

# Optimalizace datové struktury v systémech technické bezpečnosti

Data Structure Optimization in Technical Safety Systems

Tereza Špalková

---

Bakalářská práce  
2013



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2012/2013

## **ZADÁNÍ BAKALÁŘSKÉ PRÁCE**

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Tereza ŠPALKOVÁ**  
Osobní číslo: **A10159**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **prezenční**

Téma práce: **Optimalizace datové struktury v systémech  
technické bezpečnosti**

Zásady pro vypracování:

1. Seznamte se s relačními databázemi a jazykem SQL.
2. Analyzujte metody práce s daty.
3. Popište způsob a metody optimalizace SQL dotazu.
4. Seznamte se současným přístupem k ukládání dat.
5. Navrhněte vhodnou optimalizaci a řešení nalezených nedostatků.
6. Navržené řešení otestujte a proveďte vyhodnocení přínosů.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **MARTIN GRUBER. Mistroství v SQL. Praha 8: SoftPress s.r.o., 2004. ISBN 80-86497-62-3**
2. **LUBOSLAV LACKO. Oracle Správa, programování a použití databázového systému. Praha: Computer Press, 2003. ISBN 80-7226-699-3.**
3. **MARK MASLAKOWSKI. Naučte se v MySQL za 21 dní. Praha 4: Computer press, 2001. ISBN 80-7226-448-6.**
4. **THOMAS KYTE. Oracle: Návrh a tvorba aplikací. Brno: CP Books, a.s., 2005. ISBN 80-251- 0569-5.**
5. **BEGG, Carolyn a Richard HOLOWCZAK. THOMAS CONOLLY. Mistrovství – databáze: Profesionální průvodce tvorbou efektivních databází. Brno: Computer press, 2009. ISBN 978-80-251-2328-7.**
6. **WEINBERG, Paul N. JAMES R. GROFF. SQL Kompletní průvodce. Brno: CP Books, a.s., 2005. ISBN 80-251-0369-2.**

Vedoucí bakalářské práce:

**Ing. Petr Šilhavý, Ph.D.**

Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce:

**24. února 2013**


Termín odevzdání bakalářské práce:

**14. června 2013**

Ve Zlíně dne 24. února 2013

  
prof. Ing. Vladimír Vašek, CSc.  
*děkan*



  
prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## ABSTRAKT

Bakalářská práce je zaměřena na analýzu a optimalizaci datové struktury v systémech technické bezpečnosti.

Cílem práce je seznámit se současným systémem a posléze navrhnout vhodnou optimalizaci systému, která povede ke zvýšení jejího výkonu a snížení množství ukládaných dat a tabulek.

Navržené řešení systému bylo posléze otestováno a vyhodnoceno.

Klíčová slova:

optimalizace, databáze, SQL dotaz, data

## ABSTRACT

The bachelor's work is focused on data structure analysis and optimization of technical safety systems.

The aim of work is to familiarize oneself with the current system and propose suitable optimization of the system to increase system power and to reduce the amount of stored data and tables.

The proposed system solution was subsequently tested and evaluated.

Keywords:

optimization, database, SQL query, data

**Poděkování**

Ráda bych tímto poděkovala vedoucímu bakalářské práce Ing. Petrovi Šilhavému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

Děkuji rovněž panu Miloši Hruškovi z firmy Envinet, a.s. za odborné konzultace a rady v průběhu zpracování této bakalářské práce.

**Prohlašuji, že**

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

**Prohlašuji,**

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

Ve Zlíně 26. 5. 2013

.....

## OBSAH

<b>ÚVOD.....</b>	<b>10</b>
<b>I TEORETICKÁ ČÁST .....</b>	<b>11</b>
<b>1 ÚVOD DO RELAČNÍCH DATABÁZÍ.....</b>	<b>12</b>
1.1 DATA A INFORMACE .....	12
1.2 DATABÁZE ANEB RELAČNÍ DATABÁZE.....	13
1.2.1 Relaçní datový model .....	14
1.2.2 Tabulky.....	14
1.2.3 Relace .....	16
1.3 JAK RELAČNÍ DATABÁZE FUNGUJÍ.....	17
1.4 VZOROVÉ DATABÁZOVÉ SCHÉMA .....	18
1.5 SHRNUÍ.....	20
<b>2 JAZYK SQL .....</b>	<b>21</b>
2.1 CO JE JAZYK SQL.....	21
2.2 ÚLOHA JAZYKA SQL .....	22
2.3 JAK SQL PRACUJE .....	24
2.4 RYSY A VÝHODY JAZYKA SQL .....	25
2.5 SHRNUÍ.....	26
<b>3 PRÁCE S DATY V DATABÁZÍCH.....</b>	<b>27</b>
3.1 JEDNODUCHÉ DOTAZY - SELECT .....	27
3.1.1.1 Vyvolání všech sloupců a všech řádků .....	28
3.1.2 Vyvolání zadaného sloupce, všech řádků.....	29
3.1.3 Použití DISTINCT .....	30
3.2 VÝBĚR ŘÁDKŮ – KLAUZULE WHERE .....	31
3.2.1 Vyhledávací podmínky porovnání .....	31
3.2.2 Vyhledávací podmínka rozsahu (BETWEEN / NOT BETWEEN) .....	32
3.2.3 Vyhledávací podmínka náležení do množiny (IN / NOT IN) .....	33
3.2.4 Vyhledávací podmínka porovnávání se vzorem (LIKE / NOT LIKE).....	34
3.2.5 Vyhledávací podmínka hodnoty (IS NULL / IS NOT NULL).....	35
3.3 ŘAZENÍ VÝSLEDKŮ – KLAUZULE ORDER BY .....	35
3.3.1 Uspořádání výsledků .....	36
3.3.2 Použití agregačních funkcí SQL.....	37
3.3.2.1 Použití funkce SUM.....	37
3.3.2.2 Použití funkcí MIN, MAX a AVG .....	38
3.4 SESKUPENÍ VÝSLEDKŮ – KLAUZULE GROUP BY .....	39
3.4.1 Použití GROUP BY .....	39
3.4.2 Použití HAVING.....	41
3.5 SHRNUÍ.....	41
<b>4 OPTIMALIZACE SQL DOTAZŮ .....</b>	<b>42</b>

4.1	INDEXY.....	42
4.1.1	Kdy používat indexy .....	43
4.1.2	Typy indexů.....	43
4.1.2.1	Indexy stromů B+.....	44
4.1.2.2	Indexy spojení a klastry .....	44
4.1.2.3	Bitmapové indexy .....	45
4.1.2.4	Různé formulace dotazů .....	45
4.2	SHRNUTÍ.....	48
<b>5</b>	<b>SEZNÁMENÍ S DATABÁZOVÝM SYSTÉMEM ORACLE .....</b>	<b>49</b>
5.1	HISTORIE ORACLE .....	49
5.2	SOUČASNÁ PODOBA ORACLE.....	50
5.3	PROČ ORACLE? .....	51
5.4	POROVNÁNÍ ORACLE DATABASE S MYSQL.....	51
5.4.1	Oracle Database.....	51
5.4.1.1	Výhody.....	51
5.4.1.2	Nevýhody .....	52
5.4.2	MySQL.....	52
5.4.2.1	Výhody.....	52
5.4.2.2	Nevýhody .....	52
5.4.3	Porovnání .....	53
	SHRNUTÍ	53
<b>II</b>	<b>PRAKTICKÁ ČÁST .....</b>	<b>54</b>
<b>6</b>	<b>OPTIMALIZACE VYBRANÉ DATABÁZE .....</b>	<b>55</b>
6.1	SOUČASNÝ STAV DATABÁZE PŘED OPTIMALIZACÍ.....	55
6.1.1	Soupis použitých tabulek v databázi – před optimalizací .....	56
6.1.2	Analýza současné databáze .....	56
6.1.3	Návrh optimalizace .....	62
6.1.3.1	Vytvoření indexů .....	62
6.1.3.2	Zmenšit objem dat přesunutím starších dat do archivní tabulky. ....	64
6.1.3.3	Optimalizace číselníků.....	64
6.2	STAV PO OPTIMALIZACI .....	65
6.2.1	Soupis použitých tabulek v databázi – po optimalizaci .....	65
6.2.2	Ověření navrženého řešení .....	66
6.2.2.1	Vytvoření archivních tabulek.....	66
6.2.2.2	Doplnění nových indexů .....	66
	<b>ZÁVĚR .....</b>	<b>73</b>
	<b>CONCLUSION .....</b>	<b>75</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>77</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>78</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>79</b>
	<b>SEZNAM TABULEK.....</b>	<b>81</b>



<b>SEZNAM PŘÍLOH.....</b>	<b>82</b>
---------------------------	-----------

## ÚVOD

Pro svoji bakalářskou práci jsem si zvolila téma Optimalizace datové struktury v systémech technické bezpečnosti. Výběr tématu byl ovlivněn zvědavostí se dozvědět více o jazyku SQL a jeho využití při práci s databázovými systémy.

Bakalářská práce bude rozdělena na teoretickou a praktickou část. V teoretické části se zaměřím na relační databázi, která patří v současnosti k nejčastěji používané. Při představení databáze popíši základní pojmy, schémata jednotlivých relačních databází, co je to primární a cizí klíč a na to jak relační databáze fungují.

V další části popíši jazyk SQL. Představím pro lepší pochopení a srozumitelnost, jak jazyk SQL pracuje, a jakou má úlohu při vytváření a práci s daty v relační databázi.

Podrobněji se zaměřím na nepoužívanější příkaz SELECT, který se používá k výběru a prezentaci uložených dat. Možnosti optimalizace databáze pomocí indexů a optimalizace SQL dotazů. Hlavním cílem optimalizace je zvýšení rychlosti práce s databází.

V závěru teoretické části se částečně zmíním o databázovém systému Oracle, jeho historii, současnosti a srovnáním s databázovým systémem MySQL.

V praktické části zpracuji návrh optimalizace vybrané databáze, která se používá pro evidování, plánování a vyhodnocení prováděných kontrolních činností na zařízení podléhající předepsané legislativě, sledování a dokladování technické bezpečnosti.

Cílem celé práce je analyzovat a navrhnout optimalizaci vybrané databáze. Identifikovat možnosti optimalizace, navrhnout řešení a ověřit a vyhodnotit optimalizovanou databázi.

Důraz bude kladen na analýzu a optimalizaci celého systému z pohledu výkonu a snížení množství ukládaných dat a tabulek, tak i z pohledu budoucích modifikací, a zjednodušení údržby celé databáze. Vhodně navržená optimalizace datové struktury, nám umožní účelné a přehledné uložení dat.

Použité zdroje mi poskytly dostačující informace pro zpracování této bakalářské práce. Při své práci jsem se setkala s velice milým přístupem k poskytnutí dat, údajů a zodpovězení všech mých dotazů.

## **I. TEORETICKÁ ČÁST**

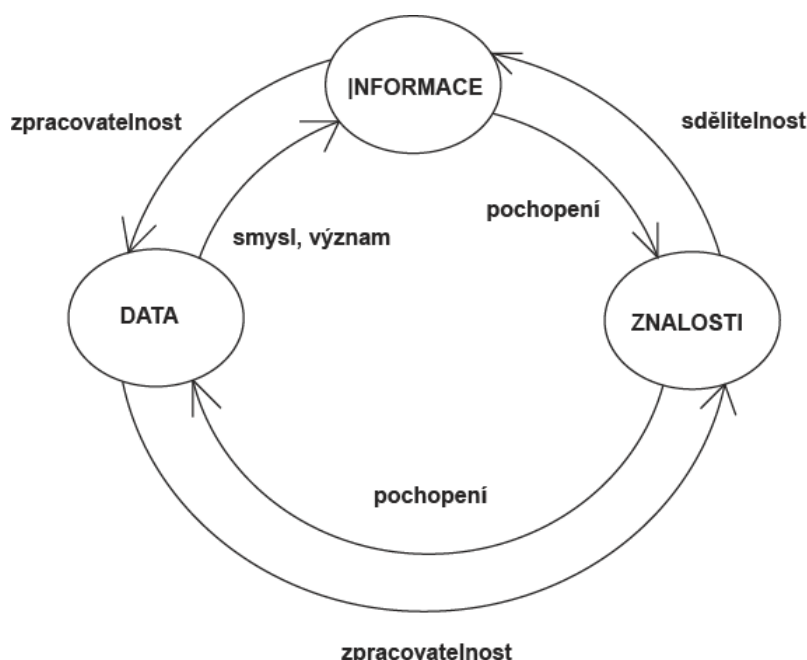
# 1 ÚVOD DO RELAČNÍCH DATABÁZÍ

## 1.1 Data a Informace

**Data** jsou nezpracovaná fakta, [3] která vykazují určitou důležitost pro jednotlivce nebo organizace.

**Informace** jsou data, která jsou zpracována nebo dostala strukturu, která jim poskytuje význam [3] (organizaci nebo i jednotlivci).

Například si můžeme představit data „*Ann Peters I Casino Royale*“. [3] Všechny tyto části informace, které se nacházejí pohromadě, nemají pro nás žádný význam bez další struktury. Můžeme usuzovat že „*Ann Peters*“ je osoba a „*Casino Royale*“ je jméno knihy nebo filmu, ale pak „*I*“ může znamenat několik různých věcí. Zatím co „*Ann Peters*“ má co do činění se společností pronajímající DVD tak „*I*“ a „*Casino Royale*“ představují data týkající se seznamu jednotlivých požadovaných DVD. S použitím další struktury jsme jednotlivá data (surová data) transformovali na informace, tedy na data, která pro nás mají nějaký význam (obrázek 1).



Obrázek 1 – Data a informace [6]

## 1.2 Databáze aneb relační databáze

Databáze je jediné veliké úložiště dat, které je možné používat současně mnoha odděleními nebo jednotlivými uživateli. Obecně je databáze strukturovaný soubor trvalých informací uložených počítačovým programem. [1] Pojem trvalý znamená, že po skončení programu nebo uživatelské relaci, jednotlivá data přežijí a neztratí se.

Relační databáze je databáze složená z několika tabulek, kde některé sloupce jsou spojeny se sloupci v jiných tabulkách. Určitým způsobem, jsou takto spojená datová pole na sobě závislá.

Představme si tabulku „*Objednávky*“, která obsahuje informace, které jsou potřeba k úspěšnému vytvoření objednávky (číslo objednávky, datum objednání určité položky a datum jejího odeslání). Máme, taky tabulku „*Zákazníci*“, která obsahuje všechny informace o daném zákazníkovi (název společnosti nebo adresa). Neobdrželi bychom žádné objednávky, pokud bychom nezískali žádné zákazníky. Obě tabulky tedy mohou být ve vzájemném vztahu neboli relaci (obrázek 2). [3]

V 70. letech byl navržen panem E. F. Coddem *model relační databáze*. [3] Navrhoval, že by se databáze měla skládat z tabulek, jejichž sloupce by měli být se sloupci jiných tabulek v určitém vztahu. Od tehdejšího hierarchického uspořádání souborového systému byly tyto analýzy, avšak na míle vzdáleny. V tvorbě a používání databází, jeho myšlenky vyvolaly revoluci.

Relační databáze je velmi intuitivní a zároveň kopíruje způsob myšlení lidí. Lidé potom mají tendenci rozebírat složité objekty na jednodušší nebo do uspořádaných [3] (systematických) celků shromažďovat podobné objekty. Ve skutečnosti charakter relační databáze je úplně stejný. Její vnitřní logiku lehce zvládneme především proto, že styl našeho myšlení napodobuje.

K provedení svých úkolů většina moderních databází používá relační model. [3] Systém T-SQL je v tomto stejný, protože je naprosto poddaný relačnímu modelu a to ulehčuje ještě více jeho používání.

### 1.2.1 Relační datový model

Tabulka PRODUKTY

POPIS	CENA	POCET_SKLADEM
Size 3 Widget	\$107.00	207
Size 4 Widget	\$117.00	139
Hinge Pin	\$350.00	14
.		
.		
.		

Tabulka OBJEDNAVKY

CISLO_OBJ	ZAKAZNIK	PRODUKT	POCET_KUSU
112963	Acme Mfg.	41004	28
112975	JCP Inc.	2A44G	6
112983	Acme Mfg.	41004	6
113012	JCP Inc.	41003	35
.			
.			
.			

Tabulka ZAKAZNICI

SPOLECNOST	ZAK_PRODEJCE	MAX_UVER
Acme Mfg.	105	\$50,000.00
JCP Inc.	103	\$50,000.00
.		
.		
.		

Obrázek 2 – Relační databáze pro zpracování objednávky [6]

### 1.2.2 Tabulky

Tabulky jsou základním organizačním prvkem v relačních databázích [6], kde jsou uspořádány hodnoty dat v řádcích a sloupcích. Všechny tabulky, které relační databáze obsahuje, mají jedinečný název, který ztotožňuje jejich obsah.

Pro představu, struktura tabulky řádek-sloupec je srozumitelně zobrazena na obrázku 3, kde vidíme zvětšené znázornění tabulky „*POBOCKY*“. Samostatnou fyzickou entitu – jednu prodejní kancelář, reprezentuje každý řádek tabulky „*POBOCKA*“. Pospolu všech pět řádků tabulky, pak tvoří všech pět prodejních kanceláří dané společnosti. Data vymezená v daném řádku tabulky se vztahují k dané kanceláři, reprezentované tímto řádkem. [6]

Jeden datový údaj, který se uloží v databázi pro každou kancelář, představuje každý sloupec tabulky „*POBOCKY*“. [6] Místo, kde jsou umístěny, jednotlivé kanceláře představuje sloupec „*MESTO*“. Informace o celkovém obratu kanceláře je uložena ve sloupci „*PRODEJE*“. Evidenční číslo pracovníka, který zodpovídá za řízení kanceláří je uvedeno ve sloupci „*VED*“.

POBOCKA	MESTO	OBLAST	VED	PLAN_RODEJ	PRODEJE
22	Denver	Western	108	\$300,000.00	\$186,042.00
11	New York	Eastern	106	\$575,000.00	\$692,637.00
12	Chicago	Eastern	104	\$800,000.00	\$735,042.00
13	Atlanta	Eastern	105	\$350,000.00	\$367,911.00
21	Los Angeles	Western	108	\$725,000.00	\$835,915.00
.					
.					
.					

