Úvod do MS SQL Server

1) Vytvorenie DB a Tab a príkaz `IF`
2) GROUP BY, ROLLUP a CUBE
3) Množinové operácie
4) Dátové typy a číselné, reťazcové a dátumové funkcie
5) Premenné a DECLARE, príkaz cyklu `WHILE`
6) Dočasné tabuľky #, ## a @
7) Prehľad príkazu INSERT – štyri spôsoby
8) ER diagram vzťahu 1:1
[9) CHECK obmedzenie; Common Language Runtime a SQL Server]

Ako sa nižšie presvedčíme, SQL Server používa rovnaké štandardné príkazy jazyka SQL ako MySQL. Našou prioritou dnes je poukázať na základné rozdiely medzi týmito jazykami.
V prvej polovici LS uvedieme ďalšie črty SQL Servra, ktoré sme v rámci MySQL buď nestihli, alebo MySQL nemá. Posledné týždne semestra budú venované nerelačným databázam, ako je **NoSQL**.

Prvé jednoduché príkazy s komentárom (run: select + F5):

```
SELECT 1+1 --; za zaciatkom komentara nemusi byt medzera
select /* vo vnutri prikazu */ 1
```

1) Vytvorenie DB a Tab a príkaz `IF`

Príkaz `IF` jazyka T-SQL a jeho využitie pri overení existencie DB a TAB.

a) Použitie DB_ID a OBJECT_ID

```
USE master;
IF DB_ID (N'DBMaz') IS NOT NULL SELECT 1
```

```
USE Poliklinika
IF OBJECT_ID('Lekari', 'U') IS NOT NULL SELECT 1
Select OBJECT_ID('Lekari')
```

Zoznam vybraných **objektových typov**:
- štruktúrové objektové typy U|K|F|I|C|D definujú štruktúru v DB
- kódové objektové typy V|P|FN|IF|TF|TR sa skladajú z SQL kódu.

Objektové typy:

U - user table
K - constraint, primary key (or unique constraint)
F - constraint, foreign key
I - index
C - constraint, check
D - constraint, default

V - view
P - stored procedure
FN - user-defined function, scalar
IF - user-defined function, table-valued, in-line
TF - user-defined function, table-valued, multi-statement
TR - trigger

Pozri ďalšie objektové typy: http://www.sqlusa.com/bestpractices/objecttype/

b) Použitie globálnych premenných @@x

```
-- Server name:
Select @@SERVERNAME
-- SQL Server Version:
Select @@VERSION
-- SQL Server Instance:
Select @@ServiceName
-- Current Database:
Select DB_NAME()
```

c) Použitie systémových pohľadov

```
SELECT * FROM sys.databases -- tempdb!!

USE Poliklinika;
SELECT * FROM sys.tables
```

Vytvorenie DB s kontrolou na existenciu (tri spôsoby)

```
USE master;
GO

-- IF EXISTS (SELECT * FROM sys.databases WHERE NAME = 'DBMaz') DROP DATABASE DBMaz
-- <=>
-- DROP DATABASE IF EXISTS DBMaz -- NOT sa neda
-- <=>
```

```
IF DB_ID (N'DBMaz') IS NOT NULL DROP DATABASE DBMaz;
GO
CREATE DATABASE DBmaz;
```

d) Vytvorenie tabuľky s kontrolou na existenciu (tri spôs.)

```
GO
USE DBMaz;
-- DROP TABLE IF EXISTS TabHaha -- NOT sa neda
-- <=>
IF OBJECT_ID ('TabHaha') IS NOT NULL DROP TABLE TabHaha;
GO
CREATE TABLE TabMaz (id int, meno varchar(20));
INSERT TabMaz VALUES
(1,'Ja'), (2,'Ty'), (3,'On'),
(4,'Ona'),(5,'Ono');

SELECT * FROM TabMaz;
```

2) GROUP BY, ROLLUP a CUBE

a) Standard SQL GROUP BY

Najprv ilustrujeme, že SQL SERVER používa GROUP BY iba podľa SQL Standardu, nie tak, ako MySQL (pozri šiestu prednášku zo ZS).

