VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Srovnání výkonu vybraných SŘBD

Comparison of performance of selected DBMS

2013 Petr Stodůlka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Stodůlka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Srovnání výkonu vybraných SŘBD
Comparison of Performance of Selected DBMS

Zásady pro vypracování:

V současnosti existuje velké množství různých SŘBD s odlišným výkonem provádění dotazů a uložených procedur. Cílem této práce je srovnání několika významných SŘBD pro různé typy dat a dotazů.

1. Popište základní charakteristiky vybraných SŘBD a jimi používané typy tabulek a indexů.
2. Pro vybraná data a dotazy srovnejte výkon vybraných SŘBD.
3. Porovnejte výkon provádění uložených procedur a funkcí.
4. Zhodnoťte výsledky.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Bednář**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

V této bakalářské práci se budu zabývat výkonnostním porovnáním vybraných SŘBD, jelikož jich v dnešní době existuje velké množství. Každé z těchto SŘBD disponuje odlišným výkonem prováděných dotazů a uložených procedur. Cílem této práce je srovnání několika významných SŘBD pro různé typy dat a dotazů. Tato práce se bude zaměřovat na SŘBD jako je Oracle a MS SQL Server. Budou zde popsány typy tabulek a indexů nad oběma SŘBD. Dále pak popis měřených transakcí a následně výsledky a porovnání z tohoto měření.

Klíčová slova: SŘBD, Oracle, MS SQL Server, tabulka, index, Benchmark, transakce

Abstract

In this master thesis I will discuss the performance comparison of selected DBMS, since there exists nowadays a large amount. Each of these DBMS has a different performance of queries and stored procedures. The goal of this work is to compare several major DBMSs for different types of data and queries. This work will be focus on the DBMS such as Oracle and MS SQL Server. There will describe the types of tables and indexes both DBMS. Furthermore, the description of the measured transaction and then comparing results of this measurement.

Keywords: DBMS, Oracle, MS SQL Server, table, index, Benchmark, transaction

Seznam použitých zkratek a symbolů

DDL	–	Data definition language
DML	–	Data manipulation language
OLTP	–	Online transaction processing
SŘBD	–	Systém řízení báze dat
TPC	–	Transaction Processing Performance
TPS	–	Transaction per second

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V současnosti existuje velké množství různých SŘBD s odlišným výkonem provádění dotazů a uložených procedur. Pokud jde o ty nejpoužívanější, patří zde například Oracle, MS SQL Server, MySQL atd. Dnes nejpoužívanějším SŘBD je jednoznačně Oracle z pohledu komerčního využití, kterého využívá cca 40% firem. Druhou příčku zaujímá MS SQL Server s 15%. Co se týče nekomerčního využití, zde převládá MySQL, který sice nedisponuje tak dobrými vlastnostmi, jak tomu je u Oracle a MS SQL Server, ale za to je cenově dostupnější.

Tato diplomová práce bude zaměřena na Oracle a MS SQL Server. Nad oběma těmito SŘBD budou popsány možné typy tabulek a indexů. Dále pak Benchmark TPC-E s použitým schématem, generátorem a příslušnými transakcemi. Každá z těchto uvedených transakcí bude změřena jak v původním návrhu databáze, tak i po optimalizaci tohoto návrhu. Ke každému návrhu budou uvedeny použité typy tabulek a indexů.

Součástí této práce budou také přiložené dva programy, kde jeden z nich obstarává vytvoření databáze na základě vybraných typů tabulek a indexů. Dále pak naplnění této databáze daty. V rámci druhého programu můžeme spouštět zatížení nad těmito SŘBD paralelním spouštěním vybraných transakcí s možností nastavení počtu spuštění. Pro spouštění těchto programů je využit virtuální stroj na školní síti pro lepší síťovou komunikaci.

2 Typy SŘBD

2.1 Microsoft SQL Server

Microsoft SQL Server představuje ucelenou sadu technologií a nástrojů, které uživatelům pomáhají vytěžit z informací maximální hodnotu. Nabízí vysokou úroveň výkonnosti, dostupnosti i zabezpečení, produktivnější nástroje pro správu a vývoj. SQL Server provádí měření na několika z nějvětších světových zatížení, což dokládá i výraznými standardními testovanými výsledky. SQL Server využívají firmy, jako jsou například Siemens nebo RedPrairie. SQL Server také pracuje s Microsoft Visual Studio [1] k poskytování integritních vývojových zkušeností, což umožňuje vývojářům pracovat v jednom prostředí napříč klientem, business vrstvy, a datové vrstvy.

2.1.1 Typy tabulek

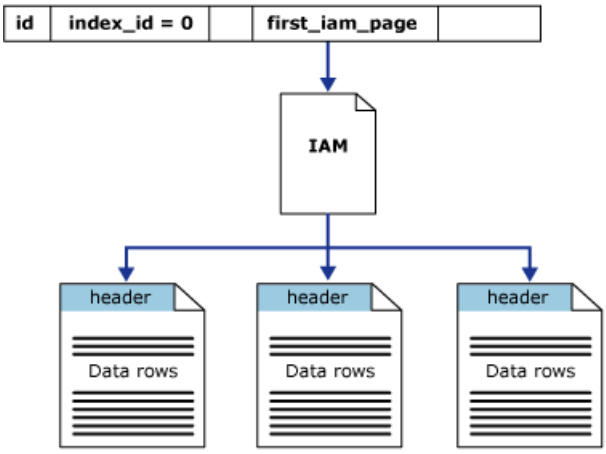
2.1.1.1 Heap table (tabulka typu halda) Jedná se o standardní databázovou tabulku [6], jenž se vytvoří po zadání příkazu *CREATE TABLE*. Po vytvoření nejsou na ni aplikovány žádné indexy s výjimkou primárních klíčů. Ovšem nad touto tabulkou můžeme později vytvořit indexy. Data jsou zde uloženy bez specifického pořadí. Obvykle jsou zpočátku data uloženy v takovém pořadí, jak byly vloženy do tabulky, ale databázový engine může přesouvat data k efektivnějšímu ukládání řádků. Tím pádem nelze předvídat pořadí dat. Aby bylo zaručeno pořadí řádků vrácených z tabulky, musíme použít klauzuli *ORDER BY*. Pro specifikaci pořadí ukládání řádků, je nutné vytvořit cluster index nad touto tabulkou, tím pádem tabulka už nebude typu halda. Cluster index nalezneme v kapitole 2.1.2.1.

Tento typ tabulky je výhodný v případě velkého počtu operací vkládání. Pokud nad tabulkou nebudou vytvořeny žádné nonclustered indexy, pak musí být procházena celá tabulka (operace *TABLE SCAN*) k nalezení patřičného záznamu. Toto může být přijatelné, pokud se bude jednat o tabulku s malým počtem záznamů. U tohoto typu tabulky jsou jednotlivé řádky identifikovány pomocí reference na řádkový identifikátor (RID) obsahující číslo souboru, číslo stránky a slot na stránce. Jedná se o malou efektivní strukturu. V některých případech se tabulka typu halda používá, když jsou data vždy přístupná prostřednictvím nonclustered indexů a RID je menší než clustered index klíče.

Tabulku typu halda bychom neměli používat při častém vrácení záznamů v setříděném pořadí, při seskupování záznamů a při rozsáhlém dotazování dat. Tento typ tabulky není vhodný, pokud se jedná o rozsáhlou tabulku bez indexů nad touto tabulkou, jelikož veškeré záznamy musí být přečteny pro nalezení daného záznamu tabulky.

Strukturu tabulky typu halda [7] znázorňuje obrázek 1.

[1] https://www.microsoft.com/cze/msdn/vstudio/



Obrázek 1: Struktura heap table

2.1.2 Typy indexů

2.1.2.1 Clustered index Tento typ indexu [8] třídí a ukládá záznamy v tabulce na základě jejich klíčové hodnoty. Můžeme jej vytvořit pouze jeden v rámci tabulky, jelikož záznamy samotné mohou být setříděny pouze v jednom pořadí.

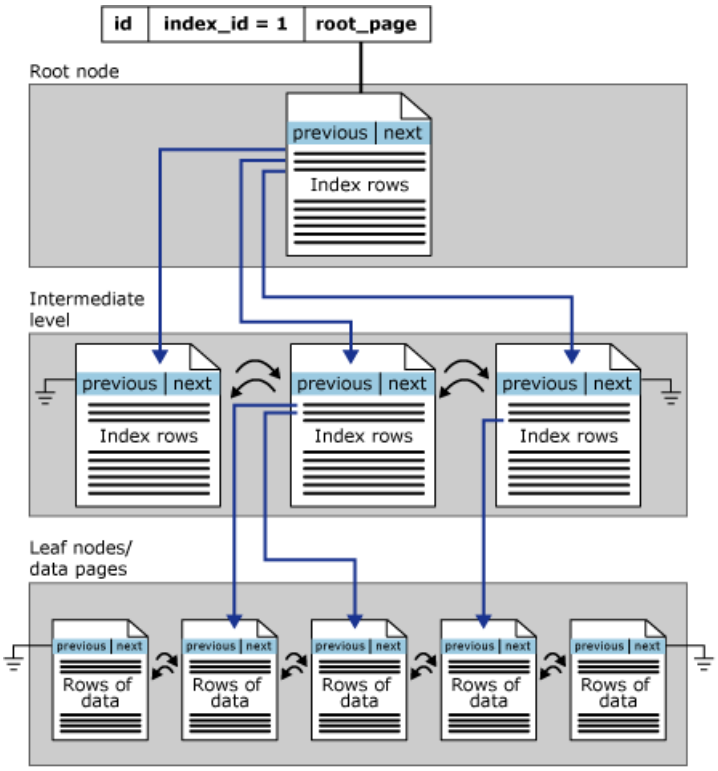
V SQL Serveru jsou indexy organizovány do struktury B-Stromu. U indexu typu B-Strom je každá stránka nazývána tzv. „index node“ neboli indexový uzel. Horní uzel B-Stromu se nazývá kořenový uzel. Naproti tomu spodní úroveň uzlů indexu nazýváme listové uzly. Jakékoliv úrovně indexu mezi kořenem a listovými uzly jsou označovány jako střední úrovně. U clustered indexu, listové uzly obsahují stránky příslušné tabulky. Kořen a uzly střední úrovně obsahují stránky indexu držící řádky indexu. Každý řádek indexu obsahuje klíčovou hodnotu a ukazatel a to buď na stránku střední úrovně v B-Stromu, nebo na řádek v listové úrovni indexu. Stránky každé úrovně indexu jsou propojeny tzv. dvojitým propojeným seznamem.

Clustered indexy můžeme naleznout v pohledu sys.partitions. Defaultně má Clustered index jediný „partition“ neboli segment. Pokud má ovšem více segmentů, tak každý segment má vlastní B-Stromovou strukturu obsahující data pro specifický segment. Pokud by měl index 4 segmenty, tak bychom měli 4 B-Stromové struktury, každá ve svém segmentu.

Strukturu cluster indexu [8] v jednom segmentu znázorňuje obrázek 2.

2.1.2.2 Nonclustered index Nonclustered indexy [9] mají stejnou B-Stromovou strukturu jako cluster indexy, až na následující významné rozdíly:

- Řádky příslušné tabulky nejsou setříděny a uloženy na základě jejich nonclustered klíče.



Obrázek 2: Struktura clustered indexu

- Listy nonclustered indexu jsou tvořeny ze stránek indexu nikoliv ze stránek tabulky.

Noncluster index může být definován na tabulce nebo pohledu s clustered indexem nebo také na tabulce typu halda. Každý řádek indexu obsahuje nonclustered klíčovou hodnotu a řádkový lokátor. Tento lokátor ukazuje na řádek tabulky v clustered indexu nebo tabulce typu halda mající klíčovou hodnotu.

Řádkové lokátory v řádcích nonclustered indexu jsou buď to ukazatel na řádek, nebo seskupený indexový klíč pro daný řádek, jak je popsáno níže:

- Pokud se jedná o tabulku typu halda, což znamená, že nedisponuje cluster indexem, řádkový lokátor je ukazatel na řádek. Ukazatel je sestaven z identifikátoru souboru (ID), čísla stránky, a počtu řádků na stránce. Tento ukazatel je znám jako Row ID (RID).
- Pokud tabulka obsahuje cluster index nebo je index na indexovaném pohledu, tak řádkový lokátor je klíčová hodnota cluster indexu pro daný řádek. Pokud cluster index není unikátní index, SQL Server vytvoří několik duplikátních unikátních klíčů přidáním interně generované hodnoty zvané uniquefier. Tato čtyř bytová hodnota není viditelná pro uživatele. Je přidána pouze tehdy, když se vyskytne požadavek na vytvoření unikátního clustered klíče pro použití v nonclustered indexech. SQL Server vyhledává řádek tabulky vyhledáním cluster indexu s využitím cluster index klíče uloženého v listu řádku noncluster indexu.

Strukturu cluster indexu [9] v jednom segmentu znázorňuje obrázek 3.

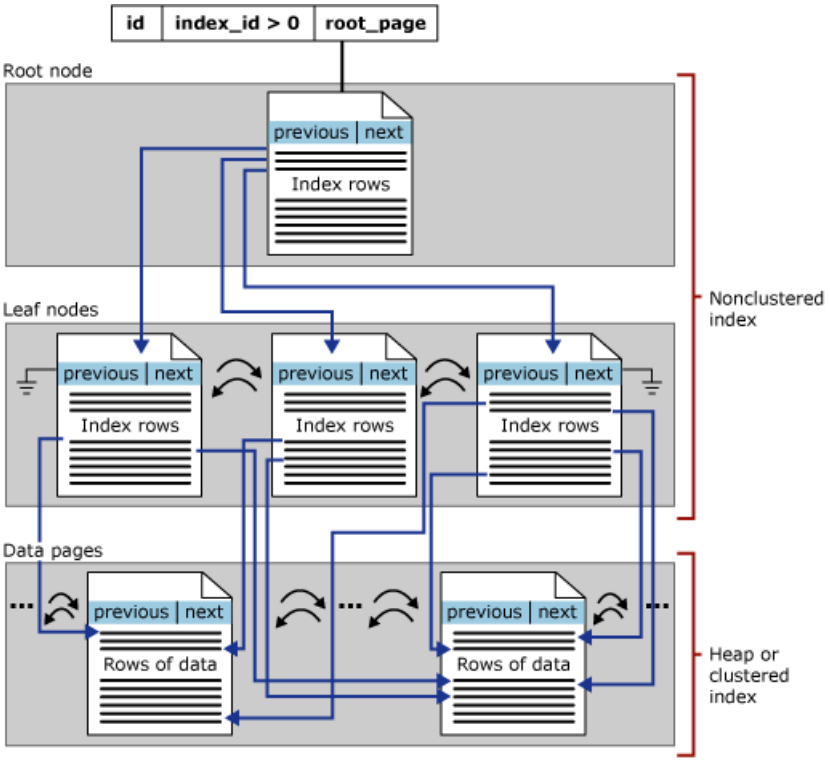
2.2 Oracle

Oracle se podílí na několika klíčových standardních testech k ověření výkonu a škálovatelnosti databáze Oracle. Transaction Processing Council (TPC) [2] organizuje srovnávací testy jak pro podporu rozhodování (DSS), tak i pro on-line zpracování transakcí (OLTP) [3]. DSS je počítačový informační systém, který podporuje řízení nebo organizační rozhodovací činnosti. Tyto testy jsou dále členěny na základě velikosti dané databáze. Kromě TPC, se Oracle podílí na SAP [4] testech pro posouzení výkonu databáze Oracle pomocí Enterprise Software.

