# Příkazy jazyka SQL ve VFP a na SQL Serveru

*Hana Horová,* *hana.horova@daquas.cz**, seminář červen 2010*

První část je přibližným opakováním přednášky Nejen SELECTem živ je člověk z DevCon 2009. Druhá část představuje možnosti využití příkazu SELECT rovnou k  výstupům a uplatnění některých obohacení tohoto příkazu, která přinesl SQL Server 2005 a 2008.

V materiálech ke stažení jsou dva adresáře s příklady. Adresář fox obsahuje příklady v souborech .prg a databázový kontejner se třemi tabulkami. Adresář sql obsahuje příklady v souborech .sql. Když spustíte skript generuj\_data.sql, založí v databázi, k níž budete právě připojeni, tři tabulky a naplní je daty.

### Ukázková data



Data popisují silně zjednodušený příklad podnikání v oboru finančnictví. Pan Vydřiduch má firmu, která půjčuje peníze. V tabulce Prodejci je on, jeho přímí podřízení Harpagon a Lakomec a pan Lakomec má ještě další tři podřízené: Skrblíka, Brzobohatou a Hrabivého, tedy celkem šest řádků a tři sloupce. IdP je primární klíč typu integer s automaticky zvyšovanou hodnotou, čili Autoincremental ve VFP a Identity na SQL Serveru. Osoba, to je jméno prodejce, a IdN je ukazatel do stejné tabulky – obsahuje IdP přímého nadřízeného dané osoby.

V tabulce Klienti jsou ti, kdo si peníze půjčují[[1]](#footnote-1). Má primární klíč IdK (opět Autoincremental) a dále sloupce jméno, příjmení, datum narození, ulice, PSČ a obec; v obou tabulkách chybí telefon, e-mail, poznámka apod. Datum narození potřebujeme proto, abychom rozlišili např. otce a syna, kteří se jmenují stejně a bydlí na stejné adrese.

Třetí tabulka Obnosy obsahuje uskutečněné finanční operace. Má primární klíč IdO, cizí klíče IdP, IdK a k tomu sloupce obnos, datum půjčky, datum splatnosti a datum úhrady. Úrok se rovná počtu kalendářních dnů mezi datem půjčky a datem úhrady a obnos je pro jednoduchost vždy v celých stokorunách. Pan Vydřiduch neuznává žádné částečné splátky, buď klient přinese celou úhradu včetně úroku, nebo přijde zaplatit zítra, ovšem úrok mu naroste o další procento. Datum splatnosti je tam hlavně proto, aby bylo zřejmé, kdy začít klienta upomínat.

### Příkazy

Na úvod dvě zkratky: DDL a DML. DDL znamená Data Definition Language a asi nejčastěji využijeme příkazy CREATE TABLE pro založení nové tabulky, ALTER TABLE pro změnu vlastností tabulky nebo jejích sloupců a DROP TABLE pro odstranění tabulky. (Na SQL Serveru je příkazů jazyka DDL podstatně víc, ale my si zde představíme jenom tyto tři.) DML znamená Data Manipulation Language a patří sem příkazy INSERT, který přidává do tabulky nové řádky, UPDATE pro aktualizaci hodnot v řádcích a DELETE, který řádky maže. Někteří autoři sem řadí i příkaz SELECT, jiní mu vyhradili speciální kategorii DQL – Data Query Language.

### CREATE TABLE

Začneme příkazem CREATE TABLE; abychom mohli zpracovávat data, musíme je napřed někam uložit, a k tomu potřebujeme tabulku.

Když se podíváme do nápovědy na syntaxi příkazu CREATE TABLE, působí na první pohled trochu odstrašujícím dojmem. Pokud bychom ale umazali všechny nepovinné fráze, zjistíme, že ten zbytek je velmi jednoduchý. Celý příkaz se dá rozdělit na tři části.

CREATE TABLE název\_tabulky
*[„obecné“ vlastnosti tabulky]*

( seznam definic sloupců

 [, omezení tabulky] )

„Obecné“ vlastnosti tabulky jsou čistě foxovou záležitostí, na SQL Serveru následuje za názvem tabulky rovnou výčet sloupců.

*[FREE]* Když chceme založit volnou tabulku a máme otevřenou nějakou databázi, musíme použít klíčové slovo FREE; když žádná databáze otevřena není, nemusíme slovo FREE zadat a přesto vznikne volná tabulka.