1a) Najdite datum narodenia najmladsieho/ej lekara/ky (maximalny datum narodenia):

```
USE Poliklinika;
SELECT MAX(datNar) najmladsi FROM Lekari;
```

1b) Vypiste aj jeho/jej krstne a specializaciu:

Nasledujúci riadok v Standard SQL je chybný:

```
-- OK - warning
SELECT krstne, spec, MAX(datNar) najmladsi FROM Lekari;

-- Riesenie:
SELECT krstne, spec, L2.datNar FROM Lekari L2
     WHERE L2.datNar =
          ( SELECT MAX(L1.datNar) FROM Lekari L1 );
```

2) Group By - With RollUp a With Cube

ROLLUP a CUBE sa využívajú pri vypočítavaní sumárnych veličín.
- ROLLUP generuje agregačné hodnoty pre **hierarchické hodnoty** vo vybraných stĺpcoch. To sme v rámci MySQL ukázali. Novinkou je:
- CUBE generuje agregačné hodnoty pre **všetky kombinacie hodnôt** vo vybraných stĺpcoch

```
-- Pokračovanie: plus sumarne poplatky podla spec.
SELECT L.Spec, N.poplatok, sum(N.Poplatok) suma
      FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
      Where N.Poplatok IN(500,200,800) --ostane Ocny, Zubny
      Group by spec, poplatok With Rollup
```

	Spec	poplatok	suma
1	Ocny	200	1000
2	Ocny	500	500
3	Ocny	NULL	1500
4	Zubny	500	1000
5	Zubny	800	1600
6	Zubny	NULL	2600
7	NULL	NULL	4100

```
-- Nahrada NULL textami SumPop a ZVsetci
SELECT CASE WHEN L.Spec IS NULL THEN 'ZVsetci'
        Else L.Spec End Spec,
       CASE WHEN N.poplatok IS NULL THEN 'SumPop'
        Else cast(N.poplatok as Varchar(10)) End Popl,
       sum(N.Poplatok) Suma
      FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
      Where N.Poplatok IN(500,200,800)
      Group by spec, poplatok With Rollup
```

	Spec	Popl	Suma
1	Ocny	200	1000
2	Ocny	500	500
3	Ocny	SumPop	1500
4	Zubny	500	1000
5	Zubny	800	1600
6	Zubny	SumPop	2600
7	ZVsetci	SumPop	4100

```
-- With Cube – kazda hodnota poplatku u specialistov sumarne
SELECT CASE WHEN L.Spec IS NULL THEN 'ZVsetci'
        Else L.Spec End Spec,
       CASE WHEN N.poplatok IS NULL THEN 'SumPop'
        Else cast(N.poplatok as Varchar(10)) End Popl,
       sum(N.Poplatok) Suma
      FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
      Where N.Poplatok IN(500,200,800)
      Group by spec, poplatok With Cube
      Order By Spec, Poplatok Desc
```

	Spec	Popl	Suma
1	Ocny	500	500
2	Ocny	200	1000
3	Ocny	SumPop	1500
4	Zubny	800	1600
5	Zubny	500	1000
6	Zubny	SumPop	2600
7	ZVsetci	800	1600
8	ZVsetci	500	1500
9	ZVsetci	200	1000
10	ZVsetci	SumPop	4100

3) Množinové operácie

a) UNION [ALL]
b) INTERSECT
c) EXCEPT

V SELECTe pred množinovou operáciou nie je dovolené použitie ORDER BY.

a) UNION [ALL]

UNION a UNION ALL operátory umožňujú spojiť viac výsledkov (dopytov) do jedného. Na rozdiel od JOIN, ktorý predovšetkým používame na spojenie stĺpcov (a pochopiteľne aj riadkov), UNION sa používa na **spojenie riadkov**, pritom:

- počet a poradie stĺpcov (ich typov) musia byť rovnaké
- dátové typy zodpovedajúcich stĺpcov musia byť kompatibilné

UNION ALL na rozdiel od UNION vráti aj duplicitné riadky.

```
USE Poliklinika;
GO
SELECT p.krstne, p.idP, 'P' typ FROM Pacienti p
UNION -- ALL  -- 15 / 15
SELECT L.krstne, L.idL, 'L' typ FROM Lekari L

SELECT p.mesPrijem FROM Pacienti p
UNION -- ALL  -- 18 / 32=10+22
SELECT n.poplatok FROM Navstevy n

-- Reakcia na NULL - nic zvlastne:
SELECT p.mesPrijem FROM Pacienti p
UNION -- ALL   -- 18 / 32
SELECT n.poplatok FROM Navstevy n
```

b) INTERSECT

Operátor INTERSECT porovnáva výsledky viac SELECT príkazov a vráti DISTINCT hodnoty.

```
SELECT p.krstne FROM Pacienti p -- 10
INTERSECT -- 2
SELECT L.krstne FROM Lekari L   -- 5

-- Ako reaguje na duplicitne hodnoty:
INSERT Pacienti(idP, krstne)
VALUES(100, 'Klara')
INSERT Lekari(idL, krstne)
VALUES(200, 'Klara')

SELECT p.krstne FROM Pacienti p -- 11
INTERSECT -- 2
SELECT L.krstne FROM Lekari L   -- 6
```

```
DELETE FROM Pacienti WHERE idP = 100 -- NO alias !!!!!!!!!!!!
DELETE FROM Lekari WHERE idL = 200

-- Reakcia na NULL - nic zvlastne, lebo vrati distinct a teda jeden NULL
SELECT p.mesPrijem FROM Pacienti p -- 10
INTERSECT -- 1
SELECT n.poplatok FROM Navstevy n
```