2.2.1 Typy tabulek

2.2.1.1 Heap table (Tabulka typu halda) Jde o standardní databázovou tabulku [1], jenž se vytvoří po zadání příkazu *CREATE TABLE*. Jedná se o perzistentní stránkované pole s velikostí bloku nejčastěji 8kB. Data jsou řízena hromadným zpracováním a jsou uložena na první volnou pozici v segmentu, který velikostně odpovídá daným datům. Když jsou z tabulky data odstraněna, je uvolněn prostor k následnému opakovatelnému

[2] http://www.tpc.org/
[3] http://www.tpc.org/information/pdg.asp
[4] http://www.oracle.com/us/solutions/sap/overview/index.html



Obrázek 3: Struktura nonclustered indexu

použití operací *INSERT* nebo *UPDATE*. Toto je původ jména "heap", jenž se odkazuje na tento typ tabulky. Můžeme říci, že *heap* je shluk prostoru a je poněkud využíván v náhodném zpracování.

2.2.1.2 Index Organized Table Indexově organizovaná tabulka [10] má obdobnou úložnou strukturu jako B-Strom. Na rozdíl od běžné tabulky typu halda, jejichž data jsou uložena v neuspořádaném pořadí. Data v indexově organizované tabulce jsou uložena ve struktuře indexově B-Stromu, setříděné dle primárního klíče. Každý listový blok ukládá klíč a také sloupce neklíčových hodnot.

Struktura indexově organizované tabulky umožňuje následující výhody:

- Rychlý náhodný přístup k primárnímu klíči, jelikož je dostačující pouze prozkoumání indexu. A, protože zde neexistuje žádná samostatná tabulka jako úložný prostor, změna týkajících se dat (např. přidávání nových záznamů, mazání nebo aktualizace záznamů) povede pouze ke změně indexové struktury.
- Rychlý přístup v daném rozsahu primárního klíče, jelikož jsou řádky seskupeny dle pořadí primárního klíče.
- Nižší požadavky na uložení, jelikož zde nedochází k duplicitě primárních klíčů. Nejsou oba uloženy v indexu a tabulce, jak tomu je u tabulky typu halda.

Indexově organizovaná tabulka je vhodná pro OLTP aplikace, které vyžadují rychlý přístup k primárnímu klíči a vysokou dostupnost. Dotazy a DML příkazy nad seřazenými tabulkami používané v elektronickém zpracování objednávek jsou převážně založeny na primárním klíči a to nutně vede k časté potřebě reorganizace.

2.2.1.3 Index Clustered Table Indexově shlukovaná tabulka [1] je skupina tabulek, které sdílejí stejné datové bloky, jelikož sdílejí společné sloupce a je k nim často přistupováno společně. Pokud vytvoříme shlukovanou tabulku, Oracle fyzicky ukládá veškeré záznamy každé tabulky do stejných datových bloků. Hodnota shlukujícího klíče je hodnota sloupců shlukujícího se klíče pro konkrétní záznam. K takto vytvořené tabulce je třeba vytvořit také index na shlukující se klíč, před tím než použijeme DML příkazy nad touto tabulkou. Tento index se nazývá shlukující se index. Zajišťuje nám rychlý přístup k záznamům v rámci shluku, na základě shlukující se klíče. Pokud použijeme dotaz pro vyhledání záznamu v shluku na základě hodnoty shlukovaného klíče, Oracle vyhledá shlukovaný index pro nalezení hodnoty shlukovaného klíče a dále lokalizuje záznam ve shluku na základě ROWID. Shlukované tabulky můžeme aplikovat nad jednou nebo více tabulek.

2.2.1.4 Index Hash Table Tento typ tabulky [1] je velmi podobný konceptu indexově shlukovaným tabulkám s jedním hlavním rozdílem: indexově shlukovaný klíč je nahrazen hašovací funkcí. Index v tabulce reprezentuje data, není zde žádný fyzický index. Oracle vezme hodnotu klíče pro daný záznam, hašuje ji pomocí buď to vnitřní funkce, nebo

takovou jakou zadáme a použije ji k vyřešení, kde se data na disku nalézají. Z jedné strany se jedná o efektivní vyhledávání dat, ovšem efektivita používání hašovacího algoritmu k nalezení dat, je problematická u rozsahových dotazů, jelikož hašování nezachovává uspořádání. V takovém případě je nutné na tabulku aplikovat klasický index. Při tvorbě hašovací tabulky je nutné dobře zvolit nastavení tabulky:

- *HASHKEYS* - počet hašovacích hodnot, číslo je zaokrouhleno nahoru na nejbližší prvočíslo.
- *SIZE* - Celková velikost položek, které budou přiřazeny jedné hašovací hodnotě.

2.2.2 Typy indexů

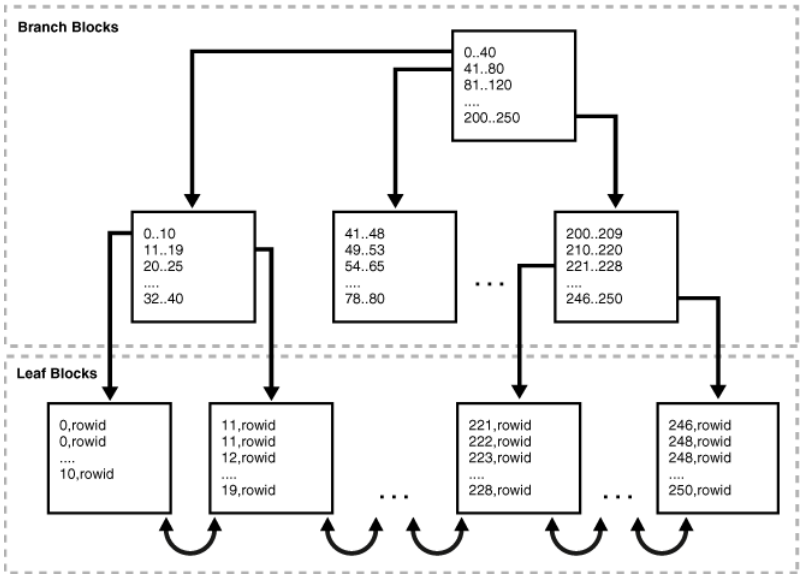
Indexy jsou výběrové struktury spojené s tabulkami a clustery, které umožňují SQL příkazům rychlé vykonání nad tabulkou. Oracle Database index [11] poskytuje rychlejší přístupovou cestu k datům tabulky. Indexy můžeme využívat bez přepisování dotazů. Výsledky těchto dotazů jsou stejné, ale rychlost vykonání je značně rychlejší. Indexy jsou logicky i fyzicky nezávislé na datech asociované tabulky. Tím, že se jedná o nezávislé struktury, vyžadují úložný prostor. Můžeme vytvářet nebo mazat indexy aniž by to ovlivnilo základní tabulky, databázové aplikace či ostatní indexy. Databáze automaticky udržuje indexy, když vkládáme, mažeme nebo aktualizujeme záznamy asociované tabulky. Pokud index smažeme, ostatní aplikace stále pracují. Ovšem přístup k předešlým indexovaným datům může být pomalejší.

2.2.2.1 B-Tree index B-Stromy, zkratka pro vyvážené stromy jsou nejčastějším typem databázového indexu. Tento index je seřazený seznam hodnot rozdělený do pásem. Sdružení klíče a záznamu nebo rozsahem záznamů, B-Stromy poskytují vynikající výkon vyhledávání pro širokou škálu dotazů, včetně přesné shody a rozsáhlému vyhledávání.

Obrázek 4 [11] znázorňuje strukturu indexu typu B-Strom. Index má dva typy bloků: *branch blocks* neboli větvové bloky pro vyhledávání a *leaf blocks* neboli listové bloky, které ukládají záznamy. Vyšší úroveň větvových bloků indexu typu B-Strom obsahují data indexu, které odkazují na nižší úroveň indexových bloků. Jak můžeme vidět na obrázku, kořenový větvový blok má záznamy 0-40, který odkazuje na blok umístěný nejvíce vlevo na následující úrovni větví. Tento větvový blok obsahuje záznamy jako 0-10 a 11-19. Každý z těchto záznamů odkazuje na listový blok obsahující klíčové hodnoty, které spadají do rozsahu.

Index je vyvážený, protože veškeré listové bloky automaticky zůstávají ve stejné hloubce. Takto, získávání jakéhokoliv záznamu z libovolného místa z indexu, přibližně zabere stejné množství času. Výška indexu je určena počtem bloků potřebných k přechodu od kořenového bloku k listovému. Úroveň větve je výška stromu mínus 1. Na obrázku 4 je výška indexu 3 a úroveň větví 2.

2.2.2.2 Bitmap index U bitmapového indexu [11], databáze ukládá bitmapy pro každý index klíče. U obvyklého indexu typu B-Strom, jeden záznam indexu okazuje na jediný



Obrázek 4: Struktura B-Tree indexu

záznam. Naproti tomu u bitmapového indexu, každý indexový klíč ukládá ukazatele na více řádků.

Bitmapové indexy jsou primárně určeny pro skladování dat nebo prostředí, ve kterém se dotazy odkazují na více sloupců v ad hoc režimu. Situace, ve kterých je vhodné použít bitmapový index, jsou následující:

- Indexované sloupce mají nízkou kardinalitu, to znamená, že počet různých hodnot, je malý ve srovnání s počtem řádků tabulky.
- Indexovaná tabulka je buď to pro čtení, nebo nepodléhá významné úpravě DML příkazů.

Každý bit v bitmapě odpovídá možnému ROWID. Pokud je bit nastaven, pak řádek s odpovídajícím ROWID obsahuje klíčovou hodnotu. Mapovací funkce převede pozici bitu na skutečné ROWID, takže bitmapový index poskytuje stejnou funkcionalitu jako index typu B-Strom, ačkoliv využívá odlišnou vnitřní reprezentaci.

3 Benchmark TPC-E

TPC-E benchmark [12] simuluje tzv. On-line Transaction Processing (OLTP) zatížení makléřské firmy. Zaměření benchmarku je centrální databáze, která vykonává transakce související s firemními zákaznickými účty, průzkumu trhu a obchodováním. Makléřská firma spolupracuje s finančními trhy, což vede k vykonávání dotazů jménem zákazníků a aktualizace příslušných informací o účtu. Benchmark je stupňovatelný, což znamená, že počet zákazníků stanovených pro makléřské firmy může být odlišný, aby se stanovilo pracovní zatížení různě velkých podniků. Benchmark stanovuje požadovanou kombinaci transakcí, jež musí podporovat. TPC-E metrika je dána transakcemi za vteřinu transaction per second (tps). To konkrétně odkazuje k počtu obchodních transakcí, které server musí v průběhu času provést. V souladu s cílem měření výkonnostních charakteristik databázového systému, neklade důraz na měření komplexního toku dat mezi více aplikačními systémy, které by existovaly v reálném prostředí.

3.1 Návrh databáze

TPC-E databáze se skládá ze 33 samostatných a individuálních tabulek. Schéma databáze je rozděleno do 4 sad tabulek:

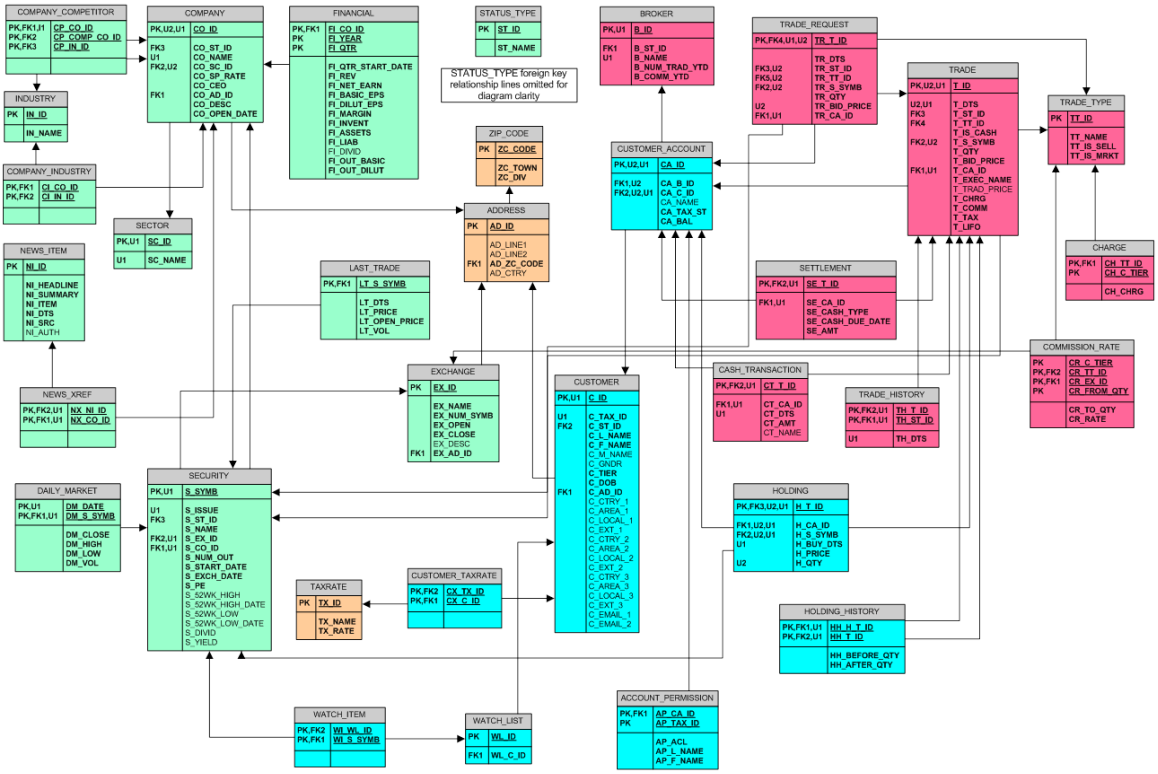
- **Customer Tables:** Tato sada obsahuje 9 tabulek, které obsahují informace o zákaznících z makléřské firmy.
- **Broker Tables:** Tato sada obsahuje 9 tabulek, které obsahují informace o makléřské firmě a makléřských souvisejících údajích.
- **Market Tables:** Tato sada obsahuje 11 tabulek, které obsahují informace o společnostech, trzích, burzách a průmyslových sektorech.
- **Dimension Tables:** Tato sada obsahuje 4 dimenzionální tabulky, které obsahují obecné informace, jako jsou adresy a poštovních směrovací čísla.

