usar subconsultas em certos cenários. E precisa estar avisado das restrições ao escrevê-las.

Subconsultas permitidas

Que substitui uma expressão válida em um SELECT

Ao usar UPDATE, INSERT e DELETE.

Substituir uma expressão quando um único valor ou uma lista de valores são retornados.

Substituir uma tabela ou desempenhar um função de join quando um conjunto de registros é retornado.

Não permitidas

As que são usadas na cláusula ORDER BY, contudo uma subconsulta pode incluir um ORDER BY quando o operador TOP é usado.

As que incluem as cláusulas COMPUTE e FOR BROWSE.

As que usam o DISTINCT incluindo o GROUP BY.

Orientações:

- * Devem ser escritas entre parênteses.

- * Se uma tabela aparece apenas em uma subconsulta e não na consulta exterior, a partir de então colunas de uma tabela não podem ser incluídas na saída a não ser as colunas que são passadas para a escolha da lista da consulta exterior.

- * Subconsultas não podem ser usadas para recuperar colunas que contêm os tipos de dados text, ntext e image.

- * Subconsultas podem ter subconsultas aninhadas, até 32 níveis. Este limite varia dependendo da memória disponível e à complexidade das outras expressões na consulta. Consultas individuais podem não suportar aninhamento até 32 níveis.

- * Subconsultas são executadas sequencialmente por SQL Server. Join tipicamente permitem ao otimizador de consulta recuperar dados da maneira mais eficiente. Para melhorar o desempenho, sempre que possível, deve-se reescrever a subconsulta como um join.

- * A lista de uma subconsulta introduzida com um operador de comparação pode incluir apenas uma expressão ou o nome de uma coluna.

- * Se a cláusula WHERE de uma consulta externa inclui um nome de coluna, é preciso associar com a coluna compatível na lista da subconsulta.

- * Você não pode atualizar uma visão que foi criada com uma subconsulta.

Exemplo:

Sub

```
SELECT Name, ProductNumber
FROM Production.Product
WHERE ProductID =
(SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE OrderQty = 41)
```

Join

```
SELECT p.Name, p.ProductNumber
FROM Production.Product p
JOIN Sales.SalesOrderDetail AS sod
```

```
ON p.ProductID = sod.ProductID
WHERE sod.OrderQty=41
```

Name	ProductNumber
Women's Mountain Shorts, L SH-W890-L	

Questões

Você precisa fazer uma lista de produtos cuja quantidade de pedidos de cada um é maior do que a média da quantidade de pedidos de todos os produtos. A seguir a consulta que você vai executar. Em que ordem as subconsultas serão executadas?

```
(1)SELECT Name FROM Production.Product
WHERE ProductID IN
(2)( SELECT ProductID FROM Sales.SalesOrderDetail
WHERE OrderQty >(3) ( SELECT avg(OrderQty)
as OrderQty FROM Sales.SalesOrderDetail ))
```

- 1, 2,3
- 3,1,2
- 3,2,1 X
- 1,3,2

Qual das afirmações é uma consideração básica quando se escreve uma subconsulta?

Uma subconsulta é usada para recuperar dados de todos os tipos.

Subconsultas podem ser aninhadas em 32 níveis. X

Você pode usar uma subconsulta em qualquer cláusula de uma consulta.

Uma subconsulta pode sempre retornar somente um único valor.

Escrevendo Subconsultas Correlacionadas

Você pode usar subconsultas correlacionadas a executar a subconsulta uma vez, e depois substituir o valor ou valores resultantes na cláusula WHERE da consulta externa. Você também pode escrever repetindo consultas usando subconsultas correlacionadas. Para determinar se os dados requeridos existe em uma lista de valores, você pode usar o EXISTS e NOT EXISTS operadores correlacionada com subconsultas. Quando a consulta interna e externa consulta referir-se à mesma mesa, você precisará usar tabela aliases para evitar ambigüidade sobre a tabela que é referenciada.

Como Subconsultas correlacionadas funcionam

Subconsulta na base, a consulta é aninhada dentro de um SELECT, INSERT, UPDATE, ou DELETE declaração. Ao invés, num correlacionados consulta, você pode executar a subconsulta uma vez e substituir o valor ou valores resultantes na cláusula WHERE da consulta externa.

Uma subconsulta correlacionada também é conhecida como uma repetição subconsulta porque a subconsulta exterior depende da consulta para os seus valores.

A subconsulta é executada repetidamente, para cada linha no exterior consulta com uma linha de cada vez, que, por sua vez, é avaliada como uma expressão de que a fila e passado para o exterior consulta.

Então, SQL Server compara os resultados da subconsulta aos resultados fora da subconsulta. A correlação subconsulta é efetivamente uma JOIN entre a subconsulta e executado dinamicamente a linha do exterior consulta.

Você também pode usar subconsultas correlacionadas para quebrar queries complexas em duas ou mais simples, as consultas relacionadas.

Quando você usa correlacionados subconsultas para operações como a seleção de dados de uma tabela referenciada no exterior consulta, você precisará criar uma tabela alias para especificar, inequivocamente, que tabela referência a utilizar. Você tem que usar apelidos ou nomes correlação para evitar ambigüidade, quando tanto o interior e o exterior da consulta referir-se à mesma mesa.

Na mesma tabela

O exemplo a seguir mostra uma correlação subconsulta na qual tanto o interior e no exterior consultas referem-se à mesma mesa. A consulta interna referências a ProductID coluna da consulta externa.

```
SELECT DISTINCT ProductID, OrderQty
FROM Sales.SalesOrderDetail AS sod outer
WHERE OrderQty = (SELECT MAX(OrderQty)
FROM Sales.SalesOrderDetail AS sod_inner
WHERE sod_outer.ProductID= sod_inner.ProductID)
```

Em tabelas diferentes

O exemplo a seguir mostra uma correlação subconsulta onde o interior e o exterior consultas usar duas tabelas diferentes. Neste exemplo, a consulta junta a Sales.SalesPerson e os Sales.SalesOrderHeader tabelas. O SalesPersonID é transmitida a partir do exterior consulta para o interior consulta.

```
SELECT SalesPersonID, CommissionPct FROM Sales.SalesPerson p
WHERE 2000 < (SELECT Max(totaldue) * CommissionPct AS Commission
FROM Sales.SalesOrderHeader soh
WHERE p.SalesPersonID = soh.SalesPersonID)
```

Como usar a cláusula EXISTS com Subconsultas Correlacionadas

Quando uma subconsulta é introduzida com o operador EXISTS, ela funciona como um teste existência. A cláusula WHERE da consulta externa testa a existência de linhas retornadas pela subconsulta. A subconsulta retorna o valor TRUE ou FALSE. E na verdade não produz quaisquer dados. Portanto, subconsultas que usam os operadores EXISTS ou NOT EXISTS são muito eficientes porque a subconsulta sai logo que a primeira linha é retornada.

Como o Operador EXISTS é Utilizado com Subconsultas Correlacionadas

Para determinar se os dados requeridos existem em uma lista de valores, você pode usar os operadores EXISTS e NOT EXISTS correlacionados com subconsultas. Você também pode usar esses operadores para restringir o resultado de uma consulta externa para linhas que satisfaçam a subconsulta. Baseada em saber se algumas linhas são devolvidas para subconsultas, os operadores EXISTS e NOT EXISTS retornam TRUE ou FALSE. Eles não produzir quaisquer dados.

Sintaxe para os Operadores EXISTS e NOT EXISTS

```
WHERE [NOT] EXISTS (subquery)
```

Nesta sintaxe, a subconsulta é o SELECT. É preciso lembrar que a cláusula COMPUTE, e os operadores INTO não são permitidos na subconsulta.

Diferenças entre subconsultas utilizando exists e outras subconsultas

Subconsultas introduzidas com o operador EXISTS são diferentes de outras subconsultas da seguinte forma:

O operador EXISTS não é precedido por uma coluna nome, constante, ou outra expressão.

A escolha de uma lista de subconsultas introduzida pelo operador EXISTS quase sempre constituída por um asterisco (*). Você não precisa da lista dos nomes de colunas porque você é apenas testar a existência de linhas que satisfaçam as condições especificadas na subquery.

Embora algumas consultas formuladas com o operador EXISTS não podem ser expressas em qualquer outra forma, todas as consultas que utilizam o operador AND ou um operador de comparação modificado pelos operadores ANY ou ALL podem ser expressos com o operador EXISTS.

O seguinte código ilustra o uso do operador EXISTS e NOT EXISTS.

```
SELECT EmployeeID, FirstName, LastName
FROM HumanResources.Employee AS e
INNER JOIN Person.Contact c ON e.ContactID = c.ContactID
WHERE EXISTS (SELECT * FROM Sales.SalesOrderHeader AS soh
WHERE e.EmployeeID = soh.SalesPersonID
AND soh.OrderDate BETWEEN '20030101' AND '20030131')
```

EmployeeID	FirstName	LastName
275	Michael	Blythe
276	Linda	Mitchell
277	Jillian	Carson
278	Garrett	Vargas
279	Tsvi	Reiter
280	Pamela	Ansman-Wolfe

Construindo Subconsultas Correlacionadas

Questões

Você foi solicitado a obter alguns detalhes dos empregados que trabalham em Adventure Works. Então você faz uso tanto do interior consulta, bem como a consulta externa para obter as informações pedidas. Ambas as consultas são referencias as mesmas tabelas e isto leva a uma série de ambigüidade. Que uma das seguintes técnicas que você irá usar para evitar ambigüidade na tabela de dados referências?

[Join](#)

[Alias X](#)

[Subconsultas correlacionadas](#)

[Subconsultas](#)

Você foi solicitado um obter alguns detalhes dos empregados que trabalham em Adventure Works. Então você faz uso tanto do interior consultas, bem como uma consulta externa como para obter informações pedidas. Ambas como consultas são referencias como mesmas tabelas e isto leva um uma série de ambigüidade. Que uma das seguintes técnicas que você irá usar para evitar ambigüidade no quadro de dados referências?

[COMPUTE](#)

[INTO](#)

[EXISTS X](#)

[FROM](#)

Em uma correlação subquery, que uma das seguintes consultas inicia o processo de passagem da coluna valores para o interior query?

[Consulta externa X](#)

[Consulta interna](#)

Subconsulta com alias

Subconsulta

Comparando Subconsultas com Joins e Tabelas Temporárias

Você pode escrever consultas eficientes no SQL Server, utilizando diversas técnicas, tais como subconsultas, joins e tabelas temporárias. Dependendo dos dados em um determinado cenário, um método específico pode ser mais rápido. Para determinar o método mais eficiente para o dado de dados, você precisa estar ciente de como subconsultas são processadas pelo SQL Server em comparação a join e tabelas temporárias.

Subconsulta X Join

Você pode executar consultas em SQL Server utilizando junta e subconsultas. Uma consulta que utiliza uma subquery muitas vezes podem ser escritos como uma join. A consulta otimizador normalmente otimiza subconsultas, para que ele possa usar a amostra execução de um plano semanticamente equivalente aderir iria usar, mas isso nem sempre é possível. Dependendo do dado e os dados necessários resultado conjunto, você pode usar subconsultas ou join. Ao testar ambos os tipos de consultas, você pode determinar qual opção é mais rápida para um determinado cenário.

Subconsultas

O código seguinte exemplo mostra como uma consulta utiliza subconsultas para obter uma lista de todos os produtos que tenham inventário níveis superiores a 500 em um determinado local.

```
SELECT ProductID, Name FROM Production.Product prod
WHERE EXISTS(SELECT Quantity FROM Production.ProductInventory inv
WHERE prod.ProductID = inv.ProductID AND Quantity > 500)
```

SQL Server executa subconsultas seqüencialmente. Uma subquery pode exigir a consulta otimizador para realizar etapas adicionais, tais como a ordenação. Como resultado, subqueries podem levar mais tempo para ser executada do que junta. No entanto, se o tamanho de um resultado conjunto é pequena ou se não houver índices sobre as colunas, uma subquery pode ser mais rápido do que um join. A maioria das subqueries podem ser reescritas como junta para produzir idêntico resultado conjuntos.

Join

O código seguinte exemplo mostra como uma consulta utiliza junta para obter uma lista de todos os produtos que tenham inventário níveis superiores a 500 em um determinado local.

```
SELECT DISTINCT Prod.ProductID, Prod.Name
FROM Production.Product prod
INNER JOIN Production.ProductInventory inv
ON prod.ProductID = inv.ProductID AND Quantity > 500
```

SQL Server executa junta concomitantemente. Em alguns casos, quando existência precisa de ser verificado, uma join executa melhor. No entanto, junta são mais eficientes em termos de dados onde os aderiram colunas são indexadas. Ao usar junta, você pode obter o mesmo resultado retornado por uma subquery.

Subconsulta X Tabela Temporária

Quando você executa uma subquery diversas vezes entre várias declarações, que tem que ser avaliada de cada vez. Em vez disso você pode usar tabelas temporárias. A tabela temporária pode ser tanto na sua base de dados ou tempdb e, em seguida, você pode executar SELECT posteriores declarações contra esta tabela temporária. Tabelas temporárias podem ser usadas para o tratamento de dados intermédios ou para partilhar alguns dados processados com outras conexões.

Existem dois tipos de tabelas temporárias: locais e globais. Local tabelas temporárias são visíveis apenas para a atual ligação para o usuário, e elas são excluídas quando o usuário desconecta da

instância do SQL Server 2005. Global tabelas temporárias são visíveis a qualquer usuário depois que eles são criados, e elas são excluídas quando todos os usuários referenciando a tabela desconectar. É recomendado que você explicitamente a queda tabelas temporárias quando eles não estão mais em uso.

Subconsulta

Cumprimento de uma subquery várias vezes ou em múltiplas declarações resultados em esgotar recursos do sistema devido SQL Server tem de reavaliar a subquery cada vez que for executado.

Tabela Temporária

Quando uma subquery deve ser reutilizados, recomenda-se que uma tabela temporária será criada e os resultados da subquery ser inserido na tabela temporária. Então, qualquer declaração SELECT que tem a referir-se aos dados retornados pelo original subquery tem-se apenas a referir-se a tabela temporária. Os cálculos são feitos apenas uma vez e os resultados são armazenados na tabela temporária. Você também pode executar o INSERT, UPDATE, ou DELETE declarações sobre a tabelas temporárias. Se existem muitas linhas de dados, você pode aplicar em determinadas colunas índices para melhorar o desempenho das consultas.

Questões

Como administrador de banco de dados Adventure Works, o que você precisa para preparar um relatório sobre o volume de negócios de todos os produtos produzidos durante o ano. Você também precisará dar a relatórios individuais do volume de vendas de cada cor do produto. Para preparar este relatório, você optar por usar junta ao invés de subqueries na sua declaração SELECT. Qual seria o motivo para escolher junta durante subqueries?

- * As participações são sempre executadas mais rápido do que subqueries em SQL Server.
- * As participações estão muito melhor maneira de executar uma consulta, quando comparados com subqueries porque SQL Server realiza a execução de junta em ordem seqüencial.
- * Junta são mais eficientes do que as subqueries quando ingressou colunas são indexadas. X
- * Junta são mais rápidos do que subqueries quando o tamanho do conjunto de resultados é pequena.

O gerente de finanças em Adventure Works, solicitou-lhe que elabore um relatório sobre dívidas pendentes, a partir do último dia do ano anterior. Com base neste relatório, ele também quer que você para descobrir 5% desse montante, que pode ser definida como provisão para o próximo ano e 2% desse montante, que deve ser amortizado como devedores duvidosos. Esta consulta é executado somente para sua referência e não é obrigada por ninguém. Que um dos seguintes métodos são recomendados para melhor descobrir rapidamente essas informações?

Usando subconsultas

[Usando tabelas temporárias locais X](#)

[Usando tabelas temporárias globais](#)

[Utilizando tabelas permanentes](#)

Usando expressões Comuns de Tabela

Expressões comuns tabela (CTEs) são ANSI SQL-99 construções que foram adicionadas ao SQL Server 2005. Eles fornecem um mecanismo alternativo para escrever consultas complexas. Consultas usando CTEs são mais simples e mais compreensível do que subqueries. Ao usar CTEs, você pode criar e construir queries recursivas facilmente no SQL Server 2005.

Como expressões Comuns de Tabela funcionam

A CTE é como um resultado temporário conjunto que é definido no âmbito da execução âmbito de um SELECT, INSERT, UPDATE, DELETE, CREATE VIEW ou declaração.

A CTE é semelhante a uma mesa ou um derivado subquery. Não é armazenada como um objeto no banco de dados, e dura apenas durante a vigência da consulta. O código exemplo mostra como escrever uma consulta usando um simples CTE.

O resultado mostra o primeiro conjunto consulta utilizando-se uma CTE. Ao contrário de uma subquery, uma CTE pode ser auto-referencial e você poderá consultá-la várias vezes na mesma consulta. Ao usar CTEs, a consulta pode ser dividido em separado, simples e blocos lógicos. Estes simples blocos também pode ser usado para construir mais complexo e CTEs interino até que o resultado final previsto é gerado.

Como escrever expressões Comuns de Tabela

CTEs são ANSI SQL-99 construções que foram adicionadas ao SQL Server 2005. Ao usar CTEs, você pode criar tabelas derivadas recursivas e consultas que são mais legível do que aqueles criados utilizando padrão T-SQL sintaxe.

Sintaxe

**COM expression_name [(column_list,...)] AS
(CTE_query_definition)**

A sintaxe básica de uma CTE é constituído pelos seguintes elementos:

Expression_name. Este é o nome que é usado em referência a uma consulta aos CTE.

Expression_name pode ser qualquer identificador válido, mesmo nome, que existe uma tabela no banco de dados. O CTE nome indicado por **expression_name** terão prioridade no âmbito local.

Column_list. Esta especifica os nomes das colunas para o CTE. O número de colunas em **column_list** deve ser igual ao número de colunas no resultado. Se o **column_list** não é especificado, os nomes das colunas são tomadas a partir do conjunto de resultados.

CTE_query_definition. Isto especifica o SELECT declaração que faz do conjunto de resultados e, portanto, define o CTE.

Uso

CTEs pode ser usada em uma instrução SELECT, INSERT, UPDATE, ou DELETE declaração semelhante a uma tabela ou exibição. CTEs também pode ser utilizado na definição de uma declaração SELECT CREATE VIEW declaração.

Vantagens

Seguem-se as vantagens de utilizar CTEs:

Consultas com derivados tabelas são mais simples e legível, em comparação ao padrão T-SQL construções que trabalhar com tabelas derivadas. T-SQL constrói exigem uma definição separada para os derivados de dados, como uma tabela derivada, tabela temporária ou uma mesa-valorizada função. Ao usar CTEs, é mais fácil ver a definição da tabela derivada com o código que ele usa.

CTEs reduzir a quantidade de código necessário para uma consulta que movimenta recursiva hierarquias. Por exemplo, quando as linhas da mesma tabela estão relacionados com um auto-join, a consulta movimenta recursiva hierarquias.

Exemplo

**USE AdventureWorks
GO WITH TopSales (SalesPersonID, NumSales) AS (SELECT SalesPersonID, Count(*)**

```
FROM Sales.SalesOrderHeader GROUP BY SalesPersonID
) SELECT TOP(5) * FROM TopSales
WHERE SalesPersonID IS NOT NULL
ORDER BY NumSales DESC
```

SalesPersonID	NumSales
277	473
275	450
279	429
276	418
285	348

Como escrever consultas recursivas usando expressões comuns de tabela

Pode escrever queries recursivas muito facilmente, criando CTEs. Além disso, uma CTE-baseado recursiva consulta é mais eficiente do que uma consulta usando padrão T-SQL declarações, porque uma CTE-baseado recursiva query usa menos linhas de código, quando comparado ao padrão T-SQL declarações. Criando um recursiva CTE consulta é similar à criação de uma CTE normal, com exceção de algumas considerações adicionais no SELECT declaração do CTE definição. SELECT A declaração deve consistir em duas consultas: uma não recursiva ponto de partida, conhecido como o âncora membro consulta, bem como uma consulta que se refere à CTE, conhecido como o membro recursiva query.

Para criar uma recursiva CTE, você deve executar as seguintes três etapas:

1. Crie o âncora membro consulta

O primeiro passo na criação de uma consulta recursiva é a criação da âncora membro consulta. O âncora membro consulta formas o topo da árvore recursão. Ele determina os nomes das colunas, a menos que você precisar deles no parâmetro opcional `column_list` do `COM` declaração.

2. Acrescentar o operador UNION ALL

O segundo passo na criação de uma consulta recursiva é a inclusão do operador `UNION ALL` na consulta. O operador `UNION ALL` é essencial para ligar os dois conjuntos resultado da consulta. O operador `UNION ALL` é o único operador que é permitido fixar em uma consulta recursiva.

3. Crie o recursiva membro consulta

O passo final na criação de uma consulta recursiva é a criação do recursiva membro consulta. A consulta deve referência recursiva membro do CTE como uma das tabelas na consulta `SELECT`. Além disso, o número de colunas no recursiva membro consulta deve ser igual ao número de colunas da âncora membro consulta. A recursão continua até o recursiva membro consulta deixa de produzir os conjuntos de resultados.

Ao escrever queries recursivas, um erro na programação recursiva membro poderá resultar na consulta inscrição de um ciclo infinito. O número infinito de iterações, então monopolizar os recursos do sistema e impacto desempenho. Para limitar o nível de recursão, você pode especificar a opção `MAXRECURSION` no CTE consulta. Se o recursiva membro exceder esse limite, ocorre uma exceção e não a consulta. O valor padrão para `MAXRECURSION` é 100.

Exemplo


```

WITH Managers AS
(
SELECT EmployeeID, LoginID, Title, ManagerID
FROM HumanResources.Employee
WHERE EmployeeID = 109
UNION ALL
SELECT e.EmployeeID , e.LoginID, e.Title, e.ManagerID
FROM HumanResources.Employee e
INNER JOIN Managers mgr
ON e.ManagerID = mgr.EmployeeID
)
SELECT * FROM Managers

```

EmployeeID	LoginID	Title	ManagerID
109	adventure-works\ken0	Chief Executive Officer	NULL
6	adventure-works\david0	Marketing Manager	109
12	adventure-works\terri0	Vice President of Engineering	109
42	adventure-works\jean0	Information Services Manager	109
140	adventure-works\laura1	Chief Financial Officer	109

Construindo consultas recursivas usando expressões comuns de tabela

Questões

Você foi convidado pelo FC departamento para criar uma lista de funcionários com os seus números de vendas. Você especifica o seguinte consulta com um CTE. Que uma das seguintes afirmações é TRUE sobre o resultado desta consulta?

```

With TopSales (SalesPersonID, NumSales)
AS (SELECT SalesPersonID, Count (*) FROM Sales.SalesOrderHeader GROUP BY SalesPersonID)
SELECT LoginId, NumSales FROM HumanResources.Employee e interior JOIN TopSales sobre
TopSales.SalesPersonID = e.EmployeeID
Por fim NumSales desc

```

As colunas da saída será o LoginID, NumSales colunas da tabela de empregados e as SalesPersonID e Count (*), a partir da SalesOrderHeader tabela.

As colunas da saída será o LoginID e os NumSales do Trabalhador tabela. X

As colunas na saída serão ordenados em ordem decrescente da NumSales coluna no SalesOrderHeader tabela.

As colunas da saída será agrupada por EmployeeID

Qual das seguintes declarações são TRUE sobre os argumentos que são utilizados por escrito CTEs?

O column_list especifica o número de colunas para o CTE.

O expression_name CTE é o nome que irá referência na consulta que usa o CTE. X

O CTE_query_definition SELECT especifica a declaração de que forma o resultado conjunto e, portanto, as formas CTE. X

O expression_name deve ser uma tabela nome, que já existe no banco de dados.

O column_list deve especificar colunas mais do que o número de colunas no resultado.

A declaração SELECT requisitos para a criação de uma CTE são as mesmas para a criação de um ponto de vista.X

Qual dos seguintes componentes é uma parte válida de um recursiva query?

A recursiva ponto de partida, o que é conhecido como o membro recursiva query.

O MAXRECURSION opção, o que limita o nível de recursão. X

Uma consulta que se refere à CTE, que é conhecido como o âncora membro consulta.

O operador JOIN que liga os dois conjuntos resultado da consulta.

Resumo

Neste laboratório, você criou relatórios sobre vários pormenores relacionados com as vendas dos empregados por meio da execução de Aventura Obras subqueries, subqueries correlacionadas, e CTEs. Você também comparado se você precisa usar subqueries, junta, ou tabelas temporárias. Em seguida, você executa SELECT declarações que tinham escalar subqueries na declaração e as SELECT WHERE. Você executa tabular subqueries. Você também executa uma tabela derivada ingressou com outra tabela. Após isto, você executa SELECT declarações que tinham subqueries correlacionadas com dois quadros separados e com a mesma tabela. Você executa uma simples declaração e um EXISTS subquery correlacionado com a cláusula EXISTS. Você executa SELECT declarações com subqueries. Você criou uma tabela temporária. Em seguida, você executa o SELECT declarações com tabelas temporárias em vez de subqueries. Finalmente, você executa consultas com uma simples tabela expressão comum, um quadro comum recursiva expressão e do MAXRECURSION opção.

Você precisa escrever uma subquery interna que retorna múltiplos valores para o exterior subquery. Subquery qual seria a sua utilização?

[subconsulta tabular](#).

Você precisa limitar o número de recursões em uma recursiva CTE. Que opção ou dica que você usa? [opção MAXRECURSION](#).

Operador que restringe o resultado de um conjunto exterior consulta para linhas que satisfaçam a subquery?

[operador EXISTS](#).

-----*****-----
-

Modificando Dados com o Microsoft SQL Server 2005

Modificando dados em tabelas

Para gerar relatórios em função de uma exigência, que você precisa para manipular os dados em tabelas. Transações são uma lógica sequencial da unidade de trabalho. Por conseguinte, as operações levar à alteração de dados. Transações pode ser explícita ou implícita. Você pode inserir, excluir ou atualizar um ou vários registros de dados nas tabelas. Para fazer isso, você pode utilizar as diversas declarações, tais como INSERT, DELETE, UPDATE, SELECT... INSERT, SELECT INTO, e TRUNCATE TABLE.

Propriedades das Transações ACID

A lógica da unidade de trabalho deve expor quatro propriedades para se qualificar como uma transação, a saber, Atomicidade, consistência, isolamento e durabilidade. Estas propriedades são também comumente referidas como ACID.

Atomicidade exige que cada operação deva ser uma unidade atômica de trabalho, o que significa que todos os dados modificações em uma transação são executadas ou nenhum dos dados modificações em uma transação são realizados.

A coerência exige que, após a operação for concluída, deve deixar todos os dados no banco de dados em um estado consistente para manter a integridade dos dados. Por exemplo, todas as estruturas de dados internas devem ser consistentes no final da operação.

Isolamento exige que as modificações feitas por uma transação concorrente deve ser isolado a partir de modificações feitas por qualquer outra transação concorrente. Por conseguinte, uma operação deve utilizar os dados nem antes, nem depois de outra transação concorrente modificá-lo, e nunca deve utilizar os dados em um estado intermediário.

Durabilidade exige que, após a operação for concluída, as modificações devem persistir permanentemente no sistema, mesmo que exista um sistema de falha.

Transações de Log

SQL Server registros todos os dados modificações feitas na base de dados em uma operação rápida. O registro dos registros de início e fim de cada transação.

Uma instância do SQL Server armazena informações suficientes no diário para tanto rolo para frente ou para trás os dados roll modificações que compõem uma transação.

Cada registro da operação no log é identificado por um único registro número seqüencial (LSN).

Modos de Transação

Transações em SQL Server pode ser explícitas, ou AUTOCOMMIT transações. Em transações explícito, você precisa definir explicitamente tanto no início e no fim da transação.

Cada operação é explicitamente começou iniciar a transação com a declaração explícita e terminou com a transação ou de uma declaração repetir a transação.

Múltiplas declarações para uma transação podem ser agrupadas em transações explícitas. O modo padrão no SQL Server é AUTOCOMMIT transações, onde cada indivíduo T-SQL é cometida quando ele estiver concluído.

No entanto, se a afirmação de encontros qualquer erro, então é retirada.

O SQL Server também suporta transações implícitas, onde se inicia automaticamente a uma nova operação após a transação atual é cometido ou laminado de volta.

Uma nova operação é implicitamente quando começou a operação prévia completa, mas cada operação é explicitamente a transação concluída com uma declaração ou repetir a transação. Cada comando SQL como um INSERT, UPDATE, ou DELETE é implicitamente uma transação ou automático.

Como o SQL Server modifica os dados na tabela

O SQL Server utiliza as tabelas suprimidos e aditados ao executar Dados Manipulação Language (DML) declarações. LMG acionar declarações suprimidos e aditados a usar tabelas para testar os efeitos de certas modificações ou dados para definir as condições para desencadear ações LMG.

SQL Server cria e gere esses temporário, residente em memória, tabelas, automaticamente. Durante um INSERT ou UPDATE operação, as novas linhas estão a aditado simultaneamente tanto a tabela e inserido o quadro temporário. As linhas inseridas na tabela são cópias das novas linhas na tabela afetada pela INSERT e UPDATE declarações.

Durante um UPDATE ou DELETE operação, as linhas afetadas pela declaração UPDATE ou DELETE são retirados e transferidos para a tabela temporária suprimido.

A tabela temporária e suprimida a mesa afetado pelo INSERT ou UPDATE declarações, geralmente não têm linhas em comum. Uma atualização operação é semelhante a uma operação DELETE seguido por uma operação INSERT.

Durante uma operação UPDATE, as antigas linhas são copiados para a tabela excluído primeiro, e depois as novas linhas são copiados para os quadros e para o quadro inserido.

Identificando conceitos relacionados a Transações

Cada azulejo contém uma declaração verdadeira e uma declaração falsa. Clique nos azulejos em uma linha ou coluna até que cada um mostre a verdadeira declaração e, em seguida, clique na seta ao

lado da linha ou em cima da coluna para verificar a sua resposta. Você usa voltar quando você clicar Enviar em um azulejo incorreto. Remover todas as peças antes de utilizar todas as suas voltas.

Como gerenciar Transações

A operação pode ser iniciado quer como uma transação explícita ou implicitamente uma transação. Com implícita transações, uma nova operação é implicitamente começou quando a transação anterior está concluído, mas cada operação é explicitamente concluída com um COMMIT ou rollback declaração. Com explícito transações, você explicitamente definir o início e o fim de uma transação. Explícita operações proporcionam um maior controle sobre transações. Você pode evitar problemas e de recursos concurrency e, por conseguinte, melhorar o desempenho das operações. Com explícito transações, você pode também realizar-se que uma análise de modificações. Para preservar a integridade dos dados, é recomendado que você inicie uma transação antes de modificar os dados e verificar se os dados tenham sido modificados e, então, empenhar-lo explicitamente. Uma transação explícita tem vários componentes. Eles são os BEGIN TRAN, SAVE TRAN, COMMIT TRAN, e Rollback TRAN declarações.

BEGIN TRANSACTION

Esta afirmação é o ponto de partida de uma transação explícita. O seguinte é a sintaxe parcial de a declaração iniciar a transação.

```
BEGIN (TRAN | OPERAÇÃO)
[(Transaction_name | @ tran_name_variable)
[Com Mark [ 'descrição']]
:]
```

Na sintaxe, o transaction_name é o nome atribuído à transação, e @ tran_name_variable é o nome de uma variável definido pelo usuário, com um nome válido transação. As variáveis devem ser declaradas com um char, VARCHAR, nchar, ou nvarchar tipo de dados. Os com Mark ['descrição'] especifica que a operação está marcada, em nome do log. Quando restabelecimento de um banco de dados para um estado anterior, marcou a operação pode ser usada em lugar de uma data e hora.

SAVE TRAN

Esta declaração estabelece dentro de um savepoint de uma transação. O savepoint define um local para o qual uma operação pode ser retirada se parte da operação é condicionalmente cancelada.

```
Save TRAN (|) OPERAÇÃO
(Savepoint_name | @ savepoint_variable)
```

No savepoint_name a sintaxe é atribuído ao savepoint e as @ savepoint_variable é o nome de uma variável definido pelo usuário, com um nome válido savepoint. As variáveis devem ser declaradas com um char, VARCHAR, nchar, ou nvarchar tipo de dados.

COMMIT TRANSACTION

```
COMMIT TRAN (|) OPERAÇÃO
[(Transaction_name | @ tran_name_variable)]
```

Neste sintaxe, transação transaction_name especifica um nome atribuído por um anterior e iniciar a transação @ tran_name_variable é o nome de uma variável definido pelo usuário, com um nome válido transação. As variáveis devem ser declaradas com um char, VARCHAR, nchar, ou nvarchar tipo de dados. transaction_name é ignorado pelo Banco de dados SQL Server Engine.

ROLLBACK TRANSACTION

Isso retorna uma declaração explícita ou implícita transação para o início da operação, ou para um savepoint dentro da transação. O seguinte é a sintaxe parcial de a declaração repetir a transação.

Rollback TRAN (|) OPERAÇÃO

[Transaction_name | @ tran_name_variable | savepoint_name | @ savepoint_variable]

Neste sintaxe, transaction_name é o nome atribuído à operação em iniciar a transação e devem obedecer às regras para identificadores. Durante operações, a transaction_name deve ser o nome a partir de a declaração ultra periféricas iniciar a transação. A @ tran_name_variable é o nome de uma variável definido pelo usuário, com um nome válido transação. O savepoint_name é o nome de uma declaração SAVE OPERAÇÃO e devem obedecer às regras para identificadores. A @ savepoint_variable é o nome de uma variável definido pelo usuário, com um nome válido savepoint.

Transações Implícitas

Implicitamente uma transação pode ser iniciada com o SET IMPLICIT_TRANSACTIONS ON declaração. Quando implícita transações são ativados, todas as declarações são considerados como uma parte de uma transação e não as operações são autorizados até uma declaração a transação é executado. SQL Server inicia uma nova operação após a transação atual é cometido ou laminado de volta. O modo implícito transação gera uma cadeia de operações contínuas. No entanto, porque você não tem controle sobre quando começa uma transação implícita, que pode potencialmente provocar problemas de bloqueio de um ambiente que tenham um grande número de usuários.

Se houver uma transação modo implícito para uma conexão, o SQL Server inicia uma transação, sempre que a base de dados estadual ou mudanças no momento em que primeiro executa qualquer das seguintes declarações, ou seja, ALTER TABLE, SELECT, UPDATE, DELETE, INSERT, truncar TABLE, CREATE , Aberta, revogar, DROP, buscar ou concessão. Após a primeira transação seja retirada, a base de dados motor começa uma nova operação quando qualquer uma dessas declarações é executado. Você deve voltar expressamente todas as transações implícita. Se você não explicitamente cometer uma transação implícita, então a operação e todos os dados mudanças são rolados para trás quando a conexão é fechada.

A transação modo implícito permanece em vigor até ao ligação IMPLICIT_TRANSACTIONS OFF SET executa uma declaração, que retorna a ligação à auto-commit modo.

O que são Transações Aninhadas?

Explícita transações podem ser aninhadas, isto é, uma nova operação pode começar ainda que a anterior operação não esteja completo. Transações aninhadas podem ser usadas em Stored Procedures. Uma Stored procedure é uma coleção de pré-comandos T-SQL que está armazenada no servidor.

Ao utilizar Stored Procedures em transações aninhadas, um processo pode ser chamado a partir de um processo já em uma transação ou de processos que não têm qualquer transação ativa. A função de sistema @@ TRANCOUNT registra a transação de nível atual.

Cada declaração BEGIN TRANSACTION irá aumentar @@ TRANCOUNT por um, enquanto que cada COMMIT TRANSACTION diminuir-se a um. Uma declaração REPEAT TRANSACTION não tem um nome e a operação retorna todas as transações aninhadas e repõe @@ TRANCOUNT a 0.

Um REPEAT TRANSACTION que usa o nome da operação externa em um conjunto de transações aninhadas retorna todas as transações aninhadas e repõe @@ TRANCOUNT a 0.

Para verificar se o processo já iniciou uma operação, utilizar a declaração SELECT @@ TRANCOUNT. Se a @@ TRANCOUNT é de 1 ou superior a 1 e, em seguida, a operação está sendo processado. Se a @@ TRANCOUNT é 0, então, a operação não está sendo processado. Esse código exemplo ilustra os

processos em uma transação aninhada.

Em primeiro lugar, uma tabela, TestTrans, é criada.

O TransProc processo impõe a sua operação, independentemente do modo de operação a qualquer processo que executá-lo. Em seguida, a iniciar a transação declaração assinala o ponto de partida de uma transação explícita.

Se o TransProc procedimento é chamado quando uma transação está ativa, então a transação aninhada no TransProc procedimento não é considerada, e as suas declarações INSERT são cometidos ou retirada com base em medidas tomadas para a final da operação ultra periféricas. Então, você pode usar a declaração para a transação marca o fim da transação. Você também pode usar a repetir a transação declaração para voltar para o início da operação, ou para um savepoint dentro da transação.

Uma transação é ou cometeu ou retirada com base sobre as medidas adotadas no final da operação ultra periféricas. Se a iniciar a transação declarações estão encaixados e, em seguida, a transação só serão autorizados se houver uma declaração a transação no final da operação ultra periféricas.

No entanto, uma declaração rollback irá rolar para trás toda a operação em qualquer nível. Além disso, se você ninho transações, você precisa ter cuidado para não incluir vários Rollback afirmações de uma forma que possa permitir que mais de um rollback declaração de executar.

Como o nível de isolamento de transação funciona

Quando vários usuários acessar os dados ao mesmo tempo e não existe qualquer controle concurrency, você precisa lidar com várias questões concurrency. Há vários tipos de concurrency questões que podem ocorrer durante operações, tais como "sujo lê, lê fantasma, e não repetível lê.

Você pode especificar um nível isolamento em transações e definir o grau em que uma transação deve ser isolada da recursão dos dados ou das alterações feitas por outras transações. Níveis de isolamento são descritos em termos de efeitos associados à concorrência como leitura suja e leitura fantasma. ANSI SQL define quatro níveis de isolamento e as transações associadas à questões de concorrência que têm de ser resolvidas.

Leitura Uncommitted (ANSI TIL 0)

Ao nível READ UNCOMMITTED, SQL Server não bloqueia qualquer pedido ao ler uma linha, e não honrar qualquer trava existente. Você pode ler as linhas com valores que têm sido atualizado anteriormente, mas ainda não cometeu.

Nível inferior, como o isolamento READ UNCOMMITTED nível aumentar a capacidade de muitos usuários para acessar os dados ao mesmo tempo. Mas também aumenta o número de concorrentes efeitos, tais como "sujo lê ou perdido atualizações que os usuários possam encontrar.

Você precisa ter cuidado ao usar o READ UNCOMMITTED nível que ela permite que lê e sujo, por isso, não é recomendado.

Leitura Comited (ANSI TIL1)

O nível DEDICAÇÃO a LER, SQL Server adquire um lock compartilhado ao ler dados, mas libera o lock imediatamente após a leitura dos dados. Uma vez que os pedidos sejam partilhados bloqueio bloqueado por um bloqueio exclusivo, não é possível ler as linhas que foram atualizadas por uma outra tarefa, mas ainda não tenham sido cometidos.

LEIA AUTORIZADOS é o padrão para a fixação de isolamento nível tanto SQL Server e ODBC. Este nível concurrency resolve questões como lê sujo.

Leitura Repetitiva (ANSI TIL 2)

nível REPEATABLE READ, SQL Server 2005 solicita um bloqueio compartilhado em cada linha, uma vez que é lido no LEIA DEDICAÇÃO nível. No entanto, se a linha está acessível no âmbito de uma operação, a partilha de eclusas são detidos até ao final da operação, em vez de ser libertado depois

da linha é lido.

Este tem o mesmo efeito que HOLDLOCK especificando o operador em um SELECT. O REPEATABLE READ nível concurrency resolve o problema da sujeira e lê lê fantasma.

Serializavel (ANSI TIL3)

Ao nível SERIALIZABLE, SQL Server desempenha as mesmas funções como na REPEATABLE READ nível.

