Práca s viacerými tabuľkami
JOIN 2 a extrémy s ALL

1) Tri základné typy JOIN
2)(Ďalšie typy JOIN)
3) NULL hodnoty a OUTER JOIN - Príklad
4) Extrémy - iba s All

1) Tri základné typy JOIN

JOIN slúži na získanie dát z dvoch alebo viac tabuliek na základe logických vzťahov medzi tabuľkami.

Typy

a) CROSS – CROSS JOIN
b) INNER – [INNER] JOIN
c) OUTER – LEFT/RIGHT/FULL [OUTER] JOIN

Syntax:

```
FROM T1 join_typ T2 [ON (join_podmienka)][WHERE ]
```

Celkový výsledok, počet vrátených riadkov JOIN dopytu predovšetkým závisí od:
- jeho typu
- podmienok v ON

ale aj napr. od
- podmienok vo WHERE klauzule.

Podmienka JOIN:

- V podmienke JOIN sa najčastejšie používa porovnávací operátor = (ale je možné použiť aj iné porovnania [redacted]).
- Podmienky sú najčastejšie založené na dvojici **PK** a **FK**.
- Čitateľnosť (napr. podmienky) zvyšuje použitie aliasov pre tabuľky, z kadiaľ sú stĺpce.

Typ JOIN-u ovplyvňuje výsledok.
join_typ:

a) `... FROM T1 CROSS JOIN T2 ...`

- CROSS JOIN vráti Karteziánsky súčin hodnôt dvoch stĺpcov - ku každému riadku jednej tabuľky pridá všetky riadky druhej.

b) `... FROM T1 JOIN T2 ON ...`
`... FROM T1 INNER JOIN T2 ON ...`

- INNER JOIN vráti iba také riadky, ktorých zodpovedajúce stĺpcové hodnoty vyhovujú podmienke v ON klauzule, teda pre každý vrátený riadok platí, že k nastávajúcemu riadku T1 existuje riadok T2, ktoré spĺňajú podmienku ON. Ostatné riadky eliminuje.

c) ... FROM T1 **out_typ** OUTER JOIN T2 ON ...

out_typ:
c1) LEFT
c2) RIGHT
c3) FULL

```
... FROM T1 LEFT OUTER JOIN T2 ON ...
... FROM T1 LEFT       JOIN T2 ON ...
... FROM T2 RIGHT      JOIN T1 ON ...
```

- T1 LEFT OUTER JOIN T2 vráti všetky riadky z tabuľky T1 (môže sa stať, že vráti všetky riadky aj z T2).
- Ak riadok z T2 nevyhovuje podmienke, do jej stĺpcov sa zapíšu NULL hodnoty.

- FULL OUTER JOIN vráti všetky riadky oboch tabuliek

Syntax ešte raz:

```
1)  SELECT * FROM T1 CROSS JOIN T2

2)  SELECT * FROM T1 JOIN T2 ON ...
    SELECT * FROM T1 INNER JOIN T2 ON ...

3   SELECT * FROM T1 LEFT JOIN T2 ON ...
    SELECT * FROM T1 LEFT OUTER JOIN T2 ON ...
```

Ku každému riadku T1 môže pridať riadky z T2

```
    SELECT * FROM T1 RIGHT JOIN T2 ON ...
    SELECT * FROM T1 RIGHT OUTER JOIN T2 ON ...

    SELECT * FROM T1 FULL JOIN T2 ON ...
    SELECT * FROM T1 FULL OUTER JOIN T2 ON ...
```

```
USE tempdb;
GO

IF OBJECT_ID ('T1') IS NOT NULL DROP TABLE T2
GO
IF OBJECT_ID ('T2') IS NOT NULL DROP TABLE T1
GO

CREATE TABLE T1( id1 INT, x CHAR(1) )
INSERT T1 VALUES(1, 'x')
INSERT T1 VALUES(2, 'y')

CREATE TABLE T2( id2 INT, a CHAR(1) )
INSERT T2 VALUES(1, 'a')
INSERT T2 VALUES(2, 'b')
INSERT T2 VALUES(3, 'c')
INSERT T2 VALUES(4, 'd')
```

