ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

PROVOZNĚ EKONOMICKÁ FAKULTA

DIPLOMOVÁ PRÁCE

OPTIMALIZACE VÝBĚRU DAT V CMDB

Autor: Petr Šorš

Vedoucí bakalářské práce: doc. Ing. Václav Vostrovský, Ph.D.

Praha 2012

ČZU Praha, PEF, obor Informatika

Poděkování:

Chtěl bych poděkovat doc. Ing. Václavu Vostrovskému Ph.D. za čas a úsilí věnované formální a věcné kontrole textu a za vstřícný přístup při konzultacích.

Souhlasím s tím, že s výsledky mé diplomové práce může být naloženo podle uvážení vedoucího diplomové práce a vedoucího katedry. V případě publikace budu uveden jako autor.

Prohlašuji, že na celé diplomové práci jsem pracoval samostatně a použitou literaturu jsem citoval.

V Praze 2012

.....

Petr Šorš

Optimalizace výběru dat v CMDB

Abstrakt: Obsahem práce je problematika optimalizace výběrů dat z relační databáze. V kapitolách jsou popsány užité technologie relačních databází a přístupů k optimalizaci výběrů dat pomocí SQL dotazů. Na konkrétních příkladech je předvedena koncepce indexní analýzy, jako stěžejní techniky pro optimalizaci SQL dotazů. Výsledky jsou uvedeny formou analýz, výsledků a poznatků získaných při realizaci konceptu autorem.

Klíčová slova:

SQL, relační databáze, optimalizace, SQL dotazování, výběr dat, indexní analýza, CMDB

Optimization of data manipulation in CMDB

Summary: The theme of this literary work is optimization of data selection operations on relational database. Following chapters contain information on the used technologies of relational database and approaches to SQL query optimization. An actual implementation of a combination of the technologies is described in a sample scenario. The results gathered from real world testing by the author are presented as a set of analysis, results, examples and best practice recommendations.

The keywords:

SQL, relational database, optimization, SQL query, data selection, index analysis, CMDB

Obsah

1 ÚVOD

Jako téma mé diplomové práce jsem si vybral problematiku, která je mi blízká z mého profesního zaměření. V posledních pěti letech jsem pracoval jako konzultant systémového managementu pro významnou společnost v odvětví logistiky. Tato společnost má velmi rozsáhlé informační prostředí a využívá poslední dostupné technologie na trhu IT. Pro správu výpočetních zařízení společnost využívá systém Symantec Management Platform, jehož jádrem je relační databáze CMDB. Veškeré úlohy prováděné uživateli systému jsou závislé na výkonnosti databáze CMDB. Je proto nutné uplatnit optimalizační přístupy práce s relačními databázemi a SQL dotazováním, aby výkon databáze a potažmo celého systému byl co nejlepší. V této práci jsem popsal nejefektivnější způsoby optimalizace výběru dat, na základě prostudování dostupné literatury a uplatnění doporučovaných postupů v rámci vzorových implementací.

Aktuálnost tématu je podložena faktem, že databáze jsou dnes součástí téměř každého řešení některého s informačních problémů. Faktem je rovněž dominance relačních databází, přičemž implementace objektových databází většinou představují speciální případy. Relačními databázemi, jejich projektování, nasazování do provozu, laděním a údržbou, se celosvětově zabývá značné množství odborníků, jejichž význam roste spolu s významem dobře pracujících databází a datových skladů.

Tato práce je zaměřena na problematiku optimalizace dílčích operací nad relační databází. V práci je čerpáno z aktuálních odborných publikací, jsou uplatňovány konkrétní poznatky a prováděna měření. Výsledky jsou podkladem pro závěr a zobecněná doporučení, aplikovatelná na prakticky všechny implementace relačních databází. Principy optimalizace popsané v této práci byly úspěšně uplatněny v praxi, nejen v rámci systému Symantec Management Platform a databáze CMDB.

2 CÍL PRÁCE A METODIKA

Cílem práce je vymezit teoretické principy relačně databázové technologie a CMDB, zmapovat současnou úroveň využívání CMDB v problematice výběrů dat, identifikovat s tím spojené přetrvávající nedostatky, bariéry a potenciální možnosti, navrhnout odstranění těchto nedostatků či bariér, navržené záležitosti ověřit a demonstrovat v rámci konkrétního řešení, ověřené záležitosti zobecnit pro další možná použití. Konkrétním cílem práce je posoudit přínos optimalizačních přístupů k výběru dat z relační databáze.

Použitá metodika řešené diplomové práce je založena na studiu a analýze dostupných informačních zdrojů. Vybrané technologie jsou systematicky představené a popsané v přiměřené míře podrobnosti. Čtenář je následně seznámen s názorným uplatněním optimalizačních postupů formou navrhovaného řešení, které je realizováno formou praktické implementace respektující identifikované požadavky na tato řešení a ověřuje navrhovaná řešení. Výsledky jsou prezentovány ve formě srovnání naměřených hodnot před a po optimalizační úpravě. Na podkladě syntézy teoretických poznatků a výsledků jsou formulovány závěry této diplomové práce a následně zobecněny pro další možná použití. Rovněž uvedena doporučení a zkušenosti získané autorem během práce s popisovanými technologiemi.

3 TEORETICKÝ POPIS UŽITÝCH TECHNOLOGIÍ A PRODUKTŮ

3.1 STRUCTURED QUERY LANGUAGE – SQL

3.1.1 HISTORIE SQL

Na začátku 70. let dvacátého století výzkumný pracovník doktor E. F. Codd, pracující pro společnost IBM, dal svou seminární prací impuls pro vznik relačního datového modelu nazvaného SEQUEL. Později byl tento model pojmenován zkráceně SQL, tedy Structured Query Language. (2, kap. 1.1)

Společnost IBM a další výrobci relačních databázových produktů, měli za cíl vytvořit standardizovaný způsob přístupu k datům a manipulaci s nimi v relačním datovém modelu. Ačkoliv to bylo právě IBM, které první vyvinulo teoretické principy relační databáze, společnost Oracle s touto technologií jako první začala obchodovat. Později se díky značné popularitě o SQL začal zajímat Americký národní institut pro standardy (ANSI). Tak vznikla v letech 1986, 1989, 1992, 1999 a 2003 série standardů pro SQL. (2, kap. 1.1)

Se zrodem SQL, vzniklo množství konkurenčních jazyků určených k manipulaci a přístupu k relačním datům. Pouze malé množství z těchto jazyků bylo srovnatelně srozumitelných a snadných pro osvojení, jako standardizované SQL. Dnešní programátoři, administrátoři a uživatelé jazyka SQL po osvojení jeho standardní formy mohou tuto znalost uplatnit na různých implementacích bez větších problémů. (2, kap. 1.1)

Mezi nejvýznamnější implementace jazyka SQL se řadí:

- IBM DB2 Universal Database
- MySQL
- Oracle Database
- PostgreSQL
- Microsoft SQL Server

3.3 RELAČNÍ DATOVÝ MODEL

Relační systém řízení báze dat lze definovat jako systém, jehož uživatelé nahlíží na data jako na kolekci tabulek navzájem spojených společnými hodnotami. Data jsou uložená v tabulkách, které se skládají z řádků a sloupců. Tabulky nezávislých dat mohou být navzájem spojeny (resp. může mezi nimi být vytvořena relace), mají-li sloupce s jedinečnými (ve smyslu unikátními) daty identifikujícími každý řádek. Tato data se nazývají klíče a reprezentují právě to hodnotu, kterou tabulky mají společnou (např. identifikátor).

Relační model byl poprvé teoreticky popsán E. F. Coddem v práci „A Relational Model of Data for Large Shared Data Banks“, volně přeloženo Model relačních dat v robustních sdílených datových skladech. Práce, kterou vydala Asociace výpočetní techniky (ACM) už v roce 1970, byla Coddovou stěžejní a proslavila jej.

Vedle systému tabulek, sloupců a řádků byly v Coddově práci zavedeny způsoby pro práci s daty a uchování jejich správnosti a v neposlední řadě pravidla pro tvorbu normalizovaných relačních databází. (2, kap. 1.1)

3.3.1 CODDOVA PRAVIDLA PRO RELAČNÍ DATABÁZOVÉ SYSTÉMY

V definici řízení dat Codd aplikoval rigorózní matematické teorie, především tu o množinách. Zároveň sestavil seznam kritérií, která musejí databáze splnit, aby byly relační. Koncept relačního modelu je založen na tabulkách, což se zprvu zdálo jako nepraktické. Avšak dodržením dvanácti následujících pravidel pro relační databáze se systém osvědčil a relační databáze jsou nyní převládající (vedle například těch objektových).

Pravidlo první: informace jsou na logické úrovni reprezentovány tabulkami.

Pravidlo druhé: všechna data musí být na logické úrovni dostupná na základě jména tabulky, sloupce a primárního klíče daného řádku.

Pravidlo třetí: prázdná pole, musí být reprezentována jako chybějící data (NULL), nikoliv jako prázdný řetězec, nulová hodnota, nebo mezera. Dále SŘBD musí podporovat systematickou manipulaci s NULL poli.

Pravidlo čtvrté: metadata, neboli datový katalog, musí být rovněž uložen v databázi, stejně jako data samotná. Uživatelé musí být schopni přistupovat k metadatům stejným způsobem (stejnými dotazy), jako k datům uživatelským.

Pravidlo páté: jediný, vyčerpávající jazyk, lze použít pro definování a manipulaci s daty, k práci s prvky pro bezpečnost a integritu, řízení transakcí a to jak interaktivně, tak jako součást aplikací.

Pravidlo šesté: pohledy (při sestavení) musí obsahovat aktuální data, to znamená, že při změně dat v jedné ze zdrojových tabulek se změna musí projevit i v pohledu.

Pravidlo sedmé: v jediné operaci musí být možné provést vložení, změnu nebo vymazání množiny dat, a to bez ohledu na jejich umístění v databázi (jsou-li v jedné tabulce, nebo v mnoha).

Pravidlo osmé: je zajištěna nezávislost mezi způsobem uložení fyzických dat databáze, a logickou reprezentací dat. Dávkové a uživatelské operace nesouvisí a nezohledňují žádným způsobem fyzická data, pouze jejich logickou vrstvu. Veškeré operace s fyzickými daty (vlastními soubory databáze) musí zajistit SŘBD.

Pravidlo deváté: je zajištěna nezávislost logické vrstvy dat a aplikační vrstvy. Při změně schématu logických dat, není nutné učinit změny v aplikační vrstvě.

Pravidlo desáté: integritní omezení jsou definována v metadatech, čili uložena v databázi a nikoliv v aplikaci. Změnou integritních omezení tak není nutné měnit aplikační kód.

Pravidlo jedenácté: SŘBD, respektive vlastní relační model a jazyk SQL musí zajistit nezávislost na formě uložení fyzických souborů databáze. V případě centrálního, nebo distribuovaného uložení, uživatelé, potažmo aplikace nesmí zaznamenat změnu v přístupu k datům a práci s nimi, změní-li se například uložení datových souborů z centrálního na distribuovaný a naopak.

Pravidlo dvanácté: pravidla pro relační databázi platí ve všech případech. Nesmí existovat způsob, kterým by bylo možné tato pravidla porušit (například porušení omezení integrity dat při přístupu k datům speciálním rozhraním).

Pravidlo nulté (třinácté): Aby bylo možné konkrétní systém kvalifikovat jako SŘBD, musí se skládat z databáze a databázového stroje a zcela splňovat pravidla relačního modelu. Dále musí veškeré své součásti a funkce využívat pro řízení databáze, nebo databází (například provoz, přístup, rozhraní). (2, kap. 1.1.1)

3.3.2 METADATA

Metadata jsou data o datech. Metadata popisují schéma databáze, její objekty, prvky. Například každá tabulka vlastní databáze musí mít záznam v příslušné tabulce metadat. Takový záznam minimálně musí obsahovat název tabulky, její schéma, jméno katalogu a její typ.

Definice pro uložení metadat podléhá 4. zákonu Coddových pravidel pro relační datový model. (2, kap. 1)

3.3.3 POHLEDY

Pohled je pomyslná, neboli virtuální tabulka, která se sestaví ve chvíli, kdy nad ní některý uživatelský, nebo aplikační proces vykoná operaci. Pohled tedy není fyzickým objektem v podobě tabulky, ale jediný se o kus SQL kódu, kterým se zkonstruuje virtuální tabulka naplněná daty.

Původním významem pohledů, bylo prezentovat část báze dat každému z uživatelů, nebo skupině uživatelů odlišným způsobem. Typickým příkladem můžou být dva pohledy na data docházkového systému, jeden pro zaměstnance, druhý pro personálního ředitele. Zdrojem jsou tatáž data, ale každý z pohledů umožňuje zobrazení podmnožiny dle kritérií uživatele.

Původní význam pohledů byl ale překonán a jejich využití je mnohem širší. (2, kap. 1)

3.3.4 OPERACE NAD MNOŽINAMI DAT

SQL není jediným druhem jazyka určeným pro manipulaci s databázemi. Existují jazyky jako je Xbase, které jsou od SQL velmi rozdílné. Přistupují totiž k datům jako k jednotlivým řádkům, nebo záznamům. Práce s databází prostřednictvím těchto jazyků je v podstatě procedurálním programováním, respektive řádkovým zpracováváním (row processing) dat.

Rozdíl SQL je v přístupu k datům jako k množinám a práci s daty jako k operacím nad množinami. Teorie množin je uplatněna ve většině příkazů (například SELECT, INSERT,

UPDATE, DELETE, JOIN a UNION). V podstatě jsou data vybírána z množiny, které se říká tabulka. Na rozdíl od řádkového procesování, umožňuje množinové procesování dat programátorovi (obecně uživateli databáze) formulovat příkaz pro SŘBD ve formě co je požadováno, nikoliv jak se dobrat k výsledným datům.

Množinové zpracovávání se někdy nazývá deklarativním zpracováváním.

3.3.5 ZÁKLADNÍ RELAČNÍ OPERACE

Relační operace se dělí do tří základních kategorií:

- Selekce – operace výběru řádků.
- Projekce – operace výběru sloupců.
- Spojení – operace výběru řádků ze dvou a více tabulek spojených klíčovou hodnotou.

Příklad:

TABULKA 1: ZÁKLADNÍ RELAČNÍ OPERACE, VZOREK DAT

id_studenta	znamka	datum
xnovakj	2	11/27/2011
xnovako	3	11/27/2011
xmachl	2	11/27/2011
xortp	2	11/27/2011
xtomani	1	11/28/2011
xmareko	3	11/29/2011
xtrnkaj	3	11/30/2011
xdivisj	1	12/1/2011

SQL 1: ZÁKLADNÍ RELAČNÍ OPERACE, PŘÍKLAD PROJEKCE

```
SELECT id_studenta, znamka
FROM hodnoceni
ORDER BY id_studenta ASC
```

Příkaz v rámečku výše je projekcí, protože SŘBD navrátí všechny řádky pro dané sloupce, tedy celou množinu.

TABULKA 2: ZÁKLADNÍ RELAČNÍ OPERACE, VÝSLEDEK PROJEKCE

id_studenta	znamka
Xdivisj	1
Xmachl	2
Xmareko	3
Xnovakj	2
Xnovako	3
Xortp	2
Xtomani	1
Xtrnkaj	3

SQL 2: ZÁKLADNÍ RELAČNÍ OPERACE, PŘÍKLAD SELEKCE

```
SELECT id_studenta, znamka, datum
FROM hodnoceni
WHERE znamka <= 2
ORDER BY datum DESC
```

Příkaz v rámečku výše je selekcí, protože SŘBD navrátí pouze ty řádky splňující danou podmínku, tedy podmnožinu.

TABULKA 3: ZÁKLADNÍ RELAČNÍ OPERACE, VÝSLEDEK SELEKCE

id_studenta	znamka	datum
Xdivisj	1	12/1/2011
Xmachl	2	11/27/2011
Xnovakj	2	11/27/2011
Xortp	2	11/27/2011
Xtomani	1	11/28/2011

Operace spojení umožňuje práci s daty, jako s průnikem, či rozdílem množin. Základním příkazem je JOIN, kterým lze spojit dvě a více tabulek na základě společných atributů.

SQL 3: ZÁKLADNÍ RELAČNÍ OPERACE, PŘÍKLAD SPOJENÍ

```
SELECT s.id_studenta, obor, znamka
FROM studenti AS s
JOIN hodnoceni AS h
ON s.id_studenta = h.id_studenta
```

SQL 4: ZÁKLADNÍ RELAČNÍ OPERACE, PŘÍKLAD THÉTA SPOJENÍ

```
SELECT s.id_studenta, obor, znamka
FROM studenti s, hodnoceni h
WHERE s.id_studenta = h.id_studenta
```

TABULKA 4: ZÁKLADNÍ RELAČNÍ OPERACE, VÝSLEDEK SPOJENÍ

id_studenta	obor	rocnik
xdivisj	INF	1
xmachl	SYS	2
xmareko	INF	2
xnovakj	SYS	1
xnovako	INF	1
xortp	SYS	2
xtomani	INF	2
xtrnkaj	SYS	1

Dle standardu ANSI, lze realizovat spojení tabulek příkazem JOIN vnořeném v příkazu SELECT. Existuje však starší metoda, která není standardem ANSI podporovaná, ale ve většině implementací funguje. Alternativní metoda se nazývá théta spojení (theta join) a parametry relace dvou a více tabulek jsou definovány v podmínce WHERE. (2, kap. 1)

3.4 VARIANTY JAZYKA SQL

Během období vývoje standardu pro jazyk SQL vzniklo množství jeho variant. Nejčastější příčinnou bylo vydání konkrétních implementací jazyka před tím, než pro jazyk vznikl standard. Varianty jsou odlišné i z důvodu integrace s dalšími jazyky, například s Javou

(Oracle, DB2 a Sybase) nebo C# či Visual Basic (Microsoft SQL Server). Novodobé implementace jazyka SQL převzaly určité vlastnosti procedurálních jazyků. Díky tomu má dnes mnoho variant jazyka SQL prvky umožňující efektivněji pracovat s daty pomocí smyček, podmínek, rekurzívního volání a dalších programovacích technik.

Standard jazyka SQL byl institutem ANSI rozšířen o definice týkající se procedurálních vlastností. Pojmenování standardu je Structured Query Language/Persistent Stored Modules (SQL/PSM) a jeho první implementace byla uskutečněna v roce 1996. Označení standardu je ISO/IEC 9075-4:2003.

Nejvýznamnější varianty jazyka SQL jsou:

- Procedural Language /SQL (PL/SQL) společnosti Oracle.
- Transact-SQL, použitý v produktech společností Sybase a Microsoft.
- Procedural Language /postgreSQL (PL/pgSQL) implementován v PostgreSQL
- SQLPL implementován v produktech DB2 společnosti IBM. (2, kap. 1.3)

3.5 STANDARDY A NORMY PRO SQL

První standard pro SQL byl institutem ANSI vydán v roce 1986. Následovalo druhé vydání (rok 1989), které bylo mnohem lépe přijaté a šířeji použité v databázových produktech. Ve stejném roce byl SQL standard uznán organizací ISO. Následovaly aktualizace SQL92, známé jako SQL2 (rok 1992) a později SQL99, známé jako SQL3 (rok 1999). Standard SQL platný v dnešní době se nazývá SQL2003.

3.5.1 SOUHRN ZMĚN SQL2003

Standard SQL2003 obsahuje obě původní části SQL99 – „základ“ (Foundation) 1999 a „vazby“ 1999 (Bindings). Změnou standardu samotného je nová část nazvaná „schémata“ (Schemata).

Jádro SQL se v SQL2003 od předchozích verzí nezměnilo vůbec. Tím byla zajištěna zpětná kompatibilita pro existující implementace. Zatímco až na několik nových rezervovaných slov nepřibyly nové prvky, několik jich bylo odebráno, například:

- datové typy BIT a BIT VARYING (přestože například MS SQL si ponechal BIT);
- klauzule UNION JOIN;
- příkaz UPDATE … ROW SET.

Do nového standardu bylo přidáno i několik (zatím ryze akademických) vlastností:

- základní funkce OLAP;
- vzorkování (klauzule TABLESAMPLE pro podporu statistických funkcí);
- vylepšení numerických funkcí. (2, kap. 1.2)

3.6 KATEGORIE PŘÍKAZŮ VE STANDARDU SQL2003

Dle standardu SQL92 (SQL2) se výrazy jazyka SQL mohou zařadit do jedné z následujících tří kategorií: (2, kap. 2)

Data Manipulation Language (DML)

- Kategorie příkazů pro manipulaci s daty
- Příkladem jsou SELECT, INSERT, UPDATE a DELETE

Data Definition Language (DDL)

- Kategorie příkazů a výrazů určených pro tvorbu a práci s databázovými objekty
- Příkladem jsou CREATE a DROP

Data Control Language (DCL)

- Kategorie příkazů pro řízení přístupu k databázi
- Příkladem jsou GRANT a REVOKE

Standard SQL2003 rozšiřuje původní tři kategorie na sedm tříd tvořících rámec SQL příkazů.

Název třídy	Popis	Vzorové příkazy
Příkazy pro navazování spojení	Pro spouštění a ukončení spojení	CONNECT, DISCONNECT
Příkazy pro řízení	Pro řízení spouštění série příkazů	CALL, RETURN
Příkazy pro práci s daty	Pro čtení i stálé změny dat	SELECT, INSERT, UPDATE, DELETE
Příkazy pro diagnostiku	K diagnostice a sledování událostí	GET DIAGNOSTICS
Příkazy pro práci se schématem	Pro změny schématu databáze	ALTER, CREATE, DROP
Příkazy k nastavování prostředí	Řízení výchozího chování relace	SET, SET CONSTRAINT
Příkazy pro řízení transakcí	Spouštění, končení, dávkování	COMMIT, ROLLBACK

3.7 SYNTAXE JAZYKA SQL

Jazyk SQL má čtyři syntaktické kategorie: identifikátory, literály, operátory a rezervovaná a klíčová slova.

3.7.1 IDENTIFIKÁTORY

Identifikátory jsou popisné názvy pro objekty (prvky) databáze. Popisné proto, že z jejich názvu by mělo být lze odvodit účel objektu, jeho roli. Identifikátory tedy pojmenovávají tabulky, atributy (sloupce), pohledy, funkce, spouště a další objekty. (2, kap. 2)

Příklad identifikátoru pohledu:

Skolni_databaze.dbo.vStudenti

Jméno pohledu je vStudenti, samotné jméno může být použito v souvislosti s uživatelskou databází Skolni_databaze, kde pohled je v katalogu dbo. Mimo souvislost (session) s databází Skolni_databaze je nutné použít celý (absolutní) název výše. (2, kap. 2)

3.7.2 LITERÁLY

Literál je jakýkoliv textový řetězec, číselná hodnota, dočasná hodnota (například datum) nebo Booleovská hodnota, která není identifikátorem, operátorem, nebo rezervovaným či klíčovým slovem.

Numerické literály nemusí být uzavřené v jednoduchých uvozovkách, zatímco textové ano.

