Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Porovnání databázových systémů MySQL a PostgreSQL

Sami Salama

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Sami Salama

Informatika

Název práce

Porovnání databázových systémů MySQL a PostgreSQL

Název anglicky

Comparison of MySQL and PostgreSQL database systems

Cíle práce

Hlavním cílem této bakalářské práce bude analýza a srovnání databázových systémů MySQL a PostgreSQL na zvoleném operačním systému. Dílčími cíli bude
- stanovení teoretických východisek a zhodnocení analyzovaných databázových systémů pomocí vícekriteriální analýzy variant,
- implementace databázového systému na konkrétním databázovém modelu.

Metodika

Metodika zpracování bakalářské práce bude vycházet ze studia a analýzy dostupných informačních zdrojů a současných řešení v oblasti provozu a metodik měření databázových systémů. Výběrem vhodných hodnotících kritérií a poté porovnáním daných databázových systémů podle zvolených kritérií bude identifikován optimální databázový systém, který bude použit pro implementaci na konkrétním databázovém modelu.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

MySQL, PostgreSQL, databáze, databázový systém, operační systém, analýza, srovnání, implementace

---

Doporučené zdroje informací

BALLIN, Derek J. a kolektiv. MySQL profesionálně: optimalizace pro vysoký výkon. Vyd. 1. Brno: Zoner Press, 2009, 712 s. Encyklopedie webdesignera. ISBN 978-80-7413-035-9.

MASLAKOWSKI, Mark. Naučte se MySQL za 21 dní. Praha: Computer Press, 2001. Databáze. Pro každého uživatele. ISBN 80-7226-448-6.

MySQL Documentation: MySQL Reference Manuals. ORACLE. [online]. [cit. 23.04.2017]. Dostupné z: https://dev.mysql.com/doc/

PostgreSQL: Documentation. PostgreSQL: The world's most advanced open source database [online]. [cit. 23.04.2017]. Dostupné z: https://www.postgresql.org/docs/

Předběžný termín obhajoby

2017/18 LS – PEF

Vedoucí práce

Ing. Václav Lohr, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 31. 10. 2017

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 1. 11. 2017

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 12. 03. 2018

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Porovnání databázových systémů MySQL a PostgreSQL" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2017 Sami Salama

Poděkování

Rád bych touto cestou poděkoval Ing. Václavu Lohrovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Porovnání databázových systémů MySQL a PostgreSQL

Abstrakt

Hlavním zaměřením práce je porovnání dvou open-source databázových systémů MySQL a PostgreSQL. Teoretická část nejdříve definuje pojem databáze a popisuje jazyk SQL, se kterým oba systémy pracují. Dále jsou v této části blíže popsány oba systémy z hlediska historie vývoje, licenčních podmínek používání, storage engines (tj. úložné databázové stroje) a dostupných odvozenin systémů.

V praktické části je specifikované prostředí měření a je provedena instalace obou systémů. Dalším krokem je provedení měření pomocí nástroje HammerDB. Součástí tohoto kroku je detailnější popsání postupu měření, tzn. nastavení knihoven, vytvoření testovacích dat, předběžné otestování, měření a porovnání výsledků měření. Posledním krokem této části je výběr optimálního databázového systému pomocí vícekriteriální analýzy variant.

Nakonec jsou zhodnoceny zjištěné výsledky a je sepsán závěr.

Klíčová slova: MySQL, PostgreSQL, databáze, databázový systém, operační systém, analýza, srovnání, implementace, HammerDB

Comparison of MySQL and PostgreSQL database systems

Abstract

The major objective of this study is comparison of two open-source database systems MySQL and PostgreSQL. The theoretical part at first defines the concept of database and describes the SQL language used by both systems. Furthermore, this part describes both systems in terms of development history, licensing conditions, storage engines, and available system derivations.

The practical part specifies the measurement environment and installation of both systems is performed. The next step is to perform measurements using HammerDB. This step is a more detailed description of the measurement procedure, i.e. setting up libraries, creating test data, pre-testing, measuring, and comparing measurement results. The final step of this section is to select an optimal database system by multiple-criteria decision analysis.

Finally, the results are evaluated, and the conclusion is drawn.

Keywords: MySQL, PostgreSQL, database, database system, operating system, analysis, comparison, implementation, HammerDB

Obsah

1 Úvod

V dnešní době je pro většinu aplikací nedílnou součástí databáze pro uchovávání dat. Aby s těmito daty mohl uživatel pracovat, potřebuje dotazový systém, který tvoří rozhraní mezi uživatelem a databází obsahující data.

Na oblíbenosti mezi uživateli, začínajícími i zkušenými vývojáři, získávají open source softwary, které se díky své politice otevřeného kódu, stávají zdrojem inovací a vědomostí. Při spojením termínů „open source“ a „databázový systém“, se spoustu lidem ze světa informatiky určitě vybaví MySQL a PostgreSQL.

Data lze ukládat i jinými způsoby něž do databáze, například textové soubory, excel a klidně i vytisknout na papír. Ale práce s takto uloženými daty je často složitější a omezená. Lepším přístupem jsou právě databázové systémy, které za nás řeší spoustu záležitostí, například zabezpečení, standardizaci, optimalizaci, integritu, konzistenci dat apod.

Ke komunikaci s databázovým systémem (z uživatelského pohledu přímo s databází) slouží jazyk SQL (anglicky Structured Query Language), v překladu strukturovaný dotazovací jazyk. Tento jazyk se snaží být srozumitelný pro člověka, proto má podobu anglických vět. Napsáním a spuštěním dotazu nám databázový systém vypíše požadovaný výsledek.

Základem pro MySQL i PostgreSQL je tzv. relační databázový model. Relační databázový model lze pochopit jako databázi, která ukládá svá data do tabulek, které jsou navzájem propojeny – mezi tabulkami jsou relace.

Tato práce je rozdělena na dvě části. První část si klade za cíl teoretické seznámení s pojmem relační databáze, jazykem SQL a oběma databázovými systémy. Druhá část, praktická, si klade za cíl porovnání obou systémů pomocí měřícího nástroje a výběr optimálního systému na základě získaných výsledků měření a studia dostupných informačních zdrojů.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem této bakalářské práce bude analýza a srovnání databázových systémů MySQL a PostgreSQL na zvoleném operačním systému.

Dílčími cíli bude:

- stanovení teoretických východisek a zhodnocení analyzovaných databázových systémů pomocí vícekriteriální analýzy variant,
- implementace databázového systému na konkrétním databázovém modelu.

2.2 Metodika

Metodika zpracování bakalářské práce bude vycházet ze studia a analýzy dostupných informačních zdrojů a současných řešení v oblasti provozu a metodik měření databázových systémů.

Výběrem vhodných hodnotících kritérií a poté porovnáním daných databázových systémů podle zvolených kritérií bude identifikován optimální databázový systém, který bude použit pro implementaci na konkrétním databázovém modelu.

3 Teoretická část

3.1 Jazyk SQL

Pro pochopení, k čemu slouží jazyk SQL, je si nedříve zapotřebí ujasnit, co je to databáze. Volně dostupný online slovník výpočetní techniky definuje databázi jako „*Jedna nebo více velkých strukturovaných sad perzistentních dat, obvykle spojené se softwarem pro aktualizaci a dotazování nad těmito daty. Jednoduchá databáze může být jediný soubor obsahující mnoho záznamů, z nichž každý obsahuje stejnou sadu polí, kde každé pole má určitou pevnou šířku.“* (Howe, 2005)

Dr. E. F. Codd z IBM v roce 1970 poprvé formuloval relační databázový model v periodiku vydaném organizací Association of Computer Machinery pod názvem *„A Relational Model of Data for Large Shared Data Banks“*. Tento Coddův model je přijat jako model pro relační systém řízení báze dat (relační SŘBD). Relační databázový model lze pochopit jako databázi, která ukládá svá data do tabulek, které jsou navzájem propojeny – mezi tabulkami jsou relace. Databáze založená na tomto modelu umožňuje definovat datové struktury, operace ukládání a vyhledávání, a omezení integrity. (Oracle Corporation, c1993-2013) (Howe, 2002) (Taylor, 2010)

S relačními databázemi je úzce spojen jazyk SQL (Structured Query Language). Používá se pro vytváření a aktualizaci dat v databázi a určování jejich struktury. Dotazování se k získaní požadovaných dat a kontrolu zabezpečení.

Jazyk byl vyvinut společností IBM v roce 1970 pod původním názvem SEQUEL (Structured English Query Language). S postupem času vznikalo spoustu různých verzí od různých jiných společností, což způsobilo nejednotnost. Až v roce 1986 byl jazyk převzat Americkou standardizační asociací (ANSI) a v roce 1987 Mezinárodní standardizační organizací (ISO) přidružené k Mezinárodní elektrotechnické komisi (IEC), které jazyk SQL standardizovali. To napomohlo k výrazné minimalizaci rozdílů mezi různými SQL dialekty. Prvním komerční implementaci SQL představila v roce 1979 firma Oracle (dříve Relational Software). (Oracle Corporation, c1993-2013) (Wilton, 2005)

Současnou verzí je standart ISO/IEC 9075: 2016 vydaný Mezinárodní organizací pro normalizaci/Mezinárodní elektrotechnickou komisí (ISO/IEC) v prosinci roku 2016. (ANSI, 2017)

Příkazy SQL jazyka lze rozdělit do tří hlavních částí, kde každá část slouží určitému účelu. Některé zdroje uvádějí navíc i čtvrtou část, která je zde také uvedena:

- Data Definition Language (DDL): Obsahuje příkazy, které definují strukturu databáze – tabulek, pohledů (views), apod. Těmito příkazy lze strukturu databáze vytvářet, modifikovat nebo mazat. Příkladem lze uvést příkazy CREATE TABLE (vytvoření nové tabulky) a ALTER TABLE (přidání, mazání, anebo modifikace sloupců v tabulce).
- Data Manipulation Language (DML): Obsahuje příkazy, které vkládají, aktualizují, odstraňují a dotazují hodnoty dat ze struktur definovaných jazykem DDL. Například INSERT INTO (vložení nových záznamů do tabulky), UPDATE (modifikace záznamů), SELECT (výběr dat z databáze).
- Data Control Language (DCL): Obsahuje příkazy, které chrání databázi před neoprávněným zneužitím nebo neúmyslném poškození a kontrolou přístupu. Uděluje nebo odebírá uživatelům práva používaní příkazu DDL a DML. Příkladem lze uvést GRANT (přidělení práv) a REVOKE (odebrání práv).
- Transaction Control Language (TCL): Příkazy určené ke správě transakcí v databázi. Používají se ke správě změn provedených příkazy DML. Například COMMIT (uložení transakce) a ROLLBACK (obnovení stavu na poslední vykonaný stav). (Jay A. Kreibich., 2010) (Data, c1999-2017)

3.2 MySQL

V průběhu let rostly požadavky na ukládání dat a databázové systémy, které se pokoušely tyto požadavky uspokojit. Avšak s tímto vývojem rostly náklady spojené s ukládáním dat, stejně jako zvýšení poptávky po produktech, které mohou běžet na více platformách a lze je optimalizovat na základě potřeb konkrétních typů organizací. V reakci na tyto změny se MySQL stal nejoblíbenějším systémem pro správu databáze s otevřeným zdrojovým kódem (DBMS) na světě. (Sheldon, 2005)

Obrázek 1 Logo MySQL (zdroj: mysql.com)

MySQL je v současnosti vyvíjen, distribuován a podporován společností Oracle Corporation. Domovská stránka, kde se nachází i obsáhlý a aktuální referenční manuál, je k nalezení na oficiálních stránkách (https://dev.mysql.com/doc/).