Nível mais elevado isolamento, tais como SERIALIZABLE, reduz os tipos de efeitos concurrency que os usuários podem encontrar, mas exige mais recursos do sistema. Ele também aumenta as chances de uma operação bloqueio outro. O SERIALIZABLE ler concurrency resolve questões de sujo lê, não repetitivo lê, lê e fantasma.

Snapshot

SQL Server 2005, introduziu um novo instantâneo isolamento nível que se destina a aumentar a concorrência para a operação de transformação on-line (OLTP) aplicações. SNAPSHOT isolamento depende de melhorias de versões fila. Não se trata de uma ANSI TIL. Além disso, melhora o desempenho, evitando leitor-escritor bloqueio cenários.

As operações que começam sob instantâneo isolamento ler um banco de dados instantâneo no momento em que começa a operação. Este instantâneo não inclui fechaduras e isso pode reduzir o bloqueio no servidor.

Para fazer uso do instantâneo isolamento nível, o que você precisa para permitir que o banco de dados sobre a execução da seguinte declaração: ALTER DATABASE AdventureWorks SET ALLOW_SNAPSHOT_ISOLATION ON;

A tabela a seguir mostra problemas de concorrência resolvidos pelo isolamento vários níveis.

Problemas de Concorrencia	Descrição
Leitura Suja	Leitura suja ou dependência não autorizada ocorre quando uma segunda operação seleciona uma linha que está a ser atualizada por uma outra operação. A segunda transação lê os dados que não foram cometidos e ainda podem ser alterados pela operação atualizando a linha.
Leitura Fantasma	Leitura fantasma ocorre quando inserir ou eliminar uma ação é realizada contra uma fila que pertence a uma série de linhas a serem lidas por uma transação. A primeira leitura do intervalo de linhas com a transação mostra uma linha que não existe mais no segundo ou suceda a leitura. Isso ocorre devido a uma supressão feita por uma operação anterior.
Leitura não repetível ou incoerente	Leitura não repetível ou análise incoerente ocorre quando uma segunda operação acessa a mesma linha várias vezes e lê dados

	diferentes de cada vez. Uma transação altera os dados de que uma segunda operação faz a leitura.
--	--

Nível de Isolamento	Leitura Suja	Leitura não repetível	Leitura fantasma
Leitura Não Comitada	Sim	Sim	Sim
Leitura Comitada	Não	Sim	Sim
Leitura Repetitiva	Não	Não	Sim
Serialized	Não	Não	Não
Visão geral	Não	Não	Não

Questões

O gerente de financiar a Adventure Works tem pediu-lhe para atualizar a tabela Investments.SharesAndStocks no AdventureWorks banco de dados, baseado no ações adquiridas e vendidas pela empresa no ano anterior. Para fazer isso, você precisa inserir, atualizar e apagar informações de que a Investments.SharesAndStocks tabela. As alterações feitas a este quadro reflecte-se igualmente a inserido e excluído tabelas. Qual dos seguintes é a principal função desses quadros?

[Para estender a integridade referencial entre tabelas.](#)

[Para assinalar erros em tabelas temporárias.](#)

[Para testar os efeitos das modificações dados e definir as condições para desencadear ações LMG.](#)

X

[Para criar um log dos quadros criados ou suprimidos.](#)

Você foi convidado pelo gerente de compras, para preparar os relatórios de fim de ano as compras do departamento, para todos os tipos de produtos. Você também precisa atualizar o preço de produtos específicos. Para fazer isso, você está usando uma transação explícita. Que uma das seguintes afirmações você vai usar no final da operação permanentemente a acrescentar aos dados modificados para o banco de dados?

[BEGIN TRANSACTION](#)

[COMMIT TRANSACTION](#) X

[SAVE TRANSACTION](#)

[ROLLBACK TRANSACTION](#)

O gerente de vendas em Adventure Works tem pediu-lhe que prepare um relatório sobre o volume de negócios realizados em cada um dos departamentos da empresa. Ele também quer um relatório sobre o desconto percentual permitido no que diz respeito a cada produto. Você precisa usar transações aninhadas para preparar o relatório. Qual dos seguintes componentes de uma transação explícita incrementos @ @ TRANCOUNT por um?

[Iniciar a transação](#)

[Repetir a transação sem que a transação Nome](#) X

[A transação](#)

[Repetir a transação com a transação Nome](#)

Inserindo Dados em Tabelas

Você pode inserir dados em uma tabela através da utilização de diferentes métodos baseados em seus negócios exigência. Você pode executar um inserir única linha ou inserir múltiplas linhas. Você pode inserir valores em explícita a identidade coluna de uma tabela. Você também pode usar a cláusula OUTPUT com o INSERT declaração de orientar os dados que foram afectados por qualquer alteração em uma tabela ou uma tabela variável.

Como inserir uma única linha em uma tabela

Você pode usar a declaração INSERT para inserir dados em uma tabela. Você pode inserir uma única linha ou várias linhas em uma tabela.

Inserir declaração sintaxe

O seguinte é a sintaxe parcial da inserção declaração.

```
INSERT
  [INTO]
  View_name) (table_or
(
  [(Column_list)]
  [<OUTPUT Clause>]
  (VALUES ((DEFAULT | NULL | expressão) [, ... n])
  | Derived_table
  | Execute_statement
  )
)
```

O table_name ou view_name é o nome da tabela ou visualizar os dados em que tem de ser inserido e column_list é uma lista de um ou mais colunas nas quais a inserir dados. O INTO palavra-chave é opcional. A cláusula OUTPUT retorna as linhas que foram inseridas como parte de inserir a operação. Os valores cláusula introduz a lista dos valores dos dados a serem inseridos nas colunas de column_list. TheDEFAULT palavra-chave forças do SQL Server Database Engine para carregar o valor padrão definido para uma coluna. Se um valor padrão não existe para a coluna e na coluna permite valores nulos, theNULL valor é inserido. Quando valores não são especificados, o SQL Server pode inserir valores de identidade colunas, colunas com dados tipo timestamp, colunas computadorizada, e colunas com defeito e constrangimentos padrão objeto vinculado. O conceito de expressão é uma constante, uma variável, ou uma expressão.

O derived_table é válido SELECT qualquer declaração que retorna linhas de dados que devem ser carregados em cima da mesa. O resultado conjunto deve ser compatível com a definição do quadro em que as linhas devem ser inseridos. O execute_statement é válido EXECUTE qualquer declaração que retorna os dados com SELECT ou READTEXT declarações. No entanto, a declaração SELECT não pode conter um quadro comum expressão (CTE).

Inserir declarações e listas de colunas

INSERT uma declaração pode ser especificado com ou sem uma coluna lista. Se uma declaração INSERT é especificado sem uma coluna lista, a lista de valores deve corresponder exactamente com a lista de colunas definidas na tabela.

Inserção parcial

Você pode inserir uma completa linha de dados parciais ou uma linha de dados em uma tabela. A inserção parcial só é possível se as colunas que são omitidos ou são identidade colunas, têm valores padrão definido, ou permitir valores NULL. Caso contrário, as linhas não serão inseridos. Você pode inserir explicitamente um valor NULL em uma coluna NULL, especificando a palavra-chave na lista de valores. Este valor NULL explícito que sobrepõe o valor padrão que foi atribuído à coluna.

O código a seguir exemplos demonstram como você pode usar a declaração INSERT para inserir linhas em uma tabela.

A inserção de uma linha sem especificar uma Lista de Coluna
O seguinte exemplo usa o código INSERT declaração para adicionar uma nova linha para o Sales.ShoppingCart mesa, sem indicar explicitamente com uma coluna lista.

```
USE AdventureWorks  
INSERT INTO Sales.ShoppingCartItem  
VALORES (10451,2,718, padrão, padrão)
```

Devido ao facto de a coluna lista não é especificada, 10451 é o valor atribuído a não-identificação da primeira coluna denominada ShoppingCartID, 2 é o valor atribuído a não-identidade da segunda coluna denominada quantidade, e assim por diante. No entanto, recomenda-se a especificar nomes coluna em uma declaração INSERT, já que a consulta mais legível. Além disso, se a estrutura do quadro é alterada em data posterior, a declaração INSERT sem uma coluna lista poderia não ter executado.

A inserção de uma linha especificando Valores
O código a seguir exemplo insere uma linha na tabela Production.UnitMeasure. Os valores para todas as colunas são fornecidos e estão listados na mesma ordem que as colunas da tabela, por isso os nomes de coluna não têm de ser especificada na coluna lista.

```
USE AdventureWorks;  
INSERT INTO Production.UnitMeasure  
VALORES (N'F2 ', N'Square Pés', GETDATE ())
```

A inserção de uma linha especificando uma Lista de Coluna
O seguinte exemplo usa um código coluna lista para indicar explicitamente os valores que estão a ser inserido em cada coluna. A coluna no fim UnitMeasure tabela é UnitMeasureCode, Nome, ModifiedDate; no entanto, as colunas que não estão listados em ordem na lista da coluna.

```
USE AdventureWorks;  
INSERT INTO Production.UnitMeasure (Nome, UnitMeasureCode, ModifiedDate)  
VALORES (N'Square Yards ", N'Y2 ', GETDATE ())
```

Desempenhando uma inserção parcial
O seguinte exemplo usa um código para efectuar uma declaração INSERT parcial e inserir a declaração não explicitamente inserir valores determinados em colunas.

```
USE AdventureWorks;  
INSERT INTO Sales.ShoppingCartItem (ShoppingCartID, quantidade, ProductID)  
VALUES (10451, 2, 718)
```

Nesta consulta, os ShoppingCartItemID é uma coluna de identidade. Além disso, as colunas DateCreated e ModifiedDate tem valores padrão. O valor padrão é adicionado implicitamente a essas colunas porque o valor não foi especificado para as colunas DateCreated e ModifiedDate.

Inserindo um valor na Coluna Identidade
A declaração seguinte código exemplo INSERT tenta atribuir um valor a uma coluna identidade (ShoppingCartItemID).

```
USE AdventureWorks
```

```
INSERT INTO Sales.ShoppingCartItem (ShoppingCartItemID, ShoppingCartID, quantidade, ProductID)
VALORES (7,10451,2,718)
```

A declaração INSERT anterior irá dar o seguinte erro:

Não é possível inserir explícita valor de identidade coluna na tabela 'ShoppingCartItem' quando IDENTITY_INSERT está definido para OFF.

Como inserir múltiplas linhas em uma tabela

Você pode inserir várias linhas em uma tabela, usando duas técnicas. Você pode inserir as linhas em uma tabela existente, usando o INSERT... SELECT. Caso contrário, você pode criar uma tabela e, em seguida, inserir linhas em que, usando a declaração SELECT INTO.

Insert... Select

O SELECT na subconsulta INSERT declaração pode ser usado para adicionar uma tabela de valores para uma ou mais tabelas ou pontos de vista. Ao utilizar uma instrução SELECT, você pode inserir várias linhas simultaneamente.

```
INSERT INTO table_name) [(column_list)]
Query SELECT
```

Nesta sintaxe, table_name é o nome da tabela na qual as linhas retornadas pela consulta SELECT devem ser inseridas. Se uma lista de colunas não for especificada, então a ordem das colunas retornadas pelo SELECT tem de corresponder à ordem das colunas definidas na tabela.

Exemplo

No código a seguir o exemplo insere alguns dos dados a partir da tabela Sales.SalesReason, onde está a ReasonType Marketing, em outra tabela MySalesReason.

```
Use AdventureWorks ;
CREATE TABLE MySalesReason
(
SalesReasonID int NOT NULL,
Name nvarchar(50),
ModifiedDate datetime );
GO
INSERT INTO MySalesReason
SELECT SalesReasonID, Name, ModifiedDate FROM AdventureWorks.Sales.SalesReason
WHERE ReasonType = N'Marketing';
GO
SELECT SalesReasonID, Name, ModifiedDate FROM MySalesReason;
```

Select... Into

A declaração SELECT INTO cria uma nova tabela e preenche-lo com o resultado do conjunto de SELECT. SELECT INTO A declaração pode ser utilizada para combinar dados de várias tabelas ou vistas em uma tabela.

```
SELECT) (column_list
EM) (new_table_name
(FROM table_name | view_name | join_condition)
```

O valor especificado no parâmetro new_table_name será utilizado para indicar o nome da tabela.

Um erro ocorre se você tentar criar uma tabela que já existe. Você deve usar expressões nas colunas para selecionar a lista.

A informação técnica é comum para criar e povoar locais ou globais tabelas temporárias e, em seguida, executar consultas contra eles. Por padrão, estas tabelas temporárias são automaticamente apagados pelo SQL Server quando a sessão é fechada que os criou. No caso das tabelas temporárias globais, elas são excluídas quando nenhuma conexão utiliza-los.

Exemplo

O exemplo a seguir gera a tabela `dbo.EmployeeAddresses`, selecionando as colunas de `Empregados` e `EmployeeAddress` tabelas.

```
USE AdventureWorks ;
SELECT c.FirstName, c.LastName, e.Title, a.AddressLine1, a.City, sp.Name AS
[State/Province], a.PostalCode
INTO dbo.EmployeeAddresses
FROM Person.Contact AS c
JOIN HumanResources.Employee AS e
ON e.ContactID = c.ContactID
JOIN HumanResources.EmployeeAddress AS ea ON ea.EmployeeID = e.EmployeeID
JOIN Person.Address AS a
ON a.AddressID = ea.AddressID
JOIN Person.StateProvince AS sp
ON sp.StateProvinceID = a.StateProvinceID;
```

Como a identidade Inserir Valores em Colunas

Se uma tabela não incluem uma coluna que pode ser usado como uma chave primária, você pode especificar uma coluna em que a tabela a ser uma identidade coluna. Quando você especificar a identidade propriedade de uma coluna, o SQL Server gera automaticamente como valores únicos números sequenciais. Você pode ter uma identidade única coluna em uma tabela.

Parâmetros de coluna identidade

Quando você designar uma determinada coluna em uma tabela como uma identidade coluna, você também pode especificar dois parâmetros opcionais, ou seja, sementes e de incremento. Semente é o valor que é usado para o muito carregado na primeira linha da tabela. Incremento é o valor incremental theidentity que é adicionado ao valor da linha anterior que foi carregado. Você deve especificar valores para os dois parâmetros ou para nenhum. O valor padrão para ambas as sementes e de incremento é 1.

Como criar uma Coluna identidade

Você pode criar uma identidade coluna em uma tabela, especificando a identidade propriedade com o `CREATE TABLE` e `ALTER TABLE` T-SQL declarações. O código seguinte exemplo demonstra como você pode criar uma identidade coluna chamada `id_num` com um valor inicial de 1 de sementes e de incremento de valor 2, utilizando o comando `CREATE TABLE`.

```
USE AdventureWorks
CREATE TABLE new_employees
(
IDENTIDADE id_num int (1,2),
fname VARCHAR (20),
Minit char (1),
```

```
lname VARCHAR (30)  
)
```

A seguinte mensagem é exibida como resultado da execução do referido código.
[Comando \(s\) concluído com êxito.](#)

Inserindo Valores em Colunas identidade

Às vezes, você pode querer reinserir os dados que foram apagados acidentalmente a identidade do mesmo valor que tinha antes da sua supressão. Por padrão, um valor que a identidade tenha sido gerada uma vez não vai ser reutilizados, mesmo que a fila para o qual foi atribuído o valor da identidade tenha sido excluído. Para manter a mesma identidade valor, você pode ativar a opção para IDENTITY_INSERT uma mesa e, em seguida, especificar expressamente os valores de identidade coluna.

Diferenciando os vários tipos de INSERT

Classifique as expressões tão rapidamente quanto possível para os seus associados categorias clicando no balde apropriado. Você também pode utilizar os atalhos de teclado, premindo 1 balde para a esquerda, 2 para o meio balde, e 3 para a direita balde.

Como usar a cláusula OUTPUT com a Declaração INSERT

SQL Server 2005 introduz a nova saída cláusula que permite ao utilizador retornar dados de uma modificação declaração como um INSERT, UPDATE, ou DELETE declaração. Você pode usar a cláusula OUTPUT em combinação com a cláusula INTO para direcionar os dados que foram afectados pela alteração em uma tabela ou uma tabela variável. Esses resultados também podem ser devolvidos ao processamento pedido para envio da confirmação mensagens, arquivo, e outras exigências. A saída cláusula pode ser usada para recuperar o valor de identidade ou colunas computados após um INSERT ou UPDATE operação.

OUTPUT <dml_select_list> [INTO (output_table | @ table_variable)]

Neste sintaxe, a saída <dml_select_list> especifica que a modificadas ou suprimidas valores resultantes de uma INSERT, UPDATE, DELETE ou declaração são devolvidos. A @ table_variable ou output_table especifica uma tabela variável ou tabela que as linhas retornadas são inseridas em vez de ser devolvido ao chamador. A @ table_variable deve ser declarado antes do INSERT, UPDATE, ou DELETE declaração.

O output_table também pode ser uma tabela temporária. Pode participar em ambos os lados de uma chave estrangeira e têm CHECK CONSTRAINT ativado. No entanto, não pode ter permitido desencadeia definida sobre o assunto.

A cláusula OUTPUT com a declaração INSERT

Você pode usar a cláusula OUTPUT com a declaração INSERT para retornar os dados modificados a partir da tabela.

No código a seguir exemplo, você pode inserir uma linha na tabela ScrapReason e usar a cláusula OUTPUT para retornar os resultados da sua declaração à @ MyTableVar tabela variável. Porque é uma identidade ScrapReasonID coluna, um valor não é especificado na declaração INSERT para essa coluna.

O valor da base de dados gerada pelo motor para que a coluna é retornado na saída cláusula na coluna INSERTED.ScrapReasonID.

```
USE AdventureWorks
DECLARE @MyTableVar table( ScrapReasonID smallint,
Name varchar(50),
ModifiedDate datetime);
INSERT Production.ScrapReason
OUTPUT INSERTED.ScrapReasonID, INSERTED.Name, INSERTED.ModifiedDate
INTO @MyTableVar
VALUES (N'Operator error', GETDATE());
SELECT ScrapReasonID, Name, ModifiedDate FROM @MyTableVar;
```

ScrapReasonID	Name	ModifiedDate
17	Operator error	2006-01-31 17:54:21.140

Testes

Cada azulejo contém uma verdadeira declaração e de uma falsa declaração. Clique nos azulejos em uma linha ou coluna até que cada azulejo mostre a verdadeira declaração e, em seguida, clique na seta ao lado da linha ou superior da coluna para verificar a sua resposta. Você usa um turno quando você clicar Enviar uma incorrecta ou azulejo. Remover todas as peças antes de utilizar todas as suas voltas.

Deletando dados de tabelas

Dependendo de suas necessidades empresariais, você pode utilizar várias técnicas para apagar dados de uma tabela em seu banco de dados. Ao usar a declaração DELETE, você pode excluir uma única linha ou várias linhas de uma tabela. Você pode apagar todas as linhas de uma tabela usando o TRUNCATE TABLE declaração. Ao usar a cláusula JOIN ou de uma subquery, você pode excluir linhas baseadas em outras tabelas. Você pode usar a cláusula OUTPUT externamente referente a tabela temporária excluída que é usada pelo SQL Server durante o apagamento.

Como deletar linhas de uma tabela

A declaração DELETE é usada para remover uma linha ou conjunto de linhas de uma tabela.

```
DELETE
[TOP (expression) [PERCENT]]
[De]
(Table_name | view_name)
ONDE (<search_condition>
```

Neste sintaxe, TOP (expression) [PERCENT] especifica o número ou o acaso por cento das linhas que podem ser suprimidas. O table_name ou view_name é o nome da tabela ou visão a partir da qual as linhas estão a ser removidos. Uma tabela variável pode também ser usado como uma fonte em uma tabela DELETE declaração. A cláusula FROM é uma palavra chave opcional que pode ser utilizado entre as palavras-chave e DELETE table_or_view_name o alvo, ou rowset_function_limited. A cláusula WHERE é usada para especificar as condições utilizadas para limitar o número de linhas suprimidas. Se uma cláusula WHERE não é utilizado, a declaração DELETE remove todas as linhas da tabela.

Duas formas de apagar operações

Existem duas formas de eliminar operações com base naquilo que é especificado na cláusula WHERE:

DELETAR BUSCA - Você precisa especificar uma condição de pesquisa para se qualificar para eliminar

as linhas. Por exemplo, quando column_name = valor.

DELETAR POSIÇÃO - Você precisa usar a corrente de cláusula de especificar um cursor. A operação ocorre em suprimir a atual posição do cursor. Isso pode ser mais precisos do que uma declaração DELETE pesquisados que utiliza uma cláusula WHERE search_condition para qualificar as linhas a serem suprimidos. Uma declaração pesquisados DELETE deleta múltiplas linhas, se a pesquisa condição não o identificar uma única linha.

O exemplo a seguir mostra a forma de eliminar todas as encomendas feitas por um determinado vendedor.

```
DELETE FROM Sales.SalesOrderHeader
WHERE SalesPersonID = 275
```

Esta declaração DELETE remove uma única linha de cada vez e registra uma entrada na tabela para cada Suprimido Suprimido fila.

Utilizando uma transação explícita para excluir linhas

É comum para iniciar uma transação explícita antes de eliminar quaisquer dados de uma tabela. Você pode verificar se a declaração tem apagar as linhas afectadas através da criação de uma transação explícita. Você também pode assegurar que você não tenha apagado acidentalmente linhas, indicando uma condição errado na cláusula WHERE, antes de começarmos a operação. O exemplo a seguir mostra a forma de eliminar todas as encomendas efectuadas por um vendedor particular antes de um determinado ano. Você pode iniciar uma transação explícita para testar a excluir qualquer declaração e, em seguida, voltar a cometer ou ação.

```
USE AdventureWorks
BEGIN TRANSACTION
GO
DELETE FROM Sales.SalesOrderHeader
WHERE SalesPersonID = 290 AND Year(OrderDate) = 2004
GO
```

O que se segue é a mensagem da consulta.

56 linha (s) são afetadas.

Você também pode verificar se as linhas tenham sido suprimidas, utilizando a seguinte consulta.

```
SELECT * FROM Sales.SalesOrderHeader
WHERE SalesPersonID = 290
```

O que se segue é a mensagem da consulta.

53 linha (s) são devolvidos

Você vai achar que as linhas foram mal suprimidas porque você quis excluir os despachos do vendedor antes de um determinado ano e não durante esse ano. Por isso, precisa de rollback e inverter os efeitos da declaração anterior DELETE.

```
REPEAT TRANSACTION
GO
```

Como truncar uma tabela

A declaração TRUNCATE TABLE é um rápido, e não autenticado método de exclusão de todas as linhas de uma tabela. A funcionalidade do TRUNCATE TABLE declaração é semelhante à da declaração DELETE sem uma cláusula WHERE. No entanto, a declaração TRUNCATE TABLE é comparativamente mais rápido e utiliza menos sistema de recursos e de operação rápida do que a

declaração DELETE sem uma cláusula WHERE.

TRUNCATE TABLE

```
[(Database_name. [Schema_name]. | Schema_name. )]  
table_name  
[:]
```

Nesta sintaxe, o database_name é o nome do banco de dados, o schema_name é o nome do esquema ao qual pertence a mesa, e os table_name é o nome da tabela para truncar ou a partir da qual todas as linhas são removidos. Quando table_name é o nome da tabela cujas linhas estão a ser excluído. Ao contrário da declaração DELETE, TRUNCATE TABLE a declaração não tem uma cláusula WHERE. Por isso, ele apaga todas as colunas na tabela.

TRUNCATE TABLE A declaração prevê as seguintes vantagens sobre a declaração DELETE das seguintes maneiras.

Menos espaço no log da transação é usado.

A declaração DELETE remove uma linha de cada vez, e registra uma entrada no log para cada transação suprimido fila. Considerando que, a TRUNCATE TABLE remove declaração de-dados pela atribuição dos dados páginas que são usados para armazenar dados e de mesa apenas os registros de-página atribuições na operação rápida.

Menos exclusões são tipicamente usadas.

Quando você executar o DELETE declaração, utilizando uma linha lock, cada linha da tabela está bloqueada para a eliminação. Considerando que, a declaração TRUNCATE TABLE sempre trava a tabela e de página, mas não cada linha.

Zero páginas são deixadas na tabela.

Depois de executar um DELETE declaração, o quadro ainda pode conter páginas em branco se a declaração não foi capaz de adquirir uma tabela lock.

Truncar tabela e soltar tabela

A declaração TRUNCATE TABLE remove todas as linhas de uma tabela. No entanto, a estrutura eo quadro suprimido suas colunas, constrangimentos, índices e permanecerá assim por diante. Similar ao DELETE declaração, a definição de um quadro esvaziados pela utilização do TRUNCATE TABLE afirmação permanece no banco de dados com os seus índices e outros objectos associados.

Para remover a tabela definição, para além dos seus dados, você deve usar o DROP TABLE declaração.

Tabelas com colunas identidade

Se a tabela contém uma identidade coluna, a coluna é contraproducente para que a redefinir as sementes valor definido para a coluna. Se as sementes não foi definida, o valor padrão 1 é usado. Para manter a identidade balcão, use a palavra-chave DELETE vez.

Restrições para truncar tabela

Você não pode usar TRUNCATE TABLE declaração sobre os quadros que:

São referenciados por uma chave estrangeira.

Participar em uma vista indexada.

São publicados através de replicação transacional ou fundir replicação.

Para tabelas com uma ou mais destas características, você precisará usar o DELETE declaração TRUNCATE TABLE em vez de a declaração. Você não pode usar o TRUNCATE TABLE declaração de ativar um gatilho, porque não log individual fila rasuras.

O exemplo seguinte código remove todos os dados a partir da JobCandidate tabela. Você pode usar o SELECT declarações antes e depois da declaração TRUNCATE TABLE para comparar os resultados. Os resultados painel exibe o número de linhas que foram apagadas.

Use AdventureWorks

```
SELECT COUNT(*) AS BeforeTruncateCount
FROM HumanResources.JobCandidate;
GO
TRUNCATE TABLE HumanResources.JobCandidate;
GO
SELECT COUNT(*) AS AfterTruncateCount
FROM HumanResources.JobCandidate;
GO
```

Como deletar linhas baseando-se em outras tabelas

Usando JOIN

Você pode excluir as linhas de uma tabela, com base em dados de outra tabela, usando a aderir condição.

```
DELETE
(FROM table_name | view_name)
[JOIN condição]
[WHERE] (<search_condition>]
```

Se você quiser excluir os dados de uma tabela baseada em dados presentes em outra tabela, você precisa de duas das cláusulas, em sua declaração. A primeira cláusula FROM indica a tabela a partir da qual as linhas estão a ser excluído. A segunda cláusula FROM pode introduzir uma condição participar e atuar como uma restrição à declaração DELETE.

Usando Subconsulta

Você também pode apagar as linhas de uma tabela, com base em dados de outra tabela, usando uma subconsulta.

```
DELETE
(FROM table_name | view_name)
[EM CASO colum_name) (<subquery>]
```

Neste sintaxe, a subconsulta neste caso, pode ser tanto uma base ou uma subconsulta correlacionada.

Esta é a técnica de compilação ANSI SQL para suprimir linhas em uma tabela baseada em linhas em outra mesa. A alternativa de utilizar um join para apagar linhas é uma extensão T-SQL.

Você pode reescrever a declaração DELETE utilizando uma subconsulta base em vez de juntar uma condição.

Como usar a cláusula OUTPUT com a Declaração DELETE

Você pode retornar dados de uma INSERT, UPDATE, DELETE ou declaração, utilizando a cláusula OUTPUT do SQL Server. Você pode usar a cláusula OUTPUT com a cláusula INTO para dirigir os dados que foram afectados pela alteração em uma tabela ou uma tabela variável.

A cláusula OUTPUT informações de volta, ou baseados em expressões, cada linha afectada por um INSERT, UPDATE, ou DELETE declaração. Estes resultados podem ser devolvidos ao pedido de transformação, para ser utilizado em mensagens confirmação, arquivo, e outras, tais exigências

aplicação. Você também pode inserir os resultados em uma tabela ou uma tabela variável.

OUTPUT <dml_select_list> [INTO (output_table | @ table_variable)]

O argumento, @ table_variable ou output_table especifica uma tabela variável ou uma tabela onde as linhas retornadas são armazenadas, em vez de serem devolvidas ao chamador. Você precisa declarar o @ table_variable antes de executar o INSERT, UPDATE, DELETE ou declarações.

Você pode acessar a nova ou velha imagem das linhas modificadas, referindo-se à inserido e DELETED tabelas, da mesma forma que você acessar a desencadeia. Em uma declaração DELETE, no entanto, é-lhe permitido o acesso apenas a tabela DELETED.

Se você especificar uma cláusula OUTPUT sem a cláusula INTO, o resultado é retornado para o chamador. Quando você quer directa de dados a uma tabela ou a uma tabela variável, pode fazê-lo, especificando a cláusula OUTPUT ambos com e sem a cláusula INTO, na mesma modificação declaração. Desta forma, você também pode retornar os dados para o chamador.

O código a seguir exemplo apaga todas as colunas na tabela ShoppingCartItem.

A cláusula OUTPUT DELETED .* especifica que os resultados da declaração DELETE, ou seja, as linhas suprimido, devem ser devolvidas à chamada candidatura. A declaração SELECT depois da cláusula OUTPUT DELETED verifica os resultados da operação sobre a apagar ShoppingCartItem tabela.

[Use AdventureWorks](#)

```
DELETE Sales.ShoppingCartItem OUTPUT DELETED .*  
GO
```

O seguinte é o resultado parcial conjunto da consulta.

ShoppingCartItemID	ShoppingCartID	quantidade	ProductID	DateCreated	ModifiedDate
6	10451	2	718	2006-01-24 16:39:56.543	2006-01-24 16:39:56.543
7	10451	2	718	2006-01-24 16:40:25.513	2006-01-24 16:40:25.513
2	14951	3	862	2003-12-11 17:54:07.603	2003-12-11 17:54:07.603
5	20621	7	874	2003-12-11 17:54:07.603	2003-12-11 17:54:07.603
4	20621	4	881	2003-12-11 17:54:07.603	2003-12-11 17:54:07.603

Agora, você precisa de verificar se todas as colunas na tabela foram excluídos.

```
SELECT COUNT (*) AS [Linhas na Tabela] FROM Sales.ShoppingCartItem;
```

O seguinte é o resultado conjunto de completar a consulta.

```
RowsInTable  
0
```

O código a seguir exemplo deleta linhas na tabela baseada em pesquisa ProductProductPhoto condições especificadas na cláusula FROM da declaração DELETE. A cláusula OUTPUT retorna colunas da tabela a ser excluída e colunas da tabela Produto. Esta tabela é utilizado na cláusula FROM para especificar as linhas que estão a ser excluído.

```

Use AdventureWorks
DECLARAM @ MyTableVar tabela (
    ProductID INT NOT NULL,
    ProductName nvarchar (50) NOT NULL,
    ProductModelID INT NOT NULL,
    PhotoID int NOT NULL);
DELETE Production.ProductProductPhoto
OUTPUT DELETED.ProductID, p.Name, p.ProductModelID,
DELETED.ProductPhotoID
EM @ MyTableVar
A partir de Production.ProductProductPhoto como pH
JOIN Production.Product como p
ON ph.ProductID = p.ProductID
ONDE p.ProductModelID entre 120 e 130;

```

```

-- Apresentar os resultados da tabela variável.
SELECT ProductID, ProductName, ProductModelID, PhotoID
DESDE @ MyTableVar
Por fim ProductModelID;
GO

```

O seguinte é o resultado parcial conjunto da consulta.

ProductID	ProductName	ProductModelID	PhotoID
842	Touring-cestos, Large	120	1
878	Fender Set - Montanha	121	1

Questões

Qual dos seguintes argumentos da declaração DELETE é usado para limitar o numero de linhas deletadas?

```

TOP (expressão)[PERCENT]
Table_name
WHERE X
VIEW_NAME

```

Qual das seguintes declarações você usaria para deletar todas as linhas de uma tabela?

```

DELETE e uma condição join
DELETE e uma subconsulta
TRUNCATE TABLE
DROP TABLE

```

Qual das seguintes declarações DELETE você usaria para não deletar linhas acidentalmente?

```

DELETE com OUTPUT
DELETE explicito X
DELETE sem WHERE
DELETE com uma condição join

```

Atualizando dados em Tabelas

Você pode atualizar registros em uma tabela usando o UPDATE declaração. Você pode usar o UPDATE declaração com a cláusula FROM para modificar uma tabela baseada em valores a partir de outras tabelas. Você pode usar a cláusula OUTPUT com a declaração UPDATE para obter informações sobre as linhas afectadas pela UPDATE declaração.

Como atualizar linhas em uma tabela

A declaração UPDATE pode ser usada para alterar os valores dos dados para uma única linha, conjunto de linhas, ou todas as linhas de uma tabela ou vista. A expressão UPDATE de uma VIEW pode alterar os dados em uma única tabela base de uma só vez.

```
UPDATE
[TOP (expression) [PERCENT]]
(table_name | SET view_name) (column_name
SET
    column_name = ((expressão | DEFAULT | NULL),..., n]
FROM) (table_source
WHERE) (<search_condition>
```

Nesta sintaxe:

A cláusula SET contém uma lista de colunas que estão a ser actualizados e do novo valor para cada coluna.

A cláusula FROM identifica as tabelas ou opiniões que fornecer os valores para as expressões na cláusula SET e facultativo aderir condições entre a fonte tabelas ou pontos de vista.

A cláusula WHERE define as linhas da tabela fonte que devem ser atualizados. A cláusula WHERE pode também indicar as linhas a partir da fonte quadros que se qualificam para fornecer valores para a atualização se uma cláusula FROM também é especificado. Se nenhuma cláusula WHERE é especificado na tabela a todas as linhas são atualizadas.

A entrada valores que estão a ser passado para o UPDATE declaração deve ser compatível com o tipo de dados subjacentes à coluna. Se os tipos de dados não correspondem exatamente, o SQL Server irá tentar converter os tipos de dados implicitamente. Se conversão implícita falhar, irá ocorrer um erro.

Em cada linha que é atualizado com SQL Server, uma imagem antes e depois da linha deve ser gravado através do suprimidos e aditados tabelas. Condicional atualizações podem ser realizadas usando a case.

Transação Explícita

Ao iniciar uma transação explícita declaração antes de executar uma atualização, você pode ver os efeitos da atualização. Você pode iniciar a operação, especificando a iniciar a transação declaração. Então, você pode executar o UPDATE declaração. Você pode usar a declaração SELECT para testar se os dados tenham sido modificados corretamente. Se ele foi modificado corretamente, você pode especificar a declaração a transação, caso contrário você pode especificar a repetir a transação declaração.

Como atualizar as linhas baseadas em outras tabelas

Você pode usar uma declaração UPDATE com a cláusula FROM, ou com uma subquery para modificar uma tabela baseada em valores de outras tabelas. O seguinte é a sintaxe parcial da declaração UPDATE.

```
UPDATE (table_name | view_name)
SET column_name = ((expressão | DEFAULT | NULL) [, N]
[FROM (<table_source>]
[WHERE search_conditions]
```

Usando Join

Você pode usar junta de limitar o número de linhas afetadas pela UPDATE declaração. O código exemplo duplica o valor na coluna ListPrice no Production.Product tabela de ProductID s pertencentes a um determinado VendorID usando junta.

Usando Subconsulta

Você também pode usar subqueries para limitar o número de linhas afetadas pela UPDATE declaração. Este é o método ANSI SQL. Você pode usar básica e subqueries correlacionadas. O exemplo usa uma subquery para atualizar uma tabela baseada em valores de outras tabelas.

É importante que sua subconsulta retorna um valor único, porque SQL Server nunca atualiza a mesma linha duas vezes em uma única declaração UPDATE. Assim, quando você usar subconsultas correlacionadas, você precisará usar funções agregadas.

Exemplo:

```
USE AdventureWorks ;
UPDATE Sales.SalesPerson
SET SalesYTD = SalesYTD + SubTotal
FROM Sales.SalesPerson AS sp
JOIN Sales.SalesOrderHeader AS so
ON sp.SalesPersonID = so.SalesPersonID
AND so.OrderDate =
(SELECT MAX(OrderDate)
FROM Sales.SalesOrderHeader
WHERE SalesPersonID = sp.SalesPersonID);
```

Como usar a cláusula OUTPUT com a Declaração UPDATE

Você pode retornar dados de uma declaração UPDATE usando a cláusula OUTPUT do SQL Server. A saída pode ser devolvido ao processamento ou aplicação pode ser inserido em uma tabela ou uma tabela variável.

`OUTPUT <dml_select_list> [INTO (output_table | @ table_variable)]`

@ table_variable ou output_table: Esta especifica uma tabela variável ou tabela onde as linhas retornadas são inseridas. Também deveria ser declarado antes da declaração UPDATE é executado. Durante uma atualização, você pode ver a antiga ea nova imagem das linhas modificadas por referência aos quadros suprimidos e aditados respectivamente. Você pode especificar a cláusula OUTPUT com ou sem o uso de INTO na mesma declaração.

Exemplo:

```
USE AdventureWorks
DECLARE @MyTestVar table (
OldScrapReasonID int NOT NULL,
NewScrapReasonID int NOT NULL,
WorkOrderID int NOT NULL,
ProductID int NOT NULL,
ProductName nvarchar(50)NOT NULL);
UPDATE Production.WorkOrder SET ScrapReasonID = 4
OUTPUT DELETED.ScrapReasonID, INSERTED.ScrapReasonID,
INSERTED.WorkOrderID,INSERTED.ProductID, p.Name INTO @MyTestVar
FROM Production.WorkOrder AS wo
INNER JOIN Production.Product AS p ON wo.ProductID = p.ProductID
AND wo.ScrapReasonID= 16 AND p.ProductID = 733;
SELECT OldScrapReasonID, NewScrapReasonID, WorkOrderID, ProductID, ProductName
FROM @MyTestVar;
```



OldScrapReasonID	NewScrapReasonID	WorkOrderID	ProductID	ProductName
------------------	------------------	-------------	-----------	-------------



16	4	11696	733	ML Road Frame - Red, 52
16	4	13096	733	ML Road Frame - Red, 52



Questões

Qual das seguintes afirmações é verdadeira sobre as cláusulas usada em uma declaração UPDATE?

A cláusula SET contém uma lista de linhas que estão a ser actualizados e do novo valor para cada coluna.

A cláusula WHERE identifica as tabelas ou opiniões que fornecer os valores para as expressões na cláusula SET.

A cláusula WHERE indica as linhas a partir da fonte quadros que se qualificam para fornecer valores para a atualização se uma cláusula FROM é especificado.

Se a cláusula WHERE é especificado, todas as colunas na tabela são atualizados.

Qual das opções a seguir é um método de actualização de linhas baseadas em outros quadros?

Uma atualização declaração pode ser usada com uma cláusula WHERE para modificar uma tabela baseada em valores de outras tabelas.

Você pode usar subqueries para limitar o número de linhas afetadas pela UPDATE declaração. Este é o método não-ANSI SQL.

Você pode usar junta de limitar o número de linhas afetadas pela UPDATE declaração.

Você pode usar funções agregadas com a junta, porque SQL Server actualiza a mesma linha duas vezes em uma única declaração UPDATE.

Qual dos seguintes é verdade acerca da cláusula OUTPUT do UPDATE declaração?

@ table_variable especifica um quadro onde a variável retornou linhas são inseridas.

Você pode retornar dados de uma declaração UPDATE usando a cláusula WHERE do SQL Server.

Você pode usar a cláusula FROM INTO com a cláusula de orientar os dados em uma tabela ou uma tabela variável.

O output_table não precisa ser declarado antes do UPDATE declaração.