Odpovídající schéma databáze [4] reprezentuje obrázek 5.

3.1.1 Definice tabulek

Customer Tables

- **Account Permission:** Tato tabulka obsahuje informace o přístupu zákazníka nebo jiné osoby, než je zákazník k zákaznickým účtům. Nad těmito účty mohou být prováděny transakce, více než jednou osobou.
- **Customer:** Tabulka obsahuje informace o zákaznících makléřské firmy.
- **Customer Account:** Tabulka obsahuje informace o účtu vztahující se k účtům každého zákazníka.



Obrázek 5: Schéma databáze

- **Customer Taxrate:** Tabulka obsahuje dvě reference, a to jak na zákazníka, tak do tabulky daňové sazby. Jedna reference je pro státní/provinciální daň, druhá je pro národní daň. Tabulka obsahuje skutečné daňové sazby.
- **Holding:** Tabulka obsahuje informace o vlastněných cenných papírech zákaznických účtů.
- **Holding History:** Tabulka obsahuje informace ohledně vlastnictví (držba), které byly vloženy, aktualizovány nebo odstraněny, a které transakce provedly danou změnu.
- **Holding Summary:** Tabulka obsahuje souhrnné informace o vlastněných cenných papírech zákaznických účtů.
- **Watch Item:** Tabulka obsahuje seznam cenných papírů určených ke sledování na seznamu sledovaných.
- **Watch List:** Tabulka obsahuje informace o zákazníkovi, který vytvořil tento seznam sledovaných.

Broker Tables

- **Broker:** Tabulka obsahuje informace o makléřích.
- **Cash Transaction:** Tabulka obsahuje informace o hotovostních transakcích.
- **Charge:** Tabulka obsahuje informace o poplatcích za zavedení obchodního požadavku. Poplatky jsou založeny na rejstříku zákazníka (úrovni) a typ obchodní transakce.
- **Commission Rate:** Sazba provize závisí na několika faktorech: úrovni zákazníka, typu obchodní transakce, množství převedených cenných papírů a burza, která provádí transakci.
- **Settlement:** Tabulka obsahuje informace o tom, jak se vyřizují obchodní transakce. Obzvláště, zda vyrovnání je na marži (záloha) nebo v hotovosti a kdy je vyrovnání splatné.
- **Trade:** Tabulka obsahuje informace o obchodních transakcích.
- **Trade History:** Tabulka obsahuje historii každé obchodní transakce prostřednictvím různých stavů.
- **Trade Request:** Tabulka obsahuje informace o očekávaných obchodních transakcí, které čekají na určitou cenu cenných papírů předtím, než jsou transakce předloženy na trh.
- **Trade Type:** Tabulka obsahuje seznam platných obchodních typů.

Market Tables

- **Company:** Tabulka obsahuje informace o všech společnostech s veřejně obchodovanými cennými papíry.
- **Company Competitor:** Tato tabulka obsahuje informace pro konkurenci dané společnosti a odvětví, ve kterém firma konkuruje.

- **Daily Market:** Tabulka obsahuje denní statistiky trhu pro každý cenný papír, s použitím uzavřených tržních dat z posledního ukončeného obchodního dne. EGenLoader načte tuto tabulku s údaji pro každý cenný papír za období Od 3. ledna 2000 a do 31. prosince 2004.
- **Exchange:** Tabulka obsahuje informace o finančních burzách.
- **Financial:** Tabulka obsahuje informace ohledně čtvrtletních finančních zpráv dané společnosti. EGenLoader načte tuto tabulku s finančními informacemi pro každou společnost pro Quarters počínaje dnem 1. ledna 2000 a končící čtvrtletí, které začíná 1. října 2004.
- **Industry:** Tabulka obsahuje informace o průmyslu. Používané pro kategorizaci, ve které se nachází daná společnost.
- **Last Trade:** Tabulka obsahuje jeden řádek pro každý cenný papír s nejnovějšími tržními cenami a množství pro každý cenný papír.
- **News Item:** Tabulka obsahuje informace o novinkách obchodu.
- **News Xref:** Tabulka obsahuje vzájemný odkaz na tabulku News Item a také tabulku Company, které jsou uvedeny v tabulce News Item.
- **Sector:** Tabulka obsahuje informace o odvětví trhu.
- **Security:** Tato tabulka obsahuje informace o každém cenném papíru, s kterým bylo obchodováno na jakékoliv burze.

Dimension Tables

- **Address:** Tabulka obsahující informace o adrese.
- **Status Type:** Tato tabulka obsahuje všechny stavové hodnoty pro několik různých použitých stavů. Více tabulek odkazuje na tuto tabulku, pro získání jejich stavových hodnot.
- **Taxrate:** Tabulka obsahuje informace o daňových sazbách.
- **Zip Code:** Tabulka obsahuje směrovací čísla, města a divize, které jim náleží.

3.2 Generátor

EGen [3] je TPC poskytovaný software navržený k usnadnění implementace TPC-E. EGen poskytuje:

- konzistentní generování dat nezávisle na daném prostředí
- generování transakcí a formulace řízení toku správy
- zhotovení projektů a velkou škálu šablon

3.3 Transakce

TPC-E benchmark obsahuje 11 transakcí [3] a jednu transakci sloužící pro vyčištění. Chceme-li vytvořit rozumně vyváženou zátěž, která se podobá skutečnému výrobnímu prostředí, tak transakce musí pokrývat širokou škálu funkcí systému. Deset transakcí dodržují určitou kombinaci generování požadované zátěže při jednoduchém, opakovaném a snadném vykonání, jenž nám poskytuje prostředí benchamrku. Jedenáctá transakce není součástí této kombinace, ale je vykonávána ve stanovených intervalech. Tato transakce nazývaná "Data-Maintenance"simuluje administrativní údržbu tabulek, které jinak nejsou modifikovány, již zmíněnou kombinací transakcí. Transakce sloužící pro vyčištění nazývaná "Trade-Cleanup"je k dispozici pro vyčištění nevyřízených a předložených obchodních transakcí, které mohou existovat z předchozího běhu.

Každá z těchto databázových transakcí neboli jednotka práce s databází splňuje vlastnost ACID [5], tedy A atomičnost, C korektnost, I izolovanost a D trvalost. Jednotlivé transakce jsou definovány tabulkami. Jedna z nich nám popisuje vstupní/výstupní parametry. Druhá pak, jaké tabulky a atributy v dané sekci budou ovlivněny danou transakcí. Sloupce této tabulky specifikují následující:

- První sloupec nám udává název databázové tabulky.
- Druhý sloupec udává následující:
  - Specifikuje nám název sloupce databázové tabulky.
  - Řetězec "# rows", jenž specifikuje přesný počet řádků obsahující všechny sloupce databázové tabulky. Například, "2 rows"indikuje 2 celistvé řádky databátové tabulky.
  - Řetězec "Rows()"specifikuje variabilní počet řádků obsahující všechny sloupce databázové tabulky.
- Zbývající sloupce tabulky odpovídají každé sekci transakce obsahující databázové interakce nebo transakční řídící operace, jenž musí být provedeny v rámci této sekce.
- Tyto zbývající sloupce na posledním řádku tabulky tzv. "Transaction Controll", kde každý sloupec tohoto řádku odpovídá jedné sekci transakce. Obsah sloupců udává, které transakční řídící operace se vyskytují v této sekci. Možné transakční řídící operace jsou následující:
  - Slovo "Start"indikuje, že specifická sekce obsahuje řídící operace k zahájení databázové transakce. Zahájení databázové transakce se může vyskytovat pouze v sekci, kde je slovo "Start"specifikováno.
  - Slovo "Rollback"indikuje, že specifická sekce obsahuje řídící operace, které navrátí databázové transakce do původního stavu. Navrácení databázové transakce do původního stavu se může provést pouze v sekci, kde je slovo "Rollback"specifikováno.

[5] http://databases.about.com/od/specificproducts/a/acid.htm

– Slovo ”Commit”indikuje, že specifická sekce obsahuje řídící operace k potvrzení databázové transakce.

Jak taková to tabulka může vypadat, znázorňuje tabulka 1.

Tables	**Columns**	**Frames**		
		1	**2**	**3**
CUSTOMER_ACCOUNT	CA_BAL	Reference		
	CA_C_ID	Return		
	CA_AX_ST	Return		
HOLDING	H_PRICE		Return	
	H_QTY		Modify	
	Row(s)		Remove	
	1 row		Add	
TRADE_HISTORY	1 row			Add
Transaction Control		Start	Rollback	Commit

Tabulka 1: Názorná tabulka transakce

3.3.1 Transakce Broker-Volume

Tato transakce [3] je navržena tak, aby napodobovala interní obchodní zpracování makléřské firmy. Příkladem této transakce by měl být, manažer tvořící zprávu o současném potenciálním výkonu různých makléřů.

Transakce se skládá pouze z jedné sekce. Dále pak vyhledává nevyřízené objednávky k nalezení takových objednávek, které jsou spojené s daným seznamem makléřů odpovědných za cenné papíry daného odvětví. Hodnota každé objednávky je vypočtena na základě nabídkové ceny a množství akcií a je přidána do systému ke zpracování pro příslušného makléře. Tato transakce vrací seznam makléřů s odpovídajícím množství objednávek seřazených sestupně.

Tato transakce provádí pouze operace čtení, a tudíž neprovádí žádné modifikace v databázi. S jakými tabulkami a jejími atributy se transakce pojí, můžeme nalézt v tabulce 2. Příslušné parametry této transakce nalezneme v tabulce 3. Pseudokód nalezneme v příloze B ve výpisu 1.

3.3.1.1 Transakce Broker-Volume sekce 1 z 1

Metody přístupu do databáze v sekci 1 jsou pouze návratové.

Tables	Columns	Frames 1
BROKER	B_NAME	Return
TRADE_REQUEST	TR_BID_PRICE TR_QTY	Reference Reference
Transaction Control		Start Commit

Tabulka 2: Broker-Volume Footprint

Parameter	Direction	Description
broker_list[]	IN	Seznam dvaceti až čtyřiceti různých makléřských jmen definovaných atributem B_NAME v tabulce BROKER. Tyto jména jsou náhodně vybrána ze seznamu makléřů.
sector_name	IN	Náhodně vybraný název sektoru definovaný atributem SC_NAME v tabulce SECTOR.
broker_name	OUT	Seznam makléřských jmen uspořádaných sestupně dle množství objednávek spjatými s daným makléřem.
list_len	OUT	Počet položek v seznamu při vracení.
volume[]	OUT	Seznam čísel, uspořádaných sestupně, reprezentující součet všech hodnot nevyřízených obchodních transakcí (TR_QTY * TR_BID_PRICE) v tabulce TRADE_REQUEST pro cenné papíry v daném sektoru sdruženými se jmény makléřů stanovený pomocí broker_list. Velikost seznamu je určen parametrem list_len.

Tabulka 3: Parametry transakce Broker-Volume

3.3.2 Transakce Customer-Position

Tato transakce [3] je navržena tak, aby napodobovala proces získávání profilu zákazníka a shrnutí jeho celkového postavení na základě aktuálních tržních cen v rámci celého majetku. Toto reprezentuje vykonanou práci, kdy si zákazník položí otázku "Kolik jsem dnes vydělal"?

Tato transakce se skládá ze tří sekcí, (sekce 2 a 3 se vzájemně vylučují). Zákazníka můžeme určit, buď to pomocí ID zákazníka, nebo ID daně zákazníka. Pokud ID zákazníka předané transakci nabývá hodnoty 0, pak je využito ID daně zákazníka k vyhledání tohoto ID. Tato transakce nám vrací profil daného zákazníka. Kromě veškerých účtů zákazníka jsou nám také vráceny peněžní zůstatek na účtu a celková současná tržní hodnota vlastnictví daného účtu.

Pokud byla vyžadována historie obchodní činnosti, jsou vyhledány informace z deseti nejnovějších obchodních transakcí pro náhodně vybraný účet z účtů daného zákazníka.

S jakými tabulkami a jejími atributy se transakce pojí, můžeme nalézt v tabulce 4. Příslušné parametry této transakce nalezneme v tabulce 5.

3.3.2.1 Transakce Customer-Position sekce 1 z 3 Pokud vstupní parametr cust_id nabývá hodnoty 0, tato sekce musí použít vstupní parametr tax_id k vyhledání tabulky CUSTOMER a ID zaměstnance této tabulky. Tato sekce získává detailní informace o daném zákazníku a vyhledává peněžní zůstatek pro každý z jeho účtů, jakož i celkovou hodnotu vlastnictví na jednotlivých účtech. Kromě podrobné informace o zákazníkovi, tato sekce vrací seznam účtů, asociovaný peněžní zůstatek a hodnoty majetku seřazeny podle této hodnoty.

Metody přístupu do databáze v sekci 1 jsou referenční a návratové. Odpovídající pseudokód pro tuto sekci nalezneme v příloze B ve výpisu 2.

