Querying – Thought Experiments – Capítulo 1

1. How come you cannot use an alias you define in the SELECT list in the WHERE clause,

or even the same SELECT clause? Where can you use such an alias?

Não é possível utilizar um alias que vc definiu no SELECT na lista de WHERE pq o SELECT é lido depois do WHERE, ou seja, quando a máquina processa o WHERE, o SELECT ainda não foi executado e portanto o alias não foi criado

1. What are the differences between joins and set operators?

Os operadores UNION colocam as linhas das consultas em seguida uma da outra; os operadores JOIN fazem uma espécie de produto cartesiano, gerando o resultado baseado nos cruzamentos iguais das duas ou mais consultas

1. What could prevent SQL Server from treating a query filter optimally, meaning, from

using an index efficiently to support the filter? What other query elements could also

be affected in a similar manner and what can you do to get optimal treatment?

1. What is the difference between the ON and WHERE clauses?

O ON deve ser usado preferencialmente cruzarmos dados de tabelas diferentes, utilizamos o ON para juntarmos duas tabelas diferentes, sem controle sobre o que será retornado, podendo trazer dados das duas tabelas cruzadas; o WHERE deve ser usado para filtrarmos dados. Apesar de podermos usar ON para juntar **E** filtrar dados, a boa prática indica que devemos utilizar o WHERE para filtrar os dados e utilizar o JOIN para juntarmos os dados.

1. Explain what function determinism means and what are the implications of using non-

deterministic functions?

Funções determinísticas são aquelas que sempre retornam o mesmo resultado, se utilizadas com os mesmos parâmetros. Exemplo: a função DATEDIFF. Essa função retornará sempre o mesmo valor quando utilizados parâmetros constantes. Por exclusão, funções não determinísticas são aquelas que retornam resultados diferentes quando executadas. Exemplo de função não determinística é a função GETDATE(), que **não retornará** o mesmo resultado se executada em momentos diferentes.

O impacto de utilizar uma função que retorna valores diferentes ao longo do tempo é que esse valor será sempre diferente, mesmo que o banco continue o mesmo, outra implicação é a criação de índices. Se você cria uma coluna calculada que se utiliza de uma função não determinística, você não pode criar um índice nesta coluna, uma vez que os valores desta coluna serão diferentes ao longo do tempo

1. What are the differences between DELETE and TRUNCATE TABLE?

O DELETE é utilizado para apagar dados de um banco; é criado log de todas as linhas excluídas, facilitando assim a restauração, caso necessária. DELETE é uma clausula DML (DML, data manipulation language), ou seja, trata diretamente da manipulação, neste caso a exclusão, de linhas. A cláusula pode (e deve) ser utilizada com um filtro para a exclusão.

O TRUNCATE é mais rápido na sua execução, pois armazena **menos** logs, tornando-se mais difícil de restaurar as linhas excluídas. TRUNCATE é uma instrução DDL (DDL, data definition language) e opera sem a necessidade de filtros.

1. You need to perform a multi-row insert into a target table that has a column with an

identity property. You need to capture the newly generated identity values for further

processing. How can you achieve this?

Os valores novos, gerados pela coluna identity pode ser armazenado em uma variável para processamento posterior. A variável mais indicada neste caso seria uma variável do tipo TABLE, onde seria armazenado os índices recém gerados ou qualquer informação necessária para processamento posterior.

1. When should you use the WITH VALUES clause explicitly as part of adding a column to

a table?

Quando a nova coluna criada aceitar NULL, você deve especificar um valor em WITH VALUES, para que a as linhas existentes da coluna nova recebam o valor definido em DEFAULT; se a coluna nova não aceitar NULL, mesmo que você não especifique valores para WITH VALUES, as linhas existentes da coluna nova receberá os valores informados em DEFAULT.

Querying – Thought Experiments – Capítulo 2

Pontos de destaque deste capítulo: O operador APPLY, nos modos CROSS APPLY ou OUTER APPLY:

**O operador CROSS APPLY** avalia expressões do lado direito e do lado esquerdo e, caso o retorno da tabela da direita retorne uma linha ou conjunto de linhas vazias para a tabela da esquerda, a linha/conjunto não é retornada. O motivo do nome CROSS APPLY tem a ver com o fato deste operador funcionar como uma espécie de CROSS JOIN entre duas tabelas. As figuras abaixo ilustram bem:



Na esquerda as letras X, Y, Z funcionam como valores chaves da tabela da esquerda. F representa a expressão de tabela passadas como input pela tabela da direita e, entre os parênteses, o elemento que serve de correlação entre as duas tabelas.

Na direita, nós vemos o resultado retornado pela expressão da tabela da direita, para cada linha da tabela da esquerda.

Abaixo nós vemos o resultado do operador CROSS APPLY, onde cada linha da tabela da esquerda é combinada com o resultado retornado pela tabela da direita. Perceba que, onde linhas da tabela da esquerda que não retornaram nada na tabela da direita não são retornadas pelo CROSS APPLY.

**Expressões de tabela** são construções em T-SQL que, basicamente, são queries nomeadas. Você escreve uma query interna que retorna um resultado em tabela, você nomeia esse resultado e consulta o **resultado** para uma query externa. Os tipos de expressões de tabela são **CTE, tabelas derivadas, views e funções que retornam tabelas (table valued functions)**.

As queries internas não podem ter a cláusula ORDER BY, todas colunas precisam possuir nomes, ou seja, os seus resultados precisam ser relacionais.

**Expressões de tabela (CTE) ou tabelas temporárias?** Essa dúvida depende da sua necessidade. Se você escreve uma query e quer interagir com o resultado desta query e escreve outra query para interagir com o resultado da primeira, CTE é mais indicado pois é mais performático devido ao processamento físico da engine do SQL lidar melhor; agora, se você precisa guardar os valores da primeira query para utilizar depois, em outro contexto etc. guardar esses resultados em uma tabela temporária é mais adequado.

1. Generally, when solving tasks with T-SQL, is it more efficient to use joins or subqueries?

Depende, existem casos em que o uso de JOIN é mais performático do que a subquery e vice-versa. Em uma consulta em que são feitas agregações e acessando os mesmos dados, o JOIN tem melhor performance que a subquery, pois na subquery SQL vai criar acessos separados para cada consulta e formar as agregações. Com JOIN o acesso aos dados é feito uma única vez e as agregações feitas a partir daí.

1. What is the difference between a self-contained subquery and a correlated one?

A subquery “auto-contida” é aquele caso em que a subquery roda de forma separada e independente da query externa que a contém; uma subquery correlacionada não roda de forma separada e independente, uma vez que ela referencia e utiliza colunas da query externa que a contém.

1. In what way is the APPLY operator different than joins and subqueries? Can you pro-

vide an example when it should be used?