

# **Učební pomůcka pro Microsoft SQL Server 2014**

Jakub Kolář



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2015/2016

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jakub Kolář**  
Osobní číslo: **A12028**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **prezenční**

Téma práce: **Učební pomůcka Microsoft SQL Server 2014.**

Téma anglicky: **Lecture Notes for the Microsoft SQL Server 2014**

Zásady pro vypracování:

1. Seznamte se s vývojovým prostředím MS SQL Server 2014.
2. Uvedte nové funkce aktuální edice.
3. Popište jednotlivé edice MS SQL Server 2014 a rozdíly mezi nimi.
4. Zpracujte podklady pro výuku, zahrnující přednášky i cvičení, v rozsahu 14 týdnů.
5. V podkladech vytvořte vzorové příklady i s řešením.
6. Vytvořte 60 testových otázek vhodných k importu do systému MOODLE.

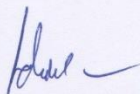
Rozsah bakalářské práce: -  
Rozsah příloh: -  
Forma zpracování bakalářské práce: tištěná/elektronická

Seznam odborné literatury:

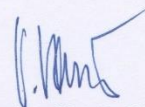
1. JORGENSEN, Adam. Professional microsoft sql server 2014 administration. 1st edition. Indianapolis, IN: John Wiley and Sons, 2014, pages cm. ISBN 1118859138.
2. BEN-GAN, Itzik, Dejan SARKA a Ron TALMAGE. Querying Microsoft SQL Server 2012: exam 70-461 training kit. Sebastopol, Calif.: Microsoft, c2012, xxx, 704 p. ISBN 0735666059.
3. KROENKE, David a David J AUER. Databáze. 1. vyd. Brno: Computer Press, 2015, 496 s. ISBN 978-80-251-4352-0.
4. MASOOD-AL-FAROOQ, B. A. SQL Server 2014 Development Essentials. UK.: Packt Publishing, 2014. ISBN 978-1-78217-255-0.
5. CORONEL, Carlos a Steven MORRIS. Database systems: design, implementation, and management. 11e [edition]. United States: Course Technology Cengage Learning, 2015, xxvii, 751 pages. ISBN 128519618x.

Vedoucí bakalářské práce: Ing. Petr Šilhavý, Ph.D.  
Ústav počítačových a komunikačních systémů  
Datum zadání bakalářské práce: 19. února 2016  
Termín odevzdání bakalářské práce: 27. května 2016

Ve Zlíně dne 19. února 2016



doc. Mgr. Milan Adámek, Ph.D.  
děkan



prof. Ing. Vladimír Vašek, CSc.  
ředitel ústavu

### **Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/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 diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/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 diplomovou/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 – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/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 diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/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 diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne: 25. 5. 2016

  
.....  
podpis diplomanta

## **ABSTRAKT**

Cílem této bakalářské práce je vytvořit učební pomůcku pro výuku Microsoft SQL Serveru 2014. Práce je rozdělena na dvě části: teoretická a praktická, kdy v teoretické části nalezneme úvod do problematiky databázových systémů a objasnění základních pojmů. Dále je v teoretické části charakterizován databázový systém Microsoft SQL server 2014, jeho novinky a dostupné edice. Část praktická se potom věnuje tvorbě SQL dotazů, popisu jednotlivých příkazů a klíčových slov pro práci s databázemi v prostředí Microsoft SQL server 2014.

Klíčová slova:

databáze, databázový systém, MS SQL server 2014, T-SQL

## **ABSTRACT**

The aim of this bachelor thesis is to create educational materials for Microsoft SQL Server 2014 training. The thesis is divided into two parts: theoretical and practical one. In the theoretical part, we can find introduction to database systems and explanation of basic concepts. There is also characteristics, news and editions of Microsoft SQL Server 2014. The practical part includes creating SQL queries, description of commands and keywords for work with databases in the Microsoft SQL Server 2014 environment.

Keywords:

database, database system, MS SQL Server 2014, T-SQL

Děkuji svému vedoucímu bakalářské práce Ing. Petru Šilhavému, Ph.D., za trpělivost a užitečné rady při vypracovávání práce.



# OBSAH

<b>ÚVOD.....</b>	<b>11</b>
<b>I TEORETICKÁ ČÁST.....</b>	<b>12</b>
<b>1 DATABÁZOVÉ SYSTÉMY .....</b>	<b>13</b>
1.1 DATABÁZE .....	13
1.2 VÝHODY DATABÁZOVÉHO PŘÍSTUPU .....	13
1.3 NEVÝHODY DATABÁZOVÉHO PŘÍSTUPU .....	14
1.4 SYSTÉM ŘÍZENÍ DATABÁZE.....	15
1.5 ZÁKLADNÍ POJMY Z DATABÁZÍ .....	15
1.5.1 Tabulky.....	15
1.5.2 Entita .....	15
1.5.3 Atributy .....	15
1.6 DATABÁZOVÉ MODELÝ .....	15
1.6.1 Hierarchický model.....	16
1.6.2 Síťový model.....	16
1.6.3 Relační model.....	16
1.6.4 Objektově orientovaný model .....	16
1.6.5 Objektově relační model .....	16
1.7 RELAČNÍ DATABÁZE.....	17
1.7.1 Hlavní složky v relační databázi .....	17
1.7.2 Typy klíčů .....	17
1.7.3 Vazby mezi tabulkami.....	18
1.7.4 Normalizace dat.....	19
1.7.5 Normální formy.....	19
1.7.6 Relační integrita .....	23
Uživatelsky definovaná integrita .....	24
1.8 JAZYK SQL .....	24
1.9 JAZYK T-SQL .....	24
1.9.1 Příkazy DDL .....	25
1.9.2 Příkazy DML.....	25
1.9.3 Příkazy DCL.....	25
1.10 TRANSAKCE .....	25
1.11 INDEXY.....	26
1.12 SPOUŠŤ (TRIGGER).....	26
1.13 ZABEZPEČENÍ A SPRÁVA DATABÁZE .....	27
1.13.1 Správa databáze.....	27
1.13.2 Autorizace .....	27
1.13.3 Autentizace.....	27
1.14 ŽIVOTNÍ CYKLUS DATABÁZOVÉHO SYSTÉMU.....	27
1.14.1 Plánování.....	28
1.14.2 Definice systému .....	28
1.14.3 Sběr a analýza požadavků .....	29
1.14.4 Návrh databáze .....	29
1.14.5 Výběr DBMS.....	29

1.14.6	Návrh aplikací .....	29
1.14.7	Vytvoření prototypu .....	29
1.14.8	Implementace .....	29
1.14.9	Konverze a načtení dat .....	30
1.14.10	Testování .....	30
1.14.11	Provozní údržba.....	30
1.15	POSTUP PŘI NÁVRHU DATABÁZE – ÚROVNĚ DATOVÉ ABSTRAKCE.....	30
1.15.1	Konceptuální návrh databáze .....	30
1.15.2	Logický návrh databáze .....	30
1.15.3	Fyzický návrh databáze .....	31
<b>2</b>	<b>MICROSOFT SQL SERVER 2014 .....</b>	<b>32</b>
2.1	MINIMÁLNÍ SYSTÉMOVÉ POŽADAVKY .....	32
2.2	NOVINKY MS SQL 2014.....	32
2.2.1	Zabezpečení – vlastní role.....	33
2.2.2	Integrace s Microsoft Azure .....	33
2.2.3	Zálohování.....	34
2.2.4	Buffer Pool Extension .....	34
2.2.5	AlwaysOn.....	34
2.2.6	Resource Governor pro IO .....	35
2.2.7	Column Store index.....	35
2.3	EDICE .....	36
2.3.1	Express .....	36
2.3.2	Business Intelligence.....	37
2.3.3	Standard.....	38
2.3.4	Podnik (Enterprise) .....	39
2.3.5	Porovnání jednotlivých edicí.....	39
2.4	VÝVOJ MICROSOFT SQL SERVERU .....	41
2.4.1	Historie .....	41
<b>II</b>	<b>PRAKTICKÁ ČÁST .....</b>	<b>43</b>
<b>3</b>	<b>PRÁCE S DATABÁZEMI V PROŠŘEDÍ SQL SERVER 2014.....</b>	<b>44</b>
3.1	INSTALACE PROGRAMU A POPIS PROSTŘEDÍ.....	45
3.2	TVORBA DATABÁZOVÝCH OBJEKTŮ .....	51
3.2.1	Vytvoření databáze.....	51
3.2.2	Smazání databáze .....	51
3.2.3	Podporované operátory .....	52
3.3	DATOVÉ TYPY .....	53
3.3.1	Znakové datové typy .....	53
3.3.2	Celočíselné datové typy .....	54
3.3.3	Plovoucí desetinná čárka.....	55
3.3.4	Datum a čas .....	56
3.3.5	Binární data .....	57
3.3.6	Řetězce Unicode.....	58
3.3.7	Priorita datových typů .....	58
3.3.8	Vytvoření tabulky.....	59
3.3.9	Smazání tabulky .....	60



3.4	PRÁCE S DATY .....	60
3.4.1	Vložení dat do tabulky .....	60
3.4.2	Smazání dat .....	61
3.4.3	Příkaz ALTER TABLE .....	61
3.4.4	Příkaz UPDATE .....	62
3.5	VÝBĚROVÉ DOTAZY .....	62
3.5.1	SELECT .....	62
3.5.2	FROM.....	62
3.5.3	WHERE.....	62
3.5.4	Konstrukce výběrového dotazu .....	63
3.5.5	Filtrace znakových dat .....	63
3.5.6	Řazení dat pomocí příkazu ORDER BY .....	64
3.5.7	Seskupování výsledků pomocí GROUP BY .....	64
3.5.8	Omezení seskupování pomocí HAVING .....	65
3.5.9	Filtrování dat pomocí TOP.....	65
3.5.10	Filtrování pomocí OFFSET – FETCH .....	65
3.5.11	Používání Aliasů .....	66
3.6	T – SQL FUNKCE V DOTAZECH.....	66
3.6.1	Agregační funkce .....	66
3.6.2	Konfigurační funkce.....	72
3.6.3	Kurzorové funkce.....	73
3.6.4	Matematické funkce .....	73
3.6.5	Funkce pro metadata .....	73
3.6.6	Řádkové funkce.....	73
3.6.7	Bezpečnostní funkce .....	74
3.6.8	Funkce pro řetězce .....	74
3.6.9	Systémové statistické funkce .....	74
3.7	SPOJOVÁNÍ TABULEK POMOCÍ JOIN .....	75
3.7.1	Křížové spojení – CROSS JOIN .....	75
3.7.2	Vnitřní spojení – INNER JOIN .....	75
3.7.3	Vnější spojení – OUTER JOIN .....	76
3.7.4	RIGHT OUTER JOIN.....	77
3.7.5	FULL OUTER JOIN.....	78
	SPOJOVÁNÍ TABULEK POMOCÍ UNION.....	79
3.7.6	UNION .....	79
3.7.7	UNION ALL .....	79
3.7.8	INTERSECT .....	79
3.7.9	EXCEPT.....	80
3.8	TABULKOVÉ VÝRAZY - TABLE EXPRESSIONS .....	80
3.8.1	Common Table Expressions.....	80
3.8.2	Pohledy – Views.....	81
3.9	PROGRAMOVÁNÍ V SQL 2014 .....	82
3.9.1	Vytváření proměnných.....	82
	Vytvoření lokální proměnné.....	82
3.9.2	Klíčová slova BEGIN & END .....	83
3.9.3	Podmínkový blok IF.....	84
3.9.4	Konstrukce CASE .....	84

3.9.5	WHILE, BREAK, CONTINUE .....	84
3.10	ULOŽENÉ PROCEDURY .....	85
3.11	VYTVÁŘENÍ TRIGERŮ .....	85
3.11.1	DML Triggery .....	85
3.11.2	DDL Triggery .....	86
3.12	UŽIVATELEM DEFINOVANÉ FUNKCE .....	86
3.12.1	Skalární funkce .....	86
3.12.2	Funkce s tabulkovou návratovou hodnotou .....	87
<b>4</b>	<b>PODKLADY PRO VÝUKU .....</b>	<b>88</b>
	<b>ZÁVĚR .....</b>	<b>89</b>
	<b>SEZNAM POUŽITÉ LITERATURY .....</b>	<b>90</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>94</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>95</b>
	<b>SEZNAM TABULEK .....</b>	<b>96</b>

## ÚVOD

I když to většinou nepostřehneme, s databázovými systémy se běžně denně setkáváme a využíváme jejich funkcí, ať už je to například v zaměstnání – docházkové systémy, evidence zaměstnanců, nebo třeba při nákupu v obchodě – evidence výrobků, databáze čárových kódů odpovídajících danému zboží. Znalost databázových systémů má proto v současné době velké uplatnění a lze i do budoucna počítat, se stále větší integrací databází, že toto uplatnění bude.

Databáze samotná představuje velké úložiště informací. Úložiště v podobě databáze musí mít určitou formu, proto cesta k získání dat z takového úložiště není složitá a má svoji logiku. Stejně tak vkládání dat a jiné administrační záležitosti jsou pomocí jazyka SQL, který popíšu v praktické části, snadné.

Tato práce by měla být základním stavebním kamenem pro začátečníky v této oblasti informační techniky. Jsou zde popsány jak teoretické, tak praktické úlohy pro běžnou práci s databázovým systémem.

V teoretické části je seznámení s problematikou v podobě pojmů a definic, které je nutné zvládnout, aby byly pochopeny funkce a principy takového systému.

Pro praktickou část jsem využil systém Microsoft SQL server 2014 – aktualizovanou verzi databázového systému, který je v praxi využíván a nabízí spoustu funkcí. V práci tento software využiju k psaní příkazů jazyka SQL a T-SQL. Popsány jsou základní a mírně pokročilé úkony, které však pro začátek bohatě postačí k plnohodnotné práci s databázovými systémy.

## **I. TEORETICKÁ ČÁST**

# 1 DATABÁZOVÉ SYSTÉMY

## 1.1 Databáze

Databázi můžeme popsat [6] jako sdílenou kolekci logicky souvisejících dat, navržená pro plnění informačních potřeb organizace, nebo jako [5] sdílenou, integrovanou strukturu, která slouží k ukládání dat koncových uživatelů a metadat (dat, která slouží k fungování databáze).

Databáze může být také jedno velké úložiště dat, která mohou být používána mnoha odděleními a uživateli. Všechna data, která jsou uživateli vyžadována, jsou integrována v databázi s minimálním množstvím duplikací. [6]

## 1.2 Výhody databázového přístupu

### Kontrola redundance dat

Databázový přístup eliminuje redundanci – duplicitu dat. Tuto vlastnost nelze odstranit úplně – například klíčové položky potřebné pro modelování relací mezi daty, nebo jindy se datové položky duplikují pro zvýšení výkonu. V těchto případech hovoříme o kontrolované redundanci dat. [6]

### Konzistence dat

Nekonzistence dat nastane, když se data, která by měla mít stejnou hodnotu, objeví na více místech, s různými hodnotami. [5] Nekonzistenci dat, lze předejít tím, že omezíme redundanci dat – ukládání stejných dat na různá místa. Pokud ovšem dojde k redundanci dat, je nutné zajistit, aby všechny kopie byly udržovány se stejnou hodnotou – konzistentní. Bohužel mnoho současných DBMS automaticky neudržuje konzistenci dat. [6]

### Sdílení dat

Pokud v podniku využíváme souborově orientovaný přístup, soubory obvykle patří lidem nebo oddělením, které je mohou využívat. [6] Přístup k takovým datům může být pro ostatní uživatele složitý, nebo i nemožný. Při použití databáze, ve které jsou data uložena, lze nejen data sdílet, ale i jednoduše řídit přístup k datům ostatním uživatelům. [5]

### Integrace dat

Integrita databáze vyjadřuje omezení, což jsou pravidla konzistence, která nesmí být v databázi porušena. Omezení se mohou vztahovat přímo na data v jednom záznamu nebo se mohou týkat vztahů mezi záznamy. Integrace umožňuje uživateli definovat a DBMS vynucovat integritní omezení. [6]

### **Nezávislost dat**

DBMS odděluje popis dat od aplikací. Když dojde ve změnách popisu dat, aplikace mohou bez změny fungovat dál. [5] Tento jev je znám jako nezávislost dat a je tím usnadněna údržba databázových aplikací. [6]

## **1.3 Nevýhody databázového přístupu**

### **Složitost**

K fungování databáze je potřeba složitý software a všichni uživatelé musí porozumět jeho funkcionalitě a možnostem, aby jej mohli plně využít. [6]

### **Pořizovací náklady**

V závislosti na požadavcích, které jsou od systému vyžadovány, velké podnikové databázové systémy mohou být velmi finančně náročné. [5]

### **Náklady přechodu**

Náklady na pořízení nového softwarového a hardwarového zařízení mohou být někdy menší, než náklady na přesun starého DBMS na nový systém. Tento fakt je většinou důvodem proč některé firmy jsou připoutány na staré verze DBMS a nemohou přejít na modernější databázovou technologii. [6]

### **Výkon**

Při souborově orientovaném systému jsou aplikace optimalizovány pro konkrétní využití – například fakturace. Tím pádem, výkon a běh aplikace bývá obvykle velmi dobrý. Oproti tomu, DBMS je napsaný univerzálně, pro více účelů. Důsledkem toho bývá, že aplikace v databázovém prostředí jsou pomalejší. [6]

### **Selhání databázového systému**

Centralizace zdrojů zvyšuje zranitelnost systému. Pokud dojde k výpadku, je nutné počítat s nedostupností dat a aplikací až do opětovného zotavení. [6]

## 1.4 Systém řízení databáze

Úkolem DBMS je vytvářet, zpracovávat a spravovat databáze. [3] Nejčastěji používanými systémy pro řízení databází jsou:

- Microsoft Access
- Microsoft SQL Server
- MySQL od Oracle Corporation
- Oracle Database od Oracle Corporation
- DB2 od společnosti IBM

## 1.5 Základní pojmy z databází

### 1.5.1 Tabulky

Tabulky jsou v databázích používány k ukládání a organizování dat. Tabulka obsahuje sloupce a řádky. Každý sloupec má nějaký datový typ, který definuje, v jakém formátu jsou data, uložená v tomto sloupci. [4]

### 1.5.2 Entita

Entity jsou objekty z reálného světa, které v databázích slouží k logickému oddělení dat. Každá entita může tvořit samostatnou tabulku. [4] Entitami jsou např. zákazníci, zaměstnanci, výrobky, apod.

### 1.5.3 Atributy

Atribut je vlastnost entity. Např. entita *Zaměstnanec* má atributy: ID Zaměstnance, Jméno Zaměstnance, Příjmení zaměstnance, Plat zaměstnance apod. Atributy se používají k organizování konkrétních údajů v rámci entity (tabulky). [4]

## 1.6 Databázové modely

Databázový model [6] je integrovaná kolekce konceptů pro popis dat, relací mezi daty a omezení dat používaných organizací.

Výběr databázového modelu je většinou první krok při návrhu databáze. Jde o architekturu, do které budeme ukládat data a vytvářet mezi nimi vztahy, podle předem daných podmínek, které záleží na konkrétním databázovém modelu. [5]



### 1.6.1 Hierarchický model

Vyvinut v roce 1960 pro správu velkého množství dat. Model je rozdělen na vrstvy, vyšší vrstva je vnímána jako nadřazená vrstvě pod ní, kterou můžeme nazvat potomkem. U tohoto modelu platí, že podřazená vrstva (potomek) může mít jenom jednu nadřazenou (jednoho rodiče), ale nadřazená vrstva může mít víc podřazených (typický příklad vazby 1:M). [5]

### 1.6.2 Síťový model

Byl vytvořen pro realizaci složitějších datových vztahů než hierarchický model, s cílem zlepšit výkon a stanovit standard databáze. V síťovém modelu se opět vyskytují data s vazbou 1:M, ale na rozdíl od hierarchického modelu, můžou záznamy mít více rodičů. I když se síťový model dnes už nepoužívá, definice standardních databázových konceptů spojené s tímto modelem se využívají dodnes. [5]

### 1.6.3 Relační model

Model, který zaznamenal velký převrat v databázových systémech, byl představen roku 1970 E. F. Coddenem ze společnosti IBM. [4] Jeho práce se jmenovala: relační model dat pro velké sdílené databanky. Pro jednoduchou představu, jednotlivé relace (tabulky) si lze představit jako matice složené z protínajících se řádků a sloupců. Každý řádek je nazván n-ticí, každý sloupec představuje atribut. Relační model také popisuje manipulaci s daty.

V době představení byla práce považována za nepraktickou, protože tehdejší výpočetní technika neměla dostatečný výkon. S exponenciálním vývojem počítačů se však tento model stal velmi používaným. [5]

### 1.6.4 Objektově orientovaný model

Stále složitější problémy z reálného světa požadovali vznik modelu, který by lépe tento reálný svět popisoval. U tohoto modelu jsou jak data, tak jejich vztahy obsaženy v jedné struktuře známé jako objekt. [5]

### 1.6.5 Objektově relační model

Poptávka pro ukládání složitějších reprezentací dat, došlo k rozšíření relačního modelu o objektově orientované prvky. Tyto generace databází umožňují jak objekty (zapouzdřenost údajů a metod), tak i rozšiřitelné datové typy na základě tříd a dědičnosti. [5]

## 1.7 Relační databáze

Relační model je založen na matematickém konceptu relace, která je fyzicky reprezentována tabulkou. [6] Tento datový model se soustředí na logickou reprezentaci dat a vztahů mezi nimi. [5]

### 1.7.1 Hlavní složky v relační databázi

Relační model dat má pět hlavních složek: [6]

**Relace** – tabulka se sloupci a řádky.

**Atribut** – pojmenovaný sloupec relace.

**Datová n-tice** – řádek relace.

**Doména** – množina přípustných hodnot pro jeden nebo více atributů.

**Relační databáze** – kolekce normalizovaných tabulek.

V relačním modelu používáme [6] relace k uložení informací o objektech, které chceme v databázi reprezentovat. Reprezentujeme relaci jako tabulku, v níž řádky tabulky odpovídají jednotlivým datovým n-ticím a sloupce tabulky odpovídají atributům.