3.3.2.2 Transakce Customer-Position sekce 2 z 3 Tato sekce je provedena pouze v případě, pokud je hodnota parametru get_history nastavena na TRUE. Použití ID účtu zákazníka, sekce musí vyhledat tabulky TRADE a TRADE_HISTORY k nalezení až 30 řádků historie, které odpovídají 10 naposledy provedených obchodních transakcí podle účtu zákazníka. Pro každou událost musí sekce vrátit hodnoty T_ID (identifikátor obchodní transakce), T_S_SYMB (symbol cenného papíru vyskytující se, v dané obchodní transakci), T_QTY (počet cenných papírů v rámci obchodní transakce), TH_DTS (Časový údaj, kdy byla aktualizována historie obchodních transakcí), a ST_NAME (stav obchodní transakce) pro všechny události v sestupném pořadí podle data nalezené v TH_DTS. Tato sekce dokončuje práci a potvrzuje transakci.

Metody přístupu do databáze v sekci 2 jsou pouze návratové. Odpovídající pseudokód pro tuto sekci nalezneme v příloze B ve výpisu 3.

3.3.2.3 Transakce Customer-Position sekce 3 z 3 Tato sekce je provedena pouze v případě, pokud je hodnota parametru get_history nastavena na FALSE. Sekce pouze potvrzuje transakci, která začala v sekci 1 a vrací status.

V této sekci nejsou metody přístupu do databáze, pouze využívá transakční řídící operace. Odpovídající pseudokód pro tuto sekci nalezneme v příloze B ve výpisu 4.

Parameter	Direction	Description
cust_id	IN/OUT	ID zákazníka nabývající hodnot 0 nebo ID.
tax_id	IN	ID daně zákazníka nebo prázdný řětězec.
acct_id	IN	ID účtu daného zákazníka.
acct_id[max_acct_len]	OUT	Seznam ID účtů zákazníka.

Pokračování na následující stránce

Tabulka 5 – pokračování z předcházející stránky

Parameter	Direction	Description
acct_len	OUT	Počet účtů zakazníka (max_acct_len (10) nebo méně).
asset_total[max_acct_len]	OUT	Seznam celkového majetku pro každý účet daného zákazníka.
c_ad_id	OUT	Adresa zákazníka.
c_area_1	OUT	Předčíslí pro zákazníkovo první telefonní číslo.
c_area_1	OUT	Předčíslí pro zákazníkovo druhé telefonní číslo.
c_area_1	OUT	Předčíslí pro zákazníkovo třetí telefonní číslo.
c_ctry_1	OUT	Státní kód pro zákazníkovo první telefonní číslo.
c_ctry_2	OUT	Státní kód pro zákazníkovo druhé telefonní číslo.
c_ctry_3	OUT	Státní kód pro zákazníkovo třetí telefonní číslo.
c_dob	OUT	Zákazníkovo datum narození.
c_email_1	OUT	Zákazníkův první email.
c_email_2	OUT	Zákazníkův druhý email.
c_ext_1	OUT	Zákazníkova pobočka pro první telefonní číslo.
c_ext_2	OUT	Zákazníkova pobočka pro druhé telefonní číslo.
c_ext_3	OUT	Zákazníkova pobočka pro třetí telefonní číslo.
c_f_name	OUT	Křestní jméno zákazníka.
c_gndr	OUT	Pohlaví zákazníka.
c_l_name	OUT	Příjmení zákazníka.
c_local_1	OUT	Zákazníkovo první telefonní číslo.
c_local_2	OUT	Zákazníkovo druhé telefonní číslo.
c_local_3	OUT	Zákazníkovo třetí telefonní číslo.
c_m_name	OUT	Prostřední jméno zákazníka.
c_st_id	OUT	ID statusu zákazníka.
c_tier	OUT	Úroveň zákazníka.
cash_bal[max_acct_len]	OUT	Seznam peněžních zůstatku pro každý z účtů daného zákazníka.
hist_dts[max_hist_len]	OUT	Seznam datumů pro každé datum transakce z historie transakcí.
hist_len	OUT	Počet záznamů z historie transakcí, nejvyšší možná hodnota max_hist_len což je 30.
qty[max_hist_len]	OUT	Počet akcií zahrnuty v každé události historie.
symbol[max_hist_len]	OUT	Cenné papíry zahrnuty v každé události historie.
trade_id[max_hist_len]	OUT	ID obchodní transakce každé události z historie.
trade_status[max_hist_len]	OUT	Status obchodní transakce každé události z historie.

Tabulka 5: Parametry transakce Customer-Position

Tables	Columns	Frames 1	2	3
CUSTOMER	C_AD_ID	Return		
	C_AREA_1	Return		
	C_AREA_2	Return		
	C_AREA_3	Return		
	C_CTRY_1	Return		
	C_CTRY_2	Return		
	C_CTRY_3	Return		
	C_DOB	Return		
	C_EMAIL_1	Return		
	C_EMAIL_2	Return		
	C_EXT_1	Return		
	C_EXT_2	Return		
	C_EXT_3	Return		
	C_F_NAME	Return		
	C_GNDR	Return		
	C_L_NAME	Return		
	C_LOCAL_1	Return		
	C_LOCAL_2	Return		
	C_LOCAL_3	Return		
	C_M_NAME	Return		
	C_ST_ID	Return		
	C_TIER	Return		
CUSTOMER_ACCOUNT	CA_BAL	Return		
	CA_ID	Return		
HOLDING_SUMMARY	HS_QTY	Reference		
LAST_TRADE	LT_PRICE	Reference		
STATUS_TYPE	ST_NAME		Return	
TRADE_HISTORY	TH_DTS		Return	
TRADE	T_ID		Return	
	T_QTY		Return	
	T_S_SYMB		Return	
Transaction Control		Start	Commit	Commit

Tabulka 4: Customer-Position Footprint

3.3.3 Transakce Market-Watch

Market-Watch transakce [3] je navržena tak, aby napodobovala proces sledování celkové výkonnosti trhu, umožňující zákazníkům sledování aktuálních denních tendencí (růst nebo propad) sledu cenných papírů. Tento sled může být založen na pozici vlastnictví zákazníka, zákazníkův potenciální seznam sledovaných cenných papírů nebo konkrétní odvětví průmyslu.

Tato transakce se skládá pouze z jedné sekce. Vypočítává procentní změnu hodnoty tržního financování sledu cenných papírů na vybraných denních uzavíracích cenách ve srovnání s běžnými tržními cenami. Zvolený den je nerovnoměrně vybrán z 1305 dnů tržních dat, která byla načtena během počátečního naplnění databáze. Výpočet se provádí při pohledu na zvolenou denní uzavírací cenu pro každý cenný papír ze seznamu a vynásobením s počtem význačných akcií pro daný cenný papír. Tento produkt se přidá do průběžného součtu ke zvolenému dni finanční uzávěrky na trhu. Kromě toho, aktuální cena každého cenného papíru ze seznamu je vynásobena počtem význačných akcií pro daný cenný papír. Tento produkt je přidán do průběžného součtu pro aktuální financování trhu. Rozdíl mezi celkovým tržním financováním dané uzávěrky a aktuální celkovou, je vyjádřeno v procentech.

Tato transakce podporuje výpočet tržního sledování skupiny cenných papírů, vybraných na základě následujících kritérií:

- **Prospective-Watch** - Sled cenných papírů je vybrán s použitím všech cenných papírů v zákazníkově seznamu sledovaných.
- **Industry-Watch** - Sled cenných papírů se volí pomocí veškerých cenných papírů v daném průmyslu spojenými se společnostmi v určitém rozsahu. Jméno průmyslu je náhodně vybráno z možných jmen pomocí rovnoměrného přidělení.
- **Portfolio-Watch** - Sled cenných papírů je vybrán pomocí veškerých vlastněných cenných papírů, které jsou uvedeny v účtu zákazníka. Identifikátor zákaznického účtu je náhodně zvolen ze všech možných účtů tohoto zákazníka.

S jakými tabulkami a jejími atributy se transakce pojí, můžeme nalézt v tabulce 6. Příslušné parametry této transakce nalezneme v tabulce 7. Pseudokód nalezneme v příloze B ve výpisu 5.

3.3.3.1 Transakce Market-Watch sekce 1 z 1

Metody přístupu do databáze v sekci 1 jsou pouze referenční.

Tables	Columns	Frames 1
COMPANY	CO_ID CO_IN_ID	Reference Reference
DAILY_MARKET	DM_CLOSE	Reference
HOLDING_SUMMARY	HS_S_SYMB	Reference
INDUSTRY	IN_ID IN_NAME	Reference Reference
LAST_TRADE	LT_PRICE	Reference
SECURITY	S_CO_ID S_NUM_OUT S_SYMB	Reference Reference Reference
WATCH_ITEM	WI_S_SYMB	Reference
WATCH_LIST	WL_C_ID WL_ID	Reference Reference
Transaction Control		Start Commit

Tabulka 6: Market-Watch Footprint

Parameter	Direction	Description
acct_id	IN	Daný zákazník je vybrán z řady možných zákazníků. ID účtu daného zákazníka je definováno pomocí atributu CA_ID tabulky CUSTOMER_ACCOUNT, tato hodnota je vybrána náhodně z rozsahu dostupných účtů daného zákazníka. Tento vstup bude využíván 35% času. Sled cenných papírů budou ty z daného účtu zákazníka. Dalších 65% času, kdy tento vstup nebude použit, bude nabývat hodnoty 0.
cust_id	IN	Číslo náhodně vybrané z možných identifikátorů zákazníka, definovaných atributem C_ID tabulky CUSTOMER. Tento vstup bude použit 60% času. Sled cenných papírů budou ty ze seznamu sledovaných zákazníkem. Dalších 40% času, kdy tento vstup nebude použit, bude nabývat hodnoty 0.

Pokračování na následující stránce

Tabulka 7 – pokračování z předcházející stránky

Parameter	Direction	Description
ending_co_id	IN	Identifikátor poslední společnosti z rozsahu 5,000 společností v daném průmyslu dle IN_NAME. Hodnota bude nabývat ending_co_id + 4,999. Tento vstup bude použit pouze v případě, když bude použit industry_name, což odpovídá 5% času. Dalších 95% času, kdy tento vstup nebude použit, bude nabývat hodnoty 0.
industry_name	IN	Náhodně vybrané jméno průmyslového odvětví definované atributem IN_NAME tabulky INDUSTRY. Tento vstup bude použit 5% času. Sled cenných papírů budou ty ze společností daného průmyslu. Dalších 95% času, kdy tento vstup nebude použit, bude nabývat hodnoty 0.
start_date	IN	Den nerovnoměrně vybrán z 1305 dnů z tabulky DAILY_MARKET. Uzavírací cena těchto cenných papírů pro tento den bude využita v tržní finanční kalkulaci.
starting_co_id	IN	Číslo náhodně vybrané z rozsahu možných společností minus 5,000. Identifikátor společnosti první společnosti v rozsahu 5,000 k nalezení IN_NAME v daném průmyslovém odvětví. Tento vstup bude použit pouze v případě, když bude použit industry_name, což odpovídá 5% času. Dalších 95% času, kdy tento vstup nebude použit, bude nabývat hodnoty 0.
pct_change	OUT	Číselná hodnota vypočtená během vykonávání transakce hledáním procentuální změny z uzávěrky obchodního financování pro sled cenných papírů a jejich aktuální financování.

Tabulka 7: Parametry transakce Market-Watch

3.3.4 Transakce Trade-Status

Tato transakce [3] je navržena tak, aby napodobovala proces poskytující aktuální informace o stavu konkrétních sérií obchodních transakcí. Jedná se o přezkoumání zákazníkova souhrnu posledních obchodních aktivit, pro jeden z jeho účtů.

Tables	Columns	Frames 1
BROKER	B_NAME	Return
CUSTOMER	C_F_NAME	Return
	C_L_NAME	Return
EXCHANGE	EX_NAME	Return
SECURITY	S_NAME	Return
STATUS_TYPE	ST_NAME	Return
TRADE	T_CHRG	Return
	T_DTS	Return
	T_EXEC_NAME	Return
	T_ID	Return
	T_QTY	Return
	T_S_SYMB	Return
TRADE_TYPE	TT_NAME	Return
Transaction Control		Start Commit

Tabulka 8: Trade-Status Footprint

S jakými tabulkami a jejími atributy se transakce pojí, můžeme nalézt v tabulce 8. Příslušné parametry této transakce nalezneme v tabulce 9. Pseudokód nalezneme v příloze B ve výpisu 6.

3.3.4.1 Transakce Trade-Status sekce 1 z 1

Metody přístupu do databáze v sekci 1 jsou pouze návratové.

Parameter	Direction	Description
acct_id	IN	Daný zákazník je vybrán z řady možných zákazníků. ID účtu daného zákazníka je definováno pomocí atributu CA_ID tabulky CUSTOMER_ACCOUNT, tato hodnota je vybrána náhodně z rozsahu dostupných účtů daného zákazníka.
status	OUT	Kód udávající stav provedení této transakce.

Pokračování na následující stránce

Tabulka 9 – pokračování z předcházející stránky

Parameter	Direction	Description
status_name[]	OUT	Seznam znakových řetězců, každý z těchto řetězců je definovaný atributem ST_NAME tabulky STATUS_TYPE, reprezentující aktuální stav obchodní transakce.
trade_id[]	OUT	Seznam čísel, kde každé číslo je definované atributem T_ID tabulky TRADE, přidělené burzou ke specifikaci určité obchodní transakce, před jejím vyžádáním.

Tabulka 9: Parametry transakce Trade-Status

3.3.5 Transakce Trade-Cleanup

Tato transakce [3] je určená ke zrušení veškerých čekajících nebo nabízených obchodních transakcí v databázi. Transakce využívá uloženou proceduru *Fill_Trade_Request*, která naplní obsah tabulky TRADE_REQUEST.

S jakými tabulkami a jejími atributy se transakce pojí, můžeme nalézt v tabulce 10. Příslušné parametry této transakce nalezneme v tabulce 11. Pseudokód nalezneme v příloze B ve výpisu 7.

Parameter	Direction	Description
st_canceled_id	IN	Identifikátor pro status zrušení obchodních transakcí - předán pro snadné srovnávání.
st_pending_id	IN	Identifikátor pro status zahájení obchodních transakcí - předán pro snadné srovnávání.
st_submitted_id	IN	Identifikátor pro status nabízených obchodních transakcí - předán pro snadné srovnávání.
trade_id	IN	Identifikátor obchodních transakcí využitý pro zjištění nedokončených, nabízených a/nebo zahájených obchodních transakcí.