Příklad numerických literálů typů smallint, float, money respektive:

9; 3.4501; $100

Booleovské, textové a datové literály typů boolean, bit, char, datetime respektive:

TRUE; 1; "Hello world"; 12/02/2011 03:19:00 (2, kap. 2)

3.7.3 OPERÁTORY

Operátorem se rozumí specifický symbol charakterizující úkon se dvěma, či více výrazy. Nejčastější použití operátorů je uvnitř dotazů, příkazů a uvnitř funkcí, uložených procedur a pohledů.

Základní dělení operátorů a jejich podpora implementacemi SQL:

Aritmetické operátory

TABULKA 5: SQL OPERÁTORY, ARITMETICKÉ OPERÁTORY

Operátor	Význam
+	Součet číselných hodnot, sloučení řetězců
-	Rozdíl číselných hodnot
*	Součin
/	Podíl
%	Modulo (zbytek po celočíselném dělení)

Jsou podporovány všemi implementacemi SQL, s výjimkou modulo, které podporuje pouze MS SQL Server. Implementace DB2, Oracle a MS SQL Server navíc umožňují použít operátory součtu a rozdílu pro práci s datovým typem Datetime.

Operátory přiřazení

Operátory přiřazení slouží k načtení hodnoty do proměnné. Obecným symbolem je rovnítko „=“, výjimkou jsou implementace Oracle, kde je užit symbol „:=“. V MS SQL Server se pro přiřazení aliasu k názvu sloupce tabulky provede použitím klíčového slova „AS“.

Logické operátory bitwise

MS SQL Server jako jediná implementace jazyka SQL umožňuje práci s logickými operátory druhu bitwise, s omezením na datové typy binary, bit, int, smallint, tinyint a varbinary.

TABULKA 6: SQL OPERÁTORY, LOGICKÉ BITWISE OPERÁTORY

Operátor	Význam
&	Logické AND
|	Logické OR
^	Logické XOR

Logické operátory booleovské

Booleovské operátory jsou předně užívány v podmínkových klauzulích WHERE. Výsledkem je buď TRUE nebo FALSE. (2, kap. 2)

TABULKA 7: SQL OPERÁTORY, LOGICKÉ OPERÁTORY BOOLEOVSKÉ

Operátor	Význam
ALL	Navrátí TRUE, je-li podmínka splněna všemi prvky, které jsou porovnávány.
AND	Navrátí TRUE, je-li podmínka splněna oběma prvky.
ANY	Navrátí TRUE, je-li podmínka splněna alespoň jedním prvkem.
BETWEEN	Navrátí TRUE, je-li operand v požadovaném rozsahu.
EXISTS	Navrátí TRUE, je-li výsledkem vnořeného dotazu alespoň jeden řádek.
IN	Navrátí TRUE, shoduje-li se alespoň jeden prvek s výsledkem vnořeného dotazu.
LIKE	Navrátí TRUE, shoduje-li se operand s částí zadaného řetězce.
NOT	Invertuje jiný booleovský operátor, například NOT LIKE.
OR	Navrátí TRUE, je-li alespoň jedna navrácená hodnota použití operátoru TRUE.
SOME	Navrátí TRUE, je-li některý z prvků v množině porovnání TRUE.

Unární logické operátory

TABULKA 8: SQL OPERÁTORY, UNÁRNÍ OPERÁTORY

Operátor	Význam
+	Explicitní nastavení kladné hodnoty čísla.
-	Explicitní nastavení záporné hodnoty čísla.
~	Bitwise NOT, doplněk do čísla.

3.7.4 REZERVOVANÁ A KLÍČOVÁ SLOVA

Některá slova a fráze mají zvláštní význam, který souvisí s konkrétními operacemi SŘBD. Klíčová slova, jako například SELECT, by neměla být použita pro pojmenování databázových objektů, proměnných, funkcí apod. Rezervovaná slova, jsou taková, co zatím nemají zvláštní význam, ale mít mohou v budoucích verzích SQL standardu, nebo produktů SŘBD. (2, kap. 2)

3.8 MICROSOFT SQL SERVER

3.8.1 HISTORIE PRODUKTU

V osmdesátých letech minulého století, měla společnost Microsoft ještě okolo tisíce zaměstnanců a její zaměření bylo zejména na produkty určené pro osobní počítače. V té době neměla v portfoliu žádný databázový produkt, ale úzce spolupracovala se společností IBM, která již měla komerční databázové produkty Database Manager a robustnější DB/2. Spolupráce mezi oběma společnostmi byla zaměřena na produkt OS/2, nástupce operačního systému DOS (MS-DOS, PC-DOS). Společnost Microsoft učinila rozhodnutí vyvinout vlastní databázový produkt a oslovila relativně málo známou společnost Sybase, Inc, která měla zatím nekomerční verzi produktu DataServer. Spolupráce byla vzájemně výhodná, jak po technologické tak obchodní stránce. (3, online)

Aby se připravovaný databázový produkt dobře uvedl na trh, učinila společnost Microsoft marketingový krok a nechala produkt posvětit v té době renomovanou firmou Ashton-Tate. První verze se tak jmenovala Ashton-Tate/Microsoft SQL Server a přestože jméno společnosti Sybase nefigurovalo v názvu produktu, bylo přítomné ve všech souvisejících informacích o produktu. První verze byla oznámena v roce 1988 a vyšla do prodeje v roce 1989.

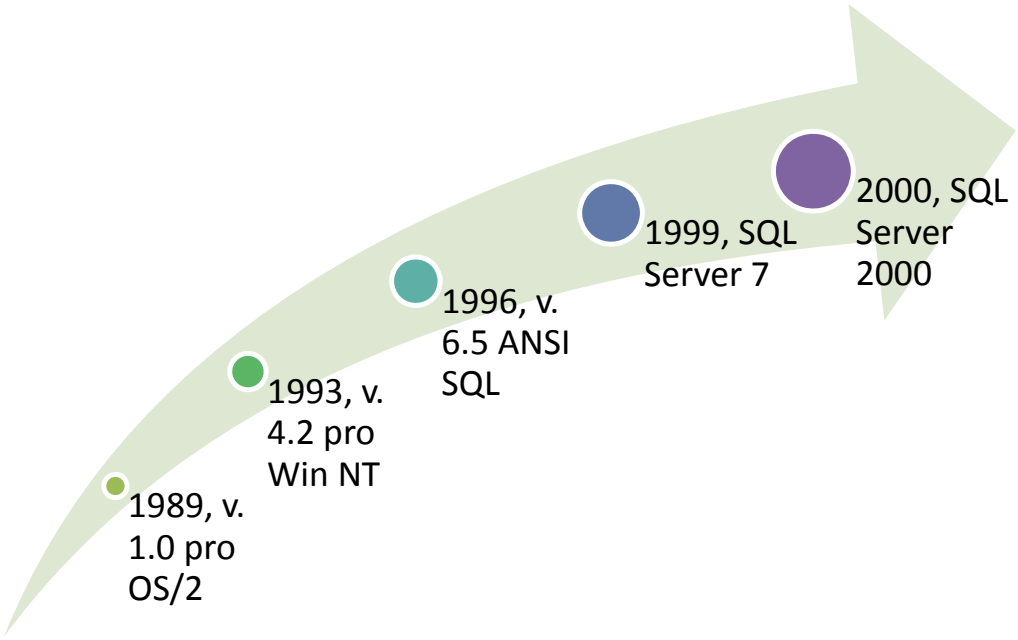
Následovaly verze 1.1 a 1.11, především obsahovaly opravy chyb a několik nových vlastností a funkcí. Produkt byl zaměřen na menší podniky, pro osobní počítače s OS/2. Pro náročnější provoz byl určen Sybase SQL Server, který byl určen pro operační systém UNIX.

V roce 1992 Bill Gates a Bob Epstein (Sybase) představili verzi 4.2, která jako první obsahovala grafické rozhraní pro systém Windows. V témže roce Microsoft vydal beta verzi SQL Serveru pro Windows NT, plně 32-bitovou verzi. V roce 1993 byl vydán operační systém Microsoft Windows NT 3.1 a s ním Microsoft SQL Server for Windows NT.

V dubnu roku 1994 společnosti Sybase a Microsoft oznámili ukončení spolupráce na vývoji a každá se vydala na vlastní dráhu.

Velmi očekávaným vydáním byla verze 6.0, kterou Microsoft slíbil vydat v roce 1995. Slib byl splněn a produkt s kódovým označením SQL95 byl přijat neobyčejně pozitivně. Za necelých 10 měsíců po vydání 6.0, byla vydána verze 6.5, která přinesla podporu pro datové sklady, jež v té době začaly vznikat. Tato verze byla první zcela splňující ANSI SQL standard.

V červnu roku 1998 byla veřejnosti uvolněna 3. beta verze SQL Server 7.0. V listopadu téhož roku byla oznámena finální verze, která šla do prodeje v lednu 1999. S verzí 7.0 vývoj neustal a brzy byla plánována verze 7.5, s kódovým označením Shiloh. Tato verze však nakonec vyšla pod názvem SQL Server 2000, aby se shodovala s tehdejší marketingovou strategií takzvaných backoffice produktů společnosti Microsoft. (3, 2011, online)



OBRÁZEK 1: NEJVÝZNAMNĚJŠÍ ETAPY VÝVOJE PRODUKTU MICROSOFT SQL SERVER (AUTOR)

Po vydání verze SQL Server 2000 (8.0) tempo vývoje zpomalilo. Následoval dodatek v podobě SQL Server 2000 Reporting Services, nástroje pro projekty z oblasti business intelligence. 4

Významným milníkem v historii produktu bylo vydání verze SQL Server 2005 v lednu roku 2006. 5 Za nejdůležitější nové vlastnosti lze považovat CLR, podporu XML, Integration Services a Report Builder. (3, online)

3.9 MICROSOFT SQL SERVER 2008 R2

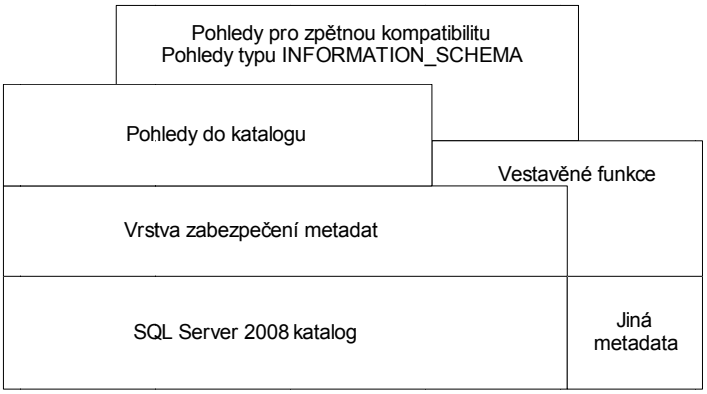
MS SQL Server 2008 R2 je k počátku roku 2012 poslední vydanou verzí.

3.9.1 ARCHITEKTURA

3.9.1.1 Metadata

Metadata jsou v SQL Serveru 2008 uložena v tabulkách *system base tables*. Některé tyto tabulky jsou uložené pouze v databázi Master, jiné jsou v každé z uživatelských databází. Systémové tabulky nejsou běžně přístupné a práce s nimi je možná pouze pomocí zvláštních nástrojů. (6, 2008, s.3)

Obrázek níže představuje strukturu metadat v SQL Serveru. Základem je katalog a nadstavbou speciální druhy pohledů. Mezivrstvou je zabezpečení. Mimo základní strukturu jsou jiné druhy metadat a vestavěné funkce, které jsou jen z části obsaženy v katalogu.



OBRÁZEK 2: SQL SERVER 2008, METADATA[6]

Kvůli zpětné kompatibilitě existují systémové pohledy *Compatibility Views*, které umožňují práci s metadaty aplikacím vytvořeným pro starší verze SQL Serveru. Pro příklad lze uvést tyto pohledy kompatibility:

- *Sysobjects*
- *Sysindexes*
- *Sysusers*
- *Sysdatabases*

Od verze 2005 jsou implementovány katalogové pohledy - Catalog Views, které jsou obecným rozhraním ke stálým metadatům. Jsou součástí schématu *sys*, na které je nutné se odkázat při práci s těmito pohledy. Pro příklad lze uvést tyto pohledy:

- *sys.objects*
- *sys.indexes*
- *sys.databases*

Shodnost jmen katalogových pohledů s pohledy kompatibility je zřejmá. Pohledy přesto zpřístupňují rozdílná data v rozdílné struktuře.

Definici pohledů na metadata, ale i všech ostatních objektů v databázi, lze zobrazit pomocí funkce *object_definition*.

SQL 5: SQL SERVER 2008, METADATA, FUNKCE OBJECT_DEFINITION

```
SELECT object_definition (object_id('sys.tables'));
```

Návratová hodnota funkce je text SQL definice objektu, jehož jméno bylo zadané jako argument. (6, 2008, s. 4)

```
CREATE VIEW sys.tables AS  SELECT o.name, o.object_id, o.principal_id, o.schema_id, o.parent_object_id, o.type, o.type_desc, o.create_date, o.modify_date,  o.is_ms_shipped, o.is_published, o.is_schema_published, lob.lobds AS lob_data_space_id,  rfs.indepid AS filestream_data_space_id,  o.property AS max_column_id_used,  o.lock_on_bulk_load, o.uses_ansi_nulls, o.is_replicated, o.has_replication_filter, o.is_merge_published, o.is_sync_tran_subscribed, o.has_unchecked_assembly_data,  lob.intprop AS text_in_row_limit,  o.large_value_types_out_of_row,  o.is_tracked_by_cdc,  o.lock_escalation_option AS lock_escalation,  ts.name AS lock_escalation_desc  FROM sys.objects$ o  LEFT JOIN sys.sysidxstats lob ON lob.id = o.object_id AND lob.indid <= 1  LEFT JOIN sys.syssingleobjrefs rfs ON rfs.depid = o.object_id AND rfs.class = 42 AND rfs.depsubid = 0 -- SRC_OBJTOFSDS  LEFT JOIN sys.syspalvalues ts ON ts.class = 'LEOP' AND ts.value = o.lock_escalation_option  WHERE o.type = 'U'
```

OBRÁZEK 3: SQL SERVER 2008, METADATA, NÁVRATOVÁ HODNOTA FUNKCE OBJECT_DEFINITION PRO ARGUMENT *SYS.TABLES*

3.9.1.2 Pohledy na schemata

Od verze SQL Server 7 jsou pro nahlížení na metadata k dispozici pohledy na schémata - Information Schema Views. Splňují standard SQL-92 a jsou kompatibilní se všemi SŘBD produkty splňujícími tento standard, tím je zajištěna přenositelnost.

Pro příklad lze uvést pohled INFORMATION_SCHEMA:

SQL 6: SQL SERVER 2008, POHLEDY NA SCHÉMATA, POHLED INFORMATION_SCHEMA

```
SELECT *
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_NAME = 'Inv_AeX_AC_Identification'
```

TABULKA 9: SQL SERVER 2008, POHLEDY NA SCHÉMATA, POHLED INFORMATION_SCHEMA

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
Symantec_CMDB	dbo	Inv_AeX_AC_Identification	BASE TABLE

3.9.1.3 Systémové funkce

Vlastnosti SŘBD, databází a jejích součástí lze zjistit pomocí systémových funkcí. Hodnoty navrácené těmito funkcemi jsou skalární. Pro příklad lze uvést tyto systémové funkce:

- SERVERPROPERTY;
- DATABASEPROPERTY;
- INDEXPROPERTY;
- OBJECTPROPERTY.

Příklad užití funkce INDEXPROPERTY

SQL 7: SQL SERVER 2008, SYSTÉMOVÉ FUNKCE, FUNKCE INDEXPROPERTY

```
SELECT INDEXPROPERTY(
 OBJECT_ID('dbo.inv_aex_ac_identification')
,'Idx_Inv_AeX_AC_Identification_Name_Domain'
,'IndexDepth')
```

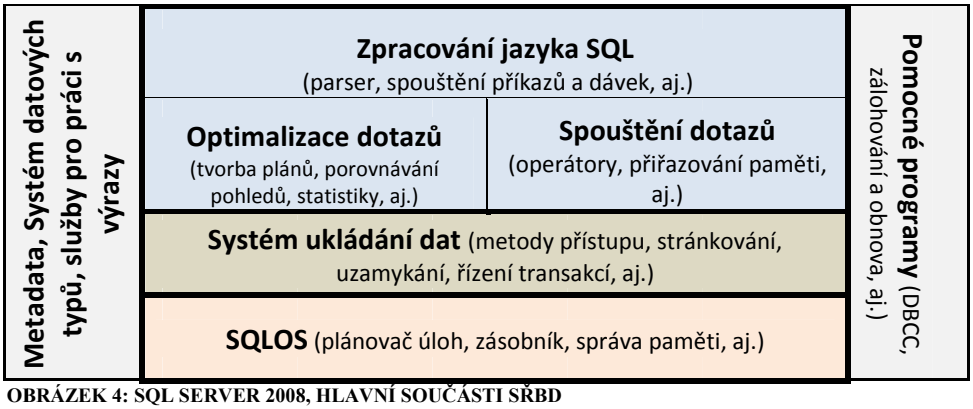
Funkce INDEXPROPERTY má 3 argumenty: identifikátor objektu, kterému index náleží, jméno indexu a vlastnost indexu. Vlastností indexů je celkem 13. V příkladu je uvedeno jméno základní vlastnosti *IndexDepth*, které informuje o počtu úrovní indexu. Funkce navrací numerické hodnoty. (6, 2008, s. 7)

3.9.1.4 SŘBD a jeho součásti

Architektura SŘBD SQL Server 2008 má čtyři hlavní části, tři z nich jsou zobrazené v tabulce níže. První součástí je relační subsystém, který má tři další součásti: zpracování jazyka SQL – Languague processing, optimalizace dotazů – Query optimization a spouštění dotazů – Query execution. Druhou součástí je systém pro ukládání dat – Storage Engine. Třetí součástí je SQLOS.

Čtvrtou součástí je vrstva protokolů, která zajišťuje rozhraní mezi klienty a serverem. Od klientských aplikací přebírá požadavky a překládá je do formátu, s kterým relační subsystém již může pracovat. Vrstva představující relační subsystém přijímá dávky T-SQL kódu a zpracovává je. Je-li při spuštění dávky zjištěna nutnost práce s daty, je zaslán požadavek systému ukládání dat. Veškerý přístup k datům a práce s nimi, včetně zálohování, nebo

hromadného vkládání dat je vykonáván právě systémem pro ukládání dat. Vrstva SQLOS zajišťuje operace, které jsou shodné s činností operačního systému: správa vláken, synchronizace procesů, detekce zámků, správa paměti a zásobníku. (6, 2008, s. 9)



OBRÁZEK 4: SQL SERVER 2008, HLAVNÍ SOUČÁSTI SŘBD

Pohledy typu: objekty dynamického řízení - *Dynamic Management Objects (views)*

SQL Server 2008 obsahuje objekty dynamického řízení, z nichž většina je typu pohled. Tyto pohledy se neskládají z tabulek. Jsou postaveny na interních strukturách SŘBD. Lze je využít pro diagnostiku, ladění operační paměti a procesů a sledování chodu celého systému. Pohledy dynamického řízení jsou rozděleny do několika kategorií dle využití. Pro příklad lze uvést kategorii *dm_db_**, která zprostředkovává informace o databázích a databázových objektech, jakými jsou například indexy. Jedním z užitečných pohledů z kategorie *dm_db_**, který lze využít při optimalizační analýze je *sys.dm_db_index_physical_stats*. Mezi informace, které podává, se řadí aktuální velikost a fragmentace indexů a hloubka indexu. (6, 2008, s. 10)

SQL 8: SQL SERVER 2008, DMV POHLEDY, PŘÍKLAD

```
SELECT
 index_id, index_type_desc, index_depth,avg_fragmentation_in_percent
,fragment_count,page_count
from sys.dm_db_index_physical_stats (
 DB_ID('symantec_cmdb')
,Object_id('inv_aex_ac_identification')
,NULL,NULL,NULL)
```

TABULKA 10: SQL SERVER 2008, DMV POHLEDY, POHLED

index_id	index_type_desc	index_depth	avg_fragmentation_in_percent	fragment_count	page_count
1	CLUSTERED INDEX	2	99.29577465	142	142
2	NONCLUSTERED INDEX	2	92.85714286	27	28
3	NONCLUSTERED INDEX	2	92.30769231	13	13
4	NONCLUSTERED INDEX	2	95.65217391	23	23

Protokoly

Při komunikaci aplikace se SŘBD je využit proprietární formát paketu - *tabular data stream (TDS)*. Vrstva protokolu *SQL Server Network Interface* (SNI) na straně klienta i serveru zabalí paket TDS do některého z běžných protokolů, jako například *Named Pipes*, nebo *TCP/IP*. Na straně serveru jsou síťové knihovny ovladačů součástí SŘBD, na straně klienta jsou součástí ODBC ovladačů *SQL Native Client*.

SŘBD SQL Server 2008 podporuje celkem čtyři protokoly, které lze provozovat paralelně.

Sdílená paměť – *Shared memory*, je nejjednodušší protokol. Lze jej využít pouze pro komunikaci mezi aplikacemi instalovanými ve stejném OS, hostícím SŘBD.

Named Pipes – protokol vyvinutý pro sítě LAN. Část paměti je užita procesem, který předává informace druhému procesu, který může být na jiném počítači v rámci lokální sítě.

TCP/IP – nejrozšířenější protokol Internetu. Výhodou je možnost navázání spojení mezi různými fyzickými sítěmi, díky možnostem směrování.

Virtual Interface Adapter (VIA) – speciální protokol určený pro použití s hardware výrobce VIA. (6, 2008, s. 11)

Relační subsystém – *relational engine*

Hlavní činností relační subsystému je zpracovávání dotazů – *SQL query*. V tabulce číslo 5 výše, je relační subsystém zobrazen ve třech součástech:

- Zpracovávání jazyka SQL – *Command parser*. Tato součást přijímá T-SQL zprávy, ve smyslu příkazů a událostí, zasílané SŘBD. Provede syntaktickou kontrolu a přeloží T-SQL kód do interního (strojového) formátu, který se nazývá *query tree*. Chyby

v syntaxi jsou nahlášeny a zpracovávání je pozastaveno. Chyby jiné, než syntaktické odhaleny v této fázi nejsou.

- Optimalizace dotazů – *Query optimization.* Tato součást přebírá zpracovaný kód a připravuje ho pro spuštění. Výroky, které nelze optimalizovat (například příkazy DDL) jsou kompilovány do jiného interního formátu. Výroky, které lze optimalizovat jsou označeny a postoupeny součásti vykonávající samotnou optimalizaci. Optimalizační součást analyzuje výroky T-SQL kategorie DML, které lze zpracovat více způsoby a vybírá ten nejlepší. Výsledkem analýzy je plán spuštění – *execution plan.*

- Spouštění dotazů – *Query execution.* Tato součást spustí obsah plánu spuštění krok po kroku a dohlíží na celkový průběh. Pro většinu operací vyžaduje součinnost systému ukládání dat pro výběry a manipulaci s daty, pro zamykání a řízení transakcí.