MySQL je systém řízení báze dat (SŘBN neboli z anglického DMS „database management system“). Databáze je strukturovaná kolekce dat, aby k těmto datům byl možný přístup, mohli se tyto data přidávat a zpracovávat, je zapotřebí systém řízení báze dat, jako je například právě MySQL Server. Může být použit jako samostatný nástroj nebo jako součást jiných aplikací.

Databáze v MySQL jsou relační, data jsou zde ukládány do samostatných tabulek. Databázové struktury jsou organizovány do fyzických souborů, které jsou optimalizovány pro rychlost. Z logického pohledu je databáze rozdělená do objektů, jako jsou tabulky, pohledy, sloupce a řádky. Uživatel má možnost nastavit pravidla upravující vztahy mezi datovými poli, jako například unikátnost a ukazatelé mezi různými tabulkami. Databáze prosazuje tato pravidla, proto ve správně navržené databázi nedojde k tomu, aby data byla nekonzistentní, duplicitní, osiřelá, zastaralá nebo chybějící.

Část „SQL“ z názvu MySQL naznačuje, že k práci s daty se používá standardizovaný jazyk SQL. V závislosti na programovacím prostředí se SQL zadává přímo, vkládáním příkazů SQL do kódu napsaném v jiném jazyce nebo použitím API, která skrývá syntaxi SQL.

MySQL Server může běžet na stolním počítači nebo notebooku společně s dalšími aplikacemi a webovými servery, nebo lze vyčlenit celý stroj, kdy se nastavení upraví tak, aby SQL Server využíval celou paměť, CPU a I/O[1] kapacitu.

MySQL pracuje jak na modelech klient/server, tak i ve vestavěných systémech. Databázový software pracující na klient/server se skládá z vícevláknového serveru SQL, který podporuje zálohy, klientské programy a knihovny, nástroje pro správu a aplikační programovací rozhraní (API). Ve vestavěných systémech je možnost integrace jako vícevláknové knihovny, kterou lze propojit s aplikací k získání menšího, rychlejšího a jednoduššího samostatného produktu. (Oracle Corporation, 2017a) (Čápka, 2017)

[1] CPU – centrální procesorová jednotka; I/O – vstup/výstup

3.2.1 Historie

Úplné počátky MySQL sahají až do 80. let 20. století s vytvořením databázového systému Unireg, který vytvořil Michael „Monty“ Widenius pro švédskou společnost TcX. V roce 1995 Monty přidal do Unireg rozhraní SQL, tímto vzniklo počáteční vydání MySQL. David Axmark později navrhl, aby se MySQL vydávalo pod duálním licenčním modelem, aby bylo možné volně dostupné použití, ale i restriktivnějšího použití, například ve vestavěných zařízeních.

Axmark, Monty a Allan Larsson založili v roce 1995 společnost MySQL AB, která měla na starosti podporu, servis a vývoj MySQL.

V roce 2001 MySQL začal podporovat transakce s integrací storage enginů DBD a InnoDB[2].

Roku 2008 nastalo spojení společnosti MySQL AB se společností Sun Microsystems. (Cabral, 2009)

Později roku 2010 společnost Oracle odkoupila společnost Sun Microsystems společně s právy a ochrannými známkami k MySQL. Na tuto událost byla vyhlášená on-line petice zakladatelem Montym, která ovšem neprošla. Reakcí na toto odkoupení společnosti, byl Montyho odchod ze Sun Microsystems a vytvoření odvozeniny na MySQL s názvem MariaDB.

Název „MySQL“ vznikl po složení zkráceniny jména Maria („My“) - dcery zakladatele Montyho a SQL. (Oracle Corporation, 2007)

[2] InnoDB je tzv. storage engine, který se stará o ukládání, čtení, změny a odstranění dat v databázi. Viz kapitola 3.2.3

V následující tabulce je uveden seznam hlavních verzí MySQL. Význam zkratky „GA“ je z anglického General Availability, takto označená verze je připravená k produkci ke koncovým uživatelům. (Cabral, 2009)

Verze	*Datum vydání (GA)*	*GA dosaženo s verzí*
Interní verze	23. května 1995	
Veřejná verze	31. srpna 1996	
Vydání na Windows	8. ledna 1998	
5.0	19. října 2005	5.0.15
5.1	14. listopadu 2008	5.1.30
5.5	3. prosince 2010	5.5.8
5.6	5. února 2013	5.6.10
5.7	21 října 2015	5.7.9
8.0	Probíhá vývoj	

Tabulka 1 Historie vývoje verzí MySQL (Cabral, 2009) (Oracle Corporation, 2017b)

3.2.2 Licence

MySQL je dostupný pro komerční nebo nekomerční použití. Rozdíl je především v různých úrovních podpory.

Open Source

MySQL lze volně stáhnout z Internetu a používat zadarmo. Zdrojový kód si lze prohlížet a upravovat. Nevýhodou je, že není zaručená technická podpora. Je na samotném uživateli, aby si vyřešil komplikace své databáze. (Oracle Corporation, 2017a) (Cabral, 2009)

Konkrétně se jedná o MySQL Community Edition, tato edice je k dispozici pod licencí GNU GPL (General Public License) verze 2. GNU GPL je licence pro svobodný software, která dovoluje volně používat, modifikovat i šířit takto licencovaný software, ale vyžaduje, aby byla odvozená díla dostupná pod toutéž licencí. Tedy musí existovat možnost získat bezplatně zdrojový kód. (Free Software Foundation, 1991)

Oracle navíc nabízí FOSS License Exception, jedná se o licenční výjimku, která dovoluje vývojářům, kteří vyvíjí software pod FOSS (Free and open-source software) licencí, použití některých součástí MySQL Client Libraries, konkrétně "MySQL Drivers" a "MySQL Connectors". MySQL Client Libraries jsou zpravidla licencovány pod GNU GPL v2.0, což by mohlo způsobit nekompatibilitu s FOSS licencí, avšak tato výjimka distribuci umožňuje. (Oracle Corporation, 2012)

Community Edition verze zahrnuje:

- **Architekturu zásuvných úložných enginů (storage engines)**
  - InnoDB, MyISAM, NDB (MySQL Cluster), Memory, Merge, Archive, CSV atd.
- **MySQL Replication** zlepšuje aplikační výkon a škálovatelnost.
- **MySQL Partitioning** zlepšuje výkon a správu databázových aplikací.
- **Stored Procedures** ke zlepšení produktivity vývojářů.
- **Triggers** k prosazování složitých pravidel na úrovni databáze.
- **Views** aby nedošlo k ohrožení citlivých informací.
- **Performance Schema** pro monitorování spotřeby zdrojů uživatelem / aplikací.
- **MySQL Connectors** pro vývoj aplikací v různých programovacích jazycích.
- **MySQL Workbench** pro vizuální modelování, SQL vývoj a administraci (Oracle Corporation, 2017i)

Komerční licence

Komerční licence je nutná v případě, kdy je kód MySQL dále prodáván jako integrovaná součást produktu. (Oracle Corporation, 2017a)

MySQL Enterprise Edition je komerční verze MySQL. Tato verze obsahuje vše, co obsahuje verze MySQL Community Edition, ale navíc obsahuje komponenty pro monitorování, online zálohu a nabízí vylepšenou škálovatelnost a bezpečnost. Technickou podporu MySQL Enterprise zastřešuje společnost Oracle.

Enterprise Edition verze zahrnuje:

- **MySQL Enterprise Monitor** je podnikový monitorovací systém, který sleduje MySQL servery, upozorňuje na možné komplikace a radí, jak tyto komplikace vyřešit.

- **MySQL Enterprise Backup** snižuje riziko ztráty dat pomocí online zálohy databází.
- **MySQL Enterprise Security** obsahuje autentizační a autorizační nástroje.
- **MySQL Enterprise Encryption** obsahuje sadu šifrovacích funkcí založených na OpenSSL[3] knihovně.
- **MySQL Enterprise Audit** pro monitorování a protokolování připojení a dotazovacích činností prováděných na konkrétních serverech.
- **MySQL Enterprise Firewall** umožnuje správci databáze povolit nebo zakázat spuštění příkazů SQL.
- **MySQL Enterprise Thread Pool** pro efektivnější správu vykonání příkazů při větším počtu připojených uživatelů. (Oracle Corporation, 2017c)

MySQL Enterprise Edition je jedna ze tří komerčních verzí, které Oracle nabízí na svých stránkách (https://www.mysql.com/products/). Jedná se o „střední" třídu. Rozdíl mezi dalšími verzemi je v množství nabízených funkcí. Zde je ceník jednotlivých verzí, detailnější výpis funkcí, které tyto verze obsahují, lze nalézt na oficiálních stránkách – viz. odkaz výše.

Verze	*Cena (1–4 socketový server) za rok*
MySQL Standard Edition	2 000 USD
MySQL Enterprise Edition	5 000 USD
MySQL Cluster Carrier Grade Edition	10 000 USD

Tabulka 2 Ceník komerčních verzí (rok 2017) (Oracle Corporation, 2017d)

3.2.3 Storage engines

Storage engines (úložné databázové stroje / úložiště dat) jsou komponenty MySQL, které vykonávají SQL operace pro různé typy tabulek. Storage engine lze také popsat jako subsystém, který spravuje tabulky v databázi. Jedná se o největší výhodu MySQL Server, většina jiných databázových systémů tuto možnost nepodporuje a používají pouze jeden storage engine, který nelze libovolně měnit.

[3] OpenSSL je knihovna obsahující kryptografické funkce a implementaci protokolů TLS a SSL (OpenSSL Software Foundation, c1999-2016)

V MySQL lze storage engine aplikovat na úrovni tabulky, což umožnuje nastavit pro jednotlivé tabulky v databázi různé storage engines podle potřeby administrátora. Při vytváření (CREATE TABLE) nebo úpravě tabulky (ALTER TABLE) lze přidat parametr ENGINE k nastavení (nebo změně) storage engine, který je spojený s tabulkou. Pokud se tento parametr neurčí, zvolí se výchozí InnoDB storage engine[4]. Například aplikace, která využívá převážně InnoDB tabulky, může mít jednu tabulku CSV pro export dat do tabulky a jednu tabulku MEMORY pro dočasné pracovní prostory.

Příklad příkazu:

```
CREATE TABLE t (i INT) ENGINE = MYISAM
```

Tento příkaz vytvoří novou tabulku s názvem „t“ a sloupcem „i“ datového typu integer (celé číslo) a storage engine MyISAM.