Údaje v tomto řádku náleží této kanceláři

Údaje v tomto řádku náleží této kanceláři

Město kde jsou umístěny jednotlivé kanceláře

Evidenční číslo vedoucího kanceláře

Obrat Kanceláře

Obrázek 3 – Struktura řádek – sloupec relační tabulky POBOCKY [6]

Každý řádek v tabulce obsahuje v každém sloupci přesně jednu datovou hodnotu. V rámci jednoho sloupce, mají všechny hodnoty dat v celé tabulce stejný datový typ. [6] Např. všechny hodnoty sloupce „*MESTO*“ jsou výrazy, všechny hodnoty sloupce „*PRODEJE*“ nebo hodnoty sloupce „*VED*“ jsou celá čísla.

Doménou sloupce, označujeme množinu hodnot dat, kterou daný sloupec obsahuje. Množina názvů všech měst, je doménou sloupce „*MESTO*“, zatímco libovolná finanční částka je doménou sloupce „*PRODEJE*“. Doménou sloupce „*OBLAST*“ jsou jen dvě hodnoty dat, „*Eastern*“ a „*Western*“, jelikož to jsou jediné prodejní oblasti (regiony), které vlastní společnost.

Všechny sloupce tabulky mají vlastní název sloupce, který se zapisuje jako záhlaví nad sloupcem. [6] Ve dvou různých tabulkách mohou mít dva sloupce totožný název, ale v rámci jedné tabulky musí mít každý sloupec jedinečný název, odlišný od ostatních sloupců tabulky. Nejčastěji se používají názvy sloupců jako např. *jméno*, *adresa*, *množství*, *cena a prodeje*, které najdeme u mnoha různých tabulek, produkční databáze.

U tabulky, která je poprvé zhotovena se přesně určí pořadí sloupce tabulky a to zleva doprava. Tabulka musí vždy obsahovat aspoň jeden sloupec. Maximální počet sloupců v tabulce není vymezen ani standardem ANSI/ISO SQL, ale většina komerčních SQL

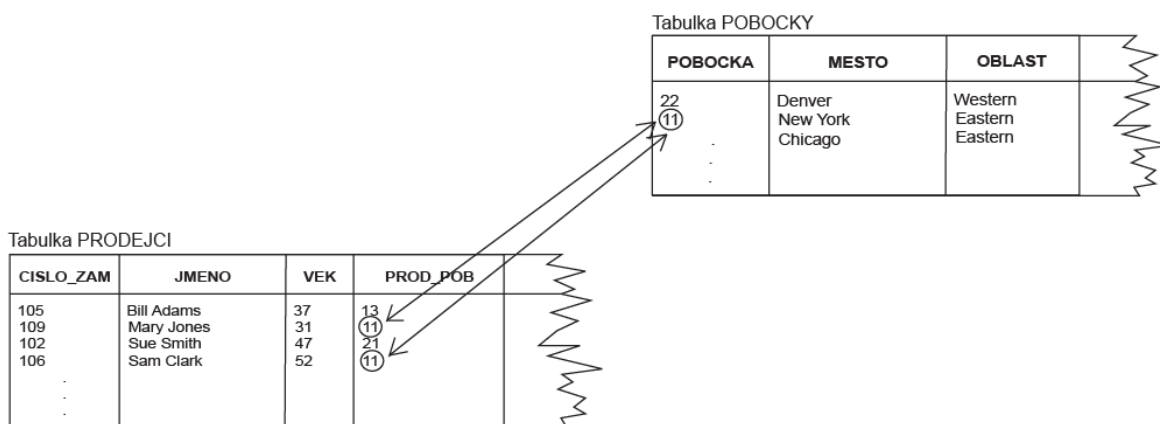
produktů zavádí určitý limit maximálního počtu sloupců. Obvykle se to pohybuje kolem 255 sloupců ale někdy i více. [6]

Sloupce v tabulce nemají specifické pořadí, ve srovnání s řádky. Ve skutečnosti není zajištěno, že řádky budou dvakrát zobrazeny ve stejném pořadí, v případě že použijeme za sebou dva databázové dotazy pro vyobrazení obsahu tabulky. [6] Tento problém lze částečně omezit tím, že požádáme prostřednictvím jazyka SQL, aby se před zobrazením jednotlivé řádky tabulky seřadili. S původním uspořádáním řádku v tabulce to nemá ale nic společného. Počet řádků v tabulce je libovolný. [6]

Prázdnou tabulkou nazýváme tabulku, která neobsahuje žádný řádek. Prázdná tabulka nemá žádná data, ale svoji strukturu má definovanou svými sloupci, které obsahuje. Většina databázových systémů dovoluje tabulce růst, tedy přidávat sloupce, do té doby než spotřebuje volné místo na disku počítače. Jiné databázové systémy vymezují maximální limit, ale ten je většinou štědrý – standardně něco kolem dvou miliard řádků nebo i více.

### 1.2.3 Relace

V relačních databázích nejsou dovoleny explicitní ukazatelé jako relace rodič-potomek hierarchické databáze, což je jeden ze základních a zároveň z hlavních rozdílů mezi relačním modelem a dřívějšími datovými modely. [6] Nicméně tyto relace existují i v relačních databázích. Například v ukázkové databázi (obrázek 4) je vidět, že relace je mezi řádky tabulky „*POBOCKY*“ a řádky tabulky „*PRODEJCI*“. Každý prodejce v tabulce se přiřadí k určité prodejní kanceláři a tím vzniká daná relace.



Obrázek 4 – Relace rodič – potomek v relační databázi [6]



Relace rodič-potomek mezi prodejní kanceláři a lidmi, kteří v ní pracují, se v relačním modelu neztratí, [6] pouze explicitními ukazateli umístěnými v databázi není zastoupena. Relace je místo toho reprezentována pomocí společných datových hodnot, které se ukládají ve dvou tabulkách. V relačních databázích, se stejným způsobem reprezentují všechny relace.

### 1.3 Jak relační databáze fungují

Relační databáze pracuje na principu seskupení jednotlivých tabulek. [1] Všechny tabulky v databázi nemusí být ve vzájemném vztahu. Skupina tabulek ve vzájemném vztahu se nazývá schéma. Každá databáze může mít různý počet schémat. Původně (v SQL89) pod vedením jednoho uživatele byly všechny tabulky automaticky ve stejném schématu. Tuto situaci zvrátil až SQL92 standart, ale stále můžeme narazit na starou verzi.

Na druhé straně se díváme na databáze z hlediska uživatelů. Operační systém, který využíváme na nezávislém osobním počítači nebo na přídavném systému, nemusí rozpoznávat jednotlivé uživatele. Evidování více uživatelů v dnešní době umožňuje většina současných systémů. S více uživateli umí pracovat obdobně také většina DBMS. Přitom databázový uživatel nemusí odpovídat uživateli operačního systému. Konkrétní uživatel operačního systému v rámci databáze se může zapojit pod více uživatelskými jmény najednou. To potom vede k tomu, že databázoví uživatelé mají následující vlastnosti totožné s uživateli operačního systému: [1]

- Jsou dáni přihlašovací procedurou, [1] která obvykle obsahuje jméno uživatele a heslo.
- Po přihlášení, zahajují relaci (občas nazýváno termíny „spojení“ nebo „sezení“) s DBMS. [1] Dokud se neukončí, tato relace nepřetržitě běží. V rozsahu relace, se tvoří posloupnost příkazů. Nezávislé posloupnosti, vytváří příkazy od souběžných relací nebo příkazy od shodných nebo různých uživatelů.
- Mají skupinu práv, určující co smí a nesmí dělat. [1]
- Databázový uživatel obdobně jako uživatel operačního systému má určitou množinu souborů a adresářů a může mít i povolení přístupu k jiným. [1] Taktéž obvykle jedno schéma vlastní databázový uživatel a může mít přístup k jiným. Jestliže databázový uživatel má jedno nebo více schémat, závisí na konkrétním databázovém systému, který použil. Bez hlediska na to, který uživatel operačního systému aplikaci zahájil, se

jako databázový uživatel, často chová aplikace, která musí přistupovat ke schématu. To pak dané věci usnadňuje.

## 1.4 Vzorové databázové schéma

Jako vzorové databázové schéma jsem zvolila tabulky 1, 2, 3, které vytvářejí jednoduchou relační databázi. K jednoduchému pozorování, je velmi primitivní [1], ale k vyobrazení nejdůležitější pojmů a technik používaných v SQL dostatečně komplikovaná.

V prvním sloupci každé tabulky si můžeme povšimnout, že obsahuje číslo s hodnotami, které je odlišné pro každý řádek tabulky. Tyto jedinečná čísla nazýváme primární klíč tabulky. [1] Pokud se některé z těchto čísel objevuje ve sloupcích jiných tabulek, nazývá se cizí klíč, který se vždy odkazuje na primární klíč. Cizí klíč, který se odkazuje, na primární klíč nemusí mít stejné jméno. Z důvodu názornosti jsme pro tuto konvenci zvolili stejné jméno, jak pro cizí klíč tak i primární klíč.

**Tabulka 1 – „Salespeople“ [1]**

Snum	Sname	City	Comm
1001	Peel	London	.12
1002	Serres	San Jose	.13
1004	Motika	London	.11
1007	Rifkin	Barcelona	.15
1003	Axelrod	New York	.10

**Tabulka 2 – „Customers“ [1]**

Cnum	Cname	City	Rating	Snum
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	NULL	1001
2008	Cisneros	San Jose	300	1007
2007	Pereira	Rome	100	1004

Tabulka 3 – „Orders“ [1]

Onum	Amt	Odate	Cnum	Snum
3001	18.69	10/03/2000	2008	1007
3003	767.19	10/03/2000	2001	1001
3002	1900.10	10/03/2000	2007	1004
3005	5160.45	10/03/2000	2003	1002
3006	1098.16	10/03/2000	2008	1007
3009	1713.23	10/04/2000	2002	1003
3007	75.75	10/04/2000	2004	1002
3008	4723.00	10/05/2000	2006	1001
3010	1309.95	10/06/2000	2004	1002
3011	9891.88	10/06/2000	2006	1001

Jak si můžeme, všimnou, ve výše uvedeném návrhu nejsou napsány podrobnější informace o prodejci a zákaznících, jako např. celé jméno či kontaktní údaje apod. [1] V reálné databázi, by se určitě tyto informace vyskytovaly. Abychom naše tabulky uchovali co nejjednodušší, tak jsme tyto informace schválně vynechali, ale přitom jsme postupovali tak, abychom neznemožnili porozumění příkladů.

Dále jsou zákazníci přiřazeni k prodejcům, a jednotlivé objednávky jsou sloučeny jak s prodejcem, tak se zákazníkem. [1] Je potřeba aby se v tabulce „Orders“ jednotlivý odkaz shodoval se záznamem v tabulce „Customers“. Každý prodejce, který je přiřazený k zákazníkovi, dostane každou zakázku tohoto zákazníka. Pokud předpokládáme ano, pak by sloupec „Snum“ v tabulce „Orders“ byl nadbytečný, proto předpokládáme, že ne. Prodejce může uskutečnit prodej, v případě když je jiný prodejce přiřazený zákazníkovi nedostupný. Oba prodejci si pak můžou rozdělit provizi. Pomocí tzv. aplikačních pravidel se může tato komplexnost uvést do databáze, kterou je lepší nezastírat.

Provize v tabulce „Salespeople“ nemusí být dána na pevno [1], může být například z ročních prodejů periodicky počítána. Určité problémy se mohou objevit, v případě že do databáze ukládáme provizi na pevno. Protože výpočet provize je většinou velmi složitý, v takovém případě se pak používá jen výjimečně. Nicméně ale i jiný přístup může být vhodný.

## 1.5 Shrnutí

V této kapitole jsme se zaměřili na to, co znamená, když uslyšíme pojem „relační databáze“. I když databáze ve skutečnosti se nám zdá velice složitá, je to pojem, který jak teď už víme, není tak složitý jak se zdá. Pochopili jsme také některé základní principy relační databáze. Poznali jsme, že tabulky jsou konstruovány pomocí sloupců a řádků. Zjistili jsme, jak se jednotlivé sloupce odkazují na hodnoty jiných sloupců, také co je primární a cizí klíč, a jak primární klíče odlišují jednotlivé řádky od ostatních a na závěr jsme se seznámili, jak vypadají jednotlivé tabulky (vzorové tabulky).

## 2 JAZYK SQL

Největší světové softwarové firmy jako jsou například Microsoft, Oracle a IBM [1], využívají jazyk SQL jako základ databázových produktů. Jazyk SQL je také srdcem všech databází nazývaných „open-source“, které pomáhají silnému růstu operačního systému Linux a hnutí „open-source“.

### 2.1 Co je jazyk SQL

Jazyk SQL slouží jako nástroj pro správu, organizování, a získávání dat, která jsou uložena v počítačových databázích. [6] Zkratka SQL znamená strukturovaný databázový jazyk, v anglickém jazyce, pak Structured Query Language. Obecně se dá říct, že jazyk SQL je počítačový jazyk, pro komunikaci s databází. Ve skutečnosti SQL pracuje s relační databází, pracuje tedy s jedním specifickým typem databáze.

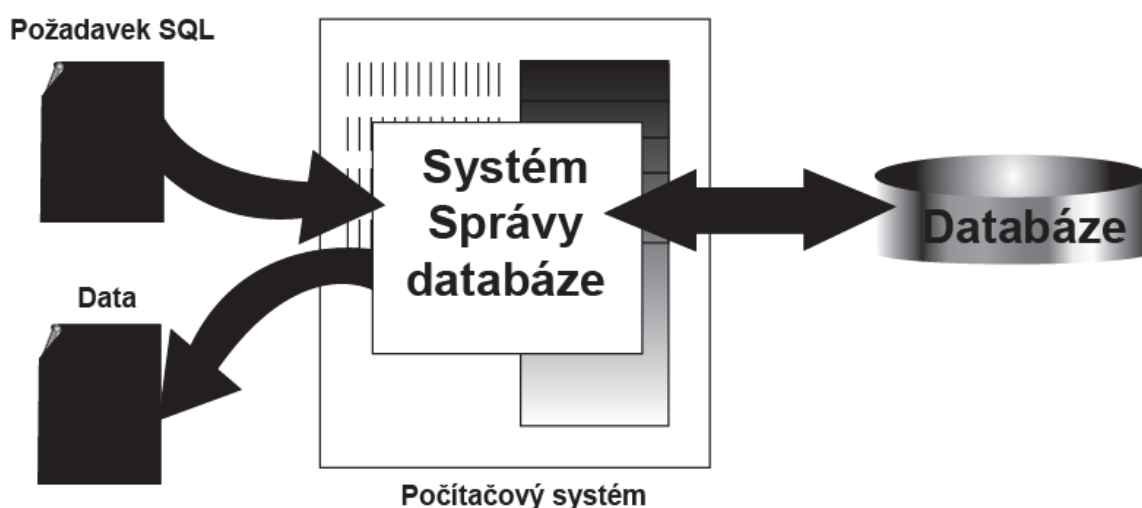
Jazyk SQL sám o sobě zahrnuje více funkcí než samotný strukturovaný dotazovací jazyk, [6] i když to byl původně jeho účel. I přesto mezi jeho nejdůležitější funkce stále patří získávání dat. Jazyk SQL lze také použít k řízení všech funkcí, které databázový systém svým uživatelům nabízí.

- **Definice dat** – SQL umožňuje uživatelům vytvářet strukturu a organizaci všech uložených dat
- **Získávání dat** – SQL dovoluje jak uživatelům nebo aplikačnímu programu, s uloženými daty pracovat tak i uložená data z databáze získávat.
- **Manipulace s daty** – SQL dovoluje uživateli tak i aplikačnímu programu obnovovat databázi přidáváním nových dat, změnou dříve uložených dat nebo odstraňováním starých dat databáze.
- **Řízení přístupu** – SQL lze použít k vyhrazení schopnosti uživatele data číst, doplňovat a modifikovat. Toto lze použít k ochraně dat před neautorizovaným přístupem.
- **Sdílení dat** – SQL se používá k vyladění dat mezi více uživateli a k ochraně, aby se uživatelé mezi sebou navzájem nerušili.
- **Integrita dat** – SQL formuluje v databázi mez integrity, aby ochránil danou databázi před porušením, které vzniká nesprávnou aktualizací (neúplnou) nebo systémovým selháním.

SQL je tedy souhrnný jazyk pro vedení a ovlivňování databázových systémů, se kterými pracujeme. [6]

## 2.2 Úloha jazyka SQL

Jazyk SQL sám o sobě, není databázový řídicí systém, ani se nejedná o samotný produkt. Samotný jazyk SQL si nemůžeme zakoupit v obchodě. SQL se bere jako jazyk a nástroj pro komunikaci s databázovým systémem [6] a přitom SQL je jeho integrovanou součástí. Na obrázku 5 můžeme vidět vybrané části standardního databázového systému. Podle obrázku 5, je také očividně vidět, že SQL plní roli „lepidla“, který spojuje jednotlivé komponenty.

