XMW4 / IW4
Jednoduché SELECT dotazy

Štefan Pataky

Agenda

- Elementy jazyka T-SQL
- SELECT
- FROM/JOIN
- WHERE
- ORDER BY
- GROUP BY
- HAVING

Elementy jazyka T-SQL

Elementy	Predikáty a operátory
Predikáty	IN, BETWEEN, LIKE
Srovnávací operátory	=, >, <, >=, <=, <>, !=, !>, !<
Logické operátory	AND, OR, NOT
Aritmetické operátory	+, -, *, /, %
Konkatenace	+

Elementy jazyka T-SQL

Pořadí vyhodnocení	Operátory
1	()
2	*,/,%
3	+,-
4	=, <, >, >=, <=, !=, !>, !<
5	NOT
6	AND
7	BETWEEN, IN, LIKE, OR
8	=(Přiřazení)

Elementy SELECT

Element	Výraz	Funkce
SELECT	<select seznam>	Definuje seznam vrácených sloupců
FROM	<tabulka>	Definice tabulky dotazu
WHERE	<podminky_hledani>	Filtrování řádků podle predikátů
GROUP BY	<seznam seskupeni>	Seskupení řádků do skupin
HAVING	<podmínky hledani>	Filtrování skupin podle predikátů
ORDER BY	<seznam razeni>	Seřazení výstupu

Elementy SELECT

- Pořadí vyhodnocení elementů

5: SELECT <select list>

1: FROM <table source>

2: WHERE <search condition>

3: GROUP BY <group by list>

4: HAVING <search condition>

6: ORDER BY <order by list>

SELECT

- SELECT – specifikuje seznam sloupců ze zdrojové tabulky/tabulek
- FROM – specifikuje tabulku/ pohled (BP: schema.object)

- Seznam sloupců:
  - * - všechny sloupce
  - Sloupec1, sloupec 2, …

SELECT

- Použití výpočtů
  - Skalární hodnota, vrací pouze jednu hodnotu pro jeden řádek

- ```
  SELECT UnitPrice, (UnitPrice * OrderQty)
  FROM Sales.SalesOrderDetail
  ```
  ```

SELECT

- Dotaz může vrátit i více stejných záznamů
- SELECT ve výchozím stavu obsahuje ALL
  - ```
    SELECT City --(19614 záznamů)
    FROM Person.Address;
    ```
  - ```
    SELECT ALL City --(19614 záznamů)
    FROM Person.Address;
    ```
- Pro eliminaci duplicit můžeme použít DISTINCT
  - ```
    SELECT DISTINCT City --(575 záznamů)
    FROM Person.Address;
    ```
    ```

SELECT

- Při získávaní dat T-SQL pojmenuje data podle zdrojového sloupce. To neplatí pro výpočetní sloupce
- Aliasy
  - Přejmenovaní/ pojmenovaní sloupce
  - Pomocí AS
    - ```
      SELECT SalesOrderID, UnitPrice, OrderQty AS Quantity
      FROM Sales.SalesOrderDetail;
      ```
  - Pomocí =
    - ```
      SELECT SalesOrderID, UnitPrice, Quantity = OrderQty
      FROM Sales.SalesOrderDetail;
      ```
  - Přidaní jména
    - ```
      SELECT SalesOrderID, UnitPrice, OrderQty Quantity
      FROM Sales.SalesOrderDetail;
      ```
      ```

SELECT

- Pojmenovaní tabulek v klauzule FROM
- Pomocí AS
  - ```
    SELECT SalesOrderID, UnitPrice, OrderQty
    FROM Sales.SalesOrderDetail AS SOD;
    ```
- Vynecháním AS
  - ```
    SELECT SalesOrderID, UnitPrice, OrderQty
    FROM Sales.SalesOrderDetail SOD;
    ```
- Po pojmenování lze použít alias např. v časti SELECT
  - ```
    SELECT SOD.SalesOrderID, SOD.UnitPrice, SOD.OrderQty
    FROM Sales.SalesOrderDetail AS SOD;
    ```
    ```

SELECT

- CASE rozšiřuje možnosti získaní dat
  - Použití v SELECT, WHERE, HAVING, ORDER BY
  - U SELECTu nutnost aliasovat jinak vznikne sloupec bez jména
- Jednoduchý CASE
  - ```
    SELECT ProductSubcategoryID, Name, ProductCategoryID,
    CASE ProductCategoryID
    WHEN 1 THEN 'Bikes‘
    WHEN 2 THEN 'Components‘
    ELSE 'Unknown Category' END AS CategoryName
    FROM Production.ProductSubcategory;
    ```
- Vyhledávací CASE
  - ```
    SELECT ProductSubcategoryID, Name, ProductCategoryID,
    CASE WHEN ProductCategoryID = 1
    THEN 'Bikes‘
    ELSE 'Unknown Category' END AS CategoryName
    FROM Production.ProductSubcategory;
    ```

FROM

- FROM klauzule určuje zdrojové tabulky , které budou použité v SELECTe
- Může obsahovat tabulky a operátory
- Výsledkem FROM je virtuální tabulka
- Použití aliasů pro další zpracování

FROM

Typ joinu	Popis
CROSS	Kombinuje všechny řádky z tabulek (vytvoření Kartézského součinu)
INNER	Začne vytvořením kartézského součinu, aplikuje filtr pro shodu řádků mezi tabulkami založenou na predikátu
OUTER	Začne vytvořením kartézského součinu, ponechá všechny řádky z vybrané tabulky, shodné řádky z druhé tabulky připojí k tabulce, jinak nahrazuje NULL

INNER JOIN

- Vrací pouze řádky z tabulek, které vyhovují podmínce
- Podmínka:
  - SQL-92 – ON
  - SQL-89 –WHERE
- Proč filtrovat v sekci ON a ne WHERE ? Naznačení úmyslu
- Typicky nemá žádný dopad na výkon
- Nezáleží na pořadí tabulek

```
FROM t1 INNER JOIN t2 ON
    t1.col = t2.col
```

OUTER JOIN

- Vrací všechny řádky z jedné tabulky a řádky vyhovující z tabulky druhé
- Nevyhovující řádky z druhé tabulky nahrazeny NULL
- Vrací všechny řádky z první tabulky a jenom vyhovující z druhé

```
FROM t1 LEFT OUTER JOIN t2 ON
    t1.col = t2.col
```

- Vrací všechny řádky z druhé tabulky a jenom vyhovující z první

```
FROM t1 RIGHT OUTER JOIN t2 ON
    t1.col = t2.col
```

- Vrací pouze řádky z první tabulky, které nemají shodu v druhé tabulce

```
FROM t1 LEFT OUTER JOIN t2 ON
    t1.col = t2.col
WHERE t2.col IS NULL
```

CROSS JOIN

- Kombinace všech řádku z první tabulky se všemi řádkami z tabulky druhé = Kartézský součin
- Logický základ pro INNER A OUTER JOIN
- Většinou nežádaný výsledek:
  - Tabulka čísel
  - Generovaní dat pro testování

```
SELECT  ...
FROM t1 CROSS JOIN t2
```

```
SELECT  ...
FROM t1, t2
```

SELF JOIN

- Porovnání řádku v tabulce
- Dvě stejné tabulky v klauzuli FROM – minimálně jedná musí mít alias

```
SELECT  …
FROM t1 AS e
LEFT OUTER JOIN t1 AS m
ON e.idself=m.id;
```

WHERE

- Klauzule WHERE používá predikáty
  - Musí být logická podmínka
  - jenom řádky, kterých podmínka je TRUE jsou povoleny
  - FALSE případně UNKNOWN jsou vyfiltrovány pryč
- Následuje po FROM a předchází dalším klauzulím

```
SELECT FirstName, LastName
FROM Person.Person
WHERE FirstName = N'John';
```

```
SELECT SalesOrderID, OrderDate
FROM Sales.SalesOrderHeader
WHERE OrderDate > '20140114';
```

```
SELECT SalesOrderID, OrderDate
FROM Sales.SalesOrderHeader
WHERE OrderDate > '20140114' AND OrderDate < '20150101';
```

ORDER BY

- Řadí řádky ve výsledku pro prezentaci
- Pokud není použité ORDER BY SQL negarantuje pořadí
- Poslední klauze ve zpracování
- Všechny NULL hodnoty seřadí spolu
- V ORDER BY lze použít:
  - Aliasy sloupců, názvy sloupců ze všech tabulek z FROM
  - Pozice sloupce v SELECT
  - Sloupce, které nejsou definované v SELECT
    - Pouze když DISTINCT není součástí SELECT
  - Definice pořadí ASC/DESC, vzestupně/sestupně

GROUP BY

- Vytváří skupiny řádku na základě jedinečných kombinací hodnot
- Vypočítavá sumární hodnoty pro agregační funkce
- Detaily řádků jsou po seskupení ztraceny
- Agregační funkce jsou většinou užity v SELECT pro sumarizaci. Nemusí používat jenom sloupce ze seskupení

```
SELECT CustomerID, COUNT(*) AS count_orders
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
```

```
SELECT SalesOrderID, MAX(OrderQty) as QTY
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID;
```

HAVING

- Filtrovací podmínky, kterým musí podléhat každá seskupená skupina
- Následuje po GROUP BY klauzule

```
SELECT CustomerID, COUNT(*) AS count_orders
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
HAVING COUNT(*) > 10;
```