MySQL Server využívá architekturu zásuvných storage engines, což znamená, že lze přidávat nebo odebírat storage engines za běhu serveru. Toho je docíleno tak, že architektura MySQL Serveru odděluje kód spravující tabulky od kódu jádra databázového serveru. Kód jádra databázové serveru spravuje například mezipaměť dotazů, optimalizátor a obslužný program připojení. Kód storage engine zpracovává aktuální I/O tabulky. Toto oddělení kódu umožňuje použití více storage engines jedním jádrem databázového serveru.

Příkazem SHOW ENGINES lze zjistit, které storage engines server podporuje. Hodnota „Support“ označuje dostupnost a nabývá hodnot „YES“, „NO“ nebo „DEFAULT“ – dostupný, nedostupný nebo aktuálně nastaven jako výchozí.

Zde je uvedena ukázku výstupu tohoto příkazu:

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign
keys
Transactions: YES
          XA: YES
  Savepoints: YES
```

[4] Platí pro verzi MySQL 5.7, což je nejaktuálnější verze dostupná k produkci (2017)

MySQL verze 5.7 podporuje celkem deset různých storage engines. Ovšem nejpoužívanější jsou dva – InnoDB (výchozí) a MyISAM. Mezi další podporované patří: Memory, CSV, Archive, Blackhole, NDB, Merge, Federated a Example. (Cabral, 2009) (Schneller, 2010)

InnoDB

Výchozí storage engine v MySQL 5.7. InnoDB podporuje transakce kompatibilní se standardem ACID[5] se schopnostmi COMMIT, ROLLBACK a obnovení po havárii, které chrání data v tabulce.

InnoDB využívá zamykání na úrovni řádků („row-level locking“). Systém zamykání slouží k tomu, aby si transakce navzájem neměnili data a došlo tak k nekonzistenci dat. Pokud má operace „A“ delší prodlevu, může nastat situace, kdy operace „B“ jiného uživatele změní data operaci „A“. Tento systém zvyšuje výkon pří souběžné práci více uživatelů. (Oracle Corporation, 2017e) (Cabral, 2009) (Oracle Corporation, 2017f)

Uzamykání je možné buďto na úrovni řádků („row-level locking“) nebo na úrovni celé tabulky („table-level locking“). V případě uzamykání na úrovni řádků jsou uzamčeny jednotlivé řádky v tabulce. Tato možnost uzamykání je efektivnější, pokud se s danou tabulkou vykonává velké množství příkazů vkládání (INSERT) nebo úpravy záznamů (UPDATE). (Gilfillan, 2003) (Schneller, 2010)

InnoDB využívá primární klíče v tabulce. Při zakládání tabulky lze určit sloupec s primárním klíčem pomocí PRIMARY KEY, pokud tak uživatel neučiní, InnoDB se pokusí nalézt vhodný sloupec automaticky. Vhodnému sloupci se rozumí takový sloupec, který obsahuje nenulové a unikátní hodnoty. Pokud takový vhodný sloupec neexistuje, InnoDB automaticky vytvoří skrytý sloupec s ID každého řádku, který určí jako primární klíč. (Oracle Corporation, 2017g)

```
CREATE TABLE Uzivatele (
  UzivatelID int NOT NULL,
  Jmeno varchar(25),
  PRIMARY KEY (UzivatelID)
);
```

[5] „*ACID (Atomic, Consistent, Isolated, Durable) je obecně uznávaný seznam požadavků na bezpečný transakční systém.*“ (The PostgreSQL Global Development Group, 2014)

Ukázka příkazu, který určí sloupec „UzivatelID“ jako primární klíč, při vytvoření tabulky „Uzivatele“.

Další klíčovou vlastností InnoDB je podpora omezení relativní integrity dat pomocí cizích klíčů – FOREIGN KEY. (Oracle Corporation, 2017f)

```
CREATE TABLE Objednavky (
  ObjednavkaID int NOT NULL,
  CisloObjednavky int NOT NULL,
  UzivatelID int,
  PRIMARY KEY (ObjednavkaID),
  FOREIGN KEY (UzivatelID) REFERENCES Uzivatele(UzivatelID)
);
```

Tento příkaz vytvoří cizí klíč na sloupec „UzivatelID“ v tabulce „Uzivatele“, která byla vytvořena v předchozím příkladu.

MyISAM

Druhým nejpoužívanější storage engine je MyISAM původně založený na starším a již nepoužívaným storage engine ISAM.

Na rozdíl od InnoDB používá uzamykání na úrovni celé tabulky („table-level locking“), tento zámek tabulky brání jakékoliv jiné transakci přístup k tabulce. Proto se doporučuje MyISAM v případě, kdy se tabulka používá především ke čtení, pro časté zápisy je vhodnější použít InnoDB.

Dále MyISAM nepodporuje transakce, nastavení primárních a cizích klíčů a kontrolu souběžného přístupu více uživatelů. Na druhou stranu, jak již bylo zmíněno, je to ideální storage engine pro velice rychlé čtení dat, vhodný pro datové sklady.

Každá MyISAM tabulka je na disku rozdělena do tří souborů. Všechny tři soubory mají název (podle názvu tabulky) a příponu. Přípony jsou .frm pro soubor obsahující formát tabulky, .MYD pro soubor dat a .MYI pro indexový soubor. Výhodou je, že tyto soubory mohou být lehce zálohovány nebo zkopírovány na jiný server. (Oracle Corporation, 2017e) (Oracle Corporation, 2017h) (Cabral, 2009) (Schneller, 2010)

3.2.4 Odvozeniny

Odvozený systém, někdy také známý jako tzv. „fork“. Dochází k tomu v případě, kdy vývojář(i) zkopírují kód nebo část kódu, na kterém začnou dále pracovat a vytvoří tak samostatný software. (Dash, 2010) V případě MySQL vzniklo několik odvozenin, nejdůležitější nebo nejznámější z nich jsou uvedeny na obrázku 2[6] a dále v textu.

Obrázek 2 Vývoj MySQL a jeho odvozenin, Daniël van Eeden (Van Eeden, 2016)

MariaDB

Poté, co společnost Oracle převzala Sun Microsystems, opustil společnost zakladatel MySQL Michael Widenius a vyvinul odvozeninu s názvem MariaDB. Později se přidali i zbývající zakladatelé Axmark a Larsson. MariaDB je odvozenina patřící komunitě, což znamená, že nemá stejné licenční omezení jako standardní verze MySQL od Oracle.

Obrázek 3 Logo MariaDB (zdroj: mariadb.com)

[6] Z důvodu velikosti oříznuto, obrázku předchází verze MySQL 3.19 až 4.1

MariaDB je kompatibilní s MySQL, takže nejsou rozdíly v příkazech a API rozhraních. (Database Friends, b.r.) Jako příklady kompatibility se uvádí MySQL 5.1 s MariaDB 5.1, 5.2 a 5.3, anebo MySQL 5.5 s MariaDB 5.5 – na obrázku 2 jsou kompatibility uvedeny pomocí šipek. Tento fakt znamená, že ve většině případech je přechod z MySQL na MariaDB jednoduchý, bez nutnosti konverze datových souborů databáze. (MariaDB, 2017a)

Nejaktuálnější verze MariaDB 10.2 přináší funkce, které jsou v MySQL 8.0 stále ve vývoji. Jedná se například o rekurzivní tabulkové výrazy a funkci „window function“, která provádí výpočet přes množinu řádků tabulky souvisejících s aktuálním řádkem. (The PostgreSQL Global Development Group, c1996-2017a) Další důležitou změnou je, že tato verze používá storage engine InnoDB, na rozdíl od starších verzí, které používají XtraDB. (Ksiazek, 2017)

Storage engine XtraDB je obohacenou verzí InnoDB navrženou pro modernější hardware a obsahuje funkce užitečné ve vysoce výkonných prostředích. Je zpětně kompatibilní s InnoDB, takže může být lehce nahrazena za InnoDB. Používá se ve verzích MariaDB 5.1, 5.2, 5.3, 5.5, 10.0 a 10.1. (MariaDB, 2017b)

Od roku 2007 je ve vývoji nový storage engine s názvem Aria, který má za cíl stát se výchozím transakčním a zároveň netransakčním enginem pro MariaDB a MySQL. Vytváří ho stejní vývojáři, kteří stáli za vznikem starších engines. (MariaDB, 2017c)

Percona

Percona zpočátku nabízela sadu patchů pro zdrojový kód MySQL, které zlepšily výkon a funkčnost. Později Percona vydala vlastní odvozeninu MySQL, která zahrnovala tyto patche a přidala další funkce.

Obrázek 4 Logo Percona Server (zdroj: gagor.pl)

Percona vydala i vlastní upravenou verzi storage engine InnoDB z MySQL s názvem XtraDB.

Percona dále nabízí zdarma funkce, které jsou v případě MySQL od Oracle dostupné pouze v komerční edici Enterprise.

Percona Server je plně zpětně kompatibilní s MySQL. V základu jsou dodatečné funkce vypnuty, což vede k jednoduchému nahrazení, nedochází totiž k nekompatibilitě. (Ksiazek, 2017) (Percona LLC, c2006-2017)

Drizzle

Drizzle je již zaniklá open-source odvozenina z MySQL verze 6.0. Hlavní charakteristikou Drizzle je odlehčenost od nepotřebných funkcí, které MySQL nabízí. Jedná se hlavně o funkce, které jsou nepotřebné v oblasti cloud computingu. (Database Friends, b.r.)

3.3 PostgreSQL

PostgreSQL[7] je výkonný, open source a objektově-relační databázový systém dostupný na široké škále platforem a operačních systémů včetně Linux, UNIX a Windows. PostgreSQL databáze jsou známé svou dlouhodobou funkčností, v mnoha případech vyžaduje pouze malou nebo žádnou údržbu. Z toho vyplývá, že PostgreSQL poskytuje nízké celkové náklady na vlastnictví a údržbu.

Obrázek 5 Logo PostgreSQL
(zdroj: wiki.postgresql.org)

Je vyvíjen a udržován velkou skupinou vývojářů a přispěvatelů během více než 15 let aktivního vývoje a vylepšování. Díky tomu nabízí mnoho funkcí, které jsou u mnoha jiných databázových systémů označovány jako Enterprise. Například MVCC (MultiVersion Concurrency Control), jedná se o funkci, která v databázi zabraňuje vzájemnému blokování uživatelů, kteří provádí zápisy nebo čtení.

PostgreSQL je kompatibilní se systémem ACID, má podporu pro joins (spojení), views (pohledy), triggers (spoušť), uložené procedury (vícejazyčné) a podporu pro integritu dat – cizí a primární klíče, kontrola omezení, nenulovost apod. Podporuje ukládání binárně

[7] PostgreSQL [ˈpoʊst.ɡrɛs ˌkjuː ˈɛl], zkráceně Postgres [ˈpoʊst.ɡrɛs]

objemných objektů (obrázky, video, audiozáznamy). Má nativní programovací rozhraní pro C/C++, Java, .Net, Perl, Python, Ruby atd. Podporuje mezinárodní znakové sady a Unicode.

Dále za zmínku stojí možnost definovat sloupec tabulky jako multidimenzionální pole s proměnnou délkou (array). Zde uvádím příklad použití:

```
CREATE TABLE tabulka (
  sloupec     text,
  sloupec2    integer[],
  sloupec3    text[][]
  );
```

Tento příkaz vytvoří tabulku se sloupcem typu „text“ (proměnná s neomezenou délkou), jednorozměrný řetězec typu integer (celé číslo) a dvourozměrný řetězec textu. (The PostgreSQL Global Development Group, c1996-2017g)

Další charakteristiky PostgreSQL:

- Dodržuje standard SQL – ANSI-SQL: 2008
- Navržený modelem klient-server
- Souběžný design, kde se uživatelé vzájemně neomezují
- Konfigurovatelný a rozšiřitelný pro mnoho typů aplikací
- Rozsáhlé funkce ladění pro škálovatelnost a větší výkon
- Odolný software s dobře komentovaným kódem
- Nízkonákladová údržba pro vestavěné i podnikavé systémy
- Bezpečnost a vysoká dostupnost
- PostgreSQL je vysoce rozšiřitelný, lze například přidávat vlastní datové typy a měnit různé části systému pomocí pluginů.

Sada funkcí PostgreSQL je více srovnatelná s databázovým systémem Oracle, než je tomu tak u MySQL. Jediné, co spojuje PostgreSQL a MySQL, je to, že jsou oba open source. Filozofie je mezi nimi odlišná.

Významnými uživateli jsou například Apple, Skype a Yahoo!. (The PostgreSQL Global Development Group, c1996-2017b) (Riggs, 2010)

3.3.1 Historie

PostgreSQL vznikl z původního projektu Ingres, který vytvořil profesor počítačových věd Michael Stonebraker na University of California v Berkeley (UCB). Ingres, vyvíjený od roku 1977 do 1985, byl vytvářený podle klasické teorie relačních databázových systémů. (Stonebraker, 1986) Později se Stonebraker stal technickým ředitelem ve společnosti Informix Corporation. V roce 1986 Stonebraker zahájil Postgres jako následný projekt svého předchůdce Ingres, který nyní vlastní společnost Computer Associates. Postgres, vyvíjený od roku 1986 do 1994, měl přinést základy objektově-relačních databázových systémů.

Během tohoto osmiletého vývoje Postgres, na kterém pracovali kromě Stonebrakera i jeho postgraduální studenti, přinesl projekt koncepty objektově-relačních databázových systémů. Postgres se později stal proprietárním systémem, s názvem Illustra, který odkoupila společnost Informix a integrovala Postgres do svého softwaru Universal Server. V roce 2001 IBM zakoupila společnost za jednu miliardu dolarů.

V roce 1995, dva Stonebrakerovy postgraduální studenti Andrew Yu a Jolly Chen, nahradili původní dotazovací jazyk POSTQUEL modernějším jazykem SQL a přejmenovali systém na Posgres95.

V roce 1966, kdy opustil Postgres95 akademickou půdu, si všimla skupina vývojářů potenciálu systému a začala pracovat na jeho dalším rozvoji. Tím se stal Posgres95 open-source systémem, který přejmenovali na PostgreSQL. Během dalších let se PostgreSQL radikálně změnil. Kód se stal konzistentní a jednotný, vytvořily se regresní testy a reporty chyb, sepsala se dokumentace a opravilo se velké množství bugů.

V následující tabulce lze je uveden vývoj hlavních verzí. (The PostgreSQL Global Development Group, c1996-2017c)

Verze	*Datum vydání*
Ingres	1977
Postgres/Illustra	1986
Postgres95	1. ledna 1995
PostgreSQL 1.0	5. září 1996
6.0	29. ledna 1997
7.0	8. května 2000
8.0	19. ledna 2005
9.0	20. září 2010
10	5. října 2017

Tabulka 3 Historie hlavních verzí PostgreSQL (The PostgreSQL Global Development Group, c1996-2017d) (The PostgreSQL Global Development Group, c1996-2017c)

3.3.2 Licence

Jasnou výhodou PostgreSQL je, že se jedná o open source projekt, což znamená, že je svobodněji licencovaný. (Riggs, 2010) Konkrétně je PostgreSQL vydán pod licencí PostgreSQL License, což je open source licence, podobná licencím BSD nebo MIT.

Používání, kopírování, úprava a distribuce softwaru a jeho dokumentace za jakémkoli účelem, bez poplatků a bez písemného souhlasu, je povolena za předpokladu uvedení poznámky o autorských právech a odstavce PostgreSQL License, ve všech kopiích. (The PostgreSQL Global Development Group, c1996-2017e) Zde jsou uvedeny zmiňované odstavce:

„The university of california ("kalifornská univerzita") není v žádném případě odpovědna žádné třetí osobě za přímou, nepřímou, zvláštní, nahodilou nebo výslednou škodu, včetně ušlého zisku, způsobenou užitím tohoto softwaru a dokumentace k němu, a to i v případě, že the university of california byla informována o možnosti vzniku takové škody.

The university of california zejména neposkytuje jakékoli záruky, a to nejen záruky obchodovatelnosti a vhodnosti tohoto výrobku ke specifickým účelům. Níže uvedený software je poskytnut "jak stojí a leží" a the university of california není povinna zajistit jeho údržbu, podporu, aktualizaci, vylepšení nebo modifikaci.“ (The PostgreSQL Global Development Group, 2007)

Na oficiálních stránkách PostgreSQL se nachází softwarový katalog obsahující rozhraní, rozšíření a software související s PostgreSQL, vytvořený firmami, jednotlivci a open source projekty, dostupný pod různými druhy licencí (komerční/open source). Jsou zde k dispozici:

- Nástroje pro správu/vývoj
- Aplikace
- Nástroje pro Clustering/replikace
- Ovladače a rozhraní
- Servery odvozené od PostgreSQL
- Rozšíření PostgreSQL
- Procedurální jazyky
- Nástroje pro vytváření přehledů

Jádro PostgreSQL je dostupné z několika zdrojů, pod různými formáty a pro řadu různých operačních systémů:

- BSD
    - FreeBSD, OpenBSD
- Linux
    - Red Hat, Debian, Ubuntu, SuSE, OpenSuSE
- macOS
- Solaris
- Windows (The PostgreSQL Global Development Group, c1996-2017f)

3.3.3 Storage engines

Aktuálně PostgreSQL ve verzi 10 nepodporuje zásuvné storage engine jako je tomu u MySQL, kde byla možnost si vybrat mezi několika storage engine, viz kapitola 3.2.3.

Momentálně veškeré ukládání dat probíhá přes heapam.c. Toto rozhraní předpokládá, že existuje relace a CTID (tuple identifier), cože je identifikátor n-tice. N-tice (tuple) je synonymum pro řádek v tabulce. CTID určuje fyzickou pozici řádku v tabulce. CTID má formát „(blok, identifikační index řádky)“, k podrobnějšímu popis struktury bloků se vrátím dále v textu. CTID je dostupný pro každou tabulku, ale není viditelný, dokud není zvlášť dotázán.

```
SELECT ctid, * from nazev_tabulky;
```

Aby mohl v budoucnu PostgreSQL podporoval výběr storage engine, musela by se změnit současná implementace heapam.c, která nyní vrací strukturu HeapTuple obsahující n-tice, protože různé implementace mohou mít zcela odlišný způsob reprezentace n-tice (řádku) v uložišti.

O tuto problematiku se již zajímala během roku 2015 společnost 2ndQuadrant jako jedna z dílčích součástí většího projektu, který si kladl za cíl implementovat do PostgreSQL sloupcové ukládání.

Fyzické uložení databáze

Fyzicky je vytvořená databáze uložená tímto způsobem – datové soubory, které používá cluster databáze, jsou uloženy společně s daty clusteru. Databázový cluster je paměťová oblast na disku, kde se nachází kolekce databází, které jsou spravovány jedinou instancí běžícího databázového serveru. Tato oblast se nazývá PGDATA. PGDATA může mít různé umístění na základě používaného operačního systému. Každá databáze má unikátní identifikační číslo – OID. Pro výpis všech databází a jejich OID lez použít například tento příkaz:

```
select oid, datname from pg_database;