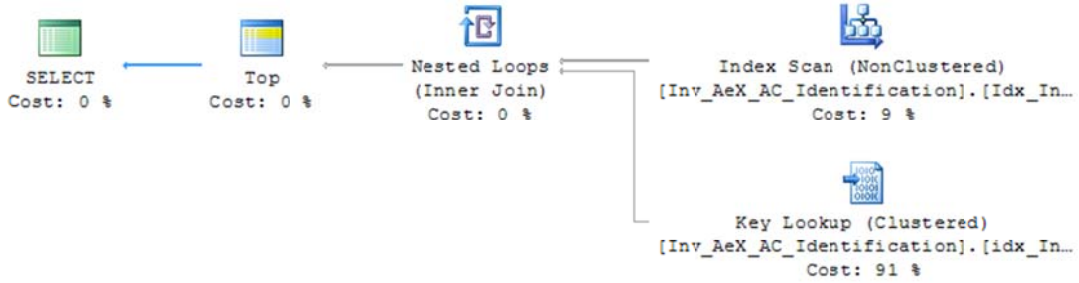
Vsuvka - Postup optimalizace

První krok v tvorbě plánu spuštění je *normalizace* každého kusu kódu SQL, a to dělením na již nedělitelné části. Druhý krok je výpočet nákladnosti každé operace, ve smyslu: kolik procesorového času, prostoru operační paměti a jaké množství vstupně/výstupních operací bude nutno pro úkon alokovat, respektive vykonat. Třetím krokem je určení typu výroku (DDL, DML, či jiné), zjištěním informací o dotčených objektech (tabulky, indexy, či jiné) a načtení vzorku dat z odkazovaného sloupce, nebo indexu. Dílčímu úkonu načítání vzorku dat se říká statistika rozdělení dat. Získané informace z těchto tří kroků se užijí pro sestavení plánu spuštění.

Následující tabulka obsahuje jednoduchý dotaz pro selekci s jednou podmínkou a řazením odspoda nahoru. Obrázek pod tabulkou zobrazuje grafickou část vzorového plánu spuštění tak, jak jej SŘBD sestavil pro daný dotaz. (6, 2008, s. 13)

SQL 9: SQL SERVER 2008, RELAČNÍ SUBSYSTÉM, DOTAZ PRO PLÁN SPUŠTĚNÍ

```
SELECT TOP 10 *
FROM Inv_AeX_AC_Identification
WHERE [Domain] = 'PRG-DC'
ORDER BY [Name] DESC
```



OBRÁZEK 5: SQL SERVER 2008, RELAČNÍ SUBSYSTÉM, VZOROVÝ PLÁN SPUŠTĚNÍ

Systém ukládání dat – *Storage engine*

V SŘBD SQL Server 2008 se systém ukládání dat skládá ze tří hlavních součástí. Jsou to: metody přístupu, podpůrné programy a služby uzamykání a transakcí. (6, 2008, s. 14)

Metody přístupu – *Access Methods*

Vždy, když SŘBD vyšle požadavek na přístup k datům, zavolá kód metod přístupu. Metody přístupu nastaví parametry prohledávání datových stránek a stránek indexů a připraví množiny řádků (*row set)* prostřednictvím OLE-DB. Samotná data jsou získávána ze zásobníku- *buffer manager*. Při čtení dat je užit mechanismus předběžného načítání řádků a částí indexů, které obsahují potřebná data.

Vsuvka – Object Linking and Embedding, Database (OLE-DB) [7]

OLE-DB je sada rozhraní založených na objektovém programovacím modelu COM, která zprostředkovává datové přenosy z různorodých zdrojů. OLE-DB poskytuje uniformní přístup k datům uloženým v různých datových úložištích, nebo typech databází. (7, 2012, online)

Metody přístupu lze rozdělit dle druhu operací následovně:

- Operace nad indexy a řádkami – *Row and Index Operations*. Kód pro práci s řádkami načítá, upravuje a jinak manipuluje se samotnými řádkami v tabulkách. Řádky jsou během manipulace uzamčené v rámci jedné transakce. Po úpravě dat náležících řádce v paměti, se data vrátí zpět do příslušné řádky (např. při příkazu UPDATE, nebo DELETE). Kód pro práci s indexy udržuje a zajišťuje prohledávání takzvaných B-sromů – *B-tree*, struktury indexů.

- Operace alokování stránek – *Page Allocation Operations.* Kód pro alokaci stránek spravuje sadu stránek pro každou databázi a udržuje přesný přehled o jejich aktuálním využití. Databáze je sada stránek – objektů o velikosti 8KB, které jsou rozmístěné na jednom, nebo více fyzickém souboru na pevném disku, či obecně na úložišti fyzických dat.

- Operace nad kopiemi řádků – *Versioning Operations.* Tento druh operací je novou vlastností a umožňuje udržovat starší verze řádků, jejich kopie. (6, 2008, s. 15)

Služby uzamykání a transakcí

Operace uzamykání – *Locking Operations*. Uzamykání je nutnou funkcí SŘBD, pro podporu práce více uživatelů najednou. Operace čtení nevyžaduje uzamykání oblasti databáze, ze které se čte. Operace zápisu uzamykání mít musí, aby nedošlo ke konfliktům při simultánní práci s identickou oblastí databáze. Existuje více typů zámků a kód obsluhující operace uzamykání mezi nimi zajišťuje kompatibilitu. Rovněž vyhodnocuje takzvané věčné zámky – *deadlocks* a eskalace zámků.

Transakční služby – *Transaction Services.* Klíčovou vlastností SŘBD SQL Server 2008 je schopnost zajistit, aby transakce byly atomické – tj. vždy úplné. Dále odolné proti přerušení a vnějším chybám. Transakce musejí vyhovět čtyřem podmínkám:

- Atomičnost – *atomicity.* Ve smyslu ”vše, nebo nic“. Jestliže je transakce dokončená příkazem COMMIT, všechny změny jsou uložené do databáze a nelze je vrátit zpět příkazem ROLLBACK. Naopak, jestliže je transakce zrušena, žádná změna se neprovede.

- Konzistence – consistency. Je zajištěna pravidly pro zachování integrity dat a logického úsudku. Například není možné mít záporné datum, nebo věk člověka.
- Izolace – *isolation.* Transakce a data, se kterými pracují, jsou oddělena od ostatních, právě probíhajících transakcí. Při nutnosti sdílení stejné oblasti databáze, SŘBD používá různé verze řádků ukládané v dočasné databázi.
- Odolnost – *durability.* Při výpadku jsou započaté transakce zapsány na disku a po obnovení aktivního stavu SŘBD se obnoví i rozdělané transakce.

SŘBD udržuje transakční zápis (*transaction log*), obsahující veškeré operace, které v rámci dané transakce byly vykonány a bezprostřední část těch operací, které teprve proběhnou. Ukončení transakce je provedeno příkazem COMMIT. V případě přerušení transakce je možné navázat díky informacím v transakčním zápisu. Práce s transakčním zápisem je asynchronní, práce se stránkami je synchronní. Řídící program transakčního zpracovávání vytváří takzvané body návratu – *save points.* To jsou místa v transakci, ke kterým je možný návrat, ať již pro opakované zpracování nebo zrušení části operace. (6, 2008, s. 17)

Podpůrné programy- *utility commands*.

Poslední součástí systému ukládání dat jsou různé podpůrné programy, pomocí kterých se například načítají data z textových souborů – *bulk load*, pracuje s textem, provádí zálohování a obnova databází. Velkou skupinou jsou takzvané DBCC příkazy. (6, 2008, s. 18)

SQL Operační systém – *SQLOS*

SQLOS je aplikační vrstva na nejnižší úrovni SŘBD. Hlavními dvěma funkcemi jsou správa operační paměti a plánování. Dodatečné funkce zahrnují synchronizaci, alokace a přidělování paměťových celků, správa výjimek, odhalování věčných zámků, sledování událostí a asynchronní vstupně výstupní operace. (6, 2008, s. 18)

3.9.1.5 Databáze

Databáze v teoretickém vymezení SQL Serveru 2008 je sada objektů, které obsahují a manipulují s daty. Sada objektů zahrnuje tabulky, pohledy, uložené procedury, funkce aj.

Systémové databáze

SŘBD SQL Server 2008 obsahuje čtyři systémové databáze: *master, model, tempdb* a *msdb*. Existuje ještě skrytá databáze, ke které nelze běžně přistoupit, její název je *mssqlsystemresource*. Význam systémových databází je odlišný od uživatelských databází:

- Systémová databáze *master* se skládá ze systémových tabulek, které obsahují informace o SŘBD a všech ostatních databázích. Obsahuje systémový katalog objektů a systémových součástí. Její existence je pro SŘBD nepostradatelná.
- Systémová databáze *model* je šablonou pro nové databáze. Při vzniku nové databáze SŘBD vytvoří kopii databáze *model*. Její datový model lze změnit a tím zajistit dědění vlastních nastavení pro všechny nové databáze.
- Systémová databáze *tempdb* je pracovní databází, která je pouze dočasná. Zaniká s ukončením SŘBD a je znovu vytvořena se spuštěním SŘBD. Obsahuje dočasné uživatelské tabulky, pracovní tabulky SŘBD obsahující data rozpracovaných úloh, průběžně ukládané verze řádkových sad a veškerá ostatní data dočasné povahy.

Systémová databáze *msdb* je používána službou *SQL Server Agent* a dalšími pomocnými službami. Je nezbytná pro speciální funkce SŘBD, jakými jsou plánované úlohy, výstrahy, předávání transakčního záznamu, interní pošty a obnovy poškozených stránek.

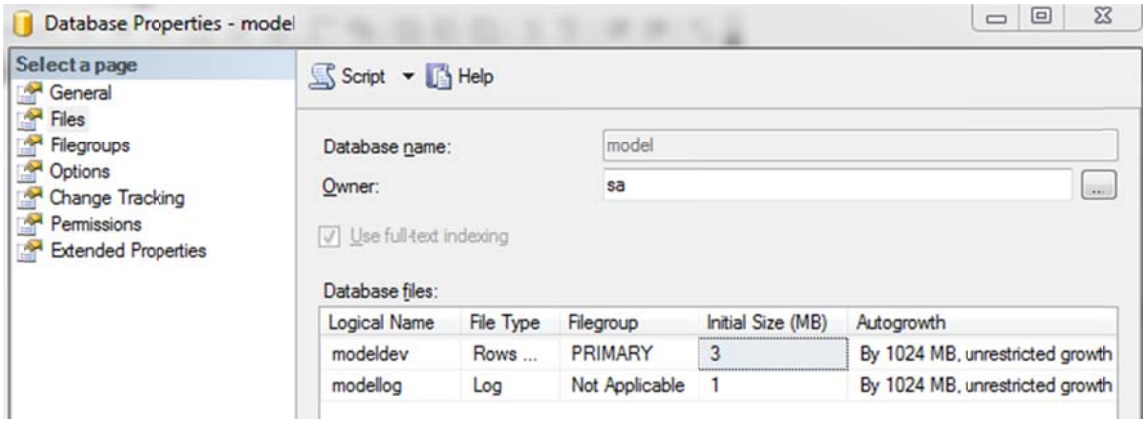
(6, 2008, s. 126)

Soubory databází – *Database files*

Soubor databáze je fyzický soubor operačního systému. Databáze je složená z minimálně dvou fyzických souborů, jednoho pro data a druhého pro transakční záznam. SŘBD SQL Server 2008 rozeznává tři typy souborů, kterými jsou:

- Prvotní datové soubory – *Primary data files*. Každá databáze má jediný primární datový soubor. Mimo samotných dat obsahuje informace o všech ostatních souborech databáze. Přípona primárního datového souboru je *.mdf*.
- Druhotné datové soubory – *Secondary data files*. Databáze může mít žádný, nebo více druhotných souborů. Druhotné soubory existují pro možnost rozdělení databáze na více fyzických úložišť operačního systému, předně kvůli zvýšení výkonosti. Přípona druhotných souborů je *.ndf*.
- Soubory transakčního záznamu – *Log files*. Každá databáze má alespoň jeden soubor transakčního záznamu, který obsahuje informace nutné pro obnovu všech transakcí v databázi. Přípona souborů transakčního záznamu je *.ldf*.

Každá databáze má pět základních vlastností: logický název, název fyzického souboru, výchozí velikost, maximální velikost a míru přírůstku. Informace o vlastnostech databázových souborů všech databází připojených k SŘBD jsou uložené v katalogu a přístupné prostřednictvím pohledu *sys.database_files*. (6, 2008, s. 130)



OBRÁZEK 6: SQL SERVER 2008, DATABÁZE, VLASTNOSTI

SQL 10: SQL SERVER 2008, DATABÁZE, PŘÍKAZ PRO POHLED NA METADATA

```
SELECT
 df.name
,df.[type]
,df.[size]
,df.[growth]
,df.type_desc
,df.physical_name
FROM sys.database_files AS df
```

TABLE 1: SQL SERVER 2008, DATABÁZE, POHLED NA METADATA

name	type	size	growth	type_desc	physical_name
modeldev	0	288	131072	ROWS	e:\CZCHOWV266_SQL135\SQLData\MSSQL10.SQL135\MSSQL\DATA\model.mdf
modellog	1	96	131072	LOG	e:\CZCHOWV266_SQL135\SQLData\MSSQL10.SQL135\MSSQL\DATA\modellog.ldf

3.9.1.6 Tabulky

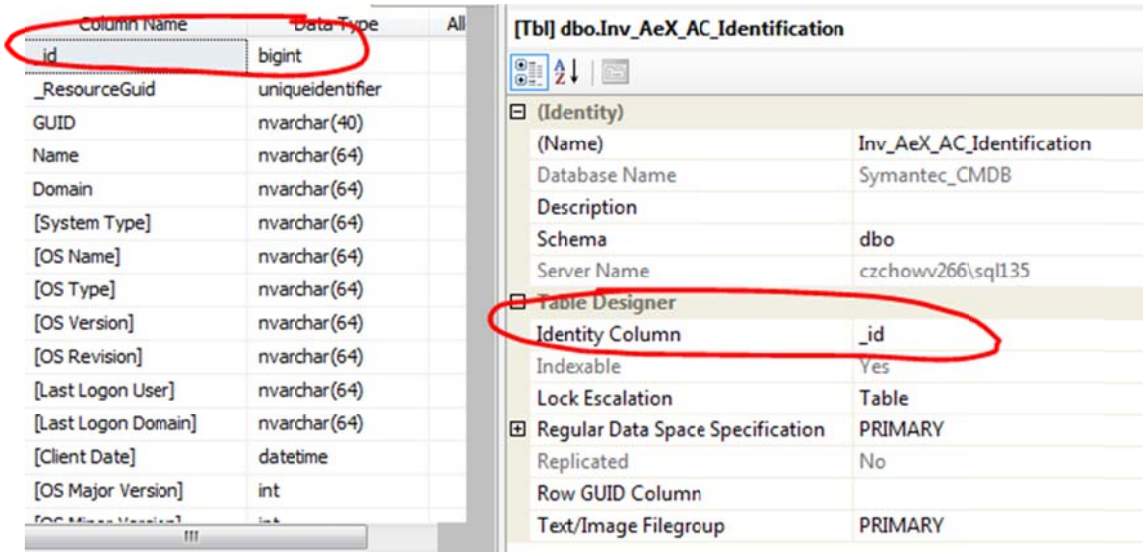
Tabulka je soubor dat vztahujících se ke konkrétní entitě (osobě, místu, věci) a má konečné množství pojmenovaných atributů. Je základním prvkem SŘBD SQL Server 2008 a relačního datového modelu.

Atributy mají formu pojmenovaných sloupců. Každý jednotlivý záznam dat má formu řádky. Dle pravidel relačního modelu je každý řádek tabulky jedinečný a je jednoznačně identifikován. Primární klíč je takový atribut, který zajišťuje jedinečnost řádku.

Většina tabulek má vztah k ostatním tabulkám v databázi. Pro navázání relací mezi tabulkami existují cizí klíče – *foreign keys.* Cizí klíče zajišťují referenční integritu. (6, 2008, s. 212)

„Referenční integrita databáze garantuje korektnost vztahů mezi logicky souvisejícím tabulkami. Vazby mezi logicky nadřízenými a podřízenými záznamy v databázi jsou vytvářeny doplněním jednoznačné identifikace řádků logicky nadřízené tabulky do logicky podřízené tabulky pomocí tzv. *cizího klíče (foreign key).*“ (Vostrovský, 2004, s. 126).

V SQL Serveru 2008 se sloupec obsahující data primárního klíče nazývá *Identity Column*. Na obrázku níže je uveden příklad možného primárního klíče, s datovým typem *int*.



OBRÁZEK 7: SQL SERVER 2008, TABULKY, PŘÍKLAD PRIMÁRNÍHO KLÍČE

SQL 11: SQL SERVER 2008, TABULKY, PŘÍKLAD TABULKY DATABÁZE SYMANTEC_CMDB

```
SELECT TOP 5

 id._id, id.[Name], id.Domain, id.[OS Name],id.FQDN,id.[Hardware Serial
Number]

FROM Inv_AeX_AC_Identification AS id
```

TABULKA 11: SQL SERVER 2008, TABULKY, PŘÍKLAD VYBRANÝCH DAT Z TABULKY

_id	Name	Domain	OS Name	FQDN	Hardware Serial Number
71413	DEBONWNC6169150	PRG-DC	Windows 7 Enterprise	DEBONWNC6169150.prg-dc.dhl.com	654ZD4J
71156	HKZUHWN2000004	KUL-DC	Windows 7 Enterprise	HKZUHWN2000004.kul-dc.dhl.com	CND0141NP5

3.9.1.8 Indexy

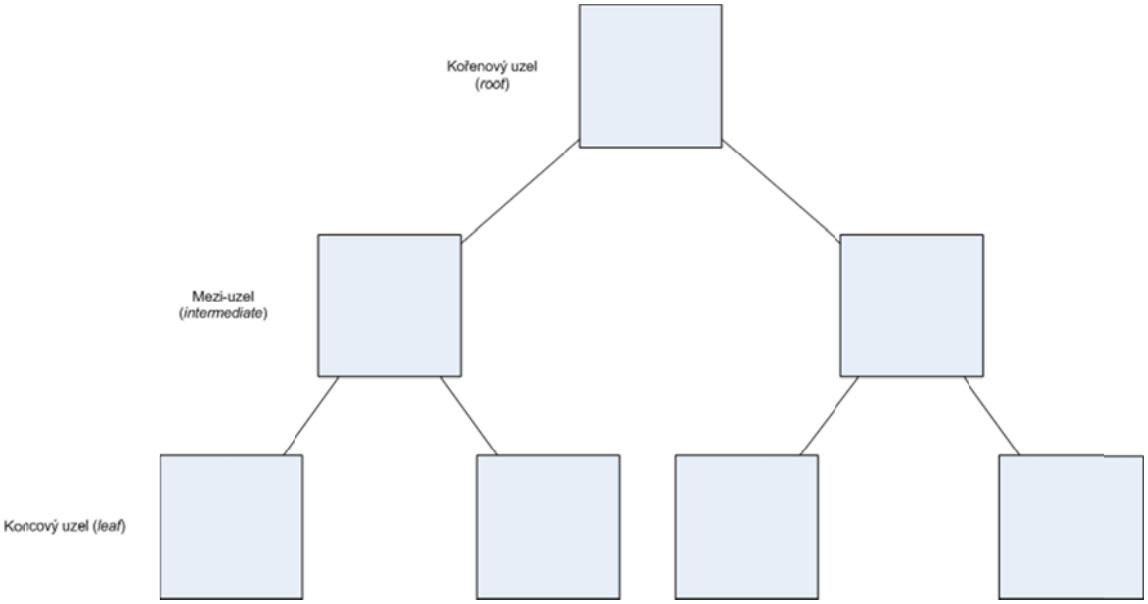
Dle (Hotek, 2009, s. 87), je index ve své podstatě jmenný rejstřík. Stejně jako v tištěné knize, kde usnadňuje vyhledávání konkrétního textu na konkrétní straně, tak SQL Server index usnadňuje vyhledávání relevantních řádků v tabulce.

Souměrné stromy – *B-trees*

Implementace indexů je v SŘBD SQL Server 2008 realizována koncepcí zvanou *souměrné stromy – balanced trees* (B-trees). Souměrný strom je konstruován z těchto prvků:

- z kořenového uzlu, který obsahuje jednu stránku dat,
- jednoho, nebo více mezi-uzlů

a jednoho, nebo více koncových uzlů, takzvaných listů – *leafs*.



OBRÁZEK 8: SQL SERVER 2008, INDEXY, B-STROM

Z obrázku nahoře je patrná symetrie struktury indexu. Na každé straně je stejné množství uzlů, index je tedy rozdělen na dvě poloviny.

Uzly indexu se nazývají stránky a obsahují seřazené záznamy v pořadí určeném při tvorbě indexu. Data na stránkách koncových uzlů obsahují každou kombinaci údajů ve sloupci, který

je indexem opatřen. Počet záznamů na stránce indexu je dán datovým typem sloupce, respektive požadavkem datového typu na úložný prostor.

Kořenový uzel a mezi-uzly indexu jsou tvořeny tak, že první záznam na jejich stránce je prvním záznamem podřazeného uzlu. Zároveň obsahují pointer na uzel, kde se odkazovaný první záznam nachází.

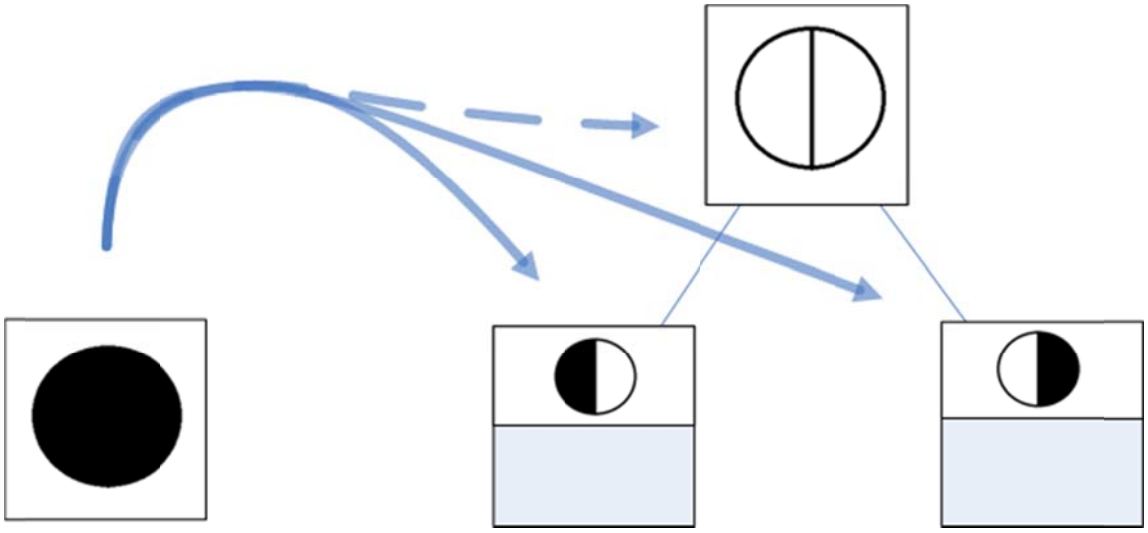
Vsuvka – procházení indexu

SQL dotaz (*query*) prochází stránku kořenového uzlu indexu, dokud nenajde odkaz na stránku, která obsahuje hledaný výraz. Použije pointer na další stránku indexu – mezi-uzlu, nebo koncového uzlu. Na další stránce opět prochází záznamy o odkazovaných hodnotách, dokud nenajde hledaný výraz, nebo odkaz na další stránku indexu a použije pointer. Takto se proces opakuje, dokud není nalezeno místo hledaného výrazu v tabulce. (Hotek, 2009, s. 89)

Hloubka indexu – *index level* [8]

Počet úrovní indexu a počet stránek v každé úrovni indexu je dán datovou kapacitou stránky v SQL Serveru a nároky na úložný prostor daného datového typu ve sloupci, pro který index existuje. Velikost datové stránky SQL Serveru je 8 192 bajtů, každá stránka pojme 8 060 bajtů.

