

Elektronická učební pomůcka pro výuku Microsoft SQL Serveru

Lecture Notes for Microsoft SQL Server

Romana Nehodová



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2013/2014

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Romana Nehodová**
Osobní číslo: **A11683**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační technologie v administrativě**
Forma studia: **prezenční**

Téma práce: **Elektronická učební pomůcka pro výuku Microsoft SQL Serveru**
Téma anglicky: **Lecture Notes for the Microsoft SQL Server**

Zásady pro vypracování:

1. Seznamte se s MS SQL v aktuální edici.
2. Zpracujte popis instalace.
3. Zpracujte podklady pro cvičení a přednášky v rozsahu 14 týdnů.
4. Realizujte sadu vzorových příkladů.
5. Navrhněte sadu 60 testovacích úkolů včetně řešení.
6. Uvedte možné směry rozvoje učební pomůcky.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. LACKO, L'uboslav. Mistrovství v SQL Server 2012. Brno: Computer Press, 2013. ISBN 978-80-251-3773-4.
2. STANEK, William R. Microsoft SQL Server 2012: Kapesní rádce administrátora. Brno: Computer Press, 2013. ISBN 978-80-251-3797-0.
3. LACKO, Luboslav. SQL Hotová řešení. Brno: Computer Press, 2003. ISBN 80-7226-975-5.
4. BEN-GAN, Itzik, Dejan SARKA a Ron TALMAGE. Querying Microsoft SQL Server 2012: Exam 70-461 Training Kit. Sebastopol, California: O'Reilly Media, Inc., 2012. ISBN 978-0-7356-6605-4.
5. ATKINSON, Paul a Robert VIEIRA. Beginning Microsoft SQL Server 2012 programming. Indianapolis: John Wiley & Sons, Inc., 2012. ISBN 978-1-118-10228-2.
6. OPPEL, Andrew. Databáze bez předchozích znalostí. Brno: Computer Press, 2006. ISBN 80-251-1199-7.
7. SHELDON, Robert. SQL: Začínáme programovat. Praha: Grada Publishing, 2005. ISBN 80-247-0999-6.

Vedoucí bakalářské práce:

Ing. Petr Šilhavý, Ph.D.

Ústav počítačových a komunikačních systémů


Datum zadání bakalářské práce:

7. února 2014

Termín odevzdání bakalářské práce:

27. května 2014

Ve Zlíně dne 7. února 2014


prof. Ing. Vladimír Vašek, CSc.
děkan




prof. Ing. Karel Vlček, CSc.
ředitel ústavu

ABSTRAKT

Hlavním cílem této bakalářské práce je vytvoření učební pomůcky pro výuku Microsoft SQL Serveru 2012. Tato práce je rozdělena na dvě části: teoretickou a praktickou. Teoretická část je zaměřena na teorii databázových systémů a aktuální edici Microsoft SQL Serveru. Praktická část se zabývá hlavně konkrétními příkazy SQL. K oběma částem je zpracováno 14 prezentací, které představují 14 vyučovacích týdnů. Teoretické prezentace jsou souhrnem teoretické části této práce. Praktické prezentace jsou doplněny o příklady a cvičení. V závěru práce jsou uvedeny možnosti, o které lze tuto učební pomůcku rozšířit.

Klíčová slova:

SQL Server, databáze, tabulka, index, procedura, relace, pohled, spoušť, transakce

ABSTRACT

The main aim of this paper is to create instructional materials for training of Microsoft SQL Server 2012. The paper is divided into two parts: the practical one and the theoretical one. The theoretical part focuses on the theory of database systems and on the current release of Microsoft SQL Server. The practical part analyses particular SQL instructions. Except the theoretical part and the practical one fourteen presentations are included into the paper. These presentations cover fourteen training weeks and they are completed by the examples and exercises. There are also other training possibilities mentioned at the end of the paper.

Keywords:

SQL Server, database, table, index, procedure, relationship, view, triggers, transaction

Velice ráda bych poděkovala svému vedoucímu bakalářské práce Ing. Petru Šilhavému, Ph.D., který mi poskytoval dobré rady.

Poděkování také patří celé mé rodině a mému příteli, kteří mě v mém studiu podporovali.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	11
I TEORETICKÁ ČÁST	12
1 DATABÁZOVÉ SYSTÉMY.....	13
1.1 DATABÁZOVÝ SYSTÉM.....	13
1.1.1 Databáze.....	13
1.1.2 Systém řízení báze dat.....	13
1.1.3 Databázová aplikace	14
1.2 HISTORIE DATABÁZÍ.....	14
1.2.1 Jazyk SQL	15
1.2.2 Jazyk Transact-SQL (T-SQL).....	15
1.3 DATABÁZOVÉ MODELÝ	16
1.3.1 Hierarchický model.....	16
1.3.2 Síťový model.....	17
1.3.3 Relační model	17
1.3.4 Objektově orientovaný model	17
1.3.5 Objektově relační model	18
1.4 VRSTVY DATOVÉ ABSTRAKCE.....	18
1.4.1 Fyzická vrstva.....	18
1.4.2 Logická vrstva	18
1.4.3 Externí vrstva	19
1.4.4 Fyzická datová nezávislost.....	19
1.4.5 Logická datová nezávislost	19
1.5 DATOVÉ MODELÝ	20
1.5.1 Konceptuální model	20
1.5.2 Logický model.....	20
1.5.3 Fyzický model	20
1.6 ENTITNĚ-RELAČNÍ MODEL.....	20
1.6.1 Entita	20
1.6.2 Vztahy (Relace).....	20
1.6.3 Atribut	21
1.6.4 Entitně-relační model (E-R model).....	21
1.6.5 Entitně-relační diagram (E-R diagram).....	21
1.7 OBJEKTY V DATABÁZI.....	22
1.7.1 Databázové tabulky.....	22
1.7.2 Pohled.....	22
1.7.3 Indexy.....	23
1.7.4 Uložené procedury	24
1.7.5 Spouště (triggers)	24
1.7.6 Integritní omezení	24
1.7.7 Primární klíč.....	24
1.7.8 Cizí klíč	25

1.8	KARDINALITA VZTAHŮ	25
1.9	NORMÁLNÍ FORMY	25
1.9.1	První normální forma (1NF)	26
1.9.2	Druhá normální forma (2NF)	26
1.9.3	Třetí normální forma (3NF)	27
1.9.4	Čtvrtá normální forma (4NF)	27
1.9.5	Pátá normální forma (5NF)	27
1.10	ŽIVOTNÍ CYKLUS DATABÁZE	27
1.10.1	Klasická metoda	28
1.10.1.1	Plánování	28
1.10.1.2	Sběr požadavků	29
1.10.1.3	Konceptuální návrh	29
1.10.1.4	Logický návrh	29
1.10.1.5	Fyzický návrh	30
1.10.1.6	Konstrukční fáze	30
1.10.1.7	Implementace a nasazení	31
1.10.1.8	Průběžná podpora	31
1.11	RELAČNÍ ALGEBRA	31
1.11.1	Selekce	32
1.11.2	Projekce	32
1.12	ZABEZPEČENÍ DATABÁZE	33
1.12.1	Přihlašovací účet	34
1.12.2	Databáze	35
1.12.3	Systémové databáze	35
1.12.4	Oprávnění	35
1.12.5	Systémová oprávnění	36
1.12.6	Objektová oprávnění	36
1.13	TRANSAKCE	36
1.13.1	Podpora transakcí	37
1.14	INDEXY	38
1.14.1	Tradiční indexy	38
1.14.2	Výběr vhodných sloupců pro indexaci	40
1.14.3	Indexování počítaných sloupců a pohledů	40
2	MICROSOFT SQL SERVER	41
2.1	HISTORIE SQL SERVERU	41
2.2	MICROSOFT SQL SERVER 2012	41
2.2.1	Standard edition	41
2.2.2	Enterprise edition	42
2.2.3	Business Intelligence edition	42
2.2.4	Specializované edice	42
2.2.4.1	Express Edition	42
2.2.4.2	Developer edition	42
2.2.4.3	Web edition	43
II	PRAKTICKÁ ČÁST	44

3	PRÁCE S MICROSOFT SQL SERVEREM 2012.....	45
3.1	INSTALACE MICROSOFT SQL SERVERU 2012.....	45
3.2	SQL MANAGEMENT STUDIO	45
3.3	VYTVOŘENÍ DATABÁZE.....	47
3.3.1	Uživatelské databáze	49
3.4	VYTVOŘENÍ TABULKY.....	49
3.4.1	Příkazy pro změnu tabulky.....	50
3.5	DATOVÉ TYPY	50
3.5.1	Číselné datové typy	50
3.5.2	Celočíselné datové typy	50
3.5.3	Znakové datové typy	51
3.5.4	Datové typy pro uložení hodnot data a času	51
3.6	PRÁCE S DATY	51
3.7	VÝBĚR ZÁZNAMŮ.....	52
3.8	VAZBY MEZI TABULKAMI	53
3.9	SPOJENÍ TABULEK	54
3.9.1	INNER JOIN	55
3.9.2	OUTER JOIN.....	55
3.9.3	CROSS JOIN.....	55
3.10	FUNKCE.....	55
3.10.1	Agregační funkce	56
3.10.2	Matematické funkce.....	56
3.10.3	Funkce pro práci s textovými řetězci	57
3.10.4	Funkce pro práci s datem a časem:	58
3.10.5	Konverzní funkce	58
3.11	VNOŘENÉ DOTAZY	59
3.12	SKLÁDÁNÍ DOTAZŮ	59
3.12.1	UNION	59
3.12.2	INTERSECT	59
3.13	PROMĚNNÉ.....	60
3.14	PODMÍNKY	60
3.15	PROCEDURY	60
3.16	TRIGGERS	61
3.17	TESTOVACÍ ÚKOLY.....	61
3.18	RŮZNÉ SMĚRY ROZVOJE UČEBNÍ POMŮCKY	62
	ZÁVĚR.....	63
	ZÁVĚR V ANGLIČTINĚ	64
	SEZNAM POUŽITÉ LITERATURY	65
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	67

SEZNAM OBRÁZKŮ	68
SEZNAM TABULEK	69
SEZNAM PŘÍLOH.....	70

ÚVOD

Tato bakalářská práce se zabývá úvodem do databázových systémů. Je určena především pro ty, kdo s databázemi teprve začínají. Také bude elektronickým doplňkem pro předmět databázové systémy, ale přečíst si ji může kdokoli, kdo se o toto téma zajímá.

Elektronické databáze slouží hlavně k uchovávání velkého množství informací na médiích. Tyto informace jsou uloženy ve strukturované formě, takže je k nim snadný přístup pomocí speciálních příkazů SQL. Tyto příkazy umožňují snadné prohlížení a vyhledávání podle různých kritérií bez složitého přepisování. Výsledky vyhledávání v databázi můžeme různě seřazovat, upravovat, třídit a seskupovat. Příkazy slouží také pro vkládání, úpravu a mazání dat a vytváření, změnu a mazání objektů v databázi. Databázové systémy obsahují velké množství nástrojů potřebných k zálohování, obnovování a zabezpečení databáze pro její ochranu v případě havárie.

Tato práce se také zabývá výukou Microsoft SQL Serveru 2012. Všechny příkazy, nastavení a další různé teorie zde uvedené jsou speciálně pro řadu Microsoft SQL Server. Na trhu existují i další produkty, např. Oracle, MySQL, Firebird, pro tvorbu a správu databází, ale základní téma této bakalářské práce se zabývá pouze produktem firmy Microsoft.

I. TEORETICKÁ ČÁST

1 DATABÁZOVÉ SYSTÉMY

1.1 Databázový systém

1.1.1 Databáze

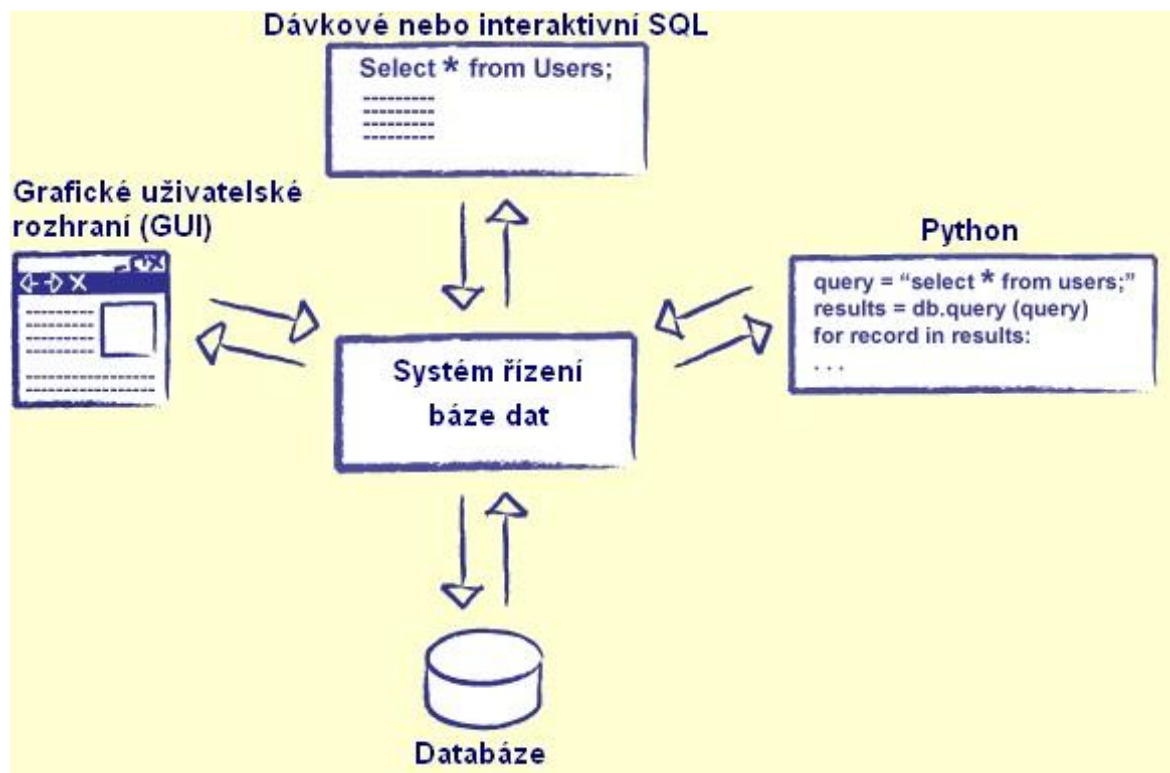
Databáze jsou data, která jsou organizována ve struktuře, která je definována jako metadata. Metadata slouží k popisu této struktury a informují o uložených datech [1]. Databáze tedy obsahuje vlastní údaje, relační vztahy, integritní omezení a schémata popisující strukturu údajů [2].