Tabulka 11: Parametry transakce Trade-Cleanup

Tables	Columns	Frames 1
TRADE	T_DTS T_ID T_ST_ID	Modify Reference Modify
TRADE_HISTORY	Row(s)	Add
TRADE_REQUEST	Row(s) TR_T_ID	Remove Reference
Transaction Control		Start Commit

Tabulka 10: Trade-Cleanup Footprint

4 Experimentální výsledky

4.1 Použité nástroje

Pro testování jsem si vytvořil vlastní program. Podrobný popis nalezneme v příloze C. Tento program zvládá vytvoření databáze s vybranými typy indexů a tabulek. V rámci tohoto programu můžeme naplnit jednotlivé tabulky databáze. Odpovídající schéma databáze nalezneme v kapitole 3.1, obrázek 5. Jako vstupní data slouží vygenerované textové soubory, jejichž velikost činí 36 GB. Tyto data jsou vygenerované pomocí generátoru TPC-E Benchmarku konkrétně tedy *EGenLoader*. Tento generátor nalezneme v kapitole 3.2. Při vkládání dat do databáze jsou zde využity hromadné operace k dosažení maximálního výkonu. Tyto zmíněné úkony můžeme provádět nad různými SŘBD a to konkrétně SQL Server a Oracle.

V tomto programu také můžeme spouštět a zároveň měřit jednotlivé transakce nad oběma SŘBD. Je zde možnost zvolení počtu volání dané transakce. Po zavolání jsou nám zobrazeny naměřené časové výsledky. Započetí měření probíhá v rámci databáze, nikoliv při zavolání procedury z programu. Ukončení měření probíhá také v rámci databáze. Toto opatření má předejít negativním vlivům při komunikaci k dosažení co nejpřesnějších výsledků.

Tento program je zcela univerzální. Pokud budeme chtít změnit věci jako například *connection string* [6] nebo cestu k vygenerovaným textovým souborům, stačí pouze změnit informace v konfiguračním souboru.

Program je napsán v jazyce C# s použitím .NET Frameworku 4 [7]. Je zde použito několik návrhových vzorů, jako například Data Mapper a Domain Model [2]. Diagram komponent pak představuje obrázek 6.

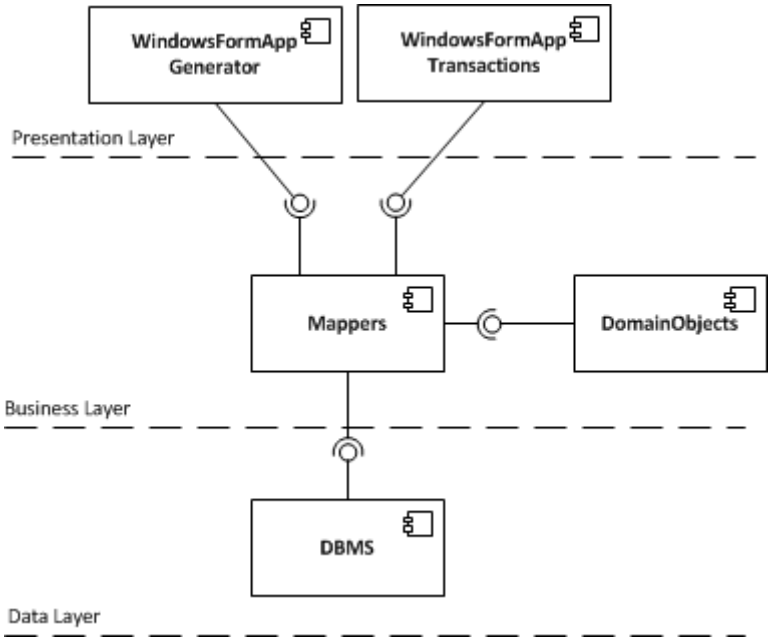
V rámci této práce jsem měl k dispozici server *dbsys.cs.vsb.cz* s parametry: 2x Intel Xeon E5 2690 2.9GHz, 2x 12C, 288GB paměti. Tento server běží na 64b operačním systému Windows server 2008 R2 datacenter service pack 1. Dále pak využité databáze jsou SQL Server 2012 a Oracle 11G R2 x64 Enterprise Edition.

4.2 Microsoft SQL Server

4.2.1 Fyzické návrhy

V této sekci budou popsány jednotlivé návrhy typů tabulek, indexů a také zdůvodnění volby těchto typů. U každého návrhu je využit tzv. execution plan [8], jedná se o plán vykonání dotazu vybraný optimalizátorem. Díky tomuto plánu můžeme odladit problematické dotazy. A to buď vytvořením indexů, nebo změnou typů tabulek. V plánu vidíme veškeré provedené operace SŘBD: průchod tabulkou, přístup k indexu, třídění, spojení atd. V mém případě jsem se zaměřoval na počet logických a diskových přístupů. Vysoký počet těchto přístupů představuje operace Table Scan tzv. průchod celou tabulkou.

[6] http://msdn.microsoft.com/en-us/library/windows/desktop/ms722656%28v=vs.85%29.aspx
[7] http://www.microsoft.com/en-us/download/details.aspx?id=17718
[8] https://www.simple-talk.com/sql/performance/execution-plan-basics/



Obrázek 6: Diagram komponent

4.2.1.1 Fyzický návrh před optimalizací

Tento návrh bude sloužit jako tzv. *baseline*, tedy základní návrh, který bude porovnáván s ostatními návrhy nad tímto SŘBD. V tomto návrhu se budou vyskytovat defaultní tabulky, konkrétně tedy typu halda, které se vytvoří po zadání příkazu *CREATE TABLE*. A také zde nebudou vytvořeny žádné indexy. Počet a velikost záznamů jednotlivých tabulek, typy tabulek a indexů můžeme nalézt v tabulce 22, která se nachází v příloze A.

4.2.1.2 Fyzický návrh č. 1

V tomto návrhu budou veškeré tabulky typu halda a na některé z těchto tabulek bude vytvořen nonclustered index. Index bude vytvářen s ohledem na co nejvyšší efektivitu dotazování. Ovšem musíme brát v potaz, že vytvořením indexu zhoršíme operace týkající se aktualizace databáze. Indexy jsou tedy vytvářeny tak, aby způsobovaly co nejmenší zhoršení těchto operací. V tomto návrhu budou uvedeny pouze takové tabulky, které se liší od těch z původního návrhu. Počet a velikost záznamů těchto tabulek, typy tabulek a indexů můžeme nalézt v tabulce 12

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Customer	5000	1824	Heap	Non-Clustered(C_ID, C_TAX_ID)

Pokračování na následující stránce

Tabulka 12 – pokračování z předcházející stránky

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Customer Account	25000	4120	Heap	Non-Clustered(CA_ID, CA_C_ID)
Watch List	5000	544	Heap	Non-Clustered(WL_ID, WL_C_ID)
Broker	50	48	Heap	Non-Clustered(B_ID, B_NAME)
Trade	86400000	19231648	Heap	Non-Clustered(T_ID, T_CA_ID, T_DTS, T_ST_ID)
Trade Request	60	632	Heap	Non-Clustered(TR_T_ID, TR_BID_PRICE, TR_QTY, TR_B_ID, TR_S_SYMB)
Company	2500	1128	Heap	Non-Clustered(CO_ID, CO_IN_ID)
Industry	102	48	Heap	Non-Clustered(IN_ID, IN_NAME)
Sector	12	48	Heap	Non-Clustered(SC_ID, SC_NAME)
Security	3425	1064	Heap	Non-Clustered(S_SYMB, S_CO_ID, S_ISSUE, S_EX_ID)
Trade History	207275690	11914080	Heap	Non-Clustered(TH_T_ID, TH_ST_ID)

Tabulka 12: SQL Server Informace o jednotlivých tabulkách - Návrh č. 1

4.2.1.3 Fyzický návrh č. 2 V tomto návrhu budou tabulky dvojího typu. Tabulky, na nichž se nedotazujeme, budou typu halda. Naproti tomu tabulky na, které se dotazujeme, budou shlukované. Tento typ tabulky nám zaručí rychlé vyhledání záznamů. Ovšem musíme brát v potaz, že tím zhoršíme operace, jako jsou *INSERT*, *UPDATE* nebo *DELETE*.

Shlukovaný index bude použit zejména na primární klíče. Na cizí klíče a potřebné atributy bude využit Non-cluster index. V tomto návrhu budou uvedeny pouze takové tabulky, které se liší od těch z původního návrhu. Počet a velikost záznamů těchto tabulek, typy tabulek a indexů můžeme nalézt v tabulce 13

Název tabulky	**Počet záznamů**	**Velikost [kB]**	**Typ tabulky**	**Typ indexu**
Customer	5000	1752	-	Clustered(C_ID),Non-Clustered(C_ID, C_TAX_ID)
Customer Account	25000	3608	-	Clustered(CA_ID), Non-Clustered(CA_ID, CA_C_ID)
Watch List	5000	416	-	Clustered(WL_ID), Non-Clustered(WL_ID, WL_C_ID)
Broker	50	32	-	Clustered(B_ID), Non-Clustered(B_ID, B_NAME)
Trade	86400000	17751544	-	Clustered(T_ID), Non-Clustered(T_ID, T_CA_ID, T_DTS, T_ST_ID)
Trade Request	60	32	-	Clustered(TR_T_ID), Non-Clustered(TR_T_ID, TR_BID_PRICE, TR_QTY, TR_B_ID, TR_S_SYMB)
Company	2500	912	-	Clustered(CO_ID), Non-Clustered(CO_ID, CO_IN_ID)
Industry	102	32	-	Clustered(IN_ID), Non-Clustered(IN_ID, IN_NAME)
Sector	12	32	-	Clustered(SC_ID), Non-Clustered(SC_ID, SC_NAME)

Pokračování na následující stránce

Tabulka 13 – pokračování z předcházející stránky

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Security	3425	984	-	Clustered(S_SYMB), Non-Clustered(S_SYMB, S_CO_ID, S_ISSUE, S_EX_ID)
Holding summary	248899	39648	Heap	Non-Clustered(HS_CA_ID, HS_S_SYMB, HS_QTY)
Trade History	207300000	17707792	Heap	Non-Clustered(TH_T_ID, TH_ST_ID)

Tabulka 13: SQL Server Informace o jednotlivých tabulkách - Návrh č. 2

4.2.2 Naměřené hodnoty

4.2.2.1 Fyzický návrh před optimalizací Jelikož v tomto návrhu nejsou vytvořeny žádné indexy a všechny tabulky jsou typu halda, nebude zde dosaženo tak dobrých výsledků, jak tomu je u následujících návrhů. Výsledné naměřené hodnoty nalezneme v tabulce 14.

Název transakce	Počet volání	Minimum [s]	Průměr [s]	Maximum [s]	tps
Broker Volume	1500	0	0.025	0.051	39
Customer Position (Fram 1,3)	1500	0	0.016	0.043	62
Customer Position (Fram 1,2)	1500	0.283	1.085	4.033	0.9
Market Watch	1500	0	0.006	0.343	144
Trade Status	1000	0.873	1.583	2.491	0.6
Trade Cleanup	50	7.623	12.395	18.197	0.08

Tabulka 14: SQL Server Naměřené výsledky - Návrh před optimalizací

4.2.2.2 Fyzický návrh č. 1 Jak můžeme vidět, vytvořením indexů se nám navýšil výkon většiny transakcí. Jedná se o navýšení řádově násobku jednotek oproti původnímu návrhu. V tomto případě nemusíme procházet celou tabulku tzv. *TABLE SCAN*, ale pouze

přistupujeme do indexu. Překvapivé výsledky dosáhla transakce Trade Cleanup, u které se očekávalo zhoršení, jelikož se skládá z většího počtu operací aktualizace databáze. U této transakce však převažuje operace čtení oproti zápisu a tím pádem došlo k patrnému zlepšení. Výsledné naměřené hodnoty nalezneme v tabulce 15.

Název transakce	Počet volání	Minimum [s]	Průměr [s]	Maximum [s]	tps
Broker Volume	1500	0	0.008	0.077	117
Customer Position (Fram 1,3)	1500	0	0.001	0.026	568
Customer Position (Fram 1,2)	1500	0.032	0.107	1.173	9
Market Watch	1500	0	0.003	0.031	300
Trade Status	1000	0.162	0.254	0.656	4
Trade Cleanup	50	5.836	6.202	6.416	0.2

Tabulka 15: SQL Server Naměřené výsledky - Návrh č. 1

4.2.2.3 Fyzický návrh č. 2 Jak můžeme vidět, vytvořením indexů se nám navýšil výkon většiny transakcí oproti původnímu návrhu. Dokonce jsme dosáhli lepšího výkonu oproti návrhu číslo 1, kde pouze transakce Trade-Cleanup dosáhla lepšího výkonu. Je to způsobeno tím, že u předešlého návrhu je použita tabulka typu halda, která se pojí s touto transakcí. V tomto návrhu je však u této tabulky použit typ shlukované tabulky. Obdobně jako u předešlého návrhu, za pomocí indexů nemusíme procházet celou tabulku tzv. *TABLE SCAN*, pouze přistupujeme do indexu. Výsledné naměřené hodnoty nalezneme v tabulce 16.

Název transakce	Počet volání	Minimum [s]	Průměr [s]	Maximum [s]	tps
Broker Volume	1500	0	0.004	0.076	202
Customer Position (Fram 1,3)	1500	0	0.001	0.021	805
Customer Position (Fram 1,2)	1500	0.026	0.072	0.156	14
Market Watch	1500	0	0.002	1.047	496
Trade Status	1000	0.161	0.201	0.656	5
Trade Cleanup	50	5.343	6.427	7.638	0.2

Tabulka 16: SQL Server Naměřené výsledky - Návrh č. 2

4.3 Oracle

4.3.1 Fyzické návrhy

V této sekci budou popsány jednotlivé návrhy typů tabulek, indexů a také zdůvodnění volby těchto typů. U každého návrhu je využit tzv. execution plan [9], jedná se o plán vykonání dotazu vybraný optimalizátorem. Díky tomuto plánu můžeme odladit problematické dotazy. A to buď vytvořením indexů, nebo změnou typů tabulek. V plánu vidíme veškeré provedené operace SŘBD: průchod tabulkou, přístup k indexu, třídění, spojení atd. V mém případě jsem se zaměřoval na počet logických a diskových přístupů. Vysoký počet těchto přístupů představuje operace Table Scan tzv. průchod celou tabulkou.

4.3.1.1 Fyzický návrh před optimalizací Tento návrh bude sloužit jako tzv. *baseline*, tedy základní návrh, který bude porovnáván s ostatními návrhy nad tímto SŘBD. V tomto návrhu se budou vyskytovat defaultní tabulky, konkrétně tedy typu halda, které se vytvoří po zadání příkazu *CREATE TABLE*. A také zde nebudou vytvořeny žádné indexy. Počet a velikost záznamů jednotlivých tabulek, typy tabulek a indexů můžeme nalézt v tabulce 23, která se nachází v příloze A.

4.3.1.2 Fyzický návrh č. 1 V tomto návrhu budou veškeré tabulky typu halda a na některé z těchto tabulek bude vytvořen index typu B-Strom. Index bude vytvářen s ohledem na co nejvyšší efektivitu dotazování. Ovšem musíme brát v potaz, že vytvořením indexu zhoršíme operace týkající se aktualizace databáze. Indexy jsou tedy vytvářeny tak, aby způsobovaly co nejmenší zhoršení těchto operací. V tomto návrhu budou uvedeny pouze takové tabulky, které se liší od těch z původního návrhu. Počet a velikost záznamů těchto tabulek, typy tabulek a indexů můžeme nalézt v tabulce 17

Název tabulky	**Počet záznamů**	**Velikost [kB]**	**Typ tabulky**	**Typ indexu**
Customer Account	25000	2960	Heap	B-Tree Index(CA_ID, CA_C_ID)
Watch List	5000	160	Heap	B-Tree Index(WL_ID, WL_C_ID)
Trade	86400000	12136008	Heap	B-Tree Index(T_ID, T_CA_ID, T_DTS, T_ST_ID), Bitmap Index(T_ST_ID)
Trade Request	60	280	Heap	B-Tree Index(TR_T_ID, TR_B_ID, TR_S_SYMB)

Pokračování na následující stránce

[9] https://www.simple-talk.com/sql/performance/execution-plan-basics/

Tabulka 17 – pokračování z předcházející stránky

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Company	2500	880	Heap	B-Tree Index(CO_ID, CO_IN_ID)
Industry	102	40	Heap	B-Tree Index(IN_ID, IN_NAME)
Sector	12	40	Heap	B-Tree Index(SC_ID, SC_NAME)
Security	3425	704	Heap	B-Tree Index(S_SYMB, S_CO_ID, S_ISSUE, S_NUM_OUT, S_EX_ID)
Last Trade	3425	160	Heap	B-Tree Index(LT_S_SYMB, LT_PRICE)
Trade History	207300000	6424536	Heap	B-Tree Index(TH_T_ID, TH_ST_ID)

Tabulka 17: Oracle Informace o jednotlivých tabulkách - Návrh č. 1

4.3.1.3 Fyzický návrh č. 2 V tomto návrhu se kromě klasických indexů typu B-Strom budou vyskytovat několik typů tabulek včetně tabulky typu halda. Jedna z nich bude indexově organizovaná tabulka (IOT), jejíž data jsou uspořádána do struktury indexově binárního stromu, setříděny dle primárního klíče. To nám umožňuje rychlý přístup k primárnímu klíči. U některých těchto typů tabulek je využita klauzule *INCLUDING* a *OVERFLOW*. Kde *INCLUDING* nám zaručuje, že i neklíčový atribut bude uložen v indexovém bloku společně s primárním klíčem. *OVERFLOW* nám zase zaručuje rychlejší dotazování, jelikož primární klíč a atributy v rámci klauzule *INCLUDING* budou uloženy ve stejném segmentu. Což nám zaručí i menší velikost indexu.

Druhým typem tabulky je indexově shlukovaná tabulka, kde nad každou takovouto tabulkou je vytvořen cluster. A dále pak nad tímto clusterem je vytvořen index na shlukující se klíč. Tento typ tabulky je zde využit proto, že se k datům často přistupuje společně.

V rámci tohoto návrhu bylo naplnění databáze dosti pomalé, je to způsobeno použitím těchto typů tabulek. Indexově organizovaná tabulka nám zaručuje uspořádání klíčů, proto je náchylnější k operacím aktualizace. V tomto návrhu budou uvedeny pouze takové tabulky, které se liší od těch z původního návrhu. Počet a velikost záznamů těchto tabulek, typy tabulek a indexů můžeme nalézt v tabulce 18

Název tabulky	**Počet záznamů**	**Velikost [kB]**	**Typ tabulky**	**Typ indexu**
Customer Account	25000	2960	Index Organized Table	B-Tree Index(CA_ID, CA_C_ID)
Customer	5000	1952	Index Organized Table	B-Tree Index(C_ID)
Broker	50	40	Index Organized Table	B-Tree Index(B_ID)
Watch List	5000	160	Index Organized Table	B-Tree Index(WL_ID, WL_C_ID)
Trade	86400000	12599040	Heap	B-Tree Index(T_CA_ID, T_DTS, T_ID), Bitmap Index(T_ST_ID)
Trade Request	60	40	Heap	B-Tree Index(TR_T_ID, TR_B_ID, TR_S_SYMB)
Company	2500	944	Index Cluster Table	B-Tree Index(CO_ID, CO_IN_ID)
Industry	102	944	Index Cluster Table	B-Tree Index(IN_SC_ID, IN_ID)
Sector	12	40	Index Organized Table	B-Tree Index(SC_ID)
Security	3425	704	Heap	B-Tree Index(S_SYMB, S_CO_ID, S_ISSUE, S_NUM_OUT, S_EX_ID)
Last Trade	3425	104	Index Organized Table	B-Tree Index(LT_S_SYMB, LT_PRICE)
Holding Summary	248899	9008	Heap	B-Tree Index(HS_CA_ID, HS_S_SYMB, HS_QTY)
Trade History	207300000	6452136	Heap	B-Tree Index(TH_T_ID, TH_ST_ID)

Tabulka 18: Oracle Informace o jednotlivých tabulkách - Návrh č. 2

4.3.2 Naměřené hodnoty

4.3.2.1 Fyzický návrh před optimalizací Jelikož v tomto návrhu nejsou vytvořeny žádné indexy a všechny tabulky jsou typu halda, nebude zde dosaženo tak dobrých výsledků, jak tomu je u následujících návrhů. Výsledné naměřené hodnoty nalezneme v tabulce 19.

Název transakce	Počet volání	Minimum [s]	Průměr [s]	Maximum [s]	tps
Broker Volume	1500	0.003	0.033	1.667	30
Customer Position (Fram 1,3)	1500	0	0.016	1.667	61
Customer Position (Fram 1,2)	1500	0.241	1.121	1.678	0.9
Market Watch	1500	0	0.025	1.678	39
Trade Status	1000	0.686	1.283	1.684	0.8
Trade Cleanup	50	6.597	9.273	14.213	0.1

Tabulka 19: Oracle Naměřené výsledky - Návrh před optimalizací

4.3.2.2 Fyzický návrh č. 1 Jak můžeme vidět, vytvořením indexů se nám navýšil výkon většiny transakcí. Jedná se o navýšení řádově násobku jednotek. Překvapivých výsledků dosáhla transakce Trade Cleanup, u které se očekávalo zhoršení, jelikož se skládá z většího počtu operací aktualizace databáze. U této transakce však převažuje operace čtení oproti zápisu a tím pádem došlo k patrnému zlepšení. Je zde také využit bitmapový index, který nám rapidně snižuje počet logických přístupů. Jedná se o index [5], který pro každou hodnotu h indexovaného atributu x a jeden záznam obsahuje jeden bit. Bit je nastaven na 1, pokud má záznam hodnotu h v atributu x, jinak je 0. Výsledné naměřené hodnoty nalezneme v tabulce 20.

Název transakce	Počet volání	Minimum [s]	Průměr [s]	Maximum [s]	tps
Broker Volume	1500	0	0.009	0.016	110
Customer Position (Fram 1,3)	1500	0	0.001	1.566	574
Customer Position (Fram 1,2)	1500	0	0.232	1.685	4
Market Watch	1500	0	0.005	1.635	175
Trade Status	1000	0	0.243	5.045	4
Trade Cleanup	50	4.971	5.823	6.172	0.2

Tabulka 20: Oracle Naměřené výsledky - Návrh č. 1

4.3.2.3 Fyzický návrh č. 2 Jak můžeme vidět, u transakcí došlo k patřičnému zlepšení. Je to způsobeno tím, že tyto transakce často pracují s indexově organizovanými tabulkami. Díky klauzuli *INCLUDING* zde odpadá operace *TABLE ACCESS (BY INDEX ROWID)*. Jedná se o přístup k jednomu záznamu (k jedné stránce) v tabulce, který následuje po vyhledání klíče (a *ROWID* záznamu) v indexu. Kromě těchto typů tabulek je zde použita tabulka typu indexově shlukovaná, která je velmi výhodná v operacích spojení. Kdy v případě indexu či IOT musíme provést operaci spojení. Při použití tohoto typu tabulky přistupujeme pouze do clusteru tzv. *TABLE ACCESS (CLUSTER)*. Výsledné naměřené hodnoty nalezneme v tabulce 21.

Název transakce	Počet volání	Minimum [s]	Průměr [s]	Maximum [s]	tps
Broker Volume	1500	0.002	0.004	1.678	241
Customer Position (Fram 1,3)	1500	0	0.001	1.666	585
Customer Position (Fram 1,2)	1500	0	0.157	1.669	6
Market Watch	1500	0	0.003	0.018	273
Trade Status	1000	0	0.079	1.667	13
Trade Cleanup	50	4.873	6.247	6.715	0.2

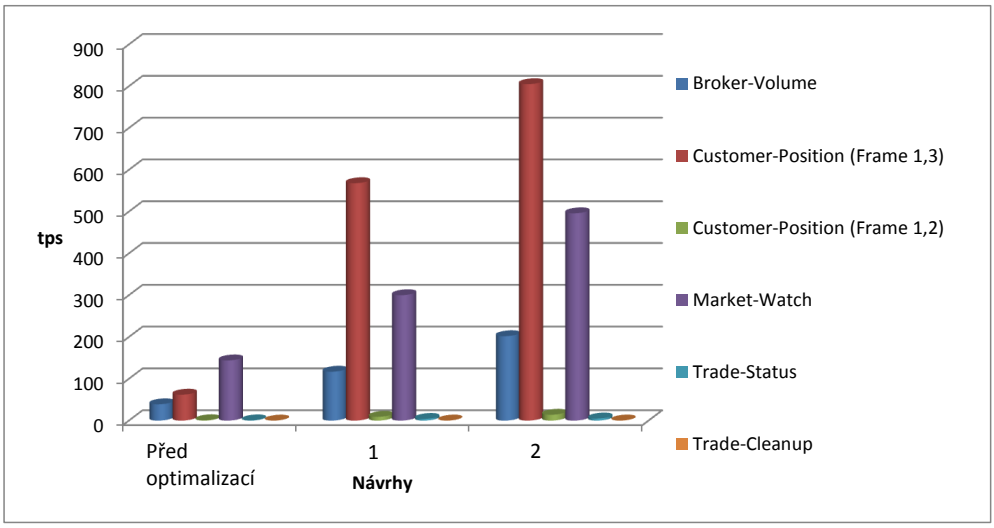
Tabulka 21: Oracle Naměřené výsledky - Návrh č. 2

4.4 Srovnání naměřených hodnot

Nejprve se zaměříme na výsledné naměřené hodnoty SQL Serveru. Můžeme vidět, že oproti původnímu návrhu se nám výsledky rapidně zlepšily. Je to způsobeno použitím různých typů indexů. U návrhu číslo jedna jsou použity indexy pouze typu non-clustered, čímž se nám zvýší počet transakcí za vteřinu (tps) oproti původnímu návrhu. U návrhu číslo dvě je toto zlepšení ještě větší, jelikož je zde také použit index typu clustered. Výsledné naměřené hodnoty můžeme vidět na obrázku 7. Jsou zde zobrazeny výsledky jednotlivých transakcí daných návrhů v tps.

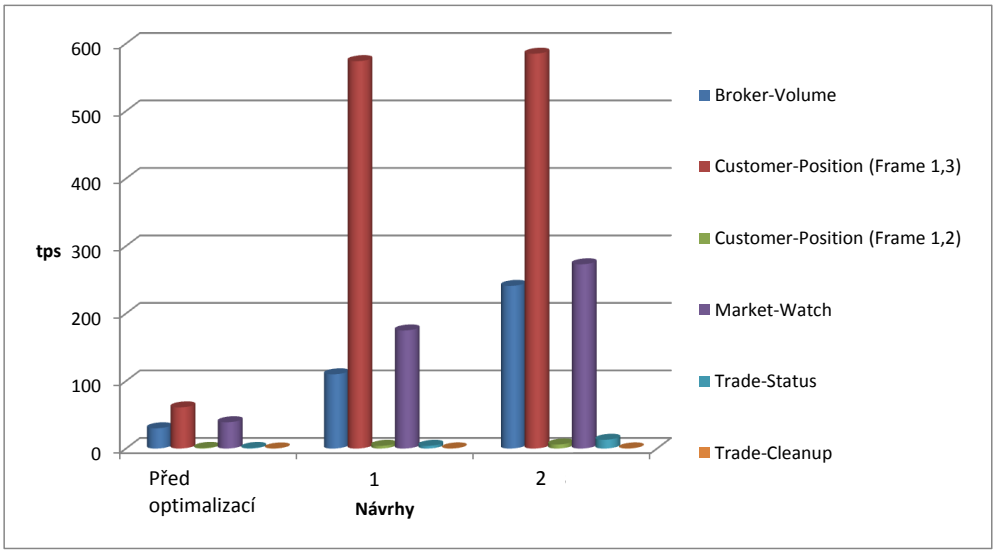
Pokud se zaměříme na naměřené výsledky v rámci Oracle, vidíme zde rapidní zlepšení výkonu oproti původnímu návrhu. V návrhu číslo jedna jsou použity indexy typu B-Strom, což nám navyšuje počet transakcí za vteřinu (tps). Tabulky v tomto návrhy jsou ponechány dle původního návrhu, tedy typu halda. Ve druhém návrhu k těmto indexům přibyly typy tabulek, jako je indexově organizovaná tabulka a indexově shlukovaná tabulka. Tento typ tabulek nám zajišťuje vyšší výkon při dotazování. Kde indexově shlukovaná tabulka nám eliminuje operaci spojení. Naproti tomu indexově organizovaná tabulka nám zase poskytuje lepší míru dotazování primárního klíče s možností použití klauzule *INCLUDING*. Výsledné naměřené hodnoty můžeme vidět na obrázku 8. Jsou zde zobrazeny výsledky jednotlivých transakcí daných návrhů v tps.

Pokud srovnáme obě SŘBD, tak u většiny transakcí dosahuje lepšího výkonu SQL Server oproti Oracle. Oracle pouze v některých transakcích výkonově předbíhá SQL

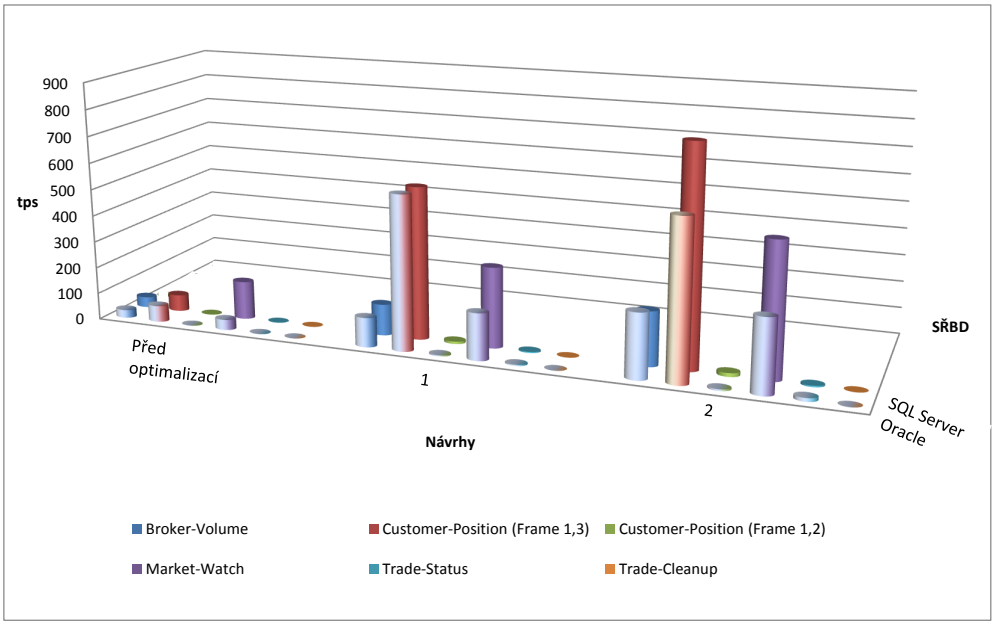


Obrázek 7: MS SQL Server srovnání naměřených hodnot

Server, konkrétně v transakcích Broker-Volume a Trade-Status, kde za zvýšením výkonu stojí indexově organizované tabulky. Srovnání těchto SŘBD můžeme vidět na obrázku 9.



Obrázek 8: Oracle srovnání naměřených hodnot



Obrázek 9: Srovnání naměřených hodnot SQL Server a Oracle

5 Závěr

V této bakalářské práci jsme si za pomocí měření ověřili výkon prováděných dotazů a uložených procedur. K tomuto měření byla vybrána široká škála transakcí, které byly následně změřeny jak na straně SQL Serveru tak i Oracle. V rámci tohoto měření vzniklo několik databázových návrhů s různými typy tabulek a indexů. Každý z těchto návrhů byl navržen tak, aby dotazy v těchto testovaných transakcích byly co nejvíce optimalizovány. Ke každému SŘBD existují tři návrhy včetně původního. Naměřené výsledky jednotlivých transakcí nalezneme v patřičných grafech. Tyto grafy srovnávají výkon jednotlivých SŘBD a také i SŘBD mezi sebou. Z naměřených výsledků vyplývá, že SQL Server v tomto případě disponuje větším výkonem něž Oracle. Je to způsobeno propracovanou strukturou tabulek a indexů tohoto SŘBD. Tím nechci říci, že u Oracle tomu tak není, ale zde už člověk potřebuje široké znalosti týkající se fyzického návrhu. Ovšem toto je pouze můj názor.

Osobně musím říci, že tato práce byla pro mě velkým přínosem. Během vypracování jsem narazil na několik zajímavostí v rámci obou SŘBD. Nyní již chápu některé souvislosti, které mi dříve unikaly. Také jsem získal hluboké znalosti týkající se T-SQL, PL-SQL a fyzického návrhu. A pokud jde o další vývoj projektu, tak by rád tento projekt rozšířil o porovnání administrace týkající se těchto SŘBD, jelikož mě toto téma ve škole dosti zaujalo. A musím říci, že tímto směrem bych se chtěl v budoucnu zabývat i nadále.

Petr Stodůlka

6 Reference

[1] T. Kyte, *Expert Oracle Database Architecture,* New York: Springer-Verlag, 2005.

[2] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford, *Patterns of Enterprise Application Architecture,* New York: Addison Wesley Pub. Co., 2002.

[3] Transaction Processing Performance Council, *TPC-E BENCHMARK,* Presidio of San Francisco, 2010.

[4] Transaction Processing Performance Council, *TPC-E BENCHMARK Overview,* Presidio of San Francisco, 2007.

[5] M. Krátký, R. Bača, *Databázové systémy*, VŠB Technická univerzita Ostrava, 2012.

[6] Microsoft SQL Server msdn, *Heaps (Tables without Clustered Indexes)*, Dostupné na http://msdn.microsoft.com/en-us/library/hh213609.aspx.

[7] Microsoft SQL Server msdn, *Heap Structures,* Dostupné na http://msdn.microsoft.com/cs-cz/library/ms188270%28v=sql.105%29.aspx.

[8] Microsoft SQL Server msdn, *Clustered Index Structures,* Dostupné na http://technet.microsoft.com/en-us/library/ms177443.aspx.

[9] Microsoft SQL Server msdn, *Nonclustered Index Structures,* Dostupné na http://technet.microsoft.com/en-us/library/ms177484.aspx.

[10] Oracle Database Concepts 11g Release 2 (11.2), *Index Organized Tables manual,* Dostupné na http://docs.oracle.com/cd/B28359_01/server.111/b28310/tables012.htm.

[11] Oracle Database Concepts 11g Release 2 (11.2), *Indexes and IOTs manual,* Dostupné na http://docs.oracle.com/cd/E14072_01/server.112/e10713/indexiot.htm.

[12] Transaction Processing Performance Council, *TPC-E Benchmark,* Dostupné na http://www.tpc.org/tpce/default.asp.

A Tabulky k jednotlivým návrhům

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Account Permission	35567	4624	Heap	Non-Clustered(AP_TAX_ID, AP_CA_ID)
Customer	5000	1552	Heap	Non-Clustered(C_ID)
Customer Account	25000	3216	Heap	Non-Clustered(CA_ID)
Customer Taxrate	10000	592	Heap	Non-Clustered(CX_TX_ID, CX_C_ID)
Holding	4400000	419800	Heap	Non-Clustered(H_T_ID)
Holding History	115700000	10322688	Heap	Non-Clustered(HH_H_T_ID, HH_T_ID)
Holding Summary	248899	18912	Heap	Non-Clustered(HS_CA_ID, HS_S_SYMB)
Watch Item	500717	38288	Heap	Non-Clustered(WI_WL_ID, WI_S_SYMB)
Watch List	5000	336	Heap	Non-Clustered(WL_ID)
Broker	50	32	Heap	Non-Clustered(B_ID)
Cash Transaction	79400000	12604032	Heap	Non-Clustered(CT_T_ID)
Charge	15	32	Heap	Non-Clustered(CH_C_TIER, CH_TT_ID)
Commission Rate	240	40	Heap	Non-Clustered(CR_C_TIER, CR_FROM_QTY, CR_TT_ID, CR_EX_ID)
Settlement	86400000	8460144	Heap	Non-Clustered(SE_T_ID)
Trade	86400000	15042688	Heap	Non-Clustered(T_ID)

Pokračování na následující stránce

Tabulka 22 – pokračování z předcházející stránky

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Trade History	207296000	11905304	Heap	Non-Clustered(TH_T_ID, TH_ST_ID)
Trade Request	0	32	Heap	Non-Clustered(TR_T_ID)
Trade Type	5	32	Heap	Non-Clustered(TT_ID)
Company	2500	976	Heap	Non-Clustered(CO_ID)
Company Competitor	7500	592	Heap	Non-Clustered(CP_IN_ID, CP_CO_ID, CP_COMP_CO_ID)
Daily Market	100000	9296	Heap	Non-Clustered(DM_DATE, DM_S_SYMB)
Exchange	4	32	Heap	Non-Clustered(EX_ID)
Financial	50000	8080	Heap	Non-Clustered(FI_YEAR, FI_QTR, FI_CO_ID)
Industry	102	32	Heap	Non-Clustered(IN_ID)
Last Trade	3425	408	Heap	Non-Clustered(LT_S_SYMB)
News Item	5000	2384	Heap	Non-Clustered(NI_ID)
News Xref	5000	608	Heap	Non-Clustered(NX_NI_ID, NX_CO_ID)
Sector	12	32	Heap	Non-Clustered(SC_ID)
Security	3425	848	Heap	Non-Clustered(S_SYMB)
Address	7504	2320	Heap	Non-Clustered(AD_ID)
Status Type	5	32	Heap	Non-Clustered(ST_ID)
Taxrate	320	48	Heap	Non-Clustered(TX_ID)

Pokračování na následující stránce

Tabulka 22 – pokračování z předcházející stránky

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Zip Code	14741	3152	Heap	Non-Clustered(ZC_CODE)

Tabulka 22: SQL Server Informace o jednotlivých tabulkách - Návrh před optimalizací

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Account Permission	35567	3968	Heap	B-Tree Index(AP_TAX_ID, AP_CA_ID)
Customer	5000	1952	Heap	B-Tree Index(C_ID)
Customer Account	25000	2960	Heap	B-Tree Index(CA_ID)
Customer Taxrate	10000	224	Heap	B-Tree Index(CX_TX_ID, CX_C_ID)
Holding	4400000	268456	Heap	B-Tree Index(H_T_ID)
Holding History	115700000	3989920	Heap	B-Tree Index(HH_H_T_ID, HH_T_ID)
Holding Summary	248899	9008	Heap	B-Tree Index(HS_CA_ID, HS_S_SYMB)
Watch Item	500717	16064	Heap	B-Tree Index(WI_WL_ID, WI_S_SYMB)
Watch List	5000	160	Heap	B-Tree Index(WL_ID)
Broker	50	40	Heap	B-Tree Index(B_ID)
Cash Transaction	79400000	11545280	Heap	B-Tree Index(CT_T_ID)
Charge	15	40	Heap	B-Tree Index(CH_C_TIER, CH_TT_ID)

Pokračování na následující stránce

Tabulka 23 – pokračování z předcházející stránky

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu
Commission Rate	240	40	Heap	B-Tree Index(CR_C_TIER, CR_FROM_QTY, CR_TT_ID, CR_EX_ID)
Settlement	86400000	6734600	Heap	B-Tree Index(SE_T_ID)
Trade	86400000	12136008	Heap	B-Tree Index(T_ID)
Trade History	207296000	6439768	Heap	B-Tree Index(TH_T_ID, TH_ST_ID)
Trade Request	0	40	Heap	B-Tree Index(TR_T_ID)
Trade Type	5	40	Heap	B-Tree Index(TT_ID)
Company	2500	880	Heap	B-Tree Index(CO_ID)
Company Competitor	7500	224	Heap	B-Tree Index(CP_IN_ID, CP_CO_ID, CP_COMP_CO_ID)
Daily Market	100000	4976	Heap	B-Tree Index(DM_DATE, DM_S_SYMB)
Exchange	4	40	Heap	B-Tree Index(EX_ID)
Financial	50000	5984	Heap	B-Tree Index(FI_YEAR, FI_QTR, FI_CO_ID)
Industry	102	40	Heap	B-Tree Index(IN_ID)
Last Trade	3425	160	Heap	B-Tree Index(LT_S_SYMB)
News Item	5000	2960	Heap	B-Tree Index(NI_ID)
News Xref	5000	131	Heap	B-Tree Index(NX_NI_ID, NX_CO_ID)
Sector	12	40	Heap	B-Tree Index(SC_ID)
Security	3425	704	Heap	B-Tree Index(S_SYMB)
Address	7504	1952	Heap	B-Tree Index(AD_ID)
Status Type	5	40	Heap	B-Tree Index(ST_ID)
Taxrate	320	40	Heap	B-Tree Index(TX_ID)
Zip Code	14741	2960	Heap	B-Tree Index(ZC_CODE)

Pokračování na následující stránce

Tabulka 23 – pokračování z předcházející stránky

Název tabulky	Počet záznamů	Velikost [kB]	Typ tabulky	Typ indexu

Tabulka 23: Oracle Informace o jednotlivých tabulkách - Návrh před optimalizací

B Pseudokódy transakcí

```
{
    start transaction
    // Should return 0 to 40 rows
    select
        broker_name[] = B_NAME,
        volume[] = sum(TR_QTY * TR_BID_PRICE)
    from
        TRADE_REQUEST,
        SECTOR,
        INDUSTRY
        COMPANY,
        BROKER,
        SECURITY
    where
        TR_B_ID = B_ID and
        TR_S_SYMB = S_SYMB and
        S_CO_ID = CO_ID and
        CO_IN_ID = IN_ID and
        SC_ID = IN_SC_ID and
        B_NAME in (broker_list) and
        SC_NAME = sector_name
    group by
        B_NAME
    order by
        2 DESC

    // row_count will frequently be zero near the start of a Test Run when
    // TRADE_REQUEST table is mostly empty.
    list_len = row_count
    commit transaction
}
```