Consultando Metadados, XML e Índices Full-Text no SQL Server 2005

Na maior parte das empresas, os dados já não está localizado em um único sistema, mas armazenados em vários sistemas em vários formatos. Você pode muitas vezes necessidade de gerar relatórios por examinando os dados nesses formatos. Para fazer isso, o SQL Server suporta diversas categorias de dados, tais como dados estruturados, semi-estruturadas de dados, e dados não estruturados. Você pode utilizar as diversas funcionalidades no SQL Server para consultar metadados, dados em XML, e índices de texto completo. Ao utilizar o índice de texto completo recurso em SQL Server, você pode fornecer funcionalidade de pesquisa avançada em um banco de dados.

Consultando Metadados

O acesso aos metadados informação ajuda você a encontrar as informações de que efectivamente lhe exigem. O SQL Server suporta diversas categorias de dados, tais como dados estruturados, semi-estruturadas de dados, e dados não estruturados. Em SQL Server 2005, você pode consultar metadados que estão expostas através de catálogo relacional objectos, tais como pontos de vista, INFORMATION_SCHEMA pontos de vista, ou Gestão de Acessos Dinâmico (DMVs). Você também pode consultar metadados usando comandos SQL Server como o sistema funciona, procedimentos armazenados, e Base de Dados Comandos Console (DBCCs).

Categorias de Dados

Se os seus dados é altamente estruturado com esquemas conhecidos e, em seguida, o modelo relacional poderia ser a melhor forma de armazenamento de dados. No entanto, se a estrutura dos seus dados é flexível ou desconhecido, então você precisa para determinar outras técnicas para armazenar essa entrevista semi-estruturada ou não estruturada de dados. SQL Server 2005 fornece

as funcionalidades e as ferramentas necessárias para lidar com estruturados, semi-estruturados, não estruturados e de dados.

Estruturados

Dados estruturados são organizados em grupos ou entidades semânticas. Entidades semelhantes são agrupadas por categorias ou utilizando as relações. Entidades do mesmo grupo têm a mesma descrição ou atributos. Além disso, descrições podem ser fornecidas para todas as entidades em um grupo, utilizando um esquema.

Dados estruturados, geralmente, é armazenado em um banco de dados relacional Management System (RDBMS) e é também referido como dados relacionais. Os dados são armazenados sob a forma de linhas e colunas. As colunas permitem a você restringir o tipo e a quantidade de dados que podem ser armazenados.

Dados estruturados proporciona-lhe várias vantagens. Você pode normalizar os seus dados em várias tabelas, reduzindo, assim, redundância. Você também pode criar estruturas de dados eficiente para minimizar o tamanho dos dados. Além disso, é possível controlar os dados, utilizando elementos diversos constrangimentos e fazer cumprir as normas para a integridade dos dados.

No entanto, dados estruturados tem alguns inconvenientes também. É difícil mudar a estrutura dos dados quando muda drasticamente a sua candidatura. Dados com muitos parâmetros de entrada variável pode resultar na criação de muitas colunas exploração apenas NULOS.

Porque os dados são armazenados em um formato binário proprietário, não pode interagir diretamente com outros sistemas externos.

Semi-Estruturados

Semi-estruturados dados são dados que está disponível eletronicamente em sistemas de arquivos ou formatos de troca de dados, tais como dados bibliográficos ou dados científicos. Semi-estruturados dados é muitas vezes organizados em entidades semântica onde entidades semelhantes são agrupadas. No entanto, entidades do mesmo grupo pode não ter os mesmos atributos.

XML é largamente utilizado um formulário de dados semi-estruturados. Semi-estruturados dados é adequado para cenários onde a estrutura dos dados não é completamente conhecida ou seja susceptível de alterar significativamente no futuro. Os dados podem ser recuperados exactamente da mesma forma que foi armazenada.

Porque XML é um padrão aberto, os dados podem ser utilizados pelos vários sistemas rodando em plataformas diferentes, sem qualquer alteração. Segurança pode ser implementado pela atribuição de vários níveis de permissões para diferentes usuários.

XML é também útil para o intercâmbio de dados entre vagamente acoplados, sistemas díspares, como as aplicações B2B e aplicações workflow. XML é uma forma adequada de armazenamento de dados em aplicativos de gestão documental, onde a maioria dos documentos, tais como mensagens de correio electrónico e os relatórios são semi-estruturados.

Usando o XML tem algumas desvantagens também. Todas as unidades de dados tem que ser marcado com tags e esta aumenta espaço de armazenamento. Isto também aumenta a quantidade de largura de banda necessária para uma unidade de transferência de dados de um sistema para outro.

Porque dados XML é representado em um formato de texto, os dados podem ser modificados. Além disso, que as restrições são mais relaxada.

Não-Estruturados

Dados não estruturados poderia ser de qualquer tipo. É possível que não cumpram as normas e não é previsível. Exemplos de dados não estruturados pode incluir texto, vídeos, sons, imagens, mensagens de e-mail, memorandos, grupos de usuários, bate-papos, relatórios, cartas, pesquisas, livros brancos, material de marketing, pesquisas, apresentações e páginas na web.

Fornecer alguns dados não estruturados vantagens. O formato dos dados pode ser alterado a qualquer momento e sem qualquer grande esforço. Os dados podem ser representados em várias formas para atender às exigências aplicação. Além disso, você pode armazenar vários formatos de dados no mesmo documento.

Dados não estruturados tem muitas desvantagens. Você não pode facilmente definir os vários elementos dos dados, uma vez que geralmente é armazenado como uma série de valores delimitados por marcadores, por exemplo, valores separados por vírgula (CSVs).

Segurança não pode ser definida em texto dados. Porque não se pode ter um mecanismo de indexação, o desempenho pode ser demorado se o tamanho dos dados é muito grande.

Agrupando Conceitos relacionados a diferentes tipos de dados

Classificar os recursos o mais rapidamente possível para os seus associados categorias clicando no balde apropriado. Você também pode utilizar os atalhos de teclado, premindo 1 balde para a esquerda, 2 para o meio balde, e 3 para a direita balde.

O que é Metadado ?

Metadados são definidos como dados sobre dados. Os usuários acesso metadados para obter mais informações e técnicas. Metadados vista proporcionar uma interna, sistema independente de mesa ponto de vista do SQL Server metadados, e conter todos os dados armazenados em objetos que especial banco de dados. Metadados consiste em duas categorias, nomeadamente, de negócios e de metadados técnicos metadados.

Categorias de Metadados

Metadados é geralmente utilizado para fornecer aos usuários com informações adicionais sobre os dados de que o acesso ou a ocultar a complexidade dos detalhes técnicos dos dados que ver. Também pode ser utilizado pelos sistemas de determinar os tipos de dados, o tempo dos dados foi alterado, e que tipo de alterações foram feitas para os dados.

Os metadados podem ser utilizadas pelas aplicações e sistemas para realizar tipo de verificação, validação e dados auto formatação de dados. Metadados consiste em duas categorias: de negócios metadados, técnico e metadados.

Metadados de negócios inclui metadados que um usuário pode acessar tais como nomes de tabela, na coluna nomes, descrições coluna, formatação de regras, as empresas ou definições de elementos de dados.

Em contrapartida, técnicos metadados contém informações que suporta o back office e as outras funções internas do capacidades oferecidas pelo SQL Server 2005 através de elementos como SQL Server Integration Services, SQL Server Analysis Services e SQL Server Reporting Services.

Gerenciamento de views dinâmicas

Gestão Dinâmica Exibições (DMVs) retorno servidor estadual informações que podem ser usados para controlar o servidor, verifique se há problemas, e para facilitar o desempenho. DMVs retornar dados específicos que podem mudar nos próximos lançamentos do SQL Server. Os dois tipos de DMVs se alcance DMVs-servidor e banco de dados de alcance DMVs.

Todas as funções e DMVs existem no esquema sys e seguir as convenções de nomenclatura dm_*. Você deve prefixo o nome da vista ou função, utilizando o esquema sys.

Exemplo:

O código seguinte exemplo mostra uma consulta que faz uso de uma DMV dm_exec_sessions chamado a dar informações sobre o número de usuários conectados atualmente com o número de sessões para cada um deles.

```
USE AdventureWorks
SELECT login_name, count (session_ID) AS session_count
FROM sys.dm_exec_sessions
GROUP BY login_name
```

O código a seguir mostra como um exemplo a consulta junta dm_exec_cursors função e dm_exec_sessions vista o regresso da cursores que foram abertos no servidor por mais de um período de tempo específico, que os criou, e por aquilo que estão a sessão. Neste exemplo, alguns dos pontos de vista utilizados são realmente funciona.

USE AdventureWorks

```
SELECT creation_time, cursor_ID, nome, c.session_ID, login_name
```

```
DESDE sys.dm_exec_cursors (0)
```

```
COMO PARTICIPAR c sys.dm_exec_sessions AS s
```

```
ON c.session_ID = s.session_ID
```

```
ONDE datediff (m, c.creation_time, getdate ())> 30
```

Views de Catálogo

Views de Catálogo retornam a informação que é usada pelo motor de banco de dados SQL Server 2005. Você pode usar views de catálogo para consultar metadados porque elas transformam presentes personalizados e formas de informação de forma eficiente. Todos os usuários de metadados disponíveis catálogo estão expostos através de views de catálogo. Algumas opiniões catálogo herdar linhas catálogo de outros pontos de vista. Por exemplo, ver os quadros catálogo herda os objetos catálogo vista. Isto significa que as tabelas catálogo opinião tem as colunas que são específicas de quadros, e também todas as colunas que pertencem aos objetos catálogo vista. Catalogar todos os pontos de vista estão presentes em um sistema de nível esquema chamado SYS.

Exemplo:

O código seguinte exemplo mostra uma consulta que retorna uma lista de todos os servidores à escala configuração opções e seus valores correspondentes.

USE AdventureWorks

```
SELECT * FROM sys.configurations
```

O código seguinte exemplo mostra uma consulta que retorna a lista de todos os bancos de dados que estão presentes no servidor, juntamente com a sua data de criação e de recuperação modelo.

USE AdventureWorks

```
SELECT Nome, create_date, recovery_model_desc
```

```
DESDE sys.databases
```

Como consultar metadados usando comandos SQL Server 2005

Você pode usar funções de sistema, banco de dados Comandos Console (DBCC), eo sistema de acesso a procedimentos armazenados metadados informação.

Funções de Sistema

Funções de Sistema executam operações em metadados e retornar informações sobre os valores, objetos e configurações no SQL Server 2005. A seguir, são algumas das funções do sistema.

SYSTEM_USER

SERVERPROPERTY

OBJECTPROPERTY

CURRENT_TIMESTAMP

CURRENT_USER

DATALength

ISNUMERIC

Exemplo:

SYSTEM_USER A função pode ser usada para retornar o nome do contexto executando atualmente. O exemplo a seguir declara uma variável CHAR, armazena o valor corrente de SYSTEM_USER na variável e, em seguida, imprime o valor armazenado na variável.

USE AdventureWorks

```
DECLARAM @ sys_usr char (30);
```

```
SET @ sys_usr = SYSTEM_USER;
```

```
SELECT 'O actual sistema de usuário é:' + @ sys_usr;
```

O exemplo a seguir utiliza a função SERVERPROPERTY para obter o nome do servidor e para verificar

se o servidor usa apenas Autenticação Integrada do Windows.

USE AdventureWorks

SELECT SERVERPROPERTY ('servidor'), SERVERPROPERTY ('IsIntegratedSecurityOnly')

DBCC

DBCC comandos pode ser usado não apenas para o regresso de vários tipos de informação, mas também para a realização das tarefas a manutenção de uma base de dados, índices ou filegroup, validação operações em bases de dados ou tabelas, e também para diversas tarefas, tais como bandeiras que permitam rastrear ou removendo DLLs da memória . A seguir estão alguns dos comandos DBCC.

DBCC OPENTRAN

DBCC USEROPTIONS

DBCC CONCURRENCYVIOLATION

DBCC SHOW_STATISTICS

DBCC CHECKALLOC

DBCC CHECKFILEGROUP

DBCC SHOWCONTIG

DBCC HELP

O exemplo a seguir é um comando DBCC que retorna todos os SET opções que são ativos para a conexão atual.

USE AdventureWorks

DBCC USEROPTIONS

O comando a seguir exibe informações fragmentação HumanResources.Employee para a tabela.

USE AdventureWorks

DBCC SHOWCONTIG ('HumanResources.Employee')

Stored Procedures de Sistema

Sistema de procedimentos armazenados, em SQL Server 2005, podem ser utilizados para a realização de actividades administrativas e informativos. A seguir estão alguns dos procedimentos armazenados no sistema.

sp_column_privileges

sp_special_columns

sp_cursor_list

sp_add_maintenance_plan

sp_catalogs

sp_send_dbmail

sp_fulltext_database

sp_spaceused

sp_help

Exemplo:

O exemplo a seguir retorna informações de uma coluna específica coluna usando a sp_columns.

USE AdventureWorks

GO

EXEC sp_columns @ table_name = N'Department ',

@ table_owner = N'HumanResources'

O exemplo a seguir desativa a indexação do texto completo para o banco de dados AdventureWorks.

USE AdventureWorks

GO

EXEC sp_spaceused N'Production. Produto ';

GO

Questões

Você é administrador do banco de dados em Adventure Works. Você foi convidado a descobrir qual SET opções estão ativos para a conexão atual. Qual dos seguintes comandos ou funções que você usa para saber os detalhes?

[DBCC OPENTRAN.](#)

[DBCC SHOWCONTIG.](#)

[DBCC USEROPTIONS. X](#)

[DBCC](#)

[CONCURRENCYVIOLATION](#)

Você precisa os metadados de todas as informações utilizadas pelo Banco de dados SQL Server motor. Que uma das seguintes vistas teria que usar para obter as informações que você precisa?

[INFORMATION_SCHEMA esquema vista.](#)

[Catálogo pontos de vista. X](#)

[Compatível com pontos de vista.](#)

[Gestão de vista dinâmico.](#)

Resumo de XML

A Extensible Markup Language (XML) é uma linguagem de marcação de aplicação geral recomendada pelo World Wide Web Consortium (W3C), para a criação de efeitos especiais de linguagens de marcação, capaz de descrever vários tipos de dados. XML simplificado é um subconjunto de SGML (SGML). Os dados podem ser na forma de um registro de uma loja ou dados de todo o documento. Quando você armazenar dados XML em um banco de dados relacional, fornece-lhe os benefícios de uma melhor gestão de dados e consulta transformação. O SQL Server fornece poderosas capacidades para consultar e modificar dados XML.

O que é XML?

XML baseada em texto prevê um meio para descrever e aplicar uma estrutura de árvore à base de informações. XML tem a capacidade de representar mais estruturas de dados, tais como registros, listas e árvores. A sintaxe e rigorosos requisitos de parsing XML permitir que o necessário analisar algoritmos para ajudá-la a permanecer simples, eficiente e consistente.

O formato XML é robusto, logicamente-verificáveis, e baseado em padrões internacionais. A estrutura hierárquica é adequado para a maioria dos tipos de documentos. A interoperabilidade dos dados definidos no formato XML também é outro dos seus benefícios. O maior poder do XML é a sua extensão.

Você pode inventar seus próprios tags com o seu próprio significado semântico. Por exemplo, você pode criar uma tag para armazenar informação sobre os clientes dados. Seus dados fracção pode conter vários clientes e de informação dos clientes. Se você quiser encontrar todos os clientes com um nome específico, você pode fazer com que a busca de Customer_First_Name tags.

Um documento XML contém vários elementos. A primeira linha é a declaração XML, uma opcional linha indicando qual a versão do XML está em uso. Um elemento geralmente consiste de uma tag e começar a tag de fechamento par. Entre o início e encerramento tag tag par, um elemento pode conter dados, conteúdos, ou outros elementos.

Um elemento pode conter apenas tag de fechamento. O primeiro elemento que o processador XML encontros deve ser constituído por uma etiqueta e começar a tag de fechamento e é chamado o elemento raiz. Todos os outros elementos serão dentro do elemento raiz e são chamados de elementos filhos.

Um elemento filho pode conter elementos de seu próprio filho. Um elemento pode conter atributos. Você pode usar atributos como uma alternativa à utilização de elementos para armazenar dados. Você pode criar um atributo na etiqueta do início de um elemento. Você pode declarar o nome do atributo, seguido por um valor atribuição.

Você pode usar aspas simples ou duplas para definir o valor de um atributo. Os comentários são opcionais. Estas são algumas das regras básicas.

Cenários onde XML é usado

SQL Server 2005 proporciona poderosa armazenagem, consulta, alteração de dados e capacidades mais de dados XML. XML dados existentes podem interoperar com dados relacional SQL e aplicações. Por isso, XML podem ser introduzidas no sistema, como modelagem de dados precisa aumentar, sem perturbar a sua aplicações existentes. Além desses requisitos, XML pode ser útil em muitos cenários. Data Exchange

XML é uma plataforma independente de formato de dados representação. Por isso, é útil para a troca de informações entre vagamente-acoplados, sistemas díspares, como na empresa-a-empresa (B2B) e aplicações workflow situações. De dados tem sido uma grande vantagem da tecnologia XML. Gerenciamento de Documentos

XML é cada vez mais utilizada em aplicações empresariais para a modelagem semi-estruturadas e não estruturadas de dados. Documentos, tais como e-mail, são de natureza semi-estruturadas. Se os documentos são armazenados dentro de um banco de dados no formato XML e, em seguida, será possível desenvolver aplicações poderosas que recuperar documentos baseadas no conteúdo documento, consulta para o conteúdo parcial, e executar documento agregação. Esses cenários estão a tornar-se viável com o aumento dos aplicativos que geram e consomem XML.

Messaging

XML pode ser usado em um ambiente mensagens, para permitir que os pedidos para troca de mensagens com outras aplicações. Em EAI cenários, quando várias aplicações precisam ser integrados, o XML é o formato utilizado para o intercâmbio de dados.

Colaboração Mid-tier

XML pode ser usado para a troca de dados entre o nível médio e os outros estratos, em uma aplicação. Tradicionalmente a apresentação tier dados enviados para o nível médio em um formato binário proprietário. Este restritas ao tipo de tecnologia que poderiam ser utilizados para aplicar a vários níveis. Também externo aplicativos que rodam em sistemas díspares não poderia comunicar com um nível médio componente comum. XML resolvidos estes problemas através da introdução de uma norma comum para representar dados. XML Web Services podem ser usadas como um meio eficaz-tier componente que recebe e envia todos os dados em formato XML.

Modelagem Ad-hoc

XML pode ser usado para ad-hoc a modelagem de dados. Se a estrutura dos dados mudanças ou evolui, as mudanças podem ser incorporados no documento XML. Normas como XPath, XQuery, XSLT e torná-lo muito fácil de tratar os dados e torná-lo em qualquer outro formato adequado.

Como o SQL Server implementa XML

SQL Server 2005 fornece suporte extensivo para processamento de dados XML. XML nativamente valores podem ser armazenados em uma coluna tipo de dados XML, o que pode ser digitado de acordo com uma coleção de esquemas XML, ou untyped esquerda. Você pode indexar a coluna XML. Além disso, a multa de grãos dados manipulação é suportado usando XQuery e XML DML, sendo este último uma extensão de dados modificação. SQL Server XML implementa a passagem à próxima XML através de vários processos.

Primeiro Passo

Tal como o primeiro passo no trabalho com XML, SQL Server alimenta a próxima XML para um analisador XML. O XML Parser interpreta o XML insumos e apresenta-las sob a forma de uma interface conhecida como um Document Object Model (DOM). Os analisadores XML também verificar se o documento XML está bem formado e está gramaticalmente e sintaticamente válido. Os analisadores XML também pode ser utilizado como parsers de validação para validar os dados contra um esquema XML.

Segundo Passo

Quando você fornecer um esquema XML para os documentos que você irá armazenar, você pode especificar o nome do esquema que contém o esquema coleção que você deseja aplicar a essa coluna. Em seguida, SQL Server alimenta o esquema para o XML Parser, o que valida a próxima XML contra o esquema.

Após isto, a XML é validado implicitamente convertidos para o tipo de dados XML. Em seguida, o XQuery métodos disponíveis no SQL Server 2005 como query (), valor (), gânglios () ou modificar () são aplicados sobre os dados convertidos XML.

Terceiro Passo

Os dados podem ser convertidos XML usado para diversos fins. Você pode usar o método OPENXML gânglios converter dados XML em um formato rowset. No entanto, antes de a dados XML é convertido para rowsets, SQL Server constrói um nó tabela internamente. Você pode aplicar métodos como XQuery consulta ou de valor sobre o conteúdo do nó tabela.

Quarto Passo

Para acelerar o processamento dos dados, SQL Server faz uso de um índice primário XML que pode ser definida com a coluna que contém dados XML. O principal índice XML índices todas as tags, valores, e ao longo dos caminhos XML instâncias na coluna e benefícios query performance.

Ao usar o principal índice XML, três tipos de índices secundários XML - o Índice Caminho, a propriedade eo valor Índice Índice - pode ser criado com o XML coluna para acelerar buscas.

Dependendo do cenário e da natureza dos dados em XML, SQL Server utiliza qualquer destes índices secundários.

Quinto Passo

Com estas características que proporcionam apoio XML, o SQL Server 2005 fornece uma poderosa plataforma de desenvolvimento de aplicações para a semi-estruturadas e não estruturadas gestão de dados.

Questões

No seguinte código XML

```
<? xml version = "1,0"?>
```

```
<colors>
```

```
<color ID="1">
```

```
<nome> Vermelhos </ nome>
```

```
</ cor>
```

```
</ cores>
```

Identifique o tipo de componente <ID>.

Elemento raiz

Processamento Instrução

Atributo X

Elemento

Qual das seguintes declarações são verdadeiras sobre os cenários em que o XML pode ser usada?

XML não é adequado para a troca de informações entre vagamente-acoplados, sistemas díspares.

Documentos XML dados armazenados em uma coluna podem ser recuperadas com base no documento de conteúdo utilizando consultas. X

XML é utilizado em aplicações empresariais para a modelagem de dados estruturados.

XML pode ser usado em um ambiente Messaging, para permitir que os diversos pedidos múltiplos para trocar mensagens com outros aplicativos. X

XPath, XQuery e XSLT normas facilitar o processamento e transformação de dados em um formato adequado. X

A apresentação XML tier envia dados para o nível médio em uma leitura, definido pelo usuário, formato.

Você precisa treinar os novos recrutas no seu departamento de pesquisa XML em dados SQL Server. Eles precisam entender como dados XML é executado em SQL Server. Qual é a correta seqüência de passos no SQL Server, que implementa o XML?

2 O XML Parser interpreta o XML insumos e verifica a entrada de sintaxe XML e erros gramaticais.

4 SQL Server aplica qualquer um dos métodos, como a consulta XQuery, valor, gânglios ou modificar o convertidos em dados XML.

1 SQL Server alimenta a próxima XML para um analisador XML.

3 SQL Server converte os dados XML em XML validado tipo.

6 SQL Server melhora o desempenho, por meio Secundária xml índices.

5 SQL Server acelera o processamento de dados, através da utilização de XML Índice Primário.

Consultando Dados XML

SQL Server 2005 oferece várias funcionalidades para consultar dados XML. Você pode consultar dados XML usando a função OPENXML T-SQL. Esta função oferece um rowset vista dos dados em documentos XML, semelhante a uma mesa ou um ponto de vista. Você também pode consultar dados XML, usando XQuery, que é uma consulta língua que é projetado para os dados armazenados em estruturados ou semi-estruturadas formato XML. Ao combinar LMG declarações com XQuery, você pode gerar relatórios com base em XML.

Como gerar Relatórios XML

Uma consulta SELECT normalmente retorna resultados, sob a forma de linhas e colunas, que também é conhecido como um rowset. Você pode opcionalmente recuperar os resultados de uma consulta SELECT no formato XML, em vez de rowset formato. Para fazer isto, você precisará anexar uma cláusula para XML para a SELECT query.

... Para XML (RAW | AUTO | EXPRESSA | PATH)... [argumentos opcionais] <optional arguments>:: = [, TIPO] [, XMLDATA] [, XMLSCHEMA] [, RAIZ ('RootName')] [, ELEMENTOS]

RAW [nome_elemento]

Neste modo, cada linha no resultado é transformado em um elemento XML que tem um identificador genérico <row /> como o elemento tag. Você pode opcionalmente especificar um nome para a fila elemento que serão utilizados pelo XML resultante. Se os elementos directiva é acrescentado à cláusula do XML, cada coluna valor é mapeado para uma subelement <row> do elemento. Você pode solicitar o esquema XML, especificando a XMLDATA para XDR XMLSCHEMA para XSD schema ou esquema. O esquema aparece no início dos dados. No resultado, o esquema namespace referência é repetido em cada elemento de nível superior.

AUTO

Neste modo, os resultados são devolvidos em busca de uma simples, XML aninhadas árvore. Cada tabela na cláusula FROM, das quais pelo menos uma coluna é listada na cláusula SELECT, é representado como um elemento XML. As colunas listadas na cláusula SELECT são mapeados para atributos ou subelementos, se os elementos opcionais opção é especificada. Você pode especificar a forma de tabelas com as colunas identificadas na cláusula SELECT baseada na hierarquia que o XML é gerado. Você pode especificar o esquema XML usando a XMLDATA ou XMLSCHEMA opção. A primeira mesa a partir da esquerda faz parte superior elemento no documento XML resultante. A segunda mesa a partir da esquerda formas subelement um elemento dentro do topo, e assim por diante. Se você especificar um nome já existente coluna então ele será adicionado como um atributo de o elemento.

Explicito

Neste modo, é necessário especificar explicitamente o esperado aninhado no XML como parte da consulta.

Você precisa escrever uma consulta para apresentar os seguintes metadados duas colunas:

A primeira coluna deve fornecer a tag eo número inteiro tipo de elemento da actual, e no nome da coluna deve ser Tag. Você deve fornecer um número ímpar tag para cada elemento que irá ser construída a partir da rowset.

A segunda coluna deve fornecer um número de codificar o elemento pai, e esta coluna nome deve ser pai. Os Tag e os Pais coluna hierarquia prestar informações.

PATH(Caminho)

Neste modo, você poderá executar consultas para XML aninhadas com o tipo directiva para retornar XML tipo instâncias. Você pode executar a maior parte das consultas no modo explicita o caminho modo. Por padrão, o caminho modo gera um elemento <row> wrapper para cada linha no conjunto de resultados. Você pode opcionalmente especificar um elemento nome que está a ser utilizada como elemento nome do invólucro. No PATH modo, a coluna nomes ou apelidos coluna são tratados como expressões XPath. Estas expressões indicam a forma como os valores estão sendo mapeados para XML. Por padrão, o caminho modo gera-centric elemento XML.

Como consultar usando Open XML

OPENXML T-SQL é uma função que proporciona um rowset vista da representação interna de dados XML. Ao usar OPENXML, você pode acessar dados XML como se fosse um rowset relacionais. Os registros no rowset podem ser armazenados em banco de dados tabelas. OPENXML só pode ser utilizado em SELECT e SELECT INTO declarações sempre que rowset fornecedores como uma tabela, um ponto de vista, ou OPENROWSET são utilizados.

OPENXML (IDOC int [em], rowpattern nvarchar [em])
[COM (TableName | SchemaDeclaration)]

Nesta sintaxe:

IDOC é o documento de lidar com a representação interna de um documento XML.

rowpattern XPath é o padrão usado para identificar os nós de lidar com a fonte documento XML que precisam de ser tratados como linhas.

e TableName ou SchemaDeclaration é utilizado com o com a cláusula opcional para especificar o esquema em que o rowsets são devolvidos.

O que é Xquery?

XQuery é uma declarativa, datilografado e funcional da língua concebido com a finalidade de resgatar os dados armazenados em estruturados ou semi-estruturadas formato. Você pode usar XQuery para trabalhar com documentos XML untyped que estão associados com nenhum esquema, documentos que são digitadas associados com esquemas XML, ou uma combinação de ambos. Ao utilizar as extensões para XQuery no SQL Server 2005, e não apenas você pode consultar dados XML, mas você também pode modificá-la. XQuery é um método muito eficiente para consulta enormes pedaços de dados. Ele também oferece a capacidade de filtrar, classificar, a ordem, e repurpose as informações exigidas.

Necessidade de uma XQuery

A linguagem XQuery foi implementado em SQL Server, pelas seguintes razões:

Ao usar o tipo de dados XML, você pode armazenar dados digitados ou untyped XML em banco de dados tabelas.

O SQL SELECT implementação em SQL Server 2005 não pode compreender dados XML. Por isso, uma alternativa língua que possam compreender e trabalhar com dados XML tiveram de ser introduzidas no SQL Server 2005.

declarações DML tais como INSERT, UPDATE, DELETE e trabalhar com todos os dados armazenados em uma coluna. Ao trabalhar com dados XML armazenados em uma coluna, você pode querer manipular ou modificar apenas alguns elementos em vez de nodos ou a totalidade do pedaço de dados. Ao usar os acessórios para a linguagem XQuery no SQL Server 2005, você pode estender LMG declarações de transformar certos elementos em nós ou os dados subjacentes XML.

Chaves para XQuery

O PARA, LET, WHERE, ORDER BY, e retornar cláusulas constituem o núcleo das expressões XQuery e são similares às declarações de SQL SELECT.

PARA A cláusula permite que você defina um declarativa iteração de uma variável associada utilizadas durante uma seqüência de entrada. É equivalente ao SQL SELECT cláusula FROM.

O LET cláusula é usada para uma variável vinculativo para os resultados de uma expressão. No entanto, o LET cláusula não é suportada pelo XQuery implementação em SQL Server 2005. Você pode usar uma expressão em vez inline.

A cláusula WHERE filtra os resultados de uma iteração através da aplicação da expressão especificada

com a cláusula WHERE.

A cláusula ORDER BY lhe permite classificar os valores devolvidos no conjunto de resultados. A cláusula ORDER BY aceita uma triagem expressão que retorna um valor atômico.

O regresso cláusula, que é análoga à cláusula SELECT em SQL, permite-lhe definir o resultado de uma consulta. Você pode especificar qualquer expressão válida XQuery no retorno cláusula. Você também pode construir estruturas em XML, especificando o regresso secção construtores de elementos e atributos.

Como consultar XML usando Xquery

Você pode utilizar várias formas previstas na linguagem XQuery para consultar e modificar dados XML em SQL Server. O XQuery diferentes métodos são: consulta (), valor (), existem (), modificar (), e nós ().

Query()

A consulta () método especifica uma expressão XQuery a ser executada contra um exemplo de dados XML. Este método retorna uma instância de untyped XML.

Exemplo:

O código a seguir mostra um exemplo variável @ myDoc tipo de XML e atribui-lhe um exemplo XML. Em seguida, uma consulta () método é usado para especificar uma XQuery contra o documento. A consulta então obter o <Warranty> elemento do <ProductDescription> elemento.

USE AdventureWorks

DECLARAM @ myDoc XML

SET @ myDoc = '<Root>

<ProductDescription ProductID="1" ProductName="Road Bike">

<Features> <Warranty> 1 ano peças e trabalhistas </ Garantia>

<Maintenance> 3 anos peças e manutenção prorrogada trabalhista está disponível </ Manutenção>

</ Recursos> </ ProductDescription>

</ Raiz> "

SELECT@myDoc.query ('/ root / ProductDescription / Recursos / Garantia ")

O código seguinte exemplo mostra como você pode reescrever a mesma consulta para usar uma expressão FLWOR vez de uma expressão XPath.

USE AdventureWorks

DECLARAM @ myDoc XML

SET @ myDoc = '<Root>

<ProductDescription ProductID="1" ProductName="Road Bike">

<Features> <Warranty> 1 ano peças e trabalhistas </ Garantia>

<Maintenance> 3 anos peças e manutenção prorrogada trabalhista está disponível </ Manutenção>

</ Características>

</ ProductDescription>

</ Raiz> 'SELECT@myDoc.query (' por US \$ d in / root / ProductDescription / Recursos / Garantia regressar \$ d ')

Value()

O valor () realiza um método XQuery contra os dados XML e retorna um tipo escalar SQL. Este método é usado para extrair um valor a partir de uma instância XML armazenados em uma coluna tipo de dados XML, ou variável. Você pode usar um namespace XML, que é uma coleção de nomes, identificada por um Uniform Resource Identifier (URI) referência, que são utilizados em documentos XML como elemento e atributo nomes. Você também pode especificar consultas SELECT, que combinam dados XML ou comparar com os não-XML colunas. A declaração seguinte mostra como o valor () método é usado normalmente.

Value (XQuery, SQLType)

Neste sintaxe, XQuery XQuery é a expressão que recupera os dados de instâncias XML. SQLType é o preferido SQL escalar tipo de ser devolvido.

Exemplo:

O código a seguir mostra como exemplo para reescrever a consulta anterior para usar o valor () método. O valor método recupera o valor do atributo ProductModelName e, depois, converte-o para o tipo de dados nvarchar.

```
WITH XMLNAMESPACES ( 'http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelDescription' como PD,
'http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelWarrAndMain'
como WM)
SELECT CatalogDescription.value ( '/ PD: ProductDescription [1] / @ ProductModelName', 'nvarchar
(50)') como resultado
FROM Production.ProductModel
WHERE CatalogDescription.exist ( '/ PD: ProductDescription / PD: Recursos / gerenciador: Garantia') =
1
```

Exist()

A existência () retorna um valor Booleano método de Verdadeiro se o XQuery expressão em uma consulta retorna um resultado não vazio. Inversamente, a existir () método retorna falso se o resultado conjunto está vazio. A existência () retorna NULL método, quando o tipo de dados XML contra a qual a consulta foi executada contém NULL.

Exemplo:

O código a seguir mostra como exemplo a existir () na cláusula WHERE método é usado para encontrar apenas linhas que contém o <Warranty> elemento no XML. Em seguida, o espaço de nome palavra-chave é utilizada para definir o namespace dois prefixos.

USE AdventureWorks

```
SELECT CatalogDescription.query ( 'namespace declarar PD =
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelDescription";
Dados (/ FD: ProductDescription [1] / @ ProductModelName) ') como resultado
FROM,Production.ProductModel
WHERE CatalogDescription.exist ( 'namespace declarar PD =
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelDescription";
Declaro namespace gerenciador = "http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelWarrAndMain";
/ PD: ProductDescription / PD: Recursos / gerenciador: Garantia ') = 1
```

Modify()

A modificação () método modifica o conteúdo de um documento XML. Este método tem um XML Dados Manipulação Language (DML) declaração de inserir, atualizar, apagar ou nódulos de dados XML. A modificação () o método de dados XML tipo só podem ser utilizadas na cláusula SET do UPDATE uma declaração. O que se segue é o formato para a modificar () método.

modify (XML_DML)

XML_DML é uma string no XML Dados Manipulação Language (DML). O documento XML é atualizado de acordo com esta expressão.

Exemplo:

O código a seguir mostra como exemplo uma consulta suprime o "MachineHours" atributo XML a partir da amostra.

```
USE AdventureWorks
DECLARAM @ myDoc XML
SET @ myDoc = '<? Instruções para TheWC.exe =?>
<Root>
<! - Instruções para a 1ª obra center ->
<Localização LocationID = "10"
LaborHours = "1,1"
MachineHours = ".2"> Algumas texto 1 <step> Manufacturing passo 1 a este trabalho center </
passo>
<step> Manufacturing passo 2 a este trabalho center </ passo>
</ Location>
</ Raiz> '
SELECT @ myDoc
SET@myDoc.modify ( 'apagar / raiz / Local / @ MachineHours')
SELECT @ myDoc
```

Nodes()

Os nodes () método é usado para modificar um exemplo dados XML em dados relacionais. Permite-lhe identificar os nós que serão mapeados em uma nova linha.

nodes(XQuery), as tabela (coluna)

XQuery é uma string literal e uma expressão XQuery. Quadro (coluna) é o nome da tabela eo nome da coluna para o resultante rowset. Você pode usar dados XML tipo de métodos, tais como a consulta (), valor (), existem (), e nósulos (), para o resultado de uma gânglios () método.

Você não pode usar a modificar () para modificar o método XML instância. O contexto nó na rowset não pode ser materializado, isto é, você não pode usá-lo em um SELECT. No entanto, você pode usá-lo na IS NULL e COUNT (*) declarações.

Exemplo:

O código seguinte exemplo mostra como você pode ter um documento XML com uma <Root> elemento de nível superior e <row> três elementos. A pesquisa utiliza os nodos () método para definir contexto nodos distintos, um para cada elemento <row>. Os gânglios () retorna um método rowset com três linhas. Cada linha tem uma lógica cópia do original XML, com cada nó identificação de um contexto diferente <row> elemento no documento original. A consulta então retorna o nó contexto de cada linha.

```
USE AdventureWorks
DECLARAM @ x XML
SET @ x = '<Root>
<row ID="1"> <nome> Larry </ nome> <oflw> algum texto </ oflw> </ row>
<row ID="2"> <nome> Moe </ nome> </ row>
<row ID="3" />
</ Raiz> '
SELECT T.c.query ('.') como resultado
FROM@x.nodes ( '/ root / linha') T (c)
```

Como consultar Dados relacionais combinados com dados XML

Em SQL Server 2005, você pode consultar os dados relacional combinadas com dados XML por escrito consultas que combinam LMG e XQuery declarações. Este tipo de consulta é conhecido como um XML-LMG consulta. SQL Server processos tais consultas na seguinte sequência de passos.

Em primeiro lugar, a consulta é enviada para o SQL Parser, o que valida a norma de partes de uma consulta SQL. O XQuery expressão na consulta é passado para o XQuery Parser que avalia o XQuery

em duas fases, a fase estáticas e dinâmicas.

Na fase estática, o XQuery Parser reúne todas as informações necessárias para a análise estática da expressão. Se os dados XML que está sendo utilizado no XQuery expressão é digitado em XML, e depois o seu esquema correspondente é copiado a partir do esquema de recebimento.

Como parte da inicialização estática contexto, a fronteira espaços brancos são despojado. O namespace prefix e consolidados também são inicializadas e verificados. Se um esquema está disponível e, em seguida, os dados XML é validado contra o esquema recuperadas.

Após a estática contexto é inicializado, a consulta expressão é compilados e analisados. Nesta análise estática fase, a consulta é analisada, a função eo tipo nomes na expressão são resolvidas, e tipagem estática da consulta é realizada para garantir a segurança de tipo.

A dinâmica fase começa após a fase estática. Na fase dinâmica, as informações necessárias para a execução da expressão é recuperada. Isto é tipicamente os dados XML que o XQuery expressão tem de trabalhar.

Questões

Como administrador do banco de dados em Adventure Works, que você precisa para obter bons resultados nos formatos XML para algumas consultas. Queremos que cada linha de dados para ser baptizado com o nome da tabela a partir da qual a linha originou. Qual dos seguintes modos de XML Query você vai usar para obter os resultados desejados?

RAW [('ElementName')]
EXPRESSA
AUTO X
PATH

O chefe de vendas de Aventura Obras quer que você se prepare um relatório sobre os detalhes do volume de negócios do ano anterior, no formato XML. Ele quer que você possa usar o FLWOR declarações de XQuery para este fim. Você precisa de definir os resultados da consulta. Que uma das seguintes expressões FLWOR das declarações que você irá usar para definir o resultado de sua consulta?

TO
RETURN X
LET
WHERE

Em um relatório de avaliação que você precisa para gerar, você precisa definir o resultado de estar em escalar tipo SQL. Qual dos seguintes XQuery MÉTODOS você vai usar para obter os resultados em escalar SQL tipo?

VALue X
QUERY
EXIST
nodes

Resumo de índices Full-Text

Os índices Full-Text são mais ricos em índices caracteres de dados não estruturados. Eles têm poderosas capacidades de pesquisa, comparada com a T-SQL índices em dados não estruturados. Você precisa analisar a forma como a indexação do texto completo e resgatar processo de SQL Server fornece aos usuários a pesquisar capacidades flexíveis. Você também precisa entender como SQL Server implementa os índices de texto completo para usar este recurso em seu banco de dados. O que são índices Full-Text?

Um índice de texto completo é um tipo especial de token de base funcional índice que é construída e mantida pela Microsoft de texto completo para o SQL Server Engine (MSFTESQL) serviço. A criação de um índice de texto completo em uma tabela no Microsoft SQL Server 2005 é um processo em duas etapas. Primeiro você precisa criar um catálogo de texto completo para armazenar índices de texto completo. Em seguida, você precisa criar os índices de texto completo. Os utilizadores podem efetuar pesquisas de texto completo sobre estes quadros de texto completo indexado.