### 1.7.2 Typy klíčů

Klíče se v relačních databázích využívají k jednoznačné identifikaci každého řádku v tabulce. Využijeme je také k vytvoření vztahů mezi tabulkami a k zajištění integrity dat. [5]

Jinými slovy, každý záznam v tabulce musí být jedinečný – musíme být schopni určit sloupce nebo kombinaci sloupců (nazývané klíče relace), které tuto jedinečnost zajišťují. [6]

#### Superklíč

Sloupec nebo množina sloupců, které jedinečně identifikují záznam v rámci relace. [5]

#### Kandidátní klíč

Superklíč, který obsahuje jen minimální počet sloupců nutných k jedinečné identifikaci záznamů. [5]

#### Primární klíč

Kandidátní klíč, který je vybrán, aby jedinečně určoval záznamy v tabulce. [6]

#### Cizí klíč

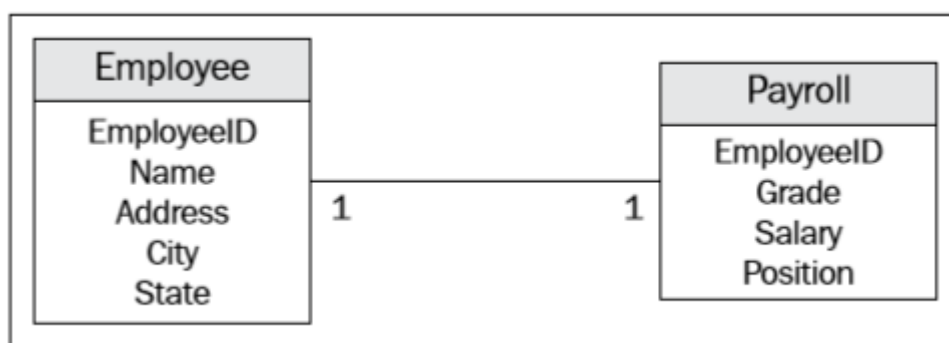
Sloupec nebo skupina sloupců v jedné tabulce, která odpovídá kandidátnímu klíči některé tabulky. [6]

### 1.7.3 Vazby mezi tabulkami

Tabulky můžeme mezi sebou provázat pomocí třech typů vazeb:

#### Vazba 1:1

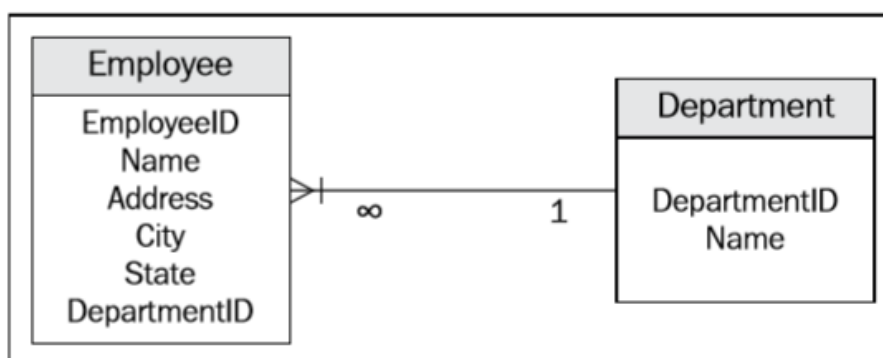
Jeden záznam z tabulky A odpovídá jednomu záznamu z tabulky B. Příklad můžeme vidět na obrázku 1 – každý zaměstnanec z tabulky Employee má jeden záznam s jeho platem v tabulce Payroll. [4] Vidíme, že tabulky jsou navzájem provázány pomocí primárního klíče EmployeeID.



Obrázek 1 – Vazba 1:1 [4]

#### Vazba 1:N

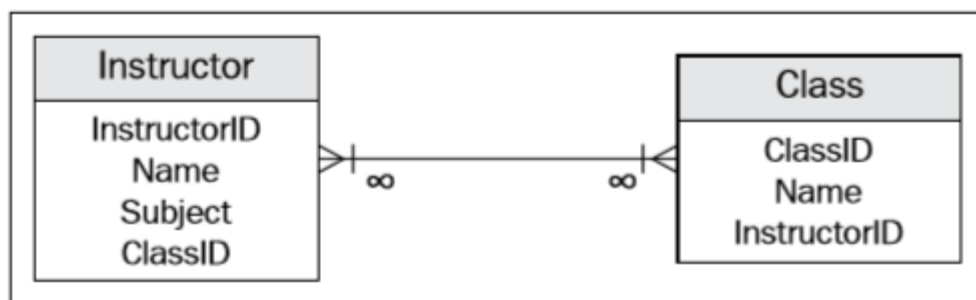
Jde o nejčastější typ vazby v relačních databázích. Jeden záznam z tabulky A odpovídá více záznamům z tabulky B. Opět na příkladu vidíme, že více zaměstnanců má jedno pracoviště. [4] Tabulka Employee je spojena pomocí cizího klíče DepartmentID s tabulkou Department, kde sloupec DepartmentID je klíčem primárním.



Obrázek 2 – Vazba 1:N [4]

### Vazba N:M

Každý záznam z tabulky A odpovídá nula nebo více záznamům v tabulce B a zároveň, každý záznam v tabulce B odpovídá nula nebo více záznamům v tabulce A. [4] Například, jeden instruktor vyučuje více tříd, a zároveň jsou třídy vyučovány mnoha instruktory, jak můžeme vidět na obrázku 3.



Obrázek 3 – vazba N:M [4]

Vazba N:M často způsobuje problémy s normalizací dat, proto se v praxi vazba N:M pomocí dalších tabulek transponuje na vazbu 1:N.

#### 1.7.4 Normalizace dat

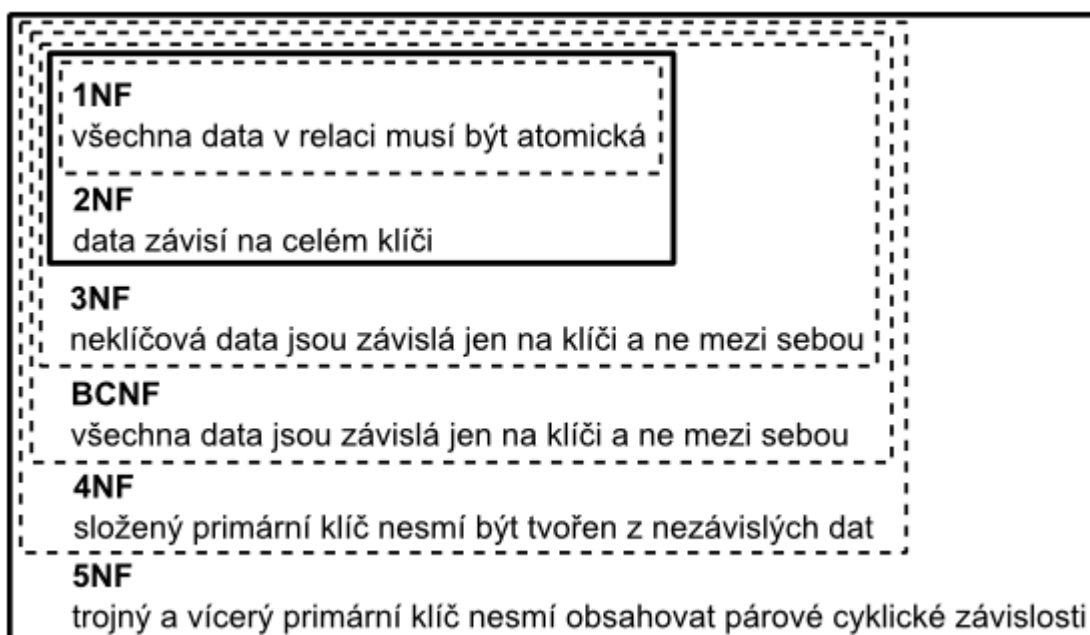
Definice říkájí, že normalizace je proces úpravy databázových tabulek k dodržování přijatých normálních forem. [4] Cílem normalizace je zajistit, aby každá tabulka vyhovovala konceptu správně formovaných vztahů. Jinými slovy, tabulky budou mít následující vlastnosti: [5]

- Každá tabulka představuje jednu entitu, např. tabulka studenti obsahuje pouze data a informace o studentech.
- Žádné datové položky nebudou zbytečně uloženy ve více než jedné tabulce. Důvodem pro tento požadavek je aktualizace dat (data aktualizujeme pouze na jednom místě).
- Všechny atributy v tabulce závisí na primárním klíči. Důvodem je jedinečná a jasná identifikace dat pomocí hodnoty primárního klíče.

#### 1.7.5 Normální formy

Normální formy jsou pravidla, která by data měla v rámci relace splňovat. [41] Čím vyšší normální forma, tím by měla být práce s daty jednodušší, např. jejich aktualizace a výběr. V praxi se při návrhu databází využívají nejvíce první tři normální formy. Bez zbylých dvou

bude návrh databáze méně dokonalým, ale funkčnost bude plně zachována. [38] Srovnání normálních forem můžeme vidět na obrázku.



Obrázek 4 – Srovnání normálních forem [41]

### První normální forma 1NF

První normální forma má za úkol: [5]

- Eliminovat opakující se skupiny v jednotlivých tabulkách.
- Vytvořit samostatné tabulky pro každou sadu souvisejících dat.
- V každé sadě souvisejících dat vytvořit identifikaci prostřednictvím primárního klíče.

Mějme tabulku Záznamy cestovní kanceláře, která obsahuje jména zákazníků, kteří si objednali zájezd, pobočku, ve které zájezd objednávali a zbylé informace o zájezdu.

Tabulka 1 – Tabulka záznamy cestovní kanceláře v nulté formě

Jméno a příjmení	Pobočka	Destinace	Hotel	Cena Zájezdu
Lucie Sochůrková	Zlín	Egypt	Magic Life	15000
Milan Kopeček	Brno	Turecko, Egypt	Jungle Aqua, Magic Life	20000, 15000
Jiří Pochylý	Praha	Turecko	Jungle Aqua	20000
Lucie Sochůrková	Zlín	Egypt	Magic Life	15000

Martin Bajer	Praha	Egypt	Magic Life	15000
--------------	-------	-------	------------	-------

Tabulka 2 – Tabulka záznamy cestovní kanceláře v první normální formě 1NF

ID	Jméno a příjmení	Pobočka	Destinace	Hotel	Cena Zájezdu
1	Lucie Sochůrková	Zlín	Egypt	Magic Life	15000
2	Milan Kopeček	Brno	Turecko	Jungle Aqua	20000
3	Milan Kopeček	Brno	Egypt	Magic Life	15000
4	Jiří Pochylý	Praha	Turecko	Jungle Aqua	20000
5	Lucie Sochůrková	Zlín	Egypt	Magic Life	15000
6	Martin Bajer	Praha	Egypt	Magic Life	15000

**Druhá normální forma 2NF**

Tabulka je v druhé normální formě, pokud je v první formě 1NF a zároveň splňuje: [5]

- Vytvoření oddělených tabulek pro sady hodnot, které zahrnují více záznamů.
- Vytvoření relací pro tyto tabulky pomocí cizího klíče.

Aby naše tabulka splňovala normu 2NF je potřeba tabulku rozdělit – v jedné budou objednávky – jméno zákazníka, pobočka. A do druhé tabulky dáme zájezdy, které budou s první tabulkou spojeny vazbou pomocí primárního a cizího klíče.

Tabulka 3 – Tabulka objednávky cestovní kanceláře 2NF

ID	Jméno a příjmení	Pobočka	ID zájezd
1	Lucie Sochůrková	Zlín	1
2	Milan Kopeček	Brno	2
3	Milan Kopeček	Brno	1
4	Jiří Pochylý	Praha	2
5	Lucie Sochůrková	Zlín	1
6	Martin Bajer	Praha	1

Tabulka 4 – Tabulka zájezdy 2NF

ID zájezd	Destinace	Hotel	Cena zájezdu
1	Egypt	Magic Life	15000
2	Turecko	Jungle Aqua	20000

**Třetí normální forma 3NF**

Třetí normální forma je splněna, když je tabulka v 1NF, 2NF a zároveň všechny její neklíčové atributy jsou navzájem nezávislé. [38]

Aby naše vzorové tabulky byly ve 3NF, provedeme další rozdělení – pobočky z tabulky objednávky cestovní kanceláře dáme do samostatné tabulky a provážíme vazbou mezi primárním a cizím klíčem.

Tabulka 5 – Tabulka objednávky cestovní kanceláře 3NF

ID	Jméno a příjmení	ID Pobočka	ID zájezd
1	Lucie Sochůrková	1	1
2	Milan Kopeček	2	2
3	Milan Kopeček	2	1
4	Jiří Pochylý	3	2
5	Lucie Sochůrková	1	1
6	Martin Bajer	3	1

Tabulka 6 – Tabulka zájezdy 3NF

ID zájezd	Destinace	Hotel	Cena zájezdu
1	Egypt	Magic Life	15000
2	Turecko	Jungle Aqua	20000

Tabulka 7 – Tabulka pobočky

ID pobočka	Město
1	Zlín
2	Brno
3	Praha



**Boyce – Coddova normální forma BCNF**

Jde o rozšíření 3NF. Většinou platí, že pokud se tabulku podaří upravit do 3NF, splňuje i BCNF. [41]

**Čtvrtá normální forma 4NF**

Čtvrtá normální forma říká, [41] že primární klíč nesmí být tvořen z nezávislých dat.

**Pátá normální forma 5NF**

Tabulka bude v 5NF, [41] pokud splňuje všechny předešlé normální formy, a zároveň trojné a vícené primární klíče neobsahují párové cyklické závislosti.

**1.7.6 Relační integrita**

Jde o pravidla, která definují přesnost dat. [6] Zajišťuje také, že jsou data spolehlivá, splňující určitý formát. [4]

**Hodnoty NULL**

Hodnotu NULL můžeme považovat za „neznámou“. Může také znamenat, že hodnotu pro konkrétní záznam nelze použít, nebo že hodnota nebyla zatím zadána. [6]

**Doménová integrita**

Doménová integrita říká, že hodnoty zadané ve sloupcích musí mít náležitý formát a hodnotu. [4] Toto dosáhneme, když budeme kontrolovat:

- Datové typy uložených hodnot.
- Formát uložených dat. (Přes kontrolní podmínky.)
- Rozsah možných hodnot. (Pomocí NOT NULL, CHECK omezení apod.)

**Entitní integrita**

Druhé pravidlo se vztahuje na primární klíče podkladových tabulek. [6] Entitní integrita je stav, ve kterém má každý řádek v tabulce svou vlastní jedinečnou identitu. Pro plné zajištění entitní integrity je potřeba splňovat dva požadavky: [5]

- 1) Každý primární klíč musí být unikátní.
- 2) Žádný z primárních klíčů v databázi nesmí obsahovat hodnotu NULL.

**Referenční integrita**

Třetí pravidlo integrity se vztahuje na cizí klíče. Pokud existuje v tabulce cizí klíč, hodnota tohoto klíče musí odpovídat hodnotě některého záznamu v domovské tabulce, nebo musí mít cizí klíč prázdnou hodnotu. [6]

### **Uživatelsky definovaná integrita**

Když potřebujeme větší přesnost dat. Uživatelsky definované integrity lze dosáhnout např. pomocí triggerů a uložených procedur. [4]

## **1.8 Jazyk SQL**

Jazyk SQL (Structured Query Language) obsahuje konstrukce, které umožňují definovat a zpracovávat databáze. Pokud bychom chtěli využít programování, musíme příkazy jazyka SQL zahrnout do skriptovacích jazyků jako je VBScript, nebo do programovacích jazyků typu Java či C#. [3]

Jazyk SQL vznikl ve společnosti IBM Corporation koncem 70. let 20. století. Vydáno bylo několik verzí, přičemž každá nová verze doplňuje nebo rozšiřuje dosavadní funkce SQL: [3]

- SQL-92
- SQL:1999
- SQL:2003
- SQL:2006
- SQL:2008

Jazyk SQL je textově orientovaný a protože vznikl dříve, než grafická uživatelská rozhraní, k psaní jeho příkazů stačí libovolný textový editor. Příkazy SQL můžeme rozdělit do několika hlavních kategorií, např: [3]

- Příkazy DDL (data definition language), použití pro definování databázové struktury.
- Příkazy DML (data manipulation language), použití pro dotazování a k úpravě dat.

## **1.9 Jazyk T-SQL**

T-SQL (Transact-SQL) je založen na jazyku SQL a je hlavním jazykem pro manipulaci a správu dat v systému Microsoft SQL Server. [2] Příkazy jazyka T-SQL můžeme rozdělit do tří kategorií: [4]

- **Příkazy DDL** (data definition language)

- **Příkazy DML** (data manipulation language)
- **Příkazy DCL** (data control language)

### 1.9.1 Příkazy DDL

T-SQL DDL příkazy obsahují klíčová slova, pomocí kterých můžeme databáze a databázové objekty vytvářet, modifikovat a mazat. [4] Do této skupiny patří příkazy:

- CREATE
- ALTER
- DROP

### 1.9.2 Příkazy DML

DML příkazy používáme pro vložení, aktualizaci, mazání a dotazování na data, která jsou uložena v SQL Server databázových tabulkách. [4] Patří sem následující klíčová slova:

- SELECT
- INSERT
- BULK INSERT
- UPDATE
- MERGE
- DELETE

### 1.9.3 Příkazy DCL

DCL příkazy slouží k udělení, odepření a odvolání oprávnění databázi a databázových objektů. [4] Klíčovými slovy v této skupině příkazů jsou:

- GRANT
- REVOKE

## 1.10 Transakce

Transakce je logická operace, která musí být dokončena. Pokud se stane, že nějaký z příkazů v transakci selže, celý proces je vrácen do stavu před transakcí. [5] Úspěšné dokončení transakce znamená přenést databázi z jednoho konzistentního stavu do druhého. Transakce musí splňovat následující vlastnosti:

- **Nedělitelnost** (Transakce nesmí být rozdělena.)

- **Konzistence** (Vyjadřuje, že transakce musí převést databázi z jednoho konzistentního stavu do druhého.)
- **Izolace** (Říká, že hodnoty, se kterými daná transakce pracuje, nesmí být souběžně použity druhou transakcí.)
- **Trvanlivost** (Změny po provedené transakce nelze vrátit zpět.)
- **Sériovost** (Sériovost požaduje, že souběžné provádění transakcí musí přinést konzistentní výsledky.)

### 1.11 Indexy

Indexy se v databázi používají pro rychlejší přístup k datům. [3] Indexy mohou být použity také pro vrácení dat v určitém pořadí, [5] například podle určitého atributu – po vytvoření indexu na sloupec příjmení zákazníka můžeme vracet data z celé tabulky, abecedně seřazené podle sloupce příjmení zákazníka.

Indexy jsou také důležité při definování primárních klíčů v tabulce, kdy DBMS automaticky vytvoří jedinečný index na sloupec primárního klíče. [5] Tabulka může mít mnoho indexů, ale každý index je spojen pouze s jednou tabulkou.

### 1.12 Spoušť (trigger)

Trigger (spoušť) je speciální druh uložené procedury, která reaguje na nějakou vzniklou událost. [4] Když událost vznikne, je automaticky systémem řízení databáze vyvolán. Triggery jsou asociovány s databázovými tabulkami, přičemž každá tabulka může mít jeden, nebo více triggerů. Možné použití: aktualizace hodnot v tabulce, vložení záznamů do tabulky, volat jiné uložené procedury [5]

Oracle například doporučuje využití spouští pro následující úlohy: [5]

- Automatické generování odvozených hodnot sloupců.
- Pro kontrolu business a bezpečnostních omezení.
- Vytváření tabulkových replik za účelem zálohování.
- Vytváření auditních záznamů.

Často jsou využívány pro udržování integrity dat, ale lze je dále využít např. pro: [4]

- Porovnání údajů před a po úpravě.
- Implementování vlastních chybových hlášek.
- Provádění administrativních úkonů, např. audit a regulaci databázových operací.

Triggery nemohou mít návratovou hodnotu ani žádné vstupní parametry. Microsoft SQL 2014 rozlišuje dva typy spouští: [4]

1. DML triggery – umožňují reagovat na jakoukoliv kombinaci příkazů INSERT, UPDATE nebo DELETE. Žádné jiné příkazy nelze v DML triggerech použít.
2. DDL triggery – spustí reakci na události, která je na úrovni definic dat, jako je např. vytváření nebo mazání objektů.

