Documentação de estudos

Certificação Microsoft 70-761 – Queryng Data with Transact-SQL

O estudo não tem como intuito tratar de boas práticas de modelagem, por este motivo, as tabelas encontradas nos arquivos .sql cobre apenas as necessidades de executar exemplos simples para o entendimento dos recursos de consulta do SQL.

O mapa mental para os estudos da certificação pode ser encontrado no seguinte link:

<https://coggle.it/diagram/W64XqwWWyfIZHOaA/t/mind-map-70-761>

**Select**

A cláusula SELECT é utilizada para realizar consultas em SQL Server e muitos outros bancos de dados. Em SQL Server, a execução de recursos que podem ser adicionados em uma query tem a seguinte ordem:

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

**Datatype**

Ao criar uma tabela, fazer uma consulta ou conversão de valores no SQL Server, precisamos saber o *datatype* dos dados que serão manipulados para não se deparar com erros de conversão implícita ou explícita. Alguns dos tipos de dados possuem compatibilidade e podem ser convertidos implicitamente pelo próprio SQL Server.

As categorias de tipos de dados disponíveis são: Exact numerics (Unicode character strings), Approximate numerics (Binary strings), Date and time (Other data types) e Character strings.

**Funções de conversão**

Para a conversão de valores no SQL Server, podem ser utilizadas as funções CAST, TRY\_CAST, CONVERT, TRY\_CONVERT, PARSE e TRY\_PARSE.

CAST e CONVERT possuem o mesmo objetivo, mas são utilizados com sintaxes diferente. Além disso CAST foi criado posteriormente e faz parte do padrão ANSI.

As funções de conversão que iniciam com “TRY\_” fazem um tratamento ao converter os dados. Caso a conversão não seja válida, a o retorno será NULL em vez de exibir um possível erro.

**NULLs**

Ao trabalhar com valores nulos, podemos utilizar as funções ISNULL, NULLIF e COALESCE. NULL pode ser descrito como “valor inexistente” ou “nada”, pois apenas indica que não há valor naquela célula. Cada uma dessas funções possui um tratamento específico para NULL.

ISNULL é utilizado para comparar uma coluna ou variável específica e caso este seja NULL, a função retornará um valor especificado pelo usuário no segundo parâmetro da função.

NULLIF verifica se a coluna ou variável específica contém determinado valor e, caso a condição seja verdadeira, a função retorna NULL. O usuário define a coluna/variável como primeiro parâmetro da função e o valor esperado nesta coluna/variável como segundo parâmetro.

A função COALESCE recebe diferentes colunas/variáveis como parâmetro e verifica parâmetro por parâmetro até que encontre um diferente de NULL. Caso não seja encontrado um parâmetro não nulo, a função retorna NULL.

* 1. **– Operadores de conjuntos**

Há 3 operadores unários, são eles + (positivo), - (negativo) e ~ (bitwise):

Positivo: O operador unário positivo é utilizado geralmente em declaração de variáveis. Ele não afeta valores negativos, para realizar a conversão de negativo para positivo é utilizada a função ABS().

Negativo: O operador unário negativo é utilizado para conversão de valores e possui as características matemáticas básicas de conversão de operadores (ex. negativo com negativo = positivo).

Bitwise: Este operador faz a conversão bit a bit.

Ex. Na expressão 0000 0000 1010 1010, ao aplicar o bitwise a expressão será convertida para 1111 1111 0101 0101.

O operador EXCEPT e INTERSECT retornam o resultado da comparação entre consultas ou expressões. Como regra básica, as consultas precisam ter o mesmo número de colunas com o mesmo tipo de dados para que não haja erro em conversões implícitas. O EXCEPT retorna os valores existentes na primeira consulta e que não existe na segunda consulta. O INTERSECT retorna os valores existentes em ambas as consultas ou expressões.

Os operadores UNION e UNION ALL unificam os dados de consultas distintas. Como regra básica, as consultas precisam ter o mesmo número de colunas com o mesmo tipo de dados para que não haja erro em conversões implícitas. O operador UNION faz com que diversos conjuntos de resultados sejam unificados e se tornem apenas um conjunto de resultado. Ao utilizar a cláusula ALL o conjunto de resultados retorna todas as linhas de todos os resultados sem remoção de duplicatas, e caso não seja especificado, o conjunto de resultados não terá duplicatas.

* 1. **– Consultar várias tabelas usando junções**

Através da cláusula JOIN possui diferentes categorias e possibilita fazer consultas em múltiplas tabelas de um banco de dados. A regra para utilização desta cláusula é fazer comparações e junções entre tabelas que possuam relacionamentos (chaves primárias, estrangeiras ou valores correlacionados).

JOIN ou INNER JOIN é utilizado para retornar dados existentes em ambas as tabelas utilizadas na query. Esta categoria é indicada quando se quer excluir possíveis NULLs que podem existir no relacionamento.

LEFT JOIN ou LEFT OUTER JOIN une registros existentes em ambas as tabelas e retorna ainda os dados da tabela à esquerda que não possuem um relacionamento com a tabela da direita.

RIGHT JOIN ou RIGHT OUTER JOIN une registros existentes em ambas as tabelas e retorna ainda os dados da tabela à direita que não possuem um relacionamento com a tabela da esquerda.

FULL JOIN ou FULL OUTER JOIN retorna a listagem unindo as regras do inner, left e right join.

CROSS JOIN é utilizado para juntar tabelas por cruzamento e o relacionamento de chaves entre tabelas não é requerido. O cross join faz com que cada linha da tabela A tenha todos os registros da tabela B, ou vice-versa.

Ao usar JOINs, podemos utilizar também operadores AND e OR para aumentar o filtro de registros nas junções. Além destes operadores, NULLs