Jsou-li nároky celočíselného datového typu *int* 4 bajty, pak obsahuje-li tabulka pouze 1 200 řádků, bude nutno uložit 4 800 bajtů dodatečného místa pro data stránek indexu. V tomto případě by byla jenom jedna stránka indexu – kořenový uzel by byl zároveň uzlem koncovým. Takto by to platilo, dokud by tabulka měla méně, nebo rovno 2015 řádků (4 x 2015 = 8060). Při přidání 2016. řádku, dojde k procesu nazvanému dělení stránek – *page splitting*. SQL Server vezme polovinu obsahu stránky indexu a přesune ji na novou stránku, vytvoří odkazy na obě stránky a zapíše je na druhou novou stránku, která se stane novým kořenovým uzlem.



OBRÁZEK 9: SQL SERVER 2008, INDEXY, DĚLENÍ STRÁNEK

Tento proces může pokračovat do celkového počtu 4 060 225 řádků v tabulce. V té chvíli bude v kořenovém uzlu 2015 záznamů, odkazů na celkem 2015 koncových uzlů, které mají 2015 záznamů každý (2015 x 2015 = 4 060 225). Jakmile se ale do tabulky přidá další jeden řádek, musí dojít k dělení stránek, tím se zvětší počet uzlů a stránka kořenového uzlu indexu by nepojmula 2016. záznam. SQL Server musí přidat další úroveň – mezi-uzly. Předchozí kořenový uzel se stane mezi-uzlem, polovina záznamu se překopíruje na druhý mezi-uzel a nový kořenový uzel bude mít záznamy o nové struktuře. (Hotek, 2009, s. 90)

Druhy indexů

SŘBD SQL Server 2008 rozeznává celkem tři základní druhy indexů:

- Složený index – *Clustered index*. Skládá se ze dvou a více sloupců, které tvoří tzv. skládací klíč (*clustering key*). Lze jej složit z maxima 16 sloupců, přičemž suma požadavků na úložný prostor všech datových typů vybraných sloupců nesmí přesáhnout 900 bajtů. V případě výběru sloupců datového typu *int* by to bylo až 225 sloupců, avšak v případě datového typu *datetime*, který má požadavek 8 bajtů by to bylo „jen“ 112 sloupců tvořících klíč složeného indexu. Hlavním významem složeného indexu je třídění. Protože data v tabulce je možné třídit pouze jedním způsobem, lze ke každé tabulce vytvořit maximálně jeden složený index. Složený index obsahuje na stránkách koncových uzlů vlastní řádky tabulky. Najde-li SQL

Server hledané záznamy, našel i samotná data a není již nutné znovu číst data ze sloupců tabulky.

- Jednoduchý index – *Nonclustered index*. Pro jednoduché indexy existují stejná omezení jako pro složené indexy, s výjimkou toho, že k jedné tabulce lze vytvořit maximálně 1000 jednoduchých indexů. Význam jednoduchých indexů není ve třídění dat, ale urychlování vyhledávání dat ve sloupcích tabulek. Existuje-li na tabulce složený index, pak odkaz na stránce koncového uzlu jednoduchého indexu směřuje na skládací klíč složeného indexu. Neexistuje-li složený index, pak odkaz směřuje na konkrétní řádku tabulky.

- XML index. Speciálním druhem indexu je XML index, který lze použít pouze pro datový typ XML. Existují dva typy XML indexu: prvotní a druhotný. Prvotní XML index je vytvořen pro všechny prvky XML dokumentu (XML dat ve sloupci tabulky). Prvotní XML index vyžaduje, aby existoval složený index, na který se odkazuje. Druhotný XML index se vytváří pro vlastní prvky (*XML elements)*, tedy PATH, VALUE a PROPERTY.

Existují další, speciální druhy indexů, které se nejčastěji používají pro vylepšení výkonu operací nad daty. Patří mezi ně: Krycí indexy – *Covering indexes* a Filtrované indexy – *Filtered indexes*, jejichž použití je popsáno v následující kapitole. Málo běžným typem indexu je Prostorový index – *Spacial index*, který nachází využití v databázích pro systémy GIS, které vyžadují datové typy *geometry* a *geography*. (Hotek, 2009, s. 95)

Možnosti indexů

Při tvorbě indexu lze zadat hodnoty parametrů, které mají vliv na budoucí funkci indexu. Nejdůležitějším parametrem je FILLFACTOR, který určí kolik volného místa bude SŘBD udržovat v každé stránce koncového uzlu indexu. Volné místo umožní zápis záznamů a omezí se tím nutnost dělení stránek indexu (*page splitting*), dojde-li k přidání řádků do tabulky. Tímto způsobem lze předcházet fragmentaci indexů, ke které dochází při příliš častém dělení stránek. Parametr FILLFACTOR se specifikuje jako procento volného místa, které má vzniknout při tvorbě, nebo přestavbě indexu a je aplikován pouze na koncové uzly. Hodnotu

parametru FILLFACTOR pro stránky kořenového uzlu a mezi-uzly nastaví SŘBD automaticky na hodnotu 100 a tím se stránka vyplní zcela.

Existuje možnost, jak obejít omezení parametru FILLFACTOR na koncové uzly, a to parametrem PAD_INDEX. Tento parametr povolí ponechání volného místa ve stránkách na nadřazených uzlech. Parametr PAD_INDEX je vhodné užít jen při velkém počtu koncových uzlů.

Další možnosti zahrnují parametr SORT_IN_TEMPDB, který všechna pracovní data spojená s tvorbou a údržbou indexu ukládá v dočasné databázi. Parametr UNIQUE určí, že ve stránkách indexu budou uložena pouze jedinečná data. Při pokusu o vložení duplicitního záznamu, dojde k chybě a k vrácení transakce. Toto chování lze změnit parametrem IGNORE_DUP_KEY, který změní chování SŘBD při zjištění duplicity tak, že duplicitní záznam bude ignorován a transakce bude pokračovat. Informace o vypuštěné duplicitní hodnotě bude uložena do seznamu událostí. (Hotek, 2009, s. 97)

3.10 Optimalizační přístupy, předmět optimalizace

Tabulka, pro kterou není vytvořený žádný index se nazývá kopa dat (*heap*). Pro SQL Server je náročné s takovými tabulkami pracovat, proto by vždy měl existovat alespoň jeden složený a jeden jednoduchý index. Tato kapitola obsahuje teoretická východiska pro optimalizaci výběrů dat, nejen pomocí indexů.

3.10.1 Optimalizace pomocí indexů

Indexy mohou značně vylepšit výkonnost dotazů, ale zároveň zatěžují SŘBD při operacích měnících data v tabulkách, jako INSERT, UPDATE, DELETE, BULK INSERT. Je nutné vytvořit takové indexy a v takovém množství, které přinese vyvážené výsledky. „Obecné pravidlo říká, že existuje-li více než pět indexů pro tabulku používanou při OLTP operacích, je nutné analyzovat jejich přínos“ (Hotek, 2009, s. 96).

Tabulky obsahující data, která se aktualizují jen jednou za čas, jednorázově, mohou mít indexů více.

3.10.1.2 Speciální druhy indexů

Krycí indexy – *Covering indexes*. Krycí indexy jsou takové indexy, které pokryjí potřebu po datech, která SQL dotaz vyžaduje. Indexy jsou v podstatě části samotné tabulky, a tak obsahují data sloupce (klíče indexu), ke kterému patří. Vytvoří-li se index ke sloupci, ze kterého je nejčastěji čteno, SŘBD většinu dotazů uspokojí použitím indexu. Tím se urychluje přístup k datům a zkracuje čas na zpracování dotazu.

Zahrnuté sloupce – *Included Columns.* Použitím tzv. zahrnutých sloupců lze obejít omezení počtu sloupců, ze kterých lze vytvořit klíč indexu (16). Lze tak učinit použitím parametru INCLUDE při tvorbě, nebo přestavbě indexu. Zahrnuté sloupce se tak stanou součástí stránek koncových uzlů indexu. Hodnoty zahrnutých sloupců se nevyskytují jako záznamy na stránce v kořenovém uzlu indexu, ani na stránkách mezi-uzlů. Nezapočítávají se tak do limitu 900 bajtů pro datovou šířku indexu.

Filtrované indexy – *Filtered indexes*, lze použít v případech, kdy rozdělení dat v klíči indexu je šikmé. K indexu se připojí podmínka WHERE a eliminují se tak opakující se hodnoty. Filtrované indexy mají tato tři omezení:

- nelze je použít pro jednoduché indexy,
- nelze je přidat na spočetné sloupce,
- klíče indexu nelze konvertovat na jiné datové typy.

Statistika rozdělení dat – *Distribution Statistics*. Součástí indexu je histogram rozdělení dat v rámci sloupce, který SŘBD vytvoří spolu s indexem. Histogram představuje selektivitu indexu, což je stupeň schopnosti vyhledat konkrétní data v rámci sloupce tabulky. Selektivita indexu roste přímo úměrně s množstvím jedinečných hodnot ve sloupci. Součást relačního subsystému SŘBD, optimalizátor dotazů, vybírá ty nejselektivnější indexy při zpracovávání SQL dotazu. (Hotek, 2009, s. 97)

3.10.1.3 Údržba indexů

Během času dochází v důsledku manipulace s daty v tabulkách ke fragmentaci indexů. Fragmentace má přímý negativní vliv na výkonnost SQL dotazů a práci SŘBD obecně. K fragmentaci dochází například při vymazání řádku, nebo řádků z tabulky. V indexu vznikne prázdné místo, které není automaticky zaceleno, nebo jinak využito. Automatické opravy

prázdných míst by stály SŘBD příliš drahocenných prostředků. Míru fragmentace lze částečně ovlivnit použitím parametru FILLFACTOR při tvorbě indexu. Vzniklou fragmentaci indexů již lze pouze odstranit v rámci údržby SŘBD a jeho databází.

Defragmenace indexů

K defragmentaci indexů slouží příkaz ALTER INDEX. Syntaxe příkazu je znázorněna níže:

SQL 12: OPTIMALIZAČNÍ PŘÍSTUPY, SYNTAXE SQL DDL PRO DEFRAGMENTACI INDEXU

```
ALTER INDEX { index_name | ALL }
    ON <object>
    { REBUILD
        [ [PARTITION = ALL]
                    [ WITH ( <rebuild_index_option> [ ,...n ] ) ]
          | [ PARTITION = partition_number
                [ WITH ( <single_partition_rebuild_index_option>
                        [ ,...n ] )]]]
    | DISABLE| REORGANIZE
        [ PARTITION = partition_number ]
        [ WITH ( LOB_COMPACTION = { ON | OFF } ) ]
  | SET ( <set_index_option> [ ,...n ] ) }
```

Příkaz přijímá parametry PAD_INDEX, FILLFACTOR, SORT_IN_TEMPDB a IGNORE_DUP_KEY popsané v předchozí kapitole.

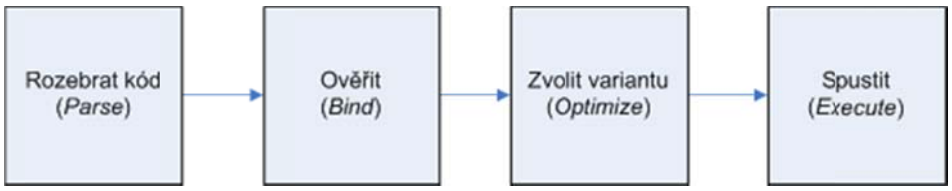
Index lze defragmentovat ve dvou režimech:

- Přestavění indexu – *REBUILD*. Tento režim postaví index znovu a naplní stránky indexu dle hodnoty parametru FILLFACTOR. Bez upřesnění se přestaví pouze složený index, s upřesněním ALL se přestaví i všechny jednoduché indexy. Během přestavby indexu je tabulka, ke které index patří, nedostupná pro jakoukoliv manipulaci s daty. To je zajištěno zámkem.

Přeuspořádání indexu – *REORGANIZE*. Tento režim odstraní fragmentaci pouze na stránkách koncových uzlů indexu. Lze jej však použít i za chodu SŘBD, kdy se s tabulkou, k níž index náleží, pracuje. (Hotek, 2009, s. 105)

3.10.2 Optimalizátor SQL dotazů – Query Optimizer

Optimalizátor SQL dotazů je součástí relačního subsystému SŘBD a jeho hlavní činností je sestavování plánů pro spuštění dotazů. Základní proces optimalizace je následující sekvence:



OBRÁZEK 10: OPTIMALIZÁTOR SQL DOTAZŮ, SEKVENCE ZÁKL. PROCESU (DELANEY, 2009, S.443)

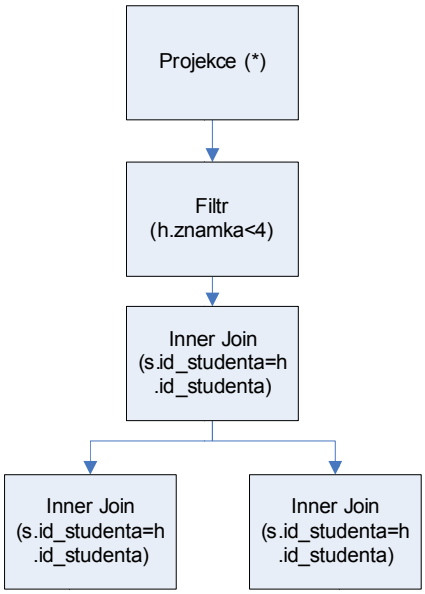
V prvním kroku je SQL kód rozebrán do stromové struktury. V druhém kroku je kód ověřen po sémantické stránce, včetně kontroly existence odkazovaných objektů v databázi. Ve třetím kroku je již ověřený kód použit pro výběr nejlepšího způsobu spuštění. Čtvrtým krokem je samotné spuštění kódu. (6, 2009, s. 443)

3.10.2.1 Stromový formát

SQL kód zpracovávaný relačním subsystémem je uspořádán do stromového formátu. Každý prvek stromu představuje jednu operaci, která bude vykonána. Například každá tabulka odkazovaná v sekci FROM, nebo každá podmínka WHERE má vlastní prvek. Operace spojování JOIN je jedním prvkem, ze kterého je tolik výstupů, kolik dalších objektů spojuje. Na obrázku níže je znázorněn příklad stromového formátu, pro zpracování dotazu z kapitoly Základní relační operace, s přidanou podmínkou WHERE.

SQL 13: OPTIMALIZAČNÍ PŘÍSTUPY, STROMOVÝ FORMÁT, VZOROVÝ SQL DOTAZ

```
SELECT s.id_studenta, obor, znamka
FROM studenti AS s
JOIN hodnoceni AS h
ON s.id_studenta = h.id_studenta

WHERE znamka < 4
```



OBRÁZEK 11: OPTIMALIZAČNÍ PŘÍSTUPY, STROMOVÝ FORMÁT, PŘÍKLAD STROMOVÉHO ROZVRŽENÍ STRUKTURY DOTAZU

Při kompilaci je logická forma stromové struktury nahrazena fyzickou formou. Například logické spoje (JOIN) jsou nahrazeny operacemi *hash join, merge join* nebo *nested loops join*). (6, 2009, s. 444)

3.10.2.2 Hledání efektivního plánu spuštění

Není možné vybrat optimální plán spuštění, ale pouze sub-optimální. Důvodem je příliš vysoký počet alternativ, zejména obsahuje-li dotaz mnoho spojovaných tabulek. Počet možností spojení pak přesahuje N!, kde N je počet tabulek. Není možné posoudit všechny alternativy také proto, že by takto složitá úloha přesáhla fyzické výpočetní možnosti SŘBD. Proto se užívá metod heuristiky a statistiky, které jsou stručně popsány v následující kapitole. (6, 2009, s. 445)

3.10.2.3 Pravidla

Optimalizátor dotazů je ve své podstatě soubor vyhledávacích nástrojů. Vyhledávání se řídí speciálními pravidly, která se dělí následovně:

- Substituční pravidla – *Substitution rules*. Vychází z principu heuristiky a určují jak přetvořit SQL kód ve stromovém formátu do nového tvaru.

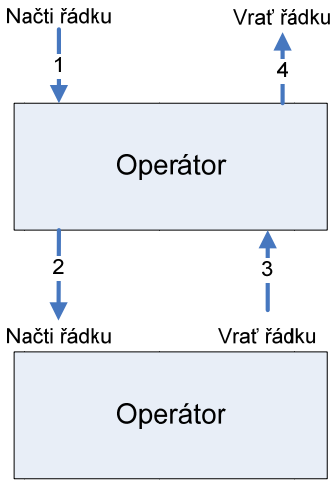
- Explorační pravidla – *Exploration rules.* Jsou založena na matematických větách komutativnosti sčítání a násobení „pořadí sčítanců při součtu, resp. pořadí činitelů při násobení můžeme zaměnit“ (Bušek,1992,s. 15).
- Implementační pravidla – *Implementation rules*. Určují, jakým způsobem se logická forma stromové struktury SQL kódu převede na fyzickou formu. (6, 2009, s. 446)

3.10.2.4 Vlastnosti

Optimalizátor sbírá informace o každém prvku stromu, tedy samostatné části SQL kódu, uspořádané do stromové struktury. Tyto informace jsou vlastnostmi prvku, které napomáhají výběru dalšího způsobu zpracování kódu. Například je-li použit příkaz GROUP BY pro sloupce, které obsahují pouze jedinečná data, je tento příkaz vypuštěn. Jedinečnost dat by v tomto případě například vyplývala s integritního omezení – parametru UNIQUE pro definici sloupce tabulky. (6, 2009, s. 447)

3.10.2.5 Operátory

SQL Server 2008 disponuje 40 logickými a 50 fyzickými operátory. Logické operátory pracují s logickou formou SQL kódu ve stromové struktuře. Fyzické operátory pracují s fyzickou formou SQL kódu a jmény interních objektů v databázi.



OBRÁZEK 12: OPTIMALIZAČNÍ PŘÍSTUP, PRÁCE OPERÁTORŮ, (DELANEY, 2009, S. 450)

Každý operátor v SQL Serveru funguje stejným způsobem: vyšle požadavek přímo podřízenému operátoru a po jeho přijetí vykoná vlastní operaci a výsledek předá nadřízenému

operátoru (respektive uživateli, neexistuje-li žádný nadřízený operátor). Každý operátor pracuje s jediným řádkem dat v jedné chvíli. Mezi nejdůležitější operátory se řadí:

- *Compute Scalar – Project*. Jednoduchý operátor, jehož hlavní funkcí je operace projekce, tedy práce se sloupci.
- *Compute Sequence – Sequence Project.* Podobně jako předchozí operátor je určen k práci se sloupci. Pracuje s řazeným proudem dat a udržuje stav zpracování pro každý řádek, čímž se liší od předchozího operátoru.
- *Semi-join.* Operátor pro operace spojování, který předává hodnoty pouze z jedné ze spojovaných tabulek. Nejčastěji je použit pro vnořené dotazy.
- *Apply.* Operátor pro práci s vnořenými dotazy. Má dvě formy: CROSS APPLY a OUTER APPLY, které jsou alternativou pro INNER JOIN a OUTER JOIN respektive. Liší se možností předávaní hodnoty parametru vnořenému dotazu. Je paralelou k volání funkce v procedurálním programování, kde pro každou řádku z vnější strany je vykonána operace na vnitřní straně a žádný, nebo více řádků je předáno volané funkci. Je také nazýván *correlated nested loops join*.
- *Spools.* Koncept operátorů typu *spool* je načíst sadu řádků ze vstupního operátoru, uložit výstup do paměti, nebo na disk a umožnit podřízeným operátorům přístup k uloženým datům. (6, 2009, s. 455)

3.10.2.6 Statistiky, odhady kardinality a nákladů

Optimalizátor SQL dotazů využívá modelu odhadu nákladů vynaložených na vykonání operace každým z operátorů. Náklady jsou vyjádřeny statistickými údaji, které slouží k odhadu počtu řádků, které každý operátor bude muset zpracovat. Ve výchozím nastavení optimalizátoru SQL dotazů se statistická data sbírají automaticky během zpracovávání dotazů. Optimalizátor samostatně identifikuje ty sloupce, pro které se statistické údaje budou sbírat.

Statistické údaje jsou buď vygenerované za chodu, nebo je užito uložených hodnot, vygenerovaných dříve. Před použitím starších statistik je určeno jejich stáří a příliš staré

údaje jsou vygenerovány znovu. Generování údajů za chodu může zpomalit chod SŘBD, lze proto tuto vlastnost řídit následovně:

- Vypnutím automatické tvorby statistik – ALTER DATABASE … SET AUTO_CREATE_STATISTICS OFF
- Vypnutím automatické aktualizace statistik – ALTER DATABASE … SET AUTO_UPDATE_SATISTICS OFF
- Zapnutím režimu aktualizace statistik na pozadí, pod jiným vláknem – ALTER DATABASE … SET AUTO_UPDATE_STATISTICS_ASYNC ON

Po vypnutí automatické tvorby a aktualizace statistik, je nutné tyto úkony naplánovat v rámci údržby databáze.

Statistická data jsou uložena v tabulkách pro metadata. Hlavní veličinou je rozdělení dat, reprezentované histogramem. Oborem hodnot je vždy jeden sloupec. Mezi ostatní informace patří hustota rozdělení dat, velikost vzorku dat, tedy počet řádků a datum sběru vzorku dat. Speciálním případem veličiny rozdělení dat jsou takzvané prefixové stromy - *trie trees*, které se užívají pro sloupce s datovým typem řetězec – *string.* Statistiky lze vytvořit pro sloupce s takovým datovým typem, který podporuje operace porovnávání.