## **1.13 Zabezpečení a správa databáze**

### **1.13.1 Správa databáze**

Pod správu databáze spadá fyzická realizace databázového systému včetně fyzického návrhu a implementace, nastavení zabezpečení a kontroly integrity, monitorování výkonnosti systému a v případě potřeby aktualizace databáze. [6]

### **1.13.2 Autorizace**

Autorizace je udělení přístupových práv, která konkrétnímu uživateli nebo aplikaci umožňují přístup do databázového systému. [6]

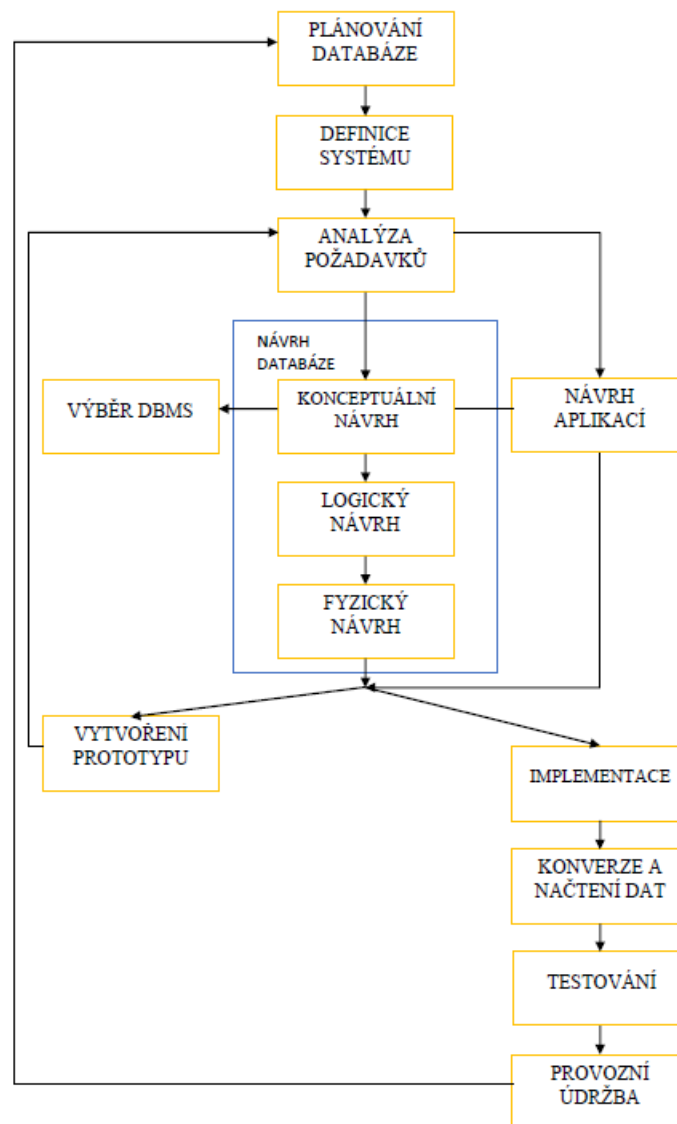
Kontrolu autorizace je možné zabudovat do softwaru, a tak kontrolovat k jakým objektům databáze mají uživatelé přístup, nebo jaké objekty mohou editovat. Proces autorizace zahrnuje autentizaci. [6]

### **1.13.3 Autentizace**

Autentizace je proces kontrolující, zda uživatel je skutečně tím, za koho se vydává. [6]

## **1.14 Životní cyklus databázového systému**

Pod tímto pojmem se skrývají jednotlivé fáze, které se při návrhu databázového systému využívají. [6]



Obrázek 5 – Životní cyklus databáze [6]

### 1.14.1 Plánování

Plánování databáze jsou činnosti managementu, které umožní v mezích možností hladký a efektivní průběh vývoje databázového systému. [6]

### 1.14.2 Definice systému

Pod tímto pojmem rozumíme určení rozsahu a hranic databázového systému včetně jeho hlavních uživatelských pohledů. Uživatelský pohled může představovat pracovní roli nebo provozní aplikační oblast. [6]

### 1.14.3 Sběr a analýza požadavků

Sbírání a analýza požadavků, které má výsledný databázový systém splňovat. [6]

### 1.14.4 Návrh databáze

Proces vytvoření návrhu, který bude plnit všechny požadavky. [6] Návrh databáze tvoří tři hlavní fáze: konceptuální, logický a fyzický návrh.

### 1.14.5 Výběr DBMS

Výběr vhodného DBMS pro podporu databázového systému. [6]

### 1.14.6 Návrh aplikací

Návrh aplikací zahrnuje návrh uživatelského rozhraní a databázových aplikací, které budou s databází pracovat. Tato fáze zahrnuje dvě hlavní činnosti: návrh transakcí a návrh uživatelského rozhraní. [6]

### 1.14.7 Vytvoření prototypu

Vytvoření fungujícího modelu databázového systému, který ale obvykle nemívá všechny požadované vlastnosti nebo neposkytuje veškerou funkčnost konečné podoby systému. Účelem prototypu je umožnit uživatelům identifikovat, které vlastnosti fungují dobře a které je naopak potřeba zlepšit. [6]

Dle současného trendu se využívají dva typy prototypů:

- Prototyp pro požadavky.
- Vývojový prototyp.

Prototyp pro požadavky, jak již z názvu patrné, se používá pro určení požadavků navrhovaného databázového systému a po jejich identifikaci je odstraněn. Vývojový prototyp má podobné využití, s tím rozdílem, že se pokračuje v jeho vývoji a stane se fungujícím databázovým systémem. [6]

### 1.14.8 Implementace

Implementace je fyzická realizace návrhu databáze a aplikací. [6]



#### **1.14.9 Konverze a načtení dat**

Tento krok je nutný pouze tehdy, když nový databázový systém nahrazuje starý a znamená přípravu všech dat do podoby vhodné pro transfer do nové databáze. [6]

V současnosti je obvyklé, že utilita pro načtení existujících dat do nové databáze je přímo součástí DBMS.

#### **1.14.10 Testování**

Proces provozu databázového systému se záměrem nalézt chyby. [6]

Před ostrým provozem by se měla nová databáze důkladně testovat. To se provádí po pečlivém naplánování testovacích strategií a s realistickými daty, aby se celý proces co nejvíce podobal reálnému provozu a otestovaly se všechny možné stavy a funkce nového databázového systému. [6]

#### **1.14.11 Provozní údržba**

Proces monitorování a údržby databázového systému po instalaci. [6]

### **1.15 Postup při návrhu databáze – úrovně datové abstrakce**

Pro ilustrování úrovně datové abstrakce [5] si můžeme představit návrh automobilu: začíná se tím, že návrháři nakreslí koncept vozu. Další technici navrhnu detaily, které převedou základní koncept do struktury, kterou lze vyrobit. A v poslední fázi jsou narýsovány výkresy pro konečnou výrobu.

Postup při návrhu databáze využívá procedury a techniky, které mají usnadnit proces návrhu databáze. [6] Začínáme od konceptuálního návrhu – nejvyšší datová abstrakce, přes logický k fyzickému – nejnižší datová abstrakce.

#### **1.15.1 Konceptuální návrh databáze**

Konceptuální fáze návrhu představuje [5] globální pohled na celou databázi. Můžeme také říct, že jde o proces, ve kterém se vytvoří model dat, který se bude v praxi využívat. [6] Při tvorbě modelu se neuvažuje o fyzické implementaci.

#### **1.15.2 Logický návrh databáze**

Při logickém návrhu se zvažuje, jaký model dat se bude v databázi využívat, výsledek je nezávislý na jakémkoliv DBMS či jiných úvahách o fyzické implementaci. [6]

### 1.15.3 Fyzický návrh databáze

Zde jsou popsány podkladové tabulky, organizace souborů, indexy používané pro dosažení efektivního přístupu k datům, všechna související integritní a bezpečnostní omezení. [6] Fyzický návrh databáze je softwarově a hardwarově závislý. [5]

## 2 MICROSOFT SQL SERVER 2014

Microsoft SQL Server je relační databázový a analytický systém. Nejnovější verze MS SQL Server byla představena 1. dubna 2014.

### 2.1 Minimální systémové požadavky

Tabulka 8 – Minimální systémové požadavky [1]

Komponenta	Požadavek
Procesor	Pro 64-bit instalaci (s rychlostí 1,4 GHz a vyšší) AMD Opteron, Athlon 64, Intel Pentium IV s podporou EM64T, Xeon s podporou EM64T
Paměť	1 GB (521 MB pro edici Express), Doporučeno 4 GB
Úložiště	Database Engine and Data files, Replication, Full-text Search and Data Quality Services: 811 MB
	Analysis Services and data files: 345 MB
	Reporting Services and Report Manager: 304 MB
	Integration Services: 591 MB
	Master Data Services: 243 MB
	Client Components 1 823 MB
	SQL Server Books Online Components to view and manage help content: 375 KB

### 2.2 Novinky MS SQL 2014

Některé funkce systému SQL 2014 jsou úplně nové, jiné už byly v předchozích verzích a došlo k jejich vylepšení či rozšíření.

### 2.2.1 Zabezpečení – vlastní role

Od verze 2012 je možné vytvářet vlastní role pro zabezpečení. U nové verze došlo k rozšíření možností práce s rolemi na úrovni serveru – přidáno několik oprávnění, která lze nastavit. Např. aby databázový administrátor nemohl číst citlivé informace z databáze. [7]

### 2.2.2 Integrace s Microsoft Azure

Microsoft Azure je tzv. Cloudová služba.

#### Cloud Computing

Nejjednodušší definice říká, že cloud computing znamená ukládání a přístup k datům a aplikacím přes internet, nikoliv přes pevný disk počítače. [9]

Výhody Cloud Computingu: [10]

- Úspora nákladů (nepotřebujeme vlastní hardwarové ani softwarové prostředky).
- Efektivita – více firem může využívat stejné aplikace a s tím souvisí také intenzivnější spolupráce, kdy např. dodavatel může mít přístup k objednávkám na cloudovém úložišti.

Nevýhody Cloud Computingu: [10]

- Zabezpečení a poškození dat.
- Spolehlivost a dostupnost služby.
- Internetové připojení je nezbytnost.
- Závislost na jednom poskytovateli.
- Uživatelské rozhraní aplikace – pouze přes webový prohlížeč.

Platforma Microsoft Azure byla poprvé představena koncem roku 2008. Dnes na ní běží tisíce aplikací od drobných webů až po komplikované cloudové služby, které pracují s terabajty dat. [12]

V Microsoft Azure jsou integrovány tyto cloudové služby: [11]

- Analytické
- Výpočetní
- Databázové
- Mobilní
- Síťové
- Úložišťové

- Webové

V SQL 2014 můžeme Azure využívat mimo zálohování také pro samotný provoz databáze. Přímě v Management Studio můžeme existující databázi přenést na cloudovou službu Azure. Do Azure lze také zálohovat datové a log soubory databází, pro takové využití však potřebujeme rychlé a spolehlivé připojení k internetu. [7]

### 2.2.3 Zálohování

Jak již bylo řečeno, zálohování nejen na disky, ale také na cloudovou službu Azure. U všech typů zálohování může být použito šifrování pro zvýšení bezpečnosti záloh. Je potřeba mít vytvořen certifikát. Bez certifikátu nebude možné databázi obnovit, je proto nutné mít tento certifikát zálohován. [7]

### 2.2.4 Buffer Pool Extension

Buffer<sup>1</sup> pool extension je nová funkce pro zvýšení výkonu serveru, dostupná u edicí Standard a Enterprise. Při zpracování dotazů jsou datové a indexové položky načítány do bufferu. Když je nedostatek paměti na serveru, jsou tyto data uložena na pevný disk a následně, když je potřeba je zpracovávat, z disku opět načteny do paměti. Tyto krátké, ale frekventované operace, velmi zatěžují disk a tím zpomalují výsledný výkon. [7]

Funkce Buffer Pool Extension umožňuje tento buffer rozšířit o SSD disk. Tím máme větší úložiště, které může dosahovat až 32násobku celkové paměti serveru. Buffer extension má i své další funkce, např. performance counters, kterými je možno sledovat zatížení a využití tohoto přídavného úložiště. [7]

### 2.2.5 AlwaysOn

AlwaysOn je technologie pro vysokou dostupnost databáze. Kombinuje dvě funkce: Database Mirroring (zrcadlení databáze) a Log Shipping (zálohování z primárního SQL server na sekundární SQL server).

---

<sup>1</sup> Buffer – tento výraz se používá pro „vyrovnávací paměť“. Obecně paměť, která slouží k zachycení dat, před jejich přesunem na jiné místo. Buffer usnadňuje komunikaci mezi zařízeními s rozdílnou rychlostí. [36]

U nové verze SQL Serveru byla služba rozšířena na možnost mít až 8 replik databáze, které je možné ukládat i do cloudové služby Azure. Tyto repliky lze používat i v případě, kdy primární replika není dostupná. [7]

### **2.2.6 Resource Governor pro IO**

Služba pro přidělování výkonu. Resource Governor není nová služba, původně ovšem bylo možné omezit paměť a procesorový výkon pro jednotlivé relace připojení k serveru. Nově je možné omezit i zatížení disků. [7]

### **2.2.7 Column Store index**

Služba pro rychlejší získávání dat z databáze. Index odpovídá nějakému místu na disku, a každý obsahuje nějaké klíče složené z jednoho nebo více sloupců. Sloupce jsou ukládány ve stromové struktuře (B-Tree) a tím je umožněno rychlé vyhledávání dat podle hodnoty klíče. [8]

Služba má dvě možnosti:

- Clustered columnstore index
- Nonclustered columnstore index

#### **Nonclustered columnstore index**

Výhoda tohoto systému uložení dat je možnost pracovat pouze s požadovanými sloupci – možnost číst z disku méně dat. Dotazy jsou proto optimalizovány tak, aby pracovaly pouze s potřebnými sloupci. Tím dochází k lepšímu využití výše zmiňovaného bufferu, protože v paměti máme pouze ta data, která potřebujeme. Rychlost také zvyšuje komprese – když sloupce obsahují obdobná data, dochází k vysokému poměru komprese. [8]

Nevýhodou je nemožnost upravovat data – tabulka s vytvořeným nonclustered columnstore indexem je read-only. Při potřebě editace dat je nutné nejdříve tento index smazat nebo deaktivovat. Nonclustered columnstore index je tedy vhodný pouze na statická data. [8]

#### **Clustered columnstore index**

Prvním rozdílem mezi Nonclustered a clustered indexem je možnost měnit data. V tomto indexu jsou také zahrnuty všechny sloupce. [8]

Tento typ indexu však nemusí vždy znamenat zrychlení systému a dále není možné použít v případě, že tabulka obsahuje sloupce s následujícími datovými typy: [8]

- Text, NText a image
- Varchar(max)
- RowVersion
- Geometry, Deography
- HierarchyID
- XML

Dále tento typ indexu nelze použít, pokud jsou v databázi použity: [8]

- Replikace
- Change tracking
- Change data capture
- Filestream

## 2.3 Edice

MS SQL 2014 je k dostání ve 4 edicích:

- Express
- Standard
- Business Intelligence
- Podnik (Enterprise)

### 2.3.1 Express

Tato distribuce může být ještě v dalších verzích:

- SQL Server Express
- SQL Server Express s nástroji
- SQL Server Management Studio
- SQL Server Express LocalDB
- SQL Server Express s pokročilými službami

#### SQL Server Express

Pouze základní databázový server, bez nástrojů a pokročilých služeb. Možné použití pro příjem vzdálených připojení nebo správy na dálku. [13]

#### SQL Server Express s nástroji

Tento balíček obsahuje základní databázi SQL Server a nástroje ke správě instancí systému SQL Server. Obsahuje: SQL Server Express, LocalDB a SQL Azure. [13]

### **SQL Management Studio**

Neobsahuje databázi SQL Server, ale pouze nástroje ke správě instancí systému SQL Server, včetně LocalDB, SQL Express, SQL Azure atd. Balíček využijeme, když už SQL Server máme a potřebujeme pouze nástroje pro správu. [13]

### **SQL Server Express LocalDB**

Neobsahuje žádné nástroje pro správu. LocalDB je jednodušší verze edice Express, která obsahuje veškeré její funkce programování. Běží v uživatelském režimu a má rychlou instalaci bez jakékoli konfigurace. Použijeme v případě, když do aplikace potřebujeme zabudovat službu SQL Server Express. [13]

### **SQL Server Express s pokročilými službami**

Nejobsáhlejší balíček z edice Express, má plnou sadu funkcí. Obsahuje modul databáze, nástroje edice Express, služby vytváření sestav, fulltextové vyhledávání, nástroje pro správu a všechny součásti edice SQL Server Express. [13]

## **2.3.2 Business Intelligence<sup>2</sup>**

Tato edice, určená pro vybudování a využívání bezpečné, rozšiřitelné a spravovatelné řešení BI, obsahuje:

### **Zkoumání a vizualizace dat**

Doplňek Power View je interaktivní aplikace s webovým rozhraním pro prohlížení, vizualizace a prezentování dat. Dalším doplňkem je Power Pivot, ten pomáhá uživatelům v přístupu k datům a jejich třídění prakticky z jakéhokoli zdroje (včetně velkoobjemových zdrojů dat), s použitím známých nástrojů. [14]

### **Analytické nástroje**

---

<sup>2</sup> Business intelligence (zkratka BI) znamená: získávání obchodně důležitých informací a znalostí potřebných pro fungování a rozhodování podniku. [42]



Za účelem dosažení flexibility a snadného používání, analytické nástroje obsažené v této edici nabízejí funkci uchování v paměti, která je zabudovaná přímo do analytických nástrojů systému SQL Server. [14]

### **Funkce pro věrohodná stabilní data**

Vylepšená služba SQL Server Integration Services a sémantický model BI, umožňující stabilní pohled na různorodé zdroje dat, zajišťují lepší řízení a správu dat. [14]

### **Stálá kvalita dat**

Kvalitu dat můžeme zlepšovat používáním organizačních znalostí a dalších poskytovatelů referenčních dat za účelem profilování, čištění a párování dat. [14]

### **Správa hlavních dat**

Pro správu hlavních dat můžeme využívat organizačních struktur používaných pro mapování objektů, referenční data a správu metadat. [14]

### **Základní dostupnost, zabezpečení**

Edice Business Intelligence obsahuje také funkce základní databáze edice SQL Server Standard pro aplikace různých oddělení, včetně základní podpory systému Windows Server a uživatelsky definovaných rolí pro nastavení rozdělení povinností. [14]

### **2.3.3 Standard**

Edice standard je určena pro základní správu dat a možnosti řešení business intelligence pro méně náročné úlohy s využitím minima prostředků IT.

Edice nabízí:

#### **Základní dostupnost a obnova po havárii**

Možnost plného využití cloudových služeb. Logickou kopii databází systému SQL Server lze přesunout přímo do databáze SQL systému Microsoft Azure za účelem ochrany dat. [15]

#### **Podpora pro dodržování shody**

Pomáhá zvýšit míru administrace a snížit složitost schématu databáze. Nabízí snadnou správu povolení přístupů k datům, včetně uživatelsky definovaných serverových rolí. [15]

#### **Reportování a analytické nástroje**

Podpora vícerozměrových modelů umožňuje vytvářet obchodní logiku se službou Reporting Services. Tvůrce sestav 3.0 poskytuje intuitivní prostředí vytváření sestav. [15]

### **Velká integrace dat**

Integrace dat z různých zdrojů pomocí služby SQL Server Integration Services, která umožňuje sledování stavu a schopnost provádět správu jako samostatnou instanci systému SQL Server. [15]

### **Služby pro administraci**

Aplikace SQL Server Management Studio a aktualizované balíčky řízení pro nástroj Systém Center. Přidaná podpora pro Windows PowerShell 2.0 automatizuje úlohy správy a aplikace Sys Prep pomáhají efektivněji vytvářet virtuální počítače. [15]

### **Vývojářské nástroje**

Pomocí nástrojů SQL Server Data Tools je vývoj databáze integrován do aplikace Visual Studio. Lze vytvářet podnikové a datové mobilní aplikace, weby, nebo business intelligence aplikace, a to místně nebo ve veřejném cloudu. [15]

### **Podpora pro správu obsahu**

Vylepšení fulltextového indexování pro lepší výkon vyhledávání. [15]

### **Jakákoli data můžeme rozšířit kamkoli**

Lze rozšířit různorodá prostředí pomocí standardních rozhraní API na různé platformy. [15]

#### **2.3.4 Podnik (Enterprise)**

Prémiová Edice SQL Server Enterprise poskytuje kompletní funkce pro špičková datová centra, náročné databáze a velké požadavky řešení business Intelligence. Tato edice obsahuje vše, co bylo zmíněno u ostatních edicí, mnohé služby jsou zde navíc i rozšířeny. Např. fyzické zpracování lze rozšířit až na 640 logických procesorů a virtuální počítače lze rozšířit až na 64 logických procesorů s 1 TB paměti RAM. [16]

#### **2.3.5 Porovnání jednotlivých edicí**

Tabulka 9 – Porovnání edic [37]

Funkce	edice			
	Podnik	BI	Standard	Express

Max. počet jader	OS Max	16 jader	16 jader	4 jádra
Max. paměť na instanci	OS Max	128 GB	128 GB	1 GB
Max velikost	524 PB	524 PB	524 PB	10 GB
Programovatelnost (T-SQL, typy dat, FileTable)	Ano	Ano	Ano	Ano
SQL Server Management Studio	Ano	Ano	Ano	Ano
Základní zabezpečení (rozdělení povinností, základní auditování)	Ano	Ano	Ano	Ano
Základní vytváření sestav	Ano	Ano	Ano	Ano
Zabudované datové konektory	Ano	Ano	Ano	Ano
Integrace dat (SSIS, přeměny návrhu)	Ano	Ano	Ano	
Základní vysoká dostupnost	Ano	Ano	Ano	
Základní podnikové řešení BI (analytické nástroje, vícerozměrový sémantický model, dolování dat)	Ano	Ano	Ano	
Samoobslužné řešení business intelligence (upozornění, Power View, Power Pivot pro SharePoint Server)	Ano	Ano		
Podniková správa dat (služby Data Quality Services, Master Data Services)	Ano	Ano		
Pokročilá integrace dat (přibližné seskupování a vyhledávání, zachycení změny dat)	Ano			
Datový sklad (ukládání sloupců v paměti, komprese, dělení)	Ano			
Pokročilá vysoká dostupnost (AlwaysOn, několik aktivních sekundárních instancí, multi-site, geo-clustering)	Ano			
Pokročilé zpracování transakcí (uchování online transakcí v paměti)	Ano			

## **2.4 Vývoj Microsoft SQL Serveru**

Vývoj SQL Serveru můžeme rozdělit na dvě zásadní období. Verze 1 až 7 jsou starší verze a jsou založeny na Sybase SQL Serveru. Až od verze 8 byl celý systém přepsán [18]

### **2.4.1 Historie**

#### **Počátky SQL Serveru**

V roce 1988 SQL Server startoval jako společná aplikace vytvořená za spolupráce s firmou Sybase. Tímto Microsoft vstoupil na trh s databázovými systémy. [17]

#### **SQL Server 4.2**

První verze pro platformu Windows NT vychází v roce 1993. Šlo o desktopovou databázovou aplikaci s nízkou funkcí, schopnou uspokojit potřeby ukládání a zpracování dat pro malá oddělení. Databázový systém byl integrován přímo v systému Windows. [17]

V roce 1994 Microsoft ukončuje spolupráci s firmou Sybase a vydává se vlastní cestou.