c) EXCEPT

Vieme, že v MySQL množinovú operáciu except môžeme riešiť pomocou

- NOT EXISTS
- NOT IN
- OUTER JOIN ... IS NULL

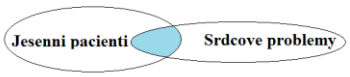
V MS SQL Server existuje operátor EXCEPT, ktorý porovnáva výsledky viac SELECT príkazov a vráti DISTINCT hodnoty.
Na rozdiel od INTERSECT nie je symetrická operácia – záleží na poradí.

(The INTERSECT operator takes precedence over EXCEPT.)

```
SELECT p.mesPrijem FROM Pacienti p -- 10
EXCEPT -- 7
SELECT 10*n.poplatok FROM Navstevy n -- 22
 -- WHERE n.poplatok IS NOT NULL -- 8 !!!
```

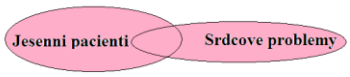
3) Príklady

```
---- 1)
---- a) Akeho lekara (idL)   navstivil pacient s id 2 (Stefan)
---- b) Akeho lekara (idL) NEnavstivil pacient s id 2 (Stefan)
---a)
SELECT N.idL FROM Navstevy N
      WHERE N.idP = 2

---b) Pokracovanie:
SELECT L.idL FROM Lekari L

EXCEPT

SELECT N.idL FROM Navstevy N
      WHERE N.idP = 2
```



Jesenni pacienti so srdcovymi problemami

Jesenni pacienti bez srdcovych problemov



Jesenni pacienti alebo pacienti so srdc.problemami

2) Jesenní pacienti so zrakovými problémami

```
Select N.idP, N.idL, Month(N.den) from Navstevy N Where Month(N.den) >= 9 -- Jesenni pac. 9
INTERSECT      -- 5
Select N.idP, N.idL, Month(N.den) from Navstevy N Where idL = 1 -- pac. so zr.prob 9
-- resp.
-- Select N.idP, N.idL, Month(N.den) from Navstevy N join lekari L on n.idL=l.idL
-- Where L.spec='Ocny' -- Pacienti so srdcovymi problemami -- 9

-- <=> strucnejsie a efektivnejsie (bez intersect):
Select N.idP, N.idL, Month(N.den) from Navstevy N Where Month(N.den) >= 9 and idL = 1
```

3) Jesenní pacienti bez zrakových problémov

```
(
Select N.idP, N.idL, Month(N.den) from Navstevy N Where Month(N.den) >= 9 -- Jesenni pac. 9
EXCEPT  -- 5
Select N.idP, N.idL, Month(N.den) from Navstevy N Where idL = 1 -- pac. bez zr. Prob. 9
)
EXCEPT -- OK - prazdna tab.
-- strucnejsie:
Select N.idP, N.idL, Month(N.den) from Navstevy N Where Month(N.den) >= 9 and idL <> 1
```