Statistické údaje jsou tvořeny funkcí STATMAN. V případě malých tabulek, jsou statistiky vypočítány pro všechny řádky. U velkých tabulek je vytvořen vzorek z 200 řádků. (6, 2009, s. 462)

3.10.2.7 Přístup k hodnotám statistik

Pro každý dotaz lze získat statistické informace příkazem SET STATISTICS PROFILE ON. Objekty statistik lze pro každou tabulku zobrazit dotazem do tabulky *sys.stats*. Metadata objektů statistik lze procházet pomocí příkazu DBCC SHOW_STATISTICS. Příklad statistických dat, vypsaných pomocí příkazu DBCC SHOW_STATISTICS je v následujících třech tabulkách:

TABULKA 12: OPTIMALIZAČNÍ PŘÍSTUPY, STATISTICKÁ DATA - VÝSTUP 1

Name	Updated	Rows	Rows Sampled	Steps	Density	Average key length	String Index	Filter Expression	Unfiltered Rows
WA_Sys	Mar 23 2012 9:01PM	1409	1409	9	0	40.44429	YES	NULL	1409

TABULKA 13: OPTIMALIZAČNÍ PŘÍSTUPY, STATISTICKÁ DATA - VÝSTUP 2

All density	Average Length	Columns
0.1111111	40.44429	OS Name

TABULKA 14: OPTIMALIZAČNÍ PŘÍSTUPY, STATISTICKÁ DATA - VÝSTUP 3

RANGE HI KEY	RANGE ROWS	EQ ROWS	DISTINCT RANGE ROWS	AVG RANGE ROWS
Microsoft Windows Server 2003	0	8	0	1
Microsoft Windows Server 2003 R2	0	13	0	1
Microsoft Windows XP	0	6	0	1
Windows 7 Enterprise	0	1374	0	1
Windows 7 Professional	0	1	0	1
Windows Server (R) 2008 Enterprise	0	1	0	1
Windows Server (R) 2008 Standard	0	1	0	1
Windows Server 2008 R2 Enterprise	0	1	0	1
Windows Server 2008 R2 Standard	0	4	0	1
			CELKEM	**9**

3.10.2.8 Veličiny hustoty a rozdělení dat

Optimalizátor vedle histogramu také sleduje počet jedinečných hodnot ve skupině sloupců. Tyto hodnoty, zkombinované s počtem řádků podávají informaci o průměrném množství duplicitních hodnot ve sloupci. Informace o duplicitách se v terminologii SQL Server 2008 nazývají informace o hustotě – *density information.* Pro výpočet hustoty se používá vzorec *1/frequency*, kde frekvence (*frequency*) představuje průměrný počet výskytů každé konkrétní hodnoty ve sloupci tabulky. V příkladu výše vystupuje hustota pod názvem sloupce *All density* (její hodnota je 0.1111111, pak 1/0.111111 je přibližně 9, což je počet jedinečných hodnot ve sloupci). (6, 2009, s. 467)

3.10.2.9 Odhady kardinality

Kardinalita v terminologii SQL Serveru 2008 znamená počet řádků, které konkrétní operátor zpracovává. Pojem vychází z teorie množin a udává mohutnost množiny, ve smyslu počtu jejích prvků. „Mohutnost množiny (také kardinalita množiny) je pojmem teorie množin vyjadřující velikost konečných, ale i nekonečných množin. Značí se většinou |M|." (Balcar, 2001,s. 77).

Hodnota kardinality je vypočtena z hodnot histogramu, konkrétně počtu řádků ve vzorku, ten je porovnán s počtem řádků užitým pro složení histogramu a je vypočítána selektivita. Tou je poté vynásoben počet řádků v tabulce a výsledkem je odhadovaná kardinalita pro daný SQL dotaz. (6, 2009, s. 470)

3.10.2.10 Výběr indexu

Výběr správného indexu patří k nejdůležitějším krokům optimalizace výběrů dat. Základní myšlenkou je aplikace omezující podmínky v SQL dotazu na existující indexy definované k použitým tabulkám. Základní operace, které lze takto nad indexem provést jsou dvě:

- Vyhledávání – *seek*. Hledání konkrétní hodnoty, nebo hodnot pomocí klíče indexu.
- Prohledávání – *scan*. Postupné prohledávání vpřed, či zpět.

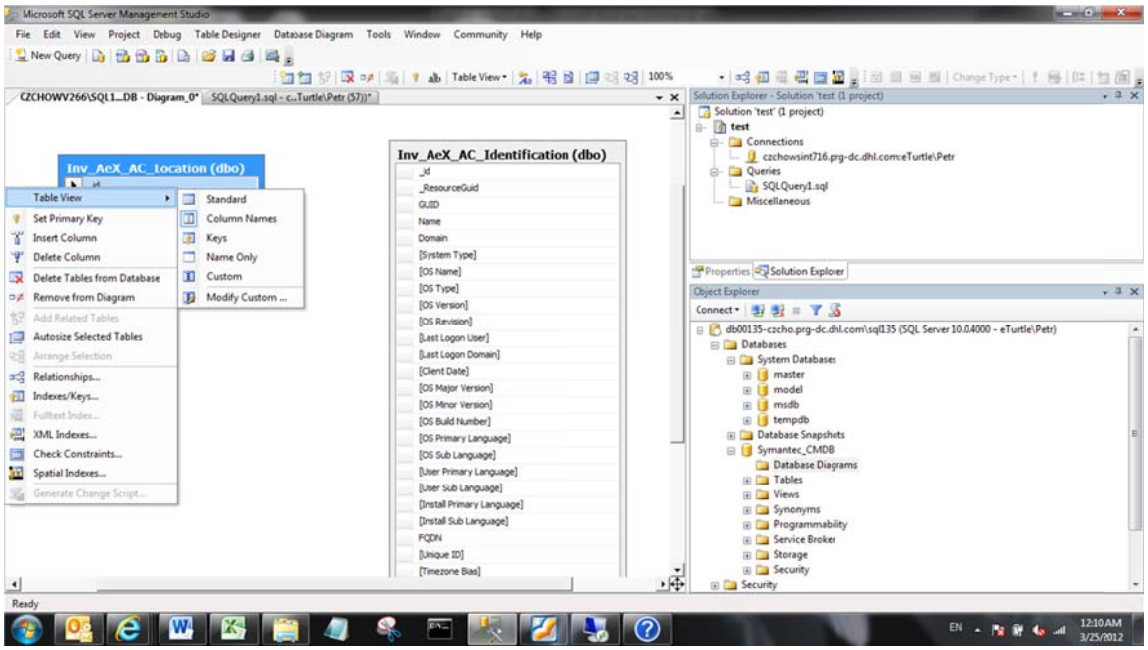
Proces vyhledávání začíná u kořene souměrného stromu a postupuje po jeho uzlech k požadovaným stránkám. Po nalezení hledaných řádků je na nich vykonaná požadovaná operace. (6, 2009, s. 472)

3.10.4 NÁSTROJE PRO OPTIMALIZACI

Základním nástrojem pro optimalizaci je SQL Server Management Studio 2008 (dále SSMS). Jedná se o samostatnou součást integrovaného vývojového prostředí Microsoft Visual Studio. Slouží pro přístup k databázovému systému prostřednictvím SŘBD SQL Server. Umožňuje konfiguraci, správu a údržbu systému. Dále práci s databázemi prostřednictvím SQL příkazů, vývoj a tvorbu nových objektů, ladění, optimalizaci a další činnosti související s databázovými systémy. Pokročilými nástroji jsou SQL Server Profiler (dále Profiler) a SQL Server Database Engine Tuning Advisor (dále DETA).

3.10.4.1 SQL Server Management Studio (SSMS) 2008

K instanci SŘBD lze připojit každým s podporovaných protokolů s výjimkou VIA. Po připojení k instanci SŘBD, lze dle úrovně oprávnění uživatele vykonávat velmi rozsáhlou řadu operací. S nástroji lze pracovat v grafickém i příkazovém režimu.



OBRÁZEK 13: NÁSTROJE PRO OPTIMALIZACI, SQL SERVER MANAGEMENT STUDIO

Rozvržení grafického rozhraní editoru SQL dotazů – Query editor

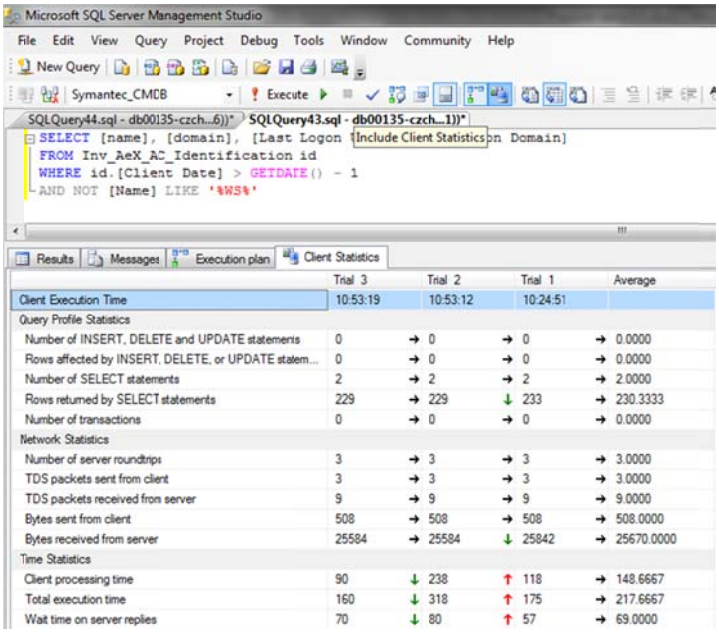
Nejběžnější využití SSMS je práce v editoru dotazů, který má tyto součásti:

- Hlavní okno editoru pro práci s SQL kódem.
- Okno výsledků, které lze zobrazit v tabulkovém či textovém rozvržení.
- Okno zpráv, které SŘBD vysílá během spuštění a s výsledkem dotazu.

Editor dotazů obsahuje 3 pomocné nástroje pro optimalizaci:

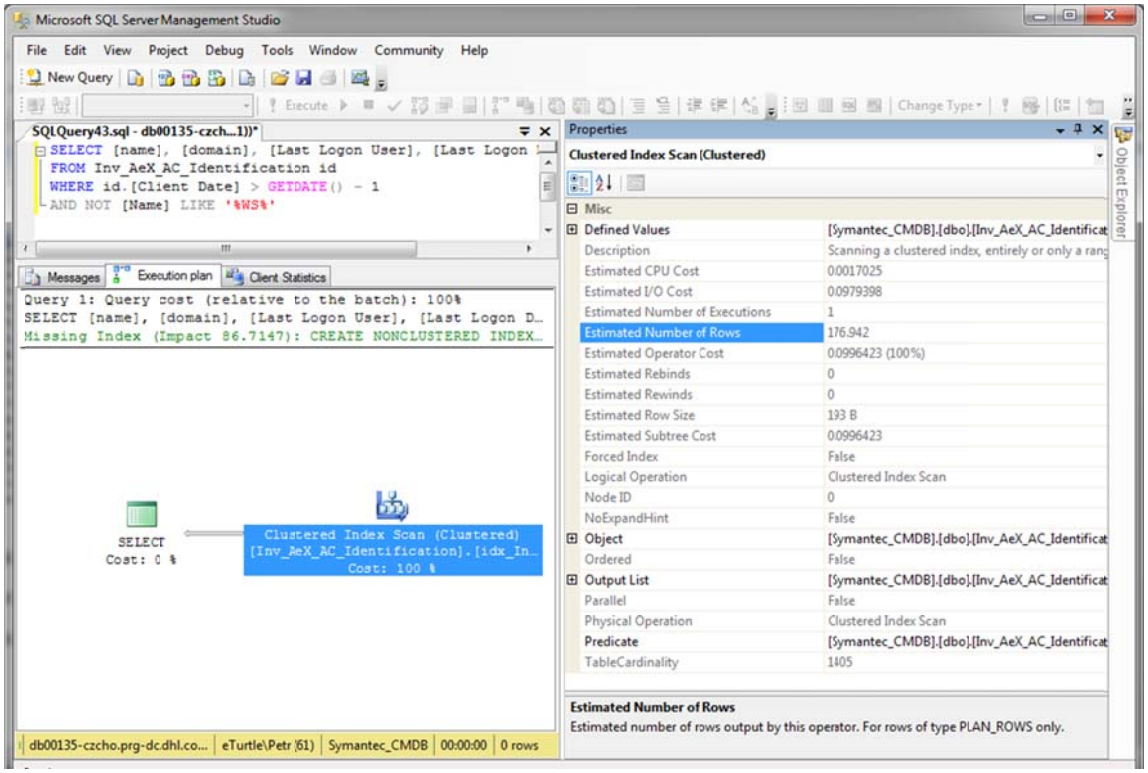
- Souhrn statistik o spuštění dotazu – *Client statistics*
- Předběžný plán spuštění – *Execution plan*
- Skutečný plán spuštění – *Actual execution plan*

Souhrnné statistiky SŘBD vypíše s každým spuštěním dotazu, lze tak sledovat vliv provedených úprav do SQL kódu. Pro každý běh dotazu, je přidán sloupec s číslem běhu. Pro všechny běhy je vypočítán průměr hodnot.



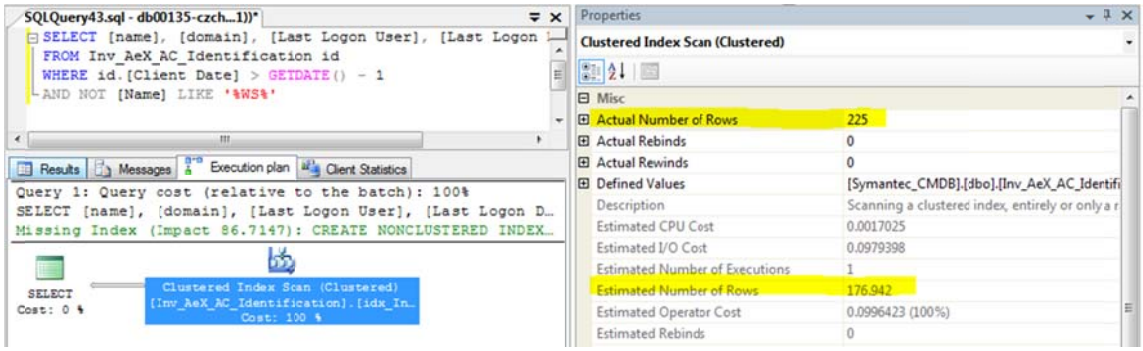
OBRÁZEK 14: NÁSTROJE PRO OPTIMALIZACI, SQL SERVER MANAGEMENT STUDIO, EDITOR SQL DOTAZŮ

Plán spuštění lze vygenerovat před samotným spuštěním příkazu. To má především význam u složitých dotazů, resp. SQL kódu, který vykonává mnoho transakcí a modifikací do struktury databáze. Lze tím předejít vypršení časových limitů (*timeout*) a dalším problémům.



OBRÁZEK 15: NÁSTROJE PRO OPTIMALIZACI, SQL SERVER MANAGEMENT STUDIO, PLÁN SPUŠTĚNÍ

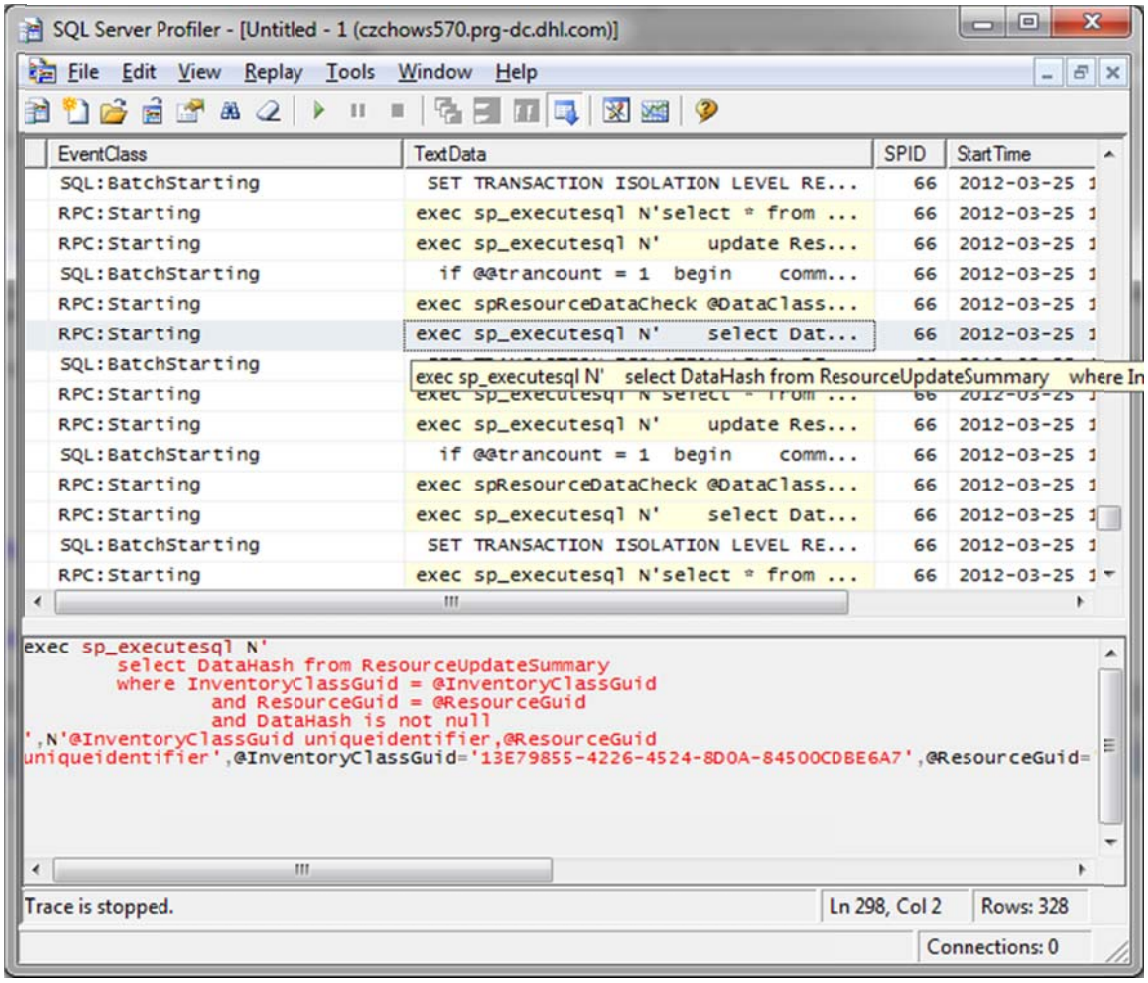
Je-li povoleno vytvoření skutečného plánu spuštění, lze po zpracování dotazu SŘBD porovnat odhad a skutečná data.



OBRÁZEK 16: NÁSTROJE PRO OPTIMALIZACI, SQL SERVER MANAGEMENT STUDIO, STATISTIKY PO SPUŠTĚNÍ

SQL Server Profiler

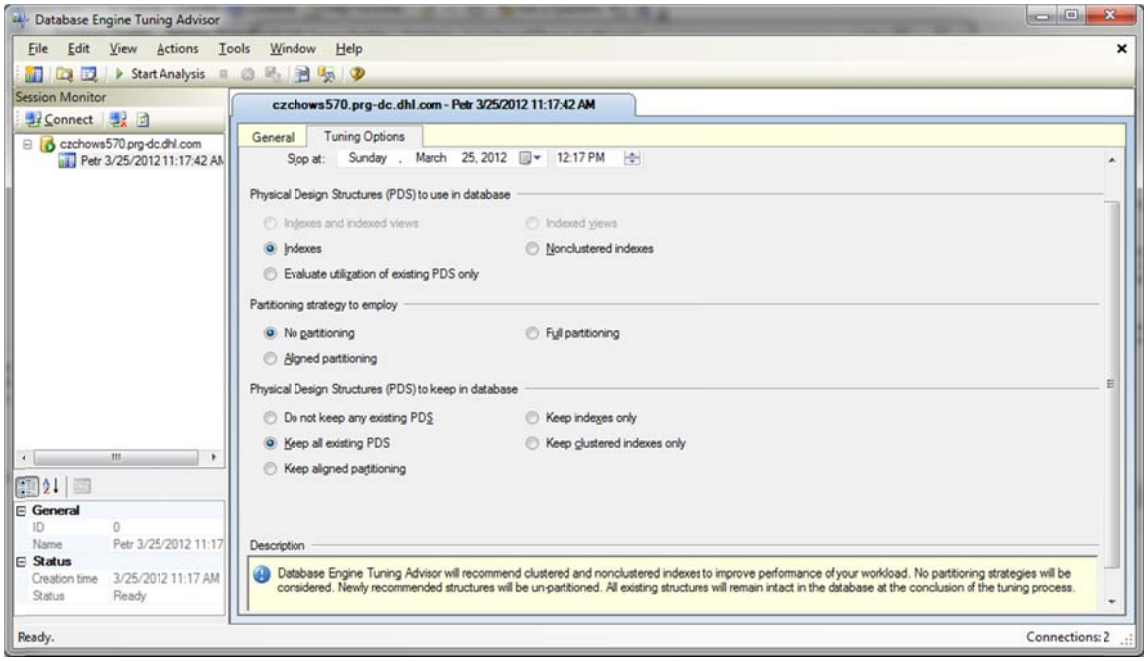
Profiler je nástroj s grafickým uživatelským rozhraním, určený pro sledování činnosti SŘBD a zachytávání událostí. Události jsou řazeny do tříd, zachytávání událostí z vybraných tříd se říká trasování – *tracing*.



OBRÁZEK 17: NÁSTROJE PRO OPTIMALIZACI, SQL SERVER MANAGEMENT STUDIO, PROFILER

SQL Server Database Engine Tuning Advisor

DETA je pomocný nástroj pro optimalizaci výběrů dat, zejména pro analýzu indexů. Na základě takzvaného pracovního souboru (*workload*) doporučí vytvoření nových, nebo úpravu stávajících indexů na tabulkách, kterých se pracovní soubor dotýká. Pracovním souborem se rozumí T-SQL kód, nebo trasa z SQL Profileru.



OBRÁZEK 18: NÁSTROJE PRO OPTIMALIZACI, SQL SERVER MANAGEMENT STUDIO, DATABASE TUNING ADVISOR

3.11 POPIS POJMU CMDB

CMDB je zkratka pro termín "Configuration Management Database". Je to název pro bázi dat, kterou využívají konkrétní softwarové nástroje určené ke správě výpočetního prostředí středních a velkých podniků.

Název CMDB je odvozen z názvu podkapitoly sady světově uznávaných standardů ITIL, která se nazývá Configuration Management.

3.11.1 KONFIGURAČNÍ MANAGEMENT – CONFIGURATION MANAGEMENT (CM)

Cílem správy konfigurací (Configuration Management) je vytvořit logický model výpočetního prostředí a služeb tím, že jednoznačně identifikuje, provozuje a udržuje verze konfiguračních položek. K naplnění tohoto cíle je zapotřebí:

- mít aktuální přehled o *všech* zařízeních tvořících výpočetní prostředí podniku;
- poskytovat přesné informace o konfiguracích a související dokumentaci všem oddělením podniku, spolupracujícím na službách IT;
- zajistit pevnou základnu pro veškeré procesy ve službách IT;

průběžně verifikovat konfigurační záznamy, jejich správnost, a zda souhlasí se skutečností.

3.11.2 Předmět správy konfigurací

Mezi činnost správy konfigurací se řadí identifikování, pořizování záznamů a poskytování informací a datových sestav (reportů) o součástech výpočetního prostředí podniku. Těmito součástmi, které se obecně nazývají konfigurační položky, jsou především hardware (počítače, síťové prvky, periferní zařízení, aj.), software (operační systémy, aplikační software, aj.) a veškerá související dokumentace. Správa spočívá také v udržování přehledu o verzích, relacích a dílčích komponentech konfiguračních položek.