#### **SQL Server 6.0**

Vychází v roce 1995, nesla kódové označení SQL95. Microsoft tuhle verzi udělal významný krok dopředu – bylo přepsáno jádro databázového engine, což umožnilo lepší výkon a zlepšení funkcí. Verze byla určena pro malé firemní databáze. Byla možnost pracovat v omezené míře s e-commerce a také s intranetovými aplikacemi, to vše za zlomek ceny oproti jeho konkurentům. [17]

#### **SQL Server 6.5**

V roce 1996 se Microsoft s verzí SQL Server 6.5 s kódovým označením Hydra dostává do popředí tak, že Oracle reaguje a nasazuje svoji verzi 7.1 pro platformu NT jako přímou konkurenci. [17]

#### **SQL Server 7.0**

Dalším významným datem byl rok 1998, kdy byla uvedena verze 7.0 s označením Sphinx. Opět byl přepsán databázový engine a přibyla možnost webové databáze. Verze poskytovala i business nástroje např. analytické služby, transformaci dat, což byly v té době u konkurenčních firem velmi drahé doplňky. [17]

#### **SQL Server 2000**

32bitová verze 8.0 s kódovým označením Shiloh nabízela vylepšení výkonu, škálovatelnosti a spolehlivosti SQL serveru. Přibyla podpora on-line obchodních operací. Sice došlo k nárůstu cen za jednotlivé edice, ale pořád byly poloviční oproti konkurenčnímu Oracle. Proto Microsoft postupně ovládl trh s databázemi. Např. v roce 2001 měl Oracle podíl na trhu s databázemi 34%, zatím co SQL Server 40%. O rok později, 2002 byl podíl Microsoftu 45% a Oracle klesl na 27%. [17]

### **SQL Server 2005**

Kódové označení Yukon, verze 9.0. Nové funkce – schopnost načítat data přes službu Integration Services, dále analysis services, Java Database Connectivity, Notification services, Reporting Service... Největší skok vpřed znamenalo zavedení .NET Framework do SQL pro vytváření specifických objektů pro SQL Server [17]

### **SQL Server 2008**

Verze 10.0 pod kódovým označením Katmai přinesla:

- Rozšíření T-SQL (MERGE klauzule, Table-Value type, mnohonásobný INSERT).
- Nové datové typy (Geografické, Geometrické, Datum a čas, hierarchické).
- Nové vlastnosti (filestream)

### **SQL Server 2008 R2**

Verze 10.5, kódového označení Kilimanjaro byla uvedena v roce 2010.

### **SQL Server 2012**

Verze 11.0, kódové označení Denali.

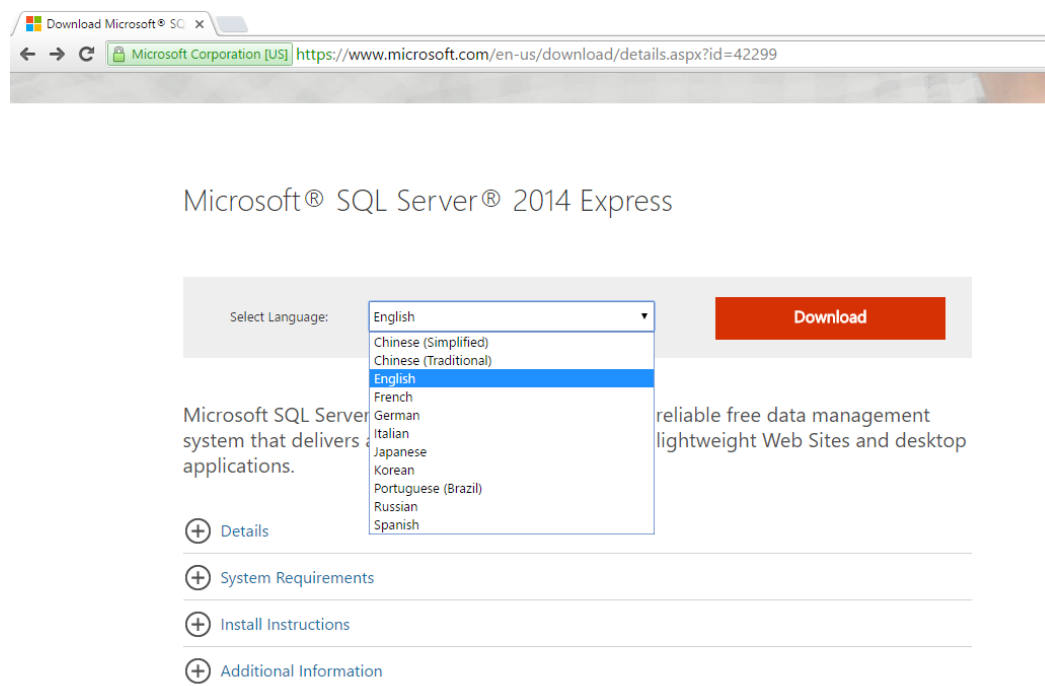
## **II. PRAKTICKÁ ČÁST**

### 3 PRÁCE S DATABÁZEMI V PROŠŘEDÍ SQL SERVER 2014

SQL Server 2014 v edici Express je zdarma ke stažení na webové adrese:

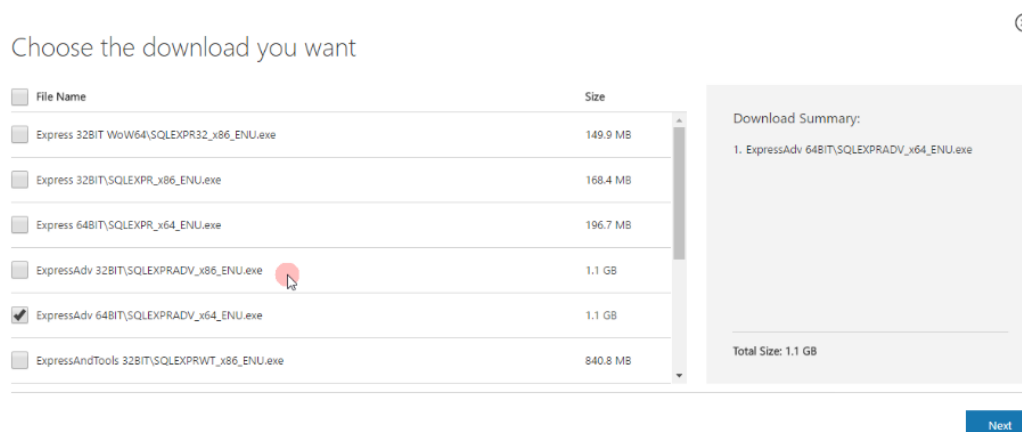
<https://www.microsoft.com/en-us/download/details.aspx?id=42299>

Po navštívení stránky je nutné vybrat jazyk:



Obrázek 6 – Výběr jazyka před stažením SQL Serveru

Po výběru jazyku klikneme na oranžové tlačítko download. Následující stránka obsahuje výběr verze, kterou chceme stáhnout. Pro naše výukové potřeby budeme potřebovat SQL Express with Services, to znamená verzi *ExpressAdv 32BIT\SQLEXPADV\_x86\_ENU* pro 32bitové systémy nebo *ExpressAdv 64BIT\SQLEXPADV\_x64\_ENU* pro 64bitové systémy.

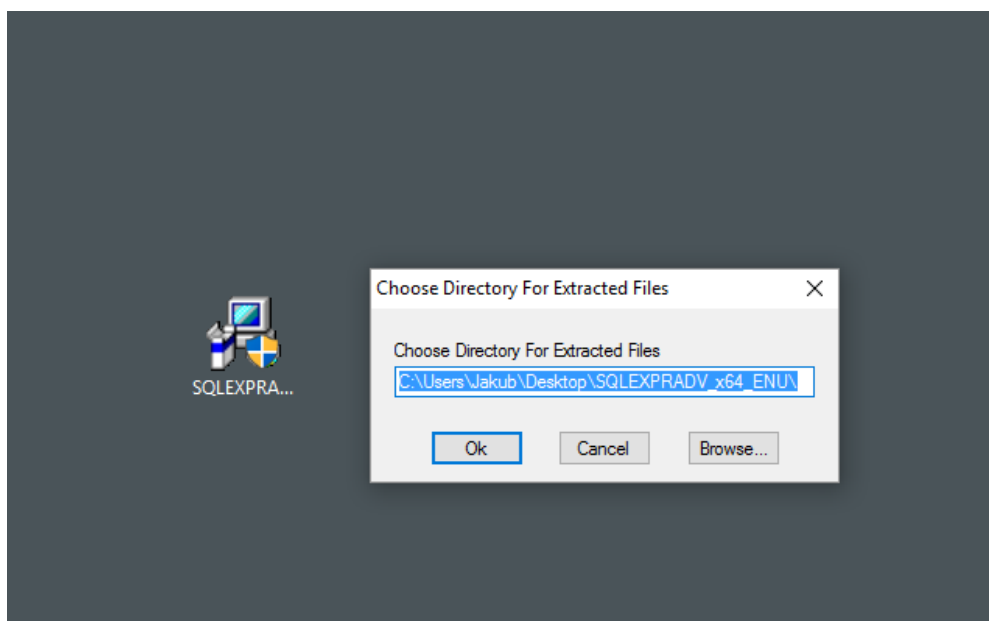


Obrázek 7 – Výběr verze ke stažení

Po kliknutí na NEXT se spustí stahování vybrané verze.

### 3.1 Instalace programu a popis prostředí

Pokud je potřebná verze stažená, je nutné jej extrahovat – dvojklikem na soubor a vybereme složku pro extrahování.



Obrázek 8 – Výběr složky pro extrahování instalačních souborů

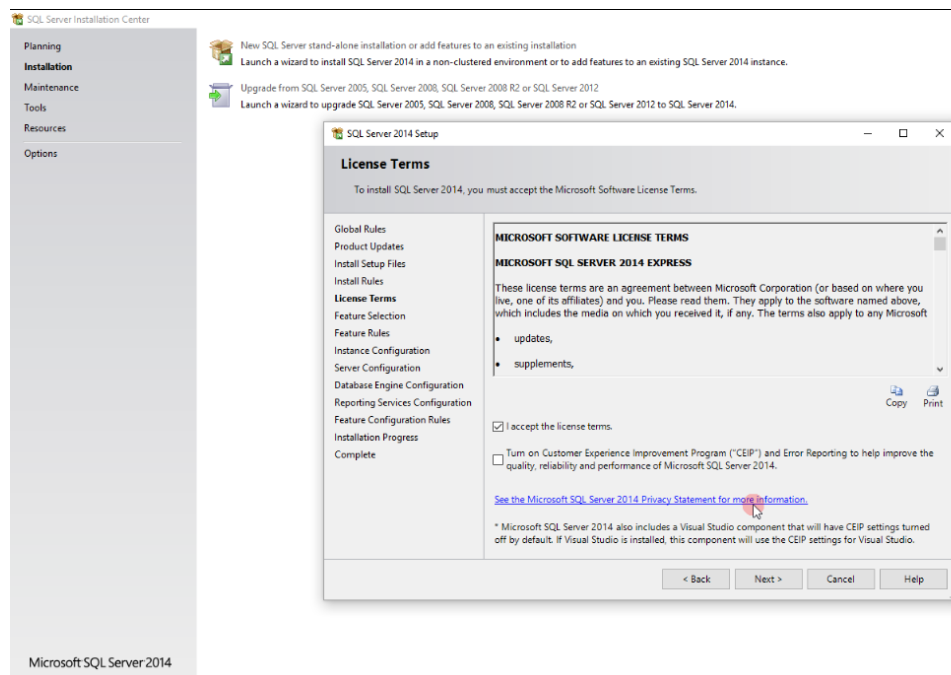
Po extrahování se automaticky spustí průvodce, pomocí kterého SQL Server nainstalujeme.



Obrázek 9 – Nabídka instalačního programu

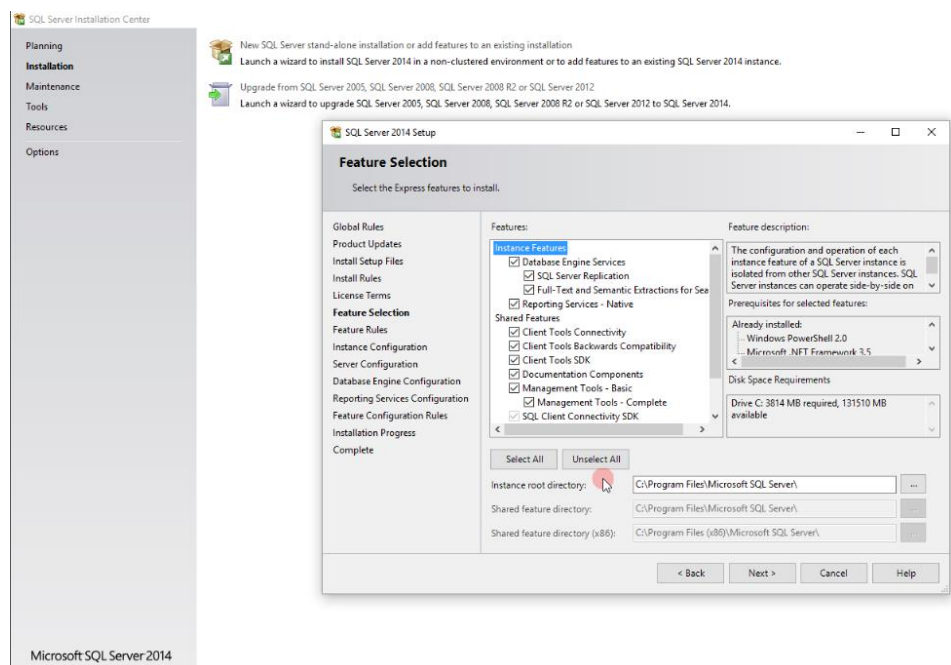
V nabídce instalačního programu zvolíme možnost: *New SQL Server stand-alone installation*.





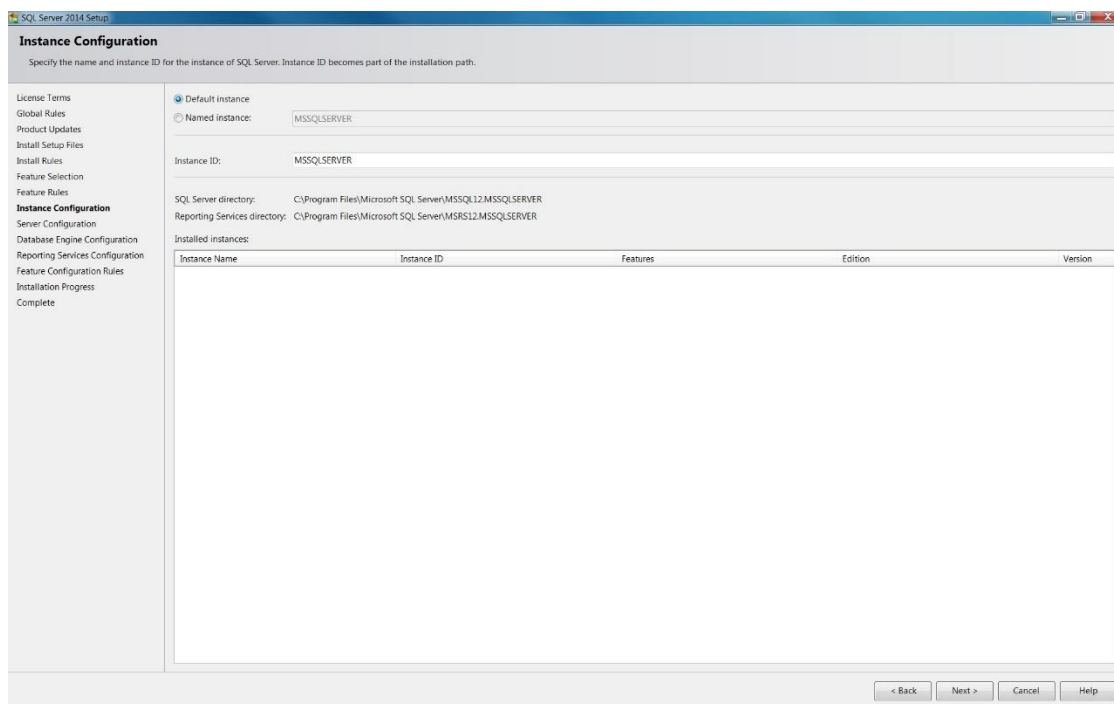
Obrázek 10 – Licenční podmínky

Po přečtení licenčních podmínek a souhlasu s nimi, klikneme na tlačítko next, kde nám instalační program nabídne výběr modulů, které chceme instalovat. Zaškrtneme možnost všechny – Select All a dáme next.



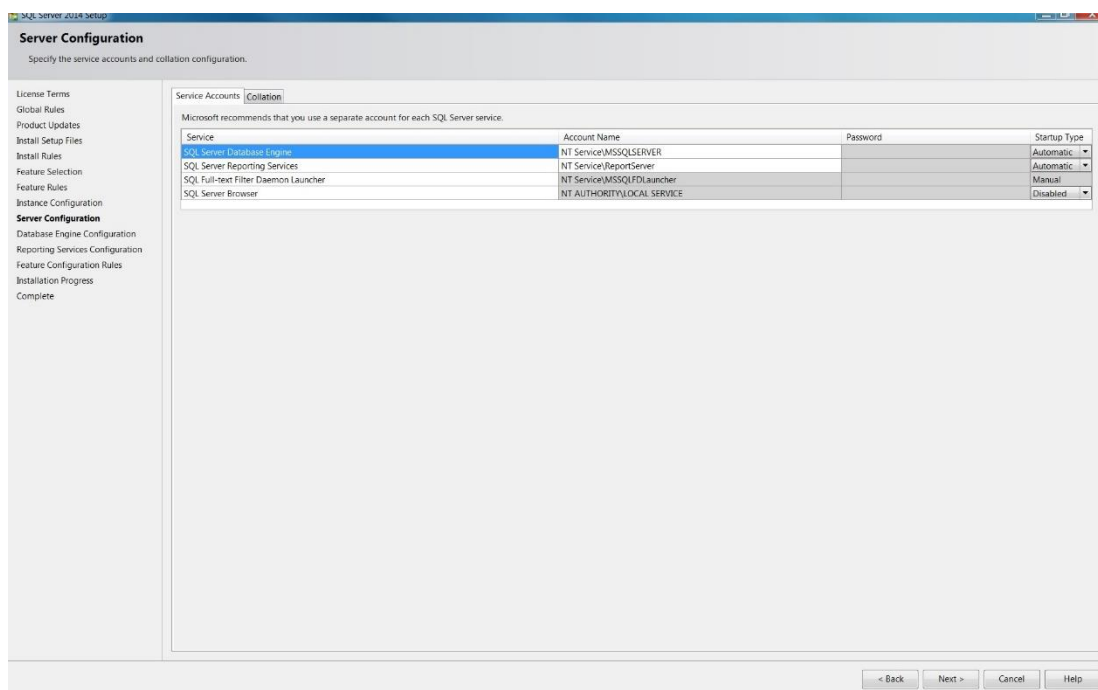
Obrázek 11 – Výběr modulů k instalaci

V dalším kroku instalace je možnost pojmenovat instanci SQL serveru. Doporučuji zaškrtnout *Default instance* a klikneme na next.



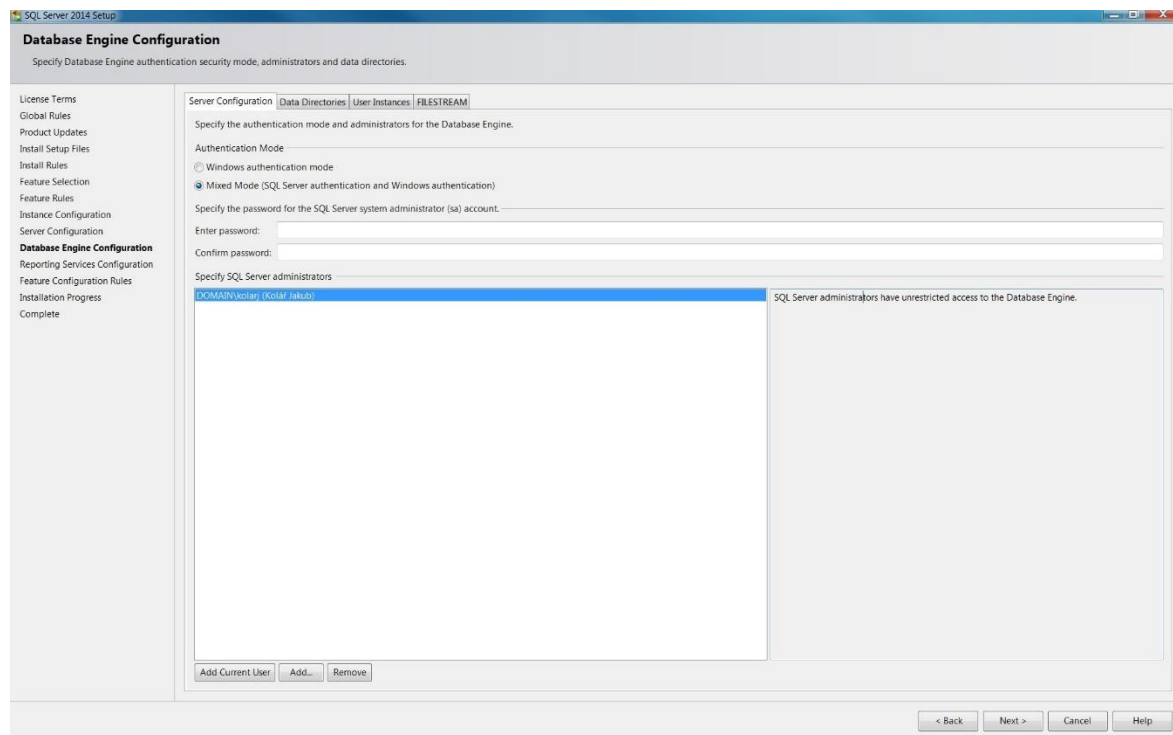
Obrázek 12 – Možnost pojmenování instance SQL Serveru

V následujícím kroku instalace je možnost volit automatické nebo manuální spouštění služeb, doporučuji ponechat přednastavené a opět kliknout na next.