4) Jesenní pacienti alebo pacienti so zrakovými problémami

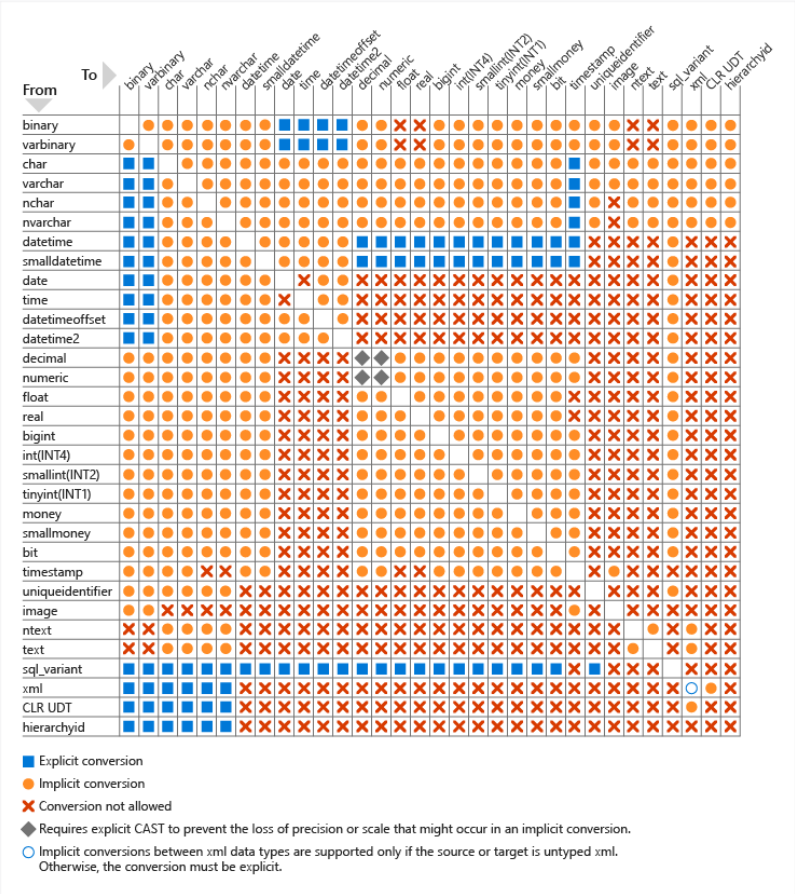
```
Select N.idP, N.idL, Month(N.den) from Navstevy N Where Month(N.den) >= 9 -- Jesenni pac. 9
Union          -- 11
Select N.idP, N.idL, Month(N.den) from Navstevy N Where idL = 1 -- Pac. so zr. Prob. 9
EXCEPT -- OK - prazdna tab.
Select N.idP, N.idL, Month(N.den) from Navstevy N Where Month(N.den) >= 9 or idL = 1
```

4) Dátové typy a číselné, reťazcové a dátumové funkcie

Cvičenie + DÚ

Číselné funkcie	https://msdn.microsoft.com/en-us/library/ms177516(v=sql.90).aspx
Reťazcové funkcie	https://msdn.microsoft.com/en-us/library/ms181984(v=sql.90).aspx
Dátumové funkcie	https://msdn.microsoft.com/en-us/library/ms186724(v=sql.90).aspx
Agregačné funkcie	https://msdn.microsoft.com/en-us/library/ms173454(v=sql.90).aspx
Dátové typy	https://msdn.microsoft.com/en-us/library/ms187752(SQL.90).aspx
Pretypovanie	https://msdn.microsoft.com/en-us/library/ms187928.aspx

SQL SERVER štandardnr používa explicitné pretypovanie na rozdiel od MySQL, ktorý skôr implicitné!



5) Premenné a príkaz cyklu WHILE

Premenné v jazyku T-SQL

Premenné môžu mať rôzne dátové typy. https://msdn.microsoft.com/en-us/library/ms187752(SQL.90).aspx

Nasledujúce kódové riadky ilustrujú

- deklaráciu premennej pomocou DECLARE
- spôsoby priradenia hodnôt do premennej pomocou SET
- spôsoby vytlačenie jej hodnoty pomocou SELECT, PRINT

```
USE tempdb;
-- IF OBJECT_ID ('T') IS NOT NULL DROP TABLE T;
DROP TABLE IF EXISTS T;
GO
CREATE TABLE T(col1 INT);
GO
```

```
DECLARE @i int;
SET @i = 2;
SELECT @i;
PRINT @i;

-- 0)
-- Go -- tu go nesmie byt - lysi sa to od MySQL
INSERT T VALUES(1), (@i)

SELECT * from T
SET @i = (SELECT * FROM T WHERE col1 <@i)
SELECT @i
```

Príkaz WHILE

https://msdn.microsoft.com/en-us/library/ms174290.aspx

Syntax

```
WHILE Boolean_expression
BEGIN
    sql_statement | statement_block | BREAK | CONTINUE
END
```

V nasledujúcom príklade vložíme do tabuľky T 5000 hodnôt.

```
USE tempdb;
IF OBJECT_ID ('T') IS NOT NULL DROP TABLE T;
GO
CREATE TABLE T(col1 INT);
GO

DECLARE @i int;
SET @i = 0;     -- 100*rand() -- seed
WHILE @i < 5000
BEGIN
     SET @i = @i+1;
     INSERT T VALUES(@i);
END;

SELECT TOP 5 * FROM T;
```

TOP(n) alebo TOP n vráti z výsledku dopytu prvých n riadkov.