3.11.3 Činnost úseku správy konfigurací

- Plánování a definování důvodu existence, předmětu, úlohy, pravidel a postupů, organizačního a technického kontextu správy konfigurací jako oboru činnosti podniku zabývajícího se službami IT.
- Určování strukturních pozic všech konfiguračních položek, jejich odpovědného správce, vztahu s ostatními konfiguračními položkami a okolním prostředím a veškeré související dokumentace. Tato činnost zahrnuje vlastní inventarizaci majetku, jeho systematické označování a zakládání záznamů v CMDB.
- Zajišťování integrity dat ve smyslu toho, že žádná změna (přidání nové, nebo úprava stávající konfigurační položky, relace, či dokumentace) nemůže být provedena mimo standardní proces a nástroje.
- Sledování stavu každé konfigurační položky během jejího životního cyklu. Toho je dosahováno uchováváním historických údajů každé položky a všech předchozích stavů (například „ve fázi vývoje“, „ve fázi testování“, „v produkci“, „staženo z produkce“).
- Ověřování stavu reálného objektu, který každá konfigurační položka představuje: to by měla být pravidelná činnost ve formě interních auditů.

3.11.4 Popis databáze konfigurací – CMDB

Dá se říci, že každý podnik s významnějším výpočetním prostředím používá alespoň některý z prvků správy konfigurací. Některé z nich však stále ještě používají tabulkové procesory (například Microsoft Excel), nebo dokonce papírové kartotéky pro evidenci a správu konfiguračních položek. V dnešních středních a velkých podnicích, nezávisle na druhu podnikání, existují rozsáhlá výpočetní prostředí, často komplexní a heterogenní. V takových

prostředích je již naprostou nutností existence podpůrných nástrojů a zejména elektronické databáze CMDB, obsluhované robustním systémem řízení báze dat, který dovede obsloužit velké množství datových transakcí. (1, 2000, s. 60)

Musí mimo jiné zajistit:

- obsah vydaných distribucí balíků aplikací a obecně software, včetně verzí a závislých součástí;
- přehled všech konfiguračních položek ovlivněných konkrétní systémovou změnou;
- výpis pořízeného majetku (výpočetní zařízení) s detaily o pořízení;
- ukládání předchozích stavů konfiguračních položek (historii);
- zařazení konfiguračních položek do organizační struktury podniku, dle lokalit, oddělení, apod.;
- zobrazení minulých, probíhajících, či budoucích změn konfiguračních položek a jejich dopad, rozsah, výsledek a související dokumenty (například formuláře, schválení manažerů);
- seznam všech konfiguračních položek ovlivněných konkrétním problémem.

Databáze konfigurací by měla obsahovat relace mezi všemi typy záznamů o skutečnostech, událostech, změnách, známých chybách, problémech a vydaných verzí software. Musí obsahovat informace o uživatelích v podniku, odděleních, pobočkách, obchodních jednotkách a dodavatelích. (1, 2000, s. 63)

3.11.5 FUNKCE DATABÁZE KONFIGURACÍ

Je nutné, aby existovaly automatizované procesy, které s CMDB pracují. Automatizací lze dosáhnout vyšší efektivity a nižší chybovosti při práci s daty.

Je důležité integrovat s CMDB nástroje pro inventarizaci majetku (výpočetní zařízení) a prohledávání sítě. Těmito nástroji je vhodné provést nejen počáteční naplnění daty, ale také provádět pravidelné aktualizace informací a vyhodnocovat zjištěné změny.

Je vhodné spravovat bázi dat o uživatelích jako součást CMDB. Zejména kvůli tvorbě relací s ostatními konfiguračními položkami, což je nezbytné například pro správu a audit licencí. Je

tak možné udržovat aktuální přehled o počtu licencí, jejich využití uživateli, souvislost s počítači, na kterých je aplikační software provozován a další relace. (1, 2000, s. 64)

3.11.6 Konfigurační položky – Configuration items

Konfigurační položky (configuration items, CI) jsou prvky CMDB. Mohou jimi být počítače, jejich komponenty, operační systémy a jejich součásti, software, specifická data, dokumenty, licence. Mohou to být i uživatelské účty a další uživatelské informace. Konfigurační položkou tedy může být každá samostatně vystupující součást výpočetního prostředí, která má vztahy s tímto prostředím.

Konfigurační položky tvoří hierarchii, respektive strukturu tak, jak existují v reálném prostředí. Jsou tedy prvky systému konfiguračních položek.

Druhy relací:

- konfigurační položka je součástí jiné konfigurační položky (například program konkrétního jména a verze je součástí aplikačního software);
- konfigurační položka je spojena s další konfigurační položkou (například konkrétní server má konkrétní síťovou kartu, kterou je připojen ke konkrétní síti s určitými parametry TCP/IP;
- konfigurační položka používá ostatní konfigurační položky (například uživatelka, jež je zaměstnankyní účetního oddělení užívá účetní software a tak čerpá jednu uživatelskou licenci k tomuto software, která je vydána pro její oddělení).

Je zřejmé, že bude existovat množství dalších druhů relací. Všechny tyto relace musí však pro správnou funkci celého systému být zaznamenány v CMDB. (1, 2000, s. 65)

3.11.7 Souvislost mezi správou konfigurací a službami IT

Správa konfigurací (Configuration Management) je nedílnou součástí všech procesů řízení služeb IT v podniku. Poskytuje aktuální, přesné a podrobné informace o veškerých součástech výpočetního prostředí, díky kterým je zejména provádění změn snazší a efektivnější. Řízení změn má správně být spojeno se správou konfigurací. Záznamy o změnách, o průběhu změn a o výsledku změn jsou pořizovány prostřednictvím správy konfigurací. Ta zachycuje souvislosti mezi změnami jedné položky a souvisejícími položkami. Výpočetní prostředí

každého podniku se neustále mění a změny musí být sledovány od začátku do konce v souvislostech. Databáze konfigurací je to jediné místo, kde jsou uložena data o konfiguračních položkách, jejich změnách, aktuálním stavu, souvislostech a vztazích s jinými konfiguračními položkami a další údaje. (1, 2000, s. 70)

Správa konfigurací není prováděna pouze ručně, ale se značnou pomocí softwarových nástrojů, které dovolují automatizaci většiny procesů. Tyto nástroje lze obecně rozdělit na uživatelské a systémové. Uživatelské nástroje jsou klientské aplikace, kterými se uživatelé připojují k CMDB za účelem práce s konfiguračními daty. Systémové nástroje jsou serverové aplikace, které jsou naprogramovány tak, aby bez uživatelského zásahu pracovaly s konfiguračními daty.

Důležitou funkcí systémových nástrojů konfiguračního managementu je identifikace vztahu (nebo více vztahů) mezi konkrétní položkou, která je předmětem změny a jakoukoli jinou položkou. Vlastníci těchto souvisejících položek mohou být informování o formě a míře dopadu změny na položky v jejich správě.

Databáze konfigurací musí být dostupná všem oddělením podniku činným v procesu řízení a podpory služeb IT. Je to nutné kvůli usnadnění a efektivnosti při podpůrných procesech IT, jakými jsou zejména Incident a Problem management. Musí rovněž být tím místem, kde jsou ukládána a udržována data o verzích software a jeho dostupných distribučních balíčcích.

Významnou se CMDB stává i pro oblast správy IT zajišťujících doručení vlastních služeb IT zákazníkům. Této oblasti se dle ITIL říká Service Delivery a má například na starosti tyto situace ve vztahu se CMDB:

- Při návrhu smlouvy o úrovni poskytovaných služeb je nutné zjistit souvislosti mezi předmětnými konfiguračními položkami a ostatními součástmi výpočetního prostředí (například, informace o aplikacích užívaných na pronajímaném osobním počítači a míře jejich využití).
- Při tvorbě různých vyúčtování se přihlíží k využití různých zařízení reprezentovaných konfiguračními položkami (například užité místo na pevném disku serveru).

Při analýze možných rizik souvisejících s určitou službou IT, CMDB poskytuje cenné informace o poruchovosti jednotlivých komponent (například vyčíslení pravděpodobnosti nedostupnosti webové stránky bude brát v úvahu historické informace o opravách vykonaných na webovém serveru za minulé období). (1, 2000, s. 75)

3.11.8 DATABÁZE KONFIGURACÍ SYMANTEC_CMDB

Databáze Symantec_CMDB je implementací databáze konfigurací. Tvoří nedílnou součást produktu Symantec Management Platform 7.1.

Datový model databáze Symantec_CMDB je příliš rozsáhlý, aby jej bylo možné v této práci plně zdokumentovat. Proto bude popsána jen základní struktura a nejdůležitější objekty, které souvisí s úsilím optimalizace.

Základní strukturu tvoří dvě skupiny objektů: správa konfiguračních položek – *item management* a správa zdrojů – *resource management.*

Správa konfiguračních položek

Evidence tříd všech konfiguračních položek, včetně těch, které tvoří součást aplikačního serveru.

TABULKA 15: DATABÁZE KONFIGURACÍ - SPRÁVA KONFIGURAČNÍCH POLOŽEK V SYMANTEC_CMDB

Logický objekt	Fyzický objekt	Popis
Item Class	vItemClass	Pohled na tabulku ItemClass, která obsahuje všechny datové třídy.
Item	vItem	Instance třídy ItemClass, primárním klíčem je atribut [GUID].
Item Reference	vItemReference	Pohled na tabulku ItemReference, obsahující relace mezi všemi položkami.
Folder	vFolder	Pohled na tabulku vNonResourceItem, zobrazuje pouze složky webového rozhraní aplikace SMP.
Report	vReport	Pohled na tabulku vNonResourceItem, zobrazuje pouze položky webových sestav.

Správa zdrojů

Zdroje představují objekty reálného světa, například počítač, uživatel, nebo pevný disk. Zdroje jsou rozřazeny dle typů a hierarchicky uspořádány. Pro typy zdrojů jsou definovány datové třídy. Každý zdroj může mít žádné, nebo více řádků v příslušné tabulce představující datovou třídu.

TABULKA 16: DATABÁZE KONFIGURACÍ - SPRÁVA ZDROJŮ V SYMANTEC_CMDB

Logický objekt	Fyzický objekt	Popis
Resource	vResource, VResourceItem	Pohled na tabulku ItemResource, spojenou s pohledem vItem. Atribut [GUID] je primární klíč.
Resource Type	vResourceType	Pohled na tabulku ResourceType. Atribut [GUID] je primární klíč.
Inventory Class	vInventoryClass	Pohled na tabulku DataClass.
Event Class	vEventClass	Pohled na tabulku DataClass, kde typ je rovný "event".
Inventory Data Table	Pro každou třídu vlastní tabulka.	Tabulky začínající na Inv_. Primární klíč je _id, cizí klíč _ResourceGuid.
Event Data Table	Pro každý druh události vlastní tabulka.	Tabulky začínající na Evt_. Primární klíč je _id, cizí klíč _ResourceGuid.

5 VLASTNÍ PRÁCE

5.1 ZMAPOVÁNÍ SOUČASNÉ ÚROVNĚ VYUŽÍVÁNÍ CMDB V PROBLEMATICE VÝBĚRŮ DAT

Databáze konfigurací CMDB, která je předmětem této práce, je součástí robustního produktu Symantec Management Platform (dále systém SMP). Jedná se o sadu aplikací určených pro správu výpočetního prostředí rozsáhlé korporátní sítě s více než 100 000 výpočetními zařízeními. Tyto aplikace jednak zajišťují komunikační rozhraní pro klientská zařízení, dále poskytují služby spojené s údržbou a provozem klientských zařízení, sbírají detailní informace a zprostředkovávají je formou sestav.

Aplikace jsou integrovány do webového serveru, který zajišťuje aplikační a uživatelské rozhraní. Klientská část aplikací se připojuje k webovým službám, které zajišťují výměnu dat ve formátu XML. Uživatelé přistupují k webovým aplikacím, které především slouží pro správu samotného aplikačního serveru a všech jeho součástí, ale i k účelům vytváření, nebo spouštění sestav informací.

Předmětem systému SMP, je správa počítačů. O každém ze spravovaných počítačů je veden záznam, „objekt počítače“, který je automaticky pravidelně aktualizován. Jedním z důležitých prvků, je sdružování objektů počítačů dle jejich vlastností, do takzvaných filtrů. Je tedy například možné vytvořit filtr pro sdružení všech počítačů, jejichž velikost operační paměti je menší než 1024MB. Součástí systému SMP je řada filtrů definovaných výrobcem. Je však možné filtry vytvářet dle potřeby organizace a věcných účelů. Filtry jsou založené na SQL kódu, který definuje výběry z databáze CMDB. Data v CMDB jsou pravidelně aktualizována prostřednictvím inventarizačních nástrojů systému SMP. Samotné filtry se obnovují pravidelně, buď dle časového rozvrhu, nebo nastane-li jedna z očekávaných systémových událostí. S obnovou filtrů je spojené časté čtení a zapisování do databáze CMDB. Je proto nutné při vytváření filtrů dodržovat pravidla optimalizace výběrů dat s relační databáze.

Dalším z prvků, které lze optimalizovat, jsou sestavy. Systém SMP je instalován s množstvím sestav, které jsou zpřístupněny pomocí dynamických webových formulářů. Správci systému, či uživatelé s potřebným oprávněním mohou vytvářet vlastní sestavy. Stejně jako filtry,

i sestavy jsou založené na SQL kódu. Podléhají stejným potenciálním problémům, tj. špatně definované dotazy.

Cílem následujících kapitol, je uplatnit nabyté znalosti z oblasti optimalizace SQL dotazů na analýze a úpravám filtrů a sestav tvořících součást systému SMP.

Shrnutí

Po provedení analýzy databáze Symantec_CMDB bylo zjištěno její využití těmito základními způsoby:

- Úložiště dat o konfiguračních položkách (počítače a jejich součásti, software, aj.)
- Úložiště dat aplikačního charakteru (data systému SMP)
- Datová báze pro sestavy (hlavní zdroj informací pro tvorbu sestav)

Datová báze pro filtry (hlavní zdroj informací pro tvorbu filtrů)

Nejčastěji využívané tabulky a pohledy

K určení nejčastěji využívaných tabulek a pohledů bylo přistoupeno dvěma způsoby:

- Subjektivním, z pohledu uživatele (správce) systému SMP;
- objektivním, sběrem empirických hodnot.

Z uživatelského, subjektivního pohledu jsou nejvyužívanějšími objekty databáze Symantec_CMDB tyto:

- vComputer
- vItem
- Inv_AeX_AC_Identification
- Inv_AeX_AC_Location
- Inv_Policy_Compliance_Status

- Inv_AddRemoveProgram

5.2 IDENTIFIKACE PŘETRVÁVAJÍCÍCH NEDOSTATKŮ, BARIÉR A POTENCIÁLNÍCH MOŽNOSTÍ

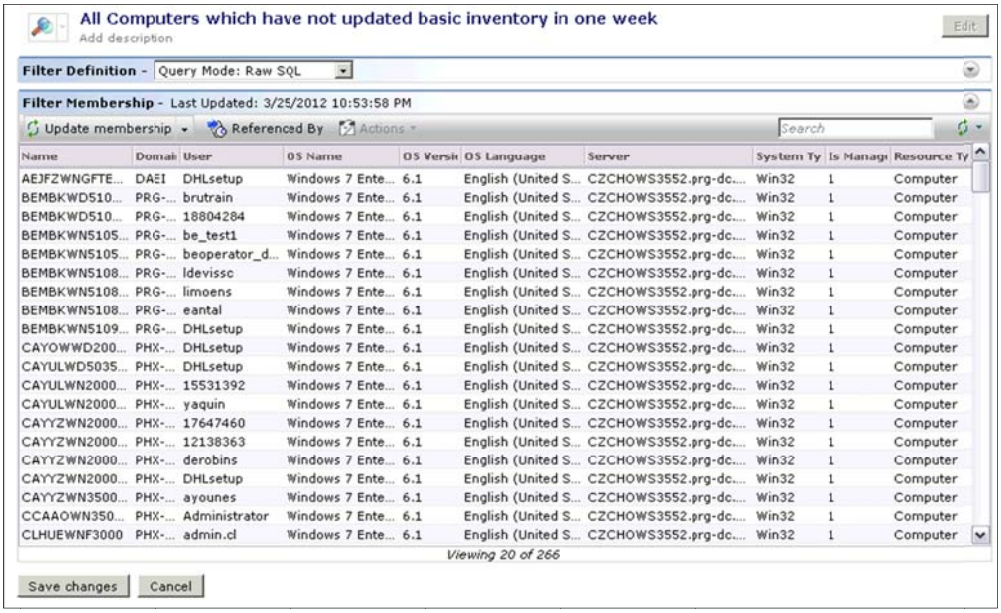
5.2.1 POMALÉ NAČÍTÁNÍ FILTRŮ

Metodika:

1. Analýza dat událostí typu: obnovení filtru
2. Analýza SQL kódu filtru
3. Identifikace potenciálních nedostatků

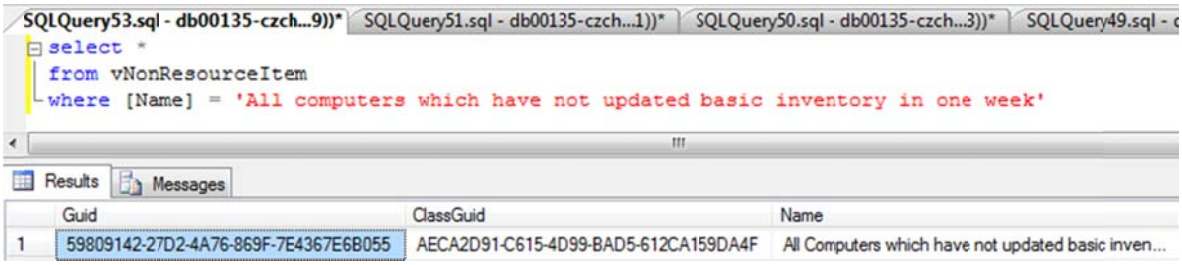
Filtr *All computers which have not updated basic inventory in one week*

Filtr obsahující ty počítače, od kterých nebyla přijata aktualizovaná data konfigurace déle, než 1 týden.



OBRÁZEK 19: IDENTIFIKACE NEDOSTATKŮ, FILTR

Vzhledem k tomu, že filtr je svým typem konfigurační položkou, lze použít pohled vNonResourceItem k získání jeho jedinečného identifikátoru.



OBRÁZEK 20: IDENTIFIKACE NEDOSTATKŮ, ANALÝZA KONSTRUKCE FILTRU, KROK 1

Aktualizace filtru, spuštěním SQL dotazu, který filtr definuje, je systémová událost, která je uložena v tabulce *dbo.Evt_NS_Filter_Update_Duration*. Výběrem řádků splňujících podmínku rovnosti parametru cizího klíče se zjištěným identifikátorem filtru, lze získat celkovou dobu trvání obnovy filtru v milisekundách. Dobu trvání, lze porovnat se zjištěným průměrem pro všechny filtry, který se ustálil na 300ms.

TABULKA 17: IDENTIFIKACE NEDOSTATKŮ, DOBA OBNOVY FILTRU DLE WEBOVÉHO SERVERU

_eventTime	UpdateDuration
3/25/12 22:52:13	4291
3/25/12 22:54:03	4292
3/25/12 23:37:16	4294
3/25/12 23:37:56	4299
3/26/12 0:12:49	4296

5.2.2 POMALÉ NAČÍTÁNÍ SESTAV

Jednou z nejčastěji využívaných sestav je přehled instalovaných aplikací na počítačích spravovaných systémem SMP. Tato sestava zobrazuje informace získané inventarizačním programem, který pravidelně z každého počítače čte všechny položky jinak zobrazené v nabídce Přidat Odebrat Programy (Add Remove Programs) ovládacího panelu operačního systému Windows. Součástí systému SMP je několik výrobcem definovaných sestav pro zobrazení informací o instalovaných aplikacích. Všechny jsou ale založeny na propojení s katalogem software a vyhledávání v příslušných objektech databáze je řešeno pomocí interních identifikátorů. Pokud však uživatel-správce systému SMP potřebuje jiný způsob pohledu na informace o instalovaných aplikacích, musí definovat vlastní sestavu, ve které bude například vyhledávat na základě výrobce, jména, nebo verze aplikace.

Sestava *Computers having Cisco VPN client*

Sestava zobrazuje všechny počítače, na kterých je nainstalována některá z verzí aplikace Cisco VPN client.

Computers having Cisco VPN client
Scope is All computers

Actions ▾ | Save As ▾ | Print | Refresh

Name	Domain	Name	displayversion	Publisher
ATLNZWN2000001	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BDDACWN028	KUL-DC	Cisco Systems VPN Client 5.0.03.0530	5.0.3	Cisco Systems, Inc.
BEANRWN5106640	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BEANRWN5107604	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BEANRWN5109167	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BEANRWN5109279	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BEANRWN5109416	PRG-DC	Cisco Systems VPN Client 5.0.07.0290	5.0.6	Cisco Systems, Inc.
BEANRWN5109416	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BEMBKWN2000001	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BEMBKWN2000009	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BEMBKWN2000009	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
COBOGWNF000461	PHX-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
CZCHONBTESTD430	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
CZCHOWN0012205A	PRG-DC	Cisco VPN Client 5.0.04.0300	5.0.4	Cisco
CZCHOWN0012241	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
CZCHOWN0012241	PRG-DC	Cisco Systems VPN Client 5.0.07.0410	5.0.7	Cisco Systems, Inc.
CZCHOWN0012241	PRG-DC	Cisco Systems VPN Client 5.0.07.0410	5.0.7	Cisco Systems, Inc.
CZCHOWN0012241	PRG-DC	Cisco Systems VPN Client 5.0.07.0410	5.0.7	Cisco Systems, Inc.
CZCHOWN0012241	PRG-DC	Cisco Systems VPN Client 5.0.07.0410	5.0.7	Cisco Systems, Inc.
CZCHOWN0012241	PRG-DC	Cisco Systems VPN Client 5.0.07.0410	5.0.7	Cisco Systems, Inc.

OBRÁZEK 21: IDENTIFIKACE NEDOSTATKŮ, SESTAVA

Pro postup optimalizace sestavy, byla zvolena stejná metodika jako u filtru v předchozí kapitole, tedy:

1. Analýza dat událostí typu: spuštění sestavy
2. Analýza SQL kódu sestavy
3. Identifikace potenciálních nedostatků

Systém SMP monitoruje práci se sestavami a události spojené se spouštěním sestav ukládá to tabulky *Evt_NS_Report_Run*. Jedním ze sloupců tabulky je *ItemName*, který obsahuje jména sestav. Následující tabulka obsahuje informace o posledních 5 spuštěních sestavy.

SQL 14: IDENTIFIKACE NEDOSTATUKŮ, SESTAVA, DOTAZ PRO ZOBRAZENÍ INF. O ZPRACOVÁNÍ SESTAVY

```
SELECT TOP 5 rr.ItemName,rr.NumRowsReturned,rr.TimeTakenSecs
FROM Evt_NS_Report_Run rr
WHERE rr.ItemName = 'Computers having Cisco VPN client'
ORDER BY rr.TimeTakenSecs DESC
```

SQL 15: IDENTIFIKACE NEDOSTATUKŮ, SESTAVA, VÝSLEDEK DOTAZU

ItemName	NumRowsReturned	TimeTakenSecs
Computers having Cisco VPN client	2545	4
Computers having Cisco VPN client	2545	4
Computers having Cisco VPN client	2545	3
Computers having Cisco VPN client	2545	2
Computers having Cisco VPN client	2545	2

Průměrná doba dokončení sestavy jsou 3 vteřiny. Může se zdát, že to je dostatečně krátký čas, ale v případě práce více uživatelů se stejnou sestavou by mohlo docházet k prostojům.