T1

id1	x
1	x
2	y

T2

id2	a
1	a
2	b
3	c
4	d

Neštandardné (1=2) a štandardné použitie JOIN:

```
CROSS a INNER Product:
-- Ku kazdemu riadku T1 prida celu T2:
SELECT * FROM T1 INNER JOIN T2 ON (1=1) -- 8r <=>
SELECT * FROM T1 JOIN T2 ON (1=1)       -- 8r <=>
SELECT * FROM T1 CROSS JOIN T2

SELECT * FROM T1 JOIN T2 (T2.a = 'a' OR T2.a = 'c')  --4r
SELECT * FROM T1 JOIN T2 ON (T2.id2 = T1.id1/10)-- 2r
         --WHERE (T2.a = 'a' OR T2.a = 'c')     -- 1r


OUTER Product:
SELECT * FROM T1 LEFT JOIN  T2 ON 1 = 2

SELECT * FROM T1 RIGHT JOIN T2 ON 1 = 2

SELECT * FROM T1 FULL JOIN  T2 ON 1 = 2
```

	id1	x	id2	a
1	1	x	NULL	NULL
2	2	y	NULL	NULL

	id1	x	id2	a
1	NULL	NULL	1	a
2	NULL	NULL	2	b
3	NULL	NULL	3	c
4	NULL	NULL	4	d

	id1	x	id2	a
1	1	x	NULL	NULL
2	2	y	NULL	NULL
3	NULL	NULL	1	a
4	NULL	NULL	2	b
5	NULL	NULL	3	c
6	NULL	NULL	4	d

```
SELECT * FROM T1 LEFT JOIN T2
             ON (T2.a = 'a' OR T2.a = 'c')
SELECT * FROM T1 RIGHT JOIN T2
             ON (T2.a = 'a' OR T2.a = 'c')
```

	id1	x	id2	a
1	1	x	1	a
2	1	x	3	c
3	2	y	1	a
4	2	y	3	c

	id1	x	id2	a
1	1	x	1	a
2	2	y	1	a
3	NULL	NULL	2	b
4	1	x	3	c
5	2	y	3	c
6	NULL	NULL	4	d

Navštívené riadky a filtrácia
- Karteziánsky súčin
- Filtrácia riadkov: ON, WHERE, HAVING
- Filtrácia stĺpcov: * vs. zoznam stĺpcov

http://msdn.microsoft.com/en-us/library/ms191517.aspx
Odporúča sa dať podmienku spájania tabuliek do ON a nie do WHERE.

```
-- Pocet "prejdenych" riadkov:
SELECT * FROM T1 INNER JOIN T2
          ON (T2.id1 = T1.id1) -- 2*4
-- INNER JOIN je lepsi ako CROSS JOIN:
SELECT * FROM T1 CROSS JOIN T2
         WHERE T2.id1 = T1.id1 -- 2*4
```

2) (Ďalšie typy JOIN) http://msdn.microsoft.com/en-us/library/ms173815.aspx http://msdn.microsoft.com/en-us/library/ms191426.aspx

- INNER LOOP JOIN (Tab1 je veľmi malá)
- INNER MERGE JOIN (Tab1 je usporiadaná)
- LEFT OUTER HASH JOIN

3) NULL hodnoty a OUTER JOIN - Príklad
Vieme, že výsledok porovnávacieho predikátu, ktorý obsahuje NULL je NULL a preto sa INNER JOIN-om riadky s NULL hodnotami v príslušných stĺpcov sa nevrátia. Avšak ako sme videli, takéto riadky, teda riadky s NULL hodnotou, je možné vrátiť s OUTER JOIN-mi, čo sa v aplikáciach využíva.