Uma pesquisa de texto completo permite rápido, flexível e poderosa indexação por palavra-chave do

texto consulta à base de dados armazenados no banco de dados SQL Server. Este texto também poderia ser dado a partir de arquivos externos como o Microsoft Office Word, Microsoft Office planilhas Excel, documentos PDF, HTML ou páginas da Web armazenadas no banco de dados SQL Server como Objetos binários grandes (BLOBs).

Vantagens dos Índices full-Text

O desempenho vantagem de usar a ferramenta de pesquisa de texto completo pode ser mais bem utilizada quando examinando contra uma grande quantidade de dados não estruturados texto. Um LIKE consulta, por exemplo, alumínio%%, contra milhões de linhas de dados de texto, leva tempo para retornar resultados, porque tem de pesquisar para cada linha um exemplo de 'alumínio'. Em contrapartida, uma pesquisa de texto completo de 'alumínio' contra milhões de linhas do mesmo texto, os dados podem ter apenas alguns segundos ou menos, dependendo do número de linhas que são exibidas.

Ao contrário do predicado LIKE, que só funciona em caráter padrões, você pode usar de texto completo de consulta para executar pesquisas lingüísticas contra os dados. O texto completo de consultas podem incluir palavras e frases, ou múltiplas formas de uma palavra ou frase. No Microsoft SQL Server 2005, pesquisa de texto completo empresa oferece funcionalidade de pesquisa.

Você pode usar a pesquisa de texto completo de aplicações baseadas na web, sistemas de gerenciamento de documentos, aplicações personalizadas e que precisam prestar texto capacidades de pesquisa, para os dados armazenados em um banco de dados SQL Server.

Pesquisa de texto completo é altamente extensível e do texto completo do motor suporta 16 idiomas. É capaz de suportar mais linguagens de indexação e pesquisa, adicionando a palavra mais demolidores de stemmers ou de terceiros fornecedores. Também pode filtrar os formatos de documentos adicionais, usando filtros de terceiros. Ela também tem a característica que é novo léxico para SQL Server 2005. O tesouro palavra-chave para pesquisar o sinônimo formas de uma palavra específica.

Criação de indexação do texto completo a capacidade de uma tabela no SQL Server é um processo em duas etapas. Primeiro você precisa criar um catálogo de texto completo para armazenar os índices de texto completo. Em seguida, você precisa criar os índices de texto completo.

A criação de um catálogo Full-Text

Para criar um catálogo de texto completo para um banco de dados, você precisará usar o CREATE FULLTEXT CATALOG declaração. Um catálogo de texto completo pode ter vários índices de texto completo, mas um índice full-text apenas pode ser parte de um catálogo de texto completo. Bancos de dados pode ter mais de um catálogo de texto completo. No entanto, não é possível criar catálogos de texto completo, no master, modelo, ou tempdb bases de dados.

```
CREATE FULLTEXT catálogo catalog_name  
[NO CAMINHO "rootpath"]  
[COM ACCENT_SENSITIVITY = (ON | OFF)]  
[Como padrão]
```

Neste sintaxe, catalog_name é o nome do novo catálogo que deve ser único entre todos os nomes no catálogo da atual base de dados. O comprimento do catálogo nome não deve exceder 120 caracteres. EM PATH'rootpath 'é o diretório raiz para o catálogo. Se rootpath não é especificado, o novo catálogo pode ser localizado no diretório especificado na configuração padrão. COM ACCENT_SENSITIVITY = (ON | OFF) especifica que o catálogo é sensível sotaque ou sotaque insensível para indexação de texto completo. Quando esta propriedade está mudado, o índice deveria ser reconstruída. Como padrão especifica que o catálogo é o procedimento padrão do catálogo. Quando full-text índices são criados sem um catálogo de texto completo explicitamente especificado, é utilizado o padrão catálogo.

Criar um índice de full-text

Para criar um índice de texto completo em uma ou mais colunas de uma tabela em um banco de dados, você precisará usar o CREATE FULLTEXT INDEX declaração. Apenas um índice de texto completo é permitida por tabela. Por padrão, um banco de dados está habilitado para pesquisa de texto completo, quando a base de dados é criada.

```
CREATE FULLTEXT INDEX em table_name  
[(column_name)]  
KEY INDEX index_name  
[ON fulltext_catalog_name]
```

Neste sintaxe, table_name é o nome da tabela que contém a coluna ou colunas incluídas no índice de texto completo. O column_name é o nome da coluna ou colunas incluídas no índice de texto completo. Apenas colunas do tipo char, VARCHAR, nchar, nvarchar, texto, ntext, imagem, xml, e varbinary podem ser indexados para pesquisa de texto completo. KEY INDEX index_name é o nome da chave única em índices table_name. Dever-se-á um única-chave, e não anulável coluna. ONfulltext_catalog_name é o catálogo de texto completo utilizado para o índice de texto completo. O catálogo deve estar presente no banco de dados.

Como O SQL Server implementa índices full-text

A indexação do texto completo processo no SQL Server 2005 conceitual consiste de duas fases: recolha de dados, e a construção da indexação do texto completo estruturas. A Microsoft Motor de texto completo para SQL Server (MSFTESQL) poderes a pesquisa de texto completo em SQL Server 2005.

Este mecanismo é baseado em Microsoft Search (MSSearch) tecnologia e está integrada no SQL Server 2005 Database Engine. O índice de texto completo processo ser iniciado depois de ter fornecido as declarações T-SQL DDL para criar um catálogo de texto completo e um índice de texto completo.

Com base nestas declarações, uma população de texto completo ou rastreamento é iniciado. O Database Engine empurra grandes quantidades de dados para a memória e instrui o motor de texto completo para começar a indexação. Com base nas instruções, os valores em uma coluna ou colunas de uma tabela são de texto completo indexado.

Usando um protocolo manipulador componente, o motor de texto completo puxa os dados da memória através da sua indexação de gasodutos e de processos de dados ainda mais, resultando em um índice full-text.

Quando você Índice dados que estão no exterior formatos de arquivo, um componente especial que implementa a interface IFilter é usado para extrair o texto, baseado no formato de arquivo especificado que o binário documento.

Como parte do tratamento, os dados recolhidos texto é passada através de uma palavra disjuntor para separar o texto em palavras-chave individuais ou 'tokens'. Você pode especificar o idioma a ser utilizado para tokenization, com um nível per-coluna. Você também pode identificar a linguagem dentro do BLOB ou filtrar os dados XML por componente.

SQL Server remove o ruído de palavras para normalizar as fichas, antes de serem armazenados no índice de texto completo ou um fragmento índice. Em seguida, um mestre final fundir processo é desencadeado após o índice de texto completo foi povoada. Quando uma população já foi concluída, durante o processo de merge, o índice fragmentos são fundidos em conjunto.

Por isso, só o capitão índice precisa ser consultada em vez de uma série de fragmentos índice. Este processo mestre junção completa a indexação do texto completo processo.

Questões

Você ativou indexação do texto completo em uma tabela com freqüência pesquisadas pelos usuários. Você tem circulado uma folha para os usuários que especificam as melhores práticas para fornecer pesquisa condições. Alguns usuários pedir-lhe os benefícios da utilização da pesquisa de texto completo sobre utilizando o predicado LIKE. Quais seriam suas respostas?

[O texto completo de pesquisas trabalhos sobre os padrões de caracteres.](#)

[O texto completo de pesquisa rápido e flexível permite a indexação por palavras-chave para a](#)

consulta com base de dados de texto. X

O texto completo de pesquisas levam mais tempo para retornar dados em comparação com o predicado LIKE.

Pesquisa de texto completo é altamente extensível. X

O texto completo de pesquisas fornecer resultados rápidos quando examinando uma grande quantidade de dados não estruturados. X

O texto completo de motor suporta apenas um idioma.

Qual é a boa ordem dos processos envolvidos na execução de texto completo índices?

2-A população de texto completo ou rastreamento é iniciado. O Database Engine empurra grandes quantidades de dados para a memória e instrui o motor de texto completo para começar a indexação.

4 Quando você Índice dados externos que são de formatos de arquivo, um componente especial que implementa a interface IFilter é usado para extrair o texto, baseado no formato de arquivo especificado que os dados.

1 T-SQL são declarações escritas para criar um catálogo de texto completo e um índice de texto completo.

3 Usando um protocolo manipulador componente, o motor de texto completo puxa os dados da memória através da sua indexação de gasodutos e de processos de dados.

6 O índice fragmentos são fundidos em conjunto em um índice mestre.

5 O texto dados é passada através de uma palavra disjuntor para separar o texto em fichas individuais ou palavras-chave. Ruído palavras foram removidas para normalizar as fichas, antes de serem armazenados no índice de texto completo ou um fragmento índice.

Consultando Índices Full-Text

O SQL Server suporta muitas declarações e as funções que tratam exclusivamente com índices de texto completo. Você pode usar o T-SQL predicados, contém e FREETEXT, para efetuar pesquisas de texto completo que correspondem a uma palavra, um prefixo de uma palavra, ou um sinônimo de uma palavra. Você também pode usar o rowset-valorizada funções, CONTAINSTABLE e FREETEXTTABLE, para efetuar pesquisas de texto completo que correspondem à única palavras e frases, para descobrir a proximidade das palavras, ou executar jogos ponderada. Além disso, você pode combinar as contém e FREETEXT predicados com qualquer um dos outros predicados T-SQL, como gosta e entre.

Resumo da Busca Full-Text

SQL Server 2005 fornece vários componentes T-SQL para consultas de texto completo. Você pode usar o contém e FREETEXT predicados como uma condição pesquisa, na cláusula WHERE de um SELECT. Você também pode usar o CONTAINSTABLE e FREETEXTTABLE rowset-valorizada funções na cláusula FROM de uma SELECT.

Predicado CONTAINS

Você pode usar o predicado CONTAINS para pesquisar o banco de dados para uma frase específica. Embora você possa executar uma consulta semelhante ao usar o LIKE predicado, o predicado CONTAINS texto fornece mais capacidades em cenários de consulta específica a pesquisa por frase. No entanto, ao contrário de usar o predicado LIKE, CONTAINS pesquisa é sempre um caso insensível.

Predicado FREETEXT

Você pode usar o predicado FREETEXT de pesquisa para colunas contendo personagem à base de dados tipos de valores que correspondem ao significado e não a formulação exata das palavras da pesquisa condição. Quando FREETEXT é utilizado, o motor de busca de texto completo atribui um peso cada termo e, em seguida, verificar os jogos.

A linguagem palavra reservada

Você pode utilizar a linguagem palavra reservada para especificar o idioma no qual você deseja executar a consulta. Uma única coluna pode armazenar dados em vários idiomas. A LÍNGUA parâmetro permite especificar o idioma no qual você precisa de pesquisa para ser feito. Isso resulta no aumento da probabilidade de um bom jogo.

Consultas multi-coluna

Em SQL Server 2005, você pode usar o predicado de consulta CONTÉM múltiplas colunas em uma única tabela, indicando uma lista de colunas para pesquisa. Se você usar o predicado contém ou FREETEXT a pesquisa em várias colunas, então você precisa se assegurar que a língua de todas as colunas é o mesmo. Você também precisa se assegurar que todas as colunas pertencem à mesma mesa.

Sotaque Sensibilidade

Você pode controlar o sotaque sensibilidade em catálogo nível. Quando você cria o catálogo, é necessário especificar se o acento catálogo é sensível ou não. Se você mudar o sotaque sensibilidade depois de criar o catálogo, então você precisa para recriar os índices. Por exemplo, a fixação ACCENT_SENSITIVITY controla se café e café são tratadas como as mesmas palavras com o motor de texto completo.

Como usar o predicado CONTAINS

Você pode usar o predicado CONTÉM para localizar linhas que tenham especificado no texto uma dada coluna. A coluna que é utilizado na CONTÉM predicado deve fazer parte de um ativo de texto completo índice. CONTÉM O predicado retorna um valor verdadeiro ou falso e tão só pode ser usado dentro de uma cláusula WHERE ou HAVING de uma SELECT. Você deve saber como usar o predicado CONTÉM e a sua sintaxe.

Uso do predicado CONTAINS

Você pode usar o predicado CONTAINS para pesquisar:

Uma palavra ou frase.

O prefixo de uma palavra ou frase.

Uma palavra perto de outra palavra.

Uma palavra gerada a partir de outra palavra. Por exemplo, a palavra 'drive' pode ser gerada a partir de palavras como "drives", "levou", "condução", e assim por diante.

Uma palavra que é um sinônimo de outra palavra usando o léxico. Por exemplo, a palavra "moto" pode ter como sinônimo 'alumínio' ou 'aço'.

Sintaxe do predicado CONTAINS

CONTAINS

```
( { column_name | (column_list) | * }  
, '< contains_search_condition >'  
[ , LANGUAGE language_term ]  
)  
<contains_search_condition > ::=  
{ < simple_term >  
| < prefix_term >  
| < generation_term >  
| < proximity_term >  
| < weighted_term >  
}  
| { ( < contains_search_condition > )  
[ { < AND > | < AND NOT > | < OR > } ]  
<contains_search_condition > [ ...n ]  
}
```

Argumentos do predicado CONTAINS

column_name ou column_list: column_name ou column_list é o nome da coluna ou colunas incluídas no índice de texto completo. Colunas de tipo char, VARCHAR, nchar, nvarchar, texto, ntext, imagem, xml, e varbinary (Max) são válidas para colunas de texto completo pesquisando.

LANGUAGE language_term: LANGUAGE language_term especifica o idioma no qual você emitir a consulta. Você pode especificar este parâmetro opcional como uma string, inteiro, ou valor

hexadecimal.

Pesquisa condição: A pesquisa condição especifica o texto de pesquisa para a coluna na lista e as condições que devem ser satisfeitos para um jogo. A pesquisa condição pode conter elementos mais complexos, tais como uma simples expressão, prefixo prazo, a proximidade prazo, ponderado prazo, e de geração prazo.

Simple_term

O **simple_term** especifica uma correspondência exata de uma palavra ou uma frase. Você precisará anexar a frase entre aspas duplas. As palavras em uma frase que são especificados em uma **<contains_search_condition>** devem aparecer na mesma ordem que aparecem na base de dados. A pesquisa não é sensível a maiúsculas. O índice de texto completo não consiste de ruído seja, como uma, e, e as. Portanto, se você usar uma palavra ruído em uma única palavra pesquisa, o SQL Server retorna um erro, indicando que a consulta contém apenas o ruído palavras. O SQL Server também ignora pontuação na condição de pesquisa.

Prefix_term

O **prefix_term** especifica um jogo de palavras ou frases começando com o texto especificado. Você precisará anexar um prefixo termo em aspas duplas e adicione um asterisco antes das aspas que termina, de modo que SQL Server pesquisas de todo o texto que começa com a simples expressão especificados antes do asterisco. A cláusula deve ser especificada na seguinte maneira.

CONTAINS (coluna ' ', texto *")

O asterisco jogos zero, um, ou mais caracteres a raiz da palavra na frase. Se você não o texto e anexar o asterisco em aspas, a pesquisa de texto completo considere o asterisco como um personagem.

Proximty_term

O **proximity_term** especifica um jogo de palavras ou frases que devem estar próximos de outro conjunto de palavras. O **<proximity_term>** é similar ao operador E, no sentido de que tanto necessitam mais de uma palavra ou frase que existem na coluna a ser pesquisada.

O **NEAR | ~** operador indica que a palavra ou frase no lado esquerdo do operador **NEAR** ou **~** deve ser aproximadamente perto da palavra ou frase sobre o lado direito da **NEAR** ou **~** operador. Você pode cadeia proximidade múltiplos termos. Quando você cadeia proximidade muitos termos, é necessário colocar todos os termos perto de si.

Weighted_term

O **weighted_term** especifica que a correspondência linhas retornadas pela consulta correspondem a uma lista de palavras e frases, opcionalmente cada um determinado peso. Pode ser especificados no seguinte formato.

ISABOUT (palavra-chave WEIGHT(weight_value))

A palavra-chave **ISABOUT** especifica o **<weighted_term>** palavra-chave. O peso (**weight_value**) palavra-chave especifica um valor peso, que é um número entre 0,0 e 1,0. Cada componente em **<weighted_term>** pode incluir uma **weight_value**. O **weight_value** mostra como as diversas partes de uma consulta afetar a classificação valor atribuído a cada linha correspondendo à consulta. O peso palavra-chave não afeta os resultados das consultas **CONTÉM**, mas impactos no rank **CONTAINSTABLE** buscas.

Generation_term

O **generation_term** permite especificar um jogo de palavras quando os termos simples que sejam incluídos são variantes do original da palavra a ser pesquisada. Pode ser especificados no seguinte formato.

FORMSOF ((INFLECTIONAL | Thesaurus), <simple_term>)

A palavra-chave **INFLECTIONAL** pesquisas para todos os diferentes tempos de um verbo ou ambas as formas singular e plural de um substantivo. O tesauro palavra-chave para pesquisar o sinônimo

formas de uma palavra específica. Você pode encontrar uma lista de arquivos localizados no tesauro SQL_Server_install_path \ Microsoft SQL Server \ MSSQL.1 \ MSSQL \ FTDATA \. Você precisa personalizar esses arquivos para expandir ou substituir um termo ou termos de incluir uma lista de sinônimos.

Como usar o predicado FREETEXT

Você pode usar o predicado FREETEXT à pesquisa colunas contendo personagem baseado em dados tipos de valores que correspondem ao significado, mas não necessariamente a formulação exata das palavras da pesquisa condição.

Quando o predicado FREETEXT é utilizado, o motor de busca de texto completo internamente realiza as seguintes ações sobre o freetext_string, atribui um peso cada termo e, em seguida, verificar os jogos.

Separa as strings em palavras individuais baseada na palavra fronteiras usando palavras-quebra.

Gera inflectional as formas de expressão através da utilização decorrentes.

Identifica uma lista de expansões ou substituições para os termos baseados em jogos no léxico.

Um FREETEXT predicado retorna uma verdadeira ou falsa e, por conseguinte, é especificado na cláusula WHERE ou HAVING de uma SELECT.

Em freetext_string, a palavra e é considerado um ruído palavra e serão descartadas. Você não pode usar o peso, FORMSOF, curingas, NEAR, sintaxe e outros. Se freetext_string seja fechada em aspas duplas, uma correspondência de frase é realizada. Verificação e tesauro não são realizadas.

```
FREETEXT ((column_name | (column_list) | *)  
, 'Freetext_string')
```

Neste sintaxe, o column_name ou column_list é o nome da coluna ou colunas incluídas no índice de texto completo. Colunas de tipo char, VARCHAR, nchar, nvarchar, texto, ntext, imagem, xml, e varbinary (Max) são válidas para colunas de texto completo pesquisando. * A especifica que todas as colunas que tenham sido registradas para a busca de texto completo deve ser usado para pesquisar o dado freetext_string. E é freetext_string texto para pesquisar na column_name. Qualquer texto, incluindo palavras, frases ou sentenças, pode ser inscrita. Jogos são gerados se a qualquer termo ou quaisquer formas de expressão são encontrados no índice de texto completo.

Exemplos :

Para utilizar o predicado FREETEXT que você precisa para criar um índice de texto completo sobre a mesa. O exemplo a seguir cria um índice full-text sobre a tabela Production.ProductDescription.

```
USE AdventureWorks  
GO  
CREATE UNIQUE INDEX ui_ukProdDescID  
ON Production.ProductDescription (ProductDescriptionID);  
CREATE FULLTEXT catálogo ftProdDesc como padrão;  
CREATE FULLTEXT INDEX ON Production.ProductDescription (Descrição)-CHAVE INDEX  
ui_ukProdDescID;  
GO
```

O exemplo a seguir faz uso do predicado FREETEXT para retornar todas as linhas que têm a palavra bicicleta e de segurança na coluna Descrição.

```
USE AdventureWorks  
SELECT Esquerda (Descrição, 50)
```

DESDE Production.ProductDescription
ONDE FREETEXT (Descrição, 'bike segurança ")

O seguinte é o resultado parcial da consulta.

(N º coluna nome)

Esta bicicleta proporciona um alto nível de desempenho em
Para a trilha verdadeira toxicodependentes. Uma moto extremamente duráveis
Top-of-the-line concorrência Mountain Bike. Realizar
Entrada nível adulto bicicleta; oferece uma viagem confortável
Valor a preços de moto com muitas características dos nossos top-of

Como usar funções Full-Text

O SQL Server fornece dois rowset à base de funções, a função CONTAINSTABLE e os FREETEXTTABLE função, além do CONTÉM e FREETEXT predicados, que pode ser utilizado com índices de texto completo. Ambas as funções retornam uma tabela de zero ou mais linhas. Portanto, você pode usar estas funções apenas na cláusula FROM de uma SELECT.

CONTAINSTABLE

O CONTAINSTABLE função retorna uma tabela de zero ou mais linhas para colunas que contêm os dados com base em tipos de caracteres. A função retorna CONTAINSTABLE precisas ou imprecisas jogos de palavras e frases simples, baseado em resultados proximidades de outros termos especificados, ponderada e jogos. O seguinte é a sintaxe parcial da CONTAINSTABLE função.

CONTAINSTABLE (tabela, (column_name | (column_list)), '<contains_search_condition>'
[, LÍNGUA language_term]
[, top_n_by_rank])

Neste sintaxe, o quadro é o nome da tabela que foi marcado para a consulta de texto completo. column_name ou column_list é o nome da coluna ou colunas incluídas no índice de texto completo. LÍNGUA language_term é a língua em que a questão consulta. top_n_by_rank especifica que só a melhor classificação n jogos, em ordem decrescente, são devolvidos. <contains_search_condition> especifica o texto para pesquisar na column_name e as condições para um jogo.

FREETEXTTABLE

O FREETEXTTABLE função retorna uma tabela de zero ou mais linhas para colunas que contêm os dados com base em tipos de caracteres. A função retorna FREETEXTTABLE valores que correspondem ao significado, mas não a formulação exata, do texto especificado na freetext_string. Você pode usar a função FREETEXTTABLE só na cláusula FROM de uma SELECT. O seguinte é a sintaxe parcial da FREETEXTTABLE função:

FREETEXTTABLE (tabela, (column_name | (column_list))
, 'Freetext_string'
[, LÍNGUA language_term]
[, Top_n_by_rank])

Neste sintaxe, <freetext_string> especifica o texto para pesquisar nas colunas e aceita qualquer texto, incluindo palavras, frases ou sentenças.

Identificando as diferenças entre funções e predicados Full-Text

Classificar os recursos o mais rapidamente possível para os seus associados categorias clicando no balde apropriado. Você também pode utilizar os atalhos de teclado, premindo 1 balde para a esquerda, direita e 2 para o balde.

Como combinar busca Full-Text e Predicados Transact-SQL

Predicados T-SQL

Você pode combinar as contém e FREETEXT predicados com qualquer um dos outros predicados Transact-SQL, como gosta e entre.

Exemplo:

Contém predicado com T-SQL predicados

O código a seguir exemplo demonstra o uso do predicado CONTÉM com o padrão T-SQL predicados. Esta consulta irá retornar todos os produtos pertencentes ao "Mountain Bikes" categoria, cujo ProductSubCategoryID equivale a uma, e que contenham quer as palavras "Black" ou "Prata" em seus nomes.

USE AdventureWorks

```
SELECT ProductID, p. nome, ProductSubCategoryID
FROM Production.Product p
WHERE ProductSubCategoryID = 1
AND CONTAINS ([nome], 'preto e prata ")
```

O resultado conjunto da consulta anterior consiste de ProductID, Name e ProductSubCategoryID de todos os produtos pertencentes ao "Mountain Bikes" categoria, cujo ProductSubCategoryID equivale a uma, e que contém as palavras "Black" ou "Prata" em seus nomes. Em resultado deste conjunto, 32 linhas são afetadas.

Você pode reescrever a consulta anterior como uma junção entre a Production.Product mesa e Production.ProductSubCategory a tabela da seguinte maneira.

USE AdventureWorks

```
SELECT ProductID, P. Nome, p.ProductSubCategoryID
FROM Production.Product p
INNER JOIN Production.ProductSubCategory subcat
ON p.ProductSubCategoryID = subcat.ProductSubCategoryID
ONDE subcat.Name = 'Mountain Bikes "
E contém (p. [Nome], 'preto e prata ")
```

Esta consulta retorna o mesmo resultado que é definido como a anterior consulta.

FREETEXT predicado com T-SQL predicados

O código a seguir exemplo demonstra o uso do predicado FREETEXT com o padrão T-SQL predicados. Esta consulta irá retornar todos os produtos pertencentes ao Production.Product tabela, onde o nome tem as palavras, ou o significado das palavras, "montanha", ou "Raça".

USE AdventureWorks

```
GO
SELECT ProductID, Nome, produto, nome do Color, ListPrice
FROM Production.Product
ONDE FREETEXT (Nome, "montanha ou raça")
GO
```

O resultado conjunto da consulta será composto pelas ProductID, Nome, ProductNumber e cores de todos os produtos pertencentes ao Production.Product tabela, onde o nome tem as palavras, ou o significado das palavras, "montanha", ou "Race ". Em resultado deste conjunto, 101 linhas são afetadas.

Em subconsulta

Você pode usar o contém e FREETEXT predicados em uma subconsulta.

Exemplo:

O seguinte exemplo usa o código CONTÉM predicado de uma subconsulta, para retornar todas as ProductIDs que contém a palavra 'negro' ou 'Prata' no nome do produto. Esta lista de ProductIDs é enviada para o exterior consulta para limitar o número de linhas para as quais o inventário é retornado. Aqui, você assumir que um texto livre-quadro sobre Production.Product já foi criada.

```
USE AdventureWorks
SELECT p.ProductID, LocationID, Quantidade
DESDE Production.ProductInventory pi
INNER JOIN Production.Product p = pi.ProductID ON p.ProductID
INNER JOIN Production.ProductSubCategory subcat
ON p.ProductSubCategoryID = subcat.ProductSubCategoryID
ONDE subcat.Name = 'Mountain Bikes "E no pi.ProductID
(SELECT ProductID de Production.Product
WHERE CONTAINS (p. [Nome], 'preto e prata'))
```

O resultado conjunto da consulta anterior será composto pelas ProductID, LocationID e quantidade de todos os produtos que tenham a palavra "negro" ou "Prata", em nome do produto. Em resultado deste conjunto, 64 linhas são afetadas.

Questões

Você quer fazer uma pesquisa de produto AdventureWorks nomes na base de dados com as palavras 'mountain bike ', 'tour bike ', e 'estrada bicicleta ', e gostaria de aplicar diferentes níveis de pesos para cada palavra. Palavra chave que você irá precisar juntamente com o termo ponderada a sua consulta?

Palavra chave NEAR
Thesaurus palavra-chave
ISABOUT palavra-chave X
INFLECTIONAL palavra-chave

Você está procurando por colunas que contêm palavras que significa "bicicleta". Você tem decidido a utilizar o predicado FREETEXT para fazer isso. Que uma das seguintes técnicas se você usar junto com o predicado FREETEXT para realizar uma correspondência de frase?

Você irá usar a palavra-chave NEAR e FORMSOF.
Você irá usar aspas duplas marca na string de pesquisa. X
E você irá usar o operador.
Você irá usar a palavra-chave ou PESO curingas.

Você está procurando por colunas que contêm palavras relacionadas com a "segurança dos veículos". Você foi convidado a executar uma consulta, que também encontra expressão que significa "segurança dos veículos". Que uma das seguintes palavras-chave que você irá precisar na cláusula FROM quando você executar esta consulta?

CONTAINS
FREETEXTTABLE X
FREETEXT
FORMSOF

-----*****-----

Usando objetos de programação para recuperar dados no SQL Server 2005

Você pode usar os diversos objetos fornecidos pela programação SQL Server 2005 para recuperar os dados. Para expor somente as informações relevantes aos usuários, você pode encapsular expressões e consultas usando definido pelo usuário, funções e SQL Server pontos de vista. Ao utilizar esses definido pelo usuário, funções, você precisa estar ciente das limitações e desempenho considerações para SQL Server. Você pode usar o sistema de vários procedimentos armazenados desde a SQL Server para implementar e gerenciar distribuídos buscas. Estas consultas podem ser executados contra destinos dados heterogêneos como bases de dados, planilhas, ou mesmo outros servidores. Para além de procedimentos armazenados, você pode criar aciona automaticamente executado quando os utilizadores que tentam inserir, atualizar ou apagar os dados em uma tabela ou vista.

Encapsulando expressões usando funções definida pelo usuário

Ao usar Transact-SQL (T-SQL), você pode criar definido pelo usuário, funções no SQL Server 2005 para implementar as funcionalidades que não são fornecidos pela construído em funções. Definido pelo

usuário funções podem ser criadas uma vez e depois reutilizados para diversos fins. Você pode usar funções definidas pelo usuário, como parametrizados pontos de vista, para melhorar a funcionalidade de uma vista indexada, a coluna manipular um valor em uma tabela, para definir um cheque constrangimento em uma coluna, ou a substituição de um procedimento armazenado. Você precisa saber como criar, modificar e implementar vários tipos de utilizadores, as funções definidas. Além disso, é necessário considerar as restrições e desempenho questões quando da execução definido pelo usuário, funções.

O que são funções definidas pelo usuário?

Definido pelo usuário funções no Microsoft SQL Server 2005 são rotinas que aceitam parâmetros, executar uma ação, como um cálculo complexo, e retornar o resultado de que a ação como um valor. O valor de retorno pode ser um único valor escalar ou um conjunto de resultados. Todas as funções definidas pelo usuário, incluir uma parte dois, estrutura de um cabeçalho e um corpo. SQL Server 2005 oferece três tipos de utilizadores, as funções definidas, a saber, escalar funções, mesa-valorizada funções, e funções built in. funções definidas pelo usuário no SQL Server 2005 fornecem diversos benefícios.

Componentes

Cabeçalho

O cabeçalho de uma função definida pelo usuário, define o nome da função opcional com um esquema proprietário ou nome. Além disso, define nome do parâmetro de entrada e tipo de dados, bem como os aplicáveis às opções do parâmetro de entrada.

Corpo

O corpo de uma função definida pelo usuário, define a ação ou lógica que deve desempenhar a função e que contém um ou mais comandos T-SQL que execute a função lógica.

Benefícios

Programação Modular

Definido pelo usuário funções permite modular programação. Você pode criar uma função uma vez e guardá-la na base de dados, e então chamar a função sem número de vezes em seu programa. Funções definidas pelo usuário podem ser modificadas, independentemente do código fonte do programa.

Execução mais rápida

Funções definidas pelo usuário permitem mais rápida execução. Semelhante a stored procedures, funções definidas pelo usuário T-SQL, reduzem o tempo de compilação T-SQL em cache os planos e reutiliza-os para repetidas execuções. Por isso, as funções definidas pelo usuário, não precisam ser reparsed e otimiza o reuso com cada um.

Tráfego de rede reduzido

Definido pelo usuário funções pode reduzir o tráfego da rede. Em um ambiente cliente-servidor, uma operação que filtra os dados baseados em complexas que não podem ser expressas em um único escalar expressão pode ser expressa como uma função. Esta função pode ser invocada em seguida, a cláusula WHERE do SELECT declaração de reduzir o número de linhas enviado para o computador cliente.

Tipos

Funções Escalares

Definido pelo usuário escalar funções retornam um único valor de dados do tipo definido na cláusula os retornos. Uma função escalar não tem nenhuma função escalar o corpo e o valor é o resultado de uma única declaração. Em contrapartida, uma declaração multi-função escalar corpo tem uma função que é definida em um bloco BEGIN ... END, e contém uma série de declarações T-SQL que retornam um valor único. O regresso do tipo definido pelo usuário, funções escalar pode ser qualquer tipo de dados exceto o texto, ntext, imagem, cursor, a data e a hora.

Funções de Valor de Tabela

Definido pelo usuário mesa-valorizada funções retornam um quadro tipo de dados. Uma tabela escalar valorizada função não tem função corpo e da mesa é o resultado de um conjunto único SELECT. Em contrapartida, uma declaração multi-mesa-valorizada função tem uma função que o corpo é definida em um bloco BEGIN ... END e contém uma série de declarações T-SQL que construir e inserir linhas na tabela que é retornado.

Funções construídas internamente

Construído em funções são fornecidas pelo SQL Server para ajudá-lo a executar uma série de operações. Eles não podem ser modificados. Você pode usar funções built in em T-SQL para acessar informações a partir de declarações SQL Server sistema de tabelas sem acessar diretamente o sistema de tabelas. Você também pode executar tarefas comuns usando funções tais como SUM, GETDATE, ou identidade. Construído em funções retornam valores, quer escalar os tipos de dados ou tabela.

Como criar ou modificar funções definidas pelo usuário

Em SQL Server 2005, definido pelo usuário, funções podem ser criados com a declaração CREATE FUNCTION, modificada pela utilização do ALTER FUNCTION declaração, e removido, utilizando o DROP FUNCTION declaração. Cada totalmente qualificado função definida pelo usuário, nome (schema_name.function_name) deve ser único. Definido pelo usuário funções podem ser invocadas em consultas, de declarações, ou expressões.

CREATE FUNCTION

A declaração CREATE FUNCTION cria um definido pelo usuário, função, que é um salva T-SQL ou linguagem comum runtime (CLR) rotina que retorna um valor. Definido pelo usuário funções não podem ser utilizados para a realização de ações que modificar o banco de dados estadual. User funções definidas, como o sistema funciona, pode ser invocada a partir de uma consulta. Escalar funções podem ser executados através de uma declaração EXECUTE como procedimentos armazenados.

A cláusula SCHEMABINDING liga a função para a base de dados objetos que lhe referências. Isso depende vinculativo sobre a alteração do objeto função. Funções criadas com a cláusula SCHEMABINDING são referidos como schema-vinculada funções.

A criptografia cláusula criptografa o catálogo vista colunas que contêm o código do CREATE FUNCTION declaração. Você pode usar o sistema sp_helptext procedimento armazenado para ver a T-SQL código fonte para um definido pelo usuário, função, procedimentos, ou acionar. Criptografia não pode ser especificado para CLR funções.

ALTER FUNCTION

A declaração ALTER FUNCTION é usada para alterar um existente T-SQL função que foi criada por executar a declaração CREATE FUNCTION, sem alterar as permissões e sem afetar quaisquer funções dependentes, procedimentos armazenados, ou dispara.

ALTER FUNCTION A declaração pode ser utilizado para:

Mudar a implementação ou a transformação lógica da função.

Criptografar o conteúdo da função.

Remover ou adicionar a cláusula SCHEMABINDING para a função.

ALTER FUNCTION A declaração não pode ser utilizado para:

Mudar uma função escalar-valorizada a uma mesa-valorizada função, ou vice-versa.

Mudar uma função inline para uma declaração multi-função, ou vice-versa.

Mudar uma T-SQL CLR função de uma função, ou vice-versa.

DROP FUNCTION

A DROP FUNCTION é usada para remover uma função definida pelo usuário, a partir de uma base de dados.

DROP FUNCTION ([schema_name.] Nome_da_função)

DROP FUNCTION irá falhar, nos seguintes casos:

Se há T-SQL funções ou pontos de vista em referência a base de dados que esta função e em que foram criados através da utilização SCHEMABINDING.

Se existem colunas computadorizada, CHECK constrangimentos, ou constrangimentos que DEFAULT referência a função.

Se há referência computadorizada colunas que esta função e que tenham sido indexados.

Mudar uma T-SQL CLR função DE UMA função, ou vice-versa.

Restrições quando se cria uma função definida pelo usuário

Uma função definida pelo usuário é armazenada como um objeto de banco de dados e fornece códigos reutilizáveis que pode ser usado de várias maneiras. No entanto, existem restrições ao criar funções definidas pelo usuário.

Você não pode usar funções definidas pelo usuário, para:

Atualização de dados. Você pode usar procedimentos armazenados vez.

Definir ou criar novos objetos no banco de dados. Todos os objetos referidos pela função, com a exceção de escalar tipos, têm de ser previamente declarados e criado.

Executar operações.

Categorias de Função

Em SQL Server 2005, funções podem ser classificadas em duas categorias: deterministas e nondeterministic. Uma função é determinística se, por um conjunto específico de valores de entrada e banco de dados estadual, a função sempre retorna o mesmo resultado. Um nondeterministic função pode retornar resultados diferentes quando é chamado repetidas vezes com o mesmo conjunto de valores de entrada. Por exemplo, a função GETDATE () é nondeterministic.

Funções que são nondeterministic não podem ser invocadas em definições de T-SQL definido pelo usuário, funções, indexado pontos de vista, colunas indexadas computadorizada, ou em colunas persistiu computadorizada.

Como implementar diferentes tipos de funções definidas pelo usuário

Definido pelo usuário quer escalar-funções são valorizadas ou mesa-valorizada. Definido pelo usuário funções são valorizadas escalar-se escalar é especificado na cláusula TROCAS. Escalar-valorizada funções podem ser definidas por múltiplos usando T-SQL declarações. As funções são valorizadas se TABELA é especificado na cláusula TROCAS. Dependendo de como o corpo da função está definida, mesa-valorizada funções podem ser classificados como inline mesa-valorizada funções ou declaração de mesa multi-funções valorizadas.

Função de Valor Escalar

Os seguintes escalar-valorizada função toma um inteiro como entrada e retorna uma descrição de caracteres dependendo do valor de entrada.

```
CREATE FUNCTION [DBO]. [UfnGetDocumentStatusText] (@ Status [tinyint])
TROCAS [nvarchar] (15)
AS
-- Retorna o status vendas ordem texto representação para o estado valor.
BEGIN
DECLARAM @ ret [nvarchar] (15);
SET @ Ret =
CASO @ Status
QUANDO 1 then 'Enquanto aguarda aprovação '
QUANDO ENTÃO 2 "Aprovado"
QUANDO 3 então 'obsoletos'
OUTROS '** ** inválida'
END;
RETURN @ Ret
END;
```

Multi declaração e tabela de valor

Os seguintes escalar-valorizada função toma um inteiro como entrada e retorna uma Descrição de Caracteres dependendo do valor de entrada.

```
CREATE FUNCTION [DBO]. [UfnGetDocumentStatusText] (@ Status [tinyint])
TROCAS [nvarchar] (15)
AS
-- Retorna o status vendas ordem texto representação para o estado valor.
BEGIN
DECLARAM @ ret [nvarchar] (15);
SET @ Ret =
CASO @ Status
QUANDO 1 então "Enquanto aguarda aprovação"
QUANDO ENTÃO 2 "Aprovado"
QUANDO 3 então 'obsoletos'
OUTROS '** ** inválida'
END;
RETURN @ Ret
END;
```

Tabela de Valor Linear

O seguinte assume função de uma região como entrada e retorna uma lista de todos os clientes que pertencem a uma determinada região.

```
Create Function Sales.ufn_CustomerNamesInRegion
(@ Região nvarchar (50))
CHANGES tabela
AS
RETURN (
SELECT DISTINCT s.Name AS Store, a.City
FROM Sales.Store como s
AS JOIN Sales.CustomerAddress ca ON ca.CustomerID = s.CustomerID
AS JOIN Person.Address um ON a.AddressID = ca.AddressID
JOIN Person.StateProvince SA
sp.StateProvinceID = a.StateProvinceID
WHERE. Nome = @ Região )
```

Considerações para executar funções definidas pelo usuário

Embora as funções definidas pelo usuário possam oferecer uma grande conveniência, eles também podem degradar o desempenho. O grau de impacto de desempenho depende de como e onde você usa uma função definido pelo usuário. Isto se aplica a função built-in também.

No entanto, as funções definidas pelo usuário, são mais vulneráveis à degradação do que o desempenho interno de funções. Isso ocorre porque as funções definidas pelo usuário, são chamados a uma linha-por-linha base, em vez de trabalhar como um conjunto base de operação. Além disso, quando definido pelo usuário, têm de desempenhar funções complexas ações, tais como mesas e procurando desempenho junta, eles se tornam mais vulneráveis à degradação desempenho. Portanto, se o resultado conjunto da sua consulta, que é definido pelo usuário, utilizando uma função, tem algumas linhas, então a performance hit será menor.