*[CODEPAGE = nCodePage]* umožňuje založit tabulku s jinou kódovou stránkou

*[NAME LongTableName]* platí jen pro tabulky v databázi

**Definice sloupce** vypadá ve VFP takto:

název\_sloupce datový\_typ [( délka [, des\_místa] )]

 [NULL | NOT NULL]

 [AUTOINC [NEXTVALUE počátek [STEP krok]]] | [DEFAULT výraz]

 [omezení sloupce]

přičemž omezení sloupce může být některé nebo všechna z následujících:

 [CHECK log\_výraz [ERROR text]]

 [PRIMARY KEY | UNIQUE [COLLATE collate\_sequence]]

 [REFERENCES název\_tabulky [TAG název\_složky]]

Na SQL Serveru se syntaxe maličko liší:

název\_sloupce datový\_typ [( délka [, des\_místa] )]

 [COLLATE collate\_sequence]

 [NULL | NOT NULL]

 [IDENTITY [(počátek, krok)] | [DEFAULT výraz]

 [omezení sloupce]

Každé omezení sloupce zde můžeme i pojmenovat.

 [CONSTRAINT název\_omezení]

 PRIMARY KEY | UNIQUE

 | [FOREIGN KEY]

 REFERENCES název\_tabulky [(název\_sloupce)]

 | CHECK (log\_výraz)

Za definicemi sloupců mohou následovat omezení tabulky; jsou v podstatě stejná jako omezení sloupce až na to, že v indexových výrazech, podmínkách apod. může figurovat více sloupců a že zde můžeme zadat více indexů UNIQUE a více relací REFERENCES (na SQL Serveru i více podmínek CHECK).

### Poznámky

Počet sloupců tabulky: ve VFP 9 až 255, na SQL Serveru až 30000.

Datový typ: ve starších verzích FoxPro jednopísmenná zkratka, na SQL Serveru slovní pojmenování, ve VFP 9 lze obojí.

Pro volné tabulky lze použít jen fráze NULL resp. NOT NULL a UNIQUE, všechny ostatní vyžadují tabulku v databázi.

Sloupec s automaticky zvyšovanou hodnotou nesmí být NULL. Ve VFP musí být typu integer. Na SQL Serveru může být i decimal resp. numeric (s 0 des. míst); smí být v tabulce jen jednou.

DEFAULT může být konstanta, skalární funkce (systémová i uživatelská) nebo NULL. Na SQL Serveru se nesmí odkazovat na jiné sloupce.

PRIMARY KEY smí být v tabulce jen jednou, nesmí připouštět hodnoty NULL. UNIQUE neznamená to co v příkazu INDEX, vytvoří kandidátní index; tabulka jich může mít víc.

Různé sloupce smí mít různé COLLATE (na SQL Serveru se COLLATE vztahuje nejen na řazení, ale i na porovnávání řetězců).

REFERENCES vytvoří trvalou relaci do „číselníkové“ tabulky – musí být ve stejné databázi; pokud neuvedeme název\_složky resp. název\_sloupce, použije se její primární index (nemá-li ho, dojde k chybě). Ve VFP se sice v nové tabulce vytvoří pravidelný index podle daného sloupce, ale mohou v něm být i hodnoty, které se v „číselníkové“ tabulce nevyskytují. Na SQL Serveru funguje REFERENCES opravdu jako omezení, připouští jen hodnoty z „číselníkové“ tabulky nebo NULL. Relace může být nastavena i do téže tabulky (self-reference).

CHECK sloupce se nesmí odvolávat na jiný sloupec, CHECK tabulky na jinou tabulku.

### Dočasné tabulky

Ve FoxPro se dočasná tabulka označuje také slovem „cursor“ a založí se výběrem SELECT … INTO CURSOR nebo příkazem CREATE CURSOR. V prvním případě vznikne dočasná tabulka jen ke čtení, pokud (od verze 8) nepoužijeme klíčové slovo READWRITE. Příkaz CREATE CURSOR zakládá zcela regulérní přepisovatelnou tabulku, pouze není uložena v souboru .dbf, nýbrž .tmp. Má stejnou syntaxi jako CREATE TABLE až na REFERENCES – kurzor se nemůže účastnit relace.

Na SQL Serveru jsou dočasná tabulka a kurzor dvě odlišné záležitosti. Dočasná tabulka je běžná tabulka, až na to, že její název začíná znakem #. SQL Server ji založí nikoliv v aktuální databázi, nýbrž v systémové databázi Tempdb. Dočasná tabulka vznikne výběrem SELECT … INTO #tabulka nebo příkazem CREATE TABLE #tabulka. Další možnosti jsou proměnná typu tabulka nebo tzv. CTE (Common Table Expression, obecný tabulkový výraz). Ani zde se dočasná tabulka nemůže účastnit relace.