Priklady – na nasledujúcej strane!

```
USE tempdb;
GO
IF OBJECT_ID ('Dni') IS NOT NULL DROP TABLE Dni;
GO
CREATE TABLE Dni(id INT, den DATETIME);
INSERT Dni VALUES(1, '2008-09-01');
INSERT Dni VALUES(2, '2008-09-03');
INSERT Dni VALUES(3, '2008-09-06');
INSERT Dni VALUES(4, '2008-09-09');
-----------------------
IF OBJECT_ID ('DniPocitadlo') IS NOT NULL DROP TABLE DniPocitadlo;
GO
CREATE TABLE DniPocitadlo(id INT NOT NULL PRIMARY KEY);
DECLARE @i INT
DECLARE @den1 DATETIME, @den2 DATETIME
SET @i = 1
SET @den1 = (SELECT MIN(d.den)-1 FROM Dni d)
---- Lepsie ale menej prehladne:
---- SET @den1 = (SELECT DATEADD(dd, -1, MIN(d.den)) FROM Dni d)
SET @den2 = (SELECT MAX(d.den) FROM Dni d)
SELECT @den1, @den2

WHILE @i <= (SELECT DATEDIFF(dd, @den1, @den2))
     BEGIN
          INSERT DniPocitadlo VALUES (@i)
          SET @i = @i + 1
     END
-----------------------
SELECT * FROM Dni
SELECT * FROM DniPocitadlo
-----------------------
```

	id	den
1	1	2008-09-01 00:00:00.000
2	2	2008-09-03 00:00:00.000
3	3	2008-09-06 00:00:00.000
4	4	2008-09-09 00:00:00.000

	id
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

```
DECLARE @den1 DATETIME
SET @den1 = (SELECT MIN(d.den)-1 FROM Dni d)

---- 1a) Dni zadane:
--SELECT id, den FROM    Dni

---- 1b) Dni vsetky - aj chybajuce:
SELECT T1.id, T2.den
FROM DniPocitadlo T1 LEFT OUTER JOIN Dni T2
                         ON T2.den = DATEADD(dd, T1.id, @den1);


-- 1c_a) Iba dni chybajuce:
SELECT T1.id, T2.id
FROM DniPocitadlo T1 LEFT OUTER JOIN Dni T2
          ON T2.den = DATEADD(dd, T1.id, @den1)
WHERE T2.den IS NULL;

-- 1c_b) Iba dni chybajuce:
SELECT T1.id, DATEADD(dd, T1.id, @den1) chyba
FROM DniPocitadlo T1 LEFT OUTER JOIN Dni T2
          ON T2.den = DATEADD(dd, T1.id, @den1)
WHERE T2.den IS NULL;
```

!!! http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html
http://blogs.technet.com/b/wpaulblog/archive/2010/02/28/simplifying-sql-server-joins-query-and-internals.aspx

	id	den
1	1	2008-09-01 00:00:00.000
2	2	NULL
3	3	2008-09-03 00:00:00.000
4	4	NULL
5	5	NULL
6	6	2008-09-06 00:00:00.000
7	7	NULL
8	8	NULL
9	9	2008-09-09 00:00:00.000

	id	id
1	2	NULL
2	4	NULL
3	5	NULL
4	7	NULL
5	8	NULL

	id	chyba
1	2	2008-09-02 00:00:00.000
2	4	2008-09-04 00:00:00.000
3	5	2008-09-05 00:00:00.000
4	7	2008-09-07 00:00:00.000
5	8	2008-09-08 00:00:00.000