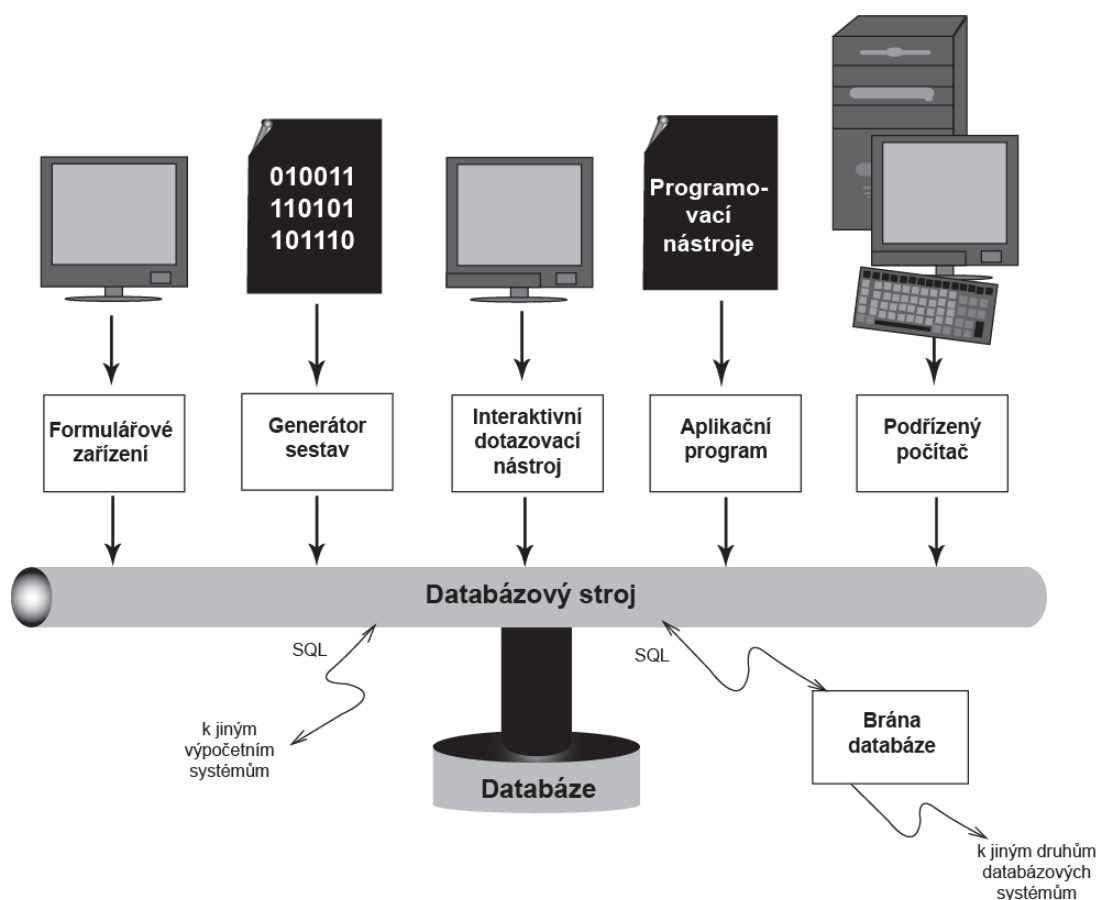


Obrázek 5 – Použití jazyka SQL pro přístup k databázi [6]

Srdce řídicího systému tvoří *databázový stroj*, který zodpovídá za vlastní segmentování, ukládání a získávání dat z databáze. Z počítačových systémů, z uživatelských aplikačních programů, nebo z různých částí systémů (formuláře, sestavy, atd.) přijímá požadavky jazyka SQL. Jazyk SQL, obsahuje několik různých funkcí: [6]

- SQL je *interaktivní dotazovací jazyk*. [6] Příkazy SQL uživatelé zapisují prostřednictvím konverzačního programu a získaná data zachycují na obrazovce.

- SQL je *databázový programovací jazyk*. Pro získání přístupu k datům v databázi, programátoři jednotlivé příkazy SQL zapojují do svých aplikačních programů. [6] Tato metoda je vhodná jak pro uživatelsky psané programy tak i pomocné databázové programy.
- SQL je *administrační databázový jazyk*. [6] Jazyk SQL k vymezení databázové struktury a k řízení přístupu k uloženým datům využívá administrátor zodpovědný za správu minipočítačové databáze nebo databáze na střediskovém počítači.
- SQL je *jazyk aplikací typu klient/server*. U podnikových aplikací, architektura klient/server patří mezi velmi oblíbené. [6] Ke komunikaci s databázovými servery, které jsou situované na síti, používají jazyk SQL, programy které se spouští na osobním počítači.
- SQL je *jazyk pro přístup k datům na Internetu*. Jazyk SQL se používá jako standardní jazyk pro vstup ke společným databázím. [6] Využívají ho internetové webové servery, pracující se společnými daty a internetové aplikační servery.
- SQL je *distribuovaný databázový jazyk*. [6] Data mezi jednotlivými propojenými počítačovými systémy se rozmísťují a k tomu jim pomáhá jazyk SQL. Pro vzájemnou komunikaci mezi všemi počítačovými systémy, zasíláním požadavků pro vstup k datům, využívá software databázového systému jazyk SQL.
- SQL je *jazyk pro databázové brány*. Komunikaci jednoho druhu databázového systému s jiným databázovým systémem umožňuje *brána*. [6] Jazyk SQL je často používán jako brána, která je tvořena v počítačové síti směsí různých databázových produktů.



Obrázek 6 – Části typického databázového řídicího systému [6]

### 2.3 Jak SQL pracuje

SQL jazyk se zaměřuje převážně na relační databáze. Kdybychom psali v obecně používaném jazyce jako je např. C jazyk, museli bychom udělat mnohem více práce. V jazyce C, bychom museli vytvořit vše od samého začátku. Vytvořit relační databázi, přesně určit objekt – tabulku, která by se mohla rozrůstat do libovolného počtu řádku. Vytvořit procedury pro ukládání a výběr dat. Pro výběr jednoho konkrétního řádku, bychom museli provést kroky podle následujícího postupu: [1]

1. Podívat se na řádek tabulky.
2. Provést zkoušku, zda se jedná o správný řádek, který nás zajímá.
3. Pokud je to správný řádek, uložit jej někam a pokračovat dokud není prohlédnuta celá tabulka.



4. Zkontrolovat, zda jsou ještě další řádky v tabulce.
5. Když se vyskytuje více takových řádků, skočit zpět na krok 1.
6. Jestliže byla prohlédnuta celá tabulka, provést výstup jednotlivých řádků, které byly uloženy v kroku 3.

Toho všeho nás ale jazyk SQL ušetří. Příkazy neboli povely v jazyce SQL jsou schopny pracovat s celými tabulkami, [1] jako by se jednalo o jednoduché objekty. Libovolné množství informací vybrané nebo odvozené z těchto objektů jsou schopny také zpracovat.

## 2.4 Rysy a výhody jazyka SQL

SQL patří mezi jednoduchý a snadno pochopitelný jazyk a současně souhrnný nástroj pro správu dat. Mezi některé z hlavních rysů jazyka SQL a obchodní úspěchy patří: [6]

- nezávislost na prodejci;
- přenositelnost mezi počítačovými systémy;
- standardy jazyka SQL;
- podpora a angažovanost firmy IBM;
- angažovanost firmy Microsoft (SQL Server, ODBC a ADO);
- relační základy;
- vysokoúrovňová struktura podobná angličtině;
- interaktivní dotazy, dotazy „ad hoc“;
- programovatelný přístup k databázi;
- vícenásobné pohledy na data;
- kompletní databázový jazyk;
- dynamické definování dat;
- architektura klient/server;
- podpora podnikových aplikací;
- rozšiřitelnost a objektová technologie;
- internetový databázový přístup;

- integrace s jazykem Java;
- průmyslová infrastruktura.

Toto jsou důvody, proč je jazyk SQL standardním nástrojem správy dat na osobních počítačích, minipočítačích a sálových počítačích. [6]

## 2.5 Shrnutí

Hlavním záměrem této kapitoly bylo, rychle a srozumitelně popsat jazyk SQL, tak abychom si vytvořili představu o tom, jak jazyk SQL vypadá a pracuje. Vysvětlili jsme si, jak daný SQL pracuje a jakou má úlohu při vytváření databáze, jaké má nejzákladnější rysy a na závěr také jaké má výhody.

### 3 PRÁCE S DATY V DATABÁZÍCH

V jazyku SQL jsou v mnoha směrech základem jednotlivé dotazy. [6] Mezi základní příkazy pro manipulaci s daty zahrnujeme: [5]

- SELECT – pro dotazy na data v databázi;
- INSERT – vkládání dat do tabulky;
- UPDATE – pro aktualizaci dat v tabulce;
- DELETE – pro vymazání dat z tabulky.

Příkaz SELECT, patří mezi nejvýkonnější a současně nejsložitější příkazy jazyka SQL, [6] který vyhledává data v databázi a vrací je v podobě výsledků dotazu. Příkaz SELECT má mnoho podob, je tedy možné v databázi nejdříve začínat jednoduchými a postupně se propracovat ke složitějším příkazům.

#### 3.1 Jednoduché dotazy - SELECT

Úkolem příkazu SELECT je vyhledat a zobrazit data z jedné nebo více tabulek. Patří mezi silné a zároveň nejpoužívanější příkazy jazyka SQL. [5] Základní forma zápisu příkazu SQL je následující.

```
SELECT [DISTINCT | ALL] { * | [sloupcovýVýraz [ AS novéJméno]]  
    [...]}  
FROM jménoTabulky [alias] [...]  
[WHERE podmínka]  
[GROUP BY seznamSloupců] [HAVING podmínka]  
[ORDER BY seznamSloupců]
```

Obrázek 7 – Základní forma zápisu příkazu SELECT

kde:

- *sloupcovýVýraz* reprezentuje jméno daného sloupce nebo výrazu;
- *novéJméno* je jméno, které se potom u daného sloupce zobrazuje v záhlaví;
- *jménoTabulky*, je jméno tabulky, která se nachází v dané databázi;
- *alias*, je libovolná zkratka pro *jménoTabulky*; [5]

- *vertikální oddělovač* “/” určuje volbu výběru mezi jednotlivými variantami, např. Připojena klíčová slova DISTINCT nebo ALL ke klauzuli SELECT;
- *povinný prvek*, určují složené závorky „{ }“;
- *volitelný prvek*, [5] určují hranaté závorky, např. volitelná klauzule WHERE

V příkazu SELECT je uspořádání zpracování následující: [5]

- SELECT – specifikuje sloupce, které se mají objevit ve výstupu;
- FROM – udává tabulku nebo tabulky, které se mají použít;
- WHERE – podle dané podmínky, třídí řádky;
- GROUP BY – s totožnou hodnotou sloupce utváří jednotlivé řádky;
- HAVING – podle dané podmínky, třídí skupiny;
- ORDER BY – udává řazení výstupu.

V příkazu SELECT, je pořadí klauzulí přesně dané, a nelze ho měnit. [5] Klauzule dělíme na povinné a nepovinné. Povinné klauzule jsou pouze dvě a to SELECT A FROM, ostatní jsou nepovinné.

#### **3.1.1.1 Vyzvání všech sloupců a všech řádků**

Klauzule WHERE není potřeba, [5] protože v dotazu nejsou zavedena žádná omezení. Proto výraz, který má vypsát všechny údaje o všech DVD, vypadá následovně:

<pre>SELECT catalogNo, title, genre, rating FROM DVD;</pre>
---

**Obrázek 8 – Výraz pro výpis údajů o všech DVD [5]**

V případě, že chcete zobrazit všechny sloupce tabulky, lze použít (\*), která nahradí názvy jednotlivých sloupců. Vypadá tedy následovně:

```
SELECT *
FROM DVD;
```

Obrázek 9 – Výraz pro výpis všech sloupců tabulky [5]

V obou případech získáme následující výpis tabulky:

Tabulka 4 – Výsledná tabulka pro dotaz na obrázku 8 a 9 [5]

catalogNo	title	genre	rating
207132	Casino Royale	Action	PG-13
902355	Harry Potter and the GOF	Children	PG
330553	Lord of the Rings	Action	PG-13
781132	Shrek2	Children	PG
445624	Million Impossible III	Action	PG-13
634817	War of the Worlds	Sci-fi	PG-13

### 3.1.2 Vyvolání zadaného sloupce, všech řádků

I v tomto případě, zde nejsou určena žádná omezení, a proto i zde nebudeme uvádět klauzuli WHERE. [5] Protože chceme zobrazit seznam katalogových čísel, titulů a žánrů všech DVD, což si můžeme představit jako seznam podmnožiny sloupců a tu vyjádříme následovně:

```
SELECT catalogNO, title, genre
FROM DVD;
```

Obrázek 10 – Výraz pro výpis jednotlivých sloupců z tabulky [5]

Výsledek dotazu ukazuje tabulka 5. Pokud se podíváte pozorněji, jednotlivé řádky ve výsledné tabulce nemusí být seřazené. (5)

Tabulka 5 – Výsledná tabulka pro dotaz na obrázku 10 [5]

catalogNo	title	genre
207132	Casino Royale	Action
902355	Harry Potter and the GOF	Children
330553	Lord of the Rings	Action
781132	Shrek2	Children
445624	Million Impossible III	Action
634817	War of the Worlds	Sci-fi

### 3.1.3 Použití DISTINCT

Pokud chcete vypsat např. všechny žánry DVD, pak příkaz vypadá následovně: [5]

```
SELECT genre  
FROM DVD;
```

Obrázek 11 – Výraz pro výpis všech žánrů z tabulky [5]

Výsledek dotazu ukazuje tabulka 6. Povšimněte si, že v tabulce se nachází několik duplicitních hodnot. [5] Pokud chcete odstranit tyto duplicitní hodnoty, musíte spolu s příkazem SELECT použít klíčové slovo DISTINCT. Pak následující příkaz bude vypadat takto:

```
SELECT DISTINCT genre  
FROM DVD;
```

Obrázek 12 – Výraz pro odstranění všech duplicitních hodnot [5]

Potom výsledná tabulka bez duplicitních hodnot je tabulka 7.

Tabulka 6 – Výsledná tabulka pro dotaz na obrázku 11 s duplicitami [5]

genre
Action
Children
Action
Children
Action
Sci-fi

Tabulka 7 – Výsledná tabulka pro dotaz na obrázku 12 po odstranění duplicit [5]

genre
Action
Children
Sci-fi

## 3.2 Výběr řádků – klauzule WHERE

V předchozích kapitolách, jsme měli příkazy bez omezení. Často, se ale setkáváme s tím, že potřebujeme omezit výběr určitou podmínkou. Tohoto lze, dosáhnout pomocí klauzule WHERE, která se skládá z klíčového slova WHERE a za ním následuje podmínka, která určuje omezení pro výběr dat z tabulky. Pro klauzuli WHERE máme pět základních vyhledávacích podmínek: [5]

- Porovnání;
- Rozsah;
- Náležení do množiny;
- Srovnání vzoru;
- Null (prázdná hodnota).

### 3.2.1 Vyhledávací podmínky porovnání

Pokud chcete např. vypsát všechny zaměstnance s platem vyšším než 40 000\$. [5]

Použijeme příkaz s omezením:

```
SELECT staffNo, name, position, salary  
FROM Staff  
WHERE salary > 40000;
```

Obrázek 13 – Výraz pro výpis všech zaměstnanců s vyšším platem než je 40 000\$ [5]

Z tabulky „Staff“, je potřeba vybrat řádky, kde hodnota, která se nachází ve sloupci „salary“ je vyšší než 40 000\$. [5] Docílíme toho, když použijeme klauzuli WHERE s podmínkou „salary > 40000“. Výsledek pak znázorňuje tabulka 8.

Tabulka 8 – Výsledná tabulka pro dotaz na obrázku 13 [5]

staffNo	name	position	salary
S1500	Tom Daniels	Manager	48000
S0010	Mary Martinez	Manager	51000
S2250	Sally Stern	Manager	48000
S0415	Art Peters	Manager	42000

Poznámka:

K dispozici jsou následující operátory porovnání: [5]

- rovná se „=“ nerovná se „<>“;
- je menší než „<“, je menší nebo rovno „<=“;
- je větší „>“, je větší nebo rovno „>=“.

Pravidla pro vyhodnocování těchto výrazů jsou následující: [5]

- výraz se vyhodnocuje zleva doprava;
- část výrazu v závorkách se vyhodnocuje jako první;
- NOT se vyhodnocuje jako AND a OR;
- AND se vyhodnocuje před OR.

Aby nedocházelo k dvoujazyčnosti, je lepší používat závorky mezi jednotlivými výrazy. [5]

### 3.2.2 Vyhledávací podmínka rozsahu (BETWEEN / NOT BETWEEN)

Vyhledávací podmínka BETWEEN slouží k vyhledávání určitého rozmezí hodnot. [5]

Například pokud chcete vypsát všechny zaměstnance, kteří mají plat v rozmezí 45 000\$ a 50 000\$, pak:

```
SELECT staffNo, name, position, salary
FROM Staff
WHERE salary >= 45000 AND salary <= 50000
```

**Obrázek 14 – Výraz pro výpis všech zaměstnanců s platem v rozmezích 45 000 – 50 000\$ [5]**

Z tabulky „Staff“ k získání řádků, kde hodnota sloupce „salary“ se nachází v rozmezí 45000\$ a 50000\$, jsme v tomto dotazu využili logický operátor a klauzuli WHERE.

**Tabulka 9 – Výsledná tabulka pro dotaz na obrázku 14 [5]**

staffNO	name	position	salary
S1500	Tom Daniels	Manager	48000
S2250	Sally Stern	Manager	48000



Výsledek znázorňuje tabulka 9. Místo logického operátoru AND a klauzule WHERE, lze také použít pro vyhledání určitého rozsahu hodnot příkaz BETWEEN. [5] Předchozí dotaz tedy můžeme přepsat následovně:

```
SELECT staffNo, name, position, salary
FROM Staff
WHERE salary BETWEEN 45000 AND 50000;
```

Obrázek 15 – Výraz pro testování rozsahu hodnot pomocí BETWEEN[5]

Vyskytuje se také negovaná verze testování rozsahu (NOT BETWEEN), která je podobná výrazu BETWEEN. [5] Jediný rozdíl je ten, že vyhledává hodnoty mimo zadaný rozsah.

### 3.2.3 Vyhledávací podmínka náležení do množiny (IN / NOT IN)

Pro testování zda daná hodnota souhlasí s některou hodnotou ze zadaného rozsahu, se používá klíčové slovo IN. Např. pokud chcete vypsat všechna DVD žánrů „sci-fi“ nebo „Children“ pak dotaz vypadá následovně: [5]

```
SELECT catalogNo, title, genre
FROM DVD
WHERE genre IN ('Sci-Fi', 'Children');
```

Obrázek 16 – Výraz pro výpis všech DVD „sci-fi“ nebo „Children“ [5]

Pro klíčové slovo IN, existuje i negovaná verze NOT IN, která se používá při vybírání dat, které neleží v zadaném seznamu hodnot. [5] Ve srovnání s BETWEEN, je klíčové slovo IN efektivnějším způsobem pro vyjádření vyhledávací podmínky.

Tabulka 10 – Výsledná tabulka pro dotaz na obrázku 16 [5]

catalogNO	title	genre
902355	Harry Potter and the GOF	Children
781132	Shrek 2	Children
634817	War of the Worlds	Sci-fi

### 3.2.4 Vyhledávací podmínka porovnávání se vzorem (LIKE / NOT LIKE)

Pro porovnání jazyk SQL používá dva speciální symboly.