1.1.2 Systém řízení báze dat

Systém řízení báze dat (SŘBD) je software, který přistupuje k údajům v databázi. V anglické literatuře se uvádí název Database Management System (DBMS). Uživatel nebo aplikace komunikuje s databází prostřednictvím SŘBD a jazyka SQL. Nemusí tedy znát fyzickou strukturu uložených údajů. [2]

Typické charakterové rysy SŘBD:

- Složitější manipulace s údaji probíhá pomocí transakčního zpracování. To buď provede všechny operace v transakci úspěšně, nebo se vrátí do původního stavu.
- SŘBD má funkci, která slouží k zotavení se z chyb a nehod, např. z havárie nebo živelné pohromy. Pokud k takovéto události dojde, systém by měl pokračovat v práci např. ze záložní databáze nebo se vrátit do původního stavu před nehodou.
- K SŘBD může přistupovat více uživatelů najednou, proto musí umět definovat přístupová práva jednotlivým uživatelům a jejich oprávnění.
- Ochrana údajů slouží před ztrátou údajů, krádeží, zneužití nebo jako zabránění přístupu neoprávněných osob.
- Údaje mohou být uloženy na jednom serveru nebo decentralizovaně na více serverech. [2]
- Přesun dat z a do fyzických datových souborů.
- Podpora dotazovacího jazyka, který slouží pro výběr dat z databáze. [3]



Obr. 1. Interakce se SŘBD [4]

1.1.3 Databázová aplikace

Databázová aplikace přistupuje prostřednictvím SŘBD k databázi. Aplikace mohou využívat jak lokální databázi, tak i přistupovat k databázovému serveru. Dále mohou být přístupné i přes web, např. internet banking, internetové obchody apod. [2]

1.2 Historie databází

Herman Hollerith v roce 1890 sestrojil první děroštitkový stroj. Tento stroj byl vyvinutý podle tkalcovského stavu, který tkal vzory podle dírek ve štítu. V té době byla potřeba a poptávka po uchování a zpracování informací ze strany státních úřadů USA, které chtěly přehled o svých zaměstnancích. [5]

Roku 1911 došlo ke spojení společností Charlese R. Flinta a Hermana Holleritha s dalšími dvěma firmami, a tím vznikla společnost Computing-Tabulating-Recording Corporation (CTR). S rozšířením společnosti se roku 1924 přejmenovala na International Business Machines Corporation (IBM). [5]

Poznatky z projektu Apollo daly základ prvnímu databázovému modelu pod názvem Information Management System (IMS). Tento model byl založen na hierarchické struktuře. Do té doby byla většina databází založena na síťové struktuře, díky seskupení Data Systems Languages (Codasyl) založené 1960 vládou USA. [5]

Zaměstnanec IBM, Dr. Codd, předložil návrh na implementaci nového datového modelu. Tím vznikl relační datový model, který se odlišoval od ostatních použitím srozumitelných příkazů, které byly přijatelné pro netechnicky vzdělané uživatele, protože vycházely z běžné angličtiny. [5]

IBM nevyužila svoji šanci na trhu a v roce 1980 představila společnost Oracle Corporation první SQL databázi, která byla pro počítače VAX. IBM svůj produkt DB2 pro počítače MVS představila později. Další projekty na sebe nenechaly dlouho čekat. [5]

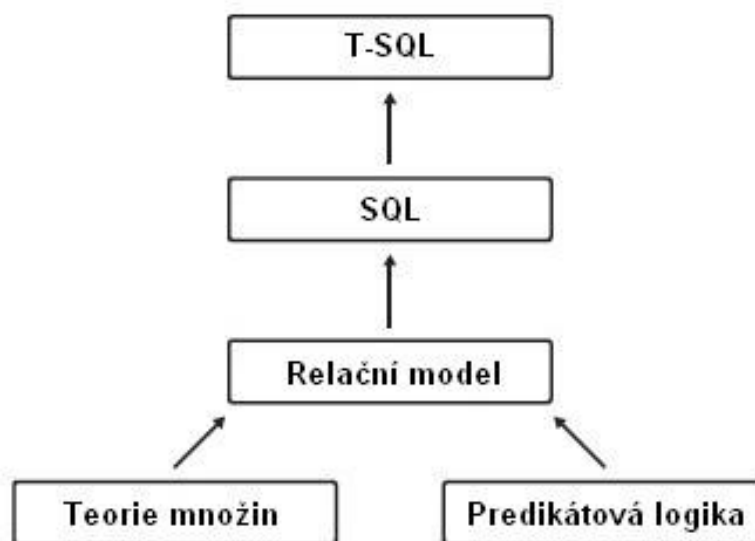
1.2.1 Jazyk SQL

První verzi databázového jazyka vytvořila firma IBM s názvem Structured English Query Language (SEQUEL). Už tehdy byla snaha o jazyk, který by vycházel z přirozeného jazyka – angličtiny. Postupně se k tomuto standardu přidávaly další firmy (Oracle, SyBase, Informix), a tak vznikl „nepsaný standard“ jazyka s názvem SQL. [2]

SQL je standard International Organization for Standards (ISO) a American National Standards Institute (ANSI). Norma SQL se stále vyvíjí. Hlavní revizní normy: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008 a SQL:2011. Všichni přední dodavatelé databází implementovali jazyk SQL jako hlavní jazyk pro správu a manipulaci s daty v databázových platformách, proto základní jazykové prvky vypadají stejně. [6]

1.2.2 Jazyk Transact-SQL (T-SQL)

T-SQL je hlavní jazyk používaný pro správu a manipulaci s daty v hlavním relačním databázovém systému Microsoft, SQL Serveru, ať už lokálně nebo v cloudu (Microsoft Windows Azure SQL databáze). Vychází z jazyka SQL a je založen na silném matematickém základu. SQL Server podporuje i další jazyky, jako je Microsoft Visual C # a Microsoft Visual Basic, ale T-SQL je zpravidla preferovaný jazyk pro správu dat a manipulaci s nimi. [6]



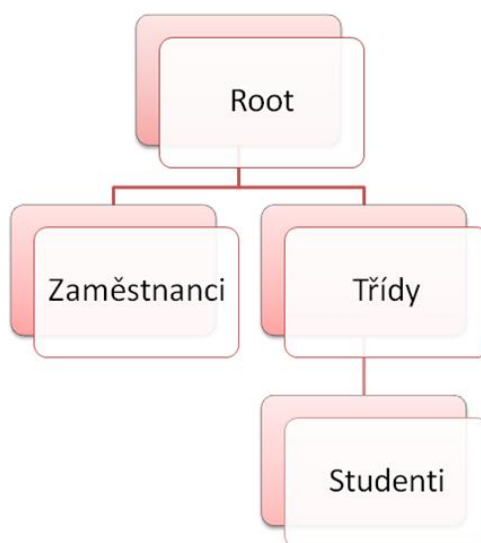
Obr. 2. Evoluce T-SQL [6]

1.3 Databázové modely

Databázový model je architektura, která určuje, jak se budou ukládat objekty do databáze a jak budou vzájemně provázané. [3]

1.3.1 Hierarchický model

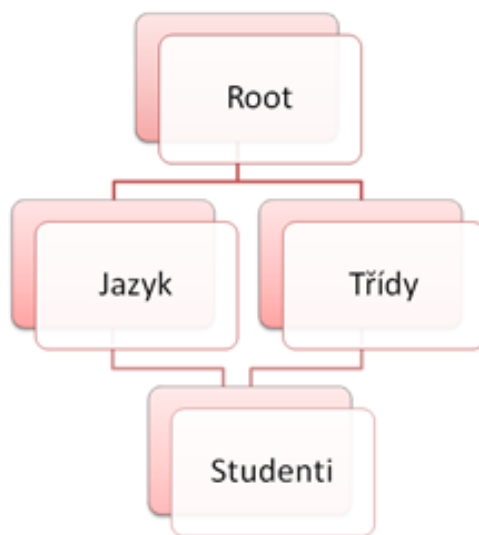
Specifikací tohoto modelu je, že podřízený záznam může mít pouze jeden nadřazený záznam, ale nadřazený záznam může mít mnoho podřízených záznamů. Nejrozšířenější hierarchickou databází býval systém ISM od IBM. [3]



Obr. 3. Hierarchický model [7]

1.3.2 Síťový model

Síťový model vychází z hierarchického modelu, který doplňuje o mnohonásobné vztahy (sety). Pomocí setů mohou být záznamy připojeny k jednomu nebo více záznamům. Síťový model podporuje operace jako posunování potomka v setu a nalezení záznamu podle klíče. Negativní aspekty tohoto modelu jsou nepružnost a obtížná změna struktury. [8]



Obr. 4. Síťový model [7]

1.3.3 Relační model

Relační model databází pochází z výzkumné práce Dr. E. F. Codd. Tento model svazuje záznamy jen podle potřeby, nikoli podle vazeb definovaných předem, jak tomu bylo u předchozích modelů. Dotazy v relačním modelu pracují vždy s určitou množinou dat, nikoli s jednotlivými záznamy jako u předchozích modelů. Data jsou reprezentována pomocí dvourozměrných tabulek. [3]

1.3.4 Objektově orientovaný model

Vznik tohoto modelu se datuje už v sedmdesátých letech, ale využívat se opravdu začal až v letech devadesátých. Důvodem vzniku objektově orientovaného modelu je neschopnost relačního databázového modelu v té době pracovat se složitějšími datovými typy, jako jsou obrázky, složité výkresy a zvukové či video soubory. Velký rozmach sítě Internet a služby World Wide Web vyvolal poptávku po vhodné technologii, která by doručovala složitější data. [3]

Objekt je logické seskupení příbuzných dat a programové logiky, které společně reprezentuje nějakou věc či osobu z reálného světa. Jednotlivé datové položky, jako je identifikátor a jméno zákazníka, se v objektově orientovaném modelu nazývají proměnné a jsou uloženy u každého objektu. Metoda je část aplikační programové logiky, která pracuje nad určitým objektem a provádí nad ním jistou konečnou funkci. Největší odlišností tohoto modelu od ostatních je, že k proměnným se přistupuje pouze prostřednictvím metod. Této vlastnosti se říká zapouzdření objektů. [3]

1.3.5 Objektově relační model

Výrobci relačních databází zaznamenali výhody objektově orientovaného modelu a doplnili objektové funkce do běžných databázových produktů. Původně se těmto produktům říkalo univerzální databáze. V odborných kruzích se tento model nikdy nechytil a začal být označován jako objektově relační. Za objektově relační systémy je možné díky postupnému vývoji do jisté míry označit databáze Oracle, DB2 a Informix. [3]

1.4 Vrstvy datové abstrakce

Architektura, na které je postavena většina moderních databázových systémů, se skládá ze tří základních vrstev: z fyzické, logické a externí vrstvy. Součástí původní architektury byla ještě konceptuální vrstva, ale tu už neimplementuje žádný z výrobců databází. [3]

1.4.1 Fyzická vrstva

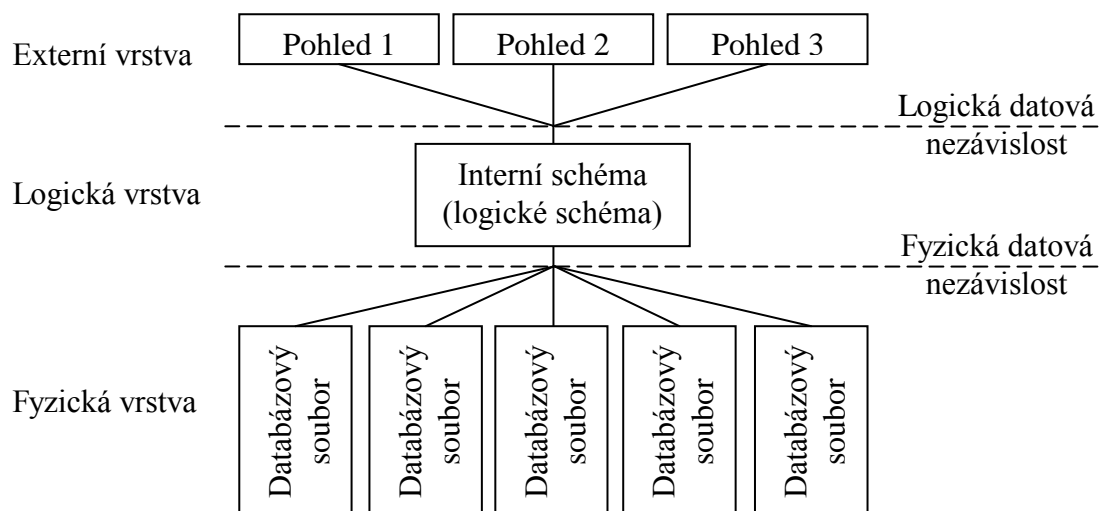
Fyzická vrstva obsahuje datové soubory, do nichž se ukládají data příslušné databáze. Jedna databáze může být uložena na různých diskových jednotkách v několika datových souborech. Serverově orientované databáze, jako Microsoft SQL Server, zajišťují správu fyzických souborů a uživatel se na ně při práci nemusí výslovně odvolávat. [3]

1.4.2 Logická vrstva

Logická vrstva představuje abstraktní vrstvu v databázi. Fyzická vrstva skutečně existuje, zatímco logická vrstva je pouze součástí abstraktních datových struktur, které se podle potřeby skládají z objektů fyzické vrstvy. Podle konkrétního databázového systému může být logická vrstva tvořena množinou dvourozměrných tabulek, hierarchickou strukturou podobnou například organizačnímu diagramu firmy nebo jinou strukturou. [3]

1.4.3 Externí vrstva

Externí vrstvu tvoří uživatelské pohledy. V této vrstvě se k databázi připojují uživatelé a aplikační programy, které s ní dále pracují a zadávají v ní dotazy. Do přímého styku s fyzickou a logickou vrstvou vstupuje v ideálním případě jen databázový administrátor. Databázový systém pak zajišťuje transformaci vybraných položek z jedné nebo více datových struktur v logické vrstvě do konkrétního uživatelského pohledu. [3]



Obr. 5. Vrstvy datové abstrakce [3]

1.4.4 Fyzická datová nezávislost

Fyzická datová nezávislost je možnost změny fyzické souborové struktury v databázi bez narušení činnosti stávajících uživatelů a procesů. Tato nezávislost je možná díky oddělení fyzické vrstvy od logické. Stupeň fyzické datové nezávislosti vyjadřuje rozsah možných změn ve fyzickém souborovém systému bez zásahu do logické vrstvy. [3]

1.4.5 Logická datová nezávislost

Logickou datovou nezávislostí označujeme možnost provádět změny v logické vrstvě bez narušení činnosti stávajících uživatelů a procesů. Za logickou datovou nezávislostí stojí transformace mezi logickou vrstvou a externí vrstvou. Je důležité si uvědomit, že většina logických změn s sebou přináší také nějakou fyzickou změnu. [3]

1.5 Datové modely

Při modelování databází přecházíme od konceptuálního modelu, který je úplně hardwarově a softwarově nezávislý, k logickému a fyzickému modelu. [2]

1.5.1 Konceptuální model

Konceptuální model popisuje údaje v databázi nezávisle na jejich fyzickém uložení. Jeho návrh se zaměřuje na aplikační logiku z pohledu člověka. Tvorba modelů se zaměřuje na objekty reálného světa, vztahy mezi nimi a funkce realizující tyto vztahy. [2]

1.5.2 Logický model

Logický model převede konceptuální model do takové podoby, která již zapadá do zvoleného databázového modelu (relačního, objektově orientovaného, atd.). [3]

1.5.3 Fyzický model

Fyzický model promítne logický model do jednoho nebo více modelů, které jsou určeny konkrétnímu databázovému systému, který bude zajišťovat správu databáze, a konkrétnímu počítačovému systému, na kterém databáze poběží. [3]

1.6 Entitně-relační model

1.6.1 Entita

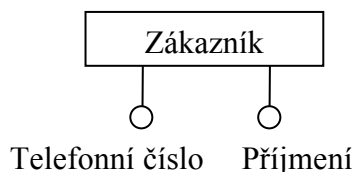
Entita je předmět reálného světa, který je pojmenovatelný, zajímavý a jednoznačně odlišný. Je to osoba, místo, věc, událost nebo myšlenka, o které se shromažďují nebo jsou shromažděna data. Entitou může být zákazník, výrobek, zaměstnanec apod. [9]

1.6.2 Vztahy (Relace)

Vztahy popisují „asociace“ mezi dvě ale i více entitami. Osoba může mít např. pozici. Pro popis se užívají většinou slovesa. [9]

1.6.3 Atribut

Atribut slouží k popisu nebo upřesnění vlastností entity nebo vztahu. Dává entitě nebo vztahu hodnotu. Může to být rodné číslo, telefonní číslo, datum narození apod. [9]