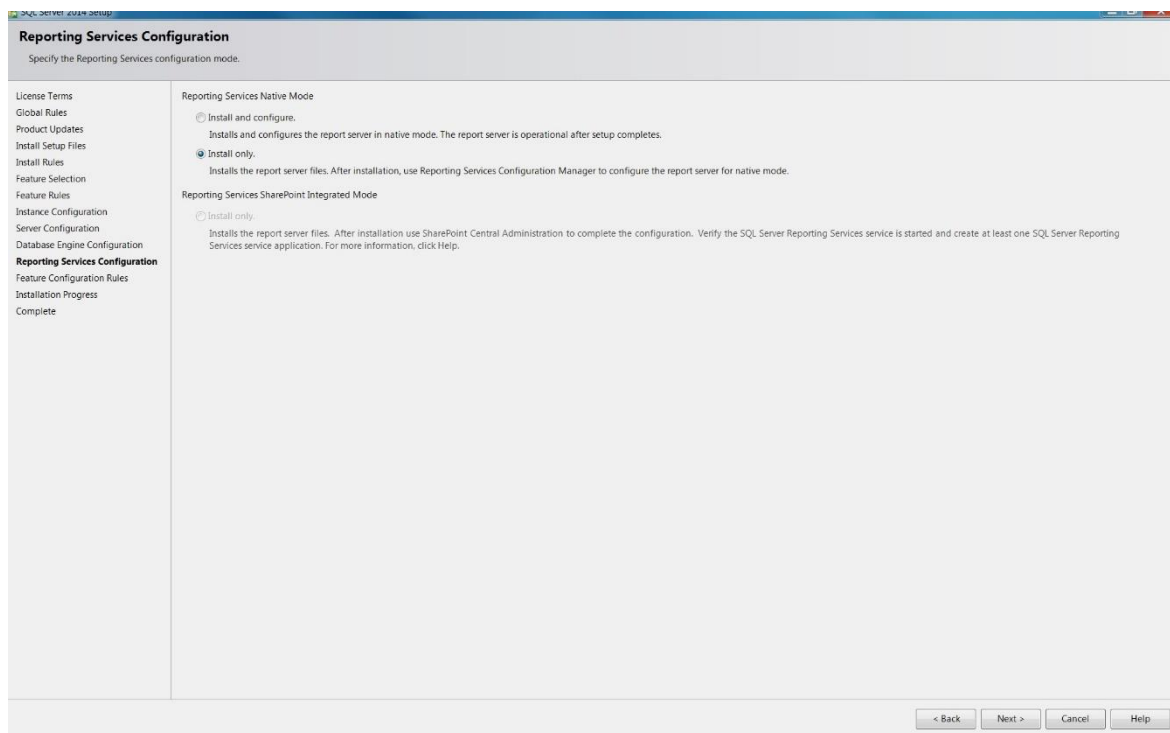
Obrázek 13 – Výběr typu spouštění

V dalším kroku konfiguruje Database Engine, zde je potřeba zaškrtnout Mixed Mode doplnit heslo, pod kterým se budeme do SQL serveru přihlašovat. Pokud není v kolonce *Specify SQL Server administrators* vidět náš uživatelský účet, přidáme jej pomocí kliku na tlačítko *Add Current User*, poté klikneme na next.



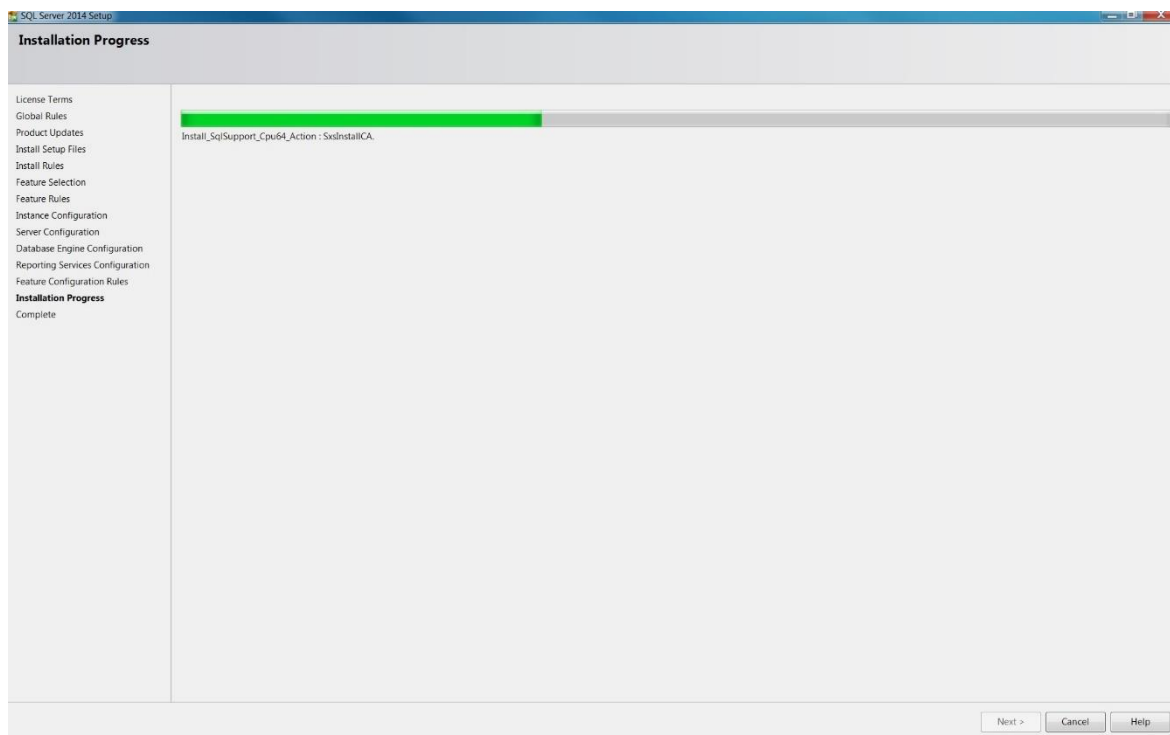
Obrázek 14 – Nastavení Database Engine

Při instalaci máme v další kroku možnost instalovat, nebo instalovat a konfigurovat *Reporting Services*. Dáme *install only*, pokud bychom je potřebovali, je možnost je v budoucnu nakonfigurovat.



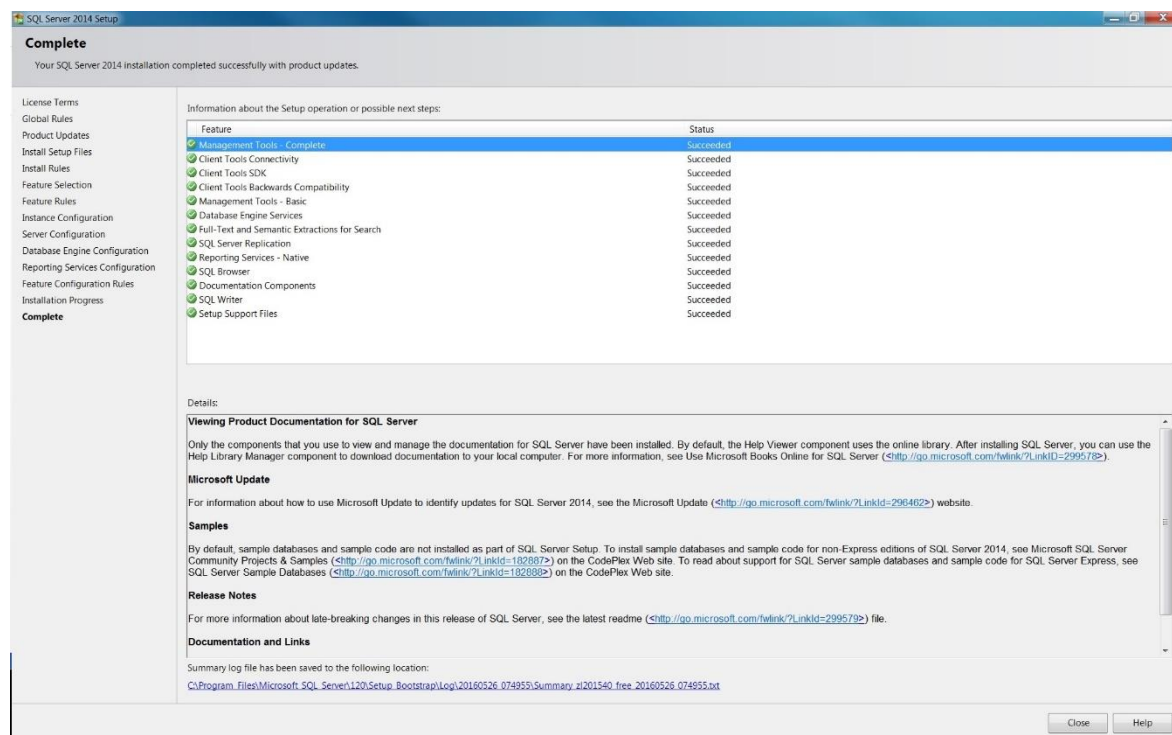
Obrázek 15 – Reporting Services

Po kliknutí na tlačítko next začne samotná instalace MS SQL Serveru.



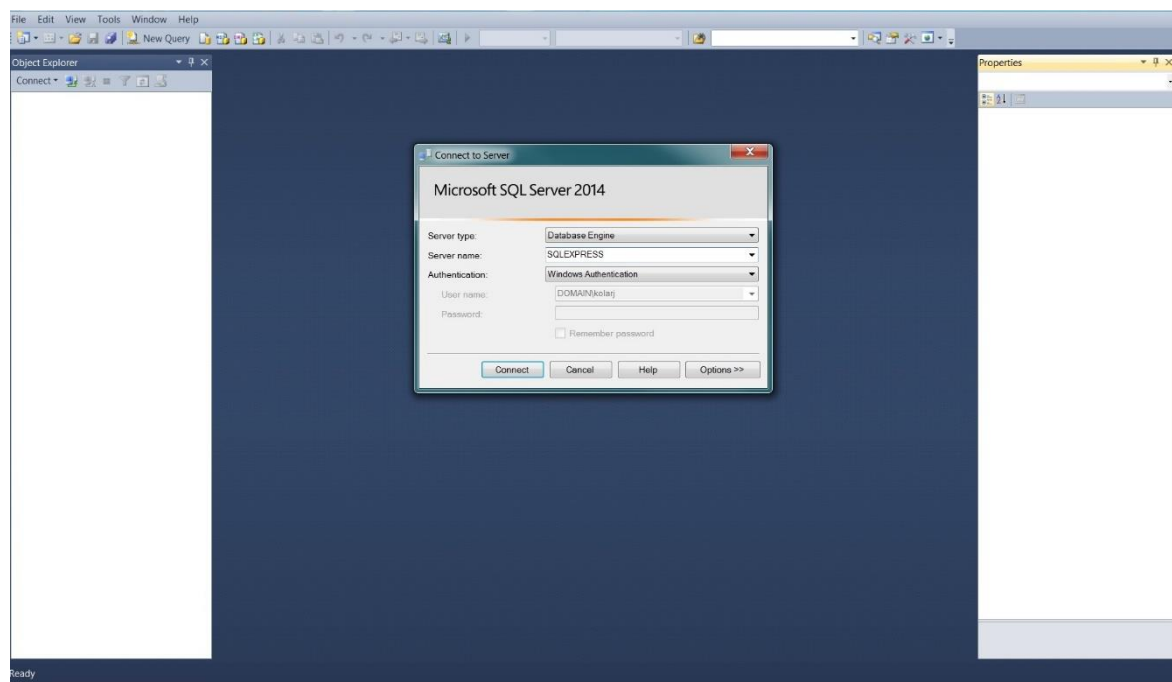
Obrázek 16 – Průběh instalace

Po dokončení se nám zobrazí report, co vše bylo úspěšně nainstalováno.



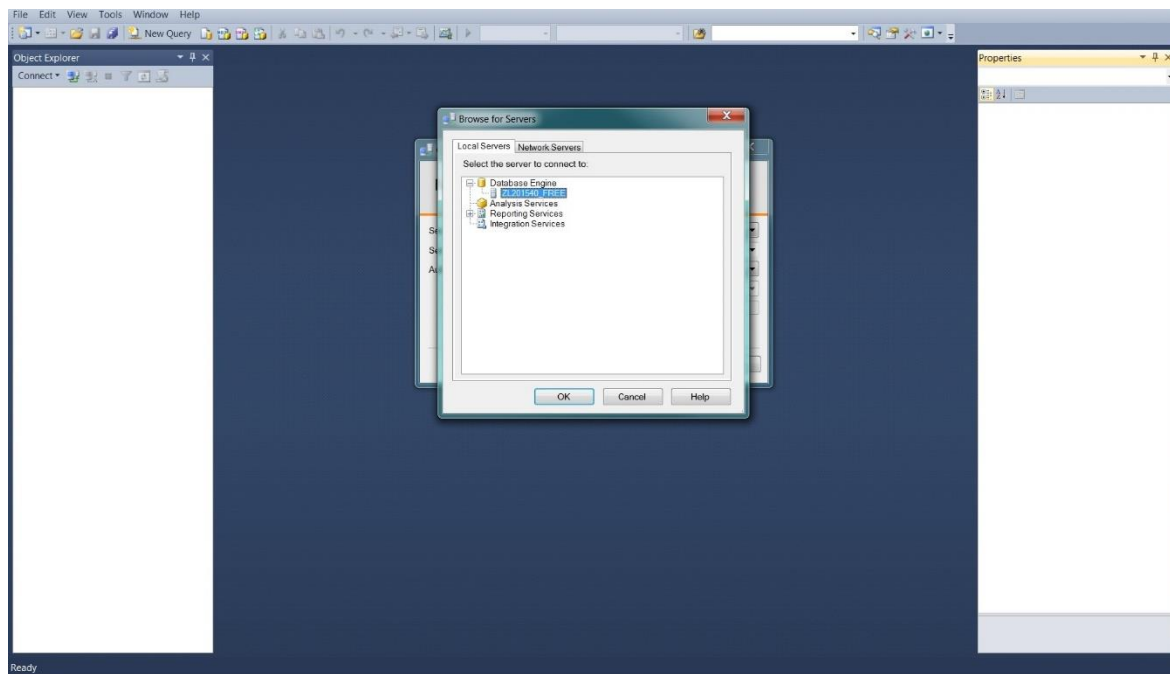
Obrázek 17 – Report z instalace

V tuto chvíli máme MS SQL Server nainstalovaný a můžeme jej začít používat. Příkazy budeme zapisovat v Management Studiu, najdeme jej v nainstalovaných programech počítače a spustíme.



Obrázek 18 – Po spuštění

Po spuštění nutno vybrat databázový engine, rozklikneme šipku u *Server Name* a dáme možnost *Browse for more*. V kartě Local Servers je položka Database Engine, rozklikneme a zde bude databázový engine, který je pojmenován podle uživatelského jména na počítači.



Obrázek 19 – Výběr databázového Engine

Nyní už se lze přihlásit do SQL Serveru a v záložce File -> New -> Database Engine Query začít psát naše příkazy pro práci s databázemi. Příkazy pak spustíme pomocí klávesy F5, nebo kliknutím na tlačítko Execute.

## 3.2 Tvorba databázových objektů

### 3.2.1 Vytvoření databáze

Pro vytvoření databáze využijeme příkaz CREATE DATABASE, který má hodně parametrů, ale pro základní vytvoření databáze stačí v následující syntaxi:

```
CREATE DATABASE nazevDatabaze
```

### 3.2.2 Smazání databáze

Příkaz DROP DATABASE smaže databázi, bez ohledu na její stav a zároveň odstraní fyzické soubory, které s danou databází souvisí. Obnovení databáze je možné pouze ze zálohy.

```
DROP DATABASE nazevDatabaze
```

### 3.2.3 Podporované operátory

#### Aritmetické operátory

Pomocí aritmetických operátorů můžeme provádět základní matematické operace dvou výrazů. Očekávány jsou číselné datové typy.

Tabulka 10 – Aritmetické operátory

Operátor	Popis
+	Sčítání
-	Odčítání
*	Násobení
/	Dělení
% (Modulo)	Vrátí zbytek po celočíselném dělení.

Sčítání, odčítání, násobení, dělení – všechny tyto operace mají typ návratové hodnoty takový, který má vyšší prioritu.

#### Operátory pro porovnávání

Slouží k testování, zda jsou dva výrazy stejné. Můžou být využity na všechny datové typy výrazů, kromě: text, nText, image. Tabulky podporovaných operátorů jsou v tabulce. Návratové hodnoty jsou typu Boolean.

Tabulka 11 – Operátory pro porovnávání

Operátor	Popis
=	Rovná se
>	Větší než
<	Menší než
>=	Větší rovno než
<=	Menší rovno než
<>	Není rovno
!=	Není rovno (není standardem ISO)
!<	Není menší než (není standardem ISO)
!>	Není větší než (není standardem ISO)

## Logické operátory

Logickými operátory můžeme testovat pravdivost nějaké podmínky. Stejně jako operátory pro porovnávání vrací hodnoty typu BOOLEAN a ty mohou být ve stavech: TRUE, FALSE nebo UNKNOWN. [40]

Tabulka 12 – Logické operátory [40]

Operátor	Popis
ALL	Vrátí TRUE, pokud všechny prvky z množiny jsou pravdivé.
AND	Vrátí TRUE, pokud oba logické výrazy jsou pravdivé.
ANY	Vrátí TRUE, pokud alespoň jeden prvek z množiny je pravdivý.
BETWEEN	V rozmezí.
EXISTS	Pokud existuje.
IN	Vrátí TRUE, pokud se operátor nachází v seznamu výrazů.
LIKE	Vrátí TRUE, pokud se výraz shoduje s maskou.
NOT	Obrátí hodnotu jakéhokoliv bool výrazu nebo operátoru
OR	Vrátí TRUE, pokud je některá z podmínek splněna.
SOME	Vrátí TRUE, pokud některé z množiny porovnávání jsou pravdivé.

## 3.3 Datové typy

### 3.3.1 Znakové datové typy

Pro ukládání znaků využíváme typy Char a Varchar.

Tabulka 13- znakové datové typy [20]

Datový typ	Popis	Paměťová náročnost
------------	-------	--------------------



Char(n)	n v rozmezí 1 – 8 000 znaků	n B
Varchar(n)	n v rozmezí 1 – 8 000 znaků	1 B na uložený znak + 2 B navíc

Základním rozdílem mezi Char a Varchar je v ukládání dat. Např. slovo AHOJ u typu VARCHAR(20) budou fyzicky využity 4 B + něco navíc. U typu CHAR(20), všech 20 B bude využito. Lepší je proto pro menší paměťovou náročnost využívat Varchar. [20]

Datový typ text zde není uveden, protože je považován za zastaralý, je doporučeno místo něj používat varchar(max). [20]

### 3.3.2 Celočíselné datové typy

Pro ukládání celých čísel máme k dispozici typy: Bit, Tinyint, Smallint, Int, Bigint, Numeric, Decimal, Money, Smallmoney.

Tabulka 14 – Číselné datové typy [20]

Datový typ	Popis	Paměťová náročnost
Bit	Hodnoty 0, 1 nebo null	1 B pro 8 hodnot tohoto typu
Tinyint	Celá čísla od 0 do 255	1 B
Smallint	Celá čísla od -32 768 do 32 767	2 B
Int	Celá čísla od -2 147 483 648 do 2 147 483 647	4 B
Bigint	Celá čísla od -9 223 372 036 854 775 808 do 9 223 372 036 854 775 807	8 B
Numeric (p, s) Decimal (p, s)	Čísla od $-10^{38} + 1$ do $10^{38} - 1$ při nejvyšší přesnosti (38)	Závisí na přesnosti: Při p = 1-9 velikost je 5 B Při p = 29-38 je velikost 17 B
Money	-922 337 203 685 477,5808 Až	8 B

	922 337 203 685 477,5807	
Smallmoney	-214 748,3648 až 214 748,3647	4 B

Prvním typem je bit, slouží pro ukládání hodnot pravda/nepravda. SQL Server navíc tabulku optimalizuje tak, že více polí tohoto typu ukládá do bajtů (tedy po osmi najednou). [20]

Základním typem je integer – int. Pro větší hodnoty lze použít bigint. Pro menší potom smallint a tinyint. [20]

Numeric (p, s) a Decimal (p, s) jsou funkčně totožné a slouží k uložení desetinných čísel se specifikovanou přesností (p, precision) a měřítkem (s, scale). [20]

Precision – udává nejvyšší počet číslic, které budou uloženy, a to jak vpravo, tak vlevo od desetinné čárky. Výchozí hodnota je 0, minimální taky 0, maximální p. Na volbě parametrů závisí výsledná velikost dat. Použití těchto typů má smysl pro data pocházející z výpočtů. [20]

Scale – udává nejvyšší možný počet číslic vpravo od desetinné čárky. [20]

Zvláštní variantou typu decimal jsou money a small money, určené pro ukládání hodnot měny. Odpovídají typu decimal s nastavenou přesností na čtyři desetinná místa. [20]

### 3.3.3 Plovoucí desetinná čárka

S plovoucí desetinnou čárkou pracují typy float a real.