Sekvenci nula nebo více znaků neboli divokou kartu, představuje „%“ znak procenta. Jeden libovolný znak představuje „\_“ znak podtržítka.

Ostatní používané znaky, představují samy sebe. Například:

- **name LIKE 'S%'** znamená, že první znak musí být S.
- **name LIKE 'S\_\_\_\_\_'** znamená, že řetězec musí mít přesně šest znaků a první musí být „S“.
- **name LIKE '%S'** znamená, že posloupnost může mít libovolný počet znaků, ale musí končit znakem S.
- **name LIKE '%Sally%'** znamená, že posloupnost o libovolné délce, musí obsahovat slovo Sally.
- **name NOT LIKE 'S%'** znamená, že první znak nesmí být S.

Pokud chceme aby vyhledávací řetězec přímo obsahoval speciální znaky, lze využít pro jejich zápis znak escape. Například pro vyhledání řetězce „15%“, lze použít predikát: *„LIKE '15#%' ESCAPE '#'“*.

Pomocí SQL dotazu a podmínky porovnání podle vzoru najdeme všechny zaměstnance, jejichž křestní jméno je „Sally“:

```
SELECT staffNo, name, position, salary
FROM Staff
WHERE name LIKE 'Sally%'
```

Obrázek 17 – Dotaz porovnání podle vzoru [5]

Výsledek dotazu ukazuje následující tabulka 11

Tabulka 11 – Výsledná tabulka pro dotaz na obrázku 17 [5]

staffNo	name	position	salary
<b>S0003</b>	Sally Adams	Assistant	30000
<b>S2250</b>	Sally Stern	Manager	48000

Poznámka:

Některé programy např. Microsoft Access, používá jako divoké karty znaky „\*“ a „?“ místo znaků „%“ a „\_“.

### 3.2.5 Vyhledávací podmínka hodnoty (IS NULL / IS NOT NULL)

V případě, že chcete vypsát všechny půjčky, které nemají zadané datum vrácení, tedy mají prázdnou (nulovou) hodnotu. [5] Pak tabulka „*DVDRental*“ obsahuje sloupec „*dateReturn*“, v němž se nachází datum vrácení, pro jednotlivé půjčky. Každý by řekl, že půjčky vyhledáme pomocí následujícího výrazu:

**WHERE (dateReturn = ‘ ‘ OR dateReturn = 0);**

Hodnota *null* v „*dateReturn*“ znamená neznámou hodnotu. [5] V tom případě by žádná podmínka nefungovala, protože nelze testovat, jestliže se hodnota *null* rovná jiné hodnotě. Výsledkem by při zavolání příkazu SELECT s těmito podmínkami, byla prázdná tabulka. Je proto potřeba pomocí specifického klíčového slova IS NULL testovat hodnotu *null*.

```
SELECT deliveryNO, DVDNo
FROM DVDRental
WHERE dateReturn IS NULL;
```

Obrázek 18 – Výraz pro testování hodnoty NULL [5]

Výsledek dotazu ukazuje následující tabulka 12. Vyskytuje se také i negovaná verte IS NOT NULL, která se používá pro vyhledávání hodnot, které se nerovnají *null*.

Tabulka 12 – Výsledná tabulka pro dotaz na obrázku 18 [5]

deliveryNo	DVDNo
R66818964	17864331

## 3.3 Řazení výsledků – klauzule ORDER BY

Běžně řádky, ve výsledné tabulce SQL dotazu nejsou řazené. Řazení výsledků můžeme docílit, pomocí klauzule ORDER BY v příkazu SELECT. [5] Obvykle je klauzule ORDER BY složena ze jmen sloupců, podle nichž je daný výsledek obvykle řazen. Vybrané řádky mohou být řazené vzestupně (ASC) nebo sestupně (DESC), podle

jakéhokoliv sloupce nebo kombinace sloupců. Sloupce, podle kterého se řadí jednotlivé řádky, nemusí být ve výsledném zobrazení. Klausule ORDER BY, musí být v každém případě vždy poslední klauzulí příkazu SELECT.

### 3.3.1 Uspořádání výsledků

Výpis všech DVD seřazených postupně podle žánru.

```
SELECT *
FROM DVD
ORDER BY genre DESC;
```

Obrázek 19 – Dotaz pro uspořádání výsledků[5]

Tabulka 13 – Výsledná tabulka pro dotaz na obrázku 19 s řazením podle „genre“ [5]

catalogNo	title	genre	rating
634817	War of the Worlds	Sci-fi	PG-13
902355	Harry Potter and the GOF	Children	PG
781132	Shrek 2	Children	PG
207132	Casino Royale	Action	PG-13
330553	Lord of the Rings III	Action	PG-13
445624	Mission Impossible III	Action	PG-13

Tabulka 14 – Výsledná tabulka pro dotaz na obrázku 19 s řazením podle „genre“ a „catalogNo“ [5]

catalogNo	title	genre	rating
634817	War of the Worlds	Sci-fi	PG-13
781132	Shrek 2	Children	PG
902355	Harry Potter and the GOF	Children	PG
207132	Casino Royale	Action	PG-13
330553	Lord of the Rings III	Action	PG-13
445624	Mission Impossible III	Action	PG-13

Výsledek dotazu je podobný výsledku v tabulce 4. Tentokrát jsme, ale použili příkaz s požadavkem, na seřazení výsledné tabulky podle hodnot ve sloupci „genre“. Toho dosáhneme, když použijeme klauzuli ORDER BY na konec příkazu SELECT, a tím se

„*genre*“ specifikuje jako sloupec podle, kterého se bude řadit, a DESC slouží k tomu, aby řazení bylo sestupné. Výsledek dotazu ukazuje tabulka 13.

Protože ve sloupci „*genre*“ se mnoho hodnot opakuje, můžeme chtít seřadit výsledek nejdřív podle „*genre*“, což bude hlavní klíč řazení a za druhé vzestupné uspořádání podle „*CatalogNo*“, což bude vedlejší klíč řazení. V tom případě bude klauzule ORDER BY vypadat následovně:

ORDER BY *genre* DESC, *CatalogNo* ASC;

Obrázek 20 – Alternativní řazení

Výsledek dotazu na obrázku 20 pro alternativní řazení zobrazuje tabulka 14.

### 3.3.2 Použití agregačních funkcí SQL

COUNT – vrací počet hodnot v zadaném sloupci

SUM – vrací součet hodnot v zadaném sloupci

AVG – vrací průměr hodnot v zadaném sloupci

MIN – vrací nejmenší hodnotu ze zadaného sloupce

MAX – vrací největší hodnotu ze zadaného sloupce

Tyto základní funkce pracují s jednotlivými sloupci a vrací jednoduchou hodnotu. [5] Pro numerické a nenumерické pole lze použít funkce COUNT, MIN a MAX. SUM a AVG, lze použít pouze pro numerické pole. Všechny funkce nejdříve eliminují hodnoty *null* a pracují jen s neprázdnými hodnotami, zatímco COUNT (\*) je speciální použití příkazu COUNT, které bez ohledu na to zda řádky tabulky obsahují hodnoty *null* nebo ne, spočítá všechny řádky tabulky.

#### 3.3.2.1 Použití funkce SUM

Funkce SUM se používá pro součet jednotlivých hodnot v tabulce. [5] Pokud chcete např. vypsát celkový počet všech zaměstnanců s platem vyšším než 40 000\$ použijeme, následující dotaz:

```
SELECT COUNT (staffNo) AS totalStaff, SUM(salary) AS totalSalary  
FROM Staff  
WHERE salary > 40000;
```

Obrázek 21 – Výraz pro vyjádření počtu všech zaměstnanců s platem vyšším než 40 000\$ [5]

V tomto dotazu jsme použili funkci COUNT na počet řádků, a zároveň jsme použili klauzuli WHERE a funkci SUM, pro součet všech platů v jednotlivých řádcích tabulky. [5] Výsledek dotazu ukazuje tabulka 15.

Tabulka 15 – Výsledná tabulka pro dotaz na obrázku 21 [5]

totalStaff	totalSalary
4	18900

### 3.3.2.2 Použití funkcí MIN, MAX a AVG

Funkce MIN, MAX a AVG se používají pro výpis nejmenší, největší a průměrné hodnoty. [5] Pokud chcete např. vypsát nejmenší, největší a průměrný plat zaměstnanců, pak použijete, následující dotaz:

```
SELECT MIN(salary) AS minSalary, MAX(salary) AS maxSalary,  
       AVG(salary) AS avgSalary  
FROM Staff;
```

Obrázek 22 – Výraz pro výpis nejmenšího, největšího a průměrného platu zaměstnanců [5]

V tomto příkazu nevyužíváme klauzuli WHERE, protože uvažujeme všechny řádky se zaměstnanci. [5] Nejmenší, největší a průměrná hodnota se vypočítá pomocí funkcí MIN, MAX a AVG. Výsledek dotazu znázorňuje tabulka 16.

Tabulka 16 – Výsledná tabulka pro dotaz na obrázku 22 [5]

minSalary	maxSalary	avgSalary
30000	51000	42000

### 3.4 Seskupení výsledků – klauzule GROUP BY

Mnohem efektivnější je mít ve výsledku jednotlivé souhrny. Pro dosažení seskupení výsledků použijeme, klauzuli GROUP BY v příkazu SELECT. [5] Termínem „*Seskupený dotaz*“ se nazývá dotaz, v něm se nachází, klauzule GROUP BY. Dotaz seskupí data z tabulky v příkazu SELECT tak, že zhotoví jednořádkový souhrn pro každou skupinu. Termínem „*Seskupovací sloupce*“ pak označujeme sloupce, použité v klauzuli GROUP BY. Pro každou skupinu, pokud se použije, klauzule GROUP BY, musí mít každá položka v seznamu SELECT pouze jednu hodnotu. Klauzule SELECT může obsahovat pouze tyto položky: [5]

- jména sloupců,
- agregační funkce,
- konstanty,
- výraz obsahující kombinaci předchozích položek.

V klauzuli GROUP BY, pokud není jejich jméno použito v agregační funkci, se musí nalézt všechny sloupce v příkazu SELECT. Naopak to ale neplatí. [5] V případě, že použijete i klauzuli WHERE, užívá se jako první WHERE a pak se utvoří skupina ze zbývajících řádků, které splňují danou podmínku.

#### 3.4.1 Použití GROUP BY

Pokud chcete např. zjistit počet zaměstnanců pro každé distribuční centrum a součet jejich platů. Použijete následující dotaz: [5]

```
SELECT dCenterNo, COUNT(staffNo) AS totalStaff, SUM(salary)
      AS totalSalary
FROM Staff
GROUP BY dCenterNo
ORDER BY dCenterNo;
```

Obrázek 23 – Výsledný dotaz s použitím GROUP BY [5]

V příkazu SELECT se sloupce „salary“ a „staffNo“ nachází jen v agregačních funkcích, nemusí je tedy zahrnout, do seznamu GROUP BY. [5] Na rozdíl, ale sloupec „dCenterNo“, není zahrnutý v agregačních funkcích, je nutné ho tedy, zahrnout do seznamu GROUP BY. Výsledek pak znázorňuje tabulka 17.

Tabulka 17 – Výsledná tabulka pro dotaz na obrázku 23 [5]

dCenterNo	totalStaff	totalSalary
D001	2	78000
D002	2	84000
D003	1	42000
D004	1	48000

Pro lepší pochopení předchozího dotazu jazyka SQL, uvádíme popis, způsobu vyhodnocení dotazu. [5] Graficky je způsob vyhodnocení znázorněn na obrázku 24.

1. Podle distribučního centra sloupec „dCenterNo“, rozčlení dotaz SQL zaměstnance do jednotlivých skupin. Distribuční číslo centra, mají všichni zaměstnanci v každé skupině stejné. Dostaneme tedy čtyři skupiny.
2. Dotaz SQL, aby dostal celkový součet platů, spočítá pro každou skupinu počet zaměstnanců a součet hodnot ve sloupci „salary“.
3. Podle distribučního centra „dCenterNo“, se vzestupně seřadí konečná (výsledná) tabulka.

dCenterNo	staffNo	salary		COUNT(staffNo)	SUM(salary)
D001	S1500	48000	} →	2	78000
D001	S0003	30000			
D002	S0010	51000	} →	2	84000
D002	S3250	33000			
D003	S0415	42000	} →	1	42000
D004	S2250	48000	} →	1	48000

Obrázek 24 – Ukázka rozčlenění distribučního centra do jednotlivých skupin [5]



### 3.4.2 Použití HAVING

Pro omezení skupin, které se mají objevit ve výsledné tabulce se používá klauzule HAVING spolu s klauzulí GROUP BY. [5] Avšak HAVING a WHERE mají podobnou syntaxi, používají se ale k odlišným účelům. Klauzule WHERE filtruje jednotlivé řádky vstupující do tabulky konečných výsledků, zatímco klauzule HAVING filtruje až vytvořené skupiny.

Pokud chcete např. pro každé distribuční centrum s více než jedním zaměstnancem najít počet zaměstnanců a součet jejich platů použijete dotaz: [5]

```
SELECT dCenterNo, COUNT(staffNo) AS totalStaff, SUM(salary)
      AS totalSalary
FROM Staff
GROUP BY dCenterNo
HAVING COUNT (staffNo) > 1
ORDER BY dCenterNo;
```

Obrázek 25 – Ukázka následujícího dotazu s použitím klauzule HAVING [5]

Klauzule HAVING se použije z toho důvodu, že omezení se vztahuje na skupiny, nikoliv např. na jednoho zaměstnance. [5] Výsledek je znázorněn v tabulce 18.

Tabulka 18 – Výsledná tabulka pro dotaz na obrázku 25 [5]

dCenterNo	totalStaff	totalSalary
D001	2	78000
D002	2	84000

## 3.5 Shrnutí

V této kapitole jsme se zaměřili na to, co znamená, když se řekne „manipulace s daty“ jaké pojmy a výrazy si můžeme představit. Pochopili jsme zde také některé základní příkazy, pro manipulaci s daty v databázi. Poznali jsme že, někdy správným výběrem příkazů a správným sestavením může být daný příkaz mnohem rychlejší a efektivnější.

## 4 OPTIMALIZACE SQL DOTAZŮ

Přístupový plán je nedílnou pomůckou při optimalizaci SQL. [1] Znamená to, že podle názoru DBMS, jak by se daný dotaz měl uskutečnit, nám přístupový plán poskytuje určitou analýzu, podle které se pak řídíme. Známe-li lepší způsob, jak dotaz provést, je možné ve většině produktů, udělat změny plánu. Přístupový plán slouží k tomu, abychom mohli zhodnotit přínos technik ukázaných v této kapitole a zároveň stanovili, které techniky jsou pro váš systém nejvhodnější.

Pro zvyšování výkonu systému jsou nejdůležitější: [1]

- *Indexy.* [1] Index znamená databázový objekt, jehož prvotním úkolem je zvětšit výkon dotazů.
- *Formulace SQL s důrazem na výkonnost.* [1]
- *Přizpůsobení fyzických parametrů DBMS.* [1] Jak je místo pro ukládání dat rozčleněno, kolik prostoru je alokováno pro rychlou vyrovnávací paměť (cache) příkazů. Těmto věcem říkáme *fyzické parametry DBMS*. Pro většinu produktů je tato oblast, zcela specifická.
- *Minimalizace izolační úrovně zámků nebo optimistického uzamykání.* [1]

### 4.1 Indexy

Indexy jsou prvotní prostředky,[1] které slouží ke zvyšování výkonu příkazů SQL, a zároveň na jednotlivých platformách nejsou závislé. Ve skutečnosti Index SQL je, jako uspořádaný seznam, který se často podobá rejstříku v knize.

Rejstřík, který obsahuje všechny odkazy na všechny slova v knize (triviální slova se přitom běžně vylučují), se nazývá *Concordance index (úplný rejstřík)*. [1]

Seznam veškerých hodnot nacházejících se v jednom nebo několika sloupcích rejstříku obsahuje Index SQL.[1] Hodnoty se řadí, podle toho, aby pro daný datový typ měly smysl (např. seřazené podle velikosti, podle abecedy atd.) [1] S každou hodnotou je sdružen ukazatel na řádek v tabulce, kde se hodnota vyskytuje.

#### 4.1.1 Kdy používat indexy

S používáním indexů jsou spojeny určité nevýhody. Do dodatečných nákladů za indexy patří: [1]

- když se doplňuje nový řádek, daný index se aktualizuje nebo se daný řádek maže, obsah indexového sloupce se aktualizuje. [1] Vše má negativní vliv na rychlost aktualizací operace, kterou značně zpomaluje. Indexy však obvykle nezpomalují aktualizaci natolik, o kolik zrychlují dotazy.
- Pro jediný index, už nestačí vymezený prostor, [1] který zabírá samotná tabulka. Musíme počítat s určitým místem navíc. Tabulka ve stejném schématu jako index může být uložena samostatně bez indexu.
- Potřebujete-li odebírat z určité tabulky velké množství řádků, je to plýtvání časem, [1] protože přístupováním k indexu, se vykonání dotazu velmi zpomalí.

Většinou se musí určit nějaký přiměřený kompromis k danému systému. [1] Doporučuje se, projít kód své aplikace, systému a objasnit, co potřebuje pokud uvažuje o konstelaci svých indexů. Měli byste sepsat co možná nejlepší předpoklady (hypotézy), budete-li využívat komunikující nebo dynamický SQL. Výhodou procesu optimalizace je to, že daný výkon můžeme průběžně vyladit tak, jak potřebujeme, protože jednotlivé indexy lze průběžně vytvářet nebo mazat.

#### 4.1.2 Typy indexů

Indexy nerozdělujeme pouze na indexy jednoznačné a nejednoznačné, ale existuje i několik dalších rozdělení. [1]

Mezi nejzákladnější rozdělení indexu patří: [1]

- *Stromy B+.*
- *Indexy spojení a klastry.*
- *Bitmapové index.*
- *Stromy R.* Stromy R jsou odlišné od ostatních tím, že každá hodnota je spojena s krajními (mezními) hodnotami. Nejčastěji se tyto indexy (Stromy R), používají v zeměpisných databázích (např. GPS), v databázích inženýrství a virtuální reality.

#### 4.1.2.1 Indexy stromů B+

Hierarchická struktura, které se z hlediska programování říká *strom* (anglicky *tree*), po které se lze rychle pohybovat a vyhledat určité položky. [1] K optimalizaci binárního prohledávání, byl navržen *Strom B+* (B+ tree). *Binární prohledávání* (anglicky *Binary search*) je takové prohledávání, kdy se hledá taková skupina seřazených hodnot, kde se sada hodnot, ve které se má ještě hledat, vždy rozdělí na polovinu. Postup je takový, že nejdříve se vyhledá hodnota uprostřed daného rozsahu a pak se informuje o to, je-li cíl větší nebo menší než tato hodnota. V závislosti na odpovědi zda je menší nebo větší než hledaná hodnota, se bude prohledávat podstrom větších nebo menších hodnot. Dále se postupuje stejným způsobem, opět se hledá hodnota uprostřed rozsahu, pokládá se stejná otázka a přechází se na příslušný podstrom, následující nižší úrovně.

Jestliže nějakým způsobem neupřesníte nějaký jiný typ, tak u mnoho produktů jsou stromy B+ počátečním typem indexů. [1]

Stromy B+ rozdělujeme jak na jednoznačné, [1] které jsou účinnější, tak i nejednoznačné.

#### 4.1.2.2 Indexy spojení a klastry

Indexy spojení a klastry jsou dvě variace na stejné téma. Jako *klastrové indexy* (anglicky *cluster indexes*) se pak označují samotné indexy spojení. [1]

Záměrem *klastru* je ušetřit čas při hledání souhlasných tabulek, které se ve skutečnosti spojují tím, že spojené hodnoty uloží jako jedinou hodnotu. [1] Jednotlivé tabulky upraví na jistý druh meta tabulky. Znamená to, že tabulka, není již v prvním normalizačním tvaru, ale protože klastry obdobně jako indexy uživatelé nevidí, nezáleží na tom. Sjednocení, která se aplikují na tyto sloupce, značně urychluje hledání, ale zároveň to může nepříznivě ovlivňovat aktualizace, a to mnohem výrazněji než u indexů.

Na podobné myšlence je založen *index spojení*, až na to že se vyskytuje společný index na nevlastním a primárním klíči, avšak se ale nejedná o sjednocení tabulek. [1] Každá položka v daném indexu poukazuje na nějakou hodnotu nevlastního klíče, odpovídající hodnotu primárního klíče, RID nevlastního klíče a RID rodičovského klíče. Index je seřazený podle rodičovského klíče.

Tabulky na rozdíl od klastrů jsou uloženy obvyklým způsobem, odlišený jsou pouze indexem. [1] Jednoznačnost je zde charakteristikou určité skupiny sloupců v jedné tabulce,

takže se zde neaplikuje a indexy spojení nejsou tedy samy od sebe jednoznačné. Indexy spojení se běžně používají na rodičovské a nevlastní klíče. Rodičovský klíč ve většině případů bývá omezen pomocí omezení PRIMARY KEY nebo UNIQUE

#### 4.1.2.3 Bitmapové indexy

Mezi exotická díla světa relačních databází, i když jsou už celkem starým programovacím trikem, zařazuje *bitmapové indexy*. [1] Urychlují získávání dat, ale mohou i ušetřit prostor, protože v závislosti na tom jsou využitelné (implementované). Jejich nevýhodou je, že se mohou aplikovat jen ve speciálních situacích. Ve skutečnosti, jsou využitelné pouze u sloupců s velmi nízkou kardinalitou.

Kardinalita je sloupec, který obvykle nemá velký počet možných hodnot [1] (např. pohlaví, ano nebo ne, rodinný stav, umístění poboček, atd.). Bitmapové indexy tuto soustavu následně zakódují do dvojkové soustavy. Pro danou strukturu vymezí, podle počtu možných různých hodnot, odpovídající počet bytů. V každém řádku se pak nachází dvojkové číslo, odpovídající jednomu bitu, který je buď nastavený na 1 a zastupuje hodnotu v tomto řádku nebo je nastavený na 0. (všechny ostatní bity).

*Má dvě výhody:* [1]

- DBMS může při vyhledávání a porovnávání hodnot používat binární operace. [1] Velmi výkonná je přímočará binární matematika, protože všechny ostatní výpočetní matematické operace se musí převádět na binární.
- DBMS konkrétní hodnotu nemusí ukládat, [1] uloží pouze její index. V případě, že se vyvine požadavek na hodnotu, utvoří ji převodem z bitmapy. Samotná hodnota bude větší než bitmapová délka indexu.

Pomocí jedné binární operace AND, můžeme v DBMS zpracovávat jednotlivé bitmapy. [1] Bitmapa je ve skutečnosti silná v situacích, jsou-li dva nebo více sloupců spojené, a mají bitmapové indexy. Výsledkem všeho je pak velmi vysoká účinnost.

#### 4.1.2.4 Různé formulace dotazů

Na tom jak příkaz SQL vyjádříme, většinou nezáleží. Od optimalizátoru v DBMS se čeká, že všechny příkazy upraví na nejúčinnější formulaci. [1] Ve skutečnosti se to ale neděje, i přesto se ale zdá, že většina přeformulování dotazů je zbytečná, protože se velmi blízko

podobá optimálnímu. Z tohoto důvodu neexistují žádná stoprocentně platná a rychlá pravidla, jak příkazy SQL zapisovat optimálně. Ke zvýšení výkonu, obvykle přivádějí všeobecné ověřené principy. Určit, které triky budou dobré pro váš systém, zahrnuje pravidlo ve stylu pokus omyl.

Existují dva typy optimalizátorů založených na: [1]

- pravidlech;
- nákladech.

Není možné oba optimalizátory používat zároveň. [1] Vaše implementace může mít pouze jeden nebo si můžete vybrat ten, který se vám hodí.

Jak se má pokračovat dál, na základě určitých pravidel, určuje *optimalizátor založený na pravidlech*. [1] Zkontroluje, jaké indexy jsou dostupné, zda omezení UNIQUE je schopno vymezit možné souhlasné hodnoty atd. Optimalizátor založený na pravidlech neobsahuje žádné informace o tom kolik řádků má, kolik je dohromady možných hodnot, ani žádné informace o obsahu, jakožto protikladu ke struktuře tabulek. Proto zde nastupuje *optimalizátor založený na nákladech*. Od optimalizátoru založeného na pravidlech se liší tím, že zachová statické informace o obsahu databáze a je schopen inteligentněji optimalizovat dotazy. Nevýhodou je, že člověk musí provést mnohem více práce, než když pracujete s optimalizátorem založených na pravidlech. Např. musíme mu dát najevo, říci mu aby učinil určitou analýzu a vytvořil potřebné statistiky. To je potřeba, udělat několikrát a říct mu aby to dělal pravidelně, takže statistiky budou pořád aktuální.

I když používáme optimalizátor založený na pravidlech [1] a máme zjištěny všechny statistické informace, které o svých datech máme, můžeme je použít pro optimalizaci založené na nákladech.

V tomto seznamu jsou představena, určitá zásadní pravidla, která SQL dotazy urychlí. [1]

1. Když vybíráme nebo provádíme konverzi datových typů pro indexované hodnoty, musíte být obezřetní při vytváření podmínky. [1] Například soudíme že, index na sloupec „odate“ je v pořádku, ale teď provedeme tohle:

```
SELECT *  
FROM Orders  
WHERE CAST odate AS char LIKE ('10%');
```

Obrázek 26 – Ukázka klauzule SELECT s využitím podmínky LIKE [1]

Protože hodnota „odate“, jako taková se v podmínce neověřuje, optimalizátor na sloupec „odate“ index nepoužije. [1] Aby se tato hodnota mohla testovat s podmínkou LIKE, změní se na jiný datový typ. V našem případě, by se podmínka LIKE dala obejít podmínkou BETWEEN, která je schopna pracovat přímo s datem, na rozdíl od podmínky LIKE.

2. Totéž platí pro <> (není rovno). [1] Na základě předpokladu, že se u většiny záznamů hodnoty rovnat nebudou, bude většina dotazů procházet indexy ve sloupcích, které se takto porovnávají.
3. Dobrým přítelem je ANY, za předpokladu známe-li zpracování hodnot NULL a prázdných poddotazů. [1] Bereme-li v úvahu co bylo řečeno, ohledně NULL a prázdných poddotazů, je EXISTS dobrým nebo špatným přítelem. Výhodou je, jakmile se objeví první souhlas, je schopný vykonávání poddotazu zastavit. Pokud optimalizátor nerozpozná, jak ho převést na jiný tvar, vyjadřuje EXISTS korelovaný poddotaz, který bude pomalejší. [1] Proto je příkaz ANY lepší než EXISTS.
4. Na počátek klauzule WHERE, zařaďte podmínky,[1] které patrně vyřadí většinu řádků z výstupu. Jestliže ale omezující podmínka vyhledává jedinou hodnotu, je nejlepší když ji umístíte jako první.
5. Spojení LEFT je schopno být rychlejší než RIGHT OUTER. [1] Spojení LEFT a RIGHT OUTER jsou vzájemně konvertibilní.
6. V klauzuli FROM, tabulku s méně řádky zařaďte ve spojení jako první. [1]
7. Zkombinováním rovností a nerovností můžete také trochu zvýšit výkon, [1] v případě jestliže se odkazujete v určité podmínce na indexovaný sloupec.

Např. column\_value >= 8, nikoliv column\_value > 7. U druhé podmínky se nejdříve najde hodnota 7 a poté se hledá hodnota s vyšším číslem, proto by danou příčinou mohl

být rozdíl ve výkonu. [1] Při zvyšování výkonu se předpokládá, že pracujete s čísli typu INTEGER a ne s čísli typu DECIMAL.

8. Potřebujete-li z tabulky získat více řádků [1] např. více než 20 procent řádků, bude nejlepší, když žádné indexy nebudete používat. Jestliže, ale pro tabulku už nějaký index existuje, optimalizátor se může rozhodnout, že ho využije automaticky.
9. NULL se nepovažuje za hodnotu, proto se většinou neindexuje. [1] Pravděpodobně nebudete schopni pomocí indexu zrychlit kontroly, zda je daná hodnota NULL nebo není (IS NULL, resp. IS NOT NULL).
10. Není potřeba se odkazovat na každý sloupec,[1] odkazujete-li se na vícesloupcový index, je ale zásadou, že musíte začít prvním z nich a pokračovat dál ve stejném pořadí v jakém byli indexovány.
11. Jestliže výrazy začínají zástupným znakem, [1] zejména znakem „%”, který představuje libovolný počet znaků, je nejlepší se výrazu LIKE vyhnout.

## 4.2 Shrnutí

V této kapitole jsme shrnuli problematiku použití indexů a jak pomocí promyšleného psaní SQL dotazů zvýšit jejich výkon. Indexů, které nejrozumnějšími způsoby zvyšují výkon, se vyskytuje několik druhů. Jejich využitelnost je více méně omezena. Na závěr, je zde ukázáno, jak lze jednotlivé dotazy SQL upravit tak, aby je optimalizátory mohly vykonávat efektivněji.



## 5 SEZNÁMENÍ S DATABÁZOVÝM SYSTÉMEM ORACLE

Systémem pro řízení báze dat se nazývá Oracle. [8] Dále někdo Oracle chápe jako multiplatformní databázový systém se značně pokročilými způsoby zpracování dat, vysokým výkonem a snadnou škálovatelností. Databázový systém Oracle vytvořila firma Oracle Corporation. Tato firma je považovaná za jednu z největších společností na světě, která organizacím všech velikostí a firmám dodává podnikový software. Vyjma podnikových aplikací a nástrojů na jejich vývoj, navrhuje i databáze, aplikační servery a nástroje pro podnikatelskou činnost.

Poprvé v České republice se společnost Oracle objevila kolem roku 1990 a v roce 1994 vznikla první česká pobočka společnosti Oracle, za účelem přiblížit českým zákazníkům produkty a služby společnosti Oracle. [4] Velmi rychle se uchytila a po velmi krátké době uvedla do České republiky řadu nových produktů a řešení založených na technologiích Oracle. Podmínkám na českém trhu a legislativě České republiky plně odpovídají všechny řešení společnosti Oracle.

### 5.1 Historie ORACLE

Více jak 30 let, trvalo, než vznikla nejnovější verze Oracle. [8]

- **1977** = Vznikla firma SDL [8] (Software development Laboratory), která se věnovala vývoji DBS, podle teorie doktora E. F. Codd. Zakladateli firmy byli Larry Ellison, Bob Miner a Ed Oates.
- **1978** = Vznikla nová verze Oracle V1, [8] kterou využívali počítače PDP-11 a společnost se přejmenovala na Relational Software Inc.
- **1980** = Byla nová verze Oracle V2, firmou Relational Software Inc. prezentována na trhu. [8]
- **1982** = Byla představena verze Oracle V3 pro sálové počítače, minipočítače a počítače PC. [8]
- **1983** = společnost Relational Software Inc. byla přejmenovaná na Oracle Corporation. [8]
- **1984** = Vznikla nová verze Oracle V4,[8] využitelná pro více platforem, kde je dovolena komunikace mezi PC a serverem.

- **1985** = Vznik nové verze Oracle V5, která byla postavena na architektuře klient-server.
- **1988** = Vznik verze Oracle V6.
- **1989** = Vylepšení verze V6 na verzi V6.2, která využívá Oracle Parallel Server.[8]
- **1991** = Rozšíření výkonnosti na 100 TPS (Transactions Per Second) na paralelně pracujících strojích.
- **1992** = Pro platformu Unix vznikla nová verze Oracle 7.
- **1994** = Verze Oracle 7 byla rozšířena pro platformu PC.
- **1997** = Pro větší podporu více uživatelů, většího množství dat, a lepší dostupnost, vznikla verze Oracle 8.
- **1999** = Rozšíření verze Oracle 8 na Oracle 8i, pro integraci Javy.
- **2000** = Mezi ERP (Enterprise Resource Planning) se zahrnuje verze Oracle8i Release 2.
- **2001** = Vznik verze Oracle 9i.
- **2003** = Vznik verze Oracle 10g.
- **současnost** = Rozšíření verze Oracle 10g na Oracle 10g Release a vznik verze Oracle 11g.

Po řadě technických obnovení a prvenství se společnost Oracle dostala na vedoucí pozici v oblasti databázových systémů.[8]

## 5.2 Současná podoba ORACLE

V současnosti nejaktuálnější verze je Oracle Database 11g. [2] Oproti ostatním verzím tento systém podle normy SQL92, podporuje nejen standardní relační dotazovací jazyk SQL, ale zároveň také proprietární firemní rozšíření Oracle, imperativní programovací jazyk PL/SQL, rozšiřující možnosti vlastního SQL (v tomto jazyce je možné vytvářet procedury, uživatelské funkce, programové balíky a triggerly), dále podporuje objektové databáze a databáze uložené v hierarchickém modelu dat (XML databáze, ...).

### 5.3 Proč Oracle?

Zákazníci mohou těžit z nejnovějšího postupu v oblasti databázových technologií, [7] protože aplikace Oracle nabízí kompletní možnost volby a bezpečný způsob zpracování dat.

Silná kombinace kompletního řešení a kompletní možnosti výběru zaručuje lepší výkonnost podniku a napomáhá zákazníkům spojit své podnikání a IT strategie, [7] optimalizovat jejich investice do informačních technologií. Aplikace příští generace společnosti Oracle Fusion Applications jsou založené na tomto závazku a vymezené pro:

- plnění politiky společnosti Oracle o podpoře po celou dobu životnosti, [7] která pomáhá zajistit, aby měli zákazníci i nadále možnost volby ohledně upgradu, na základě potřeb jejich společnosti.
- práci a vytváření strategické podnikové hodnoty v rámci Applications Unlimited společnosti Oracle. [7]

### 5.4 Porovnání Oracle Database s MySQL

#### 5.4.1 Oracle Database

Tento databázový systém je brán v dnešní době za nejpokročilejší na světě, [9] v současnosti je nejaktuálnější verze Oracle Database 11g. Využívá se především ve velkých firmách a to především pro složité a kritické aplikace s velkými nároky na výkon a bezpečnost. Obsahuje všechny funkce, které si může administrátor databáze přát.

##### 5.4.1.1 Výhody

- podpora aktuálních standardů jazyka SQL, [9] vlastní rozšíření tohoto jazyka;
- programovací jazyk PL/SQL pro tvorbu uložených procedur a jiných objektů;
- velmi pokročilé algoritmy pro zvyšování výkonu při dotazování;
- pouze jeden číselný datový typ – číslo navíc zabírá jen tolik prostoru, kolik je nutné;
- mnoho různých přídavků – například pokročilá komprese dat;
- počítá s nasazením na výkonných serverech;

- oficiální podpora a záruka;
- kromě relačního podporuje i další schémata databáze (objektový, XML).

#### **5.4.1.2 Nevýhody**

- cena až 40 000 \$ za jeden procesor; [9]
- složité nastavení a správa;
- není open source;
- špatná přenositelnost kódu na další databázové systémy.

### **5.4.2 MySQL**

Databáze MySQL patří mezi nejpoužívanější. [9] Nejvíce se využívá pro webové stránky. Tuto databázi používá celá řada velmi známých služeb jako je např. Facebook, Wikipedia, Twitter, aj. Nejdříve se zaměřovala na rychlost a pokročilejší příkazy SQL nevyužívala. To ale už v dnešní době neplatí.

#### **5.4.2.1 Výhody**

- zdarma, opensource; [9]
- snadné nastavení a správa [9] (webová rozhraní – např. phpMyAdmin);
- velké množství návodů na internetu, rozsáhlá komunita;
- možnost nasazení na více serverech;
- většina pokročilých funkcí DBS – uložené procedury, trigger, transakce;
- podporuje ho každý webhosting.

#### **5.4.2.2 Nevýhody**

- nepodporuje všechny SQL příkazy [9] – například WITH result AS (...);
- různé „storage engine“, některé nepodporují určité funkce;
- číslo zabírá prostor dle určeného datového typu;
- oproti komerčním databázovým systémům [9] nemá, tak vyspělé algoritmy pro práci s daty a pokročilé funkce nejsou tak propracované, používané a přenositelné;

- z důvodu otevřeného kódu existuje větší hrozba napadení serveru útočníkem.

### 5.4.3 Porovnání

V případech, kdy má firma dostatek peněz, záleží jí na bezpečnosti dat, zajištěné podpoře při potížích s databází si pořídí databázi Oracle. [9] Oracle se mu potom odvděčí vysokým výkonem a přidá firmě na jejím image.

V případech, kdy se jedná o malou či začínající firmu nebo dokonce se jedná o samostatného uživatele, tak ti nemají jinou možnost než využít nějaký zdarma dostupný databázový systém. [9]

### Shrnutí

Hlavním záměrem této kapitoly bylo, rychle a srozumitelně popsat databázový systém Oracle, tak abychom si vytvořili představu o tom, co je Oracle a jaká byla jeho historie. Vysvětlili jsme také proč zrovna Oracle a jaké má výhody, nevýhody a porovnali ho s velmi používaným databázovým systémem MySQL.

## **II. PRAKTICKÁ ČÁST**



### 6.1.1 Soupis použitých tabulek v databázi – před optimalizací

#### Základní tabulky pro evidenci zařízení

ZARIZENI	• evidence zařízení
ZARIZENI\$KMISTA	• evidence kontrolních míst na zařízení
ZARIZENI\$KMETODY	• evidence kontrolních metod na zařízení

#### Základní tabulky pro plánování a vyhodnocení kontrolních činností na zařízení

PLANY	• evidence plánů
PLANY\$TERMINY	• evidence plánovaných termínů pro kontroly zařízení
PROTOKOLY	• evidence vystavených protokolů z kontrolních činností
PROTOKOLY\$KMISTA	• evidence kontrolních míst vyhodnocených v protokolu

#### Pomocné tabulky (číselníky) – jednotná forma evidovaných dat v databázi

PROGRAMY	• seznam programu kontrol – seskupení zařízení do skupin
BLOKY	• seznam bloku na jaderné elektrárně
METODY	• seznam kontrolních metod
FIRMY	• seznam firem podílejících se na kontrolních činnostech
ODDELENI	• seznam oddělení podílejících se na kontrolních činnostech

### 6.1.2 Analýza současné databáze

Tabulky ZARIZENI, ZARIZENI\$KMISTA, ZARIZENI\$KMETODY tvoří základní kostru celé databáze. Tyto tabulky slouží k evidenci zařízení, kontrolních míst a kontrolních metod, které jsou předepsané státní legislativou (tzv. vybraná zařízení).

#### Tabulka **ZARIZENI**

V tabulce jsou uložena data o zařízení (např. projektové číslo, název, výrobní číslo, atd.), na kterých se provádí předepsané kontroly.

*Primární klíč:*

ZARIZENI_ID_PK	ID
----------------	----

*Unikátní klíč:*

---	---
-----	-----

*Cizí klíč:*

ZARIZENI_IDBLOKU_FK	IDBLOKU	BLOKY
ZARIZENI_IDPROGRAMU_FK	IDPROGRAMU	PROGRAMY

*Index:*

ZARIZENI_ID_PK	ID
NDX_ZARIZENI_IDBLOKU	IDBLOKU
NDX_ZARIZENI_IDPROGRAMU	IDPROGRAMU

(počet záznamů: **30 380**)



Tabulka **ZARIZENI\$KMISTA**

V tabulce jsou uloženy data o místech, na kterých se provádějí předepsané kontroly (kontrolní místa). Kontrolní místo určuje specifickou část zařízení, které podléhá dohledu z hlediska technické bezpečnosti. Každé zařízení má výrobcem zařízení stanoveny místa nebo části zařízení, na kterých se kontroly mají provádět.

*Primární klíč:*

ZARIZENI\$KMISTA_ID_PK	ID
------------------------	----

*Unikátní klíč:*

ZARIZENI\$KMISTA_ZARIZENI_KM	IDZARIZENI, KMISTO
------------------------------	--------------------

*Cizí klíč:*

ZARIZENI\$KMISTA_IDZAR_FK	IDZARIZENI	ZARIZENI
---------------------------	------------	----------

*Index:*

ZARIZENI\$KMISTA_ID_PK	ID
------------------------	----

(počet záznamů: **144 815**)

Tabulka **ZARIZENI\$KMETODY**

V tabulce jsou uložena data o metodě (metodách), kterou se určité kontrolní místo kontroluje, v jakém intervalu (periodě) a termínu se tyto kontroly provádí včetně dalších doplňujících informací potřebných k vytvoření plánu a následně protokolu o provedené kontrole.

*Primární klíč:*

ZARIZENI\$KMETODY_ID_PK	ID
-------------------------	----

*Unikátní klíč:*

---	---
-----	-----

*Cizí klíč:*

ZARIZENI\$KMETODY_IDKMISTA_FK	IDKMISTA	ZARIZENI\$KMISTA
ZARIZENI\$KMETODY_IDFIRMY_FK	IDFIRMY	FIRMY
ZARIZENI\$KMETODY_IDODDEL_FK	IDODDELENI	ODDELENI
ZARIZENI\$KMETODY_IDMETOD_FK	IDMETODY	METODY

*Index:*

ZARIZENI\$KMETODY_ID_PK	ID
-------------------------	----

(počet záznamů: **179 920**)

Tabulky **PLANY** a **PLANY\$TERMINY** se používají k naplánování kontrolních činností. Jsou zde uloženy různé typy plánu (např. koncept, aktuální a archivní). Generování plánu se provádí vytvořenou procedurou na straně ORACLE. Procedura vygeneruje dlouhodobý plán na několik let (max. 12 let). Pro každou metodu je stanoven požadovaný termín

kontroly a při generování plánu se na základě stanoveného intervalu kontroly vypočítají následující termíny.

#### Tabulka **PLANY**

V tabulce jsou uloženy informace o vytvořeném plánu, typu (např. roční, dlouhodobý) a stavu plánu (např. koncept, aktuální, archivní). Hodnoty z aktuálního plánu jsou využívány k vytvoření nabídek pro vystavení protokolu o provedené kontrole a k přehledu o plnění aktuálního plánu.

*Primární klíč:*

PLANY_ID_PK	ID
-------------	----

*Unikátní klíč:*

PLANY_XX2_UN	ELNA, ROK, TYP, REVIZE
--------------	------------------------

*Cizí klíč:*

---	---	---
-----	-----	-----

*Index:*

PLANY_ID_PK	ID
PLANY_XX2_UN	ELNA, ROK, TYP, REVIZE

(počet záznamů: **93**)

#### Tabulka **PLANY\$TERMINY**

V tabulce jsou uloženy informace o jednotlivých metodách s termínem, kdy se bude kontrola provádět s výhledem na několik let dopředu. Po vystavení protokolu o provedené kontrole se do tabulky zapisují základní informace o splnění požadované kontroly.

*Primární klíč:*

PLANY\$TERMINY_ID_PK	ID
----------------------	----

*Unikátní klíč:*

PLANY\$TERMINY_IDPL_IDKM_UQ	IDPLANU, IDKMETODY
-----------------------------	--------------------

*Cizí klíč:*

PLANY\$TERMINY_IDKMETODY_FK	IDKMETODY	ZARIZENI\$KMETODY
PLANY\$TERMINY_IDPLANU_FK	IDPLANU	PLANY

*Index:*

PLANY\$TERMINY_ID_PK	ID
PLANY\$TERMINY_IDPL_IDKM_UQ	IDPLANU, IDKMETODY

(počet záznamů: **5 392 145**)

Tabulky PROTOKOLY a PROTOKOLY\$KMISTA se používají k uložení dat o provedených kontrolách na zařízení podléhající předepsané legislativě. Kontroly se mohou provádět podle vytvořeného plánu nebo operativně podle aktuálních požadavků.

### Tabulka **PROTOKOLY**

V tabulce jsou uložena data identifikující zařízení, datum provedení kontroly (zkoušky), použitou metodu nebo způsob provedení kontroly včetně předepsaných a nastavených parametrů použitých při kontrole, použité zkušební prostředky, popis výsledku kontroly a jména pracovníků, kteří kontrolu provedli a vyhodnotili.

*Primární klíč:*

PROTOKOLY_ID_PK	ID
-----------------	----

*Unikátní klíč:*

---

---

*Cizí klíč:*

PROTOKOLY_IDMETODY	IDMETODY	METODY
PROTOKOLY_IDOBJEDNATELE	IDFIRMY	FIRMY
PROTOKOLY_IDZARIZENI	IDZARIZENI	ZARIZENI

*Index:*

PROTOKOLY_ID_PK	ID
NDX_PROTOKOLY_IDFIRMY	IDFIRMY
NDX_PROTOKOLY_IDMETODY	IDMETODY

(počet záznamů: **254 297**)

### Tabulka **PROTOKOLY\$KMISTA**

V tabulce jsou uložena data o kontrolních místech, která jsou použita v protokolech. Dokladují provedení plánovaných činností a současně vytvářejí vazbu na evidovaná zařízení, jejich kontrolní místa a příslušnou metodu. Tato vazba je využita k prokázání provedení předepsané kontroly v požadovaných termínech a intervalech dle aktuálního plánu.

*Primární klíč:*

PROTOKOLY\$KMISTA_PR	ID
----------------------	----

*Unikátní klíč:*

---

---

*Cizí klíč:*

PROTOKOLY\$KMISTA_IDMETODY	IDKMETODY	METODY
PROTOKOLY\$KMISTA_IDPROT_FK	IDPROT	PROTOKOLY

*Index:*

PROTOKOLY\$KMISTA_PR	ID
----------------------	----

(počet záznamů: **444 402**)

Tabulky BLOKY, FIRMY, PROGRAMY, ODDELENÍ a METODY jsou tabulky pro vytvoření klasických číselníků. Číselníky se používají pro zadávání dat do databáze, tak aby tyto data měly jednotnou formu. Další výhodou číselníku je snížení objemu uložených dat. Do databáze se ukládá pouze odkaz (např. ID záznamu), který slouží k vytvoření vazby do tabulky (číselníku), odkud se načtou ostatní potřebné údaje. Tyto se nemusí ukládat opakovaně.

### Tabulka **PROGRAMY**

V tabulce jsou uložena data o označení programu kontrol, který se používá k seskupení zařízení do skupin. Seskupení je možné využívat při zobrazení dat pro zařízení, které splňují určitá společná kritéria (např. stejný typ zařízení, stejné provozní parametry, atd.). Zkrácené označení programu (např. 1, 2, 3 nebo AT041, AT458, atd.) a název programu (např. Reaktor, Hlavní cirkulační čerpadlo, atd.).

*Primární klíč:*

PROGRAMY\_ID\_PK

ID

*Unikátní klíč:*

---

---

*Cizí klíč:*

---

---

---

*Index:*

PROGRAMY\_ID\_PK

ID

(počet záznamů: **649**)

### Tabulka **BLOKY**

V tabulce jsou uložena data o provozovaných blocích na jednotlivých elektrárnách. Zkrácené označení bloku (např. 1, 2, 3 nebo 01, 02, atd.) a název bloku (např. 1. blok, 2. blok, 01 blok, atd.).

*Primární klíč:*

BLOKY\_ID\_PK

ID

*Unikátní klíč:*

BLOKY\_BLOK\_UN  
BLOKY\_NAZEV\_UN

BLOK  
NAZEV

*Cizí klíč:*

---

---

---

*Index:*

BLOKY\_ID\_PK  
BLOKY\_BLOK\_UN  
BLOKY\_NAZEV\_UN

ID  
BLOK  
NAZEV

(počet záznamů: **17**)

Tabulka **METODY**

V tabulce jsou uložena data o kontrolních metodách používaných při kontrole. Zkrácené označení metody (např. RT, PT, VT, UTP atd.) a název metody (např. prozařování, kapilární kontrola, vizuální kontrola, atd.).

*Primární klíč:*

PROGRAMY\_ID\_PK

ID

*Unikátní klíč:*

METODY\_METODA\_UN

METODA, ELNA

*Cizí klíč:*

---

---

---

*Index:*

PROGRAMY\_ID\_PK

ID

METODY\_METODA\_UN

METODA, ELNA

(počet záznamů: **255**)

Tabulka **FIRMY**

V tabulce jsou uloženy data o firmách, které se podílejí na kontrolách sledovaných zařízení. Zkrácené označení firmy (např. JCV1, UAM, KPS atd.) a název firmy (např. Tediko s.r.o., KPS Metal, atd.).

*Primární klíč:*

FIRMY\_ID\_PK

ID

*Unikátní klíč:*

FIRMY\_FIRMA\_UN

NAZEV, ELNA

FIRMY\_ZKRATKA\_UN

ZKRATKA

*Cizí klíč:*

---

---

---

*Index:*

FIRMY\_ID\_PK

ID

FIRMY\_FIRMA\_UN

NAZEV, ELNA

FIRMY\_ZKRATKA\_UN

ZKRATKA

(počet záznamů: **548**)

Tabulka **ODDELENI**

V tabulce jsou uloženy data o odděleních, které jsou zodpovědné za provedení kontrol na sledovaném zařízení. Zkrácené označení oddělení (např. SPEC, DEFK, TEKO, atd.) a název oddělení (např. Speciální kontrola, Defektoskopie, Technická kontrola, atd.).

*Primární klíč:*

ODDELENI\_ID\_PK

ID

*Unikátní klíč:*

ODDELENI\_NAZEV\_UN

ELNA, NAZEV

ODDELENI\_ZKRATKA\_UN

ELNA, ZKRATKA

*Cizí klíč:*

---	---	---
<i>Index:</i>		
ODDELENI_ID_PK	ID	
ODDELENI_NAZEV_UN	ELNA, NAZEV	
ODDELENI_ZKRATKA_UN	ELNA, ZKRATKA	
(počet záznamů: 29)		

### 6.1.3 Návrh optimalizace

#### 6.1.3.1 Vytvoření indexů

Vytvořením indexů pro vybrané položky v tabulce je možné docílit urychlení SQL dotazů. Při analýze databáze pro sledování technické bezpečnosti bylo zjištěno, že pro vyhledávání zařízení se používají nejčastěji tyto základní informace:

- **pro zařízení a plány** – projektová pozice, označení kontrolního místa, použitá metoda a interval kontroly.
- **pro protokoly** – projektová pozice, označení kontrolního místa, použitá metoda, datum kontroly a výsledek kontroly (např. vyhovuje, nevyhovuje, atd.).

Pro urychlení práce s databází včetně používaných SQL dotazů navrhuji pro následující tabulky doplnit indexy.

V tabulce ZARIZENI navrhuji doplnit index pro položku SJZ, pro urychlení vyhledávání zařízení podle projektové pozice.

```
CREATE INDEX ndx_zarizeni_sjz
ON zarizeni (sjz);
```

V tabulce ZARIZENI\$KMISTA navrhuji doplnit indexy pro položku IDZARIZENI, která vytváří vazbu do tabulky ZARIZENI a pro položku KMISTO kde je uloženo označení kontrolního místa.

```
CREATE INDEX ndx_zarizeni$kmista_idzar
ON zarizeni$kmista (idzarizeni);
```

```
CREATE INDEX ndx_zarizeni$kmista_kmist  
ON zarizeni$kmista (kmisto);
```

V tabulce ZARIZENI\$KMETODY navrhuji doplnit indexy pro položku IDZARIZENI, která vytváří přímou vazbu do tabulky ZARIZENI, pro položku IDKMISTA, která vytváří vazbu do tabulky ZARIZENI\$KMISTA, pro položku IDMETODY, která slouží pro vyhledání zařízení podle metody kontroly a pro položku INTERVAL pro vyhledání zařízení podle intervalu kontroly.

```
CREATE INDEX ndx_zarizeni$kmet_idzar  
ON zarizeni$kmetody (idzarizeni);
```

```
CREATE INDEX ndx_zarizeni$kmet_idkme  
ON zarizeni$kmetody (idkmetody);
```

```
CREATE INDEX ndx_zarizeni$kmet_idkmi  
ON zarizeni$kmetody (idkmista);
```

```
CREATE INDEX ndx_zarizeni$kmet_inter  
ON zarizeni$kmetody (interval);
```

V tabulce PLANY\$TERMINY navrhuji vytvořit index pro položku IDKMETODY, která vytváří vazbu do tabulky ZARIZENI\$KMETODY.

```
CREATE INDEX ndx_plany$terminy_idkme  
ON plany$terminy (idkmetody);
```

V tabulce PROTOKOLY navrhuji doplnit index pro položku SJZ, pro urychlení vyhledávání zařízení podle projektové pozice.

```
CREATE INDEX ndx_protokoly_sjz  
ON protokoly (sjz);
```

V tabulce PROTOKOLY\$KMISTA navrhuji vytvořit indexy pro položku IDKMETODY, která vytváří vazbu do tabulky ZARIZENI\$KMETODY a pro položku IDKMISTA, která vytváří vazbu do tabulky ZARIZENI\$KMISTA.

```
CREATE INDEX protokoly$kmista_ndx_idkmetody  
ON protokoly$kmista (idkmetody);
```

```
CREATE INDEX protokoly$kmista_ndx_idkmista  
ON protokoly$kmista (idkmista);
```

#### **6.1.3.2 Zmenšit objem dat přesunutím starších dat do archivní tabulky.**

V podřízené tabulce PLANY\$TERMINY je uloženo velké množství záznamů. Velká část těchto záznamů patří k plánům, které již byly použity. Tyto plány se používají pouze výjimečně, jedná se o archivní plány. V případě použití těchto záznamů nám již nezáleží tak na rychlosti přístupu k nim proto navrhuji záznamy přesunout do nové archivní tabulky.

V tabulce PROTOKOLY a PROTOKOLY\$KMISTA jsou uloženy záznamy vystavených protokolů za několik let. Podle zažité praxe se nejčastěji používají pro dokladování technické bezpečnosti protokoly z aktuálního roku. Proto navrhuji pro tyto tabulky vytvořit nové archivní tabulky. Do archivní tabulky přesunout všechny protokoly starší než aktuální rok.