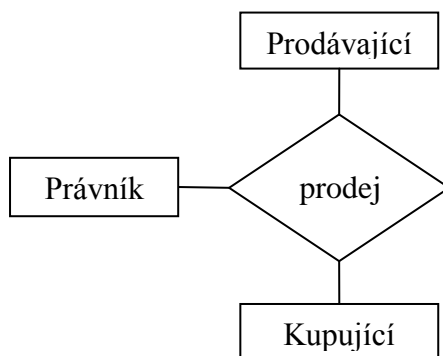
Obr. 6. Entita s atributy [2]

1.6.4 Entitně-relační model (E-R model)

E-R model je nástroj, pomocí kterého je popsána aplikace za účelem následné specifikace struktury databáze. Proces návrhu systému spočívá v identifikaci typů entit jako množin objektů stejného typu, v identifikaci typů vztahů, do kterých budou entity vstupovat a v přiřazení atributů, které blíže popisují vlastnosti jednotlivých entit a vztahů. [9]

1.6.5 Entitně-relační diagram (E-R diagram)

Pro návrh a zápis E-R vztahů můžeme použít lineární textový zápis nebo E-R diagramy. E-R diagram je ale přehlednější a používanější. Entity jsou reprezentovány obdélníky a relace kosočtverci. Spojnice prvků ukazují zapojení jednotlivých entit do jednotlivých typů vztahů. [9]



Obr. 7. Relace mezi více entitami [2]

1.7 Objekty v databázi

Hlavním komponentám databáze SQL Serveru se říká objekty. Mezi tyto objekty mimo jiné patří tabulky, pohledy, indexy, uložené procedury, spouště, certifikáty, integritní omezení, klíče. [10]

1.7.1 Databázové tabulky

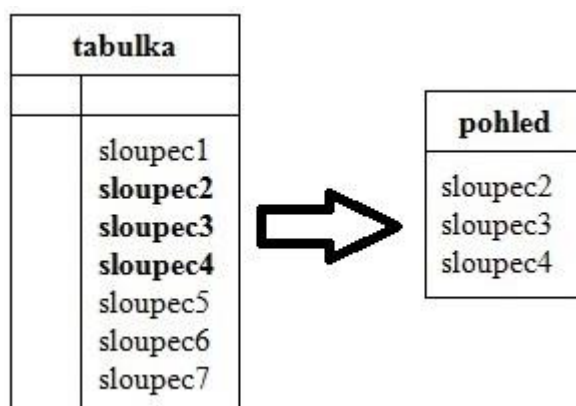
Databázové tabulky v prostředí relačních databází slouží k ukládání údajů. Mají jednoduchou dvourozměrnou strukturu. Skládají se z řádků, sloupců a hodnot:

- **Sloupec** – Sloupec, nebo také atribut, je množina údajů jednoho datového typu v tabulce.
- **Řádek** – Řádek, někdy označován jako záznam, je kombinace hodnot sloupců v tabulce. Každý řádek má jednoznačný identifikátor – primární klíč.
- **Hodnoty** – Hodnoty, také údaje, se nacházejí v průsečíku řádků a sloupců. [10]

id_zak	firma	adresa	mesto	obrat	dluh
1	Procesory s. r. o.	Volkrova 12	Praha	2567892.30	365000.00
2	Matice a. s.	Zaoralova 16	Brno	753275.65	100000.00
3	Kosmetika	Dolni 654	Olomouc	212356.20	0.00
4	Vodovahy a. s.	Zelena 4	Rajhrad	6256120.45	40000.00
5	Omigaj a synove	Nadrazni 333	Plzen	57210.00	920000.00

Tab. 1. Databázová tabulka [2]

1.7.2 Pohled

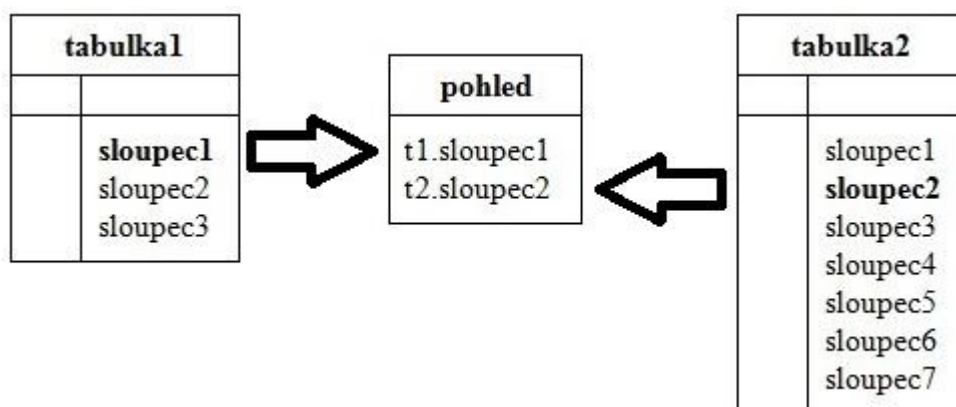


Obr. 8. Vytvoření pohledu z jiné tabulky [2]

Pohled (View) se může definovat jako předpis pro vytvoření podmnožiny dat z jedné nebo více databázových tabulek. Pohled neobsahuje samotné údaje z tabulek, ale jen předpis. Správným nastavením oprávnění je možné zamezit zneužití citlivých údajů, např. zpřístupněním jen části tabulky. Uživatelé mohou používat pohledy obdobným způsobem jako tabulky. [10]

Pohled může obsahovat údaje i z více tabulek. Uživatelé k nim mají přístup i bez komplikovaných SQL příkazů pro spojování tabulek. Pohledy se dělí podle toho z kolika tabulek a jakým způsobem vznikly na jednoduché a komplexní:

- **Jednoduché pohledy** – Údaje jsou pouze z jedné tabulky a neobsahují funkce ani skupiny dat.
- **Komplexní pohledy** – Údaje jsou z více tabulek, mohou obsahovat funkce a skupiny dat. Nevýhodou tohoto pohledu je, že zpravidla neumožňuje vykonávat změny údajů pomocí příkazů jazyka SQL. [2]



Obr. 9. Vytvoření pohledu z více tabulek [2]

1.7.3 Indexy

Indexy obsahují údaje o pozicích záznamů v tabulce, tím řeší problém s vyhledáváním konkrétního záznamu. Při vyhledávání podle indexovaného sloupce stačí najít údaj v indexu, získat odtud unikátní identifikátor záznamu, který mu přiřadil databázový server, a podle něj najde požadovaný záznam. Nevýhodou indexů je, že vyhledávání je rychlé jen při hledání malého počtu údajů. Při více než 5 % ze všech záznamů je rychlejší vyhledávání i v neindexované tabulce. Indexy samozřejmě zabírají místo na disku, proto není vhodné je používat v malých tabulkách nebo nad sloupci s nízkou variabilitou. [2]

1.7.4 Uložené procedury

Do databáze je možné kromě dat uložit také program, který se nad daty má vykonat. Tyto programy se nazývají procedury. Jsou zabezpečené a zálohované stejně jako zbytek databáze. Procedury se ukládají do databáze v předkompilované podobě. [2]

1.7.5 Spouště (triggers)

Spouště jsou v podstatě uložené procedury, které se spouští automaticky na základě nějaké předem dané události, která nastává při manipulaci s daty. Nespouštějí se přímo, ale jsou navázány na příkazy pro úpravu údajů (INSERT, DELETE, UPDATE). Nevrací tedy žádné výsledky. Používají se např. ke kontrole zadávaných údajů, zajištění datové integrity nebo při vkládání údajů přes view. [11]

1.7.6 Integritní omezení

Integritní omezení jsou pravidla, která zajišťují správnost a konzistenci uložených dat. Toto omezení je možné zavést na třech úrovních:

1. Entitní integrita – Každý řádek relace má unikátní identifikátor – zajištění jednoznačného primárního klíče.
2. Doménová integrita – Zajištění souladu každé hodnoty atributu s množinou přípustných hodnot.
3. Referenční integrita – Cizí klíče, tj. atributy nebo skupina atributů tvořící primární klíč v jiné relaci nemohou nabývat hodnot, které jsou v rozporu s hodnotami odkazovaného klíče. [2]

1.7.7 Primární klíč

Primární klíč je jednoznačný identifikátor každého záznamu v tabulce. Měl by se skládat z minimálního počtu atributů. Může to být sloupec nebo kombinace více sloupců. Hodnota pole primárního klíče musí být v rámci tabulky jedinečná a nesmí obsahovat hodnotu NULL. Bez primárního klíče není možné definovat relace mezi tabulkami. Příklad primárního klíče může být číslo účtu v bance. Klíč by měl být z hlediska nároku na paměť a čas také efektivní. Efektivitou se zde myslí, že by klíč měl být co nejmenší, měl by obsahovat co nejméně sloupců a byl v relacích využitý. [9]

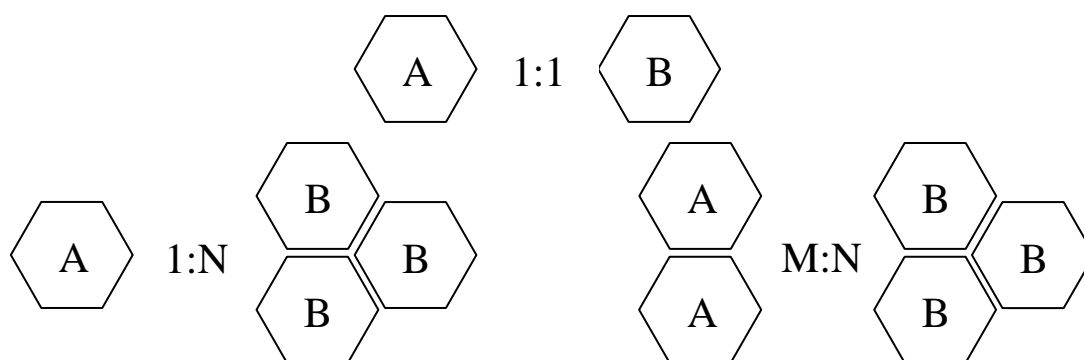
1.7.8 Cizí klíč

Cizí klíč je sloupec, nebo kombinace více sloupců, které jsou propojeny s primárním klíčem v jiné tabulce. Tímto propojením se minimalizuje objem údajů, které jsou v databázi uloženy. [9]

1.8 Kardinalita vztahů

Relace mezi tabulkami popisují vztahy mezi objekty reálného světa. Mezi databázovými tabulkami může existovat několik druhů vztahů. Vzhledem na kardinalitu budou entity nazývány jedna a druhá. Mezi nimi bude vztah (relace):

- 1:1 – každé entitě z první tabulky odpovídá maximálně jedna entita z druhé databázové tabulky a naopak.
- 1:N – první entitě odpovídá více druhých entit než jedna, ale naopak druhé entitě odpovídá maximálně jedna první entita.
- M:N – každé entitě odpovídá více než jedna entita z druhé databázové tabulky a naopak. [9]



Obr. 10. Kardinalita vztahů mezi entitami [2]

1.9 Normální formy

Koncept normalizace, což je proces organizace databáze do struktury, která zachovává integritu uložených dat a minimalizuje redundanci, je základním principem relačního modelu. Normalizovaná databáze splňuje pravidla relačního modelu. Normální formy jsou pravidla, která poskytují vodítka k organizaci dat za účelem zamezení ztráty dat při používání databáze. [1]

1.9.1 První normální forma (1NF)

Tabulka splňuje tuto formu tehdy, když všechny atributy (sloupce) jsou dále nedělitelné. Jeden sloupec nesmí obsahovat více druhů údajů nebo relace. Při nesplnění podmínky 1NF je tabulka v nulté normální formě, anglicky non-first normal form (NFNF), a je potřeba atribut rozložit na více atributů nebo na více relačně svázaných tabulek. [2]

Jmeno	Ulice
Josef X. Drda	Nábřežní 5, 100 00 Praha 1
Marta Nováková	Nálepková 12, Brno, 635 11
Josef Koudelka	Souběžná 9, 370 00 Plzeň

Tab. 2. Tabulka v NFNF [2]

Prijmeni	Jmeno	Ulice	PSC	Mesto
Drda	Josef X.	Nábřežní 5	100 00	Praha
Nováková	Marta	Nálepková 12	635 11	Brno
Koudelka	Josef	Souběžná 9	370 00	Plzeň

Tab. 3. Tabulka 1NF [2]

1.9.2 Druhá normální forma (2NF)

Tabulka splňuje druhou normální formu tehdy, když splňuje první normální formu a každý atribut, kromě primárního klíče, musí být úplně závislý na celém primárním klíči. Týká se to jen tabulek, které mají více primárních klíčů, při jednom primárním klíči je podmínka splněna automaticky. [2]

zboží	dodavatel	mail_dodavatele	cena
mýdlo	První drogistická a.s.	první@drogisticka.cz	12
mýdlo	Luxusní doplňky a.s.	doplňky@luxus.cz	74
hřeben	Plastolis	plasty@plastolis.cz	29
hřeben	Luxusní doplňky a.s.	doplňky@luxus.cz	278

Tab. 4. Tabulka nesplňující 2NF [2]

zboží	kod_dodavatele	cena
mýdlo	004	12
mýdlo	007	74
hřeben	002	29
hřeben	007	278

Tab. 5. Tabulka v 2NF [2]

kod_dodavatele	dodavatel	mail_dodavatele
004	První drogistická a.s.	první@drogisticka.cz
007	Luxusní doplňky a.s.	doplňky@luxus.cz
002	Plastolis	plasty@plastolis.cz

Tab. 6. Tabulka v 2NF [2]

1.9.3 Třetí normální forma (3NF)

Podmínky pro třetí normální formu jsou splnění 2NF a zároveň neexistující závislosti neklíčových sloupců tabulky. Například to může být závislost mezi neklíčovými sloupci PSC a Mesto v *Tab. 3*. Sloupec PSC by se nemusel zadávat, protože se dá vyhledat podle města z číselníku směrovacích čísel. [2]

1.9.4 Čtvrtá normální forma (4NF)

Ke splnění čtvrté normální formy je zapotřebí, když je tabulka ve třetí normální formě a popisuje jen jeden fakt nebo souvislost. [2]

1.9.5 Pátá normální forma (5NF)

Pro splnění páté normální formy je nutné splnit čtvrtou normální formu a není možné do ní přidat nový sloupec, popř. skupinu sloupců bez toho, aby se rozpadla na několik dílčích tabulek. [2]

1.10 Životní cyklus databáze

Pod pojmem životní cyklus databáze se rozumí veškeré události, které nastanou od okamžiku, kdy vyvstane potřeba databáze, přes vývoj a nasazení do provozu až po její odstavení z činnosti. [3]

Většina firem dodržuje určitý formální proces, který zajistí hladký a nákladově efektivní vývoj, jehož výsledkem bude kompletní počítačový systém. Návrh a implementace databází neprobíhá samostatně, ale spolu s dalšími komponentami uceleného systému, jako je aplikační rozhraní, aplikační programy a sestavy. Práce se zpravidla rozděluje do projektů, které mají své termíny a odpovědné osoby. [3]



Obr. 11. Klasický životní cyklus vývoje systémů [3]

1.10.1 Klasická metoda

Klasická metoda vývoje se opírá o proces nazvaný životní cyklus vývoje systémů (system development life cycle). Na Obr. 11 jsou v levém sloupci tradiční kroky životního cyklu, uprostřed základní aktivity projektu a napravo jsou pak potřebné kroky, které se musí v databázi udělat. Někdy nastane situace, že je potřeba se vrátit zpět a některé kroky dodělat, protože bez nich nejde jít dál. [3]

1.10.1.1 Plánování

Ve fázi plánování se musí organizace rozhodnout, kde se nachází a kam chce v projektu dojít, a rozhodnout se, jak takového stavu docílit. Plánování zahrnuje delší časové období a zde se rozhodne, jaké projekty obecně stanovených cílů zahájí. [3]

Po rozhodnutí o provedení projektu se provede tzv. studie proveditelnosti (feasibility study), která určí, jestli se reálně naplní stanovené cíle, a která provede první odhady zdrojů, financí a času. Po schválení studie proveditelnosti se začne skládat projektový tým, který se může v průběhu projektu měnit. [3]

1.10.1.2 Sběr požadavků

Během této fáze musí projektový tým shromáždit a zdokumentovat přesný popis cílů projektu. Zaměřuje se na otázku, čeho se má dosáhnout. Čím více se zvládne práce v této části projektu, tím bude hladší následující průběh. Sběr požadavků se provádí pomocí řady technik jako rozhovor, dotazník, pozorování nebo revize dokumentů. [9]

1.10.1.3 Konceptuální návrh

Cílem této fáze je návrh externího prostředí cílové aplikace (aplikací) a databáze (databází). Pro tuto etapu se také používá název externí návrh. Během této etapy se navrhne rozložení sestav, obrazovek, formulářů, webových stránek a dalších nástrojů pro pořizování a prezentaci dat. [3]

Databázový specialista provádí aktualizaci systémového datového modelu, který je ve formě ER-diagramu. Do diagramu se přidají nově objevené nebo změněné entity a poznamenají se nově doplněná či upravená aplikační pravidla. Uživatelské pohledy, entity a aplikační pravidla jsou důležitá pro úspěch další fáze, kterou je logický návrh databáze. [3]

1.10.1.4 Logický návrh

Ve fázi logického návrhu se provádí práce na technickém návrhu aplikace a databáze. Tato etapa je také známá pod pojmem interní návrh, protože se navrhují interní části projektu, se kterými se uživatelé při své práci nesetkají. Práce se rozdělí do segmentů (modulů) a každá jednotka se písemně specifikuje. Specifikace musí být natolik úplná, aby ji bez dalších informací zvládl naprogramovat jakýkoliv programátor s odpovídajícími znalostmi. Pomocí diagramů dat nebo starších vývojových diagramů se dokumentují logické toky mezi moduly. [3]

V této fázi probíhá v databázi normalizace. Po dokončení normalizace se provede aktualizace celkového logického datového modelu systému (pokud existuje) a doplní se do něj případně nově objevené entity. [9]

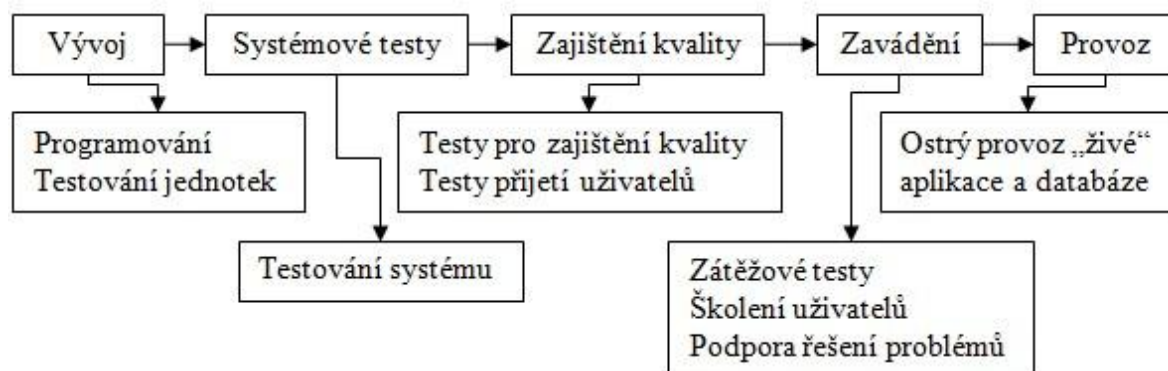
1.10.1.5 Fyzický návrh

Tato fáze zahrnuje převedení logického návrhu do podoby skutečného hardwaru a systémového softwaru, pomocí kterého se bude aplikace a databáze implementovat. Nejvíce práce zabere specifikace hardwaru, na který se bude cílová aplikace a databáze instalovat, protože součástí specifikací jsou odhady potřebných kapacit procesorů, diskových zařízení a šířky komunikačního pásma sítě, ve kterém poběží systém. [3]

V databázích se implementují normalizované relace do prostředí konkrétního databázového systému. Konkrétně se napíší a vygenerují příkazy jazyka DDL pro definici potřebných databázových objektů, včetně klauzulí SQL, které budou definovat fyzické umístění tabulek a indexů. Provádí se také předběžná analýza potřebných databázových dotazů pro odhad, jaké indexy budou pro rozumný výkon databáze potřeba. [3]

1.10.1.6 Konstrukční fáze

Vývojáři aplikací naprogramují jednotlivé programové jednotky a otestují je. Tyto jednotky se přenášejí do systémového testovacího prostředí, kde probíhá sestavování a testování celého aplikačního a databázového systému. Na Obr. 12 je znázorněno prostředí, které se při vývoji, testování a implementaci používá nejčastěji. Po dokončení systémových testů se systém předává do prostředí zajištění kvality. Testování musí být důkladné, protože pozdější opravování chyb je velmi nákladné. [3]



Obr. 12. Hardwarová a softwarová prostředí při vývoji [3]

Databázový administrátor má v době zahájení této fáze už většinu práce za sebou. Příkazy pro zavedení databázových komponent se dají zapsat do skriptu, který se dá použít ve všech následujících prostředích. Pokud se rozšiřuje stávající databáze nebo nahrazuje starší systém pro ukládání dat, jsou tyto přenosy složitější, protože se přenášejí data ze starých záznamových struktur do nových. [3]

1.10.1.7 Implementace a nasazení

Tato fáze zahrnuje instalaci nových komponent aplikačního systému (aplikačních programů, formulářů nebo webových stránek, sestav, databázových objektů atd.) do živého systému a provedení případné konverze dat. Nasazením systému se pak rozumí jeho oživení a uvedení do provozu. Někdy je možné celý nový projekt spustit najednou, což znamená, že všichni uživatelé přejdou na novou verzi současně. Při velkém množství uživatelů je ale lepší postupné zavádění systému, sníží se tím rizika spojená s přechodem systému. Běží-li nový i starý systém současně, uživatelé mnohou za tu dobu projít školením a postupně se zapojovat do nového systému. [3]

1.10.1.8 Průběžná podpora

Po dokončení všech předchozích kroků se podpora aplikace dostává do rukou provozní podpory. Tento tým má za úkol izolovat všelijaké problémy a reagovat na ně. Každá oprava chyby se stane dalším projektem, který musí znovu vstoupit do všech fází životního cyklu. Provedené změny se musí objevit v dokumentaci. Na *Obr. 12* je ideálním místem pro hledání chyb a jejich opravu zaváděcí prostředí. Chyby se mohou opravit souběžně s vývojem nejbližší nové verze rozšíření systému. [3]

1.11 Relační algebra

Relační algebra je nástroj pro práci s daty. Jedná se o jazyk vysoké úrovně, ve které se nepracuje s n -ticemi relací, ale s celými relacemi. Operátory relační algebry se použijí na relace a výsledkem jsou opět relace. [12]

Základní operace relační algebry jsou kartézský součin (\times), sjednocení (\cup), rozdíl ($-$), selekce a projekce. [12]

První tři operace jsou množinové a další dvě vybírají řádky resp. sloupce. Někdy se k těmto operacím řadí i přejmenování relace nebo jejího atributu, ale tyto operace nemění n-tice relace. [12]

1.11.1 Selekcce

Selekcce se zadává pomocí logické podmínky φ . $R(\varphi)$ znamená selekcce R podle φ . Množina n-tic z R^* , na kterých je splněna podmínka φ , jsou výsledkem relace R^* . Podmínka je obvykle booleovský výraz jednoduché podmínky ve tvaru $X > Y$ a $X \leq c$ apod. [12]

JMENO_F	REŽISÉR	ROK
Pátý element	Luc Besson	1997
Vrchní, prchni!	Ladislav Smoljak	1980
Pretty Woman	Garry Marshall	1990
Hollywood v koncích	Woody Allen	2002
Zlaté časy rádia	Woody Allen	1987
September	Woody Allen	1987
Povídky z New Yorku	Woody Allen	1989
Zločiny a poklesky	Woody Allen	1989
Samotáři	David Ondříček	2000

Tab. 7. FILMY1 [12]

Při aplikaci výrazu FILMY1(ROK > 1990) na Tab. 7 je výsledek v Tab. 8. [12]

JMENO_F	REŽISÉR	ROK
Pátý element	Luc Besson	1997
Hollywood v koncích	Woody Allen	2002
Samotáři	David Ondříček	2000

Tab. 8. Výsledek výrazu FILMY1(ROK > 1990) [12]

1.11.2 Projekce

Projekce relace $R(\mathbf{A})$ na množinu atributů $\mathbf{D} \subseteq \mathbf{A}$ je relace se schématem obsahující atributy \mathbf{D} , která se značí $R[\mathbf{D}]$. Když je R^* relace, tak projekce $R[\mathbf{D}]$ obsahuje všechny n-tice u takové, které splňují existenci n-tice v v R^* a $v[\mathbf{D}] = u$. [12]

Při aplikaci výrazu FILMY1[JMENO_F] na Tab. 7 je výsledek v Tab. 9. [12]

JMENO_F
Pátý element
Vrchní, prchni!
Pretty Woman
Hollywood v koncích
Zlaté časy rádia
September
Povídky z New Yorku
Zločiny a poklesky
Samotáři

Tab. 9. Výsledek výrazu *FILMY1*[JMENO_F] [12]

Skládání operací v relační algebře je možné díky tomu, že výsledkem aplikace každé operace je množina. Jedním výrazem je možné formulovat i složitější dotazy. [12]

Použití selekce s podmínkou $ROK < 1990$ a následně projekce na atribut REŽISÉR na Tab. 7. Výsledek je v Tab. 10. [12]

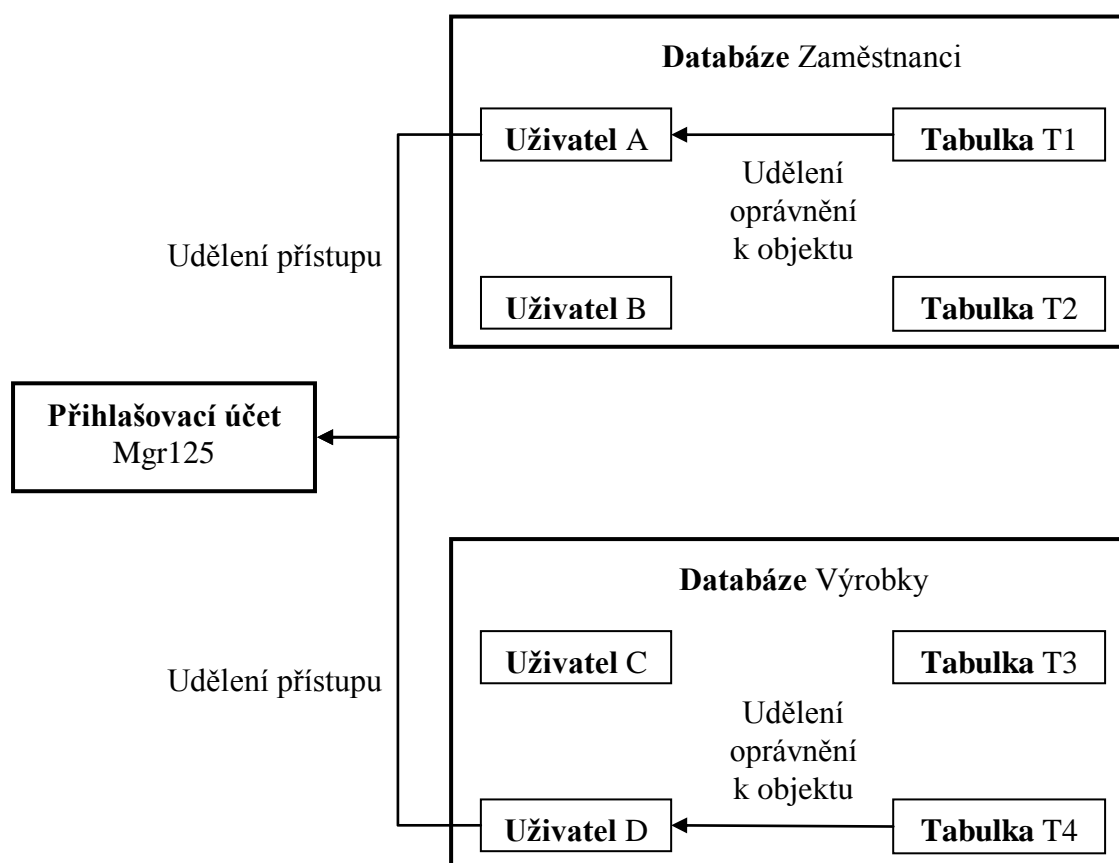
REŽISÉR
Ladislav Smoljak
Woody Allen

Tab. 10. Výsledek výrazu *FILMY1* ($ROK < 1990$) [REŽISÉR] [12]

Projekce odstraňuje duplicitní výsledky, jak je možné vidět na předchozím výsledku, protože Woody Allen natočil 4 filmy před rokem 1990. [12]

1.12 Zabezpečení databáze

Základním cílem zabezpečení přístupu databáze je přesné stanovení dat, která bude mít každý uživatel k dispozici ke své práci a jaké oprávnění bude mít (tedy výběr, vkládání, aktualizace nebo odstraňování). Každý uživatel by měl dostat jen ta oprávnění, která opravdu potřebuje – ani více ani méně. Totéž to platí i o aplikačních programech, které mají přístup k databázi. V otázce zabezpečení se tedy musí přistupovat ke všem uživatelům stejným způsobem. [3]



Obr. 13. Zjednodušený pohled na architekturu Microsoft SQL Server [3]

1.12.1 Přihlašovací účet

Přihlašovací účet je účet na serveru SQL, který není běžným účtem v operačním systému. Je možné ovšem použít autentizaci Microsoft Windows, při které si operační systém Windows uloží přihlašovací údaje a v okamžiku připojení k serveru SQL ověří jeho totožnost. Výhodou tohoto přihlášení je snadná správa uživatelského přístupu i k různým serverům SQL v celé organizaci – není nutné spravovat na každém SQL serveru SQL lokálně. To, že je v serveru SQL definován přihlašovací účet, ale neznamená, že má přístup k informacím v databázi. Hlavní přihlašovací účet se nazývá systémový administrátor. Tento účet má naprosté oprávnění ke všemu. Na Obr. 13 je znázorněn jen jeden přihlašovací účet, a sice Mgr125. [3]

1.12.2 Databáze

Každá databáze je tvořena seskupením jistých databázových objektů (tabulky, pohledy, indexy atd.). Kromě databází vytvořených uživateli se při vytvoření SQL serveru zakládá také několik specifických databází, které slouží pro správu databázového systému. Tyto databáze se nazývají systémové databáze. [10]

Každá databáze má přiřazenou jistou množinu uživatelů. Každý uživatel se dále promítá do přihlašovacího účtu. Lze tedy říci, že uživatel je v podstatě pseudoúčet, který tvoří alias k přihlašovacímu účtu SQL Serveru. Při přidělení oprávnění přihlašovacímu účtu se vytvoří v databázovém systému uživatelský účet, jenž odpovídá tomuto přihlašovacímu účtu. Oprávnění povolují přihlašovacímu účtu se připojit k databázi, ale nedávají mu oprávnění nad objekty v databázi. Další oprávnění se přidělují samostatně. [3]