2) Extrémy

- MAX, MIN
- ALL

[-DENSE_RANK() OVER(ORDER BY ... - budúca prednáška]
[- TOP(n) ... ORDER BY - po množinových operáciach]

```
USE Poliklinika;
GO
-- E0)Usporiadajte lekarov podla veku zostupne:
SELECT datNar FROM Lekari
    ORDER BY datNar DESC;   -- 5r: 1980//1970//1961//1960//1956

-- E1a) Najdite datum narodenia najmladsieho/ej lekara/ky
(maximalny datum narodenia):
SELECT MAX(datNar) najmladsi FROM Lekari;

-- E1b) Vypiste aj jeho/jej krstne a specializaciu:
-- NO, lebo napr. krstne by mal byt BUD v AGR.FUNK. ALEBO V GROUP BY:
SELECT krstne, spec, MAX(datNar) najmladsi FROM Lekari;
-- NO:
SELECT krstne, spec, MAX(datNar) najmladsi FROM Lekari
    GROUP BY krstne, spec   -- 5r
-- OK:
SELECT krstne, spec, L2.datNar FROM Lekari L2
    WHERE L2.datNar =
        ( SELECT MAX(L1.datNar) FROM Lekari L1 )

-- E1c) Riesme ulohu bez pouzitia MAX:
SELECT krstne, spec, L2.datNar FROM Lekari L2
    WHERE L2.datNar >= ALL  -- podmienka pre KAZDU/ALL DVOJICU!
        ( SELECT L1.datNar FROM Lekari L1 )

-- E2) Najdite datum narodenia druheho najmladsieho/u lekara/ku:
SELECT MAX(L2.datNar) [Druhy najmladsi] FROM Lekari L2
    WHERE L2.datNar <
        ( SELECT MAX(L1.datNar) FROM Lekari L1 )

-- E3) Najdite udaje o tretom/tej najmladsom/ej lekarovi/ke:
select krstne, datnar najmlad from lekari
  where datnar=(
    select max(datnar) najml3 from lekari where datnar <
   (select max(datnar) najml2 from lekari where datnar <
   (select max(datnar) najml1 from lekari))
                )
```

3) Extrémy iba s All

Nájdite krstné pacienta s druhým **najmenším** mesačným príjmom s použitím ALL (bez použitia MIN, MAX, TOP, ROW_NUMBER) a vypíšte aj mesPrijem.

```
use Poliklinika
SELECT mesPrijem FROM Pacienti order by mesPrijem

SELECT p4.mesPrijem FROM Pacienti p4  --  NO - 8500, 9000
WHERE p4.mesPrijem <= ALL
            (SELECT p3.mesPrijem FROM Pacienti p3   --  7 riadkov
             WHERE p3.mesPrijem >
                        (SELECT p2.mesPrijem FROM Pacienti p2   --  8500
                         WHERE p2.mesPrijem <= ALL
                                    ( SELECT p1.mesPrijem FROM Pacienti p1  --  8 riadkov
                                      WHERE p1.mesPrijem IS NOT NULL
                                    )
                        )
            )
```

```
USE Poliklinika;
GO
SELECT T.krstne, T.mesPrijem FROM -- Klara, 9000
    (SELECT p3.krstne, p3.mesPrijem FROM Pacienti p3
     WHERE p3.mesPrijem >
        (SELECT p2.mesPrijem FROM Pacienti p2 -- min
         WHERE p2.mesPrijem <= ALL
            ( SELECT p1.mesPrijem FROM Pacienti p1
              WHERE p1.mesPrijem IS NOT NULL
            )
        )
    ) T
WHERE T.mesPrijem <= ALL
    (SELECT p3.mesPrijem FROM Pacienti p3          -- 7
     WHERE p3.mesPrijem >
        (SELECT p2.mesPrijem FROM Pacienti p2      -- 1
         WHERE p2.mesPrijem <= ALL
            ( SELECT p1.mesPrijem FROM Pacienti p1 -- 8
              WHERE p1.mesPrijem IS NOT NULL
            )
        )
    );
```

T sa líši od dolného, druhého VD iba s **p3.krstne.**
Použitie WITH tabuľky by sprehľadnilo riešenie.