Naproti tomu kurzor je technologie, která přinutí SQL Server k sekvenčnímu zpracování vybraných řádků. To je něco pro foxku a další systémy XBase zcela přirozené, zatímco pro SQL Server je to naopak naprosto nepřirozené. Kurzory proto mají svou vlastní sadu příkazů, jsou velice „žravé“ co do spotřeby serverových prostředků a všichni na slovo vzatí odborníci na SQL Server důrazně doporučují používat je jen tehdy, když to opravdu nejde jinak.

### INSERT

V této chvíli odbočíme z jazyka DDL do DML, abychom si dokázali do tabulky uložit nějaká data. Příkaz INSERT má na SQL Serveru dvě varianty, ve VFP tři. Začneme tou ryze foxovou, která vlastně nepatří do jazyka SQL:

*INSERT INTO název\_tabulky FROM ARRAY název\_pole*

 *| FROM MEMVAR | FROM NAME název\_objektu*

Přidáváme-li do tabulky obsah pole, musí mít stejný počet sloupců, jako má tabulka, a datové typy sloupců pole a tabulky si musí odpovídat. Do tabulky se přidá tolik řádků, kolik jich má dané pole. Zbylými dvěma frázemi se přidá do tabulky 1 řádek a naplní se hodnotami paměťových proměnných resp. vlastností objektu, které mají stejné názvy jako sloupce tabulky.

„Patřičné“ varianty příkazu INSERT jsou

INSERT INTO název\_tabulky [(seznam sloupců])]

 VALUES (seznam výrazů)

INSERT INTO název\_tabulky [(seznam sloupců])]

 příkaz\_SELECT

První z nich přidá do tabulky 1 řádek (SQL Server 2008 dovoluje zadat několik seznamů a přidat tak naráz několik řádků), druhá tolik řádků, kolik jich vrátí zadaný příkaz SELECT. Seznam sloupců není povinný; neuvedeme-li ho však, musí seznam výrazů resp. výsledek příkazu SELECT obsahovat hodnoty pro všechny sloupce tabulky v tom pořadí, jak byly definovány ve struktuře tabulky.

Pokud seznam sloupců zadáme, nemusí v něm být všechny sloupce tabulky. Seznam výrazů resp. výsledek příkazu SELECT musí odpovídat zadanému seznamu sloupců. Nezadaným sloupcům se přiřadí hodnota podle toho, jak byly definovány: pro sloupce s frází AUTOINC resp. IDENTITY to bude nejbližší další hodnota, pro sloupce s frází DEFAULT hodnota příslušného výrazu, pro ostatní sloupce hodnota NULL (pokud ji připouštějí) nebo – jen ve VFP! – prázdná hodnota daného typu.

### Poznámky

Obsahuje-li tabulka sloupec s automaticky zvyšovanou hodnotou, musíme zadat seznam sloupců. Tomuto typu sloupce totiž **nesmíme** přiřadit hodnotu, je přístupný jen ke čtení a INSERT by tedy skončil chybou.

Je dobré zvyknout si zadávat seznam sloupců pro INSERT v programu. Jednak tím zvyšujeme samodokumentující schopnost programu, a hlavně: pokud vynecháme seznam sloupců a změní se struktura tabulky, další vyvolání příkazu INSERT pravděpodobně skončí chybou.

INSERT je lepší než „staré osvědčené“ APPEND BLANK a REPLACE resp. GATHER: zamyká jen hlavičku souboru a vkládá rovnou hodnoty, takže indexy se aktualizují jen jednou.

### UPDATE

UPDATE cílová\_tabulka

 SET název\_sloupce = výraz [, ...]

 [FROM zdrojové\_tabulky]

 [WHERE log\_výraz]

Nepovinná fráze FROM má stejnou syntaxi jako v příkazu SELECT; foxařům je k dispozici až ve VFP 9. Aktualizovat je ale možno vždy jen sloupce **jedné** tabulky.

Chybí-li fráze WHERE, provedou se změny ve všech řádcích tabulky.

### DELETE

DELETE [cílová\_tabulka]

 [FROM zdrojové\_tabulky]

 [WHERE log\_výraz]

Řádky se odstraňují vždy jen z **jedné** tabulky. Je-li tedy ve frázi FROM více tabulek, musíme povinně uvést cílovou tabulku.

Nepovinná fráze FROM má stejnou syntaxi jako v příkazu SELECT; foxařům je k dispozici až ve VFP 9.

Pozor! Chybí-li fráze WHERE, odstraní se všechny řádky tabulky!

Důležitý rozdíl mezi VFP a SQL Serverem: VFP odstraňuje řádky ve dvou krocích – příkazem DELETE je pouze označí k odstranění, lze je „vrátit“ příkazem RECALL, z tabulky se doopravdy odstraní až příkazem PACK. Na SQL Serveru se řádky odstraní rovnou příkazem DELETE.

### ALTER TABLE

Protože má příkaz ALTER TABLE širší náplň práce než CREATE TABLE, je jeho syntaxe ještě o poznání delší. Můžeme si ji ale rozčlenit na několik různých variant podle toho, jaké změny chceme v tabulce provést.

#### **Přidání sloupce**

Jako u příkazu CREATE TABLE.

ALTER TABLE název\_tabulky ADD definice\_sloupce

#### **Odebrání sloupce**

ALTER TABLE název\_tabulky DROP COLUMN název\_sloupce

#### **Přejmenování sloupce**

*ALTER TABLE název\_tabulky RENAME COLUMN starý\_název TO nový\_název*

Frázi RENAME má jen foxová syntaxe. Na SQL Serveru je nutno vyvolat uloženou proceduru sp\_rename.

#### **Změna definice sloupce ve VFP**

ALTER TABLE název\_tabulky ALTER COLUMN název\_sloupce

 datový\_typ [( délka [, des\_místa] )]

 [NULL | NOT NULL]

 [SET DEFAULT výraz]

 [SET CHECK log\_výraz]

 [DROP DEFAULT]

 [DROP CHECK]

#### **Změna definice sloupce na SQL Serveru**

ALTER TABLE název\_tabulky ALTER COLUMN název\_sloupce

 datový\_typ [( délka [, des\_místa] )]

 [NULL | NOT NULL]

 [ADD CONSTRAINT omezení sloupce]

 [DROP CONSTRAINT název\_omezení]

přičemž omezení sloupce může být jako u CREATE TABLE (a zde navíc DEFAULT).

#### **Změna omezení tabulky ve VFP**

ALTER TABLE název\_tabulky

 [SET CHECK log\_výraz] [DROP CHECK]

 [ADD PRIMARY KEY výraz TAG název\_složky

 [COLLATE collate\_sequence]] [DROP PRIMARY KEY]

 [ADD UNIQUE výraz TAG název\_složky

 [COLLATE collate\_sequence]]] [DROP UNIQUE TAG název\_složky]

 [ADD FOREIGN KEY [výraz] TAG název\_složky

 REFERENCES název\_tabulky [TAG název\_složky]]

 [DROP FOREIGN KEY TAG název\_složky [SAVE]]

#### **Změna omezení tabulky na SQL Serveru**

ALTER TABLE název\_tabulky

 [ADD CONSTRAINT omezení tabulky]

 [DROP CONSTRAINT název\_omezení]

přičemž omezení tabulky může být jako u CREATE TABLE.

### Poznámky

Je lepší provádět změny postupně, tj. volat ALTER TABLE několikrát za sebou.

Při změně definice sloupce (ALTER COLUMN) je nutno znovu zadat datový typ i v případě, že se nemění.

Není možné zmenšit rozměr sloupce, který už obsahuje data, na kratší hodnotu, než je maximální délka uložených dat.

Klíčovým slovem NOVALIDATE ve VFP resp. NOCHECK na SQL Serveru lze určit, že se nemá provést zpětná kontrola, zda data v tabulce odpovídají novým omezením.

Chceme-li odebrat nebo přejmenovat sloupec, který je součástí indexu nebo validační podmínky, musíme napřed odebrat příslušný index nebo podmínku.

Přidáváme-li sloupec s automaticky zvyšovanou hodnotou do tabulky, která už má nějaké řádky, je chování VFP a SQL Serveru rozdílné: ve VFP zůstanou dosavadní řádky beze změny a automatické číslování začne až od dalšího přidaného řádku; SQL Server dosadí do všech řádků automaticky zvyšovanou hodnotu.

### DROP TABLE

DROP TABLE název\_tabulky

### SELECT

**Základní syntaxe:**

SELECT sloupce
 [ INTO výstup ]
 FROM tabulka
 [ JOIN tabulka ON podmínky … ]
 [ WHERE podmínky ]
 [ GROUP BY sloupce ]
 [ HAVING podmínky ]
 [ ORDER BY sloupce ]
 [ … ]

**Standardní pořadí zpracování:**

1. FROM tabulky
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT sloupce
6. ORDER BY

Protože výběr sloupců probíhá až po zpracování filtrovacích podmínek a agregací, nemůžeme ve frázích WHERE a GROUP BY použít alias sloupce, ale vždy musíme rozepsat celý výraz. (Ve VFP se výběr sloupců provádí jako třetí v pořadí, před GROUP BY a HAVING, proto můžeme pro GROUP BY uvést např. čísla sloupců.)

**Typy spojení:**

[ INNER ] JOIN

Vnitřní spojení, z obou tabulek se vyberou jen ty řádky, které mají na druhé straně protějšek.

LEFT [ OUTER ] JOIN

RIGHT [ OUTER ] JOIN

FULL [ OUTER ] JOIN

Vnější spojení; LEFT JOIN říká, že z tabulky vlevo, tedy před slovem JOIN, se mají vybrat všechny řádky, i ty které nemají v druhé tabulce protějšek, z tabulky vpravo se vyberou jen ty, které protějšek mají. U RIGHT JOIN je tomu naopak a u FULL JOIN se vyberou všechny řádky z obou tabulek

Někdy se můžeme setkat s termínem SELF JOIN. Je to spojení tabulky se sebou samou a může být jak INNER, tak OUTER. Protože tu máme jednu tabulku dvakrát, musíme povinně uvést lokální alias.

Vybíráme-li ze dvou tabulek a nezadáme spojovací podmínku, vznikne tzv. CROSS JOIN – každý řádek první tabulky bude spojen s každým řádkem druhé tabulky (uplatní se však filtrovací podmínky). To lze využít například ke generování testovacích dat nebo k doplnění i takových výstupních řádků, které by jinak nevznikly. Pozor, výsledný výběr může být opravdu objemný!

**NULL:**

NULL znamená „neznámá hodnota“. Pamatujme na to, že má-li některý člen výrazu (součtu, zřetězení atd.) hodnotu NULL, vrací ji i celý výraz. K otestování hodnot NULL nikdy nepoužíváme = NULL nebo <> NULL, vždy jen IS NULL nebo IS NOT NULL! Ve VFP máme k dispozici také funkci ISNULL(par), která vrací hodnotu .T., má-li parametr hodnotu NULL.

Pro usnadnění dalšího zpracování často budeme chtít, aby ve výsledku místo hodnoty NULL byla jiná hodnota stejného datového typu, jako jsou ostatní hodnoty v daném sloupci. Ve FoxPro k tomu slouží funkce NVL(). Má 2 parametry; pokud má první z nich hodnotu ne-NULL, použije se jeho hodnota, jinak se použije hodnota druhého parametru. Na SQL Serveru slouží ke stejnému účelu funkce ISNULL() nebo COALESCE(). Navíc je zde funkce NULLIF(), která vrací hodnotu NULL, mají-li oba parametry stejnou hodnotu.

**Poddotazy:**

Poddotaz je příkaz SELECT uvnitř jiného příkazu SELECT, musí být uveden v závorkách.

Může se vyskytovat:

* ve frázi WHERE jako součást filtrovací podmínky – ve všech verzích FoxPro (až od VFP9 smí být více úrovní vnoření)
	+ výraz porovnání ALL (poddotaz) | ANY (poddotaz) | SOME (poddotaz)
	+ výraz [NOT] IN (poddotaz)
	+ [NOT] EXISTS (poddotaz)
* ve frázi FROM místo tabulky – až od VFP9; musí mít lokální alias
* v seznamu výstupních sloupců – až od VFP9; poddotaz použitý tímto způsobem smí vracet jediný sloupec a maximálně jeden záznam; nevybere-li žádný záznam, bude výslednou hodnotou NULL

Poddotazy se dělí na korelované a nekorelované. Korelovaný je poddotaz, který se odkazuje na nějaký sloupec z tabulek v hlavním dotazu, obvykle ve své frázi WHERE. Nekorelované poddotazy takto závislé nejsou, proto si je může hlavní SELECT „předzpracovat“, zatímco korelované poddotazy musí skutečně provádět zvlášť pro každou hodnotu daného sloupce. Z toho plyne, že korelované poddotazy jsou pomalejší.

**Agregace:**

Agregační funkce COUNT, SUM, AVG, MAX, MIN umožňují sdružit několik vstupních řádků do jednoho výstupního. Parametrem agregační funkce je sloupec vstupní tabulky nebo nějaký výraz zahrnující sloupec. Hodnoty NULL se do agregačních funkcí nezahrnují. Jedinou výjimkou je funkce COUNT – je-li parametrem \* a nikoliv název sloupce, spočítají se i řádky s hodnotou NULL. Agregační funkce nelze vnořovat.

Do fráze GROUP BY uvedeme seznam sloupců, podle jejichž hodnot se mají vybrané řádky seskupit. Všechny vstupní řádky, které se v těchto sloupcích shodují, budou zahrnuty do jednoho výstupního řádku. Musí zde být všechny sloupce vyjmenované za slovem SELECT s výjimkou agregačních funkcí (ale starší verze VFP to nekontrolují).

Chceme-li ve výsledném výběru jen skupiny splňující určitou podmínku, použijeme klíčové slovo HAVING.

Vynecháme-li frázi GROUP BY, bude výsledkem jediný záznam s agregovanou hodnotou za celou tabulku.

Na SQL Serveru můžeme pomocí frází WITH ROLLUP nebo WITH CUBE přidat do výsledného výběru řádky s mezisoučty. K jejich odlišení od „základních“ řádků slouží funkce GROUPING().

### Některá obohacení příkazu SELECT od SQL Serveru 2005

**ROW\_NUMBER():**

ROW\_NUMBER() OVER ( [ PARTITION BY sloupce ] ORDER BY sloupce )

Tato funkce očísluje výstupní řádky seřazené podle fráze ORDER BY, číslování začíná od 1 v rámci každé skupiny dané frází PARTITION BY.

Další „ranking functions“ (syntaxe je stejná jako u ROW\_NUMBER()):

* DENSE\_RANK() čísluje skupiny se stejnou hodnotou daného sloupce, další skupina má číslo o 1 vyšší než předchozí skupina
* RANK() čísluje skupiny se stejnou hodnotou daného sloupce, další skupina má číslo o 1 vyšší než počet řádků ve všech předchozích skupinách
* NTILE(int) rozdělí výsledek do zadaného počtu skupin se stejným počtem řádků (nebo lišícím se o 1)

**APPLY:**

{ CROSS | OUTER } APPLY

Obdoba INNER nebo LEFT OUTER JOIN, na pravé straně stojí tabulková funkce nebo poddotaz. Funkce smí vracet více řádků i sloupců, vstupními parametry jsou sloupce tabulky na levé straně. Funkce nebo poddotaz z pravé strany se zpracuje pro každý řádek tabulky na levé straně.

**FOR XML PATH:**

Fráze FOR XML slouží primárně ke generování výstupu ve formátu XML, s klíčovým slovem PATH se však dá využít k pohodlnému zřetězení hodnot z několika vstupních řádků do jednoho výstupního.

**PIVOT:**

Slouží k méně pracnému uložení hodnot z řádků do sloupců. Hranaté závorky jsou v tomto případě součástí syntaxe:

SELECT pevný\_sloupec,

 [první\_souhrnný\_sloupec] AS alias\_sloupce,

 [druhý\_souhrnný\_sloupec] AS alias\_sloupce,

 ...

 [poslední\_souhrnný\_sloupec] AS alias\_sloupce

FROM

 ( poddotaz ) AS lokální\_alias

PIVOT

 ( agregační\_výraz

FOR

 [sloupec\_jehož\_hodnoty\_budou\_hlavičkami\_sloupců]

 IN ( [první\_souhrnný\_sloupec], [druhý\_souhrnný\_sloupec],

 ... [poslední\_souhrnný\_sloupec] )

 ) AS lokální\_alias

### Zdroje informací

* nápověda
* anglické knihy Taming Visual FoxPro's SQL, Inside MS SQL Server 2008: T-SQL Querying
* kniha v češtině SQL Bez předchozích znalostí
* Nejen SELECTem živ je člověk (přednáška DevCon 2009)
* Visual FoxPro – úvod do jazyka SQL (seminář jaro 2007)
* Magický příkaz SELECT (přednáška DevCon 2005)
* webová fóra, např.
pandora.idnes.cz/conference/foxpro/
www.sqlservercentral.com

www.mssqltips.com

www.sql-server-performance.com

www.sqlteam.com

www.simple-talk.com

stackoverflow.com/

1. Všechny postavy jsou smyšlené a jakákoliv podobnost se skutečnými osobami je čistě náhodná. [↑](#footnote-ref-1)