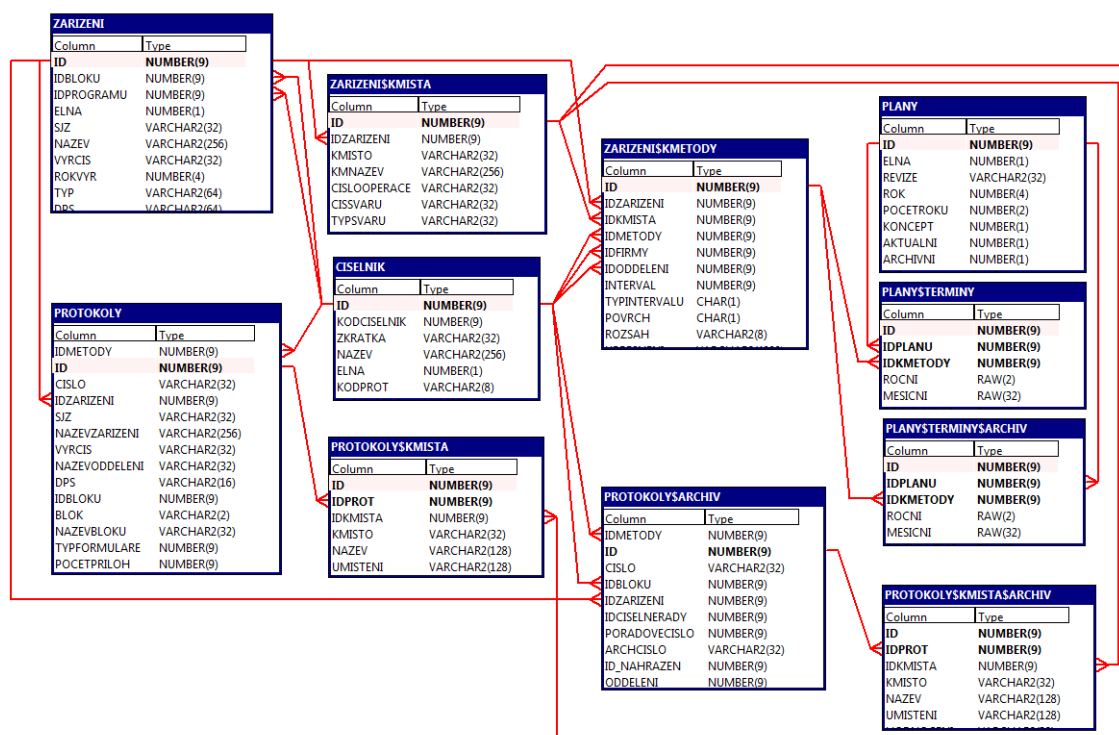
Provedením výše navržených změn se sníží objem dat ve stávajících tabulkách a všechny operace se záznamy se urychlí.

#### **6.1.3.3 Optimalizace číselníků**

Tabulky BLOKY, FIRMY, PROGRAMY, ODDELENI a METODY použité pro číselníky obsahují velmi podobná data. Převážně se jedná o název hodnoty (položky číselníku) a její zkratku. Proto navrhuji vytvořit pouze jednu tabulku a všechny tyto tabulky sjednotit do jediné centrální tabulky, kde budou jednotlivé číselníky rozlišeny číselným kódem. Tímto řešením se sníží počet tabulek v databázi, zjednoduší se údržba celé databáze a také se zjednoduší práce s číselníky při programování obslužné aplikace.



## 6.2 Stav po optimalizaci



Obrázek 28 – Diagram modelu databáze – po optimalizaci

### 6.2.1 Soupis použitých tabulek v databázi – po optimalizaci

#### Základní tabulky pro evidenci zařízení

ZARIZENI

- evidence zařízení

ZARIZENISKMISTA

- evidence kontrolních míst na zařízení

ZARIZENISKMETODY

- evidence kontrolních metod na zařízení

#### Základní tabulky pro plánování a vyhodnocení kontrolních činností na zařízení

PLANI

- evidence plánů

PLANISTERMINI

- evidence plánovaných termínů pro kontroly zařízení

PLANISTERMINYSARCHIV

- evidence plánovaných termínů pro kontroly zařízení (archiv)

PROTOKOLY

- evidence vystavených protokolů z kontrolních činností

PROTOKOLYSARCHIV

- evidence vystavených protokolů z kontrolních činností (archiv)

PROTOKOLYSKMISTA

- evidence kontrolních míst vyhodnocených v protokolu

PROTOKOLYSKMISTASARCHIV

- evidence kontrolních míst vyhodnocených v protokolu (archiv)

#### Pomocné tabulky (číselníky) – jednotná forma evidovaných dat v databázi

CISELNIK

- seznam povolených hodnot

### 6.2.2 Ověření navrženého řešení

V této kapitole jsou popsány a doloženy výsledky navrženého řešení. Výsledky rychlosti provedení použitých SQL dotazů jsou zpracovány formou grafů a u každého dotazu je uvedena průměrná rychlost dotazu před a po optimalizaci.

#### 6.2.2.1 Vytvoření archivních tabulek

Vytvořením archivních tabulek došlo k podstatnému snížení záznamů v aktivní tabulce. Pro porovnání uvádím tabulku s výsledným počtem záznamů před a po optimalizaci.

**Tabulka 19 – Porovnání počtu záznamů v jednotlivých databázových tabulkách před a po optimalizaci**

databázová tabulka	počet záznamů	
	před optimalizací	po optimalizaci
PLANY\$TERMINY	5 392 145	<b>131 389</b>
PLANY\$TERMINY\$ARCHIV	---	5 260 756
PROTOKOLY	254 297	<b>7 167</b>
PROTOKOLY\$ARCHIV	---	247 133
PROTOKOLY\$KMISTA	444 402	<b>9 818</b>
PROTOKOLY\$KMISTA\$ARCHIV	---	434 584

#### 6.2.2.2 Doplnění nových indexů

Pro ověření navržené optimalizace jsem vytvořila několik SQL dotazu, které prokazují zrychlení práce s uvedenou databází. Výběry dat podle určitých kritérií jsou mnohem rychlejší po optimalizaci než před optimalizací.

##### 6.2.2.2.1 SQL dotaz č. 1

Dotaz pro výběr zařízení z aktuálního plánu. Výstupem dotazu je seznam zařízení s kontrolními místy, včetně metody a intervalu kontroly, které jsou zařazeny do aktuálního plánu. V dotazu jsou použity tabulky PLANY, PLANY\$TERMINY, ZARIZENI, ZARIZENI\$KMISTA a ZARIZENI\$KMETODY.

```

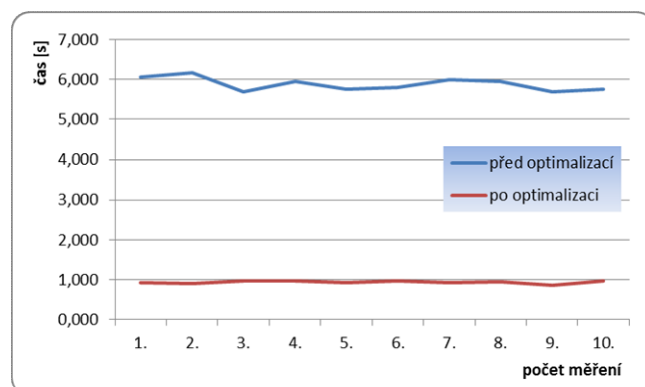
SELECT p.elna, p.revize,
       z.sjz, z.nazev, km.kmisto, m.idmetody, m.interval
FROM plany p, zarizeni z, plany$termíny pt,
     zarizeni$kmista km, zarizeni$kmety m
WHERE p.id = pt.idplanu
     AND pt.idkmetody = m.id
     AND z.id = km.idzarizeni
     AND km.id = m.idkmista
     AND p.aktualni = 1
     AND p.elna = 1
     AND z.sjz = '1YC00B01'

```

Tabulka 20 – Výstup SQL dotazu č. 1 (ukázka)

ELNA	REVIZE	SJZ	NAZEV	KMISTO	IDMETODY	INTERVAL
1	Plán 2013 - 2022 verze 01	1YC00B01	Reaktor	001.018.00.b	16	96
1	Plán 2013 - 2022 verze 01	1YC00B01	Reaktor	001.018.00.b	2	96
1	Plán 2013 - 2022 verze 01	1YC00B01	Reaktor	001.021.01.b	16	96

skutečný počet vybraných záznamů: 1 933



průměrná rychlost SQL dotazu

	hodnota
před optimalizací	5,886 s
po optimalizaci	0,941 s
zrychlení	6,3 x

Obrázek 29 – Graf porovnání rychlosti provedení SQL dotazu č. 1

Prováděcí plány SQL dotazu (explain plan):

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			7439	1	135
HASH JOIN			7439	1	135
HASH JOIN			1866	7	861
HASH JOIN			1155	5	515
MERGE JOIN CARTESIAN			228	1	82
TABLE ACCESS FULL	KONTROLY2	ZARIZENI	223	1	44
BUFFER SORT			5	1	38
TABLE ACCESS FULL	KONTROLY2	PLANY	5	1	38
TABLE ACCESS FULL	KONTROLY2	ZARIZENI\$KMISTA	924	134851	2831871
TABLE ACCESS FULL	KONTROLY2	ZARIZENI\$KMETODY	708	172779	3455580
TABLE ACCESS FULL	KONTROLY2	PLANY\$TERMINY	5501	4155457	49865484

Obrázek 30 – SQL dotaz č. 1 před optimalizací

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			29	4	544
NESTED LOOPS					
NESTED LOOPS			29	4	544
NESTED LOOPS			25	4	392
NESTED LOOPS			15	6	516
NESTED LOOPS			6	5	330
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENI	3	1	45
INDEX RANGE SCAN	KONTROLY2	ZARIZENI_SJZ_UN	2	1	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENISKMISTA	3	5	105
INDEX RANGE SCAN	KONTROLY2	NDX_ZARIZENISKMISTA_IDZAR	1	5	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENISKMETODY	2	1	20
INDEX RANGE SCAN	KONTROLY2	NDX_ZARIZENISKME_IDKMI	1	2	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	PLANYSYSTEMINY	2	1	12
INDEX RANGE SCAN	KONTROLY2	NDX_PLANYSYSTEMINY_IDKME	1	1	
INDEX UNIQUE SCAN	KONTROLY2	PLANY_ID_PK	0	1	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	PLANY	1	1	38

Obrázek 31 – SQL dotaz č. 1 po optimalizaci

## 6.2.2.2.2 SQL dotaz č. 2

Dotaz pro výběr protokolů vystavených na kontrolní činnosti. Výstupem dotazu je seznam protokolů pro vybrané zařízení a kontrolní místo. V dotazu jsou použity tabulky PROTOKOLY, PROTOKOLY\$KMISTA a ZARIZENI\$KMISTA.

```

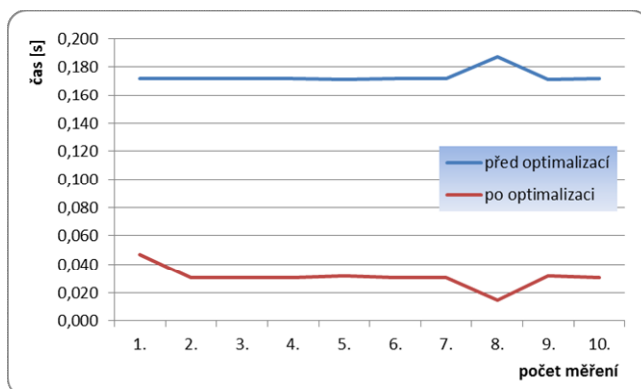
SELECT p.cislo, p.sjz, km.kmisto, km.kmnazev
  FROM protokoly p,
       protokoly$kmista pm,
       zarizeni$kmista km
 WHERE p.id = pm.idprot
       AND pm.idkmista = km.id
       AND p.elna = 1
       AND p.sjz = '3YC00B01'
       AND km.kmisto = '001.123.01.1'
       AND extract(YEAR FROM p.datumkontroly) = 2013

```

Tabulka 21 – Výstup SQL dotazu č. 2 (ukázka)

CISLO	SJZ	KMISTO	KMNAZEV
J 03.13.E.0016	3YC00B01	001.123.01.I	HB - nátrubek HRK (06-37) - vnitřní povrch košilky
J 03.13.V.0416	3YC00B01	001.123.01.I	HB - nátrubek HRK (06-37) - vnitřní povrch košilky

skutečný počet vybraných záznamů: 2



průměrná rychlost SQL dotazu

	hodnota
před optimalizací	1,173 s
po optimalizaci	0,031 s
zrychlení	5,6 x

Obrázek 32 – Graf porovnání rychlosti provedení SQL dotazu č. 2

Prováděcí plány SQL dotazu (explain plan):

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			2927	1	95
NESTED LOOPS			2927	1	95
HASH JOIN			2609	318	16536
TABLE ACCESS FULL	KONTROLY2	ZARIZENISKMISTA	925	68	2720
TABLE ACCESS FULL	KONTROLY2	PROTOKOLYSKMISTA	1677	326025	3912300
TABLE ACCESS BY INDEX ROWID	KONTROLY2	PROTOKOLY	1	1	43
INDEX UNIQUE SCAN	KONTROLY2	PROTOKOLY_ID_PK	0	1	

Obrázek 33 – SQL dotaz č. 2 před optimalizací

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			13	1	95
NESTED LOOPS					
NESTED LOOPS			13	1	95
NESTED LOOPS			12	1	55
TABLE ACCESS BY INDEX ROWID	KONTROLY2	PROTOKOLY	9	1	43
INDEX RANGE SCAN	KONTROLY2	NDX_PROTOKOLY_SJZ	3	6	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	PROTOKOLYSKMISTA	3	5	60
INDEX RANGE SCAN	KONTROLY2	PROTOKOLYSKMISTA_NDX_IDPROT	2	5	
INDEX UNIQUE SCAN	KONTROLY2	ZARIZENISKMISTA_ID_PK	0	1	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENISKMISTA	1	1	40

Obrázek 34 – SQL dotaz č. 2 po optimalizaci

## 6.2.2.2.3 SQL dotaz č. 3

Dotaz pro výběr zařízení z evidence zařízení. Výstupem dotazu je seznam zařízení včetně intervalu kontroly pro vybrané zařízení a jednu konkrétní metodu. V dotazu jsou použity tabulky ZARIZENI, a ZARIZENISKMETODY.

```

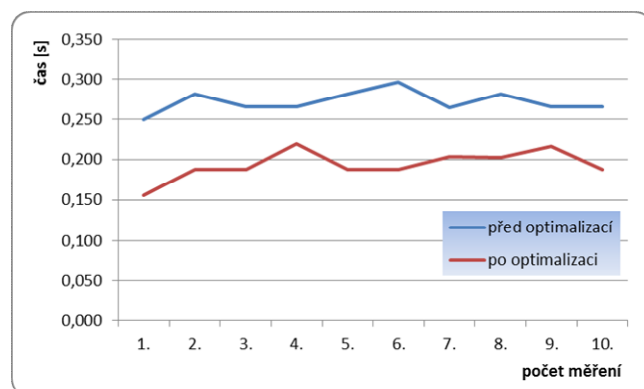
SELECT z.elna, z.sjz, z.nazev, m.interval, m.idkmista
  FROM zarizeni z,
       zarizeni$skmetody m
 WHERE z.id = m.idzarizeni
       AND z.sjz = '1YC00B01'
       AND m.idmetody = 16

```

Tabulka 22 – Výstup SQL dotazu č. 3 (ukázka)

ELNA	SJZ	NAZEV	INTERVAL	IDKMISTA
1	1YC00B01	Reaktor	12	157278
1	1YC00B01	Reaktor	12	157281
1	1YC00B01	Reaktor	36	157282

skutečný počet vybraných záznamů: 477



průměrná rychlost SQL dotazu

	hodnota
před optimalizací	2,272 s
po optimalizaci	0,193 s
zrychlení	1,4 x

Obrázek 35 – Graf porovnání rychlosti provedení SQL dotazu č. 3

Prováděcí plány SQL dotazu (explain plan):

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			930	1	67
HASH JOIN			930	1	67
TABLE ACCESS FULL	KONTROLY2	ZARIZENI	223	1	47
TABLE ACCESS FULL	KONTROLY2	ZARIZENI\$KMETODY	707	16391	327820

Obrázek 36 – SQL dotaz č. 3 před optimalizací

Description	Object owner	Object name	Cost	Cardinali...	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			6	1	68
NESTED LOOPS					
NESTED LOOPS			6	1	68
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENI	3	1	48
INDEX RANGE SCAN	KONTROLY2	ZARIZENI_SJZ_UN	2	1	
INDEX RANGE SCAN	KONTROLY2	NDX_ZARIZENISKME_IDZAR	1	7	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENISKMETODY	3	1	20

Obrázek 37 – SQL dotaz č. 3 po optimalizaci

## 6.2.2.2.4 SQL dotaz č. 4

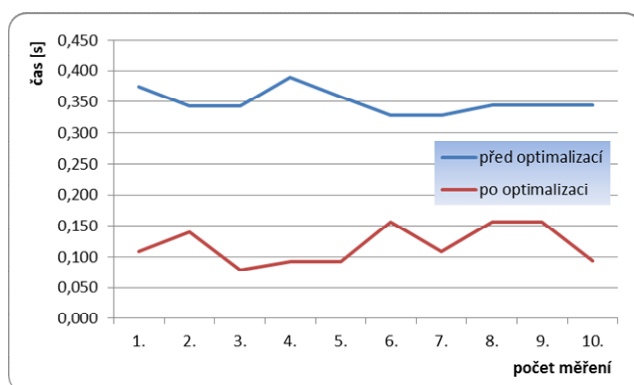
Dotaz pro výběr zařízení včetně protokolů vystavených na provedené kontrolní činnosti a metodu kontroly. Výstupem dotazu je seznam zařízení a protokolů pro vybrané zařízení a kontrolní metodu. V dotazu jsou použity tabulky ZARIZENI, ZARIZENI\$KMISTA, ZARIZENI\$KMETODY, PROTOKOLY a PROTOKOLY\$KMISTA.

```
SELECT z.elna, z.sjz, z.nazev, km.kmisto, m.idmetody, p.cislo
FROM zarizeni z,
     zarizeni$kmista km,
     zarizeni$kmety m,
     protokoly p,
     protokoly$kmista pm
WHERE z.id = km.idzarizeni
     AND km.id = m.idkmista
     AND m.id = pm.idkmetody
     AND p.id = pm.idprot
     and z.sjz = '3YD13D01'
     and m.idmetody = 16
```

Tabulka 23 – Výstup SQL dotazu č. 4 (ukázka)

ELNA	SJZ	NAZEV	KMISTO	IDMETODY	CISLO
1	3YD13D01	Hlavní cirkulační čerpadlo č.3	005.203.17.c	16	J 03.13.V.0777
1	3YD13D01	Hlavní cirkulační čerpadlo č.3	005.203.19.c	16	J 03.13.V.0777
1	3YD13D01	Hlavní cirkulační čerpadlo č.3	005.203.20.c	16	J 03.13.V.0777

skutečný počet vybraných záznamů: **164**



průměrná rychlost SQL dotazu

	hodnota
před optimalizací	0,350 s
po optimalizaci	0,119 s
zrychlení	3,0 x

Obrázek 38 – Graf porovnání rychlosti provedení SQL dotazu č. 4

Prováděcí plány SQL dotazu (explain plan):

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			3567	1	125
NESTED LOOPS			3567	1	125
HASH JOIN			3565	2	192
HASH JOIN			1857	1	85
HASH JOIN			1150	5	340
TABLE ACCESS FULL	KONTROLY2	ZARIZENI	223	1	47
TABLE ACCESS FULL	KONTROLY2	ZARIZENI\$KMISTA	924	134851	2831871
TABLE ACCESS FULL	KONTROLY2	ZARIZENI\$KMETODY	706	16391	278647
TABLE ACCESS FULL	KONTROLY2	PROTOKOLY\$KMISTA	1703	274869	3023559
TABLE ACCESS BY INDEX ROWID	KONTROLY2	PROTOKOLY	1	1	29
INDEX UNIQUE SCAN	KONTROLY2	PROTOKOLY_ID_PK	0	1	

Obrázek 39 – SQL dotaz č. 4 před optimalizací

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			19	1	118
NESTED LOOPS			19	1	118
NESTED LOOPS			18	1	97
NESTED LOOPS			15	1	86
NESTED LOOPS			6	5	345
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENI	3	1	48
INDEX RANGE SCAN	KONTROLY2	ZARIZENI_SJZ_UN	2	1	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENI\$KMISTA	3	5	105
INDEX RANGE SCAN	KONTROLY2	NDX_ZARIZENI\$KMISTA_IDZAR	1	5	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	ZARIZENI\$KMETODY	2	1	17
INDEX RANGE SCAN	KONTROLY2	NDX_ZARIZENI\$KME_IDKMI	1	2	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	PROTOKOLY\$KMISTA	3	1	11
INDEX RANGE SCAN	KONTROLY2	PROTOKOLY\$KMISTA_NXD_TER...	2	1	
INDEX UNIQUE SCAN	KONTROLY2	PROTOKOLY_ID_PK	0	1	
TABLE ACCESS BY INDEX ROWID	KONTROLY2	PROTOKOLY	1	1	21

Obrázek 40 – SQL dotaz č. 4 po optimalizaci



## ZÁVĚR

Databáze je systém pro ukládání dat a jejich následné zpracování. Uložená data mohou být sdílena současně několika uživateli. Existuje několik modelů databází. V současnosti většina moderních databází používá relační model, který je založen na tabulkách a vzájemných vazbách (relacích) mezi nimi. Data jsou uspořádány do řádků a sloupců.

Relační databáze pracuje na principu seskupení jednotlivých tabulek. V databázi nemusí být jednotlivé tabulky ve vzájemném vztahu. Více tabulek ve vzájemném vztahu se pak nazývá schéma.

Druhá kapitola je věnována jazyku SQL. Jazyk SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích a k řízení všech funkcí, které databázový systém svým uživatelům nabízí. Srdce řídicího systému tvoří databázový stroj, který zodpovídá za vlastní segmentování, ukládání a získávání dat z databáze. Jazyk SQL se skládá z několika částí, z nichž některé jsou určeny pro administrátory a návrháře databázových systémů, jiné pak pro koncové uživatele a programátory. Současně patří mezi jednoduchý a snadno pochopitelný.

Třetí kapitola popisuje práci s daty v databázi prostřednictvím jazyka SQL. Mezi základní příkazy pro manipulaci s daty patří `SELECT`, `INSERT`, `UPDATE`, `DELETE`. Mezi nejvýkonnější příkaz patří `SELECT`, který slouží k vyhledávání dat v databázi, které následně vrací v podobě výsledků dotazů. V příkazu `SELECT`, je pořadí klauzulí přesně dané, a nelze ho měnit. Klauzule dělíme na povinné a nepovinné. Příkaz `INSERT` na rozdíl od `SELECT` se nejčastěji používá při vytváření nové databáze, protože slouží pro vkládání dat do tabulky. Pro aktualizaci dat jak ve staré, tak novější databázi slouží příkaz `UPDATE`. V případě že nějaké data nesouhlasí, nebo už je v dané databázi nepotřebujeme, nám slouží příkaz `DELETE`, který se používá pro vymazání dat z tabulek.

Čtvrtá kapitola popisuje, jakým způsobem lze optimalizovat tzn. zlepšit a zefektivnit práci s daty uloženými v databázi. Jedním ze způsobů je použití indexů na vybrané sloupce v tabulkách databáze a druhým způsobem je zápis SQL dotazů.

Indexy jsou prvotní prostředky, které slouží ke zvyšování výkonu příkazů SQL, a zároveň na jednotlivých platformách nejsou závislé. Ve skutečnosti je index, jako uspořádaný seznam, který se často podobá rejstříku v knize.

V dnešní době existuje mnoho indexů, které různými způsoby zvyšují výkon. A to jak jednoznačných, tak nejednoznačných. Jejich využitelnost je více méně omezena.

Na závěr, je zde ukázáno, jakými pravidly při psaní SQL dotazu je vhodné se řídit, aby optimalizátory používané v databázových nástrojích je mohly vykonávat co nejefektivněji.

Předposlední kapitola stručně popisuje databázový systém Oracle včetně historického vývoje. Oracle chápeme, jako systém pro řízení báze dat. V České republice se společnost Oracle objevila kolem roku 1990 a v roce 1994 vznikla první česká pobočka společnosti Oracle, za účelem přiblížit českým zákazníkům produkty a služby společnosti Oracle. Trvalo několik let, než vznikla nejnovější verze Oracle. Nejnovější verzí je verze Oracle 11g., která vznikla v roce 2007.

Poslední část zahrnuje analýzu vybrané databáze pro evidování, plánování a vyhodnocení prováděných kontrolních činností na zařízení podléhající předepsané legislativě, sledování a dokladování technické bezpečnosti. Identifikaci současného stavu a navržení potřebných kroků pro její optimalizaci. Je zde kladen důraz především na analýzu z pohledu výkonu a snížení množství ukládaných dat v aktivně používaných tabulkách, tak i z pohledu budoucích modifikací, úprav a snadného a intuitivního ovládání. S touto částí úzce souvisí i následné vyhodnocení a ověření navržené optimalizace.

Vhodně navržená optimalizace datové struktury, nám umožní přehledné uložení dat a při dalším rozšiřování či úpravě systému budou případné změny jednodušší a levnější.

## CONCLUSION

The database is a system for data storage and retrieval. Several users can share the stored data at the same time. There are several database models. Recent databases use mostly relational-models based on tables and their mutual relations. The data are arranged in rows and columns.

A relational database works on the principle of grouping of individual tables. Individual tables do not need to be in a mutual relation. Several mutually related tables are called a schema.

The second chapter deals with SQL language. SQL language is Standardized Query Language used for the processing of data in relational databases and for the control of all functions provided by the database system for its users. The core of the control system is the database machine responsible for the actual data partitioning, storage and retrieval to/from database. The SQL comprises of several parts; some of them are intended for database system administrators and designers, other for end users and programmers. It is also an easy and comprehensible programming language.

The third chapter describes data processing in database using the SQL. Commands SELECT, INSERT, UPDATE, DELETE are the basic commands for data handling. The SELECT command is one of the most efficient commands. It serves for data retrieval; the data are then given in the form of query results. The order of clauses in the SELECT command is strictly defined and cannot be changed. The clauses are divided into mandatory and optional clauses. Unlike the SELECT command, the INSERT command is the most frequently used command when creating new database because it serves for data insertion into tables. The UPDATE command serves for data updating both in the old as well as in a new database. In case that any data are invalid or the data are no longer needed in a given database, they can be deleted from tables using the DELETE command.

The fourth chapter describes how to optimize (it means make better and more efficient) the processing of data stored within the database. One method is usage of indexes for selected table columns in the database; the option is the construction of SQL queries.

Indexes are the primary tools to increase the power of SQL commands and they are also platform independent. Actually an index is a sorted list similar to an index in a book.

Presently there are many indexes which increase efficiency in various ways. They are either unambiguous or ambiguous. Their applicability is more or less restricted.

Finally, it is presented which rules are suitable for writing of SQL queries to effectively utilize the power of optimizers used in database tools.

Last but one chapter describes briefly the database system Oracle including its history. We regard the Oracle system as the system for database control. The Oracle Company appeared in the Czech Republic in 1990. The first Czech branch of the Oracle Company arose in 1994 in order to familiarize Czech customers with products and services of the Oracle Company. Several years elapsed to the very latest Oracle revision development. The latest revision is Oracle 11g that aroused in 2007.

The last part comprises analysis of selected database for recording, planning and evaluation of performed inspections and checks on devices subjected to prescribed legislature, monitoring and documenting of technical safety. It comprises current condition identification and proposal of necessary steps for its optimizing. First of all the work is focused on analysis from the point of view of efficiency and reduction of amount of stored data in actively used tables as well as from the point of view of future modifications, adjustments and its easy and intuitive control. Subsequent evaluation and verification of proposed optimizing is in close relation to this part.

Appropriately proposed optimizing of data structure will allow transparent data storage, thus in case of further diffusion and/or system modification the alterations are to be easier and less expensive.

## SEZNAM POUŽITÉ LITERATURY

- [1] MARTIN GRUBER. *Mistroství v SQL*. Praha 8: SoftPress s.r.o., 2004. ISBN 80-86497-62-3
- [2] LUBOSLAV LACKO. *Oracle Správa, programování a použití databázového systému*. Praha: Computer Press, 2003. ISBN 80-7226-699-3.
- [3] MARK MASLAKOWSKI. *Naučte se v MySQL za 21 dní*. Praha 4: Computer press, 2001. ISBN 80-7226-448-6.
- [4] THOMAS KYTE. *Oracle: Návrh a tvorba aplikací*. Brno: CP Books, a.s., 2005. ISBN 80-251-0569-5.
- [5] BEGG, Carolyn a Richard HOLOWCZAK. THOMAS CONOLLY. *Mistroství - databáze: Profesionální průvodce tvorbou efektivních databází*. Brno: Computer press, 2009. ISBN 978-80-251-2328-7.
- [6] WEINBERG, Paul N. JAMES R. GROFF. *SQL Kompletní průvodce*. Brno: CP Books, a.s., 2005. ISBN 80-251-0369-2.
- [7] [www.oracle.com](http://www.oracle.com). Oracle [online]. 2001 [cit. 2013-03-28]. Dostupné z: <http://www.oracle.com/cz/products/applications/overview/index.html>
- [8] Oracle průvodce správou využitím a programováním nad databázovým systémem [online]. Havlíčkův Brod: Grada Publishing, a.s., 2009 [cit. 2013-03-28]. ISBN 978-80-247-2762-2.
- [9] Je lepší Oracle nebo MySQL?. In: Blog.petrovsky.cz [online]. 2012 [cit. 2013-04-09]. Dostupné z: <http://blog.petrovsky.cz/25/je-lepsi-oracle-nebo-mysql/>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

DBMS Databázové systémy

DBS Databázový systém

ERP Enterprise Resource Planning

IT Informační technologie

např. Například.

PC Osobní počítač

SDL Software development Laboratory

SQL Strukturovaný databázový systém.

SQL89 Strukturovaný databázový systém verze 89.

SQL92 Strukturovaný databázový systém verze 92.

TPS Transactions Per Second

XML Extensible Markup Language

## SEZNAM OBRÁZKŮ

Obrázek 1 – Data a informace [6] .....	12
Obrázek 2 – Relační databáze pro zpracování objednávky [6] .....	14
Obrázek 3 – Struktura řádek – sloupec relační tabulky POBOCKY [6] .....	15
Obrázek 4 – Relace rodič – potomek v relační databázi [6] .....	16
Obrázek 5 – Použití jazyka SQL pro přístup k databázi [6] .....	22
Obrázek 6 – Části typického databázového řídicího systému [6] .....	24
Obrázek 7 – Základní forma zápisu příkazu SELECT .....	27
Obrázek 8 – Výraz pro výpis údajů o všech DVD [5] .....	28
Obrázek 9 – Výraz pro výpis všech sloupců tabulky [5] .....	29
Obrázek 10 – Výraz pro výpis jednotlivých sloupců z tabulky [5] .....	29
Obrázek 11 – Výraz pro výpis všech žánrů z tabulky [5] .....	30
Obrázek 12 – Výraz pro odstranění všech duplicitních hodnot [5] .....	30
Obrázek 13 – Výraz pro výpis všech zaměstnanců s vyšším platem než je 40 000\$ [5] .....	31
Obrázek 14 – Výraz pro výpis všech zaměstnanců s platem v rozmezích 45 000 – 50 000\$ [5] .....	32
Obrázek 15 – Výraz pro testování rozsahu hodnot pomocí BETWEEN[5] .....	33
Obrázek 16 – Výraz pro výpis všech DVD „sci-fi“ nebo „Children“ [5] .....	33
Obrázek 17 – Dotaz porovnání podle vzoru [5] .....	34
Obrázek 18 – Výraz pro testování hodnoty NULL [5] .....	35
Obrázek 19 – Dotaz pro uspořádání výsledků[5] .....	36
Obrázek 20 – Alternativní řazení .....	37
Obrázek 21 – Výraz pro vyjádření počtu všech zaměstnanců s platem vyšším než 40 000\$ [5] .....	38
Obrázek 22 – Výraz pro výpis nejmenšího, největšího a průměrného platu zaměstnanců [5] .....	38
Obrázek 23 – Výsledný dotaz s použitím GROUP BY [5] .....	39
Obrázek 24 – Ukázka rozčlenění distribučního centra do jednotlivých skupin [5] .....	40
Obrázek 25 – Ukázka následujícího dotazu s použitím klauzule HAVING [5] .....	41
Obrázek 26 – Ukázka klauzule SELECT s využitím podmínky LIKE [1] .....	47
Obrázek 27 – Diagram modelu databáze – před optimalizací .....	55
Obrázek 28 – Diagram modelu databáze – po optimalizaci .....	65
Obrázek 29 – Graf porovnání rychlosti provedení SQL dotazu č. 1 .....	67
Obrázek 30 – SQL dotaz č. 1 před optimalizací .....	67
Obrázek 31 – SQL dotaz č. 1 po optimalizaci .....	68
Obrázek 32 – Graf porovnání rychlosti provedení SQL dotazu č. 2 .....	69
Obrázek 33 – SQL dotaz č. 2 před optimalizací .....	69
Obrázek 34 – SQL dotaz č. 2 po optimalizaci .....	69
Obrázek 35 – Graf porovnání rychlosti provedení SQL dotazu č. 3 .....	70
Obrázek 36 – SQL dotaz č. 3 před optimalizací .....	70
Obrázek 37 – SQL dotaz č. 3 po optimalizaci .....	70

---

Obrázek 38 – Graf porovnání rychlosti provedení SQL dotazu č. 4 .....	71
Obrázek 39 – SQL dotaz č. 4 před optimalizací .....	72
Obrázek 40 – SQL dotaz č. 4 po optimalizaci.....	72



## SEZNAM TABULEK

Tabulka 1 – „Salespeople“ [1] .....	18
Tabulka 2 – „Customers“ [1] .....	18
Tabulka 3 – „Orders“ [1] .....	19
Tabulka 4 – Výsledná tabulka pro dotaz na obrázku 8 a 9 [5] .....	29
Tabulka 5 – Výsledná tabulka pro dotaz na obrázku 10 [5] .....	29
Tabulka 6 – Výsledná tabulka pro dotaz na obrázku 11 s duplicitami [5] .....	30
Tabulka 7 – Výsledná tabulka pro dotaz na obrázku 12 po odstranění duplicit [5] .....	30
Tabulka 8 – Výsledná tabulka pro dotaz na obrázku 13 [5] .....	31
Tabulka 9 – Výsledná tabulka pro dotaz na obrázku 14 [5] .....	32
Tabulka 10 – Výsledná tabulka pro dotaz na obrázku 16 [5] .....	33
Tabulka 11 – Výsledná tabulka pro dotaz na obrázku 17 [5] .....	34
Tabulka 12 – Výsledná tabulka pro dotaz na obrázku 18 [5] .....	35
Tabulka 13 – Výsledná tabulka pro dotaz na obrázku 19 s řazením podle „genre“ [5] .....	36
Tabulka 14 – Výsledná tabulka pro dotaz na obrázku 19 s řazením podle „genre“ a „catalogNo“ [5] .....	36
Tabulka 15 – Výsledná tabulka pro dotaz na obrázku 21 [5] .....	38
Tabulka 16 – Výsledná tabulka pro dotaz na obrázku 22 [5] .....	38
Tabulka 17 – Výsledná tabulka pro dotaz na obrázku 23 [5] .....	40
Tabulka 18 – Výsledná tabulka pro dotaz na obrázku 25 [5] .....	41
Tabulka 19 – Porovnání počtu záznamů v jednotlivých databázových tabulkách před a po optimalizaci .....	66
Tabulka 20 – Výstup SQL dotazu č. 1 (ukázka) .....	67
Tabulka 21 – Výstup SQL dotazu č. 2 (ukázka) .....	68
Tabulka 22 – Výstup SQL dotazu č. 3 (ukázka) .....	70
Tabulka 23 – Výstup SQL dotazu č. 4 (ukázka) .....	71

## **SEZNAM PŘÍLOH**

PŘÍLOHA P I

## **PŘÍLOHA P I: CD – DISK**

### **Obsah disku:**

- Konečná verze bakalářské práce.
- Skripty pro vytvoření schéma optimalizované databáze.