Tabulka 15 - Datové typy s plovoucí desetinnou čárkou [1]

Datový typ	Popis	Paměťová náročnost
Float(n)	-1,79E + 308 až -2,23E – 308 0 a 2,23E – 308 až 1,79E + 308	Pro n ≤ 24 - 4 B Pro n > 24 - 8 B
Real()	-2,40E + 38 až -1,18E – 38 0 a 1,18E – 38 až 3,40E + 38	4 B

### 3.3.4 Datum a čas

S datem a časem pracují typy: Date, DateTime, DateTime2, DateTimeOffset, SmallDateTime a Time.

Tabulka 16 – Datum a čas [1]

Datový typ	Popis	Paměťová náročnost
Date	Ve formátu: YYYY-MM-DD 0001-01-01 až 9999-12-31	3 B
Datetime	Ve formátu: YYYY-MM-DD hh:mm:ss[.nnn] 1753-01-01 až 9999-12-31	8 B
Datetime2(n)	Ve formátu: YYYY-MM-DD hh:mm:ss[.nnnnnnnn] 0001-01-01 00:00:00.0000000 Až 9999-12-31 23:59:59.9999999	6-8 B
Datetimeoffset(n)	Ve formátu: YYYY-MM-DD hh:mm:ss[.nnnnnnnn][+ -] ]hh:mm 0001-01-01 00:00:00.0000000 Až 9999-12-31 23:59:59.9999999 (UTC)	8-10 B
SmallDateTime	Ve formátu: YYYY-MM-DD hh:mm:ss 1900-01-01 až 2079-06-06	4 B
Time(n)	Ve formátu: hh:mm:ss[.nnnnnnnn] 00:00:00.0000000 až 23:59:59.9999999	3-5 B

Time – ukládá se ve 24h formátu a představuje místní čas (nemá představu o časové zóně).

O paměťové náročnosti rozhoduje přesnost. [20]

DateTime – slouží pro uložení data a času ve 24h formátu, oproti funkci SmallDateTime má možnost přesnosti na 3,33 milivteřin. Hodnoty se zaokrouhlují na: 000, 003 nebo 007 vteřin. Příklad zaokrouhlení je uveden v tabulce 4. [19]

Tabulka 17 – DateTime [19]

Originální hodnota	Zaokrouhlená hodnota uložená do SQL
01-01-98 23:59:59.990	1998-01-01 23:59:59.990
01-01-98 23:59:59.991	
01-01-98 23:59:59.992	1998-01-01 23:59:59.993
01-01-98 23:59:59.993	
01-01-98 23:59:59.994	
01-01-98 23:59:59.995	1998-01-01 23:59:59.997
01-01-98 23:59:59.996	
01-01-98 23:59:59.997	
01-01-98 23:59:59.998	
01-01-98 23:59:59.999	1998-01-02 00:00:00.000

SmallDateTime – slouží k uložení data a času ve 24h formátu, kde hodnoty, mající na místě vteřin 29,998 a méně, budou zaokrouhleny na celé minuty směrem dolů. Hodnoty, mající na místě vteřin 29,999 a více budou zaokrouhleny na celé minuty směrem nahoru. [19]

DateTime2 – jde o rozšíření funkce DateTime, má větší rozsah a větší přesnost.

DateTimeOffset – ukládá datum a čas ve 24h formátu a umožňuje nastavit časové pásmo.

### 3.3.5 Binární data

Binární datové typy slouží k uložení dat, jako jsou grafické soubory, dokumenty Word nebo soubory MP3. Hodnoty jsou hexadecimální 0x0 nebo 0xF. [1]

Tabulka 18 - Binární datové typy [1]

Datový typ	Popis	Paměťová náročnost
Binary(n)	n v rozmezí 1 až 8 000	n B

VarBinary(n)	n v rozmezí 1 až 8000	1 B na každý znak + 2 B navíc
VarBinary(max)	1 až $2^{31} - 1$	1 B na každý znak + 2 B navíc

S binárními datovými typy se pojí také typ image. Stejně jako u znakových řetězců není doporučeno ho dále využívat, protože se s ním do budoucna nepočítá. [20] Preferovanou alternativou k typu image je VarBinary(max). [1]

### 3.3.6 Řetězce Unicode

K ukládání znaků ze sady Unicode slouží analogie ke znakovým datovým typům. [20]

Tabulka 19 – Datové typy řetězců unicode [1]

Datový typ	Popis	Paměťová náročnost
nChar	1 až 4 000 znaků	2n B
nVarChar	1 až 8 000 znaků	2m + 2 B
nVarChar(max)	1 až $2^{31} - 1$ B	2m + 2 B

### 3.3.7 Priorita datových typů

Když při použijeme operátor mezi dvěma výrazy s různými datovými typy, priorita datových typů určí, že datový typ s nižší prioritou se převede na datový typ s vyšší prioritou. Srovnání priorit datových typů vidíme v tabulce, priorita 1 je nejvyšší. [39]

Tabulka 20 – Priorita datových typů [39]

Priorita	Datový typ
1 – nejvyšší priorita	Uživatелеm definované datové typy
2	Sql_variant
3	Xml
4	dateTimeOffset
5	dateTime2
6	dateTime
7	smallDateTime
8	Date

9	Time
10	Float
11	Real
12	Decimal
13	money
14	smallMoney
15	bigInt
16	Int
17	smallInt
18	tinyInt
19	Bit
20	nText
21	Text
22	image
23	timeStamp
24	Uniqueidentifier
25	nVarChar
26	nChar
27	varChar
28	Char
29	varBnary
30 – nejnižší priorita	Binary

### 3.3.8 Vytvoření tabulky

K vytvoření tabulky slouží příkaz CREATE TABLE. Příkaz má následující syntaxi: [3]

```
CREATE TABLE nazevTabulky (  
    nazevSloupce datovyTyp volitelneOmezeni,  
    nazevSloupce datovyTyp volitelneOmezeni,  
    volitelnaOmezeniTabulky  
);
```

volitelnéOmezení = Omezení sloupce a může být: PRIMARY KEY, NOT NULL, a UNIQUE.

PRIMARY KEY – tento sloupec bude sloužit k jednoznačné identifikaci řádku v rámci databáze.

NOT NULL – v tomto sloupci nesmí být nulová hodnota (nesmí být prázdný).

NULL – tento sloupec může být prázdný

UNIQUE – toto omezení říká, že v rámci sloupce nesmí být duplicitní hodnoty.

```
INSERT INTO jmenoTabulky  
VALUES (data)
```

Dále na místo volitelného omezení při tvorbě tabulky můžeme přidat parametr DEFAULT. Nastavení DEFAULT určuje, že pokud není při vytvoření nového řádku uvedena žádná hodnota, databázový systém tam vloží hodnotu, která je uvedená u tohoto klíčového slova. [3]

### 3.3.9 Smazání tabulky

Pro smazání tabulky existuje jednoduchý příkaz:

```
DROP TABLE nazevTabulky
```

Kde za nazevTabulky dáváme název tabulky, která má být odstraněna.

Příkaz DROP TABLE nefunguje, pokud tabulka obsahuje hodnoty, které jsou potřebné pro omezení referenční integrity. [3]

## 3.4 Práce s daty

### 3.4.1 Vložení dat do tabulky

Data lze do relace přidávat pomocí příkazu INSERT INTO. Tento příkaz může mít dvě podoby podle toho, zda zadáváme data do všech sloupců, či pouze do některých. [3]

Základní formát příkazu INSERT vypadá takto:

V tomto případě systém řízení databáze předpokládá, že budeme vkládat do všech sloupců v tabulce, přičemž hodnoty musí být ve stejném pořadí a stejného datového typu, jako byly zadány při vytváření tabulky příkazem INSERT.

Pro vkládání jen do některých sloupců musíme příkaz INSERT modifikovat:

```
INSERT INTO jmenoTabulky [seznamSloupcu]
VALUES (Data)
```

[seznamSloupcu] – představuje hodnoty, které budeme do jednotlivých sloupců vkládat, oddělené čárkami.

### 3.4.2 Smazání dat

DELETE odstraní řádky z tabulky. Formát příkazu je následující:

```
DELETE FROM nazevTabulky
```

Takto uvedený příkaz smaže všechna data z tabulky. Pokud chceme mazat jen některá data, můžeme příkaz modifikovat:

```
DELETE nazevSloupce FROM nazevTabulky
WHERE vyhledavaciPodminka
```

Takto upravený příkaz DELETE odstraní jen uvedený sloupec z tabulky, který navíc splňuje uvedenou podmínku. [6]

### 3.4.3 Příkaz ALTER TABLE

Příkaz ALTER TABLE se používá při přidávání nebo odebírání sloupců v tabulce.

Syntaxe při přidávání sloupce:

```
ALTER TABLE nazevTabulky
ADD nazevSloupce datovýTyp
```

Syntaxe při odebírání sloupce:

```
ALTER TABLE nazevTabulky
DROP COLUMN nazevSloupce
```



### 3.4.4 Příkaz UPDATE

Příkaz UPDATE modifikuje / aktualizuje existující data v databázi. Formát příkazu je následující:

```
UPDATE nazezTabulky  
SET jmenoSloupce = datovaHodnota
```

Klauzole SET udává jméno sloupce, který chceme aktualizovat. Rozšíření příkazu pak vypadá takto:

```
UPDATE nazezTabulky  
SET jmenoSloupce1 = datovaHodnota1,  
    jmenoSloupce2 = datovaHodnota2  
WHERE vyhledavaciPodminka
```

WHERE je v tomto případě volitelná. Pokud není uvedena, vyjmenované sloupce, jsou aktualizovány pro všechny řádky tabulky. Pokud je WHERE uvedena, jsou aktualizovány jen řádky, které splňují vyhledávací podmínku. [6]

## 3.5 Výběrové dotazy

Pro jednoduché získání dat z databáze je třeba použít 3 příkazy: SELECT, FROM, WHERE [4]

### 3.5.1 SELECT

Účelem příkazu SELECT je vyvolat a zobrazit data z jedné nebo více tabulek. Je to nejpoužívanější příkaz jazyka SQL. [6]

### 3.5.2 FROM

Klauzuli FROM využíváme ve spojení s předešlou – SELECT a jejím úkolem je identifikovat data pro dotaz. [4] Daty pro dotaz se rozumí tabulku nebo tabulky, které se mají použít.

### 3.5.3 WHERE

Pomocí WHERE určíme kritéria dotazu – ve výsledku dat bude sada požadovaných dat vrácena. [4] Klíčové slovo WHERE je následováno vyhledávací podmínkou. Rozlišujeme pět základních vyhledávacích podmínek: [6]

- Porovnávání (porovnávání hodnoty jednoho výrazů s hodnotou jiného výrazu).
- Rozsah (testuje, zda hodnota výrazu spadá do zadaného rozsahu hodnot).

- Náleží do množiny (testuje, zda hodnota výrazu se rovná některé hodnotě z množiny).
- Srovnání vzoru (testuje, zda řetězec odpovídá určitému vzoru).
- NULL (testuje, zda sloupec má hodnotu NULL – prázdnou hodnotu).

### 3.5.4 Konstrukce výběrového dotazu

Pořadí jednotlivých klauzulí v příkazu SELECT není možné měnit. Povinné jsou však jen první dvě klauzule: SELECT a FROM, ostatní jsou nepovinné. Každý příkaz SELECT vytváří tabulku složenou z jednoho nebo více sloupců a nula nebo více řádků.

Jednoduchý dotaz můžeme vytvořit takto:

SELECT vyraz FROM nazevTabulky

Vyraz – zde může být operátor, název sloupce nebo sloupců, které chceme v dotazu vrátit.

NazevTabulky – název tabulky, na kterou chceme dotaz použít.

Pro filtrování dat po vykonání dotazu musíme do posloupnosti přidat klauzuli WHERE:

SELECT vyraz FROM nazevTabulky WHERE omezeni

Omezení – zde můžeme použít operátor, nebo další klíčová slova, abychom po vykonání dotazu měli jen taková data, která potřebujeme.

V omezení můžeme využívat logické operátory, operátory pro porovnávání a další, které uvedu v následujících kapitolách.

### 3.5.5 Filtrace znakových dat

Vyhledávací podmínka LIKE / NOT LIKE slouží k filtrování znaků na základně nějakého vzoru, např. počáteční písmeno nebo část slova. Pro zadání toho, co chceme vyhledávat, slouží znaky, které jsou uvedeny v následující tabulce.

Tabulka 21 – Filtrace znakových dat [2]

Zápis	Význam	Příklad
% (procento)	Libovolný znak, včetně mezery	,D%' pro řetězec začínající písmenem D
_ (podtržítko)	Pro jeden znak	,_D%' řetězec, kde druhé písmeno je D

[<seznam znaků>]	Jeden znak z řetězce	,[AC]%‘ řetězec, kde první písmenko je A nebo C
[<rozsah znaků>]	Jeden znak z rozsahu	,[0-9]‘ řetězec, kde první znak je číslice
[^<rozsah nebo seznam znaků>]	Jeden znak, který není v řetězci nebo v rozsahu	,[^0-9]%‘ řetězec, kde první znak není číslo

Jednoduchý dotaz pro vyhledání jmen, která začínají písmenem A:

```
SELECT jmeno
FROM tabulka
WHERE jmeno LIKE ,A%‘
```

### 3.5.6 Řazení dat pomocí příkazu ORDER BY

Přidáním klauzule ORDER BY dosáhneme toho, že data po vykonání dotazu budou seřazena. [2] Řadit můžeme vzestupně (ASC) nebo sestupně (DESC) a to podle libovolného sloupce, bez ohledu na to, zda se dané sloupce ve výsledcích nacházejí či nikoliv. ORDER BY bývá umístěn na posledním místě v příkazu SELECT. [6]

```
SELECT * FROM nazevTabulky
ORDER BY sloupec
```

Příkaz v defaultním nastavení řadí vzestupně (0-999). Pokud chceme řadit sestupně, stačí připsat DESC.

```
SELECT * FROM nazev Tabulky
ORDER BY sloupec DESC
```

### 3.5.7 Seskupování výsledků pomocí GROUP BY

Užitím příkazu GROUP BY v SELECT dostaneme seskupený dotaz, protože seskupí data z tabulky (nebo i více tabulek) tak, že pro každou skupinu vytvoří jednořádkový souhrn. Sloupce uvedené v klauzuli GROUP BY označujeme jako seskupovací sloupce. [6]

### 3.5.8 Omezení seskupování pomocí HAVING

Příkaz HAVING je určen k použití s klauzulí GROUP BY pro omezení skupin, které se mají objevit ve výsledné tabulce. I přes podobnou syntaxi, slouží HAVING a WHERE k odlišným účelům. WHERE filtruje jednotlivé řádky vstupující do tabulky konečných výsledků, zatímco HAVING filtruje skupiny vstupující do tabulky konečných výsledků. [6]

Jinými slovy, rozdíl mezi HAVING a WHERE je, že WHERE aplikujeme na řádky před seskupením, HAVING aplikujeme na již seskupená záznamy. [43]

### 3.5.9 Filtrování dat pomocí TOP

TOP využijeme pro vrácení určitého počtu nebo procent z výsledku dotazu. [2] Pro určení pořadí lze dále využít ORDER BY. Příklad využití TOP v příkazu SELECT, kdy budeme chtít vrátit první 3 řádky:

```
SELECT TOP (3) sloupce  
FROM nazevTabulky  
ORDER BY nazevSloupce
```

Kdy za parametr *sloupce* dáme název sloupců, které chceme vrátit, za *nazevTabulky* dáme tabulku, na kterou chceme dotaz aplikovat a za klauzuli ORDER BY napíšeme název sloupce, podle kterého chceme výsledek dotazu řadit.

Pokud nechceme filtrovat podle počtu řádků, ale podle procent, přidáme za TOP ( ) klíčové slovo PERCENT přičemž v závorce je očekáváno kolik procent chceme filtrovat.

```
SELECT TOP (3) PERCENT sloupce  
FROM nazevTabulky  
ORDER BY nazevSloupce
```

Takto uvedený příkaz vyfiltruje první 3 %.

### 3.5.10 Filtrování pomocí OFFSET – FETCH

Slouží k filtraci, přičemž offset udává, kolik řádků chceme přeskočit a fetch kolik řádků chceme vypsát. [2]

Předpokládejme tabulku 20 řádků. Následující použití OFFSET – FETCH bude pracovat tak,

že přeskočí prvních 10 řádků a vypíše dalších 10 – řádky od čísla 11 do čísla 20.

```
SELECT *  
FROM nazevTabulky  
ORDER BY nazevSloupce  
OFFSET 10 ROWS  
FETCH NEXT 10 ROWS ONLY
```

### 3.5.11 Používání Aliasů

Alias využijeme, když například při výpisu dvou sloupců – jméno a příjmení chceme, aby tyto dva sloupce byly po vykonání dotazu vráceny jako jeden, který pojmenujme celeJmeno.

[2] V příkazu SELECT je přidání aliasu jednoduché – za požadované sloupce přidáme klauzuli AS, za kterou uvedeme výsledné jméno sloupce, v našem případě celeJmeno. Syntaxe bude tedy následující:

```
SELECT jmeno + ' ' + prijmeni AS celeJmeno FROM zamestnanci
```

## 3.6 T – SQL funkce v dotazech

Ve výběrových dotazech lze využívat vestavěné funkce SQL Serveru, které rozdělujeme do několika kategorií: [4]

- Agregáční funkce
- Konfigurační funkce
- Kurzorové funkce
- Funkce pro datum a čas
- Matematické funkce
- Funkce pro metadata
- Řádkové funkce
- Bezpečnostní funkce
- Funkce pro řetězce
- Systémové statistické funkce
- Ostatní funkce

### 3.6.1 Agregáční funkce

Agregáční funkce zpracují více hodnot v rámci dotazu a vrátí jednu souhrnnou hodnotu. [4]

S výjimkou COUNT, agregáční funkce ignorují hodnoty NULL. Tyto funkce jsou nejčastěji používané s příkazy GROUP BY nebo SELECT. Všechny funkce jsou deterministické – funkce vrátí stejnou hodnotu kdykoliv, když jsou zavolány pro určitou sadu dat. [22] Agregáčních funkcí v SQL 2014 je celkem 13.