ISO SQL 2003 Standard

```
SELECT TOP 3 * from T      -- MS SQL Server, alebo TOP(3)
SELECT * FROM T LIMIT 3    -- MySQL
SELECT FIRST 3 * from T    -- Ingres
SELECT * from T WHERE ROWNUM <= 3 -- Oracle
```

6) Dočasné tabuľky #, ## a @ http://msdn.microsoft.com/en-us/library/ms177399(SQL.90).aspx

Tabuľku, ako výsledok dopytu, môžeme
- vytlačiť
- kombinovať s UNION, EXCEPT, INTERSECT alebo s ďalším dopytom
- **mať v pamäti** ako **CTE** alebo premennú @ tabuľku

  `WITH T3(xx) AS (SELECT … ) SELECT …`

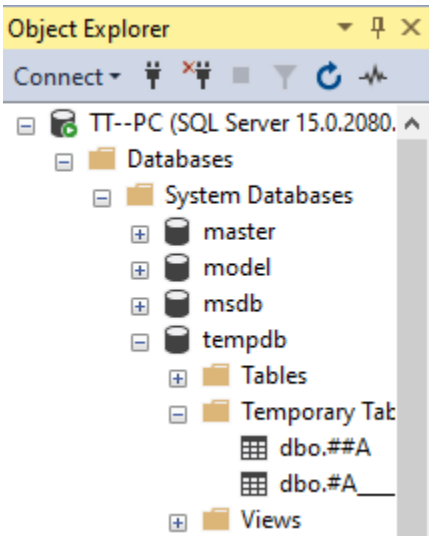
  a znovu použiť v dopytoch
- **uložiť**
  - trvalo ako VIEW (nie samotnú tabuľku, ale definujúci dopyt)

    CREATE VIEW Vmaz AS

    SELECT
  - dočasne, prechodne ako #, ## tabuľky
- sumarizovať (CUBE)

0) CREATE TABLE A(idA INT);
1) CREATE TABLE #A(idA INT); -- lokalna prechodna tabulka
2) CREATE TABLE ##A(idA INT); -- globalna prechodna tabulka
3) DECLARE @A TABLE(idA INT); -- premenna typu table
4) server.DB.owner.Table

Lokálne # a globálne ## tabuľky sú vytvorené v *tempdb* a po ukončení SQL SERVER Management Studio sa vymažú.
Kým sa lokálna #A tabuľka je viazaná k jednému *query* listu, k tej, v ktorej bola vytvorená, globálna ##A tabuľka je prístupná všetkým *quey* listom počas *prihlásenia*.




```
USE tempdb;
GO

-- 1)
IF OBJECT_ID ('#A') IS NOT NULL DROP TABLE #A;
GO
CREATE TABLE #A(col1 INT);
GO

INSERT #A VALUES(1), (2)
SELECT * FROM #A
```

Po otvorení nového (Ctrl+N) / prepnutí do iného query listu #A nie je prístupná.

```
-- 2)
IF OBJECT_ID ('##A') IS NOT NULL DROP TABLE ##A;
GO
CREATE TABLE ##A(col1 INT);
GO

INSERT ##A VALUES(10), (20)
SELECT * FROM ##A
```

Po otvorení nového/prepnutí do iného query listu ##A bude naďalej prístupná.

```
-- 3)
-- Tu namiesto SET @A ... treba INSERT @A VALUES ...
DECLARE @A TABLE(col1 INT)
INSERT  @A VALUES(100), (200)

-- Vkladanie pomocou SELECTu – pozri nasl. bod
INSERT ##A SELECT * FROM @A

SELECT * FROM @A
DELETE FROM @A

SELECT * FROM @A
-- Vkladanie pomocou SELECTu – pozri nasl. bod
INSERT @A SELECT * FROM ##A
SELECT * FROM @A

-- 4) server.DB.owner.Table Server.Database.DatabaseSchema.DatabaseObject
-- dbo = owner/schema
-- select * from [nazov srvera].poliklinika.dbo.lekari
select * from poliklinika..lekari
select * from osobaVztah..osoba

select * into #jaj from poliklinika..lekari
select * from #jaj
select * from tempdb.dbo.#jaj -- prejde
```