  oid  |   datname
-------+-------------
17447 | moje_databaze
```

Výše zobrazené OID se liší podle systému. Výsledek říká, že uživatelem vytvořená databáze „moje_databaze“ a všechny její soubory jsou uložené v „PGDATA/base/17447“.

Fyzické uložení řádků

Tabulky jsou uloženy jako řetězec bloků pevné velikosti 8 kB. Všechny bloky v tabulce jsou logicky ekvivalentní, proto může být každý konkrétní řádek uložen v libovolném bloku.

Heap file je struktura obsahující kolekci bloků. Struktura bloku obsahuje hlavičku bloku, která poskytuje například informace o kontrolním součtu. Dále obsahuje řetězec identifikátorů (CTID), které přesně odkazují na aktuální řádek. Vzhledem k tomu, že se CTID nikdy nemění, pokud není uvolněn, může být dlouhodobě používán k odkazování na řádek, a to i v případě, že se řádek pohybuje v bloku. (Belaid, 2015) (Smagen, 2017)

3.3.4 Odvozeniny

Na PostgreSQL existuje velké množství rozšíření a odvozenin, důvodem je licenční politika a otevřený kód. Některé odvozeniny se odchýlí od původní licenční politiky, stanou se proprietárními a jejich používání je zpoplatněno.

Příkladem odvozeniny, která je aktuálně vydaná pod komerční licencí je **EDB Postgres Advanced Server**. Jedná se databázi, která je určená především pro podnikovou sféru a její hlavní předností je kompatibilita s Oracle databázemi, včetně PL/SQL[8], Built-in packages[9] a nástrojů pro administrátory pro urychlení a zjednodušení migrací. (EnterpriseDB Corporation, 2017)

Obrázek 6 Logo Postgres Advanced Server (zdroj: enterprisedb.com)

Dalším příkladem komerční odvozeniny je relační databáze **Amazon Aurora** kompatibilní s PostgreSQL i MySQL, postavená především pro cloud. Je řízena službou Amazon Relational Database Service, která automatizuje administrativní úlohy, jako jsou nastavení databáze, opravy a zálohy. Příkladem další funkce je schopnost databáze automaticky zvětšit kapacitu až na 64TB bez nutnosti přerušení dostupnosti dat. (Amazon Web Services, 2017)

Obrázek 7 Logo Amazon Aurora (zdroj: aws.amazon.com)

Opakem je **Postgres-XL**, která je open source. Postgres-XL je paralelní databáze postavená na PostgreSQL 9.5. Umožňuje podporovat různorodé pracovní zátěže na stejném horizontálně škálovatelném serveru. (The PostgreSQL Global Development Group, 2016)

Obrázek 8 Logo Postgres-XL (zdroj: postgres-xl.org)

[8] PL/SQL je procedurální nadstavba jazyka SQL od Oracle.

[9] Built-in packages je kolekce PL/SQL objektů napsaných společností Oracle a uložených přímo v databázi.

Příkladem rozšíření je PostGIS, který přidává podporu pro geografické objekty. Princip spočívá v možnosti uložení geografických souřadnic a tvarů v PostgreSQL pro geografické informační systémy (GIS). Přidává své typy, funkce a indexy. (PostGIS Project Steering Committee, b.r.)

Obrázek 9 Logo PostGIS (zdroj: postgis.net)

3.4 Metodika měření výkonu

Tato kapitola slouží k seznámení s použitou metodikou měření databázových systémů.

3.4.1 Transaction Processing Performance Council

Transaction Processing Performance Council (TPC) je organizace, která je nejrozšířenějším průmyslovým subjektem pro definování benchmarků v databázovém průmyslu, které jsou uznávány všemi předními dodavateli databází. Specifikace TPC benchmarků jsou dostupné bezplatně pro veřejnost. (HammerDB, 2017c)

Mnoho společností z oblastní výpočetní techniky využívá TPC benchmarků k zjištění a ilustraci konkurenceschopnosti, a k sledování výkonu svých produktů. Kupující se podle oficiálních zveřejněných výsledků srovnání rozhodují při nákupu nových produktů.

Členy TPC jsou dodavatelé počítačových systémů a několik dodavatelů databází, jako například Cisco, Dell, IBM, Microsoft, Intel atd. V členství se nachází 20 až 40 členů z celého světa. (TPC, c2001-2018a)

Cena členství je 15 tisíc dolarů ročně a jeho výhodou je vliv člena na vývoj TPC benchmarků, včasný přistup k probíhajícím řízením a možnost zlepšení svých produktů na základě metodik benchmarků. (TPC, c2001-2018b)

4 Praktická část

4.1 Prostředí měření

Aby výsledky měření obou databázových systémů byly dostatečně adekvátní a přesné, je potřeba oba měřené databázové systémy nainstalovat a otestovat za stejných podmínek, tj. na stejný stroj a operační systém.

K měření je použit osobní počítač sestavený autorem práce s nainstalovaným operačním systémem Windows 10 Pro v 64bitové verzi. Podrobné parametry osobního počítače jsou sepsány v následující tabulce.

Komponenta	*Parametr*
CPU	Intel Core i5 6500 3.20GHz, Skylake 14nm
Počet CPU soketů	1
RAM	8 GB Single-Channel
Základní deska	MSI B150 GAMING M3
GPU	NVIDA GeForce GTX 790, 4095 MB
HDD	Seagate 931 GB
SSD	INTEL 111 GB

Tabulka 4 Parametry použitého stroje

4.2 Instalace databázových systémů

4.2.1 Instalace MySQL

V případě instalace MySQL na Windows je na oficiálních stránkách ke stažení přehledný interaktivní grafický instalátor, tzv. MySQL Installer.

Po spuštění MySQL Installer je v prvním kroku zapotřebí potvrzení licenčních podmínek. Komunitní verze MySQL je, jak již bylo zmíněno v kapitole 3.2.2., dostupná pod licencí GNU GPL. Dalším krokem je výběr instalovaného balíčku. Na výběr je defaultní nastavení, kdy se nainstalují všechny produkty potřebné pro vývoj a práci s MySQL. Mezi další možnosti patří například instalace pouze MySQL Server nebo MySQL Client, bez serveru. Po potvrzení dojde ke stažení a instalaci všech vybraných produktů.

Po úspěšné instalaci následují kroky konfigurace MySQL Server. Konfigurace byla ponechána ve výchozím nastavení.

Pro měření je použita nejaktuálnější verze MySQL 5.7 (64 bit) nainstalována a zprovozněna na SSD disku, viz. tabulka 4.

4.2.2 Instalace PostgreSQL

V případě instalace PostgreSQL je situace podobná, jako je tomu u MySQL. Na oficiálních stránkách se nachází grafický interaktivní instalátor, který obsahuje PostgreSQL Server, pgAdmin – grafický nástroj pro správu databází, a StackBuilder – nástroj pro instalaci doplňků třetích stran.

Jedná se o klasický instalátor, ve kterém je v jednotlivých krocích zvolena cesta instalace, heslo pro administrátora databáze, port, licenční podmínky apod. Konfigurační parametry byly ponechány ve výchozím nastavení.

Pro měření je použita nejaktuálnější verze PostgreSQL 10.2 (64 bit) nainstalována a zprovozněna na SSD disku, viz. tabulka 4.

4.3 Měření pomocí nástroje HammerDB

K měření databázových systémů MySQL a PostgreSQL je zvolen grafický open source nástroj pro testování zátěže HammerDB. Tento nástroj je dostupný pro Linux a Windows a nabízí testování databází běžících na libovolném operačním systému.

HammerDB je automatizovaný a podporuje skriptování. Vytvoří schéma, načte jej daty a poté simuluje pracovní zatížení více virtuálních uživatelů proti databázi podle TPC-C (OLTP[10]) nebo TPC-H (analytické) scénáře. (HammerDB, 2017a)

Implementuje specifikace některých benchmarku dle metodiky TPC, která je blíže popsána v kapitole 3.4.1.

Ovšem důležitou informací, na kterou HammerDB upozorňuje ve své dokumentaci, je ta, že plně neimplementuje benchmarky TPC-C a TPC-H, protože možnost plné simulace je neuvěřitelně omezená, zejména z důvodu požadované velikosti datových sad. Místo toho HammerDB přebírá podstatu těchto oficiálních benchmarků a implementuje jí do podoby, která je použitelná pro běžného uživatele. (HammerDB, 2017b)

[10] OLTP (Online Transaction Processing) je způsob zpracování dat v databázových systémech pro systémy orientované na transakční data. (OldanyGroup, 2018)

4.3.1 Klientské knihovny databázových systémů

Prvním důležitým krokem před zahájením měření za použití HammerDB je ověření dostupnosti klientských knihoven databáze. Ve většině případech to znamená postupovat podle pokynů instalace databázových systémů. Bez tohoto kroku se nebude HammerDB schopný připojit k databázi. Na Linux je zásadní proměnná prostředí LD_LIBRARY_PATH, která obsahuje cestu ke knihovně a nastavuje se příkazem export. Na Windows se proměnná prostředí se stejnou funkcí nazývá PATH. V případě MySQL je cesta ke klientské knihovně nastavena automaticky během instalace. Pro PostgreSQL je většinou nutno tuto cestu nastavit manuálně přes příkazovou řádku.

Nejdříve je nutné najít a spustit shell pro HammerDB.

```
C:\Program Files\HammerDB-2.23\bin>tclsh86t.exe
```

Poté otestovat dostupnost knihovny:

```
% package require Pgtcl
couldn't load library "C:/Program Files/HammerDB-
2.23/lib/pgtcl2.0.0/libpgtcl.dll": this library or a dependent
library could not be found in library path
```

Pokud se vypíše výše uvedená chyba, je potřeba přidat cestu ke klientské knihovně PostgreSQL pomocí příkazu set na proměnnou PATH.

```
set PATH=%PATH%;C:\Program Files\PostgreSQL\10\bin
```

Poté se znovu spustí shell a opakuje se postup pro ověření dostupnosti knihovny. Příklad úspěšného výstupu vypadá takto:

```
C:\Program Files\HammerDB-2.23\bin>tclsh86t.exe
% package require Pgtcl
2.0.0
```

4.3.2 OLTP testování založené na specifikaci TPC-C

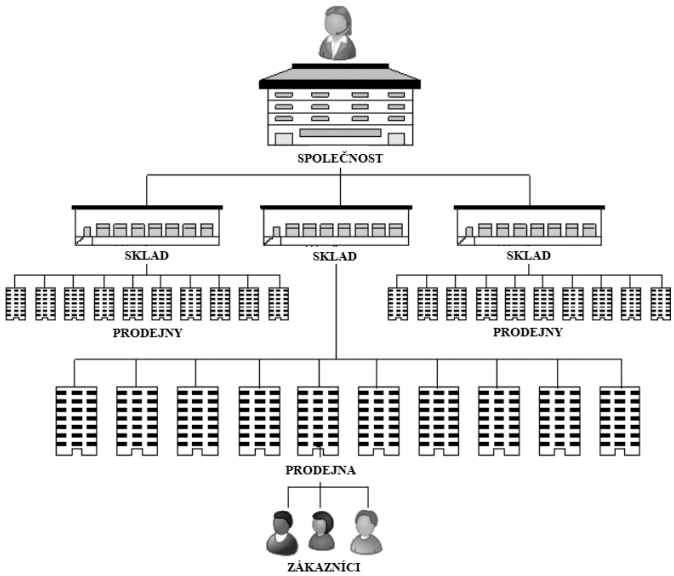
Pracovní zatížení (anglicky „workload“) je v HammerDB založeno na myšlence být co nejblíže příkladu zveřejněném ve specifikaci TPC-C. Implementace pracovního zatížení není závislá na konkrétní databázi nebo hardware, proto je spolehlivá, škálovatelná a testovaná tak, aby poskytovala opakovatelné a konzistentní výsledky. To znamená, že pokud se test provede opakovaně na totožně nakonfigurovaném hardwaru a softwaru, budou výsledky stejné při každém pokusu v mezích náhodného výběru transakcí, což činí obvykle 1 %.

Pokud se dosáhne konzistentního výkonu, znamená to, že je dosaženo maximálního využití CPU. Ovšem překážkou tomuto ideálnímu stavu se staví hardwarová konfigurace paměti, I/O a sítě. Dalším omezením je i využití procesoru, paměti a šířky pásma sítě, což ovlivňuje operační systém, na kterém databáze běží. Vliv mají i konfigurace a vlastnosti jednotlivých databázových systémů.

Z těchto důvodu nejsou výsledky měření srovnávány z absolutního hlediska, ale z relativního hlediska, tzn. zda porovnávané databázové systémy vykazují či nevykazují příliš velké odchylky ve výsledcích.

4.3.3 Pracovní zátěž (workload) TPC-C

Pracovní zátěž TPC-C v HammerDB je abstrakcí objednávkového systému společnosti, který zpracovává a dodává objednávky od zákazníků. Společnost prodává 100 000 produktů, které uchovává ve svých skladech. Každý sklad je propojený s 10 prodejnami a každá prodejna obsluhuje 3000 zákazníků. Zákazníci si objednávají produkty pomocí objednávek o různém počtu kusů. Pokud dojde produkt na skladě, je produkt dodán z jiného skladu.



Obrázek 10 Struktura společnosti (upraveno) (HammerDB, 2017c)

Kromě toho, že systém zadává objednávky, umožňuje také platby, doručování objednávek a kontrolu počtu produktů ve skladu. Pracovní zátěž se tedy skládá z pěti typů transakcí:

1. **New-order**: vytvoření nové objednávky zákazníkem.
2. **Payment**: zjištění zůstatku zákazníka a zaznamenání platby.
3. **Delivery**: dodání objednávky.
4. **Order-status**: Zjištění stavu nejnovější objednávky.
5. **Stock-level**: Zjištění počtu produktů ve skladu.

4.3.4 Vytvoření testovacího schéma TPC-C pro MySQL

Pro vytvoření testovacího schéma TPC-C je zapotřebí v HammerDB zvolit testovaný databázový systém, v tomto případě nejdříve MySQL, a MySQL Server musí být spuštěný. V možnostech vygenerování schématu se vyplní potřebné parametry. Tyto parametry jsou například:

- **MySQL Host: 127.0.0.1**, jméno hosta, které generátor schématu použije k připojení k běžící databázi. V tomto případě na lokálním serveru.
- **MySQL Port: 3306**, číslo síťového portu. 3306 je výchozí port pro MySQL databáze.
- **MySQL User: root**, jméno uživatele, který má právo vytvořit databázi.

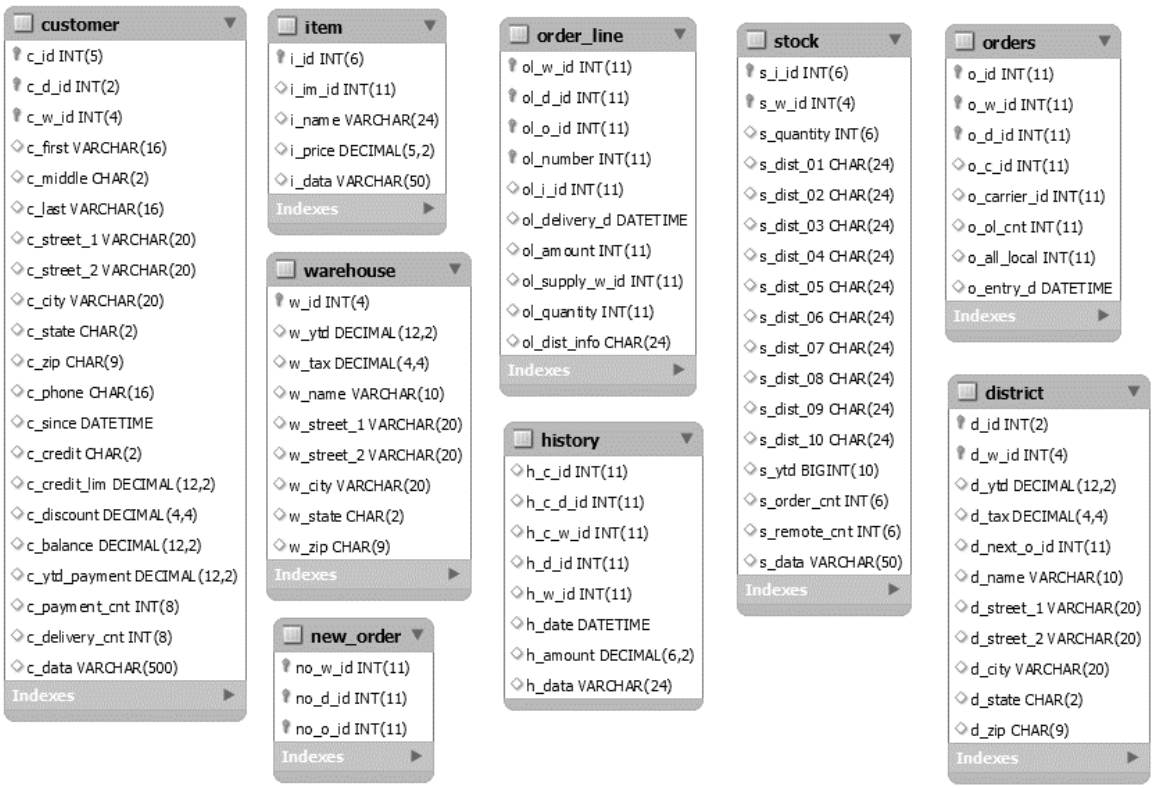
Další parametry jsou například heslo pro připojení k zadanému uživateli (root), název vytvořené databáze a storage engine. Podmínkou při výběru storage engine je podpora transakcí. Podrobně jsou popsány storage engines v kapitole 3.2.3. Dotázáním databáze příkazem `show engines` lze zjistit, který storage engine splňuje podmínku podpory transakcí. Ve výchozím nastavení je jedinou možností engine InnoDB.

Dalším důležitým parametrem je Počet Skladů (Number of Warehouses) – viz. obrázek 10. Tento parametr se podle dokumentace HammerDB nastavuje podle počtu CPU soketů (HammerDB, 2017d). To odpovídá 200 až 250 Skladů na každý soket. V tomto případě má používaný stroj jeden soket, proto je tento parametr nastaven na 225 Skladů.

Dalším parametrem je Počet Virtuálních Uživatelů (Virtual Users), který udává počet virtuálních uživatelů vykonávajících skript. Virtuální uživatelé představují zákazníky, kteří vykonávají akce nad schématem vyvoláním transakcí. Tento parametr je nastaven podle počtu CPU jader, v mém případě tedy čtyři.

Po potvrzení se HammerDB přihlásí na vybraného MySQL hosta se zadanými přihlašovacími údaji a vytvoří schéma podle konfigurace.

Vygenerované schéma databáze tpcc lze vidět na následujícím obrázku. Velikost databáze je 18,8 GB. Schéma tabulek a sloupců je vytvořené pomocí funkce reverzního inženýrství v MySQL Workbench, což je grafický administrativní nástroj dodávaný v balíku MySQL.



Obrázek 11 Schéma databáze tpcc (snímek obrazovky)

Na následujícím obrázku, který je opět pořízený z MySQL Workbench, lze vidět pět typů transakcí (procedur) vytvořených v zátěži. Tyto transakce již byly podrobněji popsány výše.

Name	Type	Definer
DELIVERY	PROCEDURE	root@localhost
NEWORD	PROCEDURE	root@localhost
OSTAT	PROCEDURE	root@localhost
PAYMENT	PROCEDURE	root@localhost
SLEV	PROCEDURE	root@localhost

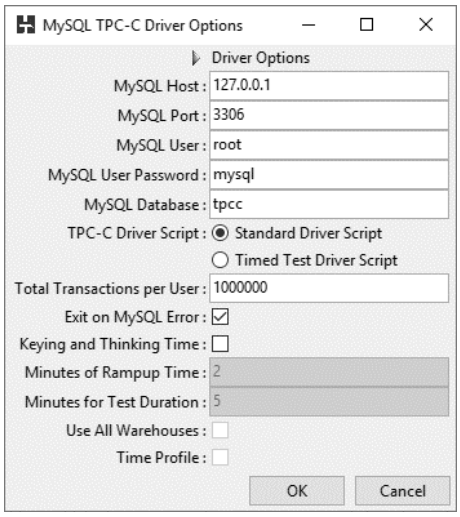
Obrázek 12 Uložená procedury databáze tpcc (snímek obrazovky)

V tento moment je vytvoření dat kompletní a lze již zahájit testovací přípravy před samotným zátěžový testem TPC-C na databázi MySQL.

4.3.5 Předběžné testy před měřením MySQL databáze