Byl proto učiněn pokus o optimalizaci výběru dat, který SŘBD při spouštění sestavy provádí.

5.4 NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ

5.4.1 ANALÝZA FILTRU

Většinu filtrů, jejichž sestavení trvá déle, než je vzhledem k chodu aplikací přípustné, lze analyzovat a upravit.

Návrh úpravy filtru *All computers which have not updated basic inventory in one week*

Metodika:

1. Analýza SQL kódu, tvořícího filtr
2. Analýza objektů, kterých se SQL kód týká
3. Návrh úpravy

K vykonání kroků 1 – 3 bylo použito SQL Management Studio. Z XML souboru filtru lze vyčíst SQL kód, tvořící dotaz pro filtr:

```
<item> <dataSource typeGuid="b29ba3f7-cc80-4988-a341-d8471b05ea1d"><configuration>

<base><parameters /></base>

<query>

 <![CDATA[select id._ResourceGuid as [GUID]  from Inv_AeX_AC_Identification id  where id.[Client Date] < GETDATE() - 7]]>

</query>

</configuration> </dataSource></item>
```

OBRÁZEK 22: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA FILTRU, VNOŘENÝ SQL KÓD

SQL 16: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA FILTRU, ANALÝZA SQL KÓDU

```
SELECT id._ResourceGuid AS [GUID]

FROM Inv_AeX_AC_Identification id

WHERE id.[Client Date] < GETDATE() - 7
```

SQL dotaz je selekcí z tabulky dbo.Inv_AeX_AC_Identification s predikátem id.[Client Date] < GETDATE() -7. Dalším krokem, je analýza existujících indexů, a zda jsou využity pro zpracování SQL dotazu s daným predikátem.

Prostřednictvím tabulky *sys.indexes* byly zjištěny možné existující indexy na tabulce.

SQL 17: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA FILTRU, ANALÝZA METADAT TABULKY

```
select *
from sys.indexes i
where i.[object_id] = object_id('Inv_AeX_AC_Identification')
order by i.name, i.type_desc
```

TABULKA 18: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA FILTRU, ANALÝZA METADAT TAB., VÝSTUP

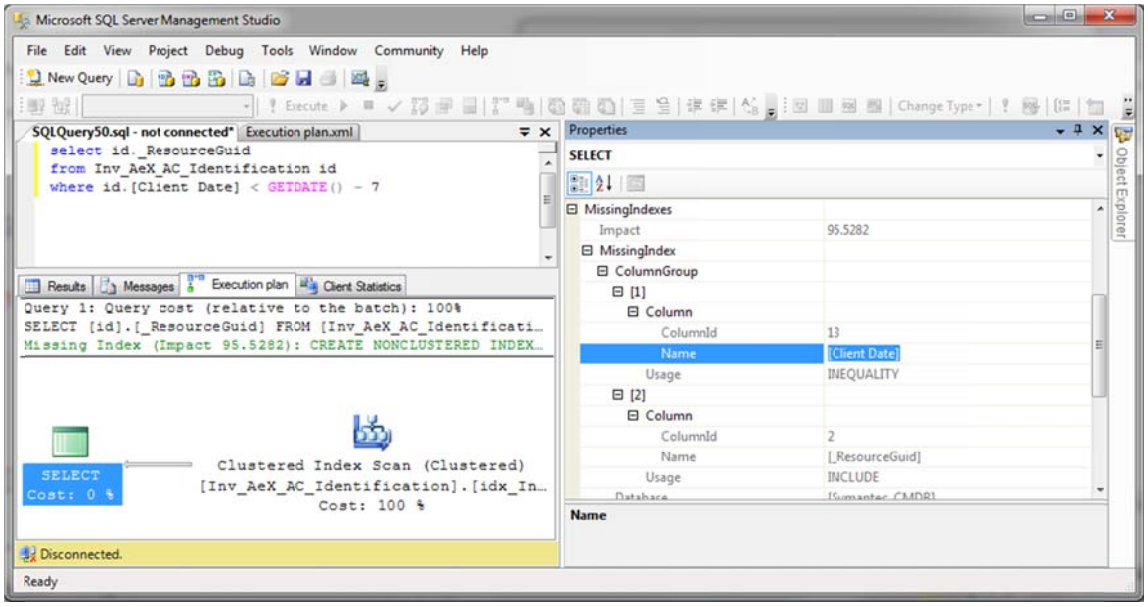
object_id	name	index_id	type	type_desc
1544392571	IDX_DC_c74002b6-c7b9-47bb-a5d6-3031af73bb8d_OSVersion	2	2	NONCLUSTERED
1544392571	IDX_DC_c74002b6-c7b9-47bb-a5d6-3031af73bb8d_SystemType	3	2	NONCLUSTERED
1544392571	Idx_Inv_AeX_AC_Identification_Name_Domain	4	2	NONCLUSTERED
1544392571	idx_Inv_AeX_AC_IdentificationResourceGuid	1	1	CLUSTERED

Zobrazením detailů indexů pomocí systémové funkce *sys.helpindex* (níže) vyplývá, že pro atribut [*Client Date*] neexistuje index (existuje pouze pro sloupce, resp. klíče ve třetím sloupci tabulky níže).

TABULKA 19: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA METADAT TABULKY, VÝSTUP 2

index_name	index_description	index_keys
IDX_DC_c74002b6-c7b9-47bb-a5d6-3031af73bb8d_OSVersion	nonclustered located on PRIMARY	OS Major Version, OS Minor Version
IDX_DC_c74002b6-c7b9-47bb-a5d6-3031af73bb8d_SystemType	nonclustered located on PRIMARY	System Type
Idx_Inv_AeX_AC_Identification_Name_Domain	nonclustered located on PRIMARY	Name, Domain
idx_Inv_AeX_AC_IdentificationResourceGuid	clustered located on PRIMARY	_ResourceGuid

Pro doplnění analýzy, je uveden plán spuštění dotazu, včetně informace o návrhu doplnění potřebného indexu.



OBRÁZEK 23: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA METADAT TABULKY, VÝSTUP 3

```
<MissingIndexes>

             <MissingIndexGroup Impact="95.5282">

               <MissingIndex Database="[Symantec_CMDB]" Schema="[dbo]"
Table="[Inv_AeX_AC_Identification]">

                 <ColumnGroup Usage="INEQUALITY">

                   <Column Name="[Client Date]" ColumnId="13" />

                 </ColumnGroup>

                 <ColumnGroup Usage="INCLUDE">

                   <Column Name="[_ResourceGuid]" ColumnId="2" />

                 </ColumnGroup>

               </MissingIndex>

             </MissingIndexGroup>

           </MissingIndexes>
```

OBRÁZEK 24: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA FILTRU, ANALÝZA METADAT TABULKY, AUTOMATICKÝ NÁVRH PRO VYTVOŘENÍ INDEXU

5.4.3 ANALÝZA SESTAVY

Pro pokus o optimalizaci výběru dat tvořících sestavu *Computers having Cisco VPN client* byla použita následující metodika:

1. Analýza SQL kódu, tvořícího sestavu
2. Analýza objektů zahrnutých v SQL kódu
3. Návrh úpravy

SQL kód, tvořící sestavu je přístupný přímo z webového formuláře samotné sestavy, nebo z XML kódu objektu sestavy.

```
<query>

   - <![CDATA[

   SELECT
    id.[name]
   ,id.[domain]
   ,ap.[displayname]
   ,ap.[displayversion]
   ,ap.[publisher]

   FROM Inv_AeX_AC_Identification AS id
   JOIN Inv_AddRemoveProgram AS ap
   on id._ResourceGuid = ap._ResourceGuid

   WHERE ap.Publisher LIKE '%Cisco%'
   AND ap.DisplayName LIKE '%VPN%'

   ORDER BY ap.DisplayName, ap.DisplayVersion ASC
]]>

</query>
```

OBRÁZEK 25: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA SESTAVY, VNOŘENÝ SQL KÓD

SQL kód je selekcí ze dvou tabulek, s celkem 3 predikáty a řazením výsledných řádků vzestupně dle dvou sloupců.

Relace mezi tabulkami *Inv_AeX_AC_Identification* a *Inv_AddRemoveProgram* je vytvořena pomocí cizího klíče *_ResourceKey*, což je jedinečný identifikátor konfigurační položky (logické reprezentace počítače).

Jeden predikát je součástí příkazu spojení JOIN (*id._ResourceGuid = ap._ResourceGuid*) a dva predikáty tvoří omezující podmínku pro příkaz SELECT (*WHERE ap.Publisher LIKE '%Cisco%'* a *AND ap.DisplayName LIKE '%VPN%'*).

Pomocí systémové uložené procedury *sp_helpindex* bylo zjištěno, že pro tabulku *Inv_AddRemoveProgram* existují definované indexy pouze pro sloupce identifikátorů. Pro tabulku *Inv_AeX_AC_Identification* existují indexy pro více sloupců, z nichž je pro analyzovanou sestavu relevantní pouze jeden, a to na sloupci cizího klíče, *_ResourceGuid*.

TABULKA 20: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA SESTAVY, ANALÝZA INDEXŮ, OBJEKT 1

index_name	index_description	index_keys
IDX_DC_c74002b6-c7b9-47bb-a5d6-3031af73bb8d_OSVersion	nonclustered located on PRIMARY	OS Major Version, OS Minor Version
IDX_DC_c74002b6-c7b9-47bb-a5d6-3031af73bb8d_SystemType	nonclustered located on PRIMARY	System Type
Idx_Inv_AeX_AC_Identification_Client_Date	nonclustered located on PRIMARY	Client Date
Idx_Inv_AeX_AC_Identification_Name_Domain	nonclustered located on PRIMARY	Name, Domain
idx_Inv_AeX_AC_IdentificationResourceGuid	clustered located on PRIMARY	_ResourceGuid

TABULKA 21: NAVRŽENÍ ODSTRANĚNÍ NEDOSTATKŮ, ANALÝZA SESTAVY, ANALÝZA INDEXU, OBJEKT 2

index_name	index_description	index_keys
idx_Inv_AddRemoveProgramResourceGuid	nonclustered located on PRIMARY	_ResourceGuid
idx_Inv_AddRemoveProgramSWComponentGuid	nonclustered located on PRIMARY	_SoftwareComponentGuid

Vzhledem k daným predikátům, je vhodné zavést dodatečný jednoduchý index, ve kterém budou jako klíč indexu zahrnuty sloupce DisplayName, _ResourceGuid a InstallFlag.

Pomocí možnosti indexu INCLUDE budou zahrnuty sloupce DisplayVersion a Publisher.

5.6 OVĚŘENÍ NAVRŽENÝCH ZMĚN FORMOU IMPLEMENTACE

5.6.1 ŘEŠENÍ OPTIMALIZACE PRO FILTR

Postup úpravy filtru *All computers which have not updated basic inventory in one week*

Metodika:

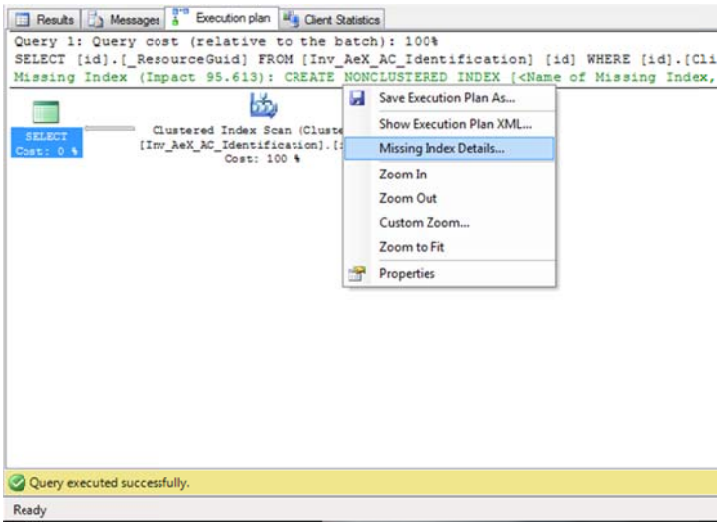
1. sběr statistik z neupraveného filtru
2. úprava filtru pomocí definice z plánu spuštění
3. sběr statistik po úpravě filtru

Pro sběr statistik byl dotaz spuštěn v SSMS se zapnutými doplňky *Client statistics* a *Actual Execution Plan*. Hodnoty statistik jsou v tabulce níže.

TABULKA 22: ŘEŠENÍ OPTIMALIZACE PRO FILTR, HODNOTY MĚŘENÍ PŘED ÚPRAVOU

Statistiky	Popis	Před úpravou
Query Profile Statistics	Statistiky operátorů	
Number of INSERT, DELETE and UPDATE statements	Počet příkazů daného typu	0
Rows affected by INSERT, DELETE, or UPDATE statements	Počet příkazů daného typu	0
Number of SELECT statements	Počet příkazů daného typu	2
Rows returned by SELECT statements	Počet řádků vrácených výběrem	279
Number of transactions	Počet transakcí	0
Time Statistics	Statistiky doby zpracování	
Client processing time	Celkový čas zpracování klientem	16
Total execution time	Celkový čas dotazu	4281
Wait time on server replies	Celkový čas čekání na server	4265

Plán spuštění automaticky vygeneroval upozornění o chybějícím indexu na sloupci [*Client date*]. Zároveň vygeneroval SQL kód, třídy DDL, kterým lze po menší úpravě chybějící index vytvořit.



OBRÁZEK 26: ŘEŠENÍ OPTIMALIZACE PRO FILTR, KONKRÉTNÍ PLÁN SPUŠTĚNÍ

```
/* Missing Index Details from SQLQuery50.sql - not connected

The Query Processor estimates that implementing the following index could improve the query
cost by 95.5282%. */

/* USE [Symantec_CMDB]

   GO

   CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]

   ON [dbo].[Inv_AeX_AC_Identification] ([Client Date])

   INCLUDE ([_ResourceGuid])

  GO*/
```

OBRÁZEK 27: ŘEŠENÍ OPTIMALIZACE PRO FILTR, AUTOMATICKÝ NÁVRH INDEXU

V tabulce výše, je zobrazen návrh DDL kódu pro vytvoření chybějícího indexu. SQL Server doplnil odhad zlepšení výkonnosti při zpracování SQL dotazu o více než 95%.

Návrh kódu byl upraven na finální kód níže, připravený pro spuštění v SSMS. Úpravy kódu byly tyto:

- Jméno indexu na *Idx_Inv_AeX_AC_Identification_Client_Date*
- Přidáním možnosti indexu FILLFACTOR (*viz kapitola 3.8.1.7*)
- Přidáním možnosti indexu SORT_IN_TEMPDB (*viz kapitola 3.8.1.7*)

```
USE [Symantec_CMDB]
GO
CREATE NONCLUSTERED INDEX [Idx_Inv_AeX_AC_Identification_Client_Date]
ON [dbo].[Inv_AeX_AC_Identification] ([Client Date])
INCLUDE ([_ResourceGuid])
WITH
 (
        FILLFACTOR = 85,
        SORT_IN_TEMPDB = ON
 );
GO
```

OBRÁZEK 28: ŘEŠENÍ OPTIMALIZACE PRO FILTR, HOTOVÝ DDL KÓD PRO VYTVOŘENÍ INDEXU

Po spuštění DDL příkazu pro vytvoření indexu, byla provedena kontrola pomocí systémové funkce *sp_helpindex*. Výstupní hodnoty funkce jsou v tabulce níže. Index byl přidán úspěšně.

TABULKA 23: ŘEŠENÍ OPTIMALIZACE PRO FILTR, OVĚŘENÍ VYTVOŘENÍ INDEXU

index_name	index_description	index_keys
IDX_DC_c74002b6-c7b9-47bb-a5d6-3031af73bb8d_OSVersion	nonclustered located on PRIMARY	OS Major Version, OS Minor Version
IDX_DC_c74002b6-c7b9-47bb-a5d6-3031af73bb8d_SystemType	nonclustered located on PRIMARY	System Type
Idx_Inv_AeX_AC_Identification_Client_Date	nonclustered located on PRIMARY	Client Date
Idx_Inv_AeX_AC_Identification_Name_Domain	nonclustered located on PRIMARY	Name, Domain
idx_Inv_AeX_AC_IdentificationResourceGuid	clustered located on PRIMARY	_ResourceGuid

Po úspěšném přidání indexu, byl SQL dotaz spuštěn znovu a byly vygenerovány aktuální statistiky o zpracování dotazu. Výsledek je v tabulce níže.

TABULKA 24: ŘEŠENÍ OPTIMALIZACE PRO FILTR, POROVNÁNÍ DOBY SPUŠTĚNÍ

Statistiky	Popis	Před úpravou	Ne/rovnost	Po úpravě
Query Profile Statistics	Statistiky operátorů			
Number of INSERT, DELETE and UPDATE statements	Počet příkazů daného typu	0	=	0
Rows affected by INSERT, DELETE, or UPDATE statements	Počet příkazů daného typu	0	=	0
Number of SELECT statements	Počet příkazů daného typu	2	=	2
Rows returned by SELECT statements	Počet řádků vrácených výběrem	279	=	279
Number of transactions	Počet transakcí	0	=	0
Time Statistics	Statistiky doby zpracování			
Client processing time	Celkový čas zpracování klientem	16	<	24
Total execution time	Celkový čas dotazu	4281	>	42
Wait time on server replies	Celkový čas čekání na server	4265	>	18

Ze statistik vyplývá, že přidáním indexu se čas zpracování snížil 101,9 krát.

Z pohledu na výpis událostí webového serveru, je rovněž patrné značné zlepšení:

TABULKA 25: ŘEŠENÍ OPTIMALIZACE PRO FILTR, INFORMACE O SPUŠTĚNÍ Z LOGU UDÁLOSTÍ WEBSERVERU

_eventTime	UpdateDuration
3/28/12 22:18:46	62
3/28/12 22:18:36	47
3/28/12 22:18:16	62
3/25/12 22:52:13	4291
3/25/12 22:54:03	4292
3/25/12 23:37:16	4294
3/25/12 23:37:56	4299
3/26/12 0:12:49	4296

5.6.3 ŘEŠENÍ OPTIMALIZACE PRO SESTAVU

Postup řešení byl vykonán dle následující metodiky:

1. sběr statistik ze spuštění sestavy před přidáním indexu
2. provedení úpravy přidáním indexu
3. opakovaný sběr statistik a porovnání

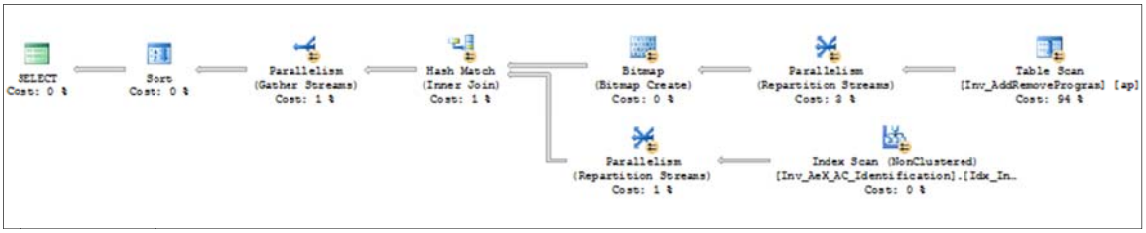
Pro sběr statistik bylo použito SQL Server Management Studio (SSMS), se zapnutými přepínači *Include Client Statistics* a *Include Actual Execution Plan*.

TABULKA 26: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, HODNOTY MĚŘENÍ PŘED ÚPRAVOU

Statistiky	Popis	Před úpravou
Query Profile Statistics	Statistiky operátorů	
Number of INSERT, DELETE and UPDATE statements	Počet příkazů daného typu	0
Rows affected by INSERT, DELETE, or UPDATE statements	Počet příkazů daného typu	0
Number of SELECT statements	Počet příkazů daného typu	2
Rows returned by SELECT statements	Počet řádků vrácených výběrem	2547
Number of transactions	Počet transakcí	0
Time Statistics	Statistiky doby zpracování	
Client processing time	Celkový čas zpracování klientem	3781
Total execution time	Celkový čas dotazu	4025
Wait time on server replies	Celkový čas čekání na server	244

Celkový čas zpracování SQL dotazu SŘBD je 4025 ms (cca 4 vteřiny).

Místo nalezení relevantních řádků prostřednictvím patřičného indexu, musí SŘBD projít každý řádek tabulky a uplatnit podmínky v predikátu.



OBRÁZEK 29: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, PLÁN SPUŠTĚNÍ

Na obrázku výše, je prohledávání celé tabulky Inv_AddRemoveProgram zobrazeno vpravo nahoře, s poměrem 94% z celkové nákladnosti dotazu.

Byl přidán index [IDX_Inv_AddRemoveProgram] jako jednoduchý index, s klíčem složeným ze tří sloupců

- _ResourceGuid, což je cizí klíč, používaný operátory JOIN
- InstallFlag, což je nula-jedničkový atribut, pro vyloučení řádků představujících odinstalovaný software
- DisplayName, atribut představující název software.

Možností INCLUDE byly dále zahrnuty sloupce DisplayVersion a Publisher.

SQL 18: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, KÓD DDL PRO VYTVOŘENÍ INDEXU

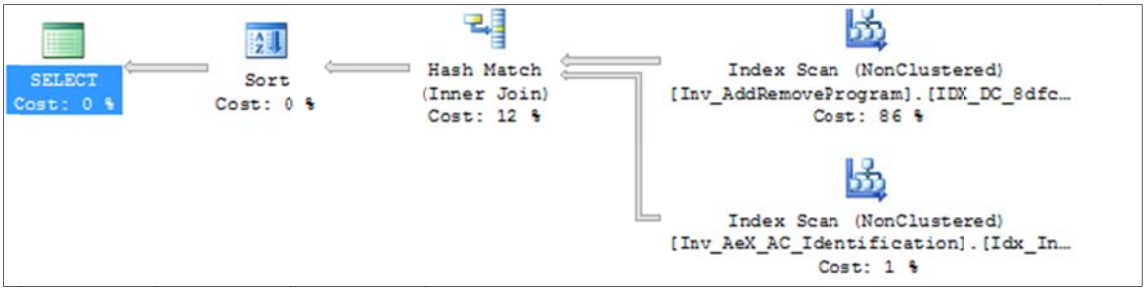
```
CREATE NONCLUSTERED INDEX [IDX_Inv_AddRemoveProgram] ON [dbo].[Inv_AddRemoveProgram]

(

	[_ResourceGuid] ASC,

	[InstallFlag] ASC,

	[DisplayName] ASC

)

INCLUDE ( [DisplayVersion],

[Publisher]) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]

GO
```

Pomocí vložené procedury bylo ověřeno úspěšné vytvoření indexu zobrazením metadat tabulky *Inv_AddRemoveProgram*:

TABULKA 27: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, OVĚŘENÍ VYTVOŘENÍ INDEXU

index_name	index_description	index_keys
IDX_Inv_AddRemoveProgram	nonclustered located on PRIMARY	_ResourceGuid, InstallFlag, DisplayName
idx_Inv_AddRemoveProgramResourceGuid	clustered located on PRIMARY	_ResourceGuid
idx_Inv_AddRemoveProgramSWComponentGuid	nonclustered located on PRIMARY	_SoftwareComponentGuid

Přidáním indexu se plán spuštění změnil následujícím způsobem:



OBRÁZEK 30: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, PLÁN SPUŠTĚNÍ PO PŘIDÁNÍ INDEXU

Na místo prohledávání řádku po řádce je uplatněn nový index, na obrázku vpravo nahoře. Byly dále vypuštěné již nadbytečné kroky paralelismu, tedy uplatnění více procesorů pro vyhledávání v tabulce a sloučení vyhledaných řádků každým vláknem do jedné sady řádků. V tabulce níže je porovnán celkový čas zpracování dotazu před zavedením indexu a po zavedení indexu.