Výpis 1: Broker-Volume pseudokód

```
{
    start transaction
    if (cust_id == null_cust_id ) then {
        select
            cust_id = C_ID
        from
            CUSTOMER
        where
            C_TAX_ID = tax_id
    }

    select
        c_st_id = C_ST_ID,
        c_l_name = C_L_NAME,
        c_f_name = C_F_NAME,
        c_m_name = C_M_NAME,
        c_gndr = C_GNDR,
```

```
        c_tier  = C_TIER,
        c_dob = C_DOB,
        c_ad_id  = C_AD_ID,
        c_ctry_1  = C_CTRY_1,
        c_area_1 = C_AREA_1,
        c_local_1  = C_LOCAL_1,
        c_ext_1  = C_EXT_1,
        c_ctry_2  = C_CTRY_2,
        c_area_2 = C_AREA_2,
        c_local_2  = C_LOCAL_2,
        c_ext_2  = C_EXT_2,
        c_ctry_3  = C_CTRY_3,
        c_area_3 = C_AREA_3,
        c_local_3  = C_LOCAL_3,
        c_ext_3  = C_EXT_3,
        c_email_1 = C_EMAIL_1,
        c_email_2 = C_EMAIL_2
    from
        CUSTOMER
    where
        C_ID = cust_id

    // Should return 1 to max_acct_len (10).
    select first max_acct_len rows
        acct_id [] = CA_ID,
        cash_bal[] = CA_BAL,
        assets_total [] = ifnull ((sum(HS_QTY * LT_PRICE)),0)
    from
        CUSTOMER_ACCOUNT left outer join
        HOLDING_SUMMARY on HS_CA_ID = CA_ID,
        LAST_TRADE
    where
        CA_C_ID = cust_id and
        LT_S_SYMB = HS_S_SYMB
    group by
        CA_ID, CA_BAL
    order by
        3 asc

    acct_len = row_count
}
```

Výpis 2: Customer-Position Frame 1 pseudokód

```
{
    // Should return 10 to 30 rows.
    select first 30 rows
        trade_id [] = T_ID,
        symbol[] = T_S_SYMB,
        qty [] = T_QTY,
        trade_status [] = ST_NAME,
        hist_dts [] = TH_DTS
    from
        (select first 10 rows
```

```
            T_ID as ID
        from
            TRADE
        where
            T_CA_ID = acct_id
        order by T_DTS desc) as T,
        TRADE,
        TRADE_HISTORY,
        STATUS_TYPE
    where
        T_ID = ID and
        TH_T_ID = T_ID and
        ST_ID = TH_ST_ID
    order by
        TH_DTS desc

        hist_len  = row_count

    commit transaction
}
```

Výpis 3: Customer-Position Frame 2 pseudokód

```
{

    commit transaction

}
```

Výpis 4: Customer-Position Frame 3 pseudokód

```
{
    start  transaction
    if  (cust_id  != 0)  then {
        declare  stock_list  cursor for
            select
                WI_S_SYMB
            from
                WATCH_ITEM,
                WATCH_LIST
            where
                WI_WL_ID = WL_ID and
                WL_C_ID = cust_id
    } else if  (industry_name != "")  then {
        declare  stock_list  cursor for
            select
                S_SYMB
            from
                INDUSTRY,
                COMPANY,
                SECURITY
            where
                IN_NAME = industry_name and
                CO_IN_ID = IN_ID and
```

```
            CO_ID between (starting_co_id and ending_co_id) and
            S_CO_ID = CO_ID
} else if (acct_id != 0) then {
    declare stock_list cursor for
        select
            HS_S_SYMB
        from
            HOLDING_SUMMARY
        where
            HS_CA_ID = acct_id
}
old_mkt_cap = 0.0
new_mkt_cap = 0.0
pct_change = 0.0
open stock_list
do until ( stock_list .end_of_cursor) {
    fetch from
         stock_list cursor
    into
        symbol

    select
        new_price = LT_PRICE
    from
        LAST_TRADE
    where
        LT_S_SYMB = symbol

    select
        s_num_out = S_NUM_OUT
    from
        SECURITY
    where
        S_SYMB = symbol

    // Closing price for this security on the chosen day.
    select
        old_price = DM_CLOSE
    from
        DAILY_MARKET
    where
        DM_S_SYMB = symbol and
        DM_DATE = start_date

    old_mkt_cap += s_num_out * old_price
    new_mkt_cap += s_num_out * new_price
}
if (old_mkt_cap != 0) then
{
    // value of 0.00 for pct_change is valid
    pct_change = 100 * (new_mkt_cap / old_mkt_cap - 1)
}
else
{
```

```
        // no rows found, this can happen rarely when an account has no holdings
        pct_change = 0.0
    }
    close stock_list
    commit transaction
}
```

Výpis 5: Market-Watch pseudokód

```
{
    start transaction
    // Only want 50 rows, the 50 most recent trades for this customer account
    select first 50 row
        trade_id [] = T_ID,
        trade_dts [] = T_DTS,
        status_name[] = ST_NAME,
        type_name[] = TT_NAME,
        symbol[] = T_S_SYMB,
        trade_qty [] = T_QTY,
        exec_name[] = T_EXEC_NAME,
        charge[] = T_CHRG,
        s_name[] = S_NAME,
        ex_name[] = EX_NAME
    from
        TRADE,
        STATUS_TYPE,
        TRADE_TYPE,
        SECURITY,
        EXCHANGE
    where
        T_CA_ID = acct_id and
        ST_ID = T_ST_ID and
        TT_ID = T_TT_ID and
        S_SYMB = T_S_SYMB and
        EX_ID = S_EX_ID
    order by
        T_DTS desc

    num_found = row_count


    select
        cust_l_name = C_L_NAME,
        cust_f_name = C_F_NAME,
        broker_name = B_NAME
    from
        CUSTOMER_ACCOUNT,
        CUSTOMER,
        BROKER
    where
        CA_ID = acct_id and
        C_ID = CA_C_ID and
        B_ID = CA_B_ID
```

```
    commit transaction
}
```

Výpis 6: Trade-Status pseudokód

```
{
    start transaction

    Declare t_id TRADE_T
    Declare tr_t_id TRADE_T
    Declare now_dts DATETIME

    /* Find pending trades from TRADE_REQUEST */

    declare pending_list for
    select
        TR_T_ID
    from
        TRADE_REQUEST
    order by
        TR_T_ID

    open pending_list

    /* Insert a submitted followed by canceled record into TRADE_HISTORY, mark the trade
    canceled and delete the pending trade */

    do until (end_of_pending_list) {
    fetch from
        pending_list
    into
        tr_t_id

    get_current_dts ( now_dts )

    insert into
        TRADE_HISTORY (
            TH_T_ID, TH_DTS, TH_ST_ID
        )
    values (
        tr_t_id , // TH_T_ID
        now_dts, // TH_DTS
        st_submitted_id // TH_ST_ID
    )

    update
        TRADE
        set
            T_ST_ID = st_canceled_id,
            T_DTS = now_dts
        where
            T_ID = tr_t_id

    insert into
```

```
    TRADE_HISTORY (
        TH_T_ID, TH_DTS, TH_ST_ID
    )
values (
     tr_t_id ,  // TH_T_ID
    now_dts, // TH_DTS
    st_canceled_id  // TH_ST_ID
)

} //end of pending_list

/* Remove all pending trades */
delete
from
    TRADE_REQUEST

/* Find submitted trades, change the status to canceled and insert a canceled record
into TRADE_HISTORY*/
declare submit_list for
select
    T_ID
from
    TRADE
where
    T_ID >= trade_id and
    T_ST_ID = st_submitted_id

open submit_list

do until ( end_of_submit_list ) {
    fetch from
        submit_list
    into
        t_id

    get_current_dts ( now_dts )

    /* Mark the trade as canceled, and record the time */
    update
        TRADE
    set
        T_ST_ID = st_canceled_id
        T_DTS = now_dts
    where
        T_ID = t_id

    insert into
        TRADE_HISTORY (
            TH_T_ID, TH_DTS, TH_ST_ID
        )
    values (
        t_id ,  // TH_T_ID
        now_dts, // TH_DTS
        st_canceled_id  // TH_ST_ID
```

```
        )

    } //end of submit_list

    commit transaction
}
```

Výpis 7: Trade-Cleanup pseudokód

C Manuál k programům

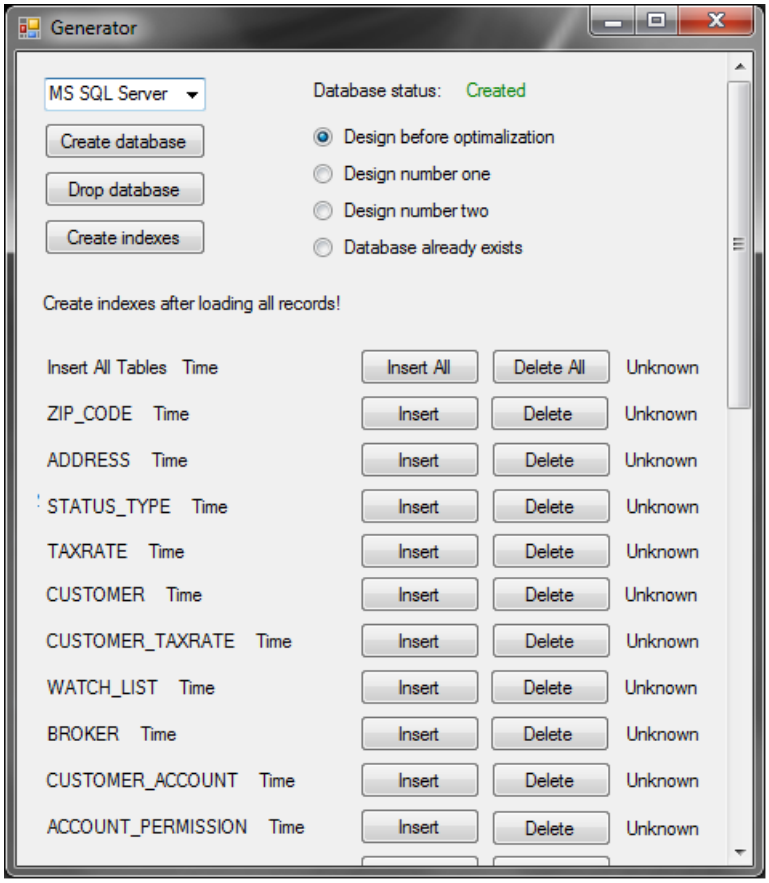
Toto je manuál k seznámení se s dvěma vytvořenými programy, které jsou součástí této bakalářské práce. Jeden z nich slouží pro vytváření a mazání databáze. Dále pak také ke vkládání záznamů. Druhý z těchto programů obsluhuje jednotlivé transakce. Pokud budeme chtít změnit connection string, otevřeme oba konfigurační soubory, tedy Generator.exe.config a Transactions.exe.config a u obou změníme patřičné údaje.

C.1 Program Generator

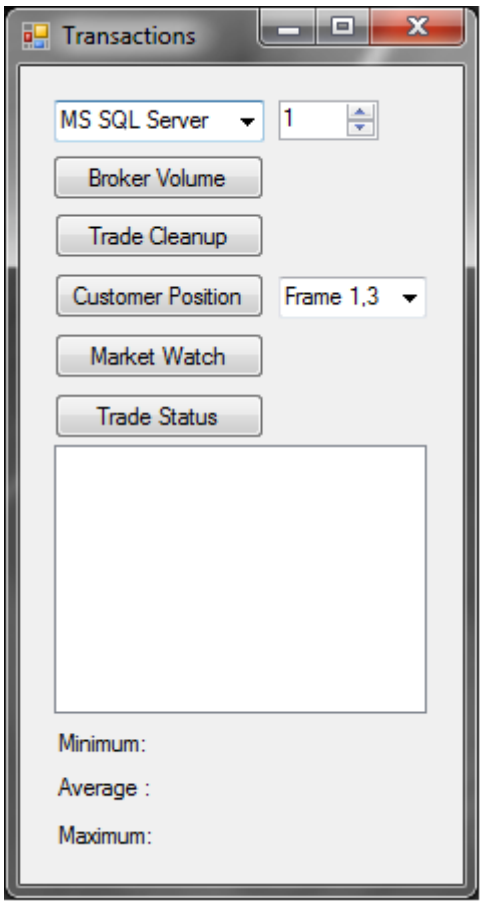
Před samotným spuštěním tohoto programu je nutné zkontrolovat obsah souboru Generator.exe.config, pro určení správných cest k adresářům s daty. Tento program výhradně slouží k vytvoření a smazání databáze, jak na straně SQL Serveru tak i Oracle. Při vytváření databáze si můžeme vybrat z jednotlivých návrhů popsaných v dokumentaci. Nad takto vytvořenou databází je možno vytvořit indexy, ale je zde doporučení vytvoření indexů až po samotném vložení záznamů, což je také součásti tohoto programu. Záznamy můžeme vkládat po jednotlivých tabulkách nebo najednou do všech tabulek. Pokud se rozhodneme vkládat po jednotlivých tabulkách, musíme být obezřetní, jelikož v rámci tohoto programu nejsou ošetřeny reference mezi tabulkami. V tomto případě tabulky vkládáme postupně, jak jdou za sebou. U každé z tabulek se nachází údaj reprezentující časový úsek zabraný vkládáním záznamů v rámci tabulky. Také se zde nachází údaj sdělující, zda byly záznamy v této tabulce vloženy či smazány. Vkládat záznamy můžeme pouze v okamžiku, kdy máme vytvořenou databázi. Buď to si ji vytvoříme, nebo zaškrtneme políčko *Database already exists*. Tento program můžeme vidět na obrázku 10.

C.2 Program Transactions

Před spuštěním tohoto programu musíme mít vytvořenou a naplněnou databázi. Tento program nám obsluhuje jednotlivé transakce a to opět jak na straně SQL Serveru tak i Oracle. Program funguje takovým způsobem, že si vybereme danou transakci, kterou chceme spustit. Nastavíme si počet spuštění této transakce a příslušným tlačítkem spustíme. Po doběhnutí této skupiny transakcí se nám zobrazí časové výsledky. Patřičné výsledky jednotlivých transakcí jsou zobrazeny v daném okně. V rámci výsledků jsou zde zobrazeny také časy s nejnižší, nejvyšší a průměrnou časovou hodnotou. U jedné z transakcí je možnost vybrat jednu ze dvou sekcí, jelikož se tato daná transakce skládá z více těchto sekcí. Tento program můžeme vidět na obrázku 11.



Obrázek 10: Program Generator



Obrázek 11: Program Transactions