**AVG**

Vrátí průměr ze zadané množiny hodnot. [23]

Syntaxe:

*AVG([ALL | DISTINCT] Výraz )*

Argumenty:

**ALL** – Aplikuje agregační funkci na všechny zadané hodnoty. Tento argument je nastaven jako defaultní. [23]

**DISTINCT** – agregační funkce bude aplikována pouze na jednu instanci určité hodnoty, bez ohledu na její opakování. [23]

**Výraz** – očekávaný číselný datový typ. Nelze do výrazu přidávat další agregační funkce nebo poddotazy. [23]

Datový typ návratové hodnoty:

Tabulka 22 - Datové typy návratových hodnot AVG [23]

Datový typ výrazu	Datový typ návratové hodnoty
tinyInt	int
smallInt	int
Int	int
bigInt	bigInt
decimal (p, s)	decimal (38, s)
money, smallMoney	money
float, real	float

**MIN a MAX**

MIN vrátí minimum a MAX maximum ze zadané množiny hodnot. Datový typ návratové hodnoty je stejný jako u dat, který do funkce vkládáme. [24] [25]

Syntaxe:

*MIN([ALL | DISTINCT] Výraz )*

*MAX([ALL | DISTINCT] Výraz )*

Argumenty:

ALL – Aplikuje agregační funkci na všechny zadané hodnoty. Tento argument je nastaven jako defaultní. [24] [25]

DISTINCT – Vynechávání hodnot pro počítání minima zde není smysluplné, ale DISTINCT je k dispozici pouze pro ISO kompatibilitu. [24] [25]

Výraz – může být konstanta, název sloupce, OR a jakákoliv kombinace aritmetických, bitových a řetězcových operátorů. MIN může být použit pro: čísla, char, varChar, iniqueIdentifier nebo sloupce obsahující dateTime. Nelze použít s bitovými datovými formáty. Nelze do výrazu přidávat další agregační funkce nebo poddotazy. [24] [25]

### SUM

Vrátí součet (sumu) všech nebo DISTINCT hodnot - záleží na syntaxi dorazu. SUM může být použita pouze se sloupci, které obsahují číselné hodnoty. Hodnoty NULL jsou ignorovány. [26]

Syntaxe:

*SUM([ALL | DISTINCT] Výraz )*

Argumenty:

ALL – Aplikuje agregační funkci na všechny zadané hodnoty. Tento argument je nastaven jako defaultní. [26]

DISTINCT – použitím tohoto argumentu budou sečteny pouze unikátní hodnoty (bez duplicit). [26]

Výraz - může být konstanta, název sloupce, OR a jakákoliv kombinace aritmetických, bitových a řetězcových operátorů. Nelze použít s bitovými datovými formáty. Nelze do výrazu přidávat další agregační funkce nebo poddotazy. [26]

Datový typ návratové hodnoty:

Tabulka 23 – Datové typy návratových hodnot SUM [26]

Datový typ výrazu	Datový typ návratové hodnoty
tinyInt	int
smallInt	int
Int	int
bigInt	bigInt

decimal (p, s)	decimal (38, s)
money, smallMoney	money
float, real	float

### CHECKSUM\_AGG

Vrací kontrolní součet hodnot ze zadaných dat. Hodnoty NULL jsou ignorovány. Datový typ návratových hodnot je u všech výrazů int. [27]

Syntaxe:

*CHECKSUM\_AGG([ALL | DISTINCT] Výraz )*

Argumenty:

ALL – Aplikuje agregační funkci na všechny zadané hodnoty. Tento argument je nastaven jako defaultní. [27]

DISTINCT – použitím tohoto argumentu bude vrácen kontrolní součet pouze unikátních hodnot (bez duplicit). [27]

Výraz – je očekáváno celé číslo. Nelze do výrazu přidávat další agregační funkce nebo poddotazy. [27]

### COUNT a COUNT\_BIG

Count vrací počet položek, datový typ návratové hodnoty je int. COUNT\_BIG je funkčně stejný, s jediným rozdílem, má návratovou hodnotu typu bigint. [28]

Syntaxe:

*COUNT({ [ [ALL | DISTINCT] Výraz ] | \* })*

*COUNT\_BIG({ [ [ALL | DISTINCT] Výraz ] | \* })*

Argumenty:

ALL – Aplikuje agregační funkci na všechny zadané hodnoty. Tento argument je nastaven jako defaultní. [28]

DISTINCT – použitím tohoto argumentu bude vrácen kontrolní součet pouze unikátních hodnot (bez duplicit). [28]



Výraz – u COUNT jsou povoleny všechny typy kromě text, image nebo nText. U COUNT\_BIG může být výraz jakéhokoliv typu. Nelze do výrazu přidávat další agregační funkce nebo poddotazy. [28] [29]

\* - hvězdičkou značíme, že chceme do výpočtu zahrnout všechny řádky určité tabulky. Při použití *COUNT(\*)* nelze použít argument DISTINCT. Výpočet zahrnuje duplikáty i hodnoty NULL. [28] [29]

### STDEV

Vrací statistickou standardní odchylku všech hodnot zahrnutých v zadaném výrazu. Datový typ návratové hodnoty je float. [30]

Syntaxe:

$$STDEV([ ALL |DISTINCT ]Výraz)$$

Argumenty:

ALL – Aplikuje agregační funkci na všechny zadané hodnoty. Tento argument je nastaven jako defaultní. [30]

DISTINCT – použitím tohoto argumentu budou zpracovány pouze unikátní hodnoty (bez duplicit). [30]

Výraz – je očekáváno číslo. Nelze do výrazu přidávat další agregační funkce nebo poddotazy. [30]

### STDEVP

Návratová hodnota je datového typu float. STDEVP vrací standardní statistickou odchylku pro populaci pro všechny hodnoty zahrnuté v zadaném výrazu. [31]

Syntaxe:

$$STDEVP([ ALL |DISTINCT ]Výraz)$$

Argumenty:

ALL – Aplikuje agregační funkci na všechny zadané hodnoty. Tento argument je nastaven jako defaultní. [31]

DISTINCT – použitím tohoto argumentu budou zpracovány pouze unikátní hodnoty (bez duplicit). [31]

Výraz – je očekáváno číslo. Nelze do výrazu přidávat další agregační funkce nebo pod-dotazy. [31]

## **GROUPING**

Grouping (seskupení) určuje, zda je zadaný sloupec ve výrazu v GROUP BY listu agregován či nikoliv. GROUPING vrací 1 pro agregované, nebo 0 pro neagregované. Typ návratové hodnoty je tinyInt. [32]

Syntaxe:

*GROUPING (< Sloupec >)*

Argumenty:

Sloupec – je sloupec nebo výraz, který obsahuje sloupce v GROUP BY. [32]

## **GROUPING\_ID**

Pomocí této agregační funkce můžeme počítat úroveň seskupení. Typem návratové hodnoty je int. [33]

Syntaxe:

*GROUPING\_ID (< Sloupec > [, ... n])*

Argumenty:

Sloupec – je sloupec nebo výraz, který obsahuje sloupce v GROUP BY. [33]

## **VAR**

Vrací statistický rozptyl všech hodnot obsažených ve výrazu. Návratovým typem je float. [34]

Syntaxe:

*VAR ( [ALL | DISTINCT] Výraz )*

Argumenty:

ALL – Aplikuje agregační funkci na všechny zadané hodnoty. Tento argument je nastaven jako defaultní. [34]

DISTINCT – použitím tohoto argumentu budou zpracovány pouze unikátní hodnoty (bez duplicit). [34]

Výraz – je očekáváno číslo. Nelze do výrazu přidávat další agregační funkce nebo pod-dotazy. [34]

## VARP

Vrátí statistický rozptyl pro populaci pro všechny hodnoty, které jsou zadane ve výrazu, návratová hodnota je typu FLOAT. [35]

Syntaxe:

VARP ( [ALL | DISTINCT] Výraz )

Argumenty:

ALL – Aplikuje agregační funkci na všechny zadane hodnoty. Tento argument je nastaven jako defaultní. [35]

DISTINCT – použitím tohoto argumentu budou zpracovány pouze unikátní hodnoty (bez duplicit). [35]

Výraz – je očekáváno číslo. Nelze do výrazu přidávat další agregační funkce nebo pod-dotazy. [35]

### 3.6.2 Konfigurační funkce

Konfigurační funkce jsou skalární a vracejí informace o aktuálním nastavení konfigurace.

[4] Konfigurační funkce jsou například:

- @@DATEFIRST
- @@OPTIONS
- @@DBTS
- @@REMSERVER
- @@LANGID
- @@SERVERNAME
- @@LANGUAGE
- @@SERVICENAME
- @@LOCK\_TIMEOUT
- @@SPID
- @@MAX\_CONNECTIONS
- @@TEXTSIZE
- @@VERSION
- @@MAX\_PRECISION
- @@NESTLEVEL

### 3.6.3 Kurzorové funkce

Informace o kurzorech, máme k dispozici funkce: [4] například:

- @@CURSOR\_ROWS
- @@CURSOR\_STATUS
- @@FETCH\_STATUS

### 3.6.4 Matematické funkce

Pro matematické operace můžeme využít vestavěné matematické funkce, které budou pracovat na základě vstupních hodnot, které jsou zadané jako parametry těchto funkcí. [4]

K dispozici jsou funkce:

- ABS (absolutní hodnota čísla)
- DEGREES (pro úhel zadaný v radiánech vrátí úhel ve stupních)
- RAND (generuje pseudo náhodné číslo)
- ACOS (vrací úhel v radiánech ze zadané hodnoty cosinus)
- EXP (vrací exponenciální hodnotu zadaného čísla)
- ROUND (zaokrouhlí číslo na požadovanou přesnost)
- ASIN (vrací úhel v radiánech ze zadané hodnoty sinus)
- FLOOR (zaokrouhlí na celé číslo směrem dolů)
- SIGN (vrací znaménko zadaného výrazu)
- ATAN (vrací úhel v radiánech ze zadané hodnoty tangens)
- LOG (pro přirozený logaritmus čísla)
- SIN (vrací sinus úhlu zadaného v radiánech)
- ATN2 (do funkce vstupují parametry x, y a funkce vrátí úhel, který bod svírá s osou x)
- LOG10 (dekadický logaritmus čísla)
- SQRT (vrací druhou odmocninu zadaného čísla)
- CEILING (zaokrouhlí na celé číslo směrem nahoru)
- PI (vrací konstantu Ludolfovo číslo)
- SQUARE (vrací druhou mocninu zadaného čísla)
- COS (vrací cosinus úhlu zadaného v radiánech)
- POWER (do funkce vstupují dva parametry: číslo a stupeň, na kolikátou chceme číslo umocnit)
- TAN (vrací tangens zadaného úhlu v radiánech)
- COT (vrací cotangens zadaného úhlu v radiánech)
- RADIANS (pro úhel zadaný ve stupních vrátí úhel v radiánech)

### 3.6.5 Funkce pro metadata

Funkce pro metadata umí vrátit informace o databázích, souborech a dalších datech s nimi spojených. [4]

### 3.6.6 Řádkové funkce

Vrací objekt, který může být použit v místě tabulky nebo pohledu v příkazu T-SQL. [4] Řádkové funkce jsou například:

- OPENDATASOURCE
- OPENROWSET
- OPENQUERY
- OPENXML

### 3.6.7 Bezpečnostní funkce

Funkce pro zabezpečení slouží k vrácení informace o uživateli a jejich rolích. [4]

### 3.6.8 Funkce pro řetězce

Slouží k manipulaci s řetězci. Všechny řetězce v těchto funkcích musí být uzavřeny v jednoduchých uvozovkách. [4] například:

- ASCII (vrátí ascii kód zadaného řetězce)
- LTRIM (vyjme mezery před řetězcem)
- SOUNDEX (vrátí kód 4 znaků, sloužící pro porovnání podobnosti řetězců)
- CHAR (převéde datový typ INT na znakový)
- NCHAR (vrací znaky unicode)
- SPACE (funkce vrací řetězec mezer, jejich počet zadáváme při volání funkce)
- STR (vrátí znaková data převedená z číselných)
- CONCAT (složí více řetězců do jednoho)
- STUFF (pro vložení řetězce do jiného řetězce)
- DIFFERENCE (vrací integer hodnotu, která indikuje rozdíl hodnot soundex mezi dvěma řetězci)
- REPLACE (pro nahrazení části znaků za jinou část znaků)
- SUBSTRING (vrátí požadovanou část řetězce)
- REPLICATE (duplikuje řetězec podle zadaného počtu opakování)
- UNICODE (vrátí unicode prvního znaku v řetězci)
- LEFT (vrací zadaný počet znaků zleva)
- REVERSE (vrátí řetězec v opačném pořadí)
- UPPER (malá písmenka v řetězci převede na velká)
- LEN (vrátí počet znaků zadaného řetězce, bez mezer)
- RIGHT (vrací zadaný počet znaků zprava)
- LOWER (velká písmenka v řetězci převede na malá)
- RTRIM (vyjme mezery za řetězcem)

### 3.6.9 Systémové statistické funkce

Můžeme sledovat informace o připojení a využití systémových prostředků SQL Serveru od posledního restartu. [4] například:

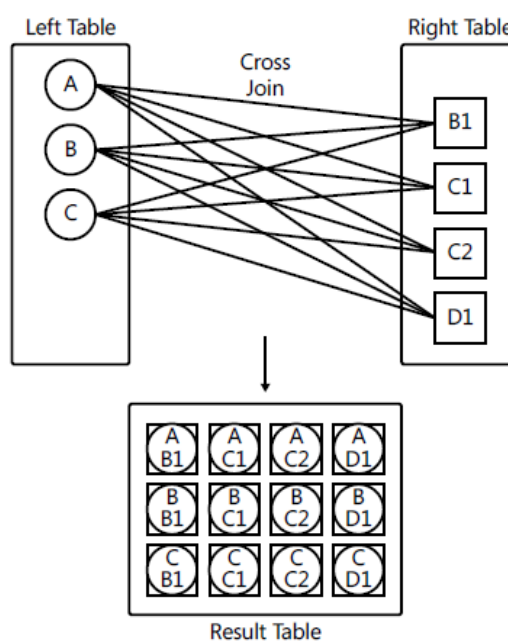
- @@CONNECTIONS (vrací počet pokusů o připojení od chvíle spuštění SQL Serveru, bez ohledu na to, jestli byli neúspěšné nebo úspěšné)
- @@TOTAL\_ERRORS (počet chyb zápisu na disk od chvíle posledního spuštění SQL Serveru)
- @@TOTAL\_READ (vrací počet čtení z disku od chvíle posledního spuštění SQL Serveru, do počtu nezapočítává vyrovnávací paměť).
- @@TOTAL\_WRITE (počet diskových zápisů od chvíle spuštění SQL Serveru)

### 3.7 Spojování tabulek pomocí JOIN

Když se v dotazu odkazujeme na data, která jsou uložena ve více tabulkách, je nutné využít příkazu JOIN, neboli spojení. T-SQL podporuje tři typy spojení – křížové, vnitřní, vnější. [2]

#### 3.7.1 Křížové spojení – CROSS JOIN

Křížové spojení je nejjednodušším typem spojení, ale méně používaným. Příkaz CROSS JOIN provede kartézský součin mezi dvěma vstupními tabulkami. [2]

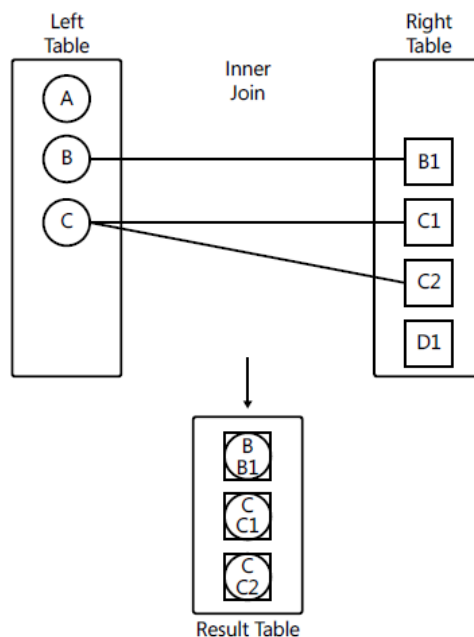


Obrázek 20 – Funkce CROSS JOIN [2]

Na obrázku máme dvě tabulky: Left Table ve které jsou tři řádky – A, B, C a Right Table, ve které máme 4 řádky – B1, C1, C2, D1. Po provedení CROSS JOIN bude vrácena tabulka o 12 řádcích obsahující kombinaci každého řádku z Left Table s každým řádkem Right table.

#### 3.7.2 Vnitřní spojení – INNER JOIN

Vnitřním spojením můžeme spojit řádky na základně predikátu – obvykle tak, že na jedné straně porovnáváme hodnotu primárního klíče s hodnotou cizího klíče na straně druhé. [2]



Obrázek 21 - Funkce INNER JOIN [2]

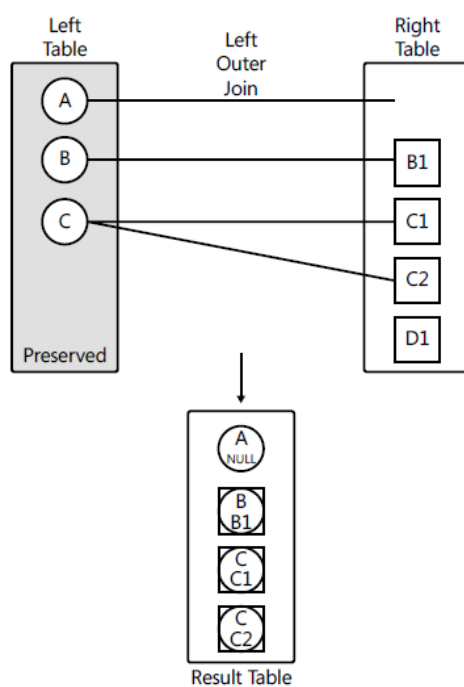
V tomto případě nám písmena v Left table označují PRIMARY KEY a hodnoty v Right table FOREIGN KEY. Vidíme, že PRIMARY KEY je jedinečný a může odpovídat více FOREIGN klíčům.

### 3.7.3 Vnější spojení – OUTER JOIN

Pomocí vnějšího spojení můžeme dostat všechny řádky z jedné, nebo obou stran. [2] Rozlišujeme tři varianty:

#### LEFT OUTER JOIN

Při volání LEFT OUTER JOIN říkáme, že chceme zachovat levou tabulku – budou vráceny také řádky z levé tabulky, které neobsahují žádnou vazbu z pravou tabulkou. [2]

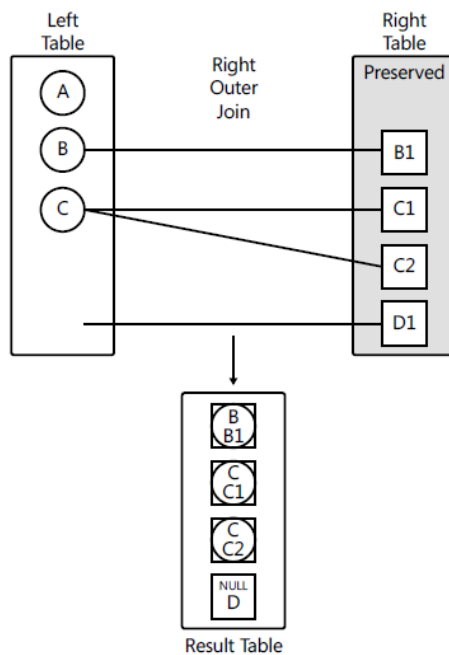


Obrázek 22- Funkce LEFT OUTER JOIN [2]

### 3.7.4 RIGHT OUTER JOIN

Chceme zachovat pravou tabulku – budou vráceny také řádky z pravé tabulky, které neobsahují žádnou vazbu s levou tabulkou. [2]

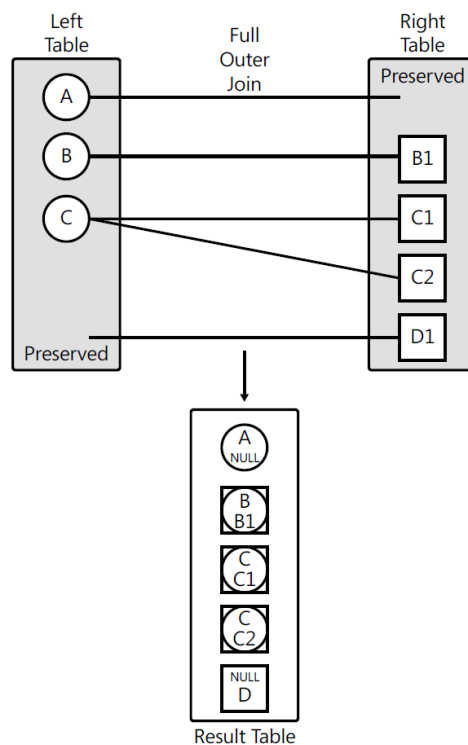




Obrázek 23 - Funkce RIGHT OUTER JOIN [2]

### 3.7.5 FULL OUTER JOIN

Zachovává obě strany. [2]



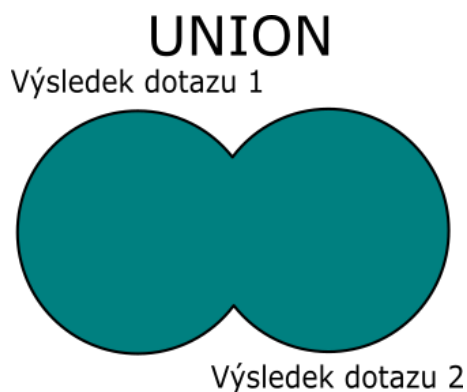
Obrázek 24 – Funkce FULL OUTER JOIN [2]

## Spojování tabulek pomocí UNION

Příkazy UNION a UNION ALL sjednocují výsledky dvou dotazů. [2]

### 3.7.6 UNION

Příkaz UNION sjednotí dva vstupní dotazy a vrátí výsledek bez duplicitních hodnot. Výsledek operace UNION můžeme vidět na obrázku



Obrázek 25 - Union

### 3.7.7 UNION ALL

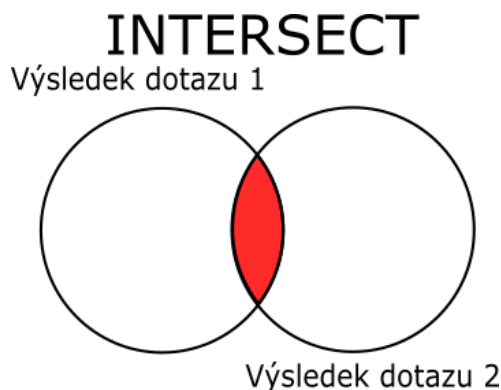
Příkaz UNION ALL také sjednotí dva vstupní dotazy, ale výsledek vrátí včetně duplicitních hodnot. Výsledek operace UNION ALL můžeme vidět na obrázku



Obrázek 26 - Union All

### 3.7.8 INTERSECT

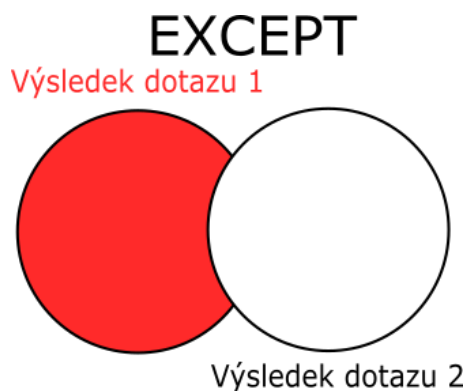
Vrátí pouze odlišné řádky, které jsou společné pro obě sady. Jinými slovy, pokud se objeví řádek alespoň jednou ve výsledku prvního dotazu a nejméně jednou ve výsledku druhého dotazu, objeví se jednou jako výsledek příkazu INTERSECT. Výsledek operace INTERSECT můžeme vidět na obrázku [2]



Obrázek 27 – Diagram operace Intersect

### 3.7.9 EXCEPT

Příkaz EXCEPT provede rozdíl mezi dvěma dotazy – vrátí odlišné řádky, které jsou ve výsledku prvního dotazu a zároveň nejsou ve výsledku druhého dotazu. Výsledek operace EXCEPT můžeme vidět na obrázku [2]



Obrázek 28 – Diagram operace Except

## 3.8 Tabulkové výrazy - Table Expressions

### 3.8.1 Common Table Expressions

Common Table Expression (CTE) jsou tzv. dočasné pohledy. CTE jsou velmi podobné odvozeným tabulkám – stejně jako u odvozených tabulek nejsou ukládány jako objekt a jsou v paměti pouze po dobu trvání dotazu. Na rozdíl od odvozených tabulek mohou CTE odkazovat sami na sebe vícekrát ve stejném dotazu. [21]

CTE mohou být použity pro:

- Rekurzivní dotazy
- Náhrada za pohled
- Povolování seskupení
- Referenční výslednou tabulku vícekrát v jedné podmínce.