7) Prehľad príkazu INSERT – štyri spôsoby

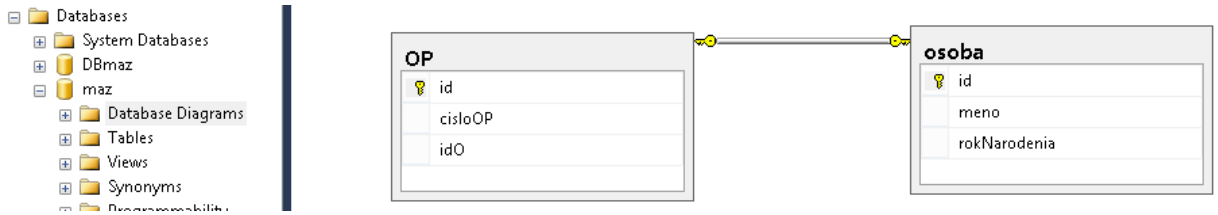
```
-- 0)
USE tempdb;
IF OBJECT_ID ('T') IS NOT NULL DROP TABLE T;
GO
CREATE TABLE T(col1 INT);
GO
```

```
-- 1)INSERT pomocou VALUES
--   INSERT [INTO] T VALUES(1)
INSERT T VALUES(1)
INSERT T VALUES(2), (3), (4)

-- 2) INSERT pomocou SELECT UNION ALL
INSERT T
SELECT 5 UNION ALL
SELECT 6 UNION ALL
SELECT 7
SELECT * FROM T

-- 3) INSERT pomocou SELECT az po CREATE TABLE T!
INSERT T SELECT * FROM T
SELECT * FROM T

-- 4) Vkladaj pomocou INTO bez CREATE a INSERT
IF OBJECT_ID ('#B') IS NOT NULL DROP TABLE #B;
-- Bez CREATE TABLE!!! FROM nutne! Prejde aj bez #
SELECT * INTO #B FROM T
SELECT * FROM #B
```

8) ER diagram vzťahu 1:1

Videli sme v ZS, že MySQL nezobrazuje správne ER diagram vzťahu 1:1.

Pravé klepnutie na +maz => pravé klepnutie na Database Diagrams => Create …



Pr.1 Vytvorte vzťah 1-1 pre Osoba a ObcianskyPreukaz

```
use master
go
drop database if exists dbmaz
go
create database dbmaz
--go -- odstran kommentar !!1
use dbmaz
```

```
create table osoba(
     id              int not null primary key,
     meno            varchar(20),
     rokNarodenia    int
)

create table OP(
     id         int not null primary key,
     cisloOP    int,
     idO        int unique FOREIGN KEY REFERENCES osoba(Id)
)
insert osoba values(1,'F', 1993),(2,'P',1987),(3,'K',1976)
insert OP values(1,45,1),(2,46,3),(3,47,2)

select * from op
```

[9) CHECK obmedzenie; Common Language Runtime a SQL Server]

- Pozri napr. vytvorenie DB OsobaVztah alebo šiestu prednášku ZS.
- Pomocou C# vďaka integrácie CLR do SQL SERVER môžeme vytvárať SP, funkcie, triggery, dátové typy.

SQL Server integration with the .NET Framework common language runtime (CLR) https://msdn.microsoft.com/en-us/library/ms131052.aspx

```
USE OsobaVztah;
GO
CREATE TABLE Osoba (
       id INT NOT NULL PRIMARY KEY,
       meno VARCHAR(10) NOT NULL,
       priezvisko VARCHAR(20) NOT NULL,
       rodne_priezvisko VARCHAR(20),
       dat_nar DATE NOT NULL,
       dat_smrti DATE,
       pohlavie CHAR(1) NOT NULL CHECK (pohlavie IN ('m','z')),
       vyska DEC(4,1) CHECK (vyska BETWEEN 30.0 AND 250.0),
       vaha DECIMAL(4,1),          -- To iste ako DEC(4,1)
       otec INT ,
       matka INT ,
       FOREIGN KEY (otec) REFERENCES Osoba(id),
       FOREIGN KEY (matka) REFERENCES Osoba(id)  -- cudzi kluc s menom
);
```

```sql
USE master;
DROP DATABASE If Exists Poliklinika;
CREATE DATABASE Poliklinika;
GO
USE Poliklinika;
GO

CREATE TABLE Pacienti
(
  idP         INT  NOT NULL PRIMARY KEY,
  krstne      VARCHAR(15),
  mesPrijem   INT
);

CREATE TABLE Lekari
(
  idL         INT  NOT NULL PRIMARY KEY,
  krstne      VARCHAR(15),
  spec        VARCHAR(20),
  datNar      DATETIME
);

CREATE TABLE Navstevy
(
  idN         INT  NOT NULL PRIMARY KEY,
--  idP             INT  NOT NULL  FOREIGN KEY REFERENCES Pacienti(idP),
  idP         INT  NOT NULL,
  idL         INT  NOT NULL,
  den         DATETIME,
  poplatok    INT,
  FOREIGN KEY (idP) REFERENCES Pacienti(idP),
  FOREIGN KEY (idL) REFERENCES Lekari(idL)
);

INSERT Pacienti VALUES      (1, 'Adam',          10000 );
INSERT Pacienti VALUES      (2, 'Stefan', 9500 );
INSERT Pacienti VALUES      (3, 'Slavo',  8500 );
INSERT Pacienti VALUES      (4, 'Klara',  9000 );
INSERT Pacienti VALUES      (5, 'Zuzana', 35000 );
INSERT Pacienti VALUES      (6, 'Tana',          20000 );
INSERT Pacienti VALUES      (7, 'Mato',          28000 );
INSERT Pacienti VALUES      (8, 'Zoli',          32000 );
INSERT Pacienti VALUES      (9, 'Misko',  NULL );
INSERT Pacienti VALUES      (10,'Janka',  NULL );

INSERT Lekari VALUES (1, 'Oto',    'Ocny',             '1960.5.5'  );
INSERT Lekari VALUES (2, 'Zoli',   'Zubny',      '1961.11.14');
INSERT Lekari VALUES (3, 'Klara','Kardiolog','1980.2.15' );
INSERT Lekari VALUES (4, 'Zuzka','Zubny', '1970.4.2'  );
INSERT Lekari VALUES (5, 'Imro',   'Interny',    '1956.11.9' );

INSERT Navstevy VALUES      (1, 1, 2, '2008.5.5', NULL );
INSERT Navstevy VALUES      (2, 2, 3, '2008.5.5', NULL );
INSERT Navstevy VALUES      (3, 6, 3, '2008.5.5', NULL);
INSERT Navstevy VALUES      (4, 4, 1, '2008.6.5', 200 );
INSERT Navstevy VALUES      (5, 5, 4, '2008.6.5', 500 );
INSERT Navstevy VALUES      (6, 7, 1, '2008.6.5', 200 );
INSERT Navstevy VALUES      (7, 6, 1, '2008.6.5', 500 );
```

```
INSERT Navstevy VALUES      (8, 8, 3, '2008.7.5', 900 );
INSERT Navstevy VALUES      (9, 2, 1, '2008.7.5', 200 );
INSERT Navstevy VALUES      (10,3, 3, '2008.7.5', 100 );
INSERT Navstevy VALUES      (11,6, 2, '2008.8.5', 700 );
INSERT Navstevy VALUES      (12,7, 2, '2008.8.5', 500 );
INSERT Navstevy VALUES      (13,6, 4, '2008.8.5', 800 );
INSERT Navstevy VALUES      (14,2, 1, '2008.9.5', NULL);
INSERT Navstevy VALUES      (15,3, 1, '2008.9.5', 200 );
INSERT Navstevy VALUES      (16,8, 1, '2008.9.5', 200 );
INSERT Navstevy VALUES      (17,9, 5, '2008.9.5', NULL);
INSERT Navstevy VALUES      (18,7, 1, '2008.10.5',300 );
INSERT Navstevy VALUES      (19,8, 4, '2008.10.5',800 );
INSERT Navstevy VALUES      (20,10,5, '2008.10.5',300 );
INSERT Navstevy VALUES      (21,1, 1, '2008.11.5',350 );
INSERT Navstevy VALUES      (22,6, 5, '2008.11.5',400 );

SELECT * FROM Pacienti;
SELECT * FROM Lekari;
SELECT * FROM Navstevy;
```

```sql
USE master;
DROP DATABASE If Exists OsobaVztah;
CREATE DATABASE OsobaVztah;
GO
USE OsobaVztah;
GO
CREATE TABLE Osoba (
       id INT NOT NULL PRIMARY KEY,
       meno VARCHAR(10) NOT NULL,
       priezvisko VARCHAR(20) NOT NULL,
       rodne_priezvisko VARCHAR(20),
       dat_nar DATE NOT NULL,
       dat_smrti DATE,
       pohlavie CHAR(1) NOT NULL CHECK (pohlavie IN ('m','z')),
       vyska DEC(4,1) CHECK (vyska BETWEEN 30.0 AND 250.0),
       vaha DECIMAL(4,1),          -- To iste ako DEC(4,1)
       otec INT ,
       matka INT ,
       FOREIGN KEY (otec) REFERENCES Osoba(id),
       FOREIGN KEY (matka) REFERENCES Osoba(id)  -- cudzi kluc s menom
);
GO
CREATE TABLE Vztah
(
  id      INT  NOT NULL PRIMARY KEY,
  id_on   INT  NOT NULL,
  id_ona  INT  NOT NULL,
  od      DATETIME NOT NULL,
  do      DATETIME
);
GO
INSERT Osoba VALUES(1, 'Adam', 'Prvy', NULL, '1918.05.11', '1968.10.01', 'm', 180.0,
80.0, NULL, NULL);
INSERT Osoba VALUES(2, 'Eva', 'Prva', 'Druha', '1919.1.9', '1988.7.22', 'z', 160.0, 60.0,
NULL, NULL);
INSERT Osoba VALUES(3, 'Zoly', 'Mudry', NULL, '1918.4.7', '19900923', 'm', 175.5, 75,
NULL, NULL);
INSERT Osoba VALUES(4, 'NASta', 'Kovacova', 'Rostova', '1928.2.5', '1965.3.11', 'z',
155.0, 99, NULL, NULL);
INSERT INTO Osoba (id, priezvisko, meno, rodne_priezvisko, dat_nar, dat_smrti, pohlavie,
vyska, vaha, otec, matka )
   VALUES(5,'Urban' , 'Jozef', NULL, '1922.10.19', NULL, 'm', 199.5, Null, NULL, NULL); -
- meno vs. priezvisko
INSERT INTO Osoba (id,meno, priezvisko, rodne_priezvisko, dat_nar, dat_smrti, pohlavie,
vyska, vaha, otec, matka )
VALUES(6, 'Maria', 'Urbanova', 'Novakova', '1937.12.8', NULL, 'z', 172.5, 57.5, 1, 2 ),
(7, 'Patrik', 'Novak', 'Novak', '1945.6.19', NULL, 'm', 182.5, 89.5, 1, 2 ),
(8, 'Patricia','Novakova', 'Haluskova', '1952.1.8', NULL, 'z', 143.5, 35, NULL, NULL),
(9, 'Michal', 'Kovac', 'Kovac', '1942.4.10', NULL, 'm', 167.0, 88, 3, 2 );
INSERT Osoba VALUES(10,'Roman', 'Kovac', 'Kovac', '1948.5.20', NULL, 'm', 179.5, 78.5, 3,
4 ), -- aj tak sa da :)
(11,'Peter', 'Horvath', 'Horvath', '1959.7.2', '2000.12.31','m', 193.0, 110.5,NULL,
NULL);
INSERT Osoba VALUES(12,'Lucia', 'Horvathova','Urbanova', '1959.1.13', NULL, 'z', 156.5,
45.5, 5, 6 );
INSERT Osoba VALUES(13,'Urban', 'Urban', 'Urban', '1957.3.31', NULL, 'm', 138.2, 24.5, 5,
6 );
```

```
INSERT Osoba VALUES(14,'DASa', 'Novakova', 'Novakova', '1970.7.17', NULL, 'z', 167.0,
55.0, 7, 8 );
INSERT Osoba VALUES(15,'Viera', 'Silna', 'Novakova', '1973.2.13', NULL, 'z', 169.5, 63.0,
7, 8 );
INSERT Osoba VALUES(16,'Vladimir','Silny', 'Silny', '1974.8.1', '2002.12.4', 'm', 175.5,
73.0, NULL, NULL);
INSERT Osoba VALUES(17,'Milena', 'Slaba', 'Slaba', '1979.9.14', NULL, 'z', 164.0, 64.0,
NULL, NULL);
INSERT Osoba VALUES(18,'Jan', 'Horvath', 'Horvath', '1982.1.16', NULL, 'm', 159.5, 65.5,
11, 12 );
INSERT Osoba VALUES(19,'Zuzana', 'Silna', 'Silna', '2002.3.1', NULL, 'z', 158.5, 60.0,
16, 15 );
INSERT Osoba VALUES(20,'Zuzana', 'Slaba', 'Slaba', '1999.12.16', NULL, 'z', 171.5, 54.5,
16, 17 );
--INSERT Osoba VALUES(21,'Zuzana', 'Prava', 'Prava', '1990.11.26', NULL, 'z', 170.5,
60.5, 16, 17 );
--INSERT Osoba VALUES(22,'Zuzana', 'Lava', 'Lava', '1931.01.14', NULL, 'z', 195.5, 58.5,
16, 17 );
--INSERT Osoba VALUES(23,'Zuzana', 'Stredna', 'Stredna', '1945.04.08', NULL, 'z', 150.5,
87, 16, 17 );
GO
INSERT INTO Vztah VALUES (1,1, 2, '1937.6.1',  '1967.5.11' );
INSERT Vztah VALUES (2,3, 2, '1967.5.12', '1988.7.22' );
INSERT Vztah VALUES (3,3, 4, '1938.12.2', '1965.3.11' );
INSERT Vztah VALUES (4,5, 6, '1953.11.11', NULL       );
INSERT Vztah VALUES (5,7, 8, '1970.7.22', '1975.9.1'  );
INSERT Vztah VALUES (6,11,12,'1980.3.4',  '2000.12.31');
INSERT Vztah VALUES (7,16,15,'1997.7.31', '2002.12.4' );

SELECT * FROM Vztah;
SELECT * FROM Osoba;
```