Ve fázi po vytvoření databáze, ale před zahájením měření, je nejdříve zapotřebí otestovat prostředí a konfiguraci databáze. Tímto předběžným testováním se lze ujistit, že je použitá vhodná konfigurace a pozdější výsledky měření budou přesné a nezkreslené.

V možnostech nazvaných „Driver Script“ se nastaví parametry pro vygenerování skriptu, který poté interaguje s TPC-C schématem na databázovém serveru. Parametry host, port, user, password a database slouží k připojení do databáze, na které se nachází TPC-C schéma.



Obrázek 13 Možnosti skriptu MySQL TPC-C v HammerDB (snímek obrazovky)

Na výběr jsou dva druhy skriptu:

1. **Standard Driver Skript**

   Tento skript je prováděn všemi virtuálními uživateli a volí se v případě, kdy je cílem měření vytvořit zátěž proti databázi a sledovat rychlost transakcí.

2. **Timed Test Driver Script**

   Tento skript nabízí navíc další možnosti. HammerDB spustí časované testování, změří výsledky a reportuje průměrnou transakční rychlost po daný časový úsek.

Dalším parametrem je „Total Transactions per User“, který určuje hodnotu, kolikrát každý virtuální uživatel provede celý skript před odhlášením z databáze.

Parametr „Exit on MySQL Error“ ovlivní chování jednotlivých virtuálních uživatelů. Pokud je tento parametr zvolen, znamená to, že pokud dojde v průběhu provádění transakcí k chybě MySQL, tak uživatel ukončí svou činnost.

Další parametr, který stojí za zmínku je „Keying and Thinking Time“. Pokud je tento parametr zvolen, znamená to, že virtuální uživatelé budou simulovat chování skutečných uživatelů, kteří potřebují čas na vytvoření objednávky a přemýšlí nad výstupy.

Všechny parametry jsou ponechány v původním nastavení, kromě parametru „Exit on MySQL Error“, aby bylo možné poznat, zda dojde k chybě.

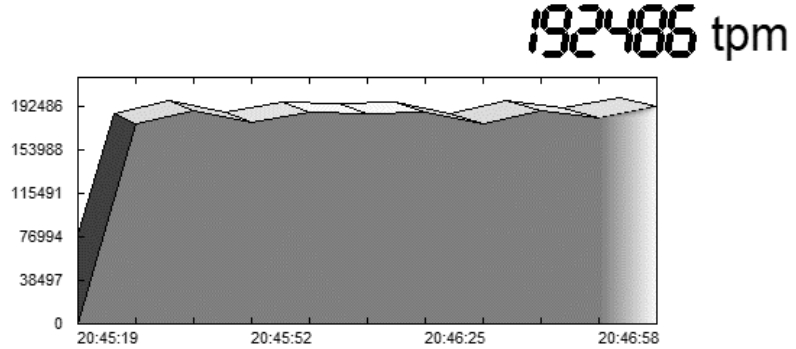
V rámci předběžného testování budou vytvoření jen dva virtuální uživatelé. Po spuštění skriptu se oba uživatelé připojí k databázi podle definovaných přihlašovacích parametrů a

zahájí provádění transakcí. Na obrázku níže lze sledovat přítomnost a aktivitu obou uživatelů v MySQL Workbench.

Id	User	Host	DB
65	root	localhost	None
1	None	None	None
61	root	localhost	tpcc
62	root	localhost	tpcc

Obrázek 14 Připojení virtuální uživatelé – MySQL Workbench (snímek obrazovky)

Na následujícím obrázku pořízeném v průběhu aktivity obou virtuálních uživatelů lze vidět ukazatel transakční rychlosti. Jedná se o snímek z HammerDB v ranné fázi průběhu testu.



Obrázek 15 Ukazatel transakční rychlosti v předběžném testování MySQL (snímek obrazovky)

4.3.6 **Měření výkonnostního profilu MySQL databáze**

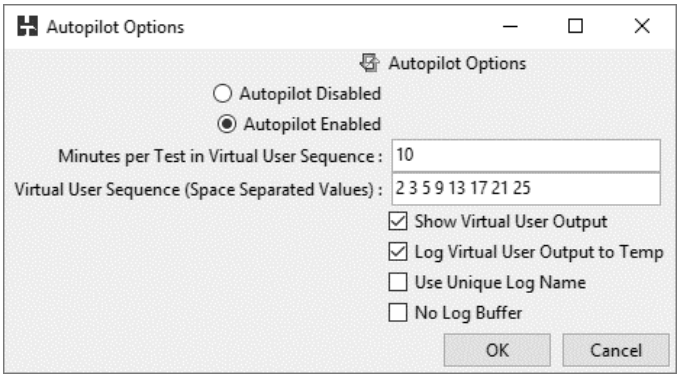
Po úspěšném otestování konfigurace přichází samotné měření výkonnostního profilu databázového systému. Cílem je změřit maximální počet transakcí, které dokáže databázový systém podporovat, při dané konfiguraci CPU, paměti, I/O a operačního systému. Dalším faktorem je i konfigurace samotného databázového systému. Tato konfigurace SŘBD se nachází v případě MySQL v souboru my.cnf (pro Linux) a my.ini (pro Windows). Tyto konfigurace by se neměli měnit v průběhu měření, protože by mohlo dojít ke zkreslení výsledků.

V možnostech „Driver Script“, které jsou již popsány výše, jsou nastaveny hodnoty podobně jako na obrázku č. 13. Rozdílné nastavení spočívá ve změně druhu scriptu na „Timed Test Driver Script“ a parametr „Exit on MySQL Error“ je nastaven na FALSE (vypnuto). Kromě těchto dvou parametrů jsou všechny ostatní ponechány ve výchozím nastavení.

Při nastaveném skriptu „Timed Test Drive Script“ je vždy jeden virtuální uživatel rezervován pro kontrolu časování testu, měření průměrných hodnot transakční rychlosti a vrácení těchto hodnot na výstup.

K měření je použita funkce Autopilot, kterou HammerDB nabízí. Tato funkce simuluje databázového správce, který automaticky vykonává sekvenci testů s rostoucím počtem virtuálních uživatelů a zaznamenává výsledky.

Na obrázku níže lez vidět nastavení autopilota.



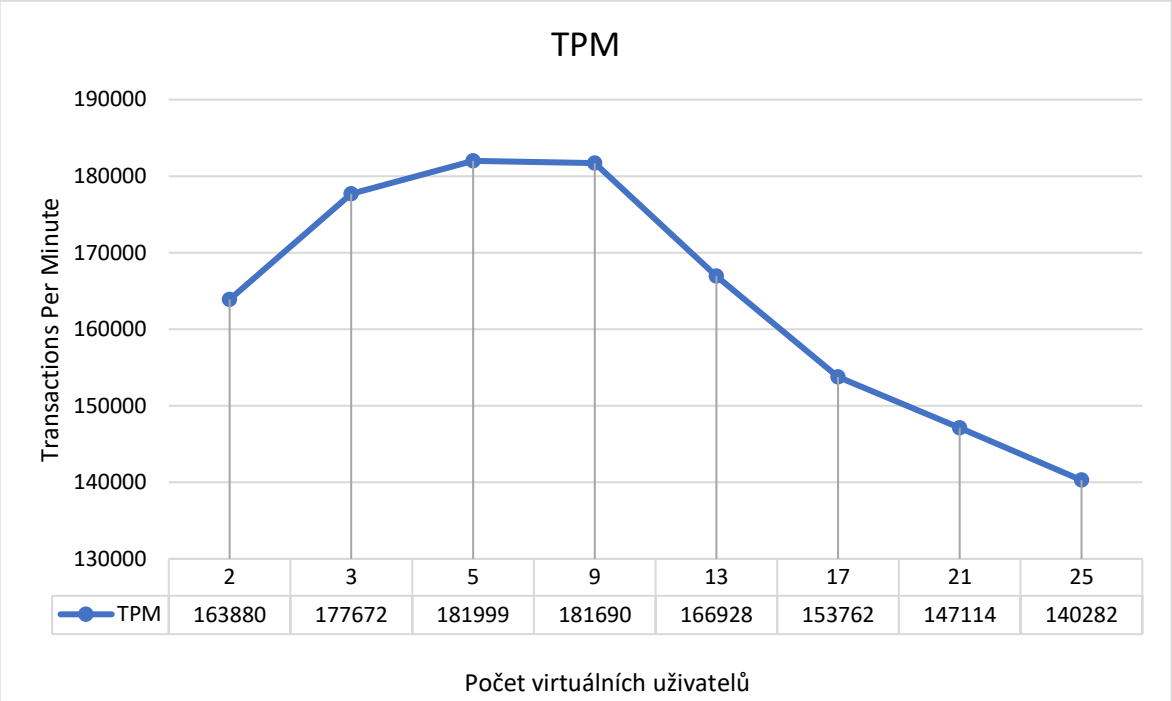
Obrázek 16 Nastavení Autopilota v HammerDB pro MySQL (snímek obrazovky)

V poli „Virtual Users Sequence“ jsou definovány rostoucí počty virtuálních uživatelů, se kterými je provedeno automatické měření. Doba, po kterou je měření vykonáváno pro jednotlivé počty virtuálních uživatelů, je definovaná v poli „Minutes per Test in Virtual User Sequence“. Například běží test nejdříve pro jednoho uživatele, po 10 minutách pro dva atd. Je důležité si uvědomit, že je vybrán časovaný skript (Timed Test Driver Script), proto je vždy počet virtuálních uživatelů, kteří provádí test, n–1, kde n je zadaný počet. Důvod již byl popsán.

Po potvrzení nastavení a spuštění měření, zahajuje autopilot svou činnost. Po dokončení jsou výsledky zaznamenány a uloženy.

4.3.7 Výsledky měření autopilotem pro MySQL

První výsledek měření lze vidět na následujícím grafu. Výsledky jsou zpracovány v MS Excel do podoby grafů.



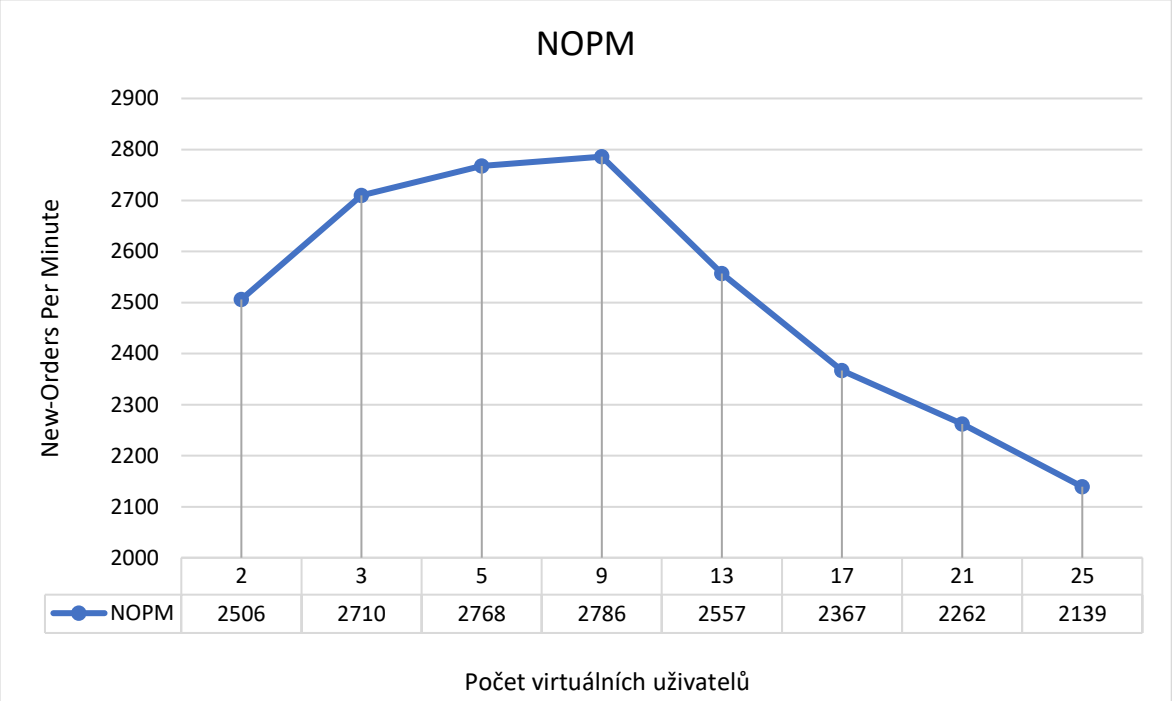
Obrázek 17 Graf výsledků měřením TPM pro MySQL (vlastní tvorba)