1.12.3 Systémové databáze

- **master** – Tato databáze obsahuje informace o všech databázích na serveru. Tato databáze se mění při vytváření nové databáze, při správě účtů nebo změně konfigurace. Databázi master je nutné pravidelně zálohovat.
- **tempdb** – Databáze tempdb slouží jako dočasný prostor pro zpracovávání dotazy a další úkoly.
- **model** – Tato databáze představuje šablonu pro nově vytvořené databáze uživatelem systému. Mají-li mít nové databáze určité objekty, vlastnosti či oprávnění, je potřeba vložit tyto změny do této databáze.
- **msdb** – Slouží službě SQL Server Agent při práci s upozorněními, notifikacemi a naplánovanými úlohami. [10]

1.12.4 Oprávnění

Uživatelskému účtu je možné přidělit jakýkoliv počet oprávnění. Systémová oprávnění jsou obecná oprávnění platná všeobecně v celé databázi. Systémová oprávnění se dále dělí na serverové a příkazové oprávnění. Serverové oprávnění je např. oprávnění ke spuštění, zastavení a zálohování serveru SQL. Mezi příkazové oprávnění například patří oprávnění k vytvoření databáze nebo tabulky. Objektová oprávnění dovolují uživateli konkrétní operace nad konkrétním objektem, např. aktualizace dat nad tabulkou T1. [3]

1.12.5 Systémová oprávnění

Jak již bylo řečeno, systémová oprávnění jsou obecná oprávnění, potřebná k výkonu činností při správě serveru a jeho databází. Systémová oprávnění se přidělují příkazem GRANT a odebírají příkazem REVOKE. Každý výrobce databázových systémů podporuje mnoho různých oprávnění, nejvíce je však systémových oprávnění. Několik příkladů běžně používaných systémových oprávnění:

- SHUTDOWN – Umožní zadání příkazu k ukončení chodu serveru.
- CREATE DATABASE – Tímto příkazem lze vytvořit novou databázi.
- BACKUP DATABASE – Umožní spouštět zálohování na serveru SQL. [3]

1.12.6 Objektová oprávnění

Také objektová oprávnění se přidělují příkazem GRANT a odebírají příkazem REVOKE. Uživatel, který oprávnění uděluje, se nazývá oprávnitel (garant, přidělc), zatímco příjemce oprávnění se nazývá oprávněný (postupník). Příkaz GRANT může obsahovat klauzuli WITH GRANT OPTION, se kterou může příjemce toto oprávnění přidělovat jiným uživatelům. Při odebrání oprávnění příjemci se odebere i všem dalším uživatelům, kterým jej dal příjemce, proto se tato klauzule nedoporučuje. [3]

1.13 Transakce

Při správné podpoře činnosti uživatelů databázového systému je nutné, aby systém obsahoval mechanismy pro správu transakcí. Transakce, jak již bylo uvedeno, je posloupnost kroků, které musí být buď provedeny správně všechny, anebo se transakce neprovede vůbec. Transakce má tedy povahu „všechno nebo nic“. Tuto vlastnost je možné si pamatovat podle anglické zkratky ACID (Atomicity, Consistency, Isolation, Durability – tedy nedělitelnost, konzistentnost, izolace a trvanlivost):

- **Nedělitelnost** – Transakce se nesmí rozdělit. Při úspěšném provedení celé transakce si musí systém zapamatovat veškeré změny provedené transakcí, při selhání se vrátí do původního stavu.

- **Konzistentnost** – Po provedení transakce se celá databáze musí dostat z jednoho konzistentního stavu do druhého. Např. transformuje se vyřízená objednávka do stavu fakturované objednávky a současně se provedou další odpovídající změny.
- **Izolace** – Transakce se musí zpracovat nezávisle na jiných transakcích, které mohou běžet souvisle.
- **Trvanlivost** – Změny, které provede transakce, se musí uložit do systému tak, aby zůstaly i po ukončení chodu systému nebo havárii databáze. [3]

1.13.1 Podpora transakcí

Většina databázových systémů má podporu transakcí. Součástí této podpory jsou příkazy nebo mechanismy, které v jazyce SQL označují začátek a konec každé transakce a záznam veškerých změn v transakcích. [3]

Microsoft SQL Server podporuje transakce ve 3 různých režimech – automatické potvrzování, explicitní a implicitní:

- **Režim automatického potvrzování** – V tomto režimu se každý příkaz SQL považuje za diskretní transakci a po jejím dokončení se potvrdí. Tento režim se zpočátku používá při každém připojení k Microsoft SQL Serveru, pokud není zahájena explicitní transformace nebo pokud se nenastaví implicitní režim.
- **Explicitní režim** – Explicitní režim se spouští příkazem `BEGIN TRANSACTION` a ukončuje `COMMIT TRANSACTION` (úspěšné dokončení) nebo `ROLLBACK TRANSACTION` (neúspěšná transakce se vrací do původního stavu). Nejčastěji s tímto režimem pracují aplikační programy, uložené procedury, spouště a skripty.
- **Implicitní režim** – Tento režim se zapíná a vypíná příkazem `SET IMPLICIT_TRANSACTIONS {ON | OFF}`. Implicitní transformace se zahájí v okamžiku spuštění SQL příkazu z předepsané množiny, která obsahuje mimo jiné příkazy `DELETE`, `INSERT` `SELECT` a `UPDATE`. Implicitně zahájená transakce pokračuje až dokud ji buď potvrdíme, nebo vrátíme zpět. [3]

Veškeré změny jsou zaznamenány do transakčního protokolu (transaction log). Zapisuje se do něj dvojí obraz každé modifikace dat v databázi – před a po transakci. Tento protokol slouží hlavně ke snadnému návratu do předchozího stavu, protože podobu dat lze snadno obnovit z původního obrazu. Potvrzení transakce je považováno za dokončené až potvrzujícím zápisem do transakčního protokolu. Změny se nemusí na disk zapsat úplně hned, tak je transakční protokol někdy jedinou možností, ze které se dají po havárii systému obnovit data. [3]

1.14 Indexy

Indexy slouží hlavně k nálezům dat a přístupu k nim. V databázi je možné vytvořit indexy nad tabulkami, pohledy a počítanými sloupci. Index nad tabulkou slouží k rychlému prohledávání dat v tabulce. Index nad pohledem umožňuje generovat výsledkovou množinu pohledu, která se uloží do databáze a indexuje. Index nad počítaným sloupcem při sčítání určitých kritérií umožňuje vyčíslit jednotlivé výrazy a výsledky indexovat. [10]

Indexy se spravují mimo tabulky. V SQL Serveru na analýzu a implementaci slouží nástroj Database Engine Tuning Advisor. Tradiční indexy, též indexy nad řádky, sdružují a ukládají data pro jednotlivé řádky a pak spojují všechny řádky do jednoho indexu. Indexy nad sloupci fungují stejně jako indexy nad řádky, jen pracují se sloupci. [10]

1.14.1 Tradiční indexy

Tradiční indexy používají stránky. Struktura těchto stránek je podobná jako struktura datových stránek tabulky. Stránky indexu mají velikost 8 kB (8 192 bajtů) a 96bajtové záhlaví. Liší se od datových stránek tím, že nemají posuny řádků. [10]

Datové typy pro velké objekty nemohou být u běžných indexů klíčovými sloupci indexu, jsou to: image, ntext, text, varchar(max), nvarchar(max), varbinary(max) a xml. Výjimku tvoří varchar(max), nvarchar(max), varbinary(max), které mohou být součástí indexu. Do indexu lze zahrnout i deterministické počítané sloupce a počítané sloupce odvozené z datových typů pro velké objekty. [10]

SQL Server udržuje indexy prostřednictvím struktury B stromu, což je základní stromová struktura složená z kořenového uzlu několika navazujících úrovní uzlů a listů. Tím je možné prohledávat rychle a efektivně. [10]

SQL Server podporuje 2 hlavní typy indexování:

- clusterové indexy
- neclusterované indexy [10]

Clusterované i neclusterované indexy lze vytvářet v postatě na jakémkoliv sloupci. Před vytvořením indexu na počítaném sloupci je třeba zjistit, zda výraz v počítaném sloupci vrací pro konkrétní množinu vstupních dat vždy stejný výsledek. Indexy je nutné vybírat s rozvahou, protože správný výběr zvyšuje rychlost odezvy a špatný výběr může dokonce odezvu zpomalit. [10]

SQL Server podporuje několik speciálních typů indexování:

- **Indexy XML** – Tento index je optimální pro data typu XML. Primárním indexem je ten, který vznikne první na sloupci s XML daty. Pro jeho vytvoření se vytvoří clusterovaný index z clusterovaného klíče uživatelské tabulky a z identifikátoru uzlů XML. Sekunární indexy je možné vytvářet až po vytvoření primárního indexu XML. Tabulka může mít maximálně 249 indexů XML.
- **Filtrované indexy** – Filtrovaný index je optimalizovaný neclusterovaný index na podmnožině tabulkových dat. Tyto indexy snižují nároky na úložiště indexu, protože není nutné indexovat celé tabulky. Jsou menší než indexy nad celými tabulkami, a tak zvyšují výkon dotazů a kvalitu prováděcího plánu. K jejich údržbě dochází pouze v případě, že jsou změněna indexovaná data.
- **Prostorové indexy** – Tento index se využívá nad sloupcem v tabulce s obsahem geometrických nebo geografických dat. Prostorový index mapuje geografická data na dvojrozměrný prostor. Tabulka může obsahovat až 249 prostorových indexů, ale jen 15 sloupců. Tyto indexy lze použít pouze nad tabulkou s primárním klíčem. Metadata v primárním klíči nelze změnit, pokud existuje v tabulce prostorový index. Nad jedním sloupcem s geometrickými nebo geografickými daty je možné vytvořit více než jeden prostorový index, což je nutné pro indexaci různých parametrů. [10]

1.14.2 Výběr vhodných sloupců pro indexaci

Sloupce mohou být vybrány podle typu dotazů, které v databázi probíhají. Nástroj SQL Server Profiler slouží právě pro určení typů spouštěných dotazů, protože vytváří seznam uživatelské aktivity v databázi. Bez ohledu na metodu vybrání sloupců, je nutné vědět, že maximální délka všech klíčových sloupců je 900 bajtů. [10]

INDEXOVAT	NEINDEXOVAT
Tabulka s mnoha řádky	Tabulka s málo řádky
Sloupce, které se často používají v dotazech	Sloupce, které se v dotazech používají zřídka
Sloupce, které mají velký rozsah hodnot a u nichž je velmi pravděpodobné, že budou součástí řádků vybíraných v typických dotazech	Sloupce, které mají velký rozsah hodnot, ale u nichž je velmi málo pravděpodobné, že budou součástí řádků vybíraných v typických dotazech
Sloupce, které se používají v agregačních funkcích	Sloupce, které jsou hodně rozsáhlé (v bajtech)
Sloupce používané v dotazech s klauzulí GROUP BY	Tabulky, v nichž dochází k mnoha aktualizacím, ale málo dotazům
Sloupce používané v dotazech s klauzulí ORDER BY	
Sloupce používané pro spojování tabulek	

Zásady pro výběr tabulek a sloupců k indexování [10]

1.14.3 Indexování počítaných sloupců a pohledů

V databázi lze indexovat počítané sloupce a pohledy stejně jako tabulky. Tyto indexy znamenají uložení výsledků pro použití v budoucnu. U počítaných sloupců se nejdříve provede výpočet hodnot, a potom se z nich vytvoří klíče použité v indexu. V případě pohledů se výsledková množina uloží tak, že vznikne clusterovaný index nad pohledem. Indexy obou případů platí, jen dokud se nezmění data, ze kterých indexy vychází. Což klade určitá specifická omezení na jejich vytváření. [10]

2 MICROSOFT SQL SERVER

2.1 Historie SQL Serveru

V roce 1988 vlastnila SQL Server společnost Sybase a dodávala ho pro systém OS/2. Roku 1993 Sybase uvedla verzi SQL Server 4.2, která byla určená pro operační systém Windows. Byla to klasická desktopová databáze pro kanceláře a malé firmy. Teprve až v roce 1994 koupil Microsoft tento produkt a začal ho vyvíjet podle svého. [13]

SQL Server 6.05 byl představen roku 1995 už plně pod Microsoftem. Byl určen do segmentu small business a bylo možné jej využívat i u internetových aplikací. Pro Windows NT vyšla verze SQL Server 6.5 v roce 1996. U verze SQL Server 7.0 z roku 1998 bylo kompletně přepsáno a optimalizované jádro. Tuto verzi je možné označit přívlastkem „webová databáze“ a začínala konkurovat příznivou cenou databázím Oracle a IBM DB2. Podporu Business Intelligence přinesla verze SQL Server 2000 a verze SQL Server 2005 představovala její inovaci. [13]

Verze SQL Server 2008 a jeho vylepšená reedice 2008 R2 přinesly další řadu vylepšení jako kompresi údajů a záloh, technologie pro práci s geometrickými a geografickými údaji, transparentní šifrování, nové datové typy pro práci s datem a časem apod. [13]

2.2 Microsoft SQL Server 2012

Microsoft SQL Server 2012 je ve třech hlavních edicích a každá edice má 32bitovou a 64bitovou verzi. [13]

2.2.1 Standard edition

Tato edice je určena pro provoz aplikací nejen databázových firemních oddělení, ale i Business Intelligence aplikací pro menší firmy a organizace nebo jejich oddělení. Standard edition podporuje 16 procesorových jader, 64 GB RAM a jeden virtuální stroj. Tato edice se používá pro aplikace firemních oddělení požadující dobrou spravovatelnost a jednoduché použití, aplikace pro online zpracování transakcí v malém až středním objemu nebo pro systémy pro podporu rozhodování. [13]

2.2.2 Enterprise edition

SQL Server 2012 Enterprise splňuje vysoké nároky podnikových aplikací, ať už pro online zpracování transakcí v datových centrech nebo pro datové sklady. Díky technologiím, které ochraňují data před nákladnými lidskými chybami, zajišťuje obchodní kontinuitu a zkracuje dobu obnovení po havárii. Umožňuje nasazení v rámci privátního cloudu. Podporuje velké centralizované Business Intelligence řešení. [13]

Splňuje požadavky legislativních norem, ochrany dat před neoprávněným přístupem a ochrany osobních údajů. Enterprise nabízí správu infrastruktury s cílem snížit provozní náklady a omezit údržbu a správu velkých objemů dat na minimum. Tato verze se hodí pro provoz nenahraditelných aplikací pro správu dat se škálovatelností, vysokou dostupností a zabezpečením na podnikové úrovni a pro práci s velkými objemy dat. [13]

2.2.3 Business Intelligence edition

Verze Business Intelligence, jak již název vypovídá, je určena pro analýzy, multidimenzionální databáze a dolování dat v podnikové praxi. Nabízí kompletní sadu škálovatelných Business Intelligence funkcí včetně Power View a PowerPivot. Je určena pro firmy nevyžadující výkonné online transakční zpracování. [13]

2.2.4 Specializované edice

2.2.4.1 Express Edition

Bezplatná edice pro studium, vytváření aplikací pro klientské počítače a malé servery a distribuci nezávislými výrobci softwaru. Edice je limitována 1 GB paměti, 10 GB velikosti databáze a využití jednoho procesoru. Tato edice je plně kompatibilní se všemi ostatními edicemi. [13]

2.2.4.2 Developer edition

Developer edition obsahuje vše co edice Enterprise, ale licence je pro účely vývoje a testování. Je tedy primárně určena pro vývojáře. [13]

2.2.4.3 *Web edition*

Web edition nabízí nízké náklady a vysokou škálovatelnost pro webové aplikace či hostovaná řešení a internetové prostředí webových služeb. [13]

II. PRAKTICKÁ ČÁST

3 PRÁCE S MICROSOFT SQL SERVEREM 2012

V následujících kapitolách je popsána instalace Microsoft SQL Serveru 2012, použití SQL Management Studia a použité příkazy SQL. Příklady a syntaxe k příkazům se nacházejí v prezentacích. Zde se vyskytuje pouze jejich teorie.

3.1 Instalace Microsoft SQL Serveru 2012

Popis instalace Microsoft SQL Serveru 2012 je v prezentaci C01. Obsahem prezentace jsou jednotlivá okna instalace. Cílem této prezentace je jednoduché nainstalování Microsoft SQL Serveru 2012 na Váš počítač.

3.2 SQL Management Studio

SQL Management Studio se spustí klepnutím na tlačítko Start → Všechny programy → Microsoft SQL Server 2012. Po spuštění programu se objeví následující okno

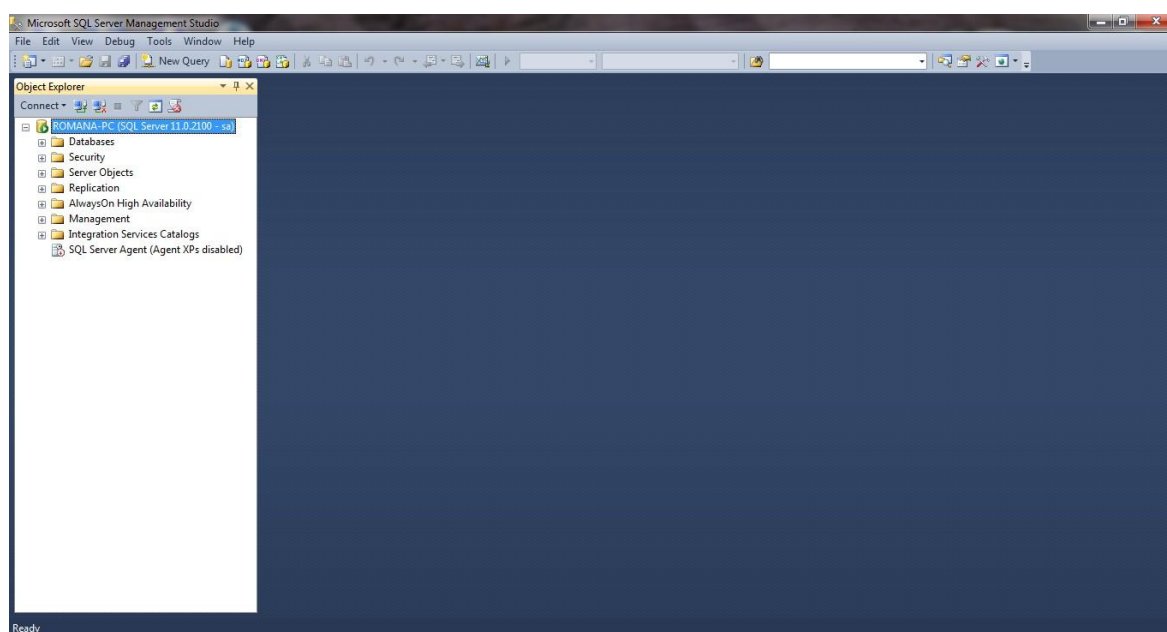


Obr. 14. Připojení lokálním účtem Windows

Po stisknutí tlačítka Connect se připojíte lokálním účtem Windows. Účet s nejvyššími právy je SA (sysadmin), přihlásit se na něj je možné změnou Authentication na SQL Server Authentication, vyplněním kolonky Login vložením slova sa a zadáním hesla. Server name je název vašeho počítače. [10]



Obr. 15. Připojení jako systémový administrátor

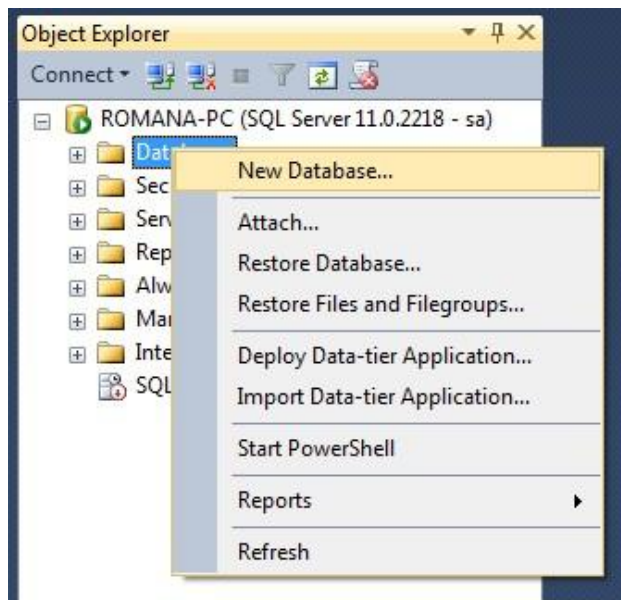


Obr. 16. Microsoft SQL Studio

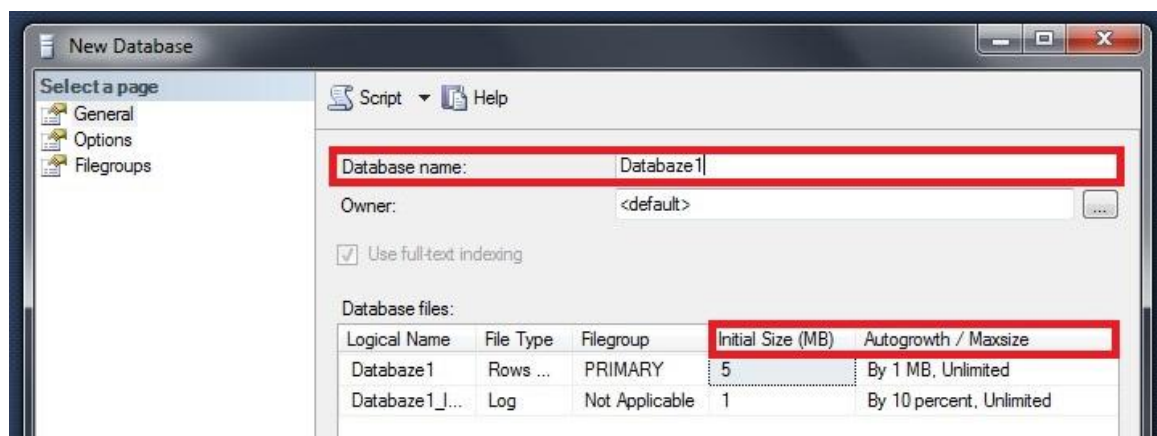
3.3 Vytvoření databáze

Novou databázi lze v SQL Management Studiu vytvořit 2 způsoby:

- 1. způsob – kliknutím pravého tlačítka na **Databases** v Object Explorer a zvolením možnosti **New Database...** [10]



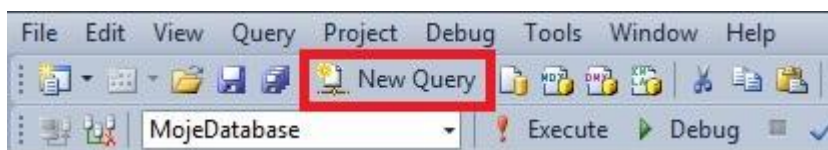
Obr. 17. Vytvoření nové databáze



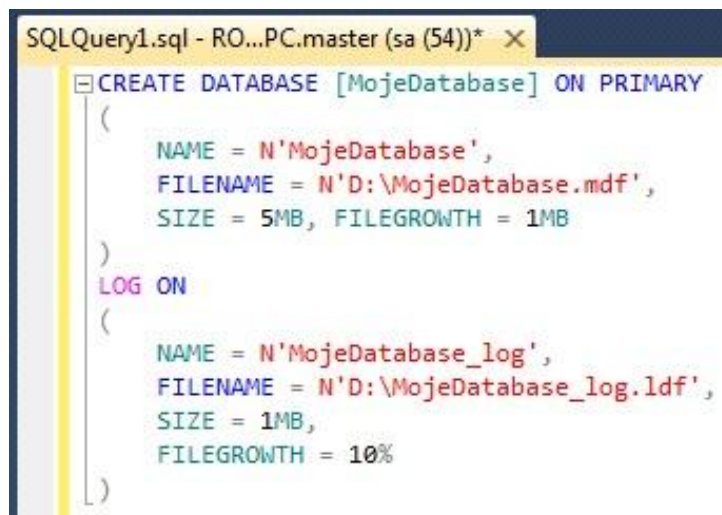
Obr. 18. Nová databáze

- Název databáze se vepíše do kolonky Database name.
- Initial Size (MB) – výchozí velikost databáze.
- Autogrowth/Maxsize – při překročení výchozí velikosti se soubor s databází bude zvětšovat o zvolenou velikost až do velikosti Maxsize. [10]

- 2. způsob – kliknutím na **New Query** se otevře okno, do kterého se vepíše příkaz, který vytvoří novou databázi. [10]



Obr. 19. New Query



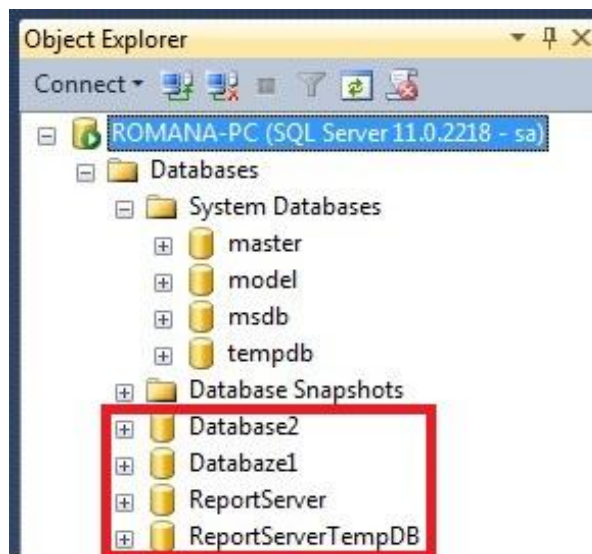
Obr. 20. Příkaz pro vytvoření databáze [10]

CREATE DATABASE – příkaz pro vytvoření databáze:

- Název databáze – může být dlouhý maximálně 128 znaků a musí být jedinečný v rámci instance SQL Serveru
- ON PRIMARY – vytvoří se v primárním datovém souboru
- FILENAME – cesta a jméno datového souboru (přípona mdf)
- SIZE – počáteční velikost
- FILEGROWTH – přírůstek datového souboru
- LOG ON – nastavení pro LOG – parametry ve stejném významu [14]

3.3.1 Uživatelské databáze

- Databáze, které se vytvořili prvním nebo druhým způsobem. [10]

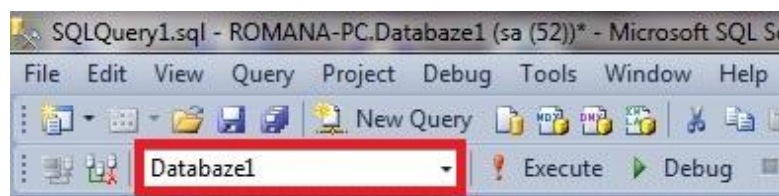


Obr. 21. Objekt Explorer

3.4 Vytvoření tabulky

Před vytvořením tabulky je potřeba nastavit databázi, do které se tabulka vytvoří.

- 1. způsob – rozevírací nabídka



Obr. 22. Nastavení databáze

- 2. Způsob – příkaz USE [JmenoDatabaze] [10]

Obecné vlastnosti tabulky:

- Název tabulky – musí být jedinečný a nesmí obsahovat diakritiku
- Název sloupce – musí být v dané tabulce jedinečný a nesmí obsahovat diakritiku
- Datový typ – definuje hodnoty, které mohou být vloženy do jednotlivých sloupců
- Primary key – neboli primární klíč je pole, které jednoznačně identifikuje každý záznam v tabulce

- Identity (x, y) – hodnota v tomto sloupci se bude generovat automaticky, první hodnota bude hodnota x, další hodnota se vygeneruje přičtením hodnoty y k hodnotě x
- NULL/NOT NULL – určuje, zda může sloupec obsahovat nulové nebo nenulové hodnoty [9]

3.4.1 Příkazy pro změnu tabulky

Příkaz ALTER TABLE slouží pro změnu již vytvořené tabulky. Je možné přidat sloupec (ADD), vymazat sloupec (DROP) nebo změnit sloupec (ALTER). Odstranit tabulku je možné příkazem DROP TABLE. [12]

Sp_RENAME slouží k přejmenování sloupců a tabulek. Nedoporučuje se používat tento příkaz k přejmenování uložených procedur, triggerů, uživatelsky definovaných funkcí nebo pohledů; raději je lepší odstranit objekt a znovu jej vytvořit s novým názvem. [15]

3.5 Datové typy

3.5.1 Číselné datové typy

- decimal – rozsah datového typu je od $-(10)^{38} - 1$ do $10^{38} - 1$
- float – datový typ s pohyblivou desetinnou čárkou v rozsahu od $-1.79E + 308$ do $1.79E + 308$
- double – datový typ s pohyblivou desetinnou čárkou s dvojnásobnou přesností
- real – datový typ s pohyblivou desetinnou čárkou v rozsahu od $-3.40E + 38$ do $3.40E + 38$ [2]

3.5.2 Celočíselné datové typy

- bit – jen dvě hodnoty 0 nebo 1
- integer (int) – celé číslo v rozsahu -231 do 231 -1, zabírá 4 bajty
- smallint – celé číslo v rozsahu -215 do 215 -1, zabírá 2 bajty
- tinyint – celé číslo v rozsahu 0 – 255, zabírá 1 bajt [2]

3.5.3 Znakové datové typy

- `char(délka)` – `char` slouží pro uložení textového řetězce pevně dané délky uvedené v závorkách
- `nchar(délka)` – tento datový typ, stejně jako `char`, slouží pro uložení textového řetězce pevné délky, která je dána parametrem v závorkách ve vybrané národní znakové sadě
- `varchar(délka)` – slouží pro uložení textového řetězce proměnlivé délky, maximální délka je dána parametrem
- `nvarchar(délka)` – tento datový typ ukládá textový řetězec proměnlivé délky ve vybrané národní znakové sadě, maximální délka řetězce je dána parametrem v závorkách [2]

3.5.4 Datové typy pro uložení hodnot data a času

- `date` – určuje hodnotu roku, měsíce a dne data ve formátu `rrrrmmdd`
- `time` – definuje hodnoty hodin, minut a vteřin ve formátu `hhmmvv`, neobsahuje desetinná čísla, dokud nejsou zadána, např. `TIME (3)` poskytne 3 desetinná čísla
- `smalldatetime` – ukládá hodnoty data a času s vteřinami vždy zaokrouhlenými na 00
- `datetime` – datový typ, který ukládá zároveň datum a čas
- `datetime2` – také ukládá zároveň datum a čas, ale s přesností na 100 nanosekund
- `datetimeoffset` – tento datový typ rovněž ukládá zároveň datum a čas a má navíc podporuje časová pásma pro globálně nasazené aplikace [16]

3.6 Práce s daty

- `INSERT` - pro vkládání záznamů do tabulky
- `SELECT` - pro zobrazení a výběr vložených dat
- `UPDATE` - pro aktualizaci vložených dat
- `DELETE` - pro vymazání dat [12]

3.7 Výběr záznamů

Pomocné příkazy vykonává příkaz SELECT v uvedeném pořadí:

1. FROM (výběr zdrojové tabulky nebo tabulek)
2. WHERE (filtrovací podmínka – výběr řádků)
3. GROUP BY (seskupení)
4. HAVING (předpis pro výběr agregovaných záznamů)
5. ORDER BY (seřídění) [2]

V příkazu SELECT je možné definovat sloupec, který se vytvoří výpočtem z již existujících sloupců, tento sloupec je možné pojmenovat pomocí alias (**AS**). K definici vypočteného sloupce lze použít i volání některého složitějšího programu, který daný sloupec vypočte. AS se také může použít na dočasné přejmenování již pojmenovaných sloupců tak, aby byl výpis přehlednější. Pokud umístíte alias do dvojitých uvozovek, můžete použít i mezery. Příkaz bude fungovat i bez klíčového slova AS. Alias je možné použít i pro přidání konstantního sloupce databáze. [2]

Pomocí příkazu **SELECT DISTINCT** se z výsledné množiny výsledků odstraní duplicitní řádky. Tento příkaz se vztahuje na kombinaci všech sloupců, které jsou za klauzulí uvedeny. Příkaz **SELECT ALL** slouží k výběru všech řádků včetně redundancí, tento výběr je zadán defaultně, proto se ALL nemusí používat. Příkaz **SELECT TOP** stanovuje počet záznamů, které budou do výpisu zahrnuty. Tento příkaz má význam u seříděných tabulek, kdy se vypíše např. 100 zákazníků s nejvyšším obratem. [2]

Klíčové slovo **FROM** udává, kde se požadované informace nacházejí. Může to být jedna tabulka, více tabulek, pohled. [2]

Podmínka **WHERE** slouží jednak k omezení vstupních dat, jednak pomocí podmínky můžeme stanovit předpis pro výběr údajů z více databázových tabulek. [2]

Operátory **LIKE** a **NOT LIKE** slouží v podmínce WHERE pro porovnání řetězců v širším měřítku. Speciální znaky používané při vyhledávání podřetězců:

- %(procenta) – libovolný počet libovolných znaků
- _(podtržítka) – jeden libovolný znak [9]

Další operátory v podmínce WHERE:

Operátor	Popis	Operátor	Popis
=	Rovná se	!= nebo <>	Nerovná se
<	Je menší než	!<	Ne méně než
>	Je větší než	!>	Ne více než
<=	Je menší než nebo roven	IS NOT NULL	Neobsahuje hodnotu NULL
>=	Je větší než nebo roven	IS NULL	Může obsahovat hodnotu NULL
OR	Aspoň 1 podmínka je splněna	BETWEEN	V rozmezí
AND	Všechny podmínky jsou splněny	NOT BETWEEN	Mimo rozmezí
NOT	Podmínka není splněna		

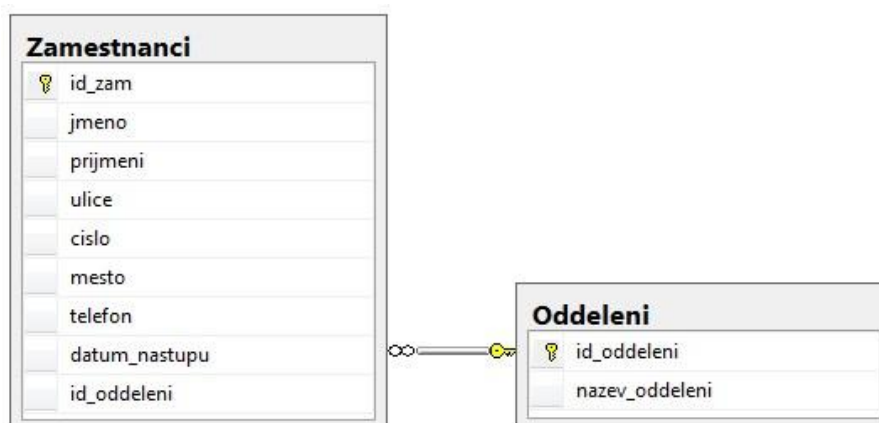
Tab. 11. Operátory v podmínce WHERE [2]

Pro seskupování údajů, na základě určitého kritéria, se v příkazu SELECT používá klauzule **GROUP BY**. Podmínka **HAVING** určuje předpis pro výběr agregovaných záznamů. Klauzule **HAVING** slouží k vyloučení skupinových výsledků, na rozdíl od klauzule WHERE, která slouží pro omezení vstupních dat. [2]

ORDER BY slouží k třídění výpisu. Pro vzestupné třídění se používá klauzule ASC, ale ta je defaultně nastavena. Pro sestupné třídění se používá klauzule DESC. [2]

3.8 Vazby mezi tabulkami

Mezi tabulkami mohou být tyto vazby: 1:1, 1:N nebo N:M. [9]

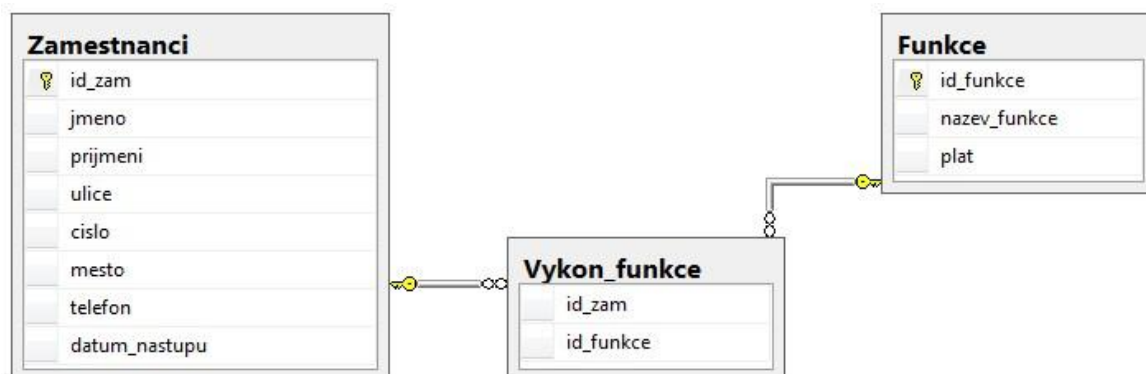


Obr. 23. Vazba 1:N

	id_zam	jmeno	prijmeni	ulice	cislo	mesto	telefon	datum_nastupu	id_oddeleni
1	1	Jan	Novak	Nova ulice	123	Olomouc	NULL	1994-12-20	1
2	2	Karel	Novy	Stefanikova	45	Zlin	542123842	1998-01-31	3
3	3	Jan	Maly	Nova ulice	48	Olomouc	589632478	1989-04-01	3
4	4	Martin	Vesely	Kostelni	86	Zlin	NULL	1995-07-25	2
5	5	Vladimir	Tichy	Stefanikova	486	Zlin	145369354	1997-04-08	2
6	6	Martin	Maly	Tyrsova	63	Bmo	NULL	2000-06-01	1
7	7	Jan	Vesely	Hrbitovni	47	Bmo	NULL	2005-08-14	1
8	8	Vit	Konecny	Wolkerova	68	Olomouc	587102256	1999-09-01	2

	id_oddeleni	nazev_oddeleni
1	1	Marketing
2	2	Finance
3	3	Vyroba

Obr. 24. Vazba 1:N v datech



Obr. 25. Vazba M:N

3.9 Spojení tabulek

Dotaz **JOIN** se používá na spojení více tabulek. Spojení tabulek je možné rozdělit na:

- INNER JOIN – vnitřní spojení
- OUTER JOIN – vnější spojení
- CROSS JOIN – křížové spojení [9]

3.9.1 INNER JOIN

Je v podstatě jakékoliv přirozené nebo předepsané spojení, které není vnější. Toto spojení vylučuje nespárované řádky. Spojení, které není označeno jako vnější, je automaticky vnitřní. [9]

3.9.2 OUTER JOIN

Vnější spojení zahrnuje nespárované řádky z jedné nebo obou spojených tabulek. Pro spojení dvou tabulek A a B existují tři možné varianty vnějšího spojení.

- **LEFT OUTER JOIN** – spojení z levé strany zahrnuje všechny řádky z tabulky A spárované nebo nespárované s hodnotami ze spárovaných řádků tabulky B
- **RIGHT OUTER JOIN** – spojení z pravé strany zahrnuje všechny řádky z tabulky B spárované nebo nespárované s hodnotami ze spárovaných řádků tabulky A
- **FULL JOIN** – u tohoto spojení se vypíší i všechny řádky z obou tabulek spojené v místech shody a doplněné o NULL v místech neshody [9]

3.9.3 CROSS JOIN

Křížové spojení vytvořené CROSS JOIN vytvoří karteziánský součin, který neprovádí žádný výběr založený na srovnávání hodnot spojených tabulek. Použijí se všechny kombinace řádků. Výsledek je stejný, jako kdyby bylo provedeno spojení a nebyl do něj zahrnut žádný predikát. [9]

3.10 Funkce

Funkce jsou bloky kódu, které vykonají na základě zadaných parametrů požadovanou operaci, např. matematickou nebo statistickou, nebo např. operaci s textovým řetězcem, a v případě bezchybného průběhu vrací nějaký výsledek. Agregační funkce operují nad množinou řádků, přičemž vracejí jeden výsledek pro celou vstupní množinu údajů. Jednořádkové funkce pracují s jedním řádkem, přičemž pro každý řádek vracejí jednu hodnotu. Funkce mohou pracovat s numerickými hodnotami, řetězci, datumovými a časovými hodnotami, případně mohou sloužit pro konverzi údajů. [2]

3.10.1 Agregační funkce

- COUNT() – počet numerických hodnot ve sloupci
- SUM() – součet numerických hodnot ve sloupci
- MIN() – minimální hodnota ve sloupci
- MAX() – maximální hodnota ve sloupci
- AVG() – aritmetický průměr numerických hodnot ve sloupci [12]

3.10.2 Matematické funkce

- ABS(n) – vrací absolutní hodnotu čísla n
- ACOS(n) – vrací hodnotu úhlu v radiánech, která je výsledkem inverzní funkce kosinus s parametrem n
- ASIN(n) – vrací hodnotu úhlu v radiánech, která je výsledkem inverzní funkce sinus s parametrem n
- ATAN(n) – vrací hodnotu úhlu v radiánech, která je výsledkem inverzní funkce tangens s parametrem n
- CEILING(n) – vrací hodnotu nejmenšího celého čísla, které je větší než parametr n
- COS(n) – vrací hodnotu kosinus úhlu
- COT(n) – vrací hodnotu kotangens úhlu
- EXP(n) – vrací hodnotu n-tou mocninu čísla e
- FLOOR(n) – vrací hodnotu největšího celého čísla, která je menší než parametr n
- LOG(n) – vrací hodnotu přirozeného logaritmu parametru n
- LOG10(n) – vrací hodnotu dekadického logaritmu parametru n
- POWER(m, n) – vrací hodnotu n-té mocniny čísla m
- PI() – vrací hodnotu čísla π
- RAND() – vrací náhodné číslo z intervalu (0, 1)

- ROUND(n, m) – vrací hodnotu čísla n matematicky zaokrouhlenou na požadovaný počet desetinných míst udávaný číslem m , při záporném čísle m se čísla zaokrouhlují před desetinnou čárkou
- SIGN(n) – tato funkce informuje o znaménku čísla n , pro kladná čísla vrací 1, pro nulu vrací 0 a pro záporná čísla vrací hodnotu -1
- SIN(n) – vrací hodnotu goniometrické funkce sinus
- SQRT(n) – vrací druhou odmocninu čísla n
- TAN(n) – vrací hodnotu goniometrické funkce tangens [2]

3.10.3 Funkce pro práci s textovými řetězci

- LEN (řetězec) – vrací délku řetězce, který je zadán jako vstupní parametr
- LOWER (řetězec) – překonvertuje všechna písmena vstupního řetězce na malá písmena abecedy, ostatní znaky zůstanou nezměněny
- UPPER (řetězec) – překonvertuje všechna písmena vstupního řetězce na velká písmena abecedy, ostatní znaky zůstanou nezměněny
- CHAR (číslo) – překonvertuje binární reprezentaci znaku na znak
- LEFT (řetězec, délka) – vrací daný počet znaků, které určíme parametrem délka z levé části vstupního řetězce
- RIGHT (řetězec, délka) – vrací daný počet znaků, které určíme parametrem délka z pravé části vstupního řetězce
- LTRIM (řetězec) – odstraní všechny mezery z levé strany vstupního řetězce
- RTRIM (řetězec) – odstraní všechny mezery z pravé strany vstupního řetězce
- REPLACE (řetězec1, řetězec2 [,řetězec3]) – vrací nový řetězec, vytvořený z původního řetězce1, přičemž nahradí všechny výskyty řetězce2 řetězcem3, když parametr řetězec3 chybí, budou všechny výskyty řetězce2 v řetězci1 vymazané
- SUBSTRING(řetězec, začátek, délka) – vrací podmnožinu vstupního řetězce, která je definovaná začátkem a počtem znaků [2]

3.10.4 Funkce pro práci s datem a časem:

- DATEADD(část_data, číslo, datum) – vrací novou hodnotu data a času, vygenerovanou z původní hodnoty, kterou udává parametr datum, k této hodnotě se připočítá příslušný počet (číslo) částí, ze kterých se skládá datum a čas
- DATEDIFF(část_data, datum1, datum2) – vrací počet sekund, dní... a let podle zadaného parametru část_data, který je mezi dvěma daty, jež jsou zadána parametry datum1 a datum2
- DATENAME(část_data, datum) – vrací pojmenovanou část data zadanou prvním parametrem, pro datum zadaný druhým parametrem
- DATEPART(část_data, datum) – vrací číselnou hodnotu části data zadanou prvním parametrem, pro datum zadané druhým parametrem
- DAY(datum) – vrací číselnou hodnotu dne ze zadaného data
- GETDATE() – vrací aktuální hodnotu data a času
- GETUTCDATE() – vrací aktuální hodnotu data a času pro UTC (Universal Time Coordinate) nebo jinak označovaný jako GMT (Greenwich Mean Time) je vůči našemu času o hodinu posunutý, pokud nevezmeme v úvahu letní čas
- MONTH(datum) – vrací číselnou hodnotu měsíce ze zadaného data
- YEAR(datum) – vrací číselnou hodnotu roku ze zadaného data [2]

3.10.5 Konverzní funkce

Úlohou těchto funkcí je konverze údajů z jednoho datového typu na jiný.

Pro konverzi se používají dvě funkce:

- CAST (výraz AS datový_typ)
- CONVERT(datový_typ [(délka)], výraz [, styl])
 - Délka je pomocný parametr pro určení rozsahu datových typů nchar, nvarchar, char, varchar, binary a datového typu varbinary.
 - Parametr styl slouží pro formátování hlavně při převodu data a času nebo čísla na řetězec. [2]

3.11 Vnořené dotazy

Vnořené dotazy se např. využívají, když je potřeba zjistit nějakou informaci, a v závislosti na této informaci nějaké další informace. Takto zformulovaný vnořený SQL dotaz musí být v závorkách. Vnořený dotaz je možné umístit do hlavního SQL příkazu SELECT ke klauzulím FROM, WHERE nebo HAVING. S vybranými příkazy lze manipulovat, když se namísto SELECT použije ve vnějším dotazu příkaz INSERT, UPDATE nebo DELETE. Při vykonávání příkazu se nejdříve vykoná vnitřní dotaz a následně se jeho výsledek aplikuje na hlavní dotaz. Ve vnořeném dotazu je možné použití i agregačních funkcí. Kromě jednořádkových vnořených dotazů SQL můžeme sestavit i víceřádkové s použitím ALL, ANY a IN, popř. vícesloupcové vnořené dotazy. [2]

Operátory:

- IN – umožňuje porovnávat operand s vyjmenovanou množinou hodnot
- ANY nebo SOME – tyto operátory porovnávají testovanou hodnotu s jakoukoliv hodnotou z intervalu hodnot
- ALL – aplikuje relaci na všechny hodnoty vrácené vnitřním dotazem
- EXISTS a NOT EXISTS – se používají pro testování toho, zda vnořený dotaz vrátí nějakou hodnotu [2]

3.12 Skládání dotazů

3.12.1 UNION

Může se použít jako alternativa k vnějšmu spojení. Stačí napsat jeden dotaz pro případy, které predikát splňují a druhý pro případy, které predikát nesplňují, a výsledek potom sloučit příkazem UNION. UNION lze také použít pouze ke sloučení výsledků dvou příkazů. [9]

3.12.2 INTERSECT

Používá se pro průnik dvou a více dotazů. Operátor nachází ty řádky, které jsou společné výstupům všech individuálních dotazů. [9]

3.13 Proměnné

Deklarace se skládá ze dvou částí. Nejdříve deklarujeme jméno kurzoru a následně SQL dotaz. Pro deklaraci proměnné slouží příkaz **DECLARE**. [2]

Pro výpis textu databázového serveru se používá příkaz **PRINT**. Kromě výpisu jednoduchých textů je možné ve výpisu kombinovat statický text s obsahy proměnných pomocí operátoru zřetězení +. Je možné tento příkaz použít jen pro výpis obsahu textových proměnných. Proměnné jiných typů je nejdříve musí nekonvertovat na textové řetězce. [2]

3.14 Podmínky

- **BEGIN a END** – některé bloky příkazů se vykonávají v cyklech, některé jen za určitých podmínek
- **IF ... ELSE** – základní programová konstrukce pro tvorbu podmínek, jednotlivé bloky jsou ohraničeny pomocí **BEGIN ... END**
- **WHILE** – pro programování cyklů, tento příkaz se vykonává do té doby, až je splněna podmínka
- **BREAK** – předčasné opuštění cyklu
- **CASE** – se používá, pokud je potřeba ošetřit více stavů proměnné, vrací pouze jednu hodnotu
- **CASE** – v **SELECT** se používá k nahrazení zkratky nebo logické hodnoty hodnotou více čitelnou hodnotou [2]

3.15 Procedury

Procedury jsou malé programy v rámci relační databáze, které mohou mít vstupní i výstupní parametry. Výsledkem činnosti procedury jsou buď úpravy v databázových tabulkách, nebo nějaký výstupní parametr. Pojmenování procedur je libovolné. [13]

Příkazy:

- **CREATE PROCEDURE** – pro vytvoření procedury
- **ALTER PROCEDURE** – pro změnu procedury

- DROP PROCEDURE – odstranění procedury
- EXEC – příkaz na volání uložené procedury [2]

3.16 Triggers

Trigger je speciální druh uložené procedury, která se aktivuje v reakci na konkrétní události. Existují dva druhy triggerů: Data Definition Language (DDL) triggers a Data Manipulation Language (DML) triggers. [17]

DDL trigger reagují na příkazy měnící strukturu databáze (CREATE, ALTER, DROP atd.). Obecně platí, že se pomocí nich zjišťují prováděné změny a historie struktury databáze. [17]

DML triggery jsou kusy kódu, které se vážou ke konkrétní tabulce nebo pohledu. Kód se spouští automaticky pokaždé, když událost, ke které je trigger vázán, se vyskytne v tabulce. Není možné explicitně vyvolat trigger - jediný způsob, jak to udělat, je vykonat požadovanou akci v tabulce, ke které jsou přiřazené. [17]

Příkazy, které aktivují trigger: INSERT, DELETE nebo UPDATE. Trigger může běžet:

- BEFORE – před vykonáním operace
- AFTER – po operaci
- INSTEAD OF – namísto operace [17]

Příkazy:

- CREATE TRIGGER – vytvoření triggeru
- UPDATE TRIGGER – změna triggeru
- DELETE TRIGGER – smazání triggeru [17]

3.17 Testovací úkoly

Sada testovacích úkolů se skládá z 60 otázek. Každá otázka se týká buď teoretické, nebo praktické části. Otázka obsahuje 4 odpovědi A, B, C nebo D, z nichž je pouze jedna správná. Vedoucímu bakalářské práce byly tyto testovací úkoly soukromě předány, aby je implementoval do výukového prostředí Moodle.

3.18 Různé směry rozvoje učební pomůcky

Při čtení této bakalářské práce je dobré nejdříve začít teoretickou, a potom praktickou částí. Prezentace k teoretické části jsou obsahově stejné jako teoretická část. Praktické prezentace jsou kromě zestručněné praktické části doplněné o příklady, obrázky a cvičení. Prezentace jsou uloženy na CD-ROM disku.

Tato práce by se mohla ještě rozšířit o další ukázky kódů, příkazy SQL a teoretické články na další témata. Proto doporučuji také prostudovat literaturu, které je uvedena v závěru této práce. Mohu doporučit knihy s indexem [2], [3], [6], [9], [11], [12] a [17]. Další zajímavé informace jistě najdete i v dalších knihách týkajících se tohoto tématu.

Téma týkající se samotného Microsoft SQL Serveru 2012, jeho ovládání, nastavení a další zajímavosti, jsou popsány v knihách [10] a [13]. Zde naleznete téměř všechno nastavení, které je možné u Microsoft SQL Serveru 2012 provést. Samozřejmostí jsou také samotné webové stránky Microsoftu, kde naleznete spoustu rad a návodů nejen o SQL.

ZÁVĚR

Hlavním cílem této bakalářské práce bylo vytvoření elektronické pomůcky pro výuku Microsoft SQL Serveru. Práce je rozdělena na teoretickou a praktickou. Teoretická část pojednává o databázových systémech z obecného hlediska a o Microsoft SQL Serveru. Praktická část je zaměřená na příkazy SQL, které se objevují také ve cvičení na konci každé prezentace. Všechny informace, které jsou v práci uvedeny, jsou pro Microsoft SQL Server, ale obecné příkazy platí i pro ostatní databázové produkty.

V teoretické části se nacházejí základní obecné informace o databázích. Na každý týden připadá jedno větší téma. Témata jsou seřazena tak, aby na sebe navazovala a student je prošel od nejzákladnějších poznatků k těm složitějším. Teoretická část je tedy rozdělena na 14 prezentací, z nichž poslední je o Microsoft SQL Serveru.

V praktické části je obsažena instalace Microsoft SQL Serveru 2012, informace o Microsoft SQL Management Studio a základní příkazy SQL. Prezentace praktické části jsou rozděleny na 11 týdnů, poslední 2 týdny jsou na práci na samostatném projektu a poslední týden je zápočtový.

Při studiu těchto materiálů se nejdříve zaměřte na teoretickou a potom až na praktickou část, protože mnoho problematiky v praktické části vychází z části teoretické. V praktické části se nachází jen velmi základní SQL příkazy pro práci s SQL Serverem. Příklady a cvičení pro každý týden jsou obsažené v prezentaci. Veškeré příkazy v nich uvedené jsou vyzkoušené a měly by fungovat.

Veškeré texty jsou napsány tak, aby byly srozumitelné i úplným začátečníkům, kteří nikdy nepřišli s databázovými systémy do styku.

ZÁVĚR V ANGLIČTINĚ

The main aim of this paper is to create instructional electronic materials for training of Microsoft SQL Server 2012. The paper is divided into the practical part and the theoretical one. The theoretical part focuses on database systems in general and on Microsoft SQL Server in detail. The practical part deals with SQL instructions which are included also in the exercises at the end of each presentation. All pieces of information mentioned at this paper are assessable for Microsoft SQL Server while general instructions are useful also for other database systems.

The theoretical part contains basic general information about database systems. Each training week covers one important topic. Topics concur each other and they are sorted according their difficulty. The theoretical part is divided into fourteen presentations and the last one focuses on Microsoft SQL Server in detail.

The practical part includes an installation of Microsoft SQL Server 2012, information about Microsoft SQL Management Studio and basic SQL instructions. Presentations of the practical part are divided into eleven training weeks, the last two ones are intended for work on the individual project task while the last week is intended to be a credit week.

While studying these materials you should start with the theoretical part and continue with the practical one. Examples and exercises for every week are listed in the presentation. All mentioned instructions are approved and they should work.

The whole text is formulated with the aim to be understandable also for beginners who have never met this topic before.

SEZNAM POUŽITÉ LITERATURY

- [1] SHELDON, Robert. *SQL: Začínáme programovat*. Praha: Grada Publishing, 2005. ISBN 80-247-0999-6.
- [2] LACKO, Luboslav. *SQL Hotová řešení*. Brno: Computer Press, 2003. ISBN 80-7226-975-5.
- [3] OPPEL, Andrew. *Databáze bez předchozích znalostí*. Brno: Computer Press, 2006. ISBN 80-251-1199-7.
- [4] Relational Databases. In: *UCSF* [online]. 2006 [cit. 2014-03-27]. Dostupné z: <http://www.cgl.ucsf.edu/Outreach/bmi219/slides/swc/lec/db.html>
- [5] Trocha z historie relačních databází. In: *Managed Dedicated Server* [online]. [cit. 2014-02-06]. Dostupné z: <http://www.managed-dedicated-server.net/trocha-z-historie-relacnich-databazi.html>
- [6] BEN-GAN, Itzik, Dejan SARKA a Ron TALMAGE. *Querying Microsoft SQL Server 2012: Exam 70-461 Training Kit*. Sebastopol, California: O'Reilly Media, Inc., 2012. ISBN 978-0-7356-6605-4.
- [7] Lehký úvod – teorie databází. In: *DotNetPortal.cz* [online]. 2009 [cit. 2014-03-27]. Dostupné z: <http://www.dotnetportal.cz/clanek/60/Lehky-uvod-teorie-databazi>
- [8] Databázové modely. *Databáze* [online]. 2010 [cit. 2014-02-06]. Dostupné z: <http://www.databaze.chytrak.cz/modely.htm>.
- [9] GRUBER, Martin. *Mistrovství v SQL*. Praha: Softpress, 2004, 2 sv. (480, 494 s.). ISBN 80-864-9762-3.
- [10] STANEK, William R. *Microsoft SQL Server 2012: kapesní rádce administrátora*. Brno: Computer Press, 2013. Microsoft (Computer Press). ISBN 978-80-251-3797-0.
- [11] ATKINSON, Paul a Robert VIEIRA. *Beginning Microsoft SQL Server 2012 programming*. Indianapolis, IN: John Wiley, 2012. ISBN 978-1-11-8262-16-0.
- [12] POKORNÝ, Jaroslav a Michal VALENTA. *Databázové systémy*. Praha: České vysoké učení technické v Praze, 2013, 265 s. ISBN 978-80-01-05212-9.

- [13] LACKO, Ľuboslav. *Mistrovství v SQL Server 2012*. Brno: Computer Press, 2013. ISBN 978-80-251-3773-4.
- [14] CREATE DATABASE (Transact-SQL). In: *Microsoft TechNet* [online]. 2014 [cit. 2014-04-14]. Dostupné z: <http://technet.microsoft.com/cs-cz/library/ms176061%28v=sql.100%29.aspx>
- [15] Sp_rename (Transact-SQL). In: *Microsoft TechNet* [online]. 2014 [cit. 2014-04-14]. Dostupné z: <http://technet.microsoft.com/en-us/library/ms188351.aspx>
- [16] Smalldatetime (Transact-SQL). In: *Microsoft TechNet* [online]. 2014 [cit. 2014-04-28]. Dostupné z: <http://technet.microsoft.com/cs-cz/library/ms182418%28v=sql.100%29.aspx>
- [17] ATKINSON, Paul, Robert VIEIRA a Ron TALMAGE. *Beginning Microsoft SQL Server 2012 programming: exam 70-461 training kit*. Indianapolis, IN: John Wiley, 2012. ISBN 978-111-8262-160.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SQL	Structured Query Language
SŘBD	Systém řízení báze dat
DBMS	Database Management System
IBM	International Business Machines Corporation
IMS	Information Management Systém
Codasyl	Data Systems Languages
SEQUEL	Structured English Query Language
ISO	International Organization for Standards
ANSI	American National Standards Institute
T-SQL	Transact-SQL
E-R model	Entitně-relační model
E-R diagram	Entitně-relační diagram
1NF	První normální forma
NFNF	Non-first normal form
2NF	Druhá normální forma
3NF	Třetí normální forma
4NF	Čtvrtá normální forma
5NF	Pátá normální forma
ACID	Atomicity, Consistency, Isolation, Durability
INT	Integer
DLL	Data Definition Language
DML	Data Manipulation Language

SEZNAM OBRÁZKŮ

<i>Obr. 1. Interakce se SŘBD [4]</i>	14
<i>Obr. 2. Evoluce T-SQL [6]</i>	16
<i>Obr. 3. Hierarchický model [7]</i>	16
<i>Obr. 4. Síťový model [7]</i>	17
<i>Obr. 5. Vrstvy datové abstrakce [3]</i>	19
<i>Obr. 6. Entita s atributy [2]</i>	21
<i>Obr. 7. Relace mezi více entitami [2]</i>	21
<i>Obr. 8. Vytvoření pohledu z jiné tabulky [2]</i>	22
<i>Obr. 9. Vytvoření pohledu z více tabulek [2]</i>	23
<i>Obr. 10. Kardinalita vztahů mezi entitami [2]</i>	25
<i>Obr. 11. Klasický životní cyklus vývoje systémů [3]</i>	28
<i>Obr. 12. Hardwarová a softwarová prostředí při vývoji [3]</i>	30
<i>Obr. 13. Zjednodušený pohled na architekturu Microsoft SQL Server [3]</i>	34
<i>Obr. 14. Připojení lokálním účtem Windows</i>	45
<i>Obr. 15. Připojení jako systémový administrátor</i>	46
<i>Obr. 16. Microsoft SQL Studio</i>	46
<i>Obr. 17. Vytvoření nové databáze</i>	47
<i>Obr. 18. Nová databáze</i>	47
<i>Obr. 19. New Query</i>	48
<i>Obr. 20. Příkaz pro vytvoření databáze [10]</i>	48
<i>Obr. 21. Objekt Explorer</i>	49
<i>Obr. 22. Nastavení databáze</i>	49
<i>Obr. 23. Vazba 1:N</i>	53
<i>Obr. 24. Vazba 1:N v datech</i>	54
<i>Obr. 25. Vazba M:N</i>	54

SEZNAM TABULEK

<i>Tab. 1. Databázová tabulka [2]</i>	22
<i>Tab. 2. Tabulka v NFNF [2]</i>	26
<i>Tab. 3. Tabulka v INF [2]</i>	26
<i>Tab. 4. Tabulka nesplňující 2NF [2]</i>	26
<i>Tab. 5. Tabulka v 2NF [2]</i>	26
<i>Tab. 6. Tabulka v 2NF [2]</i>	27
<i>Tab. 7. FILMY1 [12]</i>	32
<i>Tab. 8. Výsledek výrazu FILMY1(ROK > 1990) [9]</i>	32
<i>Tab. 9. Výsledek výrazu FILMY1[JMÉNO_F] [12]</i>	33
<i>Tab. 10. Výsledek výrazu FILMY1 (ROK < 1990) [REŽISÉR] [12]</i>	33
<i>Tab. 11. Operátory v podmínce WHERE [2]</i>	53

SEZNAM PŘÍLOH

Příloha P I: Seznam příloh na disku cd-rom

PŘÍLOHA P I: SEZNAM PŘÍLOH NA DISKU CD-ROM

Přiložený disk CD-ROM obsahuje prezentace, které jsou rozdělené na 14. týdnů. Každá ze složek obsahuje teoretickou a praktickou prezentaci. V týdnech, ve kterých je cvičení, jsou ukázkové řešení.