Mas se o resultado é grande, então a execução poderia se tornar um problema. Em tais situações, você deve usar procedimentos armazenados. Para entender como uma função definida pelo usuário, funciona, você poderia examinar esta função escalar. Você precisa invocar o definido pelo usuário, utilizando a função GetTotalInventory essa pesquisa.

Em executando esta consulta, SQL Server processos na primeira fila do Production.Product tabela e

passa o valor da coluna ProductID como entrada para a função escalar.

Em seguida, a função executa outro SELECT query usando o ProductID sobre a mesa

Production.ProductInventory, para calcular o inventário total de ações para o produto específico. A função retorna esse valor total para o exterior SELECT query.

Em seguida, SQL Server substitui o valor de retorno na cláusula WHERE e verifica se o valor for superior a 1800. Se a resposta avalia para true, então a linha seja incluída no resultado final conjunto. Caso contrário, ele é excluído.

SQL Server repete os passos subseqüentes com as filas, até que todas as linhas a partir da tabela Production.Product tenham sido processados. Existem cerca de 500 linhas no Production.Product tabela.

A função GetTotalInventory e a subseqüente consulta SELECT são executados 500 vezes. Porque o número de linhas é relativamente pequeno, os resultados atingidos é menor.

Se houver necessidade de realizar uma consulta semelhante a uma tabela que tem mais de 10000 linhas, ou se a função definida pelo usuário, teve de fazer a varredura de mais linhas, ou se houve má índices definidos, você deverá reescrever a consulta através da utilização junta ou subconsultas, tal como exigido pela situação.

Questões

Sua empresa tem recentemente interrompida a produção de um determinado produto e que tenham decidido modificar todas as funções definidas pelo usuário, que você criou, que estão relacionados com o produto. No entanto, uma das funções definidas pelo usuário, referências outros objetos no banco de dados também. Que uma das seguintes funções ou cláusulas irá impedi-lo de fazer qualquer alteração a este definido pelo usuário, função?

[The DROP function](#)

[The SCHEMABINDING clause X](#)

[The ALTER function](#)

[The ENCRYPTION clause](#)

Qual das seguintes declarações são restrições na utilização definido pelo usuário, funções?

[Você não pode usar funções definidas pelo usuário, para invocar funções determinísticas.](#)

[Você não pode usar funções definidas pelo usuário, a atualização de dados. X](#)

[Você não pode usar funções definidas pelo usuário, para substituir procedimentos armazenados.](#)

[Você não pode usar funções definidas pelo usuário, para definir ou criar novos objetos no banco de dados. X](#)

[Você não pode incluir transações em definido pelo usuário, funções. X](#)

Você precisa declarar e criar todos os objetos tipo escalar antes de poderem ser referidas pelo usuários-definidos função.Examine os seguintes BEGIN... FIM ufnGetTotalInventory o bloco de função.

[DECLARE @ Ret int;](#)

[SELECT @ ret = SUM \(p.Quantity\)](#)

[DESDE Production.ProductInventory p](#)

[ONDE p.ProductID = @ ProductID](#)

[IF \(@ ret IS NULL\)](#)

[SET @ Ret = 0](#)

[RETURN @ Ret](#)

Determinar a seqüência correta em que irá executar o SQL Server ufnGetTotalInventory função quando a seguinte declaração é executado.

[SELECT ProductID, Nome, ProductNumber](#)

[FROM Production.Product](#)

[ONDE dbo.ufnGetTotalInventory \(ProductID\)> = 1800](#)

2 SQL Server processos cada linha da tabela especificada no invocar a função consulta.

5 SQL Server substitui o valor de retorno de a função escalar na cláusula WHERE e verificações para a condição especificada.

1 SQL Server analisa a tabela especificada no invocar a função consulta.

4 A função escalar executa uma query SELECT usando a tabela e coluna nome especificado na função criar consulta.

6 SQL Server repete a analisar o processo, e verificar em todas as linhas, até que todas as linhas da tabela especificada no invocar a função query ter sido analisados.

3 SQL Server analisa a primeira linha da tabela especificada na função invocar a consulta, e passa o valor da coluna especificada no SELECT, como entrada, para a função escalar.

Você precisa obter uma lista de todos os produtos dos Production.Product de mesa. Você quer ter acesso a esta informação a partir do SQL Server sistema de tabelas, sem acessar diretamente o sistema de tabelas. Que tipo de função que você precisa para usar para obter o resultado exigido conjunto?

[Inline escalar funções](#)

[Multi-funções declaração escalar](#)

[Construído em funções X](#)

[Tabela de funções valorizadas](#)

Encapsulando Consultas usando Views

Você pode criar pontos de vista em SQL Server para simplificar o acesso aos dados. A opinião é armazenada uma consulta ou uma tabela virtual que consiste de colunas e linhas de dados como uma tabela. Com base em requisitos específicos, você pode criar, modificar, e solte vistas usando T-SQL declarações. Depois de um ponto de vista foi criada, você pode consultar o ponto de vista apenas como uma tabela normal. Você também pode modificar os dados na tabela que tem sido utilizado para definir o ponto de vista. No entanto, você precisa estar ciente das restrições que se aplicam quando você modificar os dados em tabelas, usando pontos de vista.

O que são Views?

Uma visão é uma tabela virtual que incorpora uma consulta T-SQL. É constituída por um chamado conjunto de colunas e linhas de dados. Os dados provêm de uma visão a partir da tabela ou tabelas que são referenciadas na consulta a definição do ponto de vista.

O resultado conjunto do ponto de vista é gerado dinamicamente quando o ponto de vista é referenciado. A menos que seja indexado, um ponto de vista não existir como um conjunto de linhas armazenadas no banco de dados e colunas. A consulta que define uma visão pode ser baseada em uma ou mais tabelas ou em outros pontos de vista sobre a atual base de dados.

Você pode usar a fim de se concentrar em dados específicos que você precisa em uma tabela. Com a criação de pontos de vista de um banco de dados personalizado, você pode filtrar os dados desnecessários. Você pode definir freqüentemente usados como buscas complexas vista de forma que você não tem que especificar a consulta várias vezes.

Para manter a segurança de dados, você pode permitir aos utilizadores aceder a uma tabela através de uma opinião em vez de concedê-las permissão para acessar a tabela diretamente. Você pode usar uma vista como uma interface para trás compatível para emular uma tabela que utilizar a existir, mas cujo esquema mudou.

Você também pode usar uma vista como um filtro para copiar dados de e para o SQL Server para melhorar o desempenho e para a partição de dados.

Você pode construir opiniões baseadas em outras posições. No entanto, o real limitar a nidificação de pontos de vista dependerá da complexidade do ponto de vista e da memória disponível.

Como criar e modificar uma View

Você pode criar, modificar e queda vistas usando as instruções CREATE VIEW, ALTER VIEW, e DROP VIEW declarações. Ao usar o SQL Server Management Studio e sp_helptext o procedimento armazenado, você pode exibir a T-SQL código fonte para um ponto de vista.

CREATE

Você pode criar uma visão de conjunto, utilizando a declaração CREATE VIEW. Uma visão pode ser criada apenas no atual banco de dados e que pode ter um máximo de 1024 colunas.

```
CREATE VIEW [ schema name .] view name [ (column [ ,...n ] ) ]  
[WITH [ENCRYPTION], [SCHEMABINDING], [VIEW_METADATA]  
AS select statement  
[ WITH CHECK OPTION ]
```

ALTER

Você pode modificar a definição de uma já existente com vista a usar ALTER VIEW declaração. Você pode alterar opiniões, sem afetar dependentes procedimentos armazenados ou dispara e sem alterar permissões. Você também pode alterar indexados pontos de vista.

```
ALTER VIEW [ < database_name > . ] [ < owner > . ] view_name [ ( column [ ,...n ] ) ]  
[ WITH < view_attribute > [ ,...n ] ]  
AS select_statement [ WITH CHECK OPTION ]
```

<view_attribute > ::=

```
{ ENCRYPTION | SCHEMABINDING | VIEW_METADATA }
```

view_name. O nome do ponto de vista de que é para ser mudada.

coluna. O nome de uma ou mais colunas, separadas por vírgulas, para fazer parte da opinião dada.

N. Um espaço reservado que indica o número de vezes que uma coluna pode ser repetido.

DROP

Você pode remover um ou mais pontos de vista de a atual base de dados, utilizando o DROP VIEW declaração.

```
DROP VIEW vista) ([, ... n]
```

visualizar. O nome da vista (s) para ser removido. Ver nomes deverão obedecer às regras para identificadores.

N. Um espaço reservado que indica que várias opiniões podem ser especificados.

QUERY

Depois de um ponto de vista foi criado, ele pode ser consultada apenas como uma tabela normal.

Quando você executar uma consulta sobre um ponto de vista, os controles Database Engine que todos os objetos do banco de dados referenciados no mapa existem e que são válidas no contexto da declaração. O Database Engine também verifica que os dados modificação declarações não violam a integridade dos dados quaisquer regras ou outras regras de pontos de vista, tais como modificar um campo calculado. Um cheque que não retorna uma mensagem de erro e de uma bem sucedida verificar executa a ação contra a subjacentes ou mais quadros.

Quando você tenta consultar um ponto de vista de que depende de uma mesa ou opinião que foi abandonada, a base de dados Motor produz uma mensagem de erro. No entanto, se uma nova tabela ou opinião é criado com a mesma estrutura que o quadro caiu tabela base, uma vez mais se torna o ponto de vista funcional. Se a nova tabela ou visualizar estrutura mudanças, o ponto de vista deve ser abandonada, re-criadas ou alteradas.

Especificando Nomes de Coluna nas Views

De um ponto de vista, você deve especificar nomes na coluna os seguintes cenários:

Quando qualquer uma das colunas da vista são derivadas de uma expressão aritmética, construída em função, ou constante.

Quando duas ou mais colunas na opinião tem o mesmo nome.

Quando o ponto de vista de uma definição inclui aderir e as colunas de duas ou mais tabelas diferentes têm o mesmo nome.

O código a seguir exemplos demonstram como criar e modificar pontos de vista.

Considerações quando se cria Views

Você pode utilizar várias cláusulas de limitar e filtrar os dados que podem ser acessados através de views. No entanto, é necessário considerar determinados fatores quando você estiver criando views. Estes fatores irão determinar os dados que os usuários podem acessar através de views.

Limite de Coluna

Quando você cria uma opinião a juntar várias tabelas, o número total de colunas referenciadas na opinião não pode ser superior a 1024 devido à capacidade máxima especificação do SQL Server.

COMPUTE

Você não pode usar a cláusula COMPUTE em um CREATE VIEW definição porque o resumo dos resultados das declarações, incluindo a COMPUTE BY ou COMPUTE declaração não são armazenados no banco de dados.

ORDER BY

Você não pode usar a cláusula ORDER BY em pontos de vista, funções inline, tabelas derivadas, e subconsultas, a menos que você também incluir a palavra-chave TOP.

INTO

Você não pode usar a cláusula INTO com a declaração SELECT em uma perspectiva definição.

TABELA TEMPORARIA

Você não pode definir um ponto de vista de que uma tabela temporária ou referências tabela variável, porque quando você consulta a tabelas utilizando um ponto de vista, SQL Server verificações para se certificar de que todos os objetos do banco de dados referenciados em qualquer ponto da declaração existe fisicamente no banco de dados

GO(Operador de Lote)

Você não pode combinar a declaração CREATE VIEW com qualquer outra declaração em um único lote. Portanto, você deve especificar um GO declaração antes e depois de qualquer CREATE Processual, CREATE VIEW, ou CREATE FUNCTION comandos para isolá-los do restante das instruções de um lote. Especificando quaisquer comandos antes do CRIA Processual, CREATE VIEW, ou CREATE FUNCTION irá resultar em um erro.

SELECT

Você pode usar SELECT * em vista uma definição, desde que o SCHEMABINDING cláusula não é especificada no ponto de vista. Se uma das colunas da tabela subjacente de um ponto de vista é excluído, você irá obter resultados inesperados quando você consulta o ponto de vista. Por este motivo, recomenda-se a não utilizar a cláusula SELECT *, tendo em vista as definições.

Examinando o impacto do SELECT * nas Views

Para compreender o impacto do uso da declaração SELECT *, em primeiro lugar que você precisa para abrir o SQL Server Management Studio. Na caixa de diálogo Conectar ao servidor, verifique se o tipo Server, servidor de nome, e as caixas Autenticação são preenchidos corretamente. Depois de aceitar as configurações padrão, clique em Ligar. Agora, você está conectado ao servidor. Na barra de ferramentas, clique em Nova Consulta. Em SQL Server, você precisa escolher AdventureWorks como a base de dados sobre o qual trabalhar. Para fazer isso, na barra de ferramentas, clique AdventureWorks. Você agora está conectado ao banco de dados AdventureWorks.

Você precisa criar uma tabela que inclui três colunas como CustomerID, CustomerName e idade. Para fazer isso, digite a consulta na janela de busca. Um novo quadro chamado ViewTest é criada. Você precisa inserir algumas colunas na tabela ViewTest. Para fazer isso, digite a consulta na janela de busca. Três linhas com os nomes de coluna CustomerID, CustomerName, e idade são inseridas na tabela. Em seguida, você precisa criar uma vista com o nome vCustomers. Para fazer isso, digite a consulta na janela de busca. Por conseguinte, a fim nomeado como vCustomers é criada.

Você precisa executar a SELECT query contra a opinião e verificar que os resultados sejam exibidos corretamente. Para fazer isso, digite a consulta na janela de busca. Você vai descobrir que as três linhas que você inseridos são exibidos. O resultado contém a CustomerID, CustomerName, e idade. Você agora necessidade de alterar a estrutura do ViewTest tabela. Para fazer isto, adicione uma nova coluna denominada Discount inteiro. A tabela é modificada com a adição da nova coluna denominada Discount.

Tipo a declaração para inserir uma nova linha. A nova linha com o CustomerID, CustomerName, idade, e de desconto é adicionado. Tipo o mapa para ver o conteúdo de ViewTest. Agora, as declarações de tipo para executar uma consulta SELECT contra a mesa e, em seguida, o ponto de vista. O original três colunas que estavam presentes quando o ponto de vista foi criada sejam devolvidos. Repare que a recém-criada quarta coluna não é exibido no resultado.

Você precisa alterar o quadro mais uma vez. Desta vez, precisa excluir a coluna denominada Idade. Executa uma consulta SELECT contra a mesa e da opinião. Desta vez também o resultado conjunto tem três colunas. No entanto, os valores na quarta coluna que é o Discount estão agora a ser devolvido pela opinião na Idade coluna. Você pode evitar esta incompatibilidade, utilizando a opção SCHEMABINDING ao mesmo tempo definir os pontos de vista.

Agora crie uma nova tabela. Uma nova tabela com os ViewTest2 nome é criada. Em seguida, você precisa inserir algumas colunas na tabela ViewTest2. Para fazer isso, digite a consulta na janela de busca. Três linhas com os nomes de coluna CustomerID, CustomerName, e idade são inseridas na tabela. Você precisa garantir que o ponto de vista é criada com a opção SCHEMABINDING.

Você também precisa explicitamente usar nomes na coluna SELECT query definido na opinião porque você está usando a opção SCHEMABINDING. Além disso, o quadro nomes têm de ser especificados usando dois, uma parte convenção nomeando como Schema.tablename. Para confirmar os resultados, execute o SELECT query contra o ponto de vista. Você encontrará três linhas que são exibidas. O resultado contém a CustomerID, CustomerName, e idade.

Você pode alterar a estrutura da tabela, adicionando uma nova coluna denominada Discount inteiro. A tabela é modificada com a adição da nova coluna denominada Discount. Tipo a declaração para inserir uma nova linha na tabela ViewTest2. Ao executar essa pesquisa, você tem valores inseridos em quatro colunas em vez de três colunas. Agora, execute uma consulta SELECT contra o ponto de vista. O original três colunas que estavam presentes quando o ponto de vista foi criada sejam devolvidos. Por isso, o recém-criado quarta coluna não é exibido no resultado. Você precisa de alterar o quadro, mais uma vez. Desta vez excluir a coluna denominada Idade. Agora, quando você tentar alterar a estrutura da tabela, você recebe uma mensagem de erro.

Isto ilustra a vantagem de incluir a opção SCHEMABINDING, que impede qualquer mudança para a estrutura da tabela.

Esta demonstração mostra-lhe o impacto do uso da declaração SELECT * em pontos de vista e as consequências da modificação das estruturas tabela depois de uma posição já havia sido criado.

Examinar o impacto da utilização de SELECT * em Vistas

No menu Iniciar, aponte para Todos os Programas, aponte para Microsoft SQL Server 2005 e, em seguida, clique em SQL Server Management Studio. O SQL Server Management Studio janela aparece.

Na caixa de diálogo Conectar ao servidor, verifique se o Tipo de servidor, nome do servidor, e as caixas Autenticação ter sido preenchido corretamente e, em seguida, clique em Connect. Na barra de ferramentas, clique em Nova Consulta. A nova consulta janela é exibida.

Na barra de ferramentas, Bancos de dados disponíveis a partir da lista, selecione AdventureWorks.

Para criar uma tabela, na janela de busca, digite a seguinte declaração. CREATE TABLE ViewTest (CustomerID int, CustomerName VARCHAR (50), Idade int)

Na barra de ferramentas, clique em Executar.

Para inserir linhas, na janela de busca, digite as seguintes afirmações. INSERT INTO View Test (CustomerID, CustomerName, Idade) VALUES (1, 'Victor', 35) INSERT INTO ViewTest (CustomerID, CustomerName, Idade) VALUES (2, 'Harris', 33) INSERT INTO ViewTest (CustomerID, CustomerName, Idade) VALUES (3, 'David', 37)

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para criar um ponto de vista, na janela de busca, digite a seguinte declaração. Criar vCustomers vista como SELECT * FROM ViewTest

Selecione a consulta. Na Barra de Ferramentas, clique em Executar.

Para verificar a consulta, na janela de busca, digite a seguinte declaração. SELECT * FROM vCustomers

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para alterar a estrutura da tabela, na janela de busca, digite a seguinte declaração. ALTER TABLE ADD ViewTest desconto int

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para inserir uma nova linha na tabela ViewTest, na janela nova consulta, digite a seguinte declaração. INSERT INTO ViewTest (CustomerID, CustomerName, idade, Desconto) VALUES (4, 'Maria', 31,5)

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para executar uma consulta SELECT contra a ver, na janela de busca, digite a seguinte declaração. SELECT * FROM ViewTest SELECT * FROM vCustomers

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para soltar a Idade coluna, na janela de busca, digite a seguinte declaração, selecione a declaração, e sobre a barra de ferramentas, clique em Executar. ALTER TABLE ViewTest gota coluna idade

Para executar uma consulta SELECT contra o ponto de vista, digite a seguinte declaração. SELECT * FROM ViewTest SELECT * FROM vCustomers

Selecione a consulta. Na Barra de Ferramentas, clique em Executar.

Para criar uma tabela, na janela de busca, digite a seguinte declaração. CREATE TABLE ViewTest2 (CustomerID int, CustomerName VARCHAR (50), Idade int)

Selecione a consulta, e sobre a barra de ferramentas, clique Executar.

Para inserir novas linhas no ViewTest2 mesa, na janela de busca, digite as seguintes afirmações. INSERT INTO ViewTest2 (CustomerID, CustomerName, Idade) VALUES (1, 'Victor', 35) INSERT INTO ViewTest2 (CustomerID, CustomerName, Idade) VALUES (2, 'Harris', 33) INSERT INTO ViewTest2

(CustomerID, CustomerName, Idade) VALUES (3, 'David', 38)

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para criar um ponto de vista, na janela de busca, digite a seguinte declaração. CREATE VIEW vCustomers2 COM SCHEMABINDING AS SELECT CustomerID, CustomerName, Idade FROM ViewTest2 GO Em seguida, selecione a consulta e, na barra de ferramentas, clique em Executar.

Para verificar o resultado, na janela de busca, digite a seguinte declaração. SELECT * FROM vCustomers2

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para alterar a tabela, na janela de busca, digite a seguinte declaração. ALTER TABLE ViewTest2 ADD Desconto int

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para inserir uma linha, na janela de busca, digite a seguinte declaração. INSERT INTO ViewTest2 (CustomerID, CustomerName, Idade, Desconto) VALUES (4, 'Maria', 31, 5)

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para executar uma consulta SELECT, na janela de busca, digite a seguinte declaração. SELECT * FROM vCustomers

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Para remover a Idade coluna, na janela de busca, digite a seguinte declaração. ALTER TABLE ViewTest2 DROP COLUMN Idade

Selecione a consulta. Na barra de ferramentas, clique em Executar.

Restrições quando se modifica dados usando views

Você pode usar a fim de modificar a tabela base nos dados de um ponto de vista. Você pode usar o UPDATE, INSERT e DELETE declarações contra um ponto de vista, assim como você para fazer uma tabela. No entanto, existem algumas restrições ao modificar dados através de pontos de vista.

Nas declarações

Uma visão pode ser definido através de uma consulta SELECT que junta dados de várias tabelas. No entanto, as seguintes restrições aplicam-se às declarações por escrito que modifiquem dados usando visitas:

Quaisquer alterações, tais como UPDATE, INSERT e DELETE declarações, quando aplicada a um ponto de vista, deve modificar colunas de apenas uma tabela base.

Se a opção WITH CHECK é usada na view a definição, todos os dados modificação declarações executadas contra a opinião devem estar de acordo com os critérios estabelecidos no âmbito da declaração SELECT definindo o ponto de vista.

INSERT declarações devem especificar valores para todas as colunas na tabela que não permitem valores nulos e não têm DEFAULT definições.

Nas colunas

As seguintes restrições aplicam-se às colunas ao modificar dados utilizando posições:

Colunas ser modificados de um ponto de vista deve referência os dados subjacentes nas colunas da tabela diretamente. Você não pode modificar colunas computadorizada ou derivados usando funções agregadas como AVG, COUNT, SUM, MIN, MAX, agrupamento, STDEV, STDEVP, VAR, e VARP.

Colunas computada a partir de uma expressão ou de operadores estabelecidos como UNION, UNION ALL, CROSSJOIN, com exceção, e CRUZE não pode ser alterado a menos que você especifique uma vez de acionamento.

Nota: Ao usar uma vez de acionamento, você pode especificar as medidas a tomar pelo SQL Server no lugar do original ação.

Colunas ser modificado por um ponto de vista não pode ser afetada pelo GROUP BY, ou DISTINCT cláusulas.

Os dados modificados em colunas da tabela subjacente deve manter as restrições sobre essas colunas, tais como nullability, constrangimentos, DEFAULT e definições. Por exemplo, se uma fila é excluída, todas as restrições FOREIGN KEY em tabelas relacionadas devem ainda ser preenchidas para que a DELETE para alcançar o sucesso.

Exemplos:

O código a seguir ilustra o exemplo restrições a modificando os dados, utilizando um ponto de vista.

A seguinte VIEW não é atualizada.

```
USE AdventureWorks;
GO
Criar vista TotalSalesContacts
AS
SELECT C. apelido, soma (O. TotalDue) como TotalSales
DESDE Sales.SalesOrderHeader o, c Person.Contact
ONDE c.ContactID = o.ContactID
GROUP BY apelido
```

A vista não pode ser atualizado pelos seguintes motivos:

O apelido da coluna TotalSalesContacts não podem ser modificados, porque a coluna tenha sido afetado por uma cláusula GROUP BY.

O SQL Server não será capaz de UPDATE, INSERT, ou DELETE exigida a exemplo do último nome, se houver mais de uma instância do mesmo sobrenome.

O TotalSales coluna de TotalSalesContacts não pode ser alterado porque se trata de uma coluna derivada de uma função agregada. Por conseguinte, um erro será retornado.

O SQL Server não pode rastrear a TotalSales coluna diretamente para a base SalesOrderHeader tabela.

Você pode tentar modificar a View como se pode ver no seguinte código.

```
UPDATE TotalSalesContacts
SET apelido = 'Allen'
WHERE apelido = 'Alexandre'
```

Mas você receberá a seguinte mensagem de erro.

Msg 4403, Level 16, Estado 1, Linha 2 Não é possível atualizar o ponto de vista ou função "TotalSalesContacts" porque contém agregados ou uma cláusula DISTINCT.

Construindo uma View

Você é administrador do banco de dados em Adventure Works. Você precisa modificar e atualizar as tabelas no banco de dados, utilizando um ponto de vista. Você irá construir uma perspectiva dinâmica e ver como o ponto de vista de definição varia de acordo com as opções que você escolher. Questões

Você foi convidado a criar uma perspectiva para uma tabela. Sua consulta contém a cláusula SELECT para definir o ponto de vista. Qual das seguintes cláusulas não devem ser utilizados na declaração SELECT para definir o ponto de vista?

FROM
OPTION X
WHERE
INTO X
ORDER BY
COMPUTE BY X

Você precisa alterar os dados de uma tabela base subjacentes através de um ponto de vista, mas você descobrir que existem algumas restrições quando você atualizar dados em tabelas, usando pontos de vista. Qual das seguintes palavras-chave cláusulas ou restringir a atualização das tabelas usando vistas?

GROUP BY X
COUNT X
CHECK
SCHEMABINDING

Você precisa alterar os dados em uma tabela através de um ponto de vista. Sempre que você alterar uma linha, você precisa se assegurar que os dados permanece visível na opinião mesmo após a modificação. Que uma das seguintes opções na declaração SELECT garante a visibilidade dos dados modificados?

CREATE VIEW
SCHEMABINDING
ENCRYPTION
WITH CHECK OPTION X

Resumo de Stored Procedures

Um procedimento armazenado é chamado uma coleção de T-comandos SQL que está armazenado no servidor. Ao utilizar procedimentos armazenados, você pode encapsular tarefas e regras comerciais e, assim, melhorar o desempenho e a integridade dos dados. Um desencadear-se de um tipo especial de procedimento armazenado que executa automaticamente sempre que é feita uma tentativa de modificar os dados em uma tabela que a desencadeiam é definida em. Procedimentos armazenados reduzir significativamente as necessidades dos recursos e de tempo de execução.

O que são Stored Procedures?

Um procedimento armazenado é chamado uma coleção de T-comandos SQL que está armazenado no servidor. Você pode encapsular que executam tarefas repetitivas de forma eficiente através da utilização de um procedimento armazenado. Procedimentos armazenados aceita parâmetros de entrada e de saída parâmetros para retornar a chamada processo. Contêm programação declarações que executar operações em um banco de dados. Eles também retornar um valor para um estatuto chamando procedimento para indicar sucesso ou fracasso. Você não pode trabalhar diretamente com os procedimentos armazenados, mas a escrever consultas eficientes que você precisa compreender a forma como eles são executadas em SQL Server.

Benefícios dos procedimentos armazenados

Procedimentos armazenados têm as seguintes vantagens:

Aplicação lógica pode ser compartilhado entre aplicativos, garantindo assim o acesso aos dados consistente e modificação.

Negócios regras ou políticas encapsulada em procedimentos armazenados podem ser alterados em um único local. Todos os clientes podem usar os mesmos procedimentos armazenados dados consistentes para garantir o acesso e a modificação.

Detalhes do esquema do banco de dados pode ser captada a partir de usuários.

Os usuários podem ser concedida permissão para executar um procedimento armazenado, mesmo que não tem permissão para acessar as tabelas ou opiniões que estão armazenadas no referido processo.

Desempenho de uma aplicação é melhor porque muitas tarefas implementar procedimentos armazenados como uma série de declarações T-SQL.

Rede de tráfego podem ser reduzidas.

O exemplo a seguir mostra a criação de um simples procedimento armazenado em um SELECT. Este procedimento armazenado retorna as informações sobre funcionários cujos nomes começam com um caractere especial. Ela aceita um parâmetro de entrada.

```
Use AdventureWorks;
GO
Criar processo HumanResources.usp_GetEmployeesFromName
@ NamePrefix char (1)
AS
BEGIN
SELECT EmployeeID, firstname, apelido, EMAILADDRESS
DESDE HumanResources.vEmployee
ONDE firstname LIKE @ NamePrefix + '%'
Por fim firstname
FIM
```

A seguinte consulta ilustra como um procedimento armazenado pode ser executada, utilizando o comando EXECUTE.

```
EXECUTE HumanResources.usp_GetEmployeesFromName 'A'
```

O seguinte é o resultado conjunto de completar a consulta.

EmployeeID	FirstName	LastName	EmailAddress
44	A. Scott	Wright	ascott0@adventure-works.com
170	Alan	Brewer	alan0@adventure-works.com
31	Alejandro	McGuel	alejandro0@adventure-works.com
155	Alex	Nayberg	alex0@adventure-works.com
36	Alice	Ciccu	alice0@adventure-works.com

Como as Stored Procedures são executadas

Quando você executar um procedimento armazenado pela primeira vez, o SQL Server query otimizar constrói um plano de execução para o procedimento armazenado. Este plano de execução compilado é armazenado no cache do procedimento armazenado, na memória, sobre o SQL Server. SQL Server

tenta este plano, para posterior reutilização execuções do procedimento armazenado. Isso melhora o desempenho do SQL Server porque não precisa de repetir as diversas fases de cada vez que o procedimento armazenado seja executado.

Quando um procedimento armazenado é executado pela primeira vez, SQL Server tem de passar por uma série de fases.

Parse

No parsing fase, a consulta SQL Server verifica a sintaxe e transforma-lo em um compilador-estrutura pronta. O SQL Server utiliza esse tipo de estrutura mais tarde para otimizar a consulta.

Resolving

Na resolução ou normalização fase, Servidor SQL verifica todas as referências a objetos na consulta. Isto é onde você pode obter o "Objeto não encontrado" mensagem quando um objeto referenciado na consulta não for encontrado no banco de dados.

Otimizing

Na fase de otimização, o otimizador SQL Server, sai com um plano de execução eficiente disponível para cada comando SQL. O plano eficaz é a que vai utilizar menos recursos, tais como, CPU e de I / O, para obter o resultado desejado.

Para otimizar LMG declarações, SQL Server testes diferentes índices e juntar ordens para obter o melhor plano para executar a consulta. Porque um complexo consulta terá em conta todos os índices e junta, pode haver muitos caminhos para executar a consulta.

Em tais casos, a determinação do melhor caminho para a otimização pode demorar muito tempo. Este, por sua vez, pode conduzir a custos mais elevados.

Compiling

Na fase da compilação, o SQL Server inicia a construção do plano de execução para a consulta.

Primeiro, ele cria uma seqüência árvore. A seqüência árvore está resolvido, acrescentando que inclui as conversões implícitas, caso seja necessário. Além disso, se a consulta referências pontos de vista, uma opinião definição é colocada na consulta.

Se uma declaração DML é uma declaração, um objeto especial chamado a consulta gráfico é criado. O otimizador obras sobre essa pesquisa gráfico para gerar um plano para a consulta. Este plano é compilado em seguida o procedimento armazenado em cache para reutilização.

Executing

Na fase da execução, depois de todos os planos de execução foram gerados, SQL Server escolhe a eficácia do plano que irá consumir menos recursos e, em seguida, executa a consulta.

O que são Triggers?

Um desencadear-se de um tipo especial de procedimento armazenado que executa, sempre que seja feita uma tentativa de modificar os dados em uma tabela que protege o gatilho. Desencadeia normalmente são utilizados para manter a integridade dos dados de baixo nível e não utilizados para a consulta retornar resultados. Versões anteriores do SQL Server desencadeia a ser permitido apenas para os dados especificados modificação língua (LMG) declarações. No entanto, no SQL Server 2005, você também pode especificar os dados para a definição desencadeia língua (DDL) afirmações como ALTER TABLE e CREATE LOGIN.

Existem duas categorias de LMG dispara: em vez de e depois. A vez de acionamento também é conhecido como o desencadeamento ANTES.

INSTEAD OF

INSTEAD OF desencadeadores são executadas no local de desencadear a ação.

Em substituição da desencadeia também pode ser definida em uma ou mais pontos de vista com base tabelas. Em substituição da desencadeia pode estender os tipos de atualizações vista pode apoiar.

Apenas uma vez de acionamento pode ser definida por desencadear ações.

Eles não são permitidos em tabelas que são alvos de cascata integridade referencial constrangimentos.

AFTER

Triggers AFTER são executados após a ação do INSERT, UPDATE, ou DELETE declaração seja realizada. Eles podem ser especificados apenas em tabelas.

Múltiplos triggers AFTER podem ser definidos para desencadear uma única ação. A DML e triggers DLL são aninhados quando a ação de uma acionar inicia na próxima desencadear. Quando triggers aninhados são estabelecidos a 0, aciona o APÓS não pode cascata.

Em substituição da triggers são executados no local de desencadear a ação.

Em substituição da desencadeia também pode ser definida em uma ou mais pontos de vista com base tabelas. Em substituição da desencadeia pode estender os tipos de atualizações vista pode apoiar.

Apenas uma vez de acionamento pode ser definida por desencadear ações.

Eles não são permitidos em tabelas que são alvos de integridade referencial e restrições .

Benefícios da utilização do trigger

A seguir estão alguns dos benefícios da utilização da dispara:

Eles podem cascata através de alterações relacionadas com tabelas no banco de dados.

Eles podem proteger contra maliciosos ou incorretas INSERT, UPDATE, DELETE e executar operações e outras restrições que são mais complexos do que os definidos com CHECK constrangimentos. Ao contrário CHECK constrains DML pode aciona referência colunas em outras tabelas. Por exemplo, um acionará SELECT pode utilizar um mapa a partir de outro quadro para comparar com os dados inseridos ou atualizados e para realizar ações adicionais, tais como exibir ou alterar os dados definido pelo usuário, uma mensagem de erro.

Eles podem avaliar o estado de uma tabela antes e depois de uma modificação dados e tomar ações baseadas nessa diferença.

Múltiplas desencadeia do mesmo tipo, tais como INSERT, UPDATE, ou DELETE, sobre uma mesa permite múltiplas, diversas ações a ter lugar em resposta à declaração, a mesma alteração.

Como um trigger funciona?

Desencadeia pode voltar transações se uma regra específica empresarial não está satisfeita. Ao acionar um rollback que contém uma declaração SQL é executado a partir de um lote, todo o lote será cancelada. Todos os dados que foram modificados pela desencadeia a ação é retirada a declaração de repetir a transação de desencadeamento. No entanto, qualquer declaração na sequência da declaração repetir a transação ainda será executado. Portanto, quaisquer modificações que ocorrem após o rollback não são rolados para trás. É geralmente recomendada para evitar dar qualquer declaração após uma declaração repetir a transação, a menos que sejam informativos.

INSTEAD OF

Desencadeia pode voltar transações se um determinado ANTES ou em vez de triggers, que foram introduzidas com o SQL Server 2000, são executados no local de desencadear uma ação.

É mostrado um exemplo da vez, ou acionar ANTES. Em executando esta consulta, SQL Server gera um chamado acionar dEmployee, HumanResources.Employee sobre a mesa.

A vez de acionamento é desencadeado cada vez que você executar uma operação DELETE

HumanResources.Employee sobre a mesa. Ao executar a vez de acionamento, SQL Server retorna a operação se um usuário tentar apagar um empregado. Isto porque, de acordo com as regras comerciais, excluindo trabalhadores não é permitido.

Você pode ter apenas uma vez de acionamento com o mesmo quadro e a mesma ação tipo de busca. Não pode ser, no máximo, três em substituição da desencadeia associada a uma mesa, uma ação para cada tipo de busca: INSERT, UPDATE, e DELETE.

AFTER

APÓS desencadeia são executadas na sequência de uma ação desencadeia, como uma inserção, atualização ou apagar. Neste exemplo, irá desencadear o incêndio após um UPDATE declaração já tiver sido executada contra o quadro de funcionários.

Depois de executar essa consulta, sempre que você atualizar a tabela HumanResources.Employee, o ModifiedDate coluna do mesmo quadro, fica atualizado com a data actual, para todas as linhas que são afectados.

Isso poupa-lhe a partir de actualização da coluna ModifiedDate cada vez que você atualizar a tabela. Pode haver vários APÓS desencadeia associada a uma única tabela. APÓS desencadear um incêndio só depois a acção consulta que invocou o tenha modificado após acionar os dados.

Identificando as funcionalidades dos diferentes objetos de programação

Arraste cada recurso para a correcta programação objeto. Clique em Enviar para verificar a sua resposta. Clique em Visualizar para visualizar Responda a resposta correta. Clique em Reiniciar para reiniciar a actividade

Questões

CREATE PROCEDURE HumanResources.usp_GetEmployeesFromName @NamePrefix char(1)

Que uma das seguintes afirmações é verdadeira sobre o que precede CREATE PROCEDURE declaração?

Você precisa obter informações a partir do HumanResources tabela.

Você precisa dar um contributo único personagem quando você executar esse procedimento. X

Você não precisa especificar a variável @ NamePrefix nesta declaração.

Você precisa dar um nome inteiro como entrada quando você executar esse procedimento.

Identifique a sequência correta em que SQL Server executa procedimentos armazenados.

3 SQL Server cria e resolve uma sequência árvore, acrescentando que inclui as conversões implícitas, caso seja necessário.

5 SQL Server testes diferentes índices e junta ordens para obter o melhor plano para executar a consulta.

1 SQL Server verifica a próxima consulta a sintaxe e transforma-lo em um compilador-estrutura pronta.

6 Depois de todos os planos de execução foram gerados, o SQL Server um escolhe o que vai consumir a menor quantidade de recursos e executar o comando.

4 Se a próxima consulta é referenciamento pontos de vista, uma opinião definição é colocada na consulta.

2 SQL Server verifica todas as referências a objectos na próxima consulta.

Que desencadeia uma das seguintes você vai usar para garantir que quaisquer modificações feitas para uma tabela são lançados de volta?

Rollback

APÓS X

INSTEAD OF

BEFORE

Escrevendo Consultas Distribuídas

Normalmente as organizações empresariais têm em vários dados relacionais e não-fontes de dados relacionais, tais como folhas de cálculo, bases de dados, e arquivos de texto. Ao usar consultas distribuídas no SQL Server, você pode acessar esses dados em fontes de dados heterogêneos sem ligação explícita às fontes de dados. Estas fontes de dados podem ser armazenados no mesmo computador ou em computadores diferentes. Você pode acessar dados remotos heterogêneo por escrito distribuído consultas ad hoc. Você também pode usar um servidor ligado a configuração executar comandos contra OLE DB fontes de dados em servidores remotos. Vinculado servidor configurações são diferentes das consultas ad hoc, no sentido de que criou uma ligação permanente e permitir acesso ao servidor remoto.

Como o SQL Server Trabalha com dados heterogêneos

Normalmente as organizações empresariais têm em Vários dados relacionais e NÃO-Fontes de dados relacionais, tais como folhas de cálculo, bases de dados, e Arquivos de texto. Ao usar consultas distribuídas não SQL Server, você PODE acessar esses dados em Fontes de dados heterogêneos sem ligação explícita às Fontes de dados. Estas Fontes de dados podem não MESMO sér armazenados em computador uo COMPUTADORES diferentes. Você PODE acessar dados Remotos heterogêneo por escrito distribuído consultas ad hoc. Você também PODE usar um Servidor ligado uma configuração executar comandos contra OLE DB Fontes de dados em Servidores Remotos. Vinculado Servidor configurações são diferentes das consultas ad hoc, não Sentido De que criou uma ligação permanente e permitir acesso ao Servidor Remoto.

Como escrever consultas distribuídas Ad Hoc

Você pode acessar dados remotos heterogêneo por escrito distribuído consultas ad hoc. Esta técnica é melhor utilizada se os dados remotos não precisa de ser acessados com frequência. Ao usar consultas ad hoc distribuídos, você pode evitar a criação de uma ligação permanente à fonte de dados externos e, assim, manter uma melhor segurança.

SQL Sever rowset fornece duas funções, tais como OPENROWSET e OPENDATASOURCE, que pode ser utilizado para escrita distribuída consultas ad hoc.

Antes que você possa executar uma consulta ad hoc distribuídos, você precisa para transformar o Ad Hoc de Consultas Distribuídas por opção pela utilização do procedimento armazenado sp_configure.

A declaração seguinte é usada para ativar a opção Consultas Ad Hoc Distribuídos.

```
sp_configure 'show advanced option',1
RECONFIGURE
GO
sp_configure 'Ad Hoc Distributed Queries', 1
RECONFIGURE
```

OPENROWSET

Você pode incluir todas as informações que a ligação é necessário para aceder a dados remotos de uma fonte de dados OLE DB usando a função OPENROWSET. Esta função converte os dados do objeto remoto a um SQL Server compatível rowset. O OPENROWSET função pode ser referenciada na cláusula FROM de uma consulta como se fosse uma tabela nome. Também pode ser referenciado como alvo o quadro de um INSERT, UPDATE, DELETE ou declaração, sujeito às capacidades do fornecedor OLE DB.

```
OPENROWSET
( { 'provider_name' , { 'datasource' ;
```

```
'user_id' ; 'password' |  
'provider_string' }  
, { [ catalog. ] [ schema. ]  
object  
| 'query' } )
```

provider_name. A sequência de caracteres que representa o PROGID do fornecedor OLE DB, conforme especificado no registro.

dados. Uma string constante que corresponde a uma determinada fonte de dados OLE DB.

user_id. Uma string constante que é o nome do usuário a ser passado para o especificado fornecedor OLE DB.

senha. Uma string constante que é o usuário senha a ser passado para o fornecedor OLE DB.

provider_string. Um provedor de conexão específica string que é passada para inicializar o fornecedor OLE DB.

catálogo. O nome do catálogo ou banco de dados no qual o objeto especificado reside.

schema. O nome do esquema para o proprietário ou objeto determinado objeto.

objeto. O objeto nome que identifica o objeto.

consulta. Uma string constante enviados para e executado pelo prestador.

OPENDATASOURCE

Você pode usar a função OPENDATASOURCE para criar uma consulta ad hoc distribuídos. Esta função pode ser usada em T-SQL sintaxe sempre ligada servidor de nomes são usados. OPENDATASOURCE também pode ser utilizado como a primeira parte de uma parte de quatro nome que remete para uma tabela ou um nome em vista SELECT, INSERT, UPDATE, ou DELETE declaração. Pode também ser usado para se referir a um procedimento remoto armazenados em uma declaração EXECUTE. Quando armazenado execução remota de procedimentos, OPENDATASOURCE deve referir-se a outra instância do SQL Server. OPENDATASOURCE não aceita as variáveis para seus argumentos.

OPENDATASOURCE (provider_name, init_string)

provider_name. O nome registrado como o PROGID do fornecedor OLE DB e é usada para acessar a fonte de dados.

init_string. A ligação string passada para o destino fornecedor OLE DB.

Comparação das Funções OPENROWSET e OPENDATASOURCE

Ambas as funções OPENROWSET e OPENDATASOURCE deve referir-se a fontes de dados OLE DB que são pouco acessados.

As duas funções não fornecer todas as funcionalidades do servidor-associadas, tais como definições de gestão de segurança ea capacidade de busca catalogar informações.

OPENDATASOURCE fornece informações como a ligação a uma parte dos quatro-parte objeto OPENROWSET nome que prevê a ligação à corda e ao objeto nome em uma lista separada por

vírgulas.

OPENROWSET especifica um objeto ou consulta e podem ser utilizadas na cláusula FROM de uma consulta que OPENDATASOURCE especifica uma fonte de dados e pode ser usado como a primeira parte, em nome dos quatro-partes que se refere a uma mesa em um SELECT.

Exemplos:

Por exemplo o código a seguir, você precisa ter instalado Microsoft Office Access eo banco de dados Northwind da amostra. Quando você instala Office Access, a amostra de dados Northwind estarão disponíveis por padrão no caminho: C: \ Arquivos de programas \ Microsoft Office \ OFFICE11 \ amostras.

O código a seguir mostra como exemplo uma consulta utiliza a função OPENROWSET, e os Microsoft.Jet.OLEDB.4.0 provedor para a abertura de um banco de dados do Microsoft Office Access localizado em uma pasta especial, usando uma conta de usuário chamado "admin" com uma senha em branco. A última parâmetros especificar o nome da tabela, como neste exemplo Clientes, de onde as filas são supostos ser devolvido.

```
SELECT CustomerID, FROM COMPANYNAME  
OPENROWSET ( 'Microsoft.Jet.OLEDB.4.0', 'C: \ Arquivos de programas \ Microsoft Office \ OFFICE11  
\ AMOSTRAS \ Northwind.mdb ";" admin';', Clientes)
```

O seguinte é o resultado parcial conjunto da consulta.

CustomerID	COMPANYNAME
ALFKI	Alfreds Futterkiste
ANATR	Ana Trujillo Emparedados y helados
ANTON	Antonio Moreno Taquería
AROUT	Ao redor do Corno
BERGS	Berglunds snabbköp

Usando um SELECT Query com OPENROWSET

O código a seguir mostra como exemplo a mesma consulta pode ser reescrita em uma consulta SELECT passou para a função OPENROWSET vez de um objeto nome.

```
SELECT *  
FROM OPENROWSET ( 'Microsoft.Jet.OLEDB.4.0', 'C: \ Arquivos de programas \ Microsoft Office \  
OFFICE11 \ AMOSTRAS \ Northwind.mdb';  
"admin';", 'Selecione CustomerID, COMPANYNAME de Clientes')
```

O seguinte é o resultado parcial conjunto da consulta.

CustomerID	COMPANYNAME
ALFKI	Alfreds Futterkiste
ANATR	Ana Trujillo Emparedados y helados
ANTON	Antonio Moreno Taquería
AROUT	Ao redor do Corno
BERGS	Berglunds snabbköp

Conectando um Banco de Dados com a Função OPENDATASOURCE

O código seguinte exemplo mostra como você pode se conectar ao banco de dados Northwind e recuperar todas as linhas a partir da tabela de clientes usando a função OPENDATASOURCE.


```
SELECT CustomerID, COMPANYNAME
FROM OPENDATASOURCE ( 'Microsoft.Jet.OLEDB.4.0',
"fonte de dados = C: \ Arquivos de programas \ Microsoft Office \ OFFICE11 \ AMOSTRAS \
Northwind.mdb; user id = admin')... Clientes
```

O seguinte é o resultado parcial conjunto da consulta.

CustomerID	COMPANYNAME
ALFKI	Alfreds Futterkiste
ANATR	Ana Trujillo Emparedados y helados
ANTON	Antonio Moreno Taquería
AROUT	Ao redor do Corno
BERGS	Berglunds snabbköp

Usando um cliente SQL nativa OLE DB Provider

O código seguinte exemplo mostra como você pode usar o SQL Nativo Cliente fornecedor OLE DB (SQLNCLI) para acessar o HumanResources.Department tabela AdventureWorks no banco de dados no servidor remoto GEN-DEV. A declaração SELECT é usada para definir a linha conjunto retornado. O provedor string contém o servidor de e Trusted_Connection palavras-chave. Essas palavras-chave são reconhecidas pela SQL Nativo Cliente fornecedor OLE DB.

Nota: Você precisa renomear o nome do servidor utilizado na presente consulta para um suplente para executar esta consulta com êxito.

```
SELECT A. *
FROM OPENROWSET ( 'SQLNCLI', 'Servidor = GEN-DEV; Trusted_Connection = sim;',
"SELECT groupname, Nome, DepartmentID
DESDE AdventureWorks.HumanResources.Department
ORDER BY groupname, Nome ' ) como um;
```

O seguinte é o resultado parcial conjunto da consulta.

Groupname	Name	DepartmentID
Executivo e da Administração Geral	Executivo	16
Executivo e da Administração Geral	Instalações e Manutenção	14
Executivo e da Administração Geral	Finanças	10
Executivo e da Administração Geral	Recursos Humanos	9
Executivo e da Administração Geral	Serviços de Informação	11

Usando INNER JOIN OPENROWSET

O código a seguir exemplo seleciona todos os dados da tabela Clientes locais a partir da instância do SQL Server Northwind banco de dados e as Ordens de mesa a partir do banco de dados Microsoft Access Northwind armazenados no mesmo computador.

```
SELECT o.OrderID, o.OrderDate, o.CustomerID, c.CompanyName, c.ContactName
```

A partir de Northwind.dbo.Customers como c
 INNER JOIN OPENROWSET ('Microsoft.Jet.OLEDB.4.0', 'C: \ Arquivos de programas \ Microsoft Office
 \ OFFICE11 \ AMOSTRAS \ Northwind.mdb ";" admin';", Encomendas)
 Como O
 ON c.CustomerID = o.CustomerID

O seguinte é o resultado parcial conjunto da consulta.

OrderID	OrderDate	CustomerID	COMPANYNAME	ContactName
10248	1996-07-04 00:00:00.000	VINET	Vins et alcools Chevalier	Paul Henriot
10249	1996-07-05 00:00:00.000	TOMSP	Toms Spezialitäten	Karin JOSEPHS
10250	1996-07-08 00:00:00.000	HANAR	Hanari Carnes	Mario Pontes
10251	1996-07-08 00:00:00.000	VICTE	Victuailles en stock	Mary Saveley
10252	1996-07-09 00:00:00.000	SUPRD	Suprêmes délices	Pascale artrain

Criando um servidor ligado

Esta demonstração mostra como você pode criar um servidor ligado que aponta para um banco de dados do Microsoft Office Access. Alguns dos dados de vendas que você use frequentemente está presente em um banco de dados do Microsoft Office Access em um servidor remoto. Por isso, é necessário criar um servidor ligado que aponta para o Microsoft Office Access banco de dados e facilitar aos usuários o acesso a dados usando um nome de usuário e senha.

Você precisa criar um servidor ligado que aponta para um banco de dados do Microsoft Office Access, no qual os dados de vendas que você use frequentemente está disponível. Para fazer isso, primeiro abra o Microsoft SQL Server Management Studio. Após verificar que o tipo Server, servidor de nome, e as caixas Autenticação ter sido preenchido corretamente, conectar ao servidor. Para criar uma nova consulta, na barra de ferramentas, clique em Nova Consulta. Em seguida, ligue para o banco de dados AdventureWorks.

Tal como o primeiro passo para a criação das NorthWind ligadas servidor, o que você precisa para executar a pesquisa que aponta para o Office Access banco de dados. Forneça os detalhes sobre o nome do servidor para criar ligado, o identificador único programáticas (PROGID) do fornecedor OLE DB correspondente a esta fonte de dados, o nome do produto do OLE DB como fonte de dados para adicionar um servidor associado, bem como o nome da fonte de dados, tal como foi interpretado pelo fornecedor OLE DB. Para criar um servidor ligado Northwind.mdb que aponta para o banco de dados, na janela de busca, selecione a declaração e executar a consulta.

Quando um usuário em logs para o servidor local e executa uma query distribuída que acessa uma tabela com o NorthWind ligadas servidor, o servidor local deverá registrar-se ligado ao servidor com o lado cliente para aceder a esse quadro. Você pode usar o procedimento armazenado sp_addlinkedserver para especificar as credenciais de login que utiliza o servidor local para registrar a ligados ao servidor. Você pode criar um mapa para facilitar o nome de usuário ea senha de entrada de usuários. Executa a consulta. Uma ligada servidor foi configurado para o seu banco de dados Access.

Em seguida, você precisa criar uma ligada para se conectar a um servidor do Microsoft Office Excel planilha chamado custmap.xls. Para fazer isso, você pode executar uma consulta através da utilização dos sp_addlinkedserver procedimento armazenado. Em seguida, você precisa alterar o valor do "Connection Timeout" parâmetro de um valor padrão de 0 a um novo valor de 5 para o ExcelSource ligadas servidor. Agora, você pode testar a conexão ao executar uma consulta contra o ExcelSource ligadas servidor. Executa a consulta. Um servidor ligado para o seu Gabinete está configurado

planilha Excel. Se o servidor ligado não foi criado corretamente, uma mensagem de erro será exibida.

Vinculados servidores oferecem muitas vantagens. Eles fornecem-lhe acesso ao servidor remoto, e à capacidade de emissão distribuídas consultas, atualizações, comandos, e heterogêneas fontes de dados sobre transações em toda a empresa. Além disso, proporcionam-lhe a capacidade de abordar diversas fontes de dados do mesmo modo.

Em Object Explorer, navegue para a pasta da Relacionada Servidores Server Objects pasta. Clique com o botão direito Relacionada Servidores e, em seguida, clique em Novo Servidor Vinculado. A caixa de diálogo Novo Servidor Vinculado aparece. Relacionada no servidor da caixa Geral página, digite o nome do servidor SQL remoto. Verifique se a fonte de dados Outra opção tiver sido selecionada.

No Provider lista, clique em SQL Nativo Cliente. Na caixa Nome do produto, tipo sqlcliin. A fonte de dados na caixa de tipo GEN-DEV. Na página Segurança, clique ser feita utilizando o login do actual contexto de segurança. Agora, você vai utilizar o procedimento armazenado sp_catalogs para obter uma lista dos catálogos específicos ligados a um servidor chamado LONDON2. Executa a consulta.

Você vai agora criar uma consulta para obter informações sobre os quadros que estão contidas no HumanResources esquema AdventureWorks na base de dados sobre o London2 ligadas servidor. Executa a consulta. Em seguida, você criará uma consulta que retorna o tipo de dados a coluna denominada título na tabela HumanResources.Employee os AdventureWorks na base de dados sobre o servidor ligado, LONDON2. Executa a consulta.

Esta demonstração mostra-lhe como criar um servidor ligado que aponta para um banco de dados do Microsoft Office Access.

Criando um servidor ligado

No menu Iniciar, aponte para Todos os programas e, em seguida, aponte para Microsoft SQL Server 2005 e, em seguida, clique em SQL Server Management Studio. O SQL Server Management Studio janela aparece.

Na caixa de diálogo Conectar ao servidor, verifique se o tipo Server, servidor de nome, e as caixas Autenticação ter sido preenchido corretamente e, em seguida, clique em Connect para aceitar as configurações padrão.

Na barra de ferramentas do SQL Server Management Studio, clique em Nova Consulta.

Na lista Bancos de dados disponíveis sobre a barra de ferramentas, clique AdventureWorks. Está ligado ao banco de dados AdventureWorks.

Na janela de busca, digite as seguintes afirmações. EXEC sp_addlinkedserver @ server = 'NorthWind', @ provedor = 'Microsoft.Jet.OLEDB.4.0', @ srvproduct = 'para o Jet OLE DB Provider ', @ datasrc = 'C: \ Arquivos de programas \ Microsoft Office \ OFFICE11 \ AMOSTRAS \ Northwind.mdb '

Selecione a declaração e, em seguida, na barra de ferramentas, clique em Executar.

Na janela de busca, digite as seguintes afirmações. EXEC sp_addlinkedsrvlogin 'NorthWind', 'falso', 'GEN-DEV \ Estudante ', 'Estudante ', 'Pa \$ \$ w0rd '

Em seguida, selecione a declaração e, na barra de ferramentas, clique em Executar.

Na janela de busca, digite as seguintes afirmações. EXEC sp_addlinkedserver 'ExcelSource ', 'Jet 4.0', 'Microsoft.Jet.OLEDB.4.0', 'c: \ Temp \ CustMap.xls', NULL, 'Excel 5.0 '

Na janela de busca, digite as seguintes afirmações. EXEC sp_serveroption 'ExcelSource', 'conectar Timeout', 5

Na janela de busca, digite as seguintes afirmações. EXEC sp_testlinkedserver ExcelSource

Em seguida, selecione a declaração e, na barra de ferramentas, clique em Executar.

Em Object Explorer, navegue para a pasta da Relacionada Servidores Server Objects pasta. Clique com o botão direito Relacionada Servidores e, em seguida, clique em Novo Servidor Vinculado. A caixa de diálogo Novo Servidor Vinculado aparece.

Relacionada no servidor da caixa Geral página, tipo LONDON2.

Outra fonte de dados verificar se foi selecionado, e no Provider lista, clique em SQL Nativo Cliente.

Na caixa Nome do produto, tipo sqlcliin.

A fonte de dados na caixa de tipo GEN-DEV.

Na página Segurança, clique ser feita utilizando o login do actual contexto de segurança. Clique em OK.

Na janela de busca, digite as seguintes afirmações. EXEC sp_catalogs' London2 '

Em seguida, selecione a declaração e, na barra de ferramentas, clique em Executar.

Na janela de busca, digite as seguintes afirmações. EXEC sp_tables_ex @ table_server = "LONDON2", table_catalog @ = 'AdventureWorks', @ table_schema = ' HumanResources ', @ table_type = 'TABELA'

Em seguida, selecione a declaração e, na barra de ferramentas, clique em Executar.

Na janela de busca, digite as seguintes afirmações. EXEC sp_columns_ex 'London2', 'trabalhador assalariado', 'HumanResources', ' AdventureWorks', 'Título'

Em seguida, selecione a declaração e, na barra de ferramentas, clique em Executar.

Como escrever consultas distribuídas baseadas em servidores ligados

Você pode escrever servidor ligado à base de consultas distribuídas através da utilização de uma parte totalmente qualificado quatro-o nome ea função OPENQUERY. Uma parte totalmente qualificado de quatro nome é usado para se referir a dados objetos em um servidor ligado. A função OPENQUERY executa a consulta sobre o servidor especificado associado.

Fully Qualified Four-Part Name

Depois de um servidor está ligado definida, você pode usar um nome ao lado de quatro referência os dados em que objetos ligados servidor.

A seguinte é a sintaxe parcial da utilização de uma parte totalmente qualificado, de quatro em nome T-SQL declarações.

`linked_server_name.catalog.schema.object_name`

linked_server_name. O servidor que as referências associadas a OLE DB fonte de dados.

catálogo. O catálogo, no OLE DB fonte de dados que contém o objeto.

schema. O esquema, no catálogo que contém o objeto.

object_name. Os dados objeto no esquema.

O SQL Server utiliza linked_server_name para identificar o fornecedor OLE DB e fonte de dados. O catálogo, esquema, e object_name parâmetros são passados para o fornecedor OLE DB dados específicos para identificar um objeto. O servidor refere-se associada uma instância de SQL Server 2005, catálogo refere-se a um banco de dados, e remete para o esquema do banco de dados esquema.

Você deve utilizar plenamente qualificado nomes quando estiver a trabalhar com objectos ligados a servidores. Não existe nenhum apoio para a resolução implícita a DBO proprietário em nome associado aos quadros de servidores. Por isso, uma consulta sem um esquema nome gera um erro, quando o servidor está ligado outra instância do SQL Server.

Você pode acessar dados remotos através da utilização associada servidores. É mais do que usar o relacional OPENROWSET, OPENDATASOURCE, ou OPENQUERY funções.

Funções OPENQUERY

A função OPENQUERY executa a consulta passou a ele sobre o servidor especificado associado. Esta função pode ser referenciada na cláusula FROM de uma consulta apenas como uma tabela nome. OPENQUERY também pode ser referenciado como alvo o quadro de um INSERT, UPDATE, DELETE ou declaração em função das capacidades do OLE DB do fornecedor. Embora a consulta pode retornar múltiplos conjuntos de resultados, OPENQUERY retorna apenas o primeiro conjunto de resultados.

OPENQUERY (linked_server, 'query')

linked_server. O identificador que representa o nome do servidor ligado.

consulta. A seqüência de consulta que é executado no servidor ligado.

Exemplos:

O código a seguir mostra como um exemplo ligado servidor é criado e pode ser referenciado usando seu nome de quatro lado. Neste exemplo, é um ExcelSource ligadas servidor que aponta para um arquivo chamado Microsoft Office Excel CUSTMAP.xls.

```
SELECT * FROM ExcelSource ... CustMap $
```

Nota: Para esta consulta para o trabalho, um arquivo chamado Microsoft Office Excel CUSTMAP.xls deveria existir na c: / / temp diretório e um servidor ligado ExcelSource já deve existir.

O seguinte é o resultado parcial conjunto da consulta.

ProductID	nome	ProductName	MakeFlag	FinishedGoodsFlag
1	Ajustável Race	AR-5381	0	0
2	Ball Bearing	BA-8327	0	0
3	BB Ball Bearing	BE-2349	1	0
4	Auricular	BE-2908	0	0

	rolamentos			
316	Blade	BL-2036	1	0

O código a seguir exemplo cria um servidor ligado a um banco de dados chamado Microsoft Office Access Northwind e, em seguida, utiliza uma consulta SELECT para recuperar dados a partir dele. Neste exemplo, não há necessidade de quatro-parte nome, mas você pode usar o namespace.

USE AdventureWorks;

GO

EXEC sp_addlinkedserver

@ servidor = 'NorthWind', @ provedor = 'Microsoft.Jet.OLEDB.4.0', @ srvproduct = 'OLE DB Provider para o Jet ', @ datasrc = ' C: \ Arquivos de programas \ Microsoft Office \ OFFICE11 \ AMOSTRAS \ Northwind. mdb 'SELECT o.OrderID, o.OrderDate, c.CompanyName
DESDE o Northwind.dbo.Orders
Northwind.dbo.Customers INNER JOIN c ON c.CustomerID = o.CustomerID

O seguinte é o resultado parcial conjunto da consulta.

OrderID	OrderDate	COMPANYNAME
10248	1996-07-04 00:00:00.000	Vins et alcools Chevalier
10249	1996-07-05 00:00:00.000	Toms Spezialitäten
10250	1996-07-08 00:00:00.000	Hanari Carnes
10251	1996-07-08 00:00:00.000	Victuailles en stock
10252	1996-07-09 00:00:00.000	Suprêmes délices

Usando a função OPENQUERY

O código a seguir exemplo cria um servidor ligado chamado OracleSvr contra um banco de dados Oracle, utilizando o Microsoft OLE DB Provider para Oracle. Então, este exemplo executa uma simples consulta SELECT contra esta ligada servidor.

Nota: Este exemplo assume que um banco de dados Oracle alias chamado ORCLDB foi criada.

EXEC sp_addlinkedserver 'OracleSvr',

"Oracle 7,3 ",

'MSDAORA',

'ORCLDB'

SELECT * FROM OPENQUERY (OracleSvr, "SELECT nome, ID de joe.titles")

Teste

Cada azulejo contém uma verdadeira declaração e de uma falsa declaração. Clique no azulejos em uma linha ou coluna até que cada telha mostra a verdadeira declaração e, em seguida, clique na seta ao lado da linha ou superior da coluna para verificar a sua resposta. Você usa um turno quando você clicar Enviar uma incorrecta ou azulejo. Remover todas as peças antes de utilizar todas as suas voltas.

Usando Diferentes Tecnicas Quando se Trabalha com Consultas Complexas

Às vezes, quando há necessidades empresariais complexas, pode ser necessário escrever queries complexas. Você pode simplificar consultas complexas através da utilização de várias técnicas em SQL Server. Você pode definir o modo implícito para a operação, a fim de evitar apagamento acidental de dados. Você pode usar tabelas auxiliares para simplificar buscas. Você também pode

explorar as diferentes possibilidades para escrever uma consulta SQL Server porque usa um otimizador baseado em termos de custos.

Como usar Transações Implícitas

Você pode configurar SQL Server para iniciar transações implicitamente. Se você ativar o modo implícito transação, então todas as declarações são consideradas como parte de uma transação. As operações não são autorizadas até um comando a transação é emitido. O SQL Server automaticamente inicia uma nova operação após a transação atual é cometido ou laminado de volta. Implícito Modo de operação gera uma cadeia de operações contínuas. Se você está trabalhando em um ambiente crítico, você pode usar para prevenir qualquer transação implícita acidental alteração ou supressão de dados. Uma vez que todas as operações são automaticamente armazenadas em uma transação implícita, você pode voltar acidental mudanças em um dado momento no passado, indicando um comando repetir a transação.

Quando o modo implícito operação foi fixado para a ligação para um ou para um servidor, SQL Server automaticamente inicia uma transação no momento em que primeiro executa qualquer uma das seguintes afirmações.

Declarações

ALTER TABLE

SELECT

UPDATE

DELETE

INSERT

TRUNCATE TABLE

CREATE

OPEN

REVOKE

DROP

Fetch

GRANT

Depois de uma transação é cometido ou laminado de volta, a exemplo da Base Engine automaticamente inicia uma nova transação quando se executa qualquer uma das afirmações acima especificadas. Se você não explicitamente cometer uma transação implícita, então a operação e de todas as mudanças que ele contém dados são lançados automaticamente quando você voltar desligar.

Para iniciar operações implícita, é necessário especificar o seguinte comando.

[SET IMPLICIT_TRANSACTIONS](#)

Implícito Modo de operação continua em vigor até você explicitamente defini-lo para fora, especificando o seguinte comando.

[SET IMPLICIT_TRANSACTIONS off](#)

Há algumas desvantagens da utilização de transações implícita.

Desvantagens

Você não tem controle sobre quando iniciar uma transação. Então se você quiser executar algumas declarações fora do contexto de uma transação, então você precisa explicitamente definir o modo para fora.

Tal como transações implícitas são iniciadas automaticamente, às vezes você pode se esqueça que você é, no contexto de uma transação e isso pode levar ao bloqueio questões.

Você precisa fazer um commit expressamente em transações implícitas. Se você esquecer o commit de uma operação, em seguida, o trabalho feito vai desfeito automaticamente e perdido.

Como usar tabelas auxiliares

Acessórias ou auxiliares tabelas são ajudantes que você pode criar tabelas no banco de dados para manter os dados que não estão diretamente relevantes para a aplicação. No entanto, pode ser necessário este apoio dados a fim de simplificar o processo de gestão de determinados cenários complexos. Sem estes quadros auxiliares, você pode ter que usar lógica processual complexo que envolve longos loops que poderiam resultar em desempenho de grande intensidade de buscas. Auxiliares tabelas são similares às tabelas temporárias, porque eles possuem dados que não é exigida no banco de dados. No entanto, mesas auxiliares não são armazenadas no tempdb. Ao contrário tabelas temporárias, tabelas auxiliares são permanentemente armazenadas no banco de dados e não estão caiu quando fecha a conexão. Portanto, você não precisa recriar tabelas auxiliares cada vez SQL Server recomeça. Algumas das mais utilizadas tipos de tabelas são auxiliares calendário tabelas e tabelas inteiro.

Tabela Calendário

Você pode usar uma tabela calendário para simplificar a lógica de uma consulta ao mesmo tempo manipulação data valores. Você também pode criá-lo de tal forma que ele possa lidar com certas condições que o espaço construído, em data não pode manipular funções.

Por exemplo, quando a determinação do número de dias entre dois eventos, pode ser necessário excluir feriados. O SQL Server não suporta qualquer função de indicar férias porque feriados variam em cada país, estado, e até mesmo município. Além de férias, você pode querer tratar, quer de origem humana ou ocorrências de catástrofes naturais como eventos especiais. Alternativamente, se você quiser ignorar uma data no calendário por qualquer outro motivo, SQL Server não oferece suporte a essa exigência. Em tais situações, você pode usar um calendário quadro, a apresentar um relatório dias em que não existe qualquer atividade registrada na base de dados.

O seguinte exemplo ilustra o uso de uma tabela calendário auxiliar.

Se você quiser que o cálculo do número de dias úteis entre duas datas na sua candidatura, não há espaço construído, em função SQL Server que suporta esta consulta. Assim, você pode criar uma tabela auxiliar para isso.

```
CREATE TABLE Calendar (calDate datetime NOT NULL PRIMARY KEY, calWeekEnd BIT NOT NULL)
```

Você pode então popular a tabela com os valores. Na primeira coluna, o que você precisa para armazenar a data e na segunda coluna, o que você precisa para armazenar o valor 1 se for uma semana e 0 se for um feriado. Você pode usar o seguinte código para preencher o quadro.

```
SET DATEFIRST 7
SET NOCOUNT ON
DECLARAM d @ AS datetime
SET @ d ='20000101 '
BEGIN TRAN
ENQUANTO @ d <='20091231 '
BEGIN
Inserir no calendário
VALUES (@ d,
Caso quando DATEPART (dias úteis, @ d)
IN (1,7) então OUTROS 1 FIM 0)
SET @ @ d = d + 1
FIM
COMMIT TRAN
```

Em seguida, você pode listar todos os dias úteis em 2006 a partir da tabela, usando a seguinte função.

```
SELECT COUNT (*) FROM agenda WHERE calWeekEnd = 0 e datepart (yy, calDate) = 2006
```


Tabela Inteiro (Integer)

Um inteiro auxiliares tabela pode ser usado para várias tarefas como procurar lacunas em termos de identidade valores, gerando um calendário mesa auxiliar ou gerar períodos. Você pode executar estas tarefas, sem uso de tabelas auxiliares também, mas essas consultas será mais complexas e ineficientes. O seguinte exemplo ilustra o uso de uma tabela auxiliar inteiro.

Você pode criar uma tabela chamada auxiliares Números que tem apenas uma coluna chamada Número.

```
CREATE TABLE dbo.Numbers (Número IDENTIDADE INT (1,1) PRIMARY KEY agrupado)
```

Você pode usar o código a seguir para inserir valores na tabela.

```
WHILE Coalesce (SCOPE_IDENTITY (), 0) <= 1024  
BEGIN  
Dbo.Numbers insert default values  
END
```

Agora, você pode usar esta tabela para diferentes fins. O exemplo a seguir mostra a forma como a tabela foi usada para gerar dias entre duas datas.

```
DECLARE @ sDate SMALLDATETIME, @ eDate SMALLDATETIME  
SET @ sDate ='20040101' SET @ eDate ='20040110'  
SELECT @ sDate + number da dbo.Numbers  
WHERE @ sDate + Number <= 1 ORDER BY @ eDate
```

Quebrando uma Consulta Complexa em Diferentes Etapas

Você é o DBA Junior da Adventure Works. O gerente de Marketing da sua organização pede para preparar uma lista de todos os clientes cuja media de pedido é de 50.000 ou mais, e que tem mais de 5 pedidos. Para isso você tem que escrever uma consulta complexa em pequenos pedaços ao identificar varias operações relacionais. Você vai até a DBA Sênior, Nancy, e pede ajuda para preparar a lista.

Ela então concorda em ajudar.

Você explica a Nancy que a saída da consulta inclui as seguintes colunas:

CustomerID

Nome da Loja

Valor da media dos pedidos

[Numero de pedidos]

SalesOrderID

Data do pedido dos três últimos meses

Contudo você não está certo sobre a primeira etapa para escrever e pede um conselho.

A Nancy explica que você precisa criar uma tabela UDF, na qual aceitará a CustomerID como entrada e retornará CustomerID, Nome da Loja, Valor da media dos pedidos, Numero de pedidos, SalesOrderID e Data do pedido dos três últimos meses. A seguir a declaração CREATE FUNCTION.

Use AdventureWorks;

```
GO CREATE FUNCTION fn_GetOrders (@custid AS INT) RETURNS TABLE AS RETURN
```

A seguir as clausulas FROM e WHERE:

```
FROM Sales.SalesOrderHeader
```

```
WHERE CustomerID = @custid
```

Nancy conclui que você precisa preencher os nomes das colunas na clausula SELECT. Qual você usa?

```
SELECT SalesOrderID, OrderDate
```

Nancy diz que você cometeu um erro. E dá a seguinte sugestão:

“Na declaração SELECT, especifique que você precisa apenas de três itens de dados. Você precisa dar essa condição usando o operador TOP. Mude a consulta de acordo”

```
SELECT TOP (3) SalesOrderID, OrderDate
```

Você pergunta a Nancy se a consulta está pronta para rodar. Ela diz que você precisa ordenar o resultado para pegar os três pedidos mais recentes , usando o ORDER BY na consulta.

Como o ORDER BY vai operar?

```
ORDER BY SalesOrderID DESC
```

Nancy diz que você cometeu um erro. E explica que você precisa dos três últimos pedidos, então tem que ordenar o resultado por OrderDate (Data do pedido). E pede para você pensar um pouco.

```
ORDER BY OrderDate DESC
```

Nancy mostra a consulta completa.

Use AdventureWorks;

```
GO CREATE FUNCTION fn_GetOrders (@custid AS INT) RETURNS TABLE AS RETURN
```

```
SELECT TOP (3) SalesOrderID, OrderDate
```

```
FROM Sales.SalesOrderHeader
```

```
WHERE CustomerID = @custid
```

```
ORDER BY OrderDate DESC
```

Você roda a consulta. O resultado indica que o comando foi executado com sucesso. Nancy pede a você o tipo de consulta para testar a função com um determinado cliente.

Como vai ser a sua declaração?

```
SELECT * FROM fn_GetOrders
```

Nancy explica que quando você quer testar a função com um determinado cliente é preciso especificar a CustomerId (o código), senão uma mensagem de erro vai aparecer.

Então você muda a declaração de acordo.

```
SELECT * FROM fn_GetOrders (4)
```

Você obtém o resultado. A Nancy explica que a próxima etapa é criar uma tabela derivada não qual dará a sumarização de cada cliente. Esta tabela deve ter as seguintes colunas: CustomerID, Media do Total de Contas, Numero de Pedidos. Ela pede pelo tipo de SELECT e então escreve o restante da consulta.

```
FROM Sales, SalesOrderHeader AS soh
```

```
GROUP BY CustomerID
```

Qual SELECT você faz?

```
SELECT CustomerID, AVG (TotalDue), COUNT (*)
```

Você mostra a consulta completa:

```
SELECT CustomerID, AVG (TotalDue), COUNT (*)
```

```
FROM Sales, SalesOrderHeader AS soh
```

```
GROUP BY CustomerID
```

E roda a consulta. Você descobre que duas colunas não têm nome.

A Nancy lembra que você precisa especificá-los usando a clausula AS. Qual é o correto?

```
SELECT CustomerID, AVG (TotalDue) AS AvgTotalDue, COUNT (*) AS NumOrders
```

Você mostra a consulta à Nancy e ela segue que você faça um join da tabela Sales.Customer com a tabela derivada criada. Então junte o resultado com a tabela Sales.Store para restringir e trazer somente as linhas por lojas e não por clientes. Ela escreve parte da consulta onde você colocará a cláusula WHERE.

```
SELECT c. CustomerID, s.Name, d.AvgTotalDue, d.NumOrders FROM Sales.Customer AS c JOIN  
(SELECT CustomerID, AVG(ToatlDue) AS AvgTotalDue, COUNT(*) AS NumOrders  
FROM Sales,SalesOrderHeader AS soh  
GROUP BY CustomerID) AS d ON c.CustomerID = d.CustomerID  
JOIN Sales.Store s ON c.CustomerID = s.CustomerID
```

Você mostra a consulta completa para a Nancy e ela sugere: “Junte a tabela Sales. Customer e a tabela derivada que você criou anteriormente. Então junte o resultado com a tabela Sales.Store para trazer somente as linhas por loja e não por cliente. Vou escrever uma parte e você escreve a cláusula where.”

```
SELECT c.CustomerID, s.Name, d.AvgTotalDue, d.NumOrders FROM Sales.Customers AS c JOIN (SELECT  
CustomerID, AVG(totalDue) AS AvgTotalDue, COUNT(*) AS NumOrders FROM Sales.SalesOrderHeader  
soh GROUP BY CustomerID) AS d ON c.CustomerID = d.CustomerID JOIN Sales.Store s ON  
c.CustomerID = s.CustomerID
```

Qual cláusula WHERE é a correta?

```
WHERE AvgTotalDue > $50000
```

Nancy menciona que você esqueceu a condição do número de pedidos e pede para colocá-la.

```
WHERE NumOrders > 5
```

Nancy menciona que você se esqueceu da condição da média do valor total e pede para colocá-la.

```
WHERE AvgTotalDue > $50000 AND NumOrders > 5
```

Nancy diz que a consulta está correta e mostra a consulta completa e o resultado.

```
SELECT c.CustomerID, s.Name, d.AvgTotalDue, d.NumOrders FROM Sales.Customers AS c JOIN (SELECT  
CustomerID, AVG(totalDue) AS AvgTotalDue, COUNT(*) AS NumOrders FROM Sales.SalesOrderHeader  
soh GROUP BY CustomerID) AS d ON c.CustomerID = d.CustomerID JOIN Sales.Store s ON  
c.CustomerID = s.CustomerID  
WHERE AvgTotalDue > $50000 AND NumOrders > 5
```

Nesta situação, qual condição de JOIN você vai usar para juntar os dois resultados?

```
UNION ALL fn getOrders(c.CustomerID) AS o
```

Você ainda pergunta se fez a consulta corretamente, Nancy explica que você não deve usar o UNION ALL para o propósito de juntar porque resultará um erro. Nancy pede para você usar o CROSS APPLY para obter os três pedidos mais recentes para cada loja. Então você reescreve.

```
JOIN fn getOrders(c.CustomerID) AS o
```

Você pergunta se está certo e a Nancy diz que você não deve usar o JOIN porque resultará em um erro e pede para você reescrever usando CROSS APPLY para obter os três mais recentes pedidos por loja.

```
CROSS APPLY fn getOrders(c.CustomerID) AS o
```

Nancy diz que está correto e mostra a consulta completa:

```
SELECT c.CustomerID, s.Name, d.AvgTotalDue, d.NumOrders FROM Sales.Customers AS c JOIN (SELECT  
CustomerID, AVG(totalDue) AS AvgTotalDue, COUNT(*) AS NumOrders FROM Sales.SalesOrderHeader  
soh GROUP BY CustomerID) AS d ON c.CustomerID = d.CustomerID JOIN Sales.Store s ON  
c.CustomerID = s.CustomerID  
CROSS APPLY fn getOrders(c.CustomerID) AS o  
WHERE AvgTotalDue > $50000 AND NumOrders > 5
```

E então você roda a consulta.

Recomendações para selecionar técnicas de consulta apropriadas

O SQL Server usa um otimizador baseado em termos de custos, que podem ser extremamente sensíveis a informação da estatística que é fornecida sobre tabelas, escombreyras, e índices. O SQL Server utiliza esse tipo de informação estatística para determinar o nível ótimo do plano de execução de uma consulta. Para resolver um problema particular de negócios, você pode reescrever a mesma consulta, em muitos aspectos. Você precisa testar as consultas através da utilização de várias técnicas para determinar qual consulta proporciona melhor desempenho para um determinado problema.

Nos exemplos a seguir, o resultado desejado é definir uma lista dos produtos na tabela de produtos que não tenham sido vendidos. Para fazer isso, você pode fazer uma pesquisa usando a palavra-chave IN, a cláusula JOIN, ou a palavra-chave NOT EXISTS. Ao comparar a execução destes planos de consultas, você pode determinar qual opção é a melhor.

