NULLIF - COALESCE, ISNULL, LIKE [], kvantifikátory a NOT

1) NULLIF(), COALESCE() [koö'les]
2) ISNULL() a agregácia
3) LIKE [], [^]
4) Kvantifikátory ALL, ANY (SOME), EXISTS a NOT 2

1) NULLIF, COALESCE
- COALESCE(*a*, *b*, *c*, ...) - vráti prvú ne NULL hodnotu, ináč a
- NULLIF(*a*, *b*) - vráti NULL ak *a*, *b* sa rovnajú, ináč vráti *a*

```
-- <=>
- CASE WHEN a = b THEN NULL
  ELSE a END

DECLARE @a AS CHAR
DECLARE @b AS CHAR
SET @a = 1; SET @b = 1          -- 1, NULL, NULL
SET @a = 2; SET @b = 1          -- 2, 2, 2
SET @a = ''; SET @b = ''        -- '', NULL, NULL
SET @a = NULL; SET @b = ''      -- '', NULL, NULL
SET @a = ''; SET @b = NULL      -- '', '', ''

SELECT COALESCE(@a, @b)
SELECT NULLIF(@a, @b)
---- <=> :
SELECT CASE WHEN @a = @b THEN NULL ELSE @a END

SELECT COALESCE(@a, @b, NULL) -- '' <=>:
SELECT ISNULL(@a, NULLIF(@b, NULL))
```

2) ISNULL(a, b) a agregácia - nahrádza „NULL" s hodnotou b ináč a

```
CASE WHEN a IS NULL THEN b  -- b je najcastejsie 0
  ELSE a END

-- Priemerny prijem pomocou AVG a SUM - s NULL:
SELECT AVG(mesPrijem),
       SUM(mesPrijem)/count(mesPrijem)  FROM Pacienti
---- NOT ⇔:
SELECT AVG(mesPrijem),
       SUM(mesPrijem)/count(*) FROM Pacienti
---- ⇔(podla dohody, napr. mesacna kvota na clena rodiny):
SELECT AVG( ISNULL(mesPrijem,0) ),
-- <=>: SELECT AVG( CASE WHEN mesPrijem IS NULL THEN 0 ELSE
mesPrijem end ),
       SUM(mesPrijem)/count(*) FROM Pacienti
```

3) [], [^] a LIKE

%	ľubovoľný reťazec dĺžky nula alebo viac
_	Práve jeden znak
[]	práve jeden znak z oblasti [c-e] alebo z množiny [dce]
[^]	práve jeden znak okrem znaku z oblasti [^c-e] alebo z množiny [^dce]

	idP	krstne	mesPrijem
1	1	Adam	10000
2	2	Stefan	9500
3	3	Slavo	8500
4	4	Klara	9000
5	5	Zuzana	35000
6	6	Tana	20000
7	7	Mato	28000
8	8	Zoli	32000
9	9	Misko	NULL
10	10	Janka	NULL

```
USE Poliklinika;
GO

SELECT * FROM Pacienti p
SELECT * FROM Pacienti WHERE krstne LIKE '%an%'-- Stefan, Zuzana, Tana, Jana
SELECT * FROM Pacienti WHERE krstne LIKE '_an%'        -- Tana, Jana
SELECT * FROM Pacienti WHERE krstne LIKE '[JT]an%'     -- Tana, Janka
SELECT * FROM Pacienti WHERE krstne LIKE '[JT]an[^k]%'-- Tana
SELECT * FROM Pacienti WHERE krstne LIKE '[JT]an[^ka]%'--
SELECT * FROM Pacienti WHERE krstne LIKE '%[^f]an%'-- Zuzana, Tana, Jana

SELECT * FROM Pacienti WHERE krstne LIKE '[STUVZ]an%' -- Tana
SELECT * FROM Pacienti WHERE krstne LIKE '[VZSTU]an%' -- Tana
SELECT * FROM Pacienti WHERE krstne LIKE '[S-Z]an%'   -- Tana
```

Využitie regulárnych výrazov:

```
USE tempdb
CREATE TABLE #maz ( meno VARCHAR(20) )

INSERT INTO #maz
SELECT 'FeroSpišák' UNION ALL
SELECT 'JanoBuša'    UNION ALL
SELECT 'ŠtevoČernák'
```

⇨

	meno	krstne	priezvisko
1	FeroSpišák	Fero	Spišák
2	JanoBuša	Jano	Buša
3	ŠtevoČemák	Števo	Čemák

```
SELECT meno,
     LEFT(meno, PATINDEX ('%[A-Z]%', SUBSTRING(meno, 2, 100) COLLATE Latin1_General_BIN)       ) AS
krstne,
SUBSTRING(meno, PATINDEX ('%[A-Z]%', SUBSTRING(meno, 2, 100) COLLATE Latin1_General_BIN)+1, 100) AS
priezvisko
FROM #maz
```

kde
- LEFT Vracia ľavú časť reťazca znakov so zadaným počtom znakov.
- PATINDEX vráti začiatočnú pozíciu prvého výskytu vzorky v danom výraze
- SUBSTRING vracia časť reťazca: SUBSTRING (expression ,start , length)
- Collation (COLLATE) okrem stanovenia pravidiel pre usporiadanie a porovnanie dát, taktiež určuje bitové vzorky, ktoré predstavujú znaky v sade dát. SQL Server podporuje ukladanie objektov, ktoré majú rôzne collation v jednej databáze. http://msdn.microsoft.com/en-us/library/ms143726.aspx http://msdn.microsoft.com/en-us/library/ms143515(v=sql.105).aspx

Otestujme, ktorá z rovností `l=Í` a `s=š` platí pri `collate latin1_general_BIN`:

```
select case when 'l' collate latin1_general_BIN -- alebo latin1_general_ci_as
                = 'Í' collate latin1_general_BIN
                then 'Yes'
           else 'No'
      end [l=Í?],
      case when 's' collate latin1_general_BIN
                = 'š' collate latin1_general_BIN
                then 'Yes'
           else 'No'
      end [s=š?]
```

Results | Messages

	l=Í?	s=š?
1	Yes	No

4) Kvantifikátory (operátory) ALL, ANY (SOME), EXISTS a NOT

http://msdn.microsoft.com/en-us/library/ms188336.aspx

ALL, ANY (<=>SOME)

- syntax:

skal.výraz = / <> / > / >= / < / <= **ALL / ANY** (**VD**)

ALL

- ALL porovnáva skalárny výraz s každou hodnotou zoznamu alebo čo vráti VD a vráti TRUE ak porovnanie platí pre každú dvojicu

ANY

- ANY vráti TRUE ak porovnanie platí aspoň pre jednu dvojicu

[NOT] EXISTS (VD)

= ANY ⇔ IN [⇔ NOT EXIST]

<> ALL ⇔ NOT IN [⇔ NOT EXIST ak ~~NULL~~]

EXISTS (VD)

- vráti TRUE, ak VD obsahuje aspoň jeden riadok, ktorý môže obsahovať aj samé NULL hodnoty
- na rozdiel od ALL a ANY, môžeme EXISTS – podobne ako IN - negovať s NOT. WHERE klauzula v NOT EXISTS je splnená, ak poddopyt nevráti žiadny riadok.

Poznamenáme, že NOT IN ... a NOT EXISTS ... sú drahé operácie, lebo za nimi nasledujúci poddopyt musí byť úplne preskenované, teda každý riadok sa musí skontrolovať, kým operácie IN ... a EXISTS ... je možné ukončiť skôr, teda pri kontrole toho riadku, pre ktorý podmienka je splnená.

```
USE tempdb;
GO

IF OBJECT_ID('#T') IS NOT NULL DROP TABLE #T;
GO
CREATE TABLE #T(i INT);
GO

INSERT #T VALUES(1);
INSERT #T VALUES(2);
INSERT #T VALUES(6);
INSERT #T VALUES(7);

IF OBJECT_ID('#U') IS NOT NULL DROP TABLE #U;
GO
CREATE TABLE #U(i INT);
GO
INSERT #U VALUES(11);
INSERT #U VALUES(12);
INSERT #U VALUES(13);
INSERT #U VALUES(14);

SELECT * FROM #T
SELECT * FROM #U
```

```
---- A1) 11; // 12:
SELECT i+10 FROM #T WHERE i+10 IN (SELECT * FROM #U); --⇔
SELECT i+10 FROM #T WHERE i+10 = ANY (SELECT * FROM #U);

---- B1) 16; // 17:
SELECT i+10 FROM #T WHERE i+10 NOT IN (SELECT * FROM #U); --⇔
SELECT i+10 FROM #T WHERE i+10 <> ALL (SELECT * FROM #U);
```

Zopakujme A) a B) s NULL hodnotami:

```
INSERT #U VALUES(NULL);
GO
---- A2) 11; // 12:
SELECT i+10 FROM #T WHERE i+10 IN (SELECT * FROM #U); --⇔
SELECT i+10 FROM #T WHERE i+10 = ANY (SELECT * FROM #U);

---- B2) NIC; // NIC:
SELECT i+10 FROM #T WHERE i+10 NOT IN (SELECT * FROM #U); --⇔
SELECT i+10 FROM #T WHERE i+10 <> ALL (SELECT * FROM #U);

-- riesenie problemu - IS NOT NULL:
-- B3) 16; // 17:
SELECT i+10 FROM #T WHERE i+10 NOT IN (SELECT * FROM #U WHERE i IS NOT NULL); --⇔
SELECT i+10 FROM #T WHERE i+10 <> ALL (SELECT * FROM #U WHERE i IS NOT NULL);

-------------------------------------------------------
---- OK - C1) 11 // 12 // 16 // 17
SELECT i+10 FROM #T WHERE i+10 >= ANY (SELECT * FROM #U);

---- NO - C2) nic // nic
SELECT * FROM #T WHERE i+10 >= ALL (SELECT * FROM #U);
---- riesenia - IS NOT NULL
---- OK - C3) -- 6 // 7
SELECT * FROM #T WHERE i+10 >= ALL (SELECT * FROM #U WHERE i IS NOT NULL);
```

Porovnanie [NOT] IN a [NOT] EXISTS

```
USE tempdb
--DROP TABLE t1; DROP TABLE t2;
CREATE TABLE t1 (id1 INT)
CREATE TABLE t2 (id2 INT)
INSERT t1 VALUES (1),(2),(3),(null)
INSERT t2 VALUES (1),    (3),(null)

Kým dopyty IN a EXISTS vrácajú rovnaké výsledky (dokonca aj JOIN):
SELECT t1.* FROM t1 WHERE t1.id1 IN (SELECT t2.id2 FROM t2)
SELECT t1.* FROM t1 WHERE    EXISTS (SELECT t2.*   FROM t2 WHERE t1.id1 = t2.id2)
SELECT t2.* FROM t2 LEFT JOIN t1 ON t1.id1 = t2.id2 WHERE t2.id2 IS NOT NULL

dopyty NOT IN a NOT EXISTS už nie:
SELECT t1.* FROM t1 WHERE t1.id1 NOT IN (SELECT t2.id2 FROM t2)
SELECT t1.* FROM t1 WHERE    NOT EXISTS (SELECT t2.*   FROM t2 WHERE t1.id1 = t2.id2)

Ale po odstránení NULL hodnôt, znova vrácajú rovnaké výsledky:
-- OK - equivalent
SELECT t1.* FROM t1 WHERE t1.id1 NOT IN (SELECT t2.id2 FROM t2 WHERE t2.id2 IS NOT NULL)
SELECT t1.* FROM t1 WHERE    NOT EXISTS (SELECT t2.*   FROM t2 WHERE t1.id1 = t2.id2)
                     AND t1.id1 IS NOT NULL
```

Zistite id a mená pacientov, ktorí navštívili všetkých lekárov.

⇔ Zistite všetkých pacientov, ktorí navštívili všetkých lekárov.

A - výrok, A‘ - negácia A; P=pacient, L=lekár;
A: všetky P, ktorí navštívili všetkých L
⇔ A: neexistuje P, ktorý nenavštívil všetkých L
A‘: existuje P, existuje L: P nenavštívil L

A = (A‘)‘ = Neexistuje P, neexistuje L: P nenavštívil L
(pre pacienta NIE je lekár, koho NEnavštívil)

```
USE Poliklinika;
GO

SELECT P.idP, P.krstne FROM Pacienti P
WHERE NOT EXISTS(     -- neexistuje L, koho P nenavstivil
        SELECT L.idL FROM Lekari L
        EXCEPT
        SELECT N.idL FROM Navstevy N
            WHERE N.idP = P.idP
        )
```