TABULKA 28: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, HODNOTY MĚŘENÍ DOBY ZPRACOVÁNÍ DOTAZU PO ÚPRAVĚ

Statistiky	Popis	Před úpravou	Ne/rovnost	Po úpravě
Query Profile Statistics	Statistiky operátorů			
Number of INSERT, DELETE and UPDATE statements	Počet příkazů daného typu	0	=	0
Rows affected by INSERT, DELETE, or UPDATE statements	Počet příkazů daného typu	0	=	0
Number of SELECT statements	Počet příkazů daného typu	2	=	2
Rows returned by SELECT statements	Počet řádků vrácených výběrem	2547	<	2578
Number of transactions	Počet transakcí	0	=	0
Time Statistics	Statistiky doby zpracování			
Client processing time	Celkový čas zpracování klientem	3781	>	1056
Total execution time	Celkový čas dotazu	4025	>	1686
Wait time on server replies	Celkový čas čekání na server	4265	>	630

Výsledek optimalizace není tak výrazný, jako v předchozím případě, při optimalizaci filtru *All computers which have not updated basic inventory in one week*. Přesto se ale podařilo urychlit zpracování více než dvounásobně.

Výpisem posledních událostí spojených se sestavou lze ověřit dosaženou úroveň optimalizace:

TABULKA 29: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, INF. O SPUŠTĚNÍ Z LOGU UDÁLOSTÍ WEBSERVERU

ItemName	NumRowsReturned	TimeTakenSecs
Computers having Cisco VPN client	2547	1
Computers having Cisco VPN client	2547	1
Computers having Cisco VPN client	2547	1
Computers having Cisco VPN client	2547	1
Computers having Cisco VPN client	2547	0

5.6.3.1 Ověření potenciálního negativního vlivu přidání indexu

Dle [12] může přidání indexu neúměrně zvýšit počet vstupně-výstupních operací: Příkazy DML, jako INSERT, UPDATE a DELETE mění strukturu dat v tabulce. Příčinou je nutnost úpravy indexu, zejména složeného, který obsahuje vlastní řádky tabulky řazené sestupně nebo vzestupně. To může vést k většímu počtu operací než při přímém prohledávání tabulky, tedy bez indexu. Pro úpravu sestavy *Computers having Cisco VPN client* byl tento potenciální problém analyzován následovně.

Metodika:

1. Pomocí příkazů SET STATISTICS IO a SET STATISTICS PROFILE byl zjištěn počet operací při spuštění SQL dotazu sestavy –
    a. Před přidáním indexu IDX_Inv_AddRemoveProgram
    b. Po přidání indxu IDX_Inv_AddRemoveProgram
2. Výsledné hodnoty vybraných statistik byly porovnány (počet vstupně-výstupních operací, náklady operací CPU, náklady na operace vytváření stromu spuštění)

Na obrázku níže, je znázorněno zapnutí statistik a výsledek spuštění dotazu včetně vypočtených statistik. Další obrázek obsahuje výsledek statistik vstupně-výstupních operací.

```
set statistics io on;
set statistics profile on;
go

SELECT...
```

Results | Messages

name	domain	displayname	displayversion	publisher
BEANRWN5106454	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.
BEANRWN5106640	PRG-DC	Cisco AnyConnect VPN Client	2.5.3041	Cisco Systems, Inc.

PhysicalOp	EstimateIO	EstimateCPU	TotalSubtreeCost	EstimateRows	Rows	AvgRowSize
NULL	NULL	NULL	3.123374	176.4721	2577	NULL
Sort	0.01126...	0.002155126	3.123374	176.4721	2577	507
Hash Match	0	0.1226648	3.109957	176.4721	2577	507
Index Scan	2.530532	0.1709584	2.701491	176.4721	2580	391
Index Scan	0.023125	0.0018136	0.0249386	1506	1506	157

OBRÁZEK 31: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, STATISTIKY VSTUPNĚ VÝSTUPNÍCH OPERACÍ

Results | Messages

```
(2577 row(s) affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, re
Table 'Inv_AeX_AC_Identification'. Scan count 1, logical reads 30, phy
Table 'Inv_AddRemoveProgram'. Scan count 1, logical reads 3512, physic

(5 row(s) affected)
```

OBRÁZEK 32: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, STATISTIKY VSTUPNĚ-VÝSTUPNÍCH OPERACÍ 2

TABULKA 30: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, HODNOTY STATISTIK VÝSTUPNÍCH OPERACÍ

	Popis	**Před přidáním indexu**	**Po přidání indexu**	**Po přestavění indexu**
EstimateIO	odhadovaný počet vstupně-výstupních operací	4.72055226	2.56491826	2.72565926
EstimateCPU	odhadované náklady procesorového času	0.292111726	0.297591926	0.297621294
TotalSubtree Cost	náklady na vytvoření stromu spuštění	35.11425038	12.0831346	12.7262906

Z výsledků je patrné, že zatímco náklady na procesorový čas zůstaly téměř stejné, výrazně se změnil odhadovaný počet vstupně-výstupních operací – až o 46% méně. Souvisí s tím závislá

statistika nákladů na vytvoření stromu dotazu pro plán spuštění, jejíž hodnota ukazovala na 56% zlepšení.

Z výstupu nástroje STATISTICS IO, kterým jsou změřeny skutečné vstupně-výstupní operace, vyplývají následující fakta:

- Bez indexu, provedl SŘBD 10 hledání a celkem 6408 čtení na v dotazu zahrnutých tabulkách.
- S indexem, provedl SŘBD 2 hledání a celkem 3722 čtení na v dotazu zahrnutých tabulkách.

Lze tedy konstatovat, že návrh dodatečného indexu byl správný a jeho použití nepředstavuje dodatečnou zátěž pro SŘBD takovou, aby celkový počet vstupně-výstupních operací přesáhl celkový počet těchto operací bez přítomnosti indexu. Vytvoření indexu se tedy z obou úhlů pohledu vyplatilo.

TABULKA 31: ŘEŠENÍ OPTIMALIZACE PRO SESTAVU, HODNOTY STATISTIK VÝSTUPNÍCH OPERACÍ 2

bez indexu		
Table 'Worktable'	Scan count 0	logical reads 0
Table 'Inv_AeX_AC_Identification'	Scan count 5	logical reads 85
Table 'Inv_AddRemoveProgram'	Scan count 5	logical reads 6323
s indexem		
Table 'Worktable'	Scan count 0	logical reads 0
Table 'Inv_AeX_AC_Identification'	Scan count 1	logical reads 30
Table 'Inv_AddRemoveProgram'	Scan count 1	logical reads 3512
po přestavění indexu		
Table 'Worktable'	Scan count 0	logical reads 0
Table 'Inv_AeX_AC_Identification'	Scan count 1	logical reads 30
Table 'Inv_AddRemoveProgram'	Scan count 1	logical reads 3692

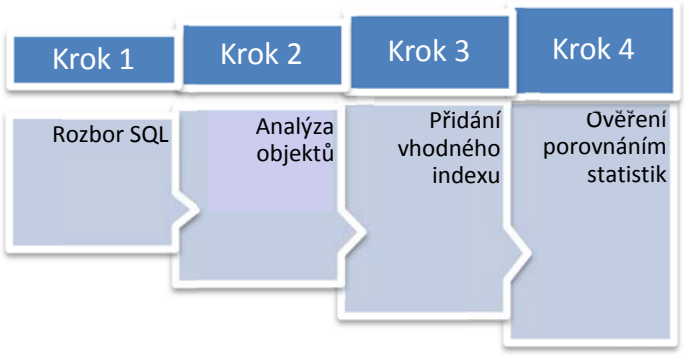
5.8 ZOBECNĚNÍ OVĚŘENÝCH POSTUPŮ OPTIMALIZACE

5.8.1 OBECNÁ PRAVIDLA PRO TVORBU FILTRŮ Z POHLEDU OPTIMALIZACE

Na základě ověřených principů optimalizace výběrů dat z relační databáze, lze vyvodit obecně platná pravidla pro tvorbu nových, „uživatelských“ filtrů:

- Analyzovat všechny objekty, které dotaz zahrnuje. Tzn. tabulky, pohledy, indexy, funkce a případně vložené procedury a spouště. K analýze použít funkce DMV, možnosti SET STATISTICS ON a případně SQL Profiler. Změřit čas zpracování dotazu bez úprav. Zaznamenat počet vstupně-výstupních operací, které SŘBD musí provést.
- Využít automatického návrhu SSMS, pokud existuje. Případně navrhnout úpravu dotazu, nebo některého ze zahrnutých objektů.
- Provést zálohu databáze, nebo její části a provést úpravu.
- Změřit nové hodnoty a porovnat hodnotami naměřenými před provedením úpravy. Nejen však hodnoty související s případným zkrácením času zpracování dotazu, ale také počet dodatečných vstupně-výstupních operací spojených s prohledáváním přidaných indexů.
- Posoudit, zda provedené úpravy jsou přínosem, či nikoliv.

V případě přijetí změn, nastavit údržbu. Například pravidelně plánovanou defragmentaci indexů.



OBRÁZEK 33: ZOBECNĚNÍ PŘÍSTUPU K OPTIMALIZACI, POSTUP

5.8.2 KONKRÉTNÍ DOPORUČENÍ – VYTVÁŘENÍ INDEXŮ

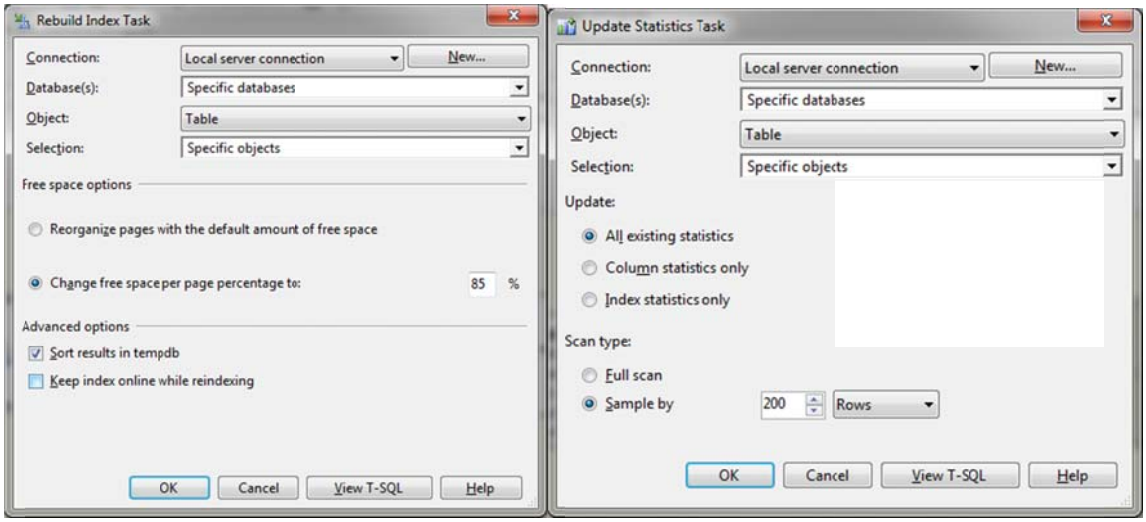
Při analýze významu úprav provedených v rámci optimalizace na objektech zahrnutých do dotazů v předchozích kapitolách, se především osvědčila tato doporučení:

- Užívání pouze tzv. úzkých indexů (Fritchey, 2009, s. 109). Tj. takových, které mají klíč indexu sestavený ze sloupce, či sloupců s datovými typy malé délky (int, uniqueid, datetime, apod.). Z čehož vyplývá, že je vhodné se vyvarovat obzvláště datových typů CHAR, VARCHAR, NCHAR a NVARCHAR.

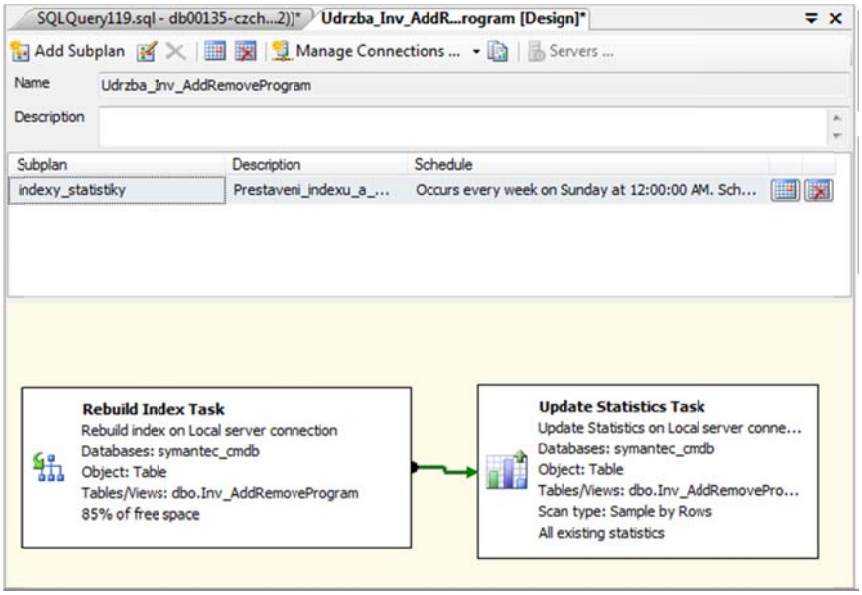
Používání takových sloupců tabulky pro klíč indexu, které mají tzv. vysokou selektivitu, tedy vysoký počet jedinečných záznamů. Z toho plyne, že není doporučeno používat pro klíč indexu například sloupce s *booleovskými* hodnotami (Fritchey, 2009, s. 111).

5.8.3 NAPLÁNOVÁNÍ ÚLOH ÚDRŽBY PRO OBJEKTY UPRAVENÉ V RÁMCI OPTIMALIZACE

Indexy, přidané k objektům databáze Symantec_CMDB v rámci optimalizace, byly nastaveny tak, aby se nepřestavovaly (REBUILD INDEX) při práci s databází. Nastavení bylo provedeno při vytvoření indexů použitím konfigurační možnosti ONLINE=OFF. Je proto nutné, naplánovat údržbu indexů přestavěním a následným přepočítáním statistik. K tomu účelu slouží plány údržby v SSMS.



OBRÁZEK 34: ÚDRŽBA VYTVOŘENÝCH INDEXŮ



OBRÁZEK 35: ÚDRŽBA VNOŘENÝCH INDEXŮ, PLÁN ÚDRŽBY

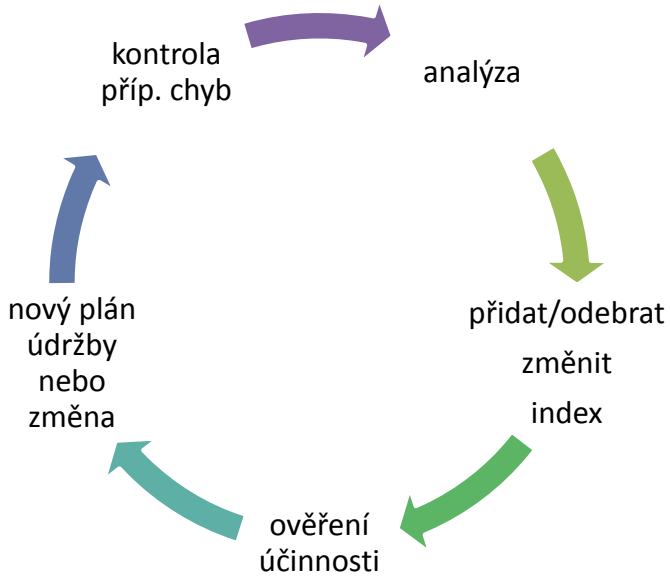
SQL 19: ÚDRŽBA VYTVOŘENÝCH INDEXU, PLÁN ÚDRŽBY, SQL KÓD PRO PŘEPOČTENÍ STATISTIK

```
USE [Symantec_CMDB]

GO

UPDATE STATISTICS [dbo].[Inv_AddRemoveProgram]

WITH SAMPLE 200 ROWS
```

SQL 20: ÚDRŽBA VYTVOŘENÝCH INDEXU, PLÁN ÚDRŽBY, SQL KÓD PRO PŘESTAVĚNÍ INDEXU

```
USE [Symantec_CMDB]

GO

ALTER INDEX [idx_Inv_AddRemoveProgram] ON dbo.Inv_AddRemovePrograms

REBUILD WITH (

FILLFACTOR= 85,

SORT_IN_TEMPDB = ON,

ONLINE = OFF )
```

Následující obrázek symbolizuje fakt, že údržba indexů je opakující se záležitost. Přesto, že lze přestavění a přepočítání statistik svěřit SŘBD dle vytvořeného plánu údržby, je vhodné v pravidelných intervalech provést kontrolu úspěšnosti údržby. Může se například změnit frekvence zápisů do tabulky a tím zvýšit úroveň fragmentace v kratším časovém období. Po odhalení takové skutečnosti, je nutné znovu učinit analýzu a ověřit, zda se ještě index ve stávající konfiguraci vyplatí používat a udržovat.



OBRÁZEK 36: ÚDRŽBA VYTVOŘENÝCH INDEXŮ, PLÁN ÚDRŽBY, SYMBOLICKÉ VYJÁDŘENÍ OPAKUJÍCÍHO SE PROCESU ÚDRŽBY INDEXU

6 ZÁVĚR

Cílem práce bylo prostudovat literární zdroje související s optimalizací dotazů SQL a získané znalosti uplatnit při práci a správě relační databáze konfigurací CMDB.

Tímto závěrem lze konstatovat, že teoretická východiska přístupů k optimalizaci výběrů dat z relačních databází byla úspěšně implementována na konkrétní databázi CMDB.

Z optimalizačních technik doporučených autory citovaných publikací, byla především použita indexní analýza. S její pomocí, bylo dosáhnuto velmi dobrých a měřitelných výsledků při výběrech dat z databáze CMDB. Celkové doby trvání zpracování dotazů byly významně zkráceny. Pochybnosti, resp. upozornění některých z autorů o negativním vlivu přidávání specifických typů indexů byly na příkladech vyvráceny.

Závěry učiněné během optimalizace byly zobecněné a lze je aplikovat při práci se systémem SMP, jehož jádrem je právě databáze CMDB. Aplikací optimalizačních přístupů v této práci ověřených, lze dosáhnout vyšší produktivity práce uživatelů tím, že doba čekání na dokončení sestav bude výrazně kratší. Zároveň dojde k relativnímu snížení čerpání systémových prostředků, a to zejména při vstupně-výstupních operacích.

7 SEZNAM LITERATURY

1. Kolektiv autorů. Service Support. 2000. Stationary Office, London, England. 200s. ISBN 0113300158.

2. Kline, K. SQL in a nutshell, 2nd edition. 2004. O'Reilly Media, Sebastobol, California. 710s. ISBN 0-596-00481-8.

3. DELANEY, Kalen. SQL Server History. Inside SQL Server [online]. 2011 [cit. 26 února 2012]. Dostupné na Internetu: <www.insidesqlserver.com/History/History%20of%20SQL%20Server.pdf>.

4. GARDEN, Euan. BLOG. SQL Server is really a Sybase product not a Microsoft one. [online]. 2006 [cit. 26. Února 2012]. Dostupné na Internetu: <blogs.msdn.com/b/euenga/archive/2006/01/19/sql-mythbusters-sql-server-is-really-a-sybase-product-not-a-microsoft-one.aspx>.

5. KLINE, Kevin. SQLServerPedia. SQL Server 2005 Release Date Calendar. 2012 [cit. 26 února 2012]. Dostupné na Internetu: <sqlserverpedia.com/wiki/SQL_Server_2005_Release_Date_Calendar>.

6. DELANEY, Kalen. SQL Server 2008 Internals. 2009. Microsoft Press, Redmond, Washington. 727s. ISBN 0735626243.

7. Dev Center – Desktop, MSDN,[online]. 2012 [cit. 17. března 2012]. Dostupné na Internetu: < http://msdn.microsoft.com/en-us/library/windows/desktop/ms722784(v=vs.85).aspx>.

8. HOTEK, Mike. Microsoft SQL Server 2008 – Implementation and Maintenance. 2009. Microsoft Press, Redmont, Washington. 623s. ISBN 978-0-7356-2605-8.

9. VOSTROVSKÝ, Václav. Vytváření databází v Oracle. 1. vyd. Praha: ČZU v Praze Provozně ekonomická fakulta. 2004. 134 s. ISBN 80-213-1191-6.

10. BUŠEK, Ivan. Matematika pro gymnázia. 3. vydání. Praha: Prometheus. 2004. 176 s. ISBN 978-80-7196-146-8.

11. BALCAR, Bohuslav; ŠTĚPÁNEK, Petr. Teorie množin. 2. vyd. Praha: Academia. 2001. 464 s. ISBN 80-200-0470-X.

12. FRITCHEY, Grant; DAM, Sajal. SQL Server 2008 Query Performance Tuning Distilled. 2009. Apress, Berkeley, California. 531 s. ISBN-13 978-1-4302-1902-6.

8 PŘÍLOHY

8.1 SEZNAM ZKRATEK

CLR ..... Common Language Runtime

COM ..... Component Object Model

CMDB ..... Configuration Management Database

DBCC ..... Database Console Commands (dříve Database consistency checker)

DBMS ..... Database management systém

DDL ..... Data Definition Language

DML ..... Data Manipulation Language

DMV ..... Dynamic Management Views

DOS ..... Disk Operating systém

GIS ..... Geografický Informační Systém

IT ..... Informační Technologie

ITIL ..... Information Technology Infrastructure Library

LAN ..... Local Area Network

MS ..... Microsoft Inc.

ODBC ..... Open Database Connectivity

OLE DB ..... Object Linking and Embedding, Database

OLTP ..... Online Transaction Processing

OS ..... Operační Systém

SMP ..... Symantec Management Platform

SNI ..... SQL Server Network Interface

SQLOS ..... SQL Operating System

SŘBD ..... Systém řízení báze dat

SSMS ..... SQL Server Management Studio

TDS ..... Tabular Data Stream

XML ..... Extensive Markup Language

8.3 SEZNAM OBRÁZKŮ

8.5 SEZNAM TABULEK

8.6 SEZNAM SQL PŘÍKAZŮ (KÓDU)