IN

Você pode definir uma subconsulta na qual o interior consulta lista todos os produtos, na tabela SalesOrderDetail. A consulta exterior então enumera os produtos que não estão presentes na lista retornada pela consulta interna. A seguinte consulta mostra este método.

```
USE AdventureWorks;
SELECT p.ProductID, p.Name
FROM Production.Product p
Productid onde não nos
(SELECT distintas productid
DESDE sales.salesorderdetail)
```

JOIN

Você pode participar do Produto e do quadro SalesOrderDetail tabela baseada na coluna ProductID. Para listar os produtos que não estão presentes no SalesOrderDetail tabela, você pode determinar as linhas que retornar valores nulos SalesOrderDetail a partir da tabela. Para fazer isso, você pode incluir a cláusula WHERE para selecionar apenas as linhas que retornar NULL valores. A seguinte consulta demonstra este método.

```
USE AdventureWorks;
SELECT p.ProductID, p.Name
FROM Production.Product p
LEFT JOIN Sales.SalesOrderDetail SOD ON p.ProductId = sod.ProductID
Sod.ProductID onde é nula
```

NOT EXISTS

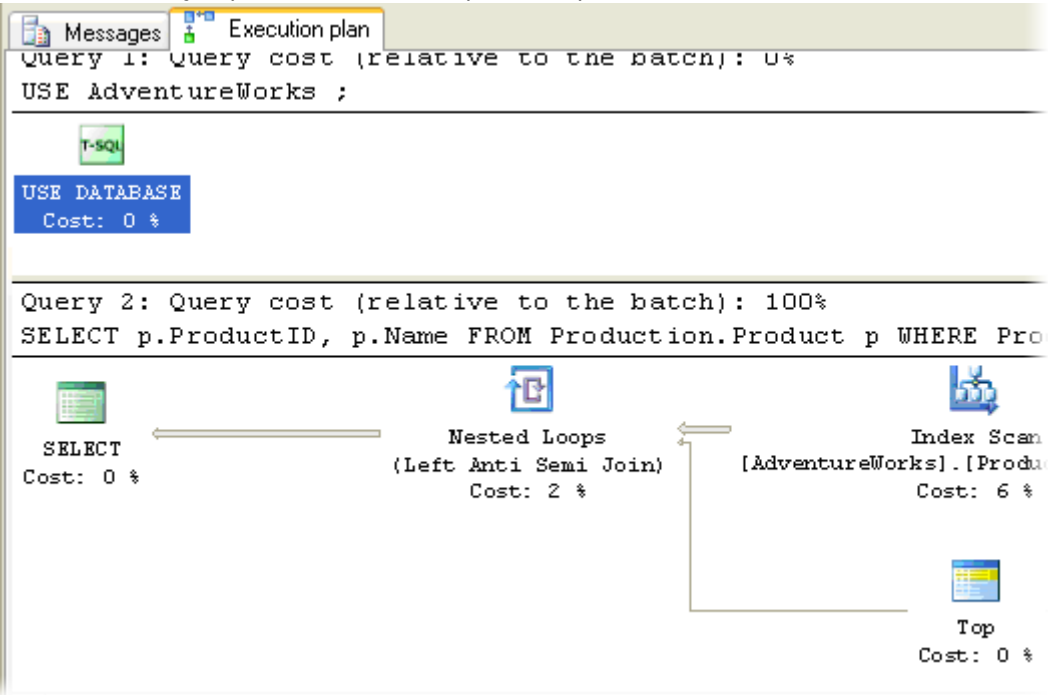
Você também pode usar a palavra-chave NOT EXISTS para retornar uma lista de produtos que não foram vendidos. Na seqüência da consulta, quando uma linha é retornada a partir da tabela SalesOrderDetail pelo interior de uma consulta particular ProductID, a fila é excluída do exterior consulta.

```
SELECT p.ProductID, p.Name
FROM Production.Product p
WHERE NOT EXISTS
(SELECT distintas productid
FROM sales.salesorderdetail SOD
WHERE sod.ProductID = p.ProductID)
```

A comparação das Consultas

Se você comparar a execução dos planos de todas as três consultas, a consulta envolvendo a cláusula JOIN executa lentamente quando comparado com as outras duas alternativas. Isto porque, no caso

de uma cláusula JOIN, todas as linhas a partir da tabela SalesOrderDetail precisam ser avaliadas. Portanto, você pode usar o IN ou a palavra-chave EXISTS para executar a consulta.
Plano de Execução para uma consulta que usa a palavra-chave IN



Plano de Execução para uma consulta que usa a clausula JOIN

Usando Técnicas de Consultas Avançadas no Microsoft SQL Server

Transact -SQL (T-SQL) é uma linguagem baseada em conjunto, de forma que geralmente há mais de uma maneira em que você pode escrever uma consulta. Mas, em muitos casos, haverá uma técnica que terá desempenho melhor que as outras. Você precisa para determinar o melhor método para executar uma consulta com os diferentes métodos de teste que estão disponíveis. Quando você executar consultas avançadas, você precisa saber como lidar com complexidades, como dados multinacionais de data / hora. Você também precisa manter controlar e gerenciar os arquivos fonte de consulta para usar a versão de software de gestão, tais como Microsoft Visual SourceSafe.

Recomendações para consultar dados complexos

Quando você trabalha com dados de data / hora, pode haver problemas, porque eles são sempre armazenados em conjunto no SQL Server. Por isso, você precisa estar ciente de algumas recomendações e ao mesmo tempo selecionar ou inserir dados data / hora. Você também precisa estar ciente das recomendações enquanto trabalha com dados multinacionais, como Unicode. Além disso, você precisa conhecer a ordem de precedência dos tipos de dados.

Recomendações para consultar dados Datetime

No SQL Server existe dois tipos de dados data e hora construídos internamente para armazenar dados, ou seja, datetime e smalldatetime. Ambos os tipos de dados armazenam componentes de data e a hora. Não existe qualquer tipo de dados SQL Server que pode armazenar somente a data ou apenas a hora. Se você especificar a apenas parte data ao inserir os dados para a coluna datetime, em seguida, SQL Server armazena 00:00: 00.000 no parte hora. Se você especificar apenas a parte hora e, em seguida, SQL Server armazena 1900-01-01 contando a parte data. Devido ao fato de que os valores de data e hora serem sempre armazenados juntos, alguns problemas inesperados podem ocorrer.

A tabela a seguir resume as principais diferenças entre os tipos de dados datetime e smalldatetime.

Name	Minimum value	Maximum value	Accuracy	Storage
smalldatetime	1900-01-01 00:00:00	2079-06-06 23:59:00	minute	4 bytes

datetime 1753-01-01 00:00:00.000 9999-12-31 23:59:59.997 3.33 milliseconds (ms) 8 bytes

Para ilustrar os problemas que podem ocorrer quando o valor de data em consultas é armazenado com o valor hora, a seguinte tabela é criada com uma coluna datetime e preenchida com alguns valores.

```
Use AdventureWorks;
CREATE TABLE #DateTimeProblems(Column1 char(1), DateAndTime
datetime)
INSERT INTO #DateTimeProblems (Column1, DateAndTime)
VALUES('a', '20050305 09:12:59')
INSERT INTO #DateTimeProblems (Column1, DateAndTime)
VALUES('b', '20050305 16:03:12')
INSERT INTO #DateTimeProblems (Column1, DateAndTime)
VALUES('c', '20050306 00:00:00')
INSERT INTO #DateTimeProblems (Column1, DateAndTime)
VALUES('d', '20050306 02:41:32')
INSERT INTO #DateTimeProblems (Column1, DateAndTime)
VALUES('e', '20050315 11:45:17')
INSERT INTO #DateTimeProblems (Column1, DateAndTime)
VALUES('f', '20050412 09:12:59')
INSERT INTO #DateTimeProblems (Column1, DateAndTime)
VALUES('g', '20050523 11:43:25')
```

Cenário

No exemplo seguinte código que você precisa para busca de registros com a data valor «2005-03-05 'a partir da tabela temporária # DateTimeProblems.

```
SELECT Column1, DateAndTime
FROM #DateTimeProblems
WHERE DateAndTime ='20050305 '
```

A consulta não retornou nenhuma linha porque há dois tipos de dados diferentes na cláusula WHERE. O lado esquerdo do estado detém o tipo de dados datetime, enquanto do lado direito da condição detém a corda tipo de dados. Em tais situações, o SQL Server converte os tipos de dados baseados em sua ordem de precedência. O tipo de dados datetime tem precedência maior do que a corda tipo de dados. Por isso, o tipo de dados string é convertida para um primeiro tipo de dados datetime. O valor é datetime 2005-03-05 00:00:00. Como resultado, não há registros de que o valor datetime.

Quando resolver SELECT

Você pode reescrever a consulta através da conversão do datetime coluna para uma string usando a função CONVERT. Esta função terá um formato de código 112 que suprime a porção de tempo antes da data da conversão dele.

```
SELECT Column1, DateAndTime
FROM # DateTimeProblems
WHERE CONVERT (char (8), DateAndTime, 112) ='20050305 '
```

Esta consulta retorna um resultado. No entanto, esta não é a melhor solução. Porque é um cálculo realizado na consulta, SQL Server não pode utilizar um índice que poderia ser definida com a coluna para pesquisar na tabela. Isso poderia afetar o desempenho da consulta.

Você pode reescrever a consulta usando a palavra-chave ENTRE.

```
SELECT Column1, DateAndTime
FROM# DateTimeProblems
```

```
WHERE DateAndTime ENTRE'20050305 00:00:00.000 '  
23:59:59.999 E'20050305 '
```

Esta consulta não retornar a correta conjunto de resultados. Em SQL Server, a proposta de resolução para a datetime tipo de dados é 3,33 ms. Isto significa que 2005-03-05 23:59:59.999 será arredondada para 2005-03-06 00:00:00.000. Portanto, do conjunto de resultados contém uma linha adicional. Para obter o resultado correto, você precisa especificar a data a partir 20050305 23:59:59.997. Mas este tipo de consulta é muito embaraçoso para escrever.

Ao resolver INSERT

Você pode eliminar examinando questões em tabelas que armazenam a data valor com o tempo valor, inserindo apenas a data componente dos dados em uma tabela. Se a tempo parcial não é necessário em uma coluna especial, então você pode inserir os dados para a coluna, indicando que o tempo que compõem devem ser eliminados.

```
INSERT INTO # DateTimeProblems (Column1,  
DateAndTime) VALUES ('h','20050729 00:00:00 ')
```

No entanto, para essa solução para o trabalho, todos os aplicativos que inserir ou atualizar dados deve indicar explicitamente que o tempo parcial ser removido.

Você também pode criar uma função definida pelo usuário, que aceita como entrada um valor datetime e retorna o valor datetime após a remoção do tempo parcial.

```
CREATE FUNCTION dbo.ufnDatePart  
(@ FullDate datetime)  
CHANGE VARCHAR (10)  
AS  
BEGIN  
RETURN (CONVERT (VARCHAR (10), @ fullDate, 101))  
END
```

Você pode, então, definir um desencadeamento em cima da mesa para a inserção e atualizar as operações. Quando um valor datetime é inserido ou atualizado, o ufnDatePart função é invocada e os dados corretos é inserido na tabela.

```
USE tempdb;  
GO  
CREATE TABLE DateTimeProblems (Column1 char (1), DateAndTime datetime)  
GO  
CREATE TRIGGER trgupdate ON DateTimeProblems para INSERT, UPDATEAS  
SET NOCOUNT ON  
BEGIN  
IF UPDATE (DateAndTime)  
BEGIN  
DECLARE@ data datetime  
SELECT @ data = DateAndTime DA inseridos  
UPDATE DateTimeProblems conjunto DateAndTime = dbo.ufnDatePart (@ data)  
WHERE Column1 = (selecione inserted.Column1 DA inseridos)  
END
```

Para testar este Trigger, você pode executar a seguinte declaração INSERT. A declaração INSERT contém um componente hora no valor datetime.

```
INSERT INTO # DateTimeProblems (Column1,  
DateAndTime) VALUES ('a','20050305 09:12:59 ')
```

O trigger ufnDatePart invoca a função e atualiza a fila com um novo valor datetime que não contém o componente hora.

Recomendações para inserir dados datetime

Ao inserir valores datetime em tabelas ou pontos de vista, você precisa se certificar de que o formato e as configurações de idioma o datetime de tipos de dados estão corretos. É recomendado que você utilize linguagem independente de formatos, em vez de linguagem-dependente formatos porque são portáteis em toda línguas. Não é recomendado que você utilize linguagem-dependente formatos mesmo com SET declarações.

Por questões de formato

Várias questões relacionadas com formato pode surgir quando se trabalha com datetime objetos. Alguns usuários podem proporcionar contributos no dd / mm / aaaa formato, enquanto que o servidor pode interpretá-la como dd / mm / aaaa. Como resultado, o servidor pode dar um erro ou produzirá resultados incorretos.

Se uma consulta envolvendo valores datetime obras, não tem necessariamente retornar o resultado esperado. Se, por exemplo, SQL Server aceita valores na data dd / mm / aaaa para o formato HireDate coluna e você passar 5 de julho de 1997 como «5 / 7 / 1997" e, em seguida, SQL Server maio interpretá-la como 7 de maio de 1997.

Em tais situações, você pode usar o SET DATEFORMAT declaração a fim de especificar a contar da data de peças de valores datetime e smalldatetime. Para definir o formato da data como dd / mm / aaaa, o que você precisa para executar a seguinte declaração.

```
SET DATEFORMAT DMY
```

Por questões de linguagem

Ao converter o texto para o tipo de dados datetime tipo de dados, você precisa se assegurar que o idioma correto, foi fixado. Por exemplo, a seguinte consulta define o idioma para o E.U. Inglês e, depois, converte a string para um valor datetime.

```
SET LANGUAGE us_english  
SELECT CAST (2003-02-28 'AS datetime) AS  
DateAndTime
```

O seguinte é o resultado conjunto da consulta.

```
DateAndTime  
2003-02-28 00:00:00.000
```

A mesma consulta podem ser modificada para usar Britânico Inglês.

```
SET língua britânica  
SELECT CAST (2003-02-28 'AS datetime) AS  
DateAndTime
```

Esta consulta resulta em um erro porque a declaração SET LANGUAGE implicitamente define os parâmetros para o campo datetime e não foi formado de acordo com o formato que está definido para a língua britânica.

Você também pode definir SQL Server para o formato ISO. No formato ISO, os valores são representados como datetime AAAAMMDD [hh: mm: ss] ou de yyyy-mm-DDTHH: mm: ss. A norma

ISO dateformat é sempre garantida a trabalhar independentemente da língua ou DATEFORMAT configuração.

Exemplos:

A seguinte consulta atualiza a HireDate valor HumanResources.Employee na tabela.

```
USE AdventureWorks;  
UPDATE HumanResources.Employee  
DEFINE HireDate ='15 / 5 / 1997 "  
WHERE EmployeeID = 1
```

O seguinte é o resultado da consulta.

A conversão de um tipo de dados char datetime de um tipo de dados resultaram em um valor datetime fora do intervalo.

A declaração foi encerrada.

A mensagem de erro foi gerado porque SQL Server aceita a data de entrada em mm / dd / yyyy, e não no formato dd / mm / aaaa formato.

Para definir o formato da data e, depois, atualizar o valor HireDate, no quadro de funcionários, o que você precisa para executar a seguinte consulta.

```
SET DATEFORMAT DMY  
UPDATE HumanResources.Employee  
SET HireDate ='15 / 5 / 1997"  
WHERE EmployeeID = 1
```

Esta definição é utilizada apenas para a interpretação da personagem cordas porque eles são convertidos para valores data. Não afeta a data de exibição valores.

A consulta é executado com sucesso e o resultado é mostrado a seguir.

1 row (s) afetado.

Usando o conjunto idioma

A seguinte consulta define o idioma padrão para russo, exibe o mês atual nome, muda para E.U. Inglês e, em seguida, exibe o mês nome novamente.

```
@ Declarar hoje DATETIME  
SET @ Hoje ='12 / 1 / 2003 »
```

```
SET idioma russo  
SELECT DATENAME (mês, @ Hoje) AS 'Nome do Mês'
```

O seguinte é o resultado conjunto de completar a consulta.

```
MonthName  
Декабрь
```

As linhas seguintes são exibidas no painel resultados.

Mudaram a configuração de idioma русский.

(1 row (s) afetado)

Para alternar de volta para E.U. Inglês, você precisará usar a seguinte consulta.

```
@ Declarar hoje DATETIME  
SET @ Hoje ='12 / 1 / 2003 »  
SET LANGUAGE us_english
```

```
SELECT DATENAME (mês, @ Hoje) AS 'Nome do Mês'
```

O seguinte é o resultado conjunto de completar a consulta.

```
MonthName  
Janeiro
```

As linhas seguintes são exibidas no painel resultados.

```
Mudaram a configuração de idioma us_english.  
(1 row (s) afetado)
```

Aplicação da norma ISO Format

A seguinte consulta demonstra a aplicação da norma ISO formato para valores datetime.

```
SET idioma russo  
SELECT CAST ('20050515 'datetime AS) AS DateAndTime
```

O seguinte é o resultado conjunto de completar a consulta.

```
DateAndTime  
2005-05-15 00:00:00.000
```

As linhas seguintes são exibidas no painel resultados.

```
Mudaram a configuração de idioma русский.  
(1 row (s) afetado)
```

Para alternar de volta para E.U. Inglês, você precisa digitar a seguinte consulta.

```
SET LANGUAGE us_english  
SELECT CAST ('20050515 'datetime AS) AS DateAndTime
```

O seguinte é o resultado conjunto de completar a consulta.

```
DateAndTime  
2005-05-15 00:00:00.000
```

As linhas seguintes são exibidas no painel resultados.

```
Mudaram a configuração de idioma us_english.  
(1 row (s) afetado)
```

Recomendações para Trabalhar com dados multifuncionais

Collation configurações, que incluem a codificação, ordem de classificação, e outras configurações específicas de cada localidade, são fundamentais para a estrutura e função das bases de dados Microsoft SQL Server. Estas definições aplicam-se apenas a personagem dados. Colaciones especificar as regras sobre a forma como cordas de caráter dados são ordenados e comparados, baseada em particular as normas de idiomas e locais. No entanto, você também pode usar a collation independente tipo de dados, Unicode, para armazenar e acessar dados de caracteres. Em SQL Server 2005, collation seqüências podem ser definidos em três níveis diferentes: nível servidor, banco de dados nível, ou nível coluna.

Se você tem caráter dados armazenados em diferentes collation seqüências e, em seguida, comparando colunas podem ser difíceis e exigem considerações especiais codificação. Você pode resolver esse problema usando a cláusula COLLATE. Collation também pode causar problemas com o banco de dados tempdb. O tempdb base de dados é criado a cada vez que o servidor seja reiniciado e tem a mesma collation padrão seqüência como o modelo base de dados. Se o seu banco de dados SQL Server tem uma espécie seqüência que seja diferente da sua configuração atual tempdb collation e, em seguida, comparando os dados entre a sua candidatura e tempdb poderia criar alguns problemas.

Sintaxe Parcial

Você pode usar a cláusula COLLATE com o ALTER DATABASE CREATE DATABASE ou declaração de criar um banco de dados em cotejo seqüência nível. Da mesma forma, você pode utilizar a cláusula COLLATE com o CREATE TABLE ou ALTER TABLE para criar uma declaração collation seqüência para cada coluna string.

O seguinte é a sintaxe parcial da cláusula COLLATE.

COLLATE (<collation_name> | database_default)

Neste sintaxe, collation_name é o nome do cotejo de ser aplicado para a coluna definição, a definição ou banco de dados.

O database_default opção faz a cláusula COLLATE para herdar o cotejo de a atual base de dados. Você pode usar esta opção na cláusula COLLATE para especificar que uma coluna em uma tabela temporária deve usar a collation padrão da atual base de dados para a conexão ao invés de tempdb.

Precedência de Comparação

Collation precedência determina o cotejo de o resultado final de uma expressão que é avaliada para uma string de caracteres. Determina igualmente o collation que é utilizada pela colação de sensibilidade de operadores que usam caracteres fatores de produção, mas não retornam uma string de caracteres como LIKE ou IN.

Collation precedência também é determinado tipo de dados depois da conversão. O operando a partir do qual a collation resultante é tomada pode ser diferente da que fornece os dados operando tipo de resultado final.

Categoria de Comparação

A seguir estão as categorias de collations.

Coercible - Default. T-SQL qualquer função que não tem uma string de entrada, mas retorna uma string de saída. Se um objeto é declarada em um definido pelo usuário, função, procedimentos, ou acionar, o objeto é atribuído ao padrão collations do banco de dados no qual a função, procedimentos, ou acionar é criada.

X. Uma referência implícita coluna. O cotejo da expressão (X) é tomada a partir do cotejo de uma coluna definida na tabela ou vista.

Explícita X. Uma expressão que é explicitamente expressos para um cotejo específicos (X) usando uma cláusula COLLATE na expressão.

Não-collation. Indica que o valor de uma expressão é o resultado de uma operação entre duas cordas que têm conflitantes collations.

Regras para as categorias de comparação

A categoria de colação o resultado de uma expressão complexa baseia-se nos seguintes regras:

Explícita tem precedência sobre implícita. Implícito tem precedência sobre Coercible-padrão.

Combinação de duas menções explícitas que foram atribuídos diferentes collations gera um erro.

Combinação de duas expressões implícito que têm diferentes collations retorna um resultado com um sem-collation rótulo.

Combinando com uma expressão não-cotejo com uma expressão de qualquer categoria, salvo expressa cotejo, produz um resultado com um sem-collation rótulo.

Combinando com uma expressão não-cotejo com uma expressão que tem collation explícita, retorna uma expressão com um rótulo explícito.

Exemplos:

O exemplo a seguir ilustra os problemas que ocorrem devido à collation seqüências.

Considere as seguintes tabelas que têm um carácter à base de coluna.

```
USE tempdb;
```

```
CREATE TABLE # um (char_set1 VARCHAR (50) cotejar Latin1_General_CI_AS)
```

```
CREATE TABLE # b (char_set2 VARCHAR (50) cotejar Latin1_General_BIN)
```

Inserir um valor idêntico em ambas as tabelas.

```
INSERT INTO # um VALUES ("Iris")
```

```
INSERT INTO # b VALUES ("Iris")
```

A seguinte consulta realiza uma junção entre as duas tabelas.

```
SELECT * FROM um JOIN # b ON char_set1 = char_set2
```

A seguinte mensagem de erro é exibida.

```
Msg 468, Level 16, State 9, Line 2
```

```
Não é possível resolver o conflito entre a collation "Latin1_General_BIN" e "Latin1_General_CI_AS" no operador igual .
```

Isto porque as duas tabelas têm diferentes cotejar seqüências. Você pode resolver esse problema, ao fazer uma conversão explícita cotejar.

```
SELECT * FROM um JOIN # b em char_set1 = char_set2
```

```
COLLATE Latin1_general_CI_AS
```

Identificando a importância da Precedência de tipos de dados

Quando há uma expressão que envolve dados de diferentes tipos de dados, um tipo de dados é implicitamente convertido para o outro. Isto é feito com base nos dados tipo precedência regras. Se a conversão não é suportada uma conversão implícita, então ocorre um erro. Às vezes as conversões implícitas podem dar resultados inesperados.

Ordem de precedência dos Tipos de Dados

1 sql_variant	2 XML	3 datetime	4 smalldatetime	5 Float
6 real	7 decimal	8 money	9 smallmoney	10 BIGINT
11 int	12 smallint	13 tinyint	14 piece	15 ntext
16 text	17 image	18 timestamp	19 uniqueidentifier	20 nvarchar
21 nchar	22 VARCHAR	23 char	24 varbinary	25 binários (mais baixa)

Exemplos de tipo de dados precedência

Adicionando diferentes tipos de dados

As seguintes operações aritméticas são executadas na manifestação de diferentes tipos de dados.

Neste exemplo, uma expressão int é adicionada a uma expressão char.

```
SELECT 5 +'4 '
```

Aqui, int tem maior precedência do que char. Portanto, a expressão char é convertida para um valor inteiro de 4 e de o seguinte resultado é retornado.

9

(1 linha(s) afetada)

A seguinte consulta irá resultar em um erro.

```
SELECT 5 +'4 .5 '
```

A seguinte mensagem de erro será exibida.

Msg 245, Nível 16, Estado 1, Linha 1

Falha na conversão ao converter as VARCHAR .5 valor'4 'para tipo de dados int.

Isto é porque o SQL Server irá converter o personagem valor de'4 .5 'para um valor inteiro, porque o tipo da expressão já foi definido com o primeiro operando. Isto irá causar um erro de novo, porque 4,5 não pode ser convertido para um valor inteiro.

Se você reescrever a mesma consulta ao usar o tipo float para a esquerda expressão, então você irá obter o resultado desejado.

```
SELECT 5.0 +'4 .5 '
```

O seguinte é o resultado conjunto da consulta.

9,5

(1 linha (s) afetada)

Desempenhando várias operações em diferentes tipos de dados

Neste exemplo, duas operações estão sendo realizadas em expressões com diferentes tipos de dados. O SQL Server executa a operação multiplicação primeiro porque a multiplicação operador tenha uma prioridade mais elevada do que a adição operador. Para realizar a multiplicação operação, tem de converter o valor de char "3" para um inteiro. O resultado da operação é a multiplicação 12, que é então adicionado ao valor inteiro de 4 a 16, tal como produzir o resultado.

```
SELECT 4 +'3 '* 4
```

O seguinte é o resultado conjunto da consulta.

16

Se a direita expressão é uma expressão em vez de um personagem inteiro expressão e, em seguida, a consulta irá resultar em um erro. Isto é porque o SQL Server irá executar a primeira operação aritmética entre as duas expressões personagem. Não há dados tipo de conversão é necessária neste caso, porque ambas as expressões são do mesmo tipo de dados. Por isso, a multiplicação operador retorna um erro.

```
SELECT 4 +'3 '* '4'
```

A seguinte mensagem de erro será exibida.

Msg 8117, Level 16, State 1, Line 1

Operando tipo de dados VARCHAR é inválido para multiplicar operador.

Para realizar a primeira operação além disso, é necessário reescrever a mesma expressão da seguinte maneira.

```
SELECT (4 +'3 ') * '4'
```

O seguinte é o resultado conjunto da consulta.

28

Nota : Você precisa ser cauteloso quando se utilizam valores de diferentes tipos de dados, pois eles podem causar resultados inesperados e, em alguns casos erros. Além disso, implícito no tipo de conversão, consultas que costumava ser executado com eficiência índice pesquisas podem agora usar Índice scans ou mesa scans, resultando em maior tempo CPU e execução tempo. Por isso, é melhor para realizar uma conversão explícita do que a confiar na conversão implícita e da primazia de tipos de dados.

Identificando conversões implícitas válidas

Classifique as seguintes expressões tão rapidamente quanto possível para os seus associados categorias clicando no balde apropriado. Você também pode utilizar os atalhos de teclado, premindo 1 balde para a esquerda para a direita e 2 balde.

Questões

Você precisa comparar dois conjuntos de nomes na parte produto AdventureWorks banco de dados que foram armazenados em diferentes collation seqüências. Como é que vai comparar os dados armazenados com o personagem collation diferentes seqüências?

[Você pode fazer uma conversão implícita cotejar.](#)

[Você não pode comparar os dados armazenados em diferentes collation seqüências.](#)

[Você pode fazer uma conversão explícita cotejar. X](#)

[Você precisa comparar os dois caracteres dados diretamente através de uma comparação operador.](#)

Você precisa escrever uma consulta que usa vários tipos de dados. Para fazer isso, você precisa conhecer a ordem de precedência dos tipos de dados. Qual é a correta ordem de precedência dos tipos de dados a partir da mais alta?

[2 sql_variant](#)

[4 datetime](#)

[1 definido pelo usuário, os tipos de dados](#)

[3 XML](#)

[6 float](#)

[5 smalldatetime](#)

Qual das seguintes declarações são verdadeiras sobre os tipos de collation?

[Collation implícito toma precedência sobre collation explícito.](#)

[Combinação de duas menções explícitas que foram atribuídos diferentes collations gera um erro. X](#)

[Combinação de duas expressões implícito que têm diferentes collations retorna um erro.](#)

[Combinando com uma expressão não-cotejo com um cotejo implícita, produz um resultado que não tenha o rótulo-collation. X](#)

[Implícito collation toma precedência sobre coercible-collation padrão. X](#)

[Combinando com uma expressão não-cotejo com uma expressão que tem uma collation explícita, uma rendimentos não-collation.](#)

Consultando Estruturas Complexas de Tabelas

Você pode muitas vezes precisam trabalhar com esquemas de dados que contêm estruturas complexas, tais como hierarquias, a auto-referencial quadros, mesas e circular referenciamento. SQL Server 2005, introduziu uma série de técnicas para trabalhar com esses desenhos complexos tabela. Você pode consultar hierarquias e de auto-referenciamento tabelas utilizando uma CTE ou uma subquery.

O que são Hierarquias?

Em qualquer organização, banco de dados modelos são concebidos para satisfazer mudando fatores, tais como extensão e expansão crescente necessidades de negócios da organização. Você pode

muitas vezes precisam trabalhar com estruturas complexas, tais como hierarquias de dados em muitos modelos.

A hierarquia é um banco de dados com um projeto modelo único progenitor ou nó raiz com um ou mais gânglios criança que podem, por seu turno, ter zero ou mais nodos filho. A hierarquia é também referida como uma árvore invertida. Exemplos de estruturas hierárquicas incluir relatórios das estruturas empresariais, agrupamentos geográfica de inventários, monitoramento, peças explosão árvores, e do sistema de arquivos no computador. Você pode utilizar diversos métodos para aplicar hierarquia em um banco de dados.

Multireferencial

Você pode armazenar hierarquias de dados, utilizando o método multi-referencial. Neste método, você pode incluir referências sucessivamente a cada nível mais elevado em cada registro, com todos os níveis partilha de uma única tabela.

Por exemplo, em uma estrutura corporativa relatórios que você pode criar um link em cada funcionário para gravar seu supervisor. Em cada supervisor registro, você pode criar um link para o seu supervisor, e assim por diante, baseado no organograma.

Esta técnica facilita simples consultas. No entanto, esta estrutura hierárquica tem um monte de redundância. Os repetidos NULOS pode assumir uma grande quantidade de espaço e reduzir a eficiência.

Uma mudança de uma relação hierárquica pode ter efeitos em cascata em toda linhas e colunas em uma forma complexa. Se você precisa para adicionar mais níveis e, em seguida, uma vez que exigiria uma mudança esquema, ou seja, um par de colunas.

Por isso, este multi-referencial estrutura é inflexível e mais adequado para simplista e estática hierarquias.

Lista de Adjacência

Você também pode armazenar em um banco de dados hierarquias adjacência lista, que é uma variação da estrutura multi-referencial, mas sem a redundância de informações. Neste método, apenas o pai campo é armazenada e sucessivamente mais elevados níveis são determinados por subir na hierarquia, programaticamente.

Esta abordagem é flexível porque alguns nodos maio ramo profundo enquanto outras podem permanecer superficial. Além disso, a profundidade da criança nodos não está dependente do esquema design. Um conjunto completo de ramos pode ser movido por uma mudança na relação mãe um único registro.

Além disso, há espaço mínimo as exigências impostas pelo pai campo usado para criar os relacionamentos. No entanto, adjacência listas podem ser complexas para uma consulta na hierárquica e pode exigir vários auto-joins ou iterativo loops.

Como consultar hierarquias

Você pode executar consultas sobre hierarquias, utilizando uma CTE ou uma subquery. Por exemplo, dados que hierarquicamente representa as regiões do mundo está armazenado na hierarquia tabela. Os exemplos na tabela a seguir demonstram como uma subquery ou uma CTE pode ser usado para consulta neste Hierarquia tabela para obter uma lista de regiões que pertencem a um determinado país ou continente.

Usando uma subconsulta

Dependendo do cenário, uma subquery pode demorar um tempo maior para executar. Além disso, uma subquery maio entra um ciclo infinito devido a um erro programação. Os seguintes subquery invoca uma função definida pelo usuário, chamado GetGeographyChain função de recuperar a lista das regiões que pertencem direta ou indiretamente para a América do Norte.

```
Select * from Hierarchy
Where Parent in
(Select Child from dbo.GetGeographyChain ('North America', 1))
```

Usando uma CTE

Você pode obter o mesmo resultado, usando um conjunto CTE ou uma subquery. No entanto, CTEs são uma forma muito mais eficiente de examinar hierarquias. O código a seguir exemplo recupera a lista das regiões que pertencem a um determinado país, utilizando uma recursiva CTE.

```
WITH HierCTE AS
(
  Select parent, child FROM Hierarchy WHERE Parent='North
  America'
  UNION ALL
  Select h.parent, h.child FROM Hierarchy h INNER JOIN HierCTE
  cte on h.Parent=cte.Child
)
Select * FROM HierCTE
```

O seguinte é o resultado completo conjunto de ambos os subquery e do CTE.

Parent	Child
New York	New York City
North America	Canada
North America	United States
United States	New York
United States	Washington
Washington	Redmond

O que são auto-referências?

Muitos modelos de dados hierárquicos implementar dados para lidar com o aumento das necessidades de uma empresa. Para lidar com complexos e díspares organização das estruturas, muitas vezes você pode precisar usar auto-referenciamento tabelas. Uma coluna em uma tabela de auto-referenciamento referências outra coluna do mesmo quadro.

Ao utilizar uma ferramenta de auto-referenciamento tabela, você pode implementar uma estrutura hierárquica de forma muito fácil. Ao contrário, uma referência circular é uma condição recursiva em uma tabela que faz referência a um segundo quadro, o qual, por sua vez, referências a primeira tabela.

Qualquer número de mesas podem ser ligados em conjunto de uma cadeia circular de chave estrangeira referências. No entanto, quando você usa referências circulares, é possível que um processo para introduzir um ciclo infinito se ele tenta seguir a cada referência. Por este motivo, recomenda-se a evitar a utilização de referências circulares.

Considere um quadro de funcionários que tem as colunas EmpID, Nome, Salário, DeptNo, e MgrID. O Departamento tabela tem as colunas DeptNo, Nome, Localização, e MgrID. A chave primária para a tabela é empregado EmpID. DeptNo é uma chave estrangeira referenciando o Departamento tabela. MgrID é também uma chave estrangeira, mas referências a ele próprio quadro de funcionários. O MgrID em uma fila de funcionários tabela de correspondências a EmpID do empregado gerente. É um exemplo de uma auto-referenciamento chave estrangeira. No quadro do departamento, o DeptNo é a chave primária. MgrID é uma chave estrangeira referenciando o quadro de funcionários. Identifica a EmpID gerente do departamento. Este é um exemplo de referências circulares. O Departamento de funcionários e quadros são circulares-referenciamento tabelas. O quadro de funcionários tem uma chave estrangeira referenciando o Departamento mesa, e ao Departamento tabela tem uma chave estrangeira referenciando o quadro de funcionários.

Como consultar Tabelas de auto-referencia

Você pode executar consultas sobre auto-referenciamento tabelas utilizando uma CTE ou usando uma subquery. Os exemplos utilizados no quadro a seguir demonstram como uma subquery ou uma CTE pode ser usado para obter o aluguel data, a hora de férias, e as horas de ausência por doença para todos os restantes trabalhadores relatório que direta ou indiretamente para o empregado com EmployeeID 263.

Usando Subconsulta

Você pode invocar o GetReportingChain função dentro de uma subquery para recuperar a lista dos funcionários que direta ou indiretamente relatório à pessoa com a EmployeeID 263. A subquery inicialmente verifica se ele está sendo chamado pela primeira vez ou se ela está sendo chamada recursivamente. Trata-itera por cada linha do quadro de funcionários e convida a GetReportingChain função recursivamente até que a subquery atinge o fim da hierarquia.

USE AdventureWorks;

SELECT EmployeeID, HireDate, VacationHours, SickLeaveHours

DESDE HumanResources.Employee

ONDE EmployeeID IN (SELECT EmployeeID DA dbo.GetReportingChain (263,1))

No entanto, uma subquery recursivas podem levar muito tempo para executar e à consulta pode ser mais complexa para escrever. Além disso, devido a um pequeno erro programação, uma subquery recursiva pode ir a um loop infinito.

Usando CTE

Você pode recuperar a lista dos funcionários que direta ou indiretamente relatório à pessoa com a EmployeeID 263, utilizando uma recursiva CTE. A âncora membro do CTE define o ponto de partida da recursão, ao passo que a invocação recursiva membro mantém automaticamente até que ele próprio não consegue retornar qualquer linhas.

USE AdventureWorks;

DECLARE @ EmployeeID int

SET @ EmployeeID = 263;

WITH CTE_Example (EmployeeID, EmployeeName, BossID)

AS

(

SELECT EmployeeID, firstname + " + apelido, e de gerente DA HumanResources.Employee

JOIN c Person.Contact em e.ContactID = c.ContactID

WHERE EmployeeID = @ EmployeeID

UNION ALL

SELECT e.EmployeeID, firstname + " + apelido, e de gerente DA HumanResources.Employee

JOIN c Person.Contact em e.ContactID = c.ContactID

JOIN CTE_Example em e.ManagerID = CTE_Example.EmployeeID

)

SELECT EmployeeID, HireDate, VacationHours, SickLeaveHours

FROM HumanResources.Employee

EmployeeID IN (SELECT EmployeeID de CTE_Example)

O CTE é um método mais simples e elegante, em comparação com uma subquery. A CTE é mais legível e para a execução da consulta é mais rápido na maioria dos casos. Além disso, quando você usa um CTE, SQL Server oferece proteção contra a built-in loops infinitos.

O resultado de ambas as consultas

EmployeeID	HireDate	VacationHours	SickLeaveHours
5	1998-01-11 00:00:00.000	9	24
263	2001-01-05 00:00:00.000	7	23
265	2001-01-23 00:00:00.000	8	24

Questões

Que uma das seguintes afirmações é verdadeira sobre hierarquias?

Em uma hierarquia multi-referencial, cada nó pode ter apenas um único nó filho.
Em uma hierarquia multi-referencial, só o pai nó é armazenado.
Em uma lista adjacência hierarquia, os níveis mais elevados da hierarquia são determinados programática. X
Em uma lista adjacência hierarquia, a ligação entre mãe e filho cada nó é armazenado.

Que uma das seguintes afirmações é verdadeira sobre auto referência?

A chave primária em uma coluna auto-referenciamento tabela referências outro quadro em que se encontra a chave estrangeira coluna.
A chave estrangeira uma coluna no quadro de auto-referenciamento referências outra coluna do mesmo quadro. X
Não pode ser apenas uma chave estrangeira em uma coluna auto-referenciamento tabela.
A chave estrangeira referências pode ser usado para ligar duas tabelas.

Escrevendo consultas eficientes

Para melhorar o desempenho das aplicações, você deve escrever consultas que executam de forma eficiente e utilização dos recursos mínimos SQL Server. Ao seguir algumas regras simples e diretrizes, você pode melhorar o desempenho das consultas. Você pode usar a cláusula EXISTS e o caso declaração em determinadas consultas para reduzir a sua execução tempo. Dependendo do cenário, é possível determinar se a utilização ou subconsultas junta. Você também pode usar variáveis tabela ou tabelas temporárias na base de dados que você deseja recuperar.

Recomendações para escrever consultas eficientes

Para escrever consultas eficientes, você precisará seguir algumas recomendações simples. Com a implementação destas recomendações, poderá melhorar o desempenho das consultas e recuperar os resultados exigidos no método mais eficiente.

SELECT (*)

Em vez de utilizar a cláusula SELECT * em uma consulta, recomenda-se a especificar as colunas por causa das seguintes razões:

Rede tráfego é reduzido. Usando a cláusula SELECT * pode ter um impacto sobre o desempenho se a tabela tem muitas colunas, ou se a tabela tem grandes colunas porque estes tipos de colunas podem demorar muito tempo para ser transferidos através da rede.

O código é mais fácil de entender.

Ele também elimina a necessidade de alterar a consulta se a tabela base estrutura seja modificada. Se qualquer coluna é adicionado ou removido da tabela base ou visualizar e, em seguida, a cláusula SELECT * pode produzir resultados errados conjuntos e à consulta também pode falhar.

WHERE

Você deve sempre incluir uma cláusula WHERE na declaração SELECT para limitar o número de linhas retornadas a menos que você deseja recuperar todas as linhas de uma tabela. Se você não incluir uma cláusula WHERE, SQL Server realiza uma varredura da tabela da mesa e retorna todas as linhas. Isto pode ter impacto na desempenho SQL Server, pelas seguintes razões:

Uma tabela scan trava a tabela durante um tempo de consumo de varredura e potencialmente impede outros usuários de acessá-lo.

Ele aumenta o tráfego da rede.

Isso resulta em desperdício de recursos para o SQL Server I / O operações que não são essenciais.

Também impacta a capacidade do SQL Server essencial para a reutilização de dados da cache.

ORDER BY

Você não deve utilizar a cláusula ORDER BY em uma declaração SELECT a menos que seja realmente necessária. Incluindo a cláusula ORDER BY resulta em um aumento geral e pode afetar o desempenho de uma consulta. Em alguns casos, pode ser mais eficiente para classificar os dados ao cliente do que a do servidor. Em outros casos, é possível que não sejam essenciais para classificar os dados para alcançar o resultado desejado. Você não deve utilizar a cláusula ORDER BY, tendo em vista definições do SQL Server 2005 como os dados não é garantida a ser devolvido em uma forma ordenada. No entanto, você pode utilizar a cláusula ORDER BY, se for apoiada com um índice.