### Struktura CTE

CTE se skládá z výrazu, jehož název reprezentuje CTE, volitelného seznamu sloupců a dotazu definující CTE. Pokud je operace CTE definována, mohou být na ni odkazovány tabulky a pohledy pomocí příkazů SELECT, INSERT, UPDATE a DELETE.

Syntaxe při definování CTE:

```
WITH nazev [seznamSloupce]
AS
(definice_CTE_dotazu)
```

Vyvolání CTE:

```
SELECT <seznamSloupce>
FROM nazev
```

### 3.8.2 Pohledy – Views

Pohled je virtuální tabulka, která vznikne jako výsledek nějakého dotazu. Jinými slovy to může být např. příkaz SELECT, který je uložen pod nějakým názvem v databázi. Pohledy jsou v praxi využívány pro: [4]

- Omezení přístupu (vynechání některých sloupců tabulek obsahujících citlivá data, např. platy zaměstnanců).
- Vytvoření sady dat pro opakované používání.
- Prezenci denormalizovaných dat, která jsou v databázi normalizovaná.

Jednoduchá syntaxe pro vytvoření pohledu:

```
CREATE VIEW jmenoPohledu
AS dotaz
```

Pro smazání pohledu je syntaxe následující:

```
DROP VIEW jmenoPohledu
```

### 3.9 Programování v SQL 2014

I když SQL není programovací jazyk, uvedu v této kapitole několik příkazů, které známe z programovacích jazyků a lze jejich funkci využívat v MS SQL Serveru.

#### 3.9.1 Vytváření proměnných

Stejně jako v ostatních programovacích jazycích, je v SQL serveru umožněno vytvářet vlastní proměnné. Rozlišujeme 3 typy: [4]

- Lokální proměnné (systémové nebo uživatelem definovaná data).
- Kurzorové proměnné
- Tabulkové proměnné

#### Vytvoření lokální proměnné

Základní syntaxe pro vytvoření lokální proměnné:

```
DECLARE @jmenoPromenne [AS] datovyTyp
```

Pokud nepřidáme žádnou hodnotu do proměnné, SQL ji při inicializování přiřadí hodnotu NULL. Přiřadit hodnotu do proměnné můžeme:

- Při deklaraci proměnné.
- Použitím klíčového slova SET.
- Použitím SELECT.

Při deklaraci proměnné:

```
DECLARE @cislo [init] = 30;  
DECLARE @text [varchar] (10) = N'ahoj';
```

Použitím klíčového slova SET:

```
DECLARE @cislo [int];  
DECLARE @text [varchar] (10);  
SET @cislo = 10;  
SET @text = N'ahoj';
```

Použitím SELECT:

```
DECLARE @cislo [int];  
DECLARE @text [varchar] (10);  
SELECT @cislo = 20;  
SELECT @text = N'ahoj';
```

### Vytvoření kurzorové proměnné

Kurzorové proměnné jsou určeny pro zajištění zpětné kompatibility s objekty, které byly vytvořeny pro starší verze SQL serveru.

```
DECLARE @kurzorovaPromenna CURSOR;
```

### Vytvoření tabulkové proměnné

Tyto proměnné umožňují ukládat data ve formě tabulek a jejich využití je vhodné pro menší datové sady. [4]

```
DECLARE @promennaTabulka TABLE  
{  
    Sloupec1 [int],  
    Sloupec2 [varchar] (30)  
};
```

### 3.9.2 Klíčová slova BEGIN & END

Klíčová slova BEGIN a END slouží k ohraničení kódového bloku. Mohou být vnořené. [4]

```
BEGIN  
{  
    prikazySQL  
}  
END
```

### 3.9.3 Podmínkový blok IF

Pro jednoduché rozhodování pravda / nepravda lze použít podmínkový blok IF. Pravidla jsou stejná jak v ostatních programovacích jazycích.

```
IF podminka { podminkaPravdiva }  
ELSE { podminkaNepravdiva }
```

### 3.9.4 Konstrukce CASE

Jako rozhodovací blok můžeme také využít CASE, jeho syntaxe je následující: [4]

```
CASE vstupniPodminka  
WHEN vysledek THEN  
prikazy  
WHEN vysledek THEN  
prikazy  
ELSE prikazy  
END;
```

### 3.9.5 WHILE, BREAK, CONTINUE

Příkaz WHILE slouží pro opakované provádění příkazů. Cyklus se provádí, dokud není splněna podmínka.

```
WHILE ukoncovaciPodminka  
{  
    prikazy  
}
```

WHILE můžeme doplnit klíčovými slovy BREAK a CONTINUE, které slouží k řízení logiky uvnitř cyklu. BREAK slouží k vyskočení ze smyčky WHILE. Naproti tomu CONTINUE slouží k restartování cyklu. Všechny příkazy, které jsou po klíčovém slovu CONTINUE budou ignorovány. [4]

### 3.10 Uložené procedury

Uložené procedury ulehčí práci – když hrozí, že budeme psát stejné dotazy znovu a znovu, můžeme vytvořit proceduru, kterou v případě nutnosti zavoláme. [2] Syntaxe při vytváření je následující:

```
CREATE PROC nazevProcedury
AS
    BEGIN
        SELECT...
    END
```

### 3.11 Vytváření triggerů

#### 3.11.1 DML Triggery

Triggery reagují na události typu INSERT, UPDATE, DELETE nebo na jejich vzájemnou kombinaci. Rozlišujeme dva typy:

- AFTER (Trigger je spuštěn, když událost, se kterou je asociován, dokončena. Může být definován pouze u permanentních tabulek.)
- INSTEAD OF (Instead of Trigger je spuštěn namísto události, se kterou je asociován. Může být definován u permanentních tabulek a pohledů.)

Vytváření AFTER triggeru:

```
CREATE TRIGGER nazevTriggeru
AFTER INSERT
AS
    BEGIN
        Tělo triggeru
    END;
```

Tento trigger se vykoná po operaci INSERT. Odstranění triggeru provedeme následujícím způsobem:



```
DROP TRIGGER nazevTriggeru
```

### 3.11.2 DDL Triggery

Pro DDL triggery namísto DML příkazu DDL event, po kterém se trigger spustí:

```
CREATE TRIGGER nazevTriggeru ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
BEGIN
    Tělo triggeru
END;
```

## 3.12 Uživatelem definované funkce

Smyslem uživatelem definované funkce je zapouzdřit opakovatelně používaný T-SQL kód, který po vykonání vrátí skalární hodnoty, nebo tabulku.

### 3.12.1 Skalární funkce

Funkce vrací skalární hodnotu a mohou být použity i v dotazech. Tělo funkcí musí být uzavřeno v bloku BEGIN/END, například jednoduchou funkcí pro sečtení dvou čísel můžeme napsat následujícím způsobem: [2]

```
CREATE FUNCTION dbo.FunctionName
(
    @param1 int,
    @param2 int
)
RETURNS INT
AS
BEGIN
    RETURN @param1 + param2
END
```

### 3.12.2 Funkce s tabulkovou návratovou hodnotou

Funkce mohou být psány bez bloku BEGIN/END a jejich použití je možné i v rámci dotazu.

[2]

```
CREATE FUNCTION dbo.FunctionName
(
    @param1 int,
    @param2 char(5)
)
RETURNS TABLE AS RETURN
(
    SELECT @param1 as c1,
           @param2 as c2
    )
```

## 4 PODKLADY PRO VÝUKU

Termíny probírané v první části bakalářské práce jsem shrnul do prezentací, které budou sloužit jako podkladový materiál pro přednášejícího. V tomto materiálu se také nachází shrnutí všech příkazů které byly probírány v části druhé.

Jako podklady pro cvičení jsem využil zejména praktickou část práce, kdy jsem popsal příkazy jazaka SQL a uvedl příklady použití. Prezentace jsem také doplnil úkoly, které mají ověřit pochopení problematiky probírané v daném týdnu. Součástí je také stažení a instalace SQL Serveru – obsah prvního cvičení.

Dále jsem vytvořil 60 testovacích otázek, které jsou z problematiky probírané v této práci. Obsahují jak teoretické tak praktické části práce a jsou vhodné k importu do systému Moodle. Testovací otázky byly předány vedoucímu bakalářské práce.

## ZÁVĚR

Cílem práce bylo vytvořit učební pomůcku pro výuku Microsoft SQL server 2014. Práce je psána tak, aby ji pochopil i začátečník.

V první části byla zpracována teorie databázových systémů obecně a vysvětleny základní pojmy pro pochopení problematiky databází. Při vytváření prezentací, jsem tuto část bakalářské práce rozdělil do logických bloků, které budou sloužit jako podklady pro vyučujícího na přednášky.

Dále v první části můžeme najít stručnou historii SQL Serveru a popis jednotlivých edicí, které jsou v současné době na trhu dostupné. Jsou zde uvedeny i nové funkce a vylepšení aktuální verze tohoto databázového systému.

V druhé části bakalářské práce je vysvětlena syntaxe a funkce příkazů, používaných v MS SQL Serveru. Použit je jazyk SQL a T-SQL, u kterého jsem popsal nejpoužívanější příkazy pro všemožnou práci s daty uloženými v databázi. Praktickou část bakalářské práce jsem taktéž rozdělil do bloků a zanesl do prezentací, které budou sloužit jako podklady pro výuku na cvičení. Tyto podklady jsem dále doplnil o příklady, ukázky funkce příkazů SQL a také o úkoly, které by měl studující po cvičení zvládnout.

Čtrnáctý týden výuky je určen pro závěrečný test, proto je součástí práce i soubor 60 testovacích otázek, vhodné k importu do systému MOODLE. Tento soubor testovacích otázek byl předán vedoucímu bakalářské práce.

## SEZNAM POUŽITÉ LITERATURY

- [1] JORGENSEN, Adam. Professional microsoft sql server 2014 administration. 1st edition. Indianapolis, IN: John Wiley and Sons, 2014, pages cm. ISBN 1118859138.
- [2] BEN-GAN, Itzik, Dejan SARKA a Ron TALMAGE. Querying Microsoft SQL Server 2012: exam 70-461 training kit. Sebastopol, Calif.: Microsoft, c2012, xxx, 704 p. ISBN 0735666059.
- [3] KROENKE, David a David J AUER. Databáze. 1. vyd. Brno: Computer Press, 2015, 496 s. ISBN 978-80-251-4352-0.
- [4] MASOOD-AL-FAROOQ, B. A. SQL Server 2014 Development Essentials. UK.: Packt Publishing, 2014. ISBN 978-1-78217-255-0.
- [5] CORONEL, Carlos a Steven MORRIS. Database systems: design, implementation, and management. 11e [edition]. United States: Course Technology Cengage Learning, 2015, xxvii, 751 pages. ISBN 128519618x.
- [6] CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK. Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází. Vyd. 1. Brno: Computer Press, 2009, 584 s. ISBN 978-80-251-2328-7.
- [7] Novinky v SQL Server 2014 - díl 1. [online]. 2014 [cit. 2016-02-10]. Dostupné z: <http://www.zive.cz/clanky/novinky-v-sql-server-2014---dil-1/sc-3-a-173564/default.aspx>
- [8] Novinky v SQL Server 2014 - díl 2. [online]. 2014 [cit. 2016-02-10]. Dostupné z: <http://www.zive.cz/clanky/novinky-v-sql-server-2014---dil-2/sc-3-a-174442/default.aspx>
- [9] *What Is Cloud Computing?* [online]. 2015 [cit. 2016-02-13]. Dostupné z: <http://www.pcmag.com/article2/0,2817,2372163,00.asp>
- [10] *Výhody Cloud computingu a překážky v jeho prosazení* [online]. 2013 [cit. 2016-02-13]. Dostupné z: <http://www.middleware.cz/cloud-computing/3-vyhody-cloud-computingu-a-prekazky-v-jeho-prosazeni>
- [11] *Microsoft Azure* [online]. [cit. 2016-02-13]. Dostupné z: <https://azure.microsoft.com/cs-cz/overview/what-is-azure/>
- [12] *Seznámení s windows azure* [online]. 2013 [cit. 2016-02-13]. Dostupné z: <http://www.dotnetportal.cz/clanek/960/Seznameni-s-Windows-Azure>

- [13] *Edice systému SQL Server: Edice Express* [online]. 2014 [cit. 2016-02-13]. Dostupné z: <https://www.microsoft.com/cs-cz/server-cloud/products/sql-server-editions/sql-server-express.aspx>
- [14] *Edice systému SQL Server: Edice Business Intelligence* [online]. 2014 [cit. 2016-02-13]. Dostupné z: <https://www.microsoft.com/cs-cz/server-cloud/products/sql-server-editions/sql-server-business-intelligence.aspx?TabIndex=0>
- [15] *Edice systému SQL Server: Edice Standard* [online]. 2014 [cit. 2016-02-13]. Dostupné z: <https://www.microsoft.com/cs-cz/server-cloud/products/sql-server-editions/sql-server-standard.aspx?TabIndex=0>
- [16] *Edice systému SQL Server: Edice Enterprise* [online]. 2014 [cit. 2016-02-13]. Dostupné z: <https://www.microsoft.com/cs-cz/server-cloud/products/sql-server-editions/sql-server-enterprise.aspx?TabIndex=0>
- [17] *The History of MS SQL Server* [online]. 2007 [cit. 2016-02-20]. Dostupné z: <http://ransara.blogspot.cz/2007/11/history-of-ms-sql-server.html>
- [18] *Seznámení a instalace microsoft sql serveru* [online]. 2009 [cit. 2016-02-20]. Dostupné z: <http://www.dotnetportal.cz/clanek/140/Seznameni-a-instalace-Microsoft-SQL-Serveru>
- [19] *DateTime a SmallDatetime. Microsoft TechNet* [online]. [cit. 2016-02-20]. Dostupné z: [https://technet.microsoft.com/en-us/library/aa258277\(v=sql.80\).aspx](https://technet.microsoft.com/en-us/library/aa258277(v=sql.80).aspx)
- [20] *SQL Server 2008 datové typy. Programujte.com* [online]. 2010 [cit. 2016-02-20]. Dostupné z: <http://programujte.com/clanek/2010022200-sql-server-2008-datove-typy/>
- [21] *Using Common Table Expression. Microsoft TechNet* [online]. [cit. 2016-02-27]. Dostupné z: [https://technet.microsoft.com/en-us/library/ms190766\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190766(v=sql.105).aspx)
- [22] *Agregate Functions (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms173454\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms173454(v=sql.110).aspx)
- [23] *AVG (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms177677\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms177677(v=sql.110).aspx)
- [24] *MIN (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms179916\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms179916(v=sql.110).aspx)
- [25] *MAX (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms187751\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms187751(v=sql.110).aspx)

- [26] *SUM (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms187810\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms187810(v=sql.110).aspx)
- [27] *CHECKSUM\_AGG (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms188920\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms188920(v=sql.110).aspx)
- [28] *COUNT (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms175997\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms175997(v=sql.110).aspx)
- [29] *COUNT\_BIG (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms190317\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms190317(v=sql.110).aspx)
- [30] *STDEV (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms190474\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms190474(v=sql.110).aspx)
- [31] *STDEVP (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms176080\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms176080(v=sql.110).aspx)
- [32] *GROUPING (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms178544\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms178544(v=sql.110).aspx)
- [33] *GROUPING\_ID (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/bb510624\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/bb510624(v=sql.110).aspx)
- [34] *VAR (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms186290\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms186290(v=sql.110).aspx)
- [35] *VARP (Transact-SQL): Microsoft Developer Network* [online]. [cit. 2016-03-07]. Dostupné z: [https://msdn.microsoft.com/en-us/library/ms188735\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/ms188735(v=sql.110).aspx)
- [36] Buffer. *IT Slovník* [online]. [cit. 2016-04-10]. Dostupné z: <http://it-slovník.cz/pojem/buffer>
- [37] Provnání edic. *Microsoft* [online]. [cit. 2016-04-10]. Dostupné z: <https://www.microsoft.com/cs-cz/server-cloud/products/sql-server-editions/overview.aspx>
- [38] Základy normalizace databáze. *Microsoft* [online]. [cit. 2016-04-11]. Dostupné z: <https://support.microsoft.com/cs-cz/kb/283878>
- [39] Priorita datových typů. *Microsoft* [online]. [cit. 2016-04-11]. Dostupné z: [https://technet.microsoft.com/cs-cz/library/ms190309\(v=sql.100\).aspx](https://technet.microsoft.com/cs-cz/library/ms190309(v=sql.100).aspx)

- [40] *Logické operátory* [online]. [cit. 2016-05-04]. Dostupné z: <https://msdn.microsoft.com/en-us/library/ms189773.aspx>
- [41] *Normalizace relačních databází* [online]. [cit. 2016-05-16]. Dostupné z: <http://programujte.com/clanek/2008071900-normalizace-relacnich-databazi/>
- [42] *Slovník Cizích Slov: Business Intelligence* [online]. [cit. 2016-05-22]. Dostupné z: <http://slovník-cizich-slov.abz.cz/web.php/slovo/business-intelligence-zkratka-bi>
- [43] *Základní přehled SQL příkazů* [online]. [cit. 2016-05-26]. Dostupné z: <http://www.fit.vutbr.cz/study/courses/TW1/public/info/lib/exe/fetch.php?media=iw4:iw4-zakladni-prehled-sql-prikazu.pdf>



**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

SQL	Structured Query Language.
DBMS	Database Management System.
IBM	International Business Machines.
T-SQL	Transact-SQL.
DDL	Data definition language.
DML	Data manipulation language.
DCL	Data control language.
MS	Microsoft.
SSD	Solid State disk.
BI	Business intelligence
NF	Normal Form
BCNF	Boyce Coddova normální forma

**SEZNAM OBRÁZKŮ**

Obrázek 1 – Vazba 1:1 [4] .....	18
Obrázek 2 – Vazba 1:N [4] .....	18
Obrázek 3 – vazba N:M [4] .....	19
Obrázek 4 – Srovnání normálních forem [41] .....	20
Obrázek 5 – Životní cyklus databáze [6] .....	28
Obrázek 6 – Výběr jazyka před stažením SQL Serveru .....	44
Obrázek 7 – Výběr verze ke stažení .....	44
Obrázek 8 – Výběr složky pro extrahování instalačních souborů .....	45
Obrázek 9 – Nabídka instalačního programu .....	45
Obrázek 10 – Licenční podmínky .....	46
Obrázek 11 – Výběr modulů k instalaci .....	46
Obrázek 12 – Možnost pojmenování instance SQL Serveru .....	47
Obrázek 13 – Výběr typu spouštění .....	47
Obrázek 14 – Nastavení Database Engine .....	48
Obrázek 15 – Reporting Services .....	49
Obrázek 16 – Průběh instalace .....	49
Obrázek 17 – Report z instalace .....	50
Obrázek 18 – Po spuštění .....	50
Obrázek 19 – Výběr databázového Engine .....	51
Obrázek 20 – Funkce CROSS JOIN [2] .....	75
Obrázek 21 - Funkce INNER JOIN [2] .....	76
Obrázek 22- Funkce LEFT OUTER JOIN [2] .....	77
Obrázek 23 - Funkce RIGHT OUTER JOIN [2] .....	78
Obrázek 24 – Funkce FULL OUTER JOIN [2] .....	78
Obrázek 25 - Union .....	79
Obrázek 26 - Union All .....	79
Obrázek 27 – Diagram operace Intersect .....	80
Obrázek 28 – Diagram operace Except .....	80

**SEZNAM TABULEK**

Tabulka 1 – Tabulka záznamy cestovní kanceláře v nulté formě.....	20
Tabulka 2 – Tabulka záznamy cestovní kanceláře v první normální formě 1NF .....	21
Tabulka 3 – Tabulka objednávky cestovní kanceláře 2NF .....	21
Tabulka 4 – Tabulka zájezdy 2NF .....	22
Tabulka 5 – Tabulka objednávky cestovní kanceláře 3NF .....	22
Tabulka 6 – Tabulka zájezdy 3NF .....	22
Tabulka 7 – Tabulka pobočky .....	22
Tabulka 8 – Minimální systémové požadavky [1].....	32
Tabulka 9 – Porovnání edic [37].....	39
Tabulka 10 – Aritmetické operátory .....	52
Tabulka 11 – Operátory pro porovnávání .....	52
Tabulka 12 – Logické operátory [40] .....	53
Tabulka 13- znakové datové typy [20] .....	53
Tabulka 14 – Číselné datové typy [20] .....	54
Tabulka 15 - Datové typy s plovoucí desetinnou čárkou [1].....	55
Tabulka 16 – Datum a čas [1].....	56
Tabulka 17 – DateTime [19].....	57
Tabulka 18 - Binární datové typy [1].....	57
Tabulka 19 – Datové typy řetězců unicode [1].....	58
Tabulka 20 – Priorita datových typů [39] .....	58
Tabulka 21 – Filtrace znakových dat [2] .....	63
Tabulka 22 - Datové typy návratových hodnot AVG [23] .....	67
Tabulka 23 – Datové typy návratových hodnot SUM [26] .....	68

## **SEZNAM PŘÍLOH**

PŘÍLOHA P I: SOUBOR VÝUKOVÝCH MATERIÁLŮ NA CD

## **PŘÍLOHA P I: SOUBOR VÝUKOVÝCH MATERIÁLŮ NA CD**

Přiložený disk CD obsahuje složku *Prednasky*, kde jsou prezentace sloužící jako podklad na přednášky a složku *Cviceni*, kde jsou prezentace sloužící jako podklad na cvičení.