Tento graf znázorňuje výsledek měření výkonnostního profilu prováděného v sekvenci s různými počty virtuálních uživatelů v intervalech 10 minut proti databázi MySQL 5.7 se storage engine InnoDB. Přitom každý virtuální uživatel provede 1 000 000 transakcí vybíraných náhodně z množiny druhů transakcí, které již byly vyjmenovány a popsány výše. Jednotkou je TPM (Transactions Per Minute), což je jednotka představující počet provedených transakcí za minutu. TPM je extrahována z transakčních dat databáze.

Z grafu lze vypozorovat, že již při 5 virtuálních uživatelích značně klesat výkon a hodnota TPM. Důvodem toho jevu je, že každý virtuální uživatel vyžívá ke svému běhu jedno jádro CPU, proto jsou čtyři uživatelé optimálním počtem pro provádění tak velkého počtu transakcí. Databáze se nachází na lokálním serveru, kde je k běhu využívané čtyřjádrové CPU, proto začne výkon klesat již při pěti připojených uživatelích.

Dalším výstupem stejného měření je i jednotka NOPM (New-Orders Per Minute). V TPC-C je tato jednotka představována jako propustnost. Je zjištěná jako počet transakcí typu New-Order, tedy vytvoření nové objednávky, za minutu, které systém generuje, zatímco systém provádí další čtyři typy transakcí (Payment, Delivery, Order-status, Stock-

level). NOPM je jednotka extrahována z databázového schématu a není závislá na konkrétní implementaci databáze. Proto je to jednotka, která se na rozdíl od TPM, primárně doporučuje k porovnávání různých databázových systémů.



Obrázek 18 Graf výsledků měřením NOPM pro MySQL (vlastní tvorba)

Na první pohled lze vidět podobnost grafu NOPM s grafem TPM. Důvody pro pokles hodnot NOPM jsou stejné, jak tomu je u TPM.

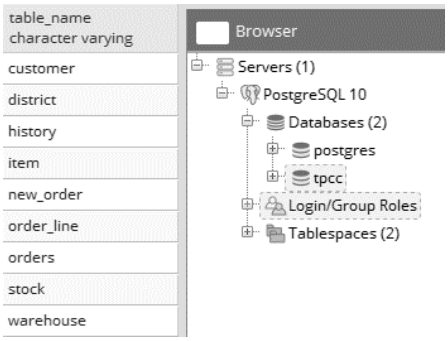
Měření probíhalo celkem 1h 24min.

4.3.8 Předběžné testy před měřením PostgreSQL databáze

Obecně je metodika postupu měření pro PostgreSQL stejná, jako je tomu v předchozích krocích s MySQL. Proto jsou následující kapitoly týkající se měření výkonu PostgreSQL již stručnější a zaměřeny spíše na zásadní změny nebo komplikace, na které jsem narazil.

Při pokusu o vytvoření tpcc databáze v databázovém systému PostgreSQL pomocí HammerDB došlo k chybě při pokusu o načtení knihovny Pgtcl. Tento problém byl vyřešen přidáním cesty ke knihovně do proměnné prostředí PATH. Postup je již detailněji popsán v kapitole 4.3.1.

Na obrázku č.19 lze vidět úspěšné vytvoření tpcc databáze v PostgreSQL. Snímek je pořízen z administrátorského nástroje pgAdmin 4. Velikost databáze je 25 GB.

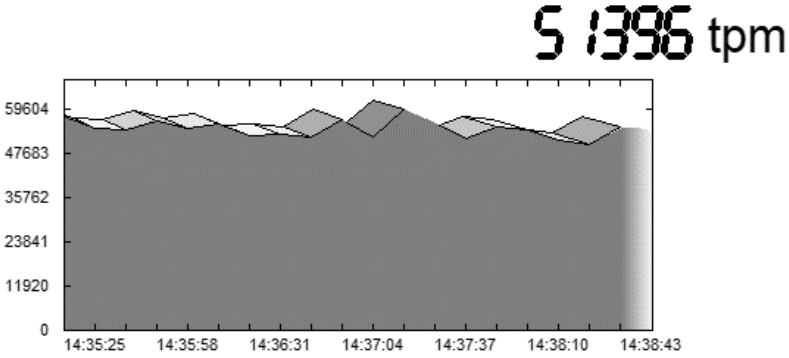


Obrázek 19 Databáze tpcc v PostgreSQL (snímek obrazovky z pgAdmin)

V rámci předběžného otestování funkčnosti, před samotným měřením, jsou v možnostech skriptu zvoleny podobné parametry, jako tomu bylo u MySQL. To znamená, že je vybrán standartní skript („Standard Driver Script“), 1 000 000 transakcí na uživatele a je nastaven parametr „Exit on PostgreSQL Error“. Ostatní parametry jsou ponechány ve výchozím nastavení. Test je opět spuštěn s dvěma virtuálními uživateli.

Přítomnost a aktivita obou virtuálních uživatelů je pozorovatelná v nástroji pgAdmin nebo spuštěním dotazu `SELECT * FROM pg_stat_activity`.

Na následujícím obrázku pořízeném v průběhu aktivity obou virtuálních uživatelů lze vidět ukazatel transakční rychlosti. Lze tedy prohlásit, že předběžným otestováním je zjištěno bezchybné připojení.



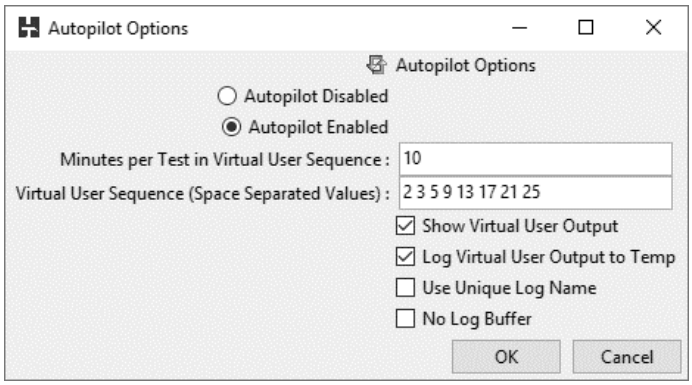
Obrázek 20 Ukazatel transakční rychlosti v předběžném testování PostgreSQL (snímek obrazovky)

4.3.9 **Měření výkonnostního profilu PostgreSQL databáze**

Po úspěšném otestování konfigurace přichází samotné měření výkonnostního profilu databázového systému. K měření je opět použita funkce Autopilot.

V možnostech „Driver Script“, je většina parametrů ponechána ve výchozím nastavení. Rozdílné nastavení spočívá ve změně druhu scriptu na „Timed Test Driver Script“ a parametr „Exit on MySQL Error“ je nastaven na FALSE (vypnuto).

Na obrázku níže lez vidět nastavení autopilota. Význam jednotlivých parametrů je již popsán v kapitole 4.3.6., kde se testovalo MySQL.

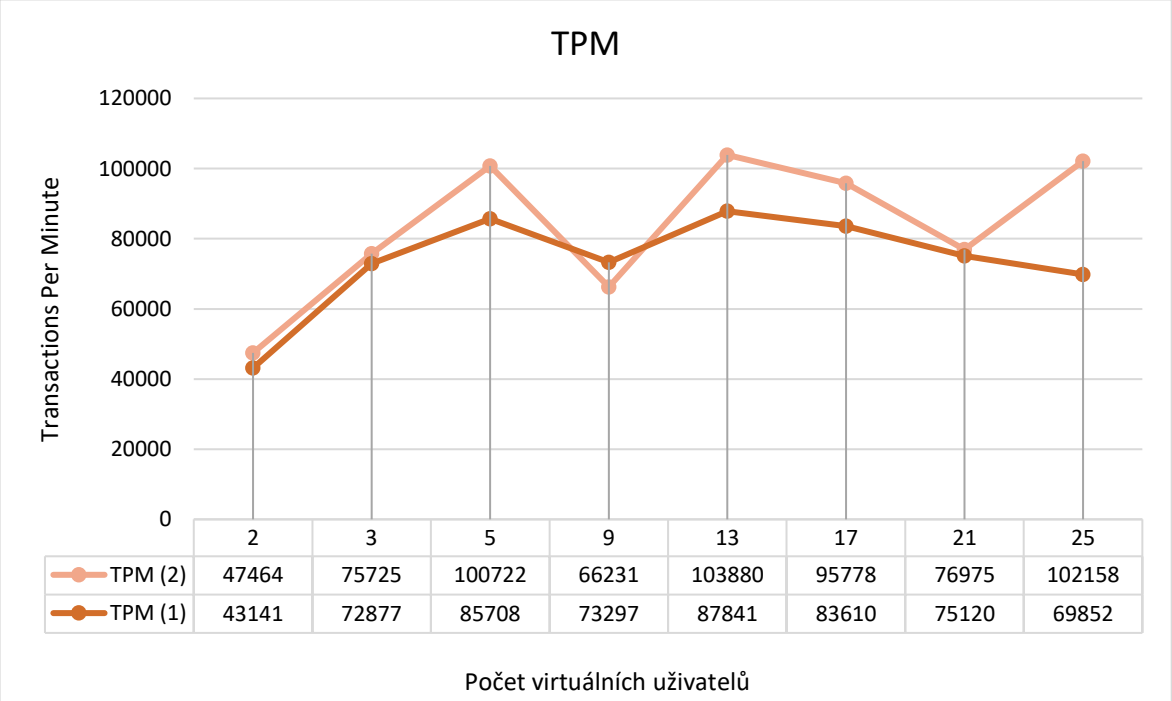


Obrázek 21 Nastavení Autopilota v HammerDB pro PostgreSQL (snímek obrazovky)

Po potvrzení nastavení a spuštění měření, zahajuje autopilot svou činnost. Po dokončení jsou výsledky zaznamenány a uloženy.

4.3.10 Výsledky měření autopilotem pro PostgreSQL

První výsledek měření lze vidět na následujícím grafu. Výsledky jsou zpracovány v MS Excel do podoby grafů.



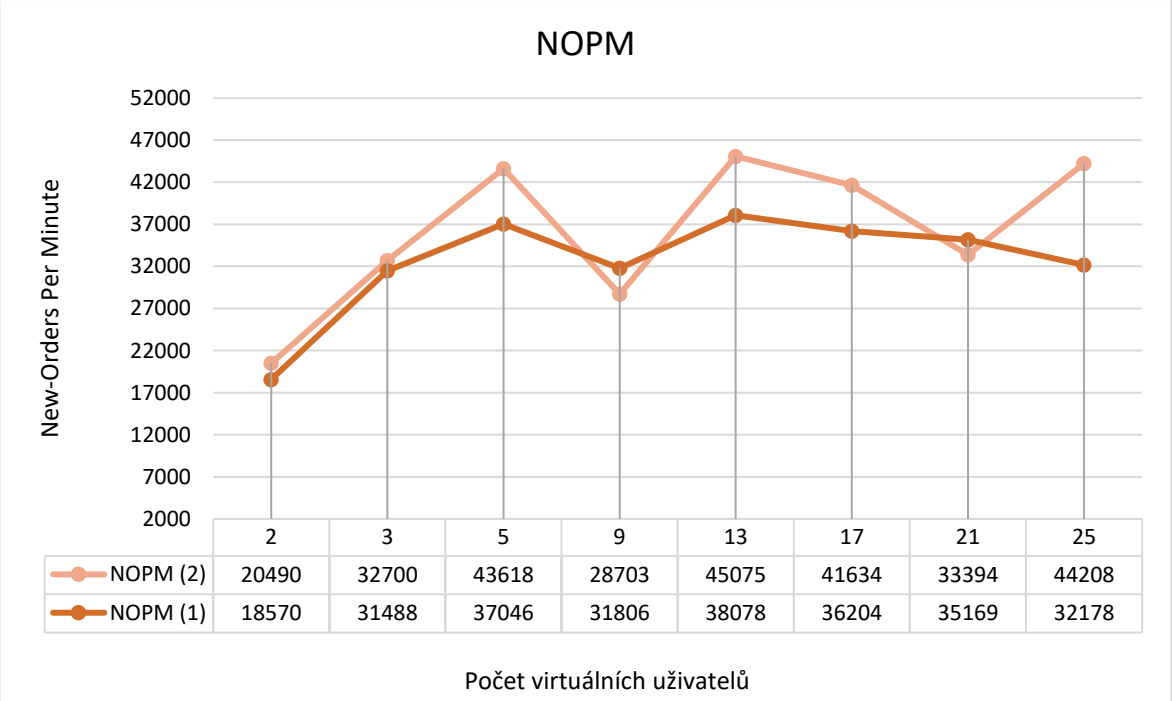
	2	3	5	9	13	17	21	25
TPM (2)	47464	75725	100722	66231	103880	95778	76975	102158
TPM (1)	43141	72877	85708	73297	87841	83610	75120	69852

Obrázek 22 Graf výsledků měřením TPM pro PostgreSQL (vlastní tvorba)

Tento graf, stejně jako u interpretace grafu TPM u MySQL, znázorňuje výsledek měření výkonnostního profilu prováděného v sekvenci s různými počty virtuálních uživatelů v intervalech 10 minut proti databázi PostgreSQL 10.2. Přitom každý virtuální uživatel provede 1 000 000 transakcí vybíraných náhodně z množiny druhů transakcí.

V případě PostgreSQL jsou výsledky překvapující. Měření bylo provedeno dvakrát, jednotlivé výsledky měření představují znázorněné řady „TPM (1)“ a „TPM (2)“. Důvodem dvou měření bylo to, že výsