3.T Operátory, agregačné funkcie WHERE, GROUP BY a HAVING klauzuly

1) **Operátory**
2) **Agregačné funkcie**
3) **Predikát**
4) **WHERE**
5) **GROUP BY**
6) **HAVING**
7) **Príklady**

SELECT [DISTINCT] *zoznam*
FROM *zoznam/zdroj_tabuliek*
[WHERE *podmienka*]
[GROUP BY *zoznam*
[HAVING *podmienka*]]

1) Operátory

T-SQL používa nasledujúce operátory
1) Aritmetické
2) Logické
3) Porovnávacie
4) Bitové
5) Unárne
6) Operátor priradenia
7) Operátor spájania reťazcov

1) Aritmetické
   +, -, *, /, %
2) Logické
   AND, OR, NOT, BETWEEN, IN, LIKE
   ALL, ANY <=> SOME, EXISTS
   - ALL a ANY porovnávajú skalárnu hodnotu so zoznamom (zoznam - množina hodnôt jediného stĺpca)
   - ALL/ANY vráti TRUE ak dané porovnanie je TRUE pre všetky dvojice/aspoň pre jednu dvojicu
     Syntax:

```
... WHERE LavStr >/=/< ALL (VnutDop) ...
```

     teda skalárna veličina LavStr VonkDopytu/ľavá strana porovnania sa porovnáva s každou vrátenou **hodnotou** VnutDopu
   - EXISTS vráti TRUE ak poddopyt vráti aspoň jeden **riadok**
3) Porovnávacie
   =, >, >=, <, <=, <>
   !=, !>, !<
4) Bitové – medzi dvomi celými typmi
   &, |, ^
   A ^ B (*xor*) je FALSE ak oba operandy sú buď TRUE alebo FALSE
5) Unárne
   + (Pozitívne), - (Negatívne), ~ (Bitwise NOT operátor)

Binárna reprezentácia 75 je 0000 0000 0100 1011. Vykonanie operácie bitwise NOT dáva 1111 1111 1011 0100, čo je decimal -76.

```
      a = 0000 0000 0100 1011
      ----------------------
     ~a = 1111 1111 1011 0100
```

```
create table T(i int, j int)
  insert T values(0,0)
  insert T values(0,1)
  insert T values(1,0)
  insert T values(1,1)
  insert T values(75,1)
select i, j, ~i AS 'bitwNOT', i^j AS XOR from T
```

	i	j	bitwNOT	XOR
1	0	0	-1	0
2	0	1	-1	1
3	1	0	-2	1
4	1	1	-2	0
5	75	1	-76	74

6) Operátor priradenia
   =
7) Operátor spájania reťazcov
   +

Operátory vyššej priority sa vykonajú pred operátormi nižšej prior.

- + (Pozitívne), - (Negatívne), ~ (Bitwise NOT)
- *, /, % (Modulo)
- + (Plus), + (Concatenate), - (Mínus), & (Bitwise AND)
- =, >, <, >=, <=, <>, !=, !>, !<
- ^, |
- NOT
- AND
- ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
- =

```
SELECT 2+5*2; a  -- 12
```

2) Agregačné funkcie

Agregačné funkcie vracajú jednu jedinú hodnotu (skalárnu veličinu) na základe množiny hodnôt.

Najzákladnejšie agregačné funkcie sú:

MIN, MAX, COUNT, SUM, AVG , VAR, STDEV, ...

- Agregačné funkcie majú jeden vstupný argument (z názvov stĺpcov), ale argumentom COUNT môže byť aj *. V tom prípade COUNT vráti počet všetkých riadkov, ináč iba počet nenulových hodnôt

COUNT(vek) vs. COUNT(*)

- Pred argument môžeme doplniť aj služobné slovo DISTINCT.
- Agregačné funkcie okrem COUNT ignorujú NULL (preskočia ich).
- Count v konkrétnych stĺpcoch ignoruje NULL ale pre celú tabuľku nie.
- Často sú použité spolu s GROUP BY.
- Funkcie AVG a SUM pracujú iba s numerickými stĺpcami.
- Funkcie MIN a MAX pracujú s numerickými, znakovými (character) a dátumovými stĺpcami.

Príklad (pozri nižšie):

A	B
1	10
NULL	NULL
2	NULL
3	50

3) Predikát

Predikát v matematike je Boolean-hodnotová funkcia P: X → {true, false}.
Predikát je výraz s názvami stĺpcov, ktorý pre danú n-ticu/riadok vráti
TRUE, FALSE alebo NULL/UNKNOWN.

Predikáty sú používané vo vyhľadávacích podmienkach WHERE a HAVING klauzúl (častí) dopytu, v JOIN ON podmienkach FROM klauzuly a na ďalších miestach, kde sa očakávajú boolové premenné.

4) WHERE

WHERE klauzula príkazu SELECT špecifikuje

vyhľadávaciu podmienku/filter

pre riadky, ktoré dopyt má vrátiť.

Syntax:

WHERE <vyhľadávacia podmienka**>**

kde vyhľadávacia podmienka je kombináciou niekoľkých predikátov spájaných logickými operátormi AND, OR a NOT.

WHERE, podobne ako HAVING, je filter: určuje sériu vyhľadávacích podmienok, a do výsledku sa zaraďujú iba tie riadky, ktoré spĺňajú dané podmienky.

Vyhľadávacia podmienka môže obsahovať:

- porovnávacie operátory ako: =, >, ...
- intervaly: [NOT] BETWEEN
- zoznamy: [NOT] IN
- zhodu schém: [NOT] LIKE
- IS NULL, IS NOT NULL
- = ALL, > ALL, <= ALL, ... ANY
- kombináciu týchto podmienok vďaka OR, AND, NOT

5) GROUP BY a sumarizácia

Výsledky dopytu sú zoskupené na základe stĺpcov vymenovaných v klauzule GROUP BY.
Všetky stĺpce klauzuly SELECT použité v neagregačných výrazoch, musia byť vymenované v GROUP BY klauzule. Teda, názvy stĺpcov v SELECTe, ktoré nie sú súčasťou agregačného výrazu, **musia byť aj v GROUP BY klauzule.**

- Ak nechcete zoskupovať podľa nejakého stĺpca, nedajte ho do zoznamu stĺpcov SELECTu mimo agregácie.
- Stĺpec môže byť v GROUP BY ale nie v SELECTe
- Stĺpec alebo výraz v GROUP BY môže byť aj v ORDER BY

Pomocou GROUP BY môžeme získať jednorozmernú tabuľku *početností*,

ale nie len početností. PIVOT je dvojrozmerným ekvivalentom GROUP BY.

6) HAVING

HAVING spolu s WHERE používame na filtráciu.

Having vráti riadky, v ktorých agregačná hodnota spĺňa podmienku.

HAVING je súčasťou GROUP BY a filtruje výsledok z GROUP BY.

V HAVING môže vystupovať agregácia ľubovoľného stĺpca, ale bez agregácie iba stĺpce, ktoré sú v SELECT zozname.

Agregačné výsledky treba filtrovať pomocou HAVING a nie WHERE.

7) Príklady

```
-- A) Agreg. funkcie:
-- CREATE TABLE T(i INT, j INT);
DECLARE @T TABLE(i INT, j INT);
INSERT INTO @T VALUES(3, 10);
INSERT INTO @T VALUES(NULL, NULL);
INSERT INTO @T VALUES(3, NULL);
INSERT INTO @T VALUES(4, 50);

SELECT * FROM @T;

SELECT COUNT(*) FROM @T;
SELECT COUNT(i) FROM @T;
SELECT COUNT(j) FROM @T;
SELECT COUNT(DISTINCT i) FROM @T;
SELECT SUM(DISTINCT i) FROM @T;

SELECT AVG(j) FROM @T;
SELECT AVG(j+2*i) Hah FROM @T;

-- 4, 3, 2, 2, 7, 30, 37
-- 1;6   -   6;7

-- B) Where klauzula:
-- CREATE TABLE T(i INT);
DECLARE @T TABLE(i INT);
INSERT INTO @T VALUES(1);
INSERT INTO @T VALUES(2);
INSERT INTO @T VALUES(6);
INSERT INTO @T VALUES(7);

SELECT * FROM @T WHERE i IN (1,3,6);
SELECT * FROM @T WHERE i BETWEEN 3 AND 7;
-- NULL;10;50   2;6;13   2,NULL;6,10;13,50   detto  50,13   10,6;50,13

-- C) Group By  +  Having
--CREATE TABLE T(i INT, j INT);
DECLARE @T TABLE(i INT, j INT, k INT);
INSERT INTO @T VALUES(1, 10, 5);
INSERT INTO @T VALUES(2, NULL, 4);
INSERT INTO @T VALUES(3, 50, 3);
INSERT INTO @T VALUES(4, 50, 2);
INSERT INTO @T VALUES(5, 10, 1);
INSERT INTO @T VALUES(6, 50, 2);

-- Zoznam vsetkych roznych hodnot v stlpci j:
SELECT j FROM @T GROUP BY j;

SELECT SUM(i) FROM @T GROUP BY j;
-- Lepsie takto:
SELECT SUM(i), j FROM @T GROUP BY j;
```

```
-- alebo:
SELECT j, SUM(i) FROM @T GROUP BY j;
SELECT j, SUM(i) FROM @T GROUP BY j
                                HAVING j > 10;
SELECT j, SUM(i) FROM @T GROUP BY j
                                HAVING SUM(i) > 5;
                                --HAVING sum(k) > 1; -- OK
                                --HAVING k > 1;      -- error

-- D) ALL, ANY (poddopyty)
DECLARE @U TABLE(i INT);
INSERT INTO @U VALUES(11);
INSERT INTO @U VALUES(12);
INSERT INTO @U VALUES(13);
INSERT INTO @U VALUES(14);

SELECT * FROM @T WHERE i+10 IN (SELECT * FROM @U);

SELECT * FROM @T WHERE i+10 >= ANY (SELECT * FROM @U);

SELECT * FROM @T WHERE i+10 >= ALL (SELECT * FROM @U);

SELECT * FROM @T WHERE i+10 <= ALL (SELECT * FROM @U);


USE OsobaVztah
SELECT avg(vyska), pohlavie FROM osoba
     GROUP BY pohlavie
     --HAVING avg(vyska)>170
```

Výsledky

	i	j
1	1	10
2	2	NULL
3	3	50
4	4	50

	i	j
1	1	10
2	2	NULL
3	3	50
4	4	50
5	5	10
6	6	50

	i	j
1	4	50
2	5	10
3	6	50

	i	j
1	1	10

	Pr_vyska	pohlavie
1	175.020000	m
2	161.800000	z