Apelidos de Tabelas

Você deve sempre usar tabela aliases e prefixo coluna todos os nomes com os correspondentes apelidos quando você estiver usando múltiplas tabelas em uma consulta. Ao fazer isso, você pode evitar ambigüidade quando se refere a colunas de tabelas diferentes que tenham nomes idênticos. Também faz com que o código mais legível, em consultas complexas.

Referência total

Você deve sempre especificar as duas partes nome a um objeto. O SQL Server armazena objetos em esquemas. O esquema padrão que está associada a objetos é DBO. No entanto, você pode ter objetos com o mesmo nome presentes em diferentes esquemas. Para evitar ambigüidades e de desempenho para benefício, você deve sempre fornecer o nome das duas partes nome do objeto durante a tentativa de acessá-lo a partir do banco de dados.

Variáveis na cláusula WHERE

Você deve evitar a utilização de variáveis na cláusula WHERE de uma consulta localizado num lote. Quando você usar uma cláusula WHERE que tem colunas indexadas em uma consulta, a consulta otimizador seleciona os índices para realizar a consulta e obtém resultados muito rapidamente. No entanto, se você usar variáveis na cláusula WHERE, a tabela varredura podem ser utilizadas em vez das colunas indexadas e do desempenho da consulta pode ser afetado. A consulta analisador não sabe o valor das variáveis quando se escolhe um método de acesso para realizar a consulta. Isso evita que o otimizador de consulta determinar todas as informações necessárias para escolher um método de acesso que utiliza índices.

Casing para nomes de objetos

Você deve usar a mesma caixa de nomes como objeto de terem sido especificados na base de dados. Você pode especificar se você quer que o cotejo de ser maiúsculas de minúsculas ou maiúsculas e minúsculas quando o dicionário ordem é utilizado para o collation. Processo de sensibilidade aplica-se a SQL identificadores e senhas, bem como aos dados. Se você especificar uma ordem de classificação padrão binária, ou maiúsculas de minúsculas para uma instância do SQL Server ou banco de dados, todas as referências a objetos devem usar o mesmo acontece com os quais foram criados. Ao usar a mesma caixa de objeto nomes, você pode melhorar o desempenho da consulta.

Como otimizar desempenho de consulta usando EXISTS

A cláusula EXISTS é uma forma muito eficaz de consultas por escrito, porque não são, na realidade, as linhas recuperadas. Quando uma subquery é introduzida com a cláusula EXISTS, SQL Server verifica se os dados que corresponda a subquery existe. SQL Server termine a recuperação de linhas quando pelo menos uma linha que satisfaça a condição WHERE na subquery é retornado. Por conseguinte, as consultas por escrito com a cláusula EXISTS executar muito rapidamente.

Você pode usar muitas vezes o valor do COUNT (*) função na aplicação de regras em T-SQL. Você

pode usar a cláusula EXISTS vez de o COUNT (*) função de avaliar se você tem pelo menos uma linha que satisfaça determinadas condições.

O código seguinte exemplo demonstra a eficiência do uso da cláusula EXISTS em comparação com o COUNT (*) função.

A seguinte consulta verifica se há pelo menos uma linha disponível nos SalesOrderHeader tabela que tem um valor de 1 na coluna ShipMethodID. Se existe a linha e, em seguida, ele exibe uma mensagem informando que o método de transporte marítimo não pode ser excluída.

```
IF (SELECT COUNT (*) FROM Sales.SalesOrderHeader
    WHERE ShipMethodID = 1) > 0
PRINT "You cannot delete this Shipping Method"
```

O plano de execução mostra que o SQL Server tem de ler todas as colunas na tabela antes de avaliar o SalesOrderHeader IF expressão. Você pode obter o mesmo resultado de forma mais eficiente usando a cláusula porque o IF EXISTS condição avalia para true, logo que SQL Server localiza a primeira ocorrência de 1 na coluna ShipMethodID.

```
IF EXISTS (SELECT * FROM Sales.SalesOrderHeader
    WHERE ShipMethodID = 1)
PRINT 'You cannot delete this Shipping Method'
```

Como reescrever subconsultas com joins

Geralmente, você pode executar uma consulta usando subqconsultas ou junta. Com base em um determinado cenário e os dados, pode ser um método mais eficiente do que os outros. Você pode reescrever a maioria como subconsultas junta para produzir o mesmo resultado.

Subconsultas

A maior parte das subconsultas podem ser reescritas como junta de produzir o mesmo resultado. O desempenho de uma consulta pode ser semelhante a um e juntar uma subquery. A consulta otimizador normalmente otimiza subconsultas, para que ele usa a amostra execução um plano que iria aderir semanticamente equivalente utilização. Por exemplo, se o resultado de uma consulta é pequeno, ou se não existem índices em colunas aderiram ao e, em seguida, uma subquery pode ser mais rápido que um join.

Joins

Tipicamente, junta são mais eficientes do que subconsultas porque SQL Server executa junta concomitantemente que subconsultas são executados sequencialmente. Uma subquery pode exigir a consulta otimizador para realizar etapas adicionais, tais como a triagem, que possam influenciar a estratégia de transformação. Ao usar junta, a consulta otimizador pode recuperar os dados da forma mais eficiente. Por isso, dependendo do cenário, você pode muitas vezes necessidade de reescrever as subconsultas junta.

Nota: É recomendado que você teste alternativas para determinar qual opção dá-lhe os resultados da melhor maneira. Dependendo de fatores como índices coluna, o tamanho do resultado fixado, e dada a dados, você precisa escolher o método mais adequado para uma consulta.

Como otimizar o desempenho escrevendo Consultas One-Pass

Você pode escrever muitas vezes precisam de consultas que fazem iteração através da mesma mesa várias vezes ou condicionalmente modificar a mesma tabela de maneira diferente para diferentes

linhas. Em tais casos, você pode reescrever a consulta a declaração utilizando o caso para que ele itere através da tabela apenas uma vez.

Ao usar o caso declaração, você pode reescrever consultas complexas em simples, e por vezes mais eficiente instruções SQL. Você pode incluir vários tipos de processamento condicional em uma instrução SQL. Você também pode incluir mais comandos lógicos SQL em vez de expressar-los em um idioma diferente acolhimento.

Usando a palavra-chave UNION

No exemplo a seguir, a União palavra-chave é utilizada para classificar os produtos em várias seções em função da sua tabela de preços. Esta consulta tem de fazer quatro passes ao longo dos Produtos mesa para retornar os resultados desejados.

Use AdventureWorks

```
SELECT ProductID, Name, ListPrice, 'Very Expensive'
FROM Production.Product
WHERE ListPrice > 1000
UNION ALL
SELECT ProductID, Name, ListPrice, 'Moderate'
FROM Production.Product
WHERE ListPrice BETWEEN 500 AND 1000
UNION ALL
SELECT ProductID, Name, ListPrice, 'InExpensive'
FROM Production.Product
WHERE ListPrice BETWEEN 1 AND 500
UNION ALL
SELECT ProductID, Name, ListPrice, 'Not for resale'
FROM Production.Product
WHERE ListPrice IS NULL OR ListPrice < 1
```

ProductID	Name	ListPrice	No column name
680	HL Road Frame - Black, 58	1431.50	Very Expensive
706	HL Road Frame - Red, 58	1431.50	Very Expensive
717	HL Road Frame - Red, 62	1431.50	Very Expensive
718	HL Road Frame - Red, 44	1431.50	Very Expensive
719	HL Road Frame - Red, 48	1431.50	Very Expensive

Utilizando a declaração CASE

No código a seguir exemplo, a mesma consulta é reescrito a declaração utilizando o processo de consulta e de itera através da tabela de uma só vez.

Use AdventureWorks

```
SELECT ProductID, Nome, ListPrice,
CASE
WHEN ListPrice > 1000,00 e 'muito caro "
    WHEN ListPrice entre 500 e 1000 e 'moderado'
    WHEN ListPrice entre 1 e 500 em 'baratos'
    ELSE 'Não é para revenda"
END
FROM Production.Product
```

Ambas as consultas produzem os mesmos resultados embora a ordem de classificação possa ser diferente. No exemplo que usa a palavra-chave **UNIÃO**, SQL Server teria que fazer quatro passagens através dos dados, uma para cada **SELECT** declaração utilizado. No exemplo que usa o caso declaração, através de um passe os dados disponíveis são suficientes para voltar a corrigir os resultados. Portanto, o caso é uma declaração muito mais eficaz forma de executar essa pesquisa.

A declaração **CASE** também pode ser utilizada em situações em que temos de atualizar um quadro especial com valores diferentes para diferentes linhas.

Exemplo:

No exemplo a seguir, as atualizações da declaração **ListPrice** na tabela **Production.Product** dependendo do **StandardCost** de um produto.

```
UPDATE Production.Product SET ListPrice = Processo
WHEN StandardCost <100 Então ListPrice * 1,1
WHEN StandardCost between 100 e 300 then ListPrice * 1,2
WHEN StandardCost between 300 e 5000 then ListPrice * 1,3
ELSE ListPrice * 1,35
END
```

A seguinte mensagem é exibida ao executar a consulta:
504 linha (s) afetadas.

Orientações para usar tabelas variáveis e tabelas temporárias

Você pode muitas vezes necessidade de armazenar dados em um local temporário. Em SQL Server, você pode optar por armazenar dados temporários quer em tabelas variáveis ou tabelas temporárias. Ambas as tabelas temporárias e tabelas variáveis são criadas e transformadas em dados do cache. Ambas são armazenadas no banco de dados tempdb. Você precisa usar a fim de determinar se a tabela ou tabelas temporárias variáveis em função dos seguintes fatores: o número de linhas que são inseridas na tabela, o número de recopilações da consulta que pode ser evitado, e do tipo de consultas e de sua dependência índices e estatísticas de desempenho.

Tabela temporária

Tabelas temporárias são semelhantes aos quadros permanentes, exceto as tabelas temporárias são armazenadas no tempdb e de que você precisa para excluí-las quando eles já não são utilizados. Tabelas temporárias são automaticamente atribuídos de 0 quando termina o lote a ser publicados. Eles podem ser usados em pesquisas como normal tabelas. Você pode criar índices em tabelas temporárias, e também manter estatísticas. Para consultas complexas em grandes quadros, a consulta otimizador pode usar estes índices e estatísticas para determinar o melhor plano de execução de uma consulta. Por isso, recomenda-se usar tabelas temporárias quando uma consulta envolve um volume significativo de dados e não há uso repetido de cima da mesa.

Vantagens

Tabelas temporárias têm as seguintes vantagens:

Você pode criar índices em uma tabela temporária para aumentar a desempenho consulta.

Você pode criar índices não-agregadas e restrições sobre as tabelas temporárias.

Ao usar a criar estatísticas declaração, poderá manter estatísticas sobre uma tabela temporária.

Desvantagens

Tabelas temporárias têm as seguintes desvantagens:

Tabelas temporárias poderia resultar em um grande número de recopilações quando são

referenciados a partir de procedimentos armazenados.

Em uma declaração CREATE ou ALTER, tabelas temporárias precisam de uma resolução tão repetido que a tabela pode ser referenciada a partir de um procedimento armazenado aninhados.

Tabelas temporárias podem exigir mais recursos para efetuar login comparado ao quadro variáveis.

Exemplo

```
CREATE TABLE #ProductSales(ProductID int, Name nvarchar(40),
TotalSales int)
INSERT INTO #ProductSales SELECT p.ProductID, p.Name,
SUM(OrderQty) AS TotalSales FROM Production.Product p
JOIN Sales.SalesOrderDetail sod ON p.ProductID=sod.ProductID
GROUP BY p.ProductID, p.Name SELECT * FROM #ProductSales
```

ProductID	Name	TotalSales
879	All-Purpose Bike Stand	249
712	AWC Logo Cap	8311
877	Bike Wash - Dissolver	3319

Tabelas variáveis

Semelhante a tabelas temporárias, o quadro também variáveis são armazenadas no banco de dados tempdb porque podem conter dados que não pode caber na memória disponível. Tabela variáveis têm um alcance bem definido, tais como a função, procedimentos, ou lote em que sejam declarados. No seu âmbito, uma tabela variável pode ser utilizada como uma tabela periódica. Tabela variáveis são automaticamente limpo no final da função, procedimentos, ou lote em que estão definidas. Recomenda-se a utilização tabela variáveis, sempre que possível a menos que exista um volume significativo de dados. No entanto, você precisa estar ciente de algumas restrições que se aplicam a tabela variáveis.

Vantagens

Tabela variáveis têm as seguintes vantagens:

Tabela variáveis têm um alcance bem definido.

Tabela variáveis podem ser usadas em qualquer lugar uma mesa ou mesa expressão é utilizada na instrução SELECT, INSERT, UPDATE, e DELETE declarações.

Tabela variáveis são completamente isoladas para o lote que criá-las. Por isso, os repetidos em uma resolução que ocorre quando uma tabela temporária CREATE ou ALTER declaração tem lugar, não irá ocorrer em uma tabela variável.

Tabela variáveis também resultar em menos recopilações de um procedimento armazenado em comparação com tabelas temporárias. Procedimentos armazenados podem usar um plano que já está compilado, economizando assim recursos para a tramitação do procedimento armazenado.

Tabela variáveis exigem menos bloqueio e registrando operações que envolvam recursos porque eles passada apenas para a duração de uma atualização na tabela variável.

Desvantagens

Tabela variáveis têm os seguintes inconvenientes:

Índices não pode ser criada a tabela variáveis, à exceção do sistema de índices que são criados por uma PRIMARY ou UNIQUE constrangimento.

As estatísticas não pode ser criada em tabela através da criação automática ou variáveis usando a criar estatísticas declaração.

A tabela a definição não pode ser alterada após a declaração inicial DECLARAM.

Tabela variáveis não podem ser usados em um INSERT EXEC ou SELECT INTO declaração.

CHECK constrangimentos, valores padrão, e computados colunas da tabela tipo de declaração não

pode pôr definido pelo usuário, funções.

EXEC A declaração ou o sp_executesql procedimento armazenado não pode ser usado para executar uma dinâmica SQL Server consulta que remete uma tabela variável, se a tabela foi criada fora da variável EXEC declaração ou o sp_executesql procedimento armazenado.

Exemplo:

```
DECLARE @ProductsSales table(ProductID int, Name nvarchar(40),
TotalSales int)
INSERT INTO @ProductsSales SELECT p.ProductID, p.Name,
SUM(OrderQty) AS TotalSales FROM Production.Product p
JOIN Sales.SalesOrderDetail sod ON p.ProductID=sod.ProductID
GROUP BY p.ProductID, p.Name SELECT * FROM @ProductsSales
```

ProductID	Name	TotalSales
879	All-Purpose Bike Stand	249
712	AWC Logo Cap	8311
877	Bike Wash - Dissolver	3319

Teste

Cada azulejo contém uma verdadeira declaração e de uma falsa declaração. Clique no azulejos em uma linha ou coluna até que cada telha mostra a verdadeira declaração e, em seguida, clique na seta ao lado da linha ou superior da coluna para verificar a sua resposta. Você usa um turno quando você clicar Enviar uma incorreta ou azulejo. Remover todas as peças antes de utilizar todas as suas voltas.

Mantendo Arquivos de Consulta

Quando você trabalhar em equipe, compartilhamento de arquivos em todo projetos podem resultar em conflitos versão. Além disso, se os arquivos são armazenados no disco rígido local, podem não estar disponíveis a partir de um repositório centralizado convenientes. Além disso, você precisa para enfrentar os riscos associados com a segurança da propriedade intelectual ou o apagamento acidental de arquivos por usuários. Para abordar estas questões, você pode usar o Microsoft Visual SourceSafe para a versão fonte de controlo e gestão. Visual SourceSafe fornece recursos, como o check in, check out, fundir, história, e comentam que você pode usar para a versão de gestão.

O que é o controle de fontes e versões?

Quando você trabalhar em equipe, o que você precisa para permitir que o desenvolvimento paralelo de projetos por membros da equipe. No entanto, quando membros da equipa trabalhar com o mesmo arquivo, eles precisam conciliar os conflitos entre as diferentes versões do arquivo e evitar a perda potencial de mudanças importantes. Em tais situações, você precisa de software, tais como Visual SourceSafe para a versão fonte de controlo e gestão.

Controle de Fonte

Uma fonte sistema de controle consiste em duas componentes: uma fonte controle provedor e uma fonte controlo cliente. Fonte controle refere-se a um sistema no qual um servidor funciona como um repositório central para todos os arquivos e uma fonte de software, tais como controlar Visual SourceSafe faixas arquivo Versões e controla o acesso a esses arquivos.

Os arquivos de controle da fonte servidor são acessadas por usuários a partir de duas ou mais fontes controle clientes.

Gerenciamento de Versões

Para evitar arquivos versões conflitantes, Visual SourceSafe arquivo controla o acesso e impõe um protocolo pelo qual os usuários que querem modificar um arquivo deve verificá-la. Depois de verificar se o arquivo, o usuário pode modificá-la. Quando o usuário verifica no arquivo de novo, que passa a ser o arquivo última versão disponível.

Você pode fornecer quer acessar arquivos compartilhados ou exclusivos arquivo acesso aos usuários. Se o acesso aos arquivos é compartilhado e, em seguida, mais do que um usuário pode verificar se um arquivo. Quando os usuários verificar nos arquivos, Visual SourceSafe prevê um mecanismo para fundir as versões.

Se o acesso aos arquivos projeto é exclusivo e, em seguida, Visual SourceSafe permite apenas um usuário em um momento de verificar arquivos e modificá-los. Porque os usuários podem compartilhar arquivos, eles podem colaborar em toda a projetos e compartilhar arquivos-origem controlada. Alterações para um arquivo compartilhado são refletidas em todos os projetos que compartilham o arquivo.

Você pode arquivar sucessivas versões do código-fonte controlada arquivos. Visual SourceSafe armazena os dados que distingue uma versão de uma fonte de controle a partir de outro arquivo. Portanto, você pode recuperar qualquer versão de uma fonte de controle de arquivo. Você também pode designar qualquer versão a ser a versão mais recente do que o arquivo.

Além disso, você pode manter versão informações detalhadas e informações sobre a história arquivo-fonte controlada arquivos. Visual SourceSafe armazena dados como a data e hora em que o arquivo foi criado, quando foi verificada no check-out ou, e o usuário que realizou o recurso.

Outros Benefícios

Visual SourceSafe facilita a automatização da fonte repetida com freqüência às operações de controle. Você pode definir uma interface para estas operações a partir do prompt, sob a forma de ficheiros batch. Então, você pode usar estes arquivos batch para automatizar a fonte controlar tarefas que você executa regularmente.

Você pode conservar espaço em disco, tanto sobre o controle fonte cliente e servidor. Visual SourceSafe conserva o espaço em disco no servidor, armazenando a última versão de um arquivo, e as diferenças entre cada versão e da versão que precede ou se segue. Visual SourceSafe também conserva espaço em disco com os clientes.

Você pode ocultar pastas e arquivos, para que eles não são transferidas para seu disco local. Além disso, quando você armazenar seus arquivos em Visual SourceSafe, você pode recuperar arquivos que podem ser suprimidas acidentalmente pelos usuários. Você pode restaurar a última versão do arquivo que foi verificada em fonte controle.

Apresentações da Fonte Visual Segura

Visual SourceSafe é utilizada para gerenciar seus projetos que podem consistir em vários tipos de arquivos como arquivos texto, planilhas, arquivos gráficos, arquivos binários, ficheiros de som, vídeo ou arquivos por salvá-las em um banco de dados. Você também pode compartilhar arquivos entre dois ou mais projetos. Você pode adicionar arquivos para consulta e Visual SourceSafe-los de volta ao banco de dados. As mudanças que foram feitas para a consulta também são salvos, e você pode recuperar uma versão antiga da consulta a qualquer momento. Visual SourceSafe mantém várias versões de um arquivo, incluindo um registro das alterações no arquivo de versão para versão. Você pode usar os diversos recursos do Visual SourceSafe para a versão de gestão eficaz.

Check In

O recurso cheque in é utilizado para fazer alterações em um arquivo de fonte de controle à disposição de outros usuários. Ao usar o SQL Server Management Studio, você pode verificar nos arquivos. Quando você verificar nos autos, a versão que você criou é copiada para controlar a fonte, que passa a ser a versão mais recente do arquivo, e está disponível para os usuários que têm permissões apropriadas. Quando você atualizar a fonte de controle, armazene periodicamente o que

you need to maintain control over the files. It is about specifying that the files that you mark will remain in check-out.

Check Out

The check-out resource is used to verify the file, after a file is added to the source control. When you do a check out of a file from source control, the source control creates a copy of the last version of the file on the local hard disk and removes the read-only attribute of the file. You can use SQL Server Enterprise Manager to check out files manually or automatically. You can find files manually, by opening the solution that contains the files in SQL Server Enterprise Manager and, then, clicking the Check Out command. You can also check out files automatically in the files if you configure the SQL Server Enterprise Manager environment.

You can find files in exclusive or shared mode, depending on the options configured in Visual SourceSafe. When you do a check out of a file in exclusive mode, you can modify the file. No other user can verify a file that has been checked out in exclusive mode. However, when you do a check out of a file in shared mode, any user can verify the same file. In case of a conflict between the version that is being verified and a more recent version, Visual SourceSafe prompts the user to resolve the conflicts.

Comment (Comentário)

Comment is an option available according to the version of the file. This option allows you to understand the changes made in the document that you incorporate in the version of the file. You can also authorize other users to obtain information about the changes in the files without having to analyze the content of the file and digitalize the changes manually.

Merge (Mesclar)

It is the process of combining the differences of two or more examples of a modified file into a new version of the file. Visual SourceSafe cannot resolve merge conflicts, but it presents you with a resolution. There are two methods that can be used for visualization and resolution of conflicts, visual merge and manual merge. Visual merge is the method that provides a graphical environment, a dialog box in which you can resolve all conflicts with maximum speed and control. Manual merge facilitates the merging of files manually.

History (Histórico)

It is a resource that records the history of changes for a file after it has been added to Visual SourceSafe. By using the details of the file history, you can return to any point in the history of a file and recover the file as it existed at that point. The history shows in the Options dialog box, the record of significant events in the current project, such as marking and suppression or addition of files and subprojects.

Using the Visual SourceSafe Source for Query Versions of Controls

You can use Microsoft Visual SourceSafe for the maintenance, management and control of various versions of your searches. Open Microsoft SQL Server Enterprise Manager. After verifying that the type of server, server name, and authentication boxes have been filled correctly, connect to the server. Then, you need to activate SQL Server Enterprise Manager to use a client source control. Now, expand the Source Control, and click the Plug-in Selection button. You need to verify if the Microsoft Visual SourceSafe option is selected, in the current source control plug-in space and, then, click OK.

Em seguida, você precisa clicar em Nova Consulta na barra de ferramentas. Em Solution Explorer, adicione um novo projeto. Em seguida, o nome do novo projeto CategorizeProducts. Em seguida, adicione uma nova consulta ao CategorizeProducts projeto. Após isto, aponte para o SQL Server Database Engine. Aponte para o banco de dados AdventureWorks. Na janela de busca, digite os comandos SQL necessários para listar todos os produtos na tabela de produtos do banco de dados AdventureWorks em Categoria, Subcategoria, Modelo, e de Produto. Executa a consulta. Verifique se os produtos que não são categorizados estão incluídos na lista.

Para adicionar o projeto para CategorizeProducts Visual SourceSafe, selecione o projeto e adicioná-la à fonte controle. Em seguida, salve o projeto no local necessários. Acrescentar o banco de dados SourceSafe assistente aparece. No Bem-vindo ao Assistente Adicionar SourceSafe Database página, clique em Next, para aceitar as configurações padrão. Na página Seleção de banco de dados, você pode especificar se pretende ligar a um banco de dados existente ou criar uma nova. Clique em Criar uma nova base de dados, e clique em Avançar.

Na nova base de dados Localização página, navegue para a pasta do projeto, e clique em Avançar. Isto denota a localização do Microsoft Windows partes arquivo que contém o banco de dados. Pode-se mencionar um nome adequado para a conexão do banco de dados que você está precisando. Nome do banco de dados Connection página, para aceitar as configurações padrão, clique em Avançar.

Visual SourceSafe trabalho prevê dois estilos: Lock-Modifica-Modelo de desbloqueio e Cópia-Modifica-Modelo de Junção. O Lock-Modifica-Modelo de desbloqueio permite apenas um usuário em um momento de mudar um arquivo. A Cópia-Modifica-Junção Modelo permite mais de um usuário para verificar se o mesmo arquivo para facilitar o desenvolvimento paralelo. No Team Version Control Model página, clique em Avançar para aceitar as configurações padrão. Quando terminar a sua base de dados configurações, volte para a etapa final do assistente e clique em Concluir. Agora você pode adicionar os seus arquivos para o Visual SourceSafe.

Para ligar para o Microsoft Visual SourceSafe login na janela, proporcionar a necessária senha. No Microsoft Visual SourceSafe caixa de mensagem que aparece indicando que a pasta com o mesmo nome e se não existir Visual SourceSafe necessidades para criar a pasta, clique em Sim. Em Solution Explorer, selecione o projeto. Em seguida, você precisa chamar a atenção para a Fonte Controle no menu Arquivo e clique em Editar para Check Out. Você também pode adicionar comentários na caixa Comentários do Check Out para a caixa de diálogo Editar.

Você deseja alterar a consulta para listar todos os produtos na tabela de produtos do banco de dados AdventureWorks na categoria, Modelo, e de Produto. Você não pode exigir que o Subcategoria coluna. Você precisa verificar no projeto completo em Visual SourceSafe. Para visualizar o histórico do arquivo, selecione o arquivo que está no Solution Explorer. Na caixa de diálogo Opções da História, é possível prever as datas entre as quais um arquivo foi modificado. Você pode comparar as duas versões dos arquivos que você criou. Você também pode visualizar as duas versões da consulta.

Esta demonstração mostra como usar o Visual SourceSafe para a versão gestão de buscas.

Usando o Visual SourceSafe para a versão de controle das consultas

No menu Iniciar, aponte para Todos os programas e, em seguida, aponte para Microsoft SQL Server 2005 e, em seguida, clique em SQL Server Management Studio. O SQL Server Management Studio janela aparece.

Na caixa de diálogo Conectar ao servidor, verifique se o tipo Server, servidor de nome, e as caixas Autenticação ter sido preenchido corretamente e, em seguida, clique em Connect para aceitar as configurações padrão.

No menu Ferramentas, clique em Opções. A caixa de diálogo Opções abre.

Ampliar Fonte Controlo e, em seguida, clique no plug-in Seleção página.

No atual fonte controle plug-in - box, verificar que a Microsoft Visual SourceSafe foi selecionado, e clique em OK.

Na barra de ferramentas, clique em Nova Consulta.

Em Solution Explorer, clique-direito Solution 'Solution1' (0 projetos), aponte para Adicionar e, em seguida, clique em New Project. A caixa de diálogo Adicionar Novo Projeto aparece.

Na caixa Nome do Projeto Adicionar nova caixa de diálogo, selecione SQL Server Scripts1 e, em seguida, digite CategorizeProducts, e clique em OK.

Clique com o botão direito a pasta de Consultas CategorizeProducts o projeto e, em seguida, clique em Nova Consulta. Ligue para a caixa de diálogo Database Engine aparece.

Na Ligue para Database Engine caixa de diálogo, clique em Connect para aceitar as configurações padrão.

Na lista da barra de ferramentas, clique AdventureWorks. Está ligado ao banco de dados AdventureWorks.

Na janela de busca, digite as seguintes afirmações. Selecione PC.Name como categoria, como PSC.Name subcategoria, PM.Name como modelo, p. nome como produto de Production.Product como P pleno aderir Production.ProductModel como a PM.ProductModelID PM = P. ProductModelID pleno aderir Production.ProductSubcategory CPS sobre como PSC.ProductSubcategoryID = P. ProductSubcategoryID aderir Production.ProductCategory PC como em PC.ProductCategoryID = PSC.ProductCategoryID PC.Name por fim, PSC.Name;

Selecione a consulta e, em seguida, clique em Executar.

Em Solution Explorer, clique-direito CategorizeProducts o projeto e, em seguida, clique em Adicionar a Solução Fonte Controle.

Salve o arquivo como na caixa de diálogo, navegue para a pasta e clique em Salvar CategorizeProducts para aceitar as configurações padrão.

No Bem-vindo ao Assistente Adicionar SourceSafe Database página, clique em Avançar para aceitar as configurações padrão.

Na página Seleção de banco de dados, clique em Criar uma nova base de dados, e clique em Avançar.

Na nova base de dados Localização página, clique em Procurar, navegue até a CategorizeProducts projeto, clique em OK e, em seguida, clique em Avançar.

Nome do banco de dados Connection página, clique em Avançar para aceitar as configurações padrão.

No Team Version Control Model página, clique em Avançar para aceitar as configurações padrão.

Em Concluindo o Assistente Adicionar SourceSafe Database página, clique em Concluir. O Log On a Visual SourceSafe Database log-in janela aparece.

Na caixa SourceSafe senha, digite senha, e clique em OK. No SourceSafe Adicionar a caixa de diálogo, clique em OK para aceitar as configurações padrão.

No Microsoft Visual SourceSafe caixa de diálogo que aparece indicando que Project \$ / Solution1.root não existe e pedindo a confirmação de saber se você gostaria de criá-la. Clique em Sim.

Em Solution Explorer, CategorizeProducts verificar que o projeto seja selecionado. No menu Arquivo, aponte para Control Source e, em seguida, clique em Editar para Check Out.

Na caixa Comentários do Check Out para a caixa de diálogo Editar, tipo versão preliminar e, em seguida, clique em Check Out.

Na janela de busca, digite as seguintes afirmações. Selecione PC.Name como categoria, PM.Name como modelo, p. nome como produto de Production.Product como P pleno aderir Production.ProductModel como a PM.ProductModelID PM = P. ProductModelID pleno Production.ProductSubcategory como aderir ao PSC PSC.ProductSubcategoryID = P. ProductSubcategoryID pleno Production.ProductCategory como aderir ao PC PC.ProductCategoryID = PSC.ProductCategoryID PC.Name por fim, PSC.Name;

Enquanto se aguarda Checkins No painel, clique check-in. Na caixa de diálogo Fonte Controle, clique em Verificar se para aceitar as configurações padrão.

Em Solution Explorer, selecione o arquivo SQLQuery1.sql do CategorizeProducts projeto. Em seguida, no menu Arquivo, aponte para Controle e clique em View Source História.

Na caixa de diálogo Opções da História, clique em OK para aceitar a configuração padrão. A História de dólares / Solution1.root/Solution/SQL Query1.sql aparece com uma lista de todas as versões do arquivo.

Selecione os arquivos e, em seguida, clique Dif. A diferença caixa de diálogo Opções aparece. Clique em OK.

Questões

Sua empresa tem diversificado suas atividades recentemente e as operações da empresa tornaram-se mais complexa. Como administrador do banco de dados, você sente que a empresa poderia aplicar o gerenciamento centralizado de todos os scripts SQL e usar um sistema de controle de arquivo fonte versão gestão. Você precisa convencer a administração superior a aprovar o uso deste sistema. Quais dos seguintes benefícios que você irá colocar diante, para ganhar a sua confiança?

Fonte controle prestadores armazenar a data e a hora em que o item foi criado, quando foi verificada no check-out ou, e o usuário que realizou o recurso. X

Fonte controle provedores suporte apenas o acesso exclusivo arquivo, permitindo assim que apenas um usuário em um momento de verificar arquivos e modificá-los para fora.

Fonte controle prestadores armazenar apenas a recente versão de arquivos, e apagar as versões anteriores.

Você pode recuperar dados a partir de apagamento acidental e restaurar a versão mais recente verificada em arquivo fonte controle.

Você pode usar a interface fornecida pela fonte controle prestadores arquivos em lote para automatizar a fonte controlar tarefas que você executa regularmente. X

Fonte controle prestadores conservar espaço no disco rígido controle servidores somente a fonte.X

Sua empresa utiliza Visual SourceSafe a manter o código fonte. Você precisa fazer alterações no código-fonte controlada um arquivo que é mantida para utilização compartilhada entre todos os membros da organização. Você também precisa se assegurar que outros usuários possam verificar os

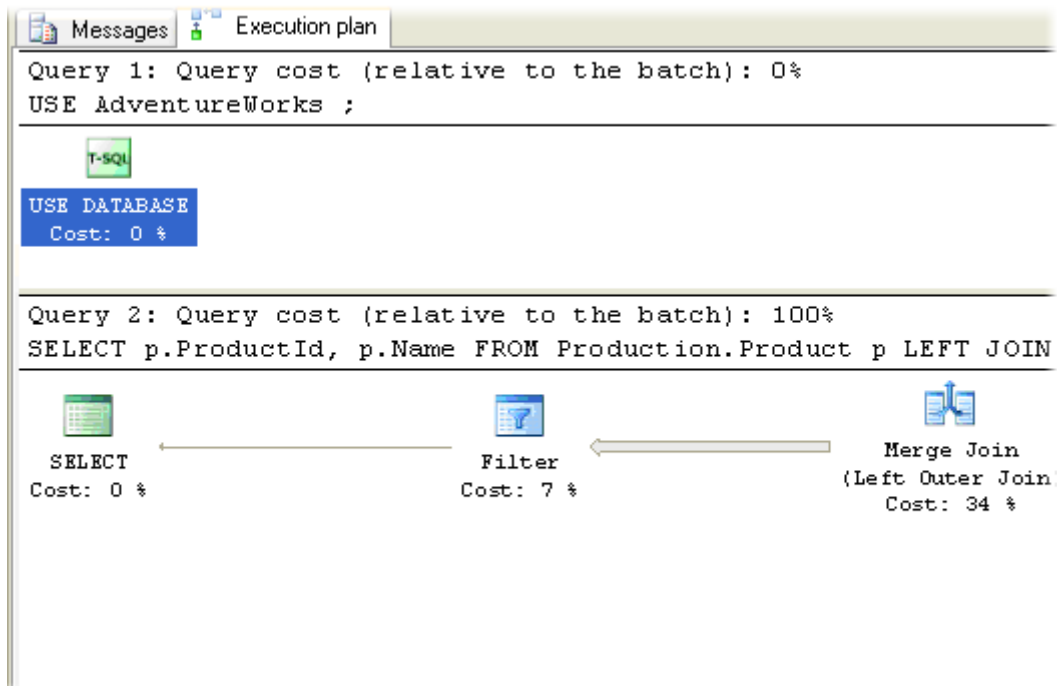
arquivos quando você trabalhar com ele. Que uma das seguintes opções em Visual SourceSafe terá que escolher para modificar o arquivo e, ao mesmo tempo permitir que outros utilizadores o acesso a ele?

Verifique no arquivo.

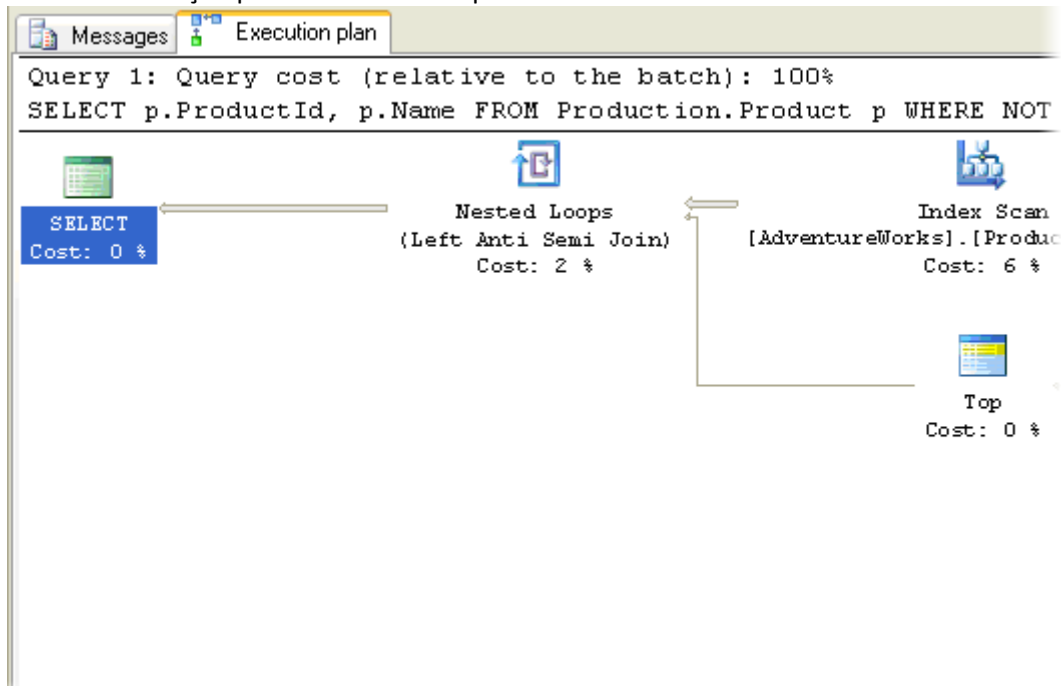
Confira o arquivo em modo compartilhado. X

Confira o arquivo no modo exclusivo.

Fundir o arquivo.



Plano de Execução para uma consulta que usa a clausula NOT EXISTS



Questões

O que acontece quando você inicia uma transação implícita no SQL Server, especificando o seguinte comando?

SET IMPLICIT_TRANSACTION

Transações obter autorização automática, sem um comando COMMIT TRANSACTION.

As declarações podem ser executadas fora do contexto de uma transação, indicando o comando ROLLBACK.

O SQL Server automaticamente inicia uma transação no momento em que primeiro executa uma instrução SQL. X

As alterações feitas aos dados na base de dados, através da utilização de várias declarações no âmbito de uma operação, são salvos automaticamente, quando você desligar a partir do servidor.

Qual dos seguintes é verdadeira sobre tabelas auxiliares?

Auxiliar tabela é apenas outro nome para tabelas temporárias.

Auxiliares tabelas não são armazenadas permanentemente na base de dados.

Auxiliares tabelas deteriorar o desempenho de uma aplicação.

Auxiliares tabelas ajudá-lo a escrever códigos mais simples e mais curtos. X

Você é administrador do banco de dados em Adventure Works. Você precisa de se preparar uma lista de todos os produtos da tabela de produtos que nunca tenham sido vendidas. Qual das seguintes palavras-chave, operadores, ou cláusulas que você pode usar em sua consulta para obter este resultado?

Cláusula HAVING

Palavra chave NOT EXISTS X

Operador UNION

Palavra chave IN X

Cláusula JOIN X

Cláusula WITH

Revisão Lab.

Neste laboratório, você criou várias funções definidas pelo usuário. Você realizou as seguintes tarefas:

Criar uma função escalar definido pelo usuário, uma tabela de valor escalar de função definida pelo usuário, e de uma declaração multi-usuário de uma função definida pelo usuário.

Executar uma declaração join referenciando uma função definida pelo usuário de multi-declaração, que você criou.

Criar uma view que permita que os usuários vejam só a partir de colunas relevantes AdventureWorks banco de dados.

Criar consultas distribuídas com diversas aplicações como o Microsoft Office Excel, Microsoft Office Access, um local remoto e SQL Server.

Executar algumas stored procedures do sistema para compreender a diferença entre usar o servidor remoto SQL Server e Microsoft Office Access ligado um servidor.

Que assistente você teria que usar para exportar ou importar dados de SQL Server Management Studio?

SQL Server Assistente de importação e de exportação.

Como você pode exportar dados para um destino, a partir de dados externos SQL Server Management Studio?

Clique no botão direito do nome do banco de dados, aponte para Tarefas e, em seguida, clique em Exportar Dados.

Quais seriam as Stored Procedures utilizadas para visualizar o código fonte de uma View?

Sp_helptext.

Opção que você usa para assegurar que quando uma linha é modificada em um ponto de vista, os dados permanecem visíveis através da visão após a modificação está empenhada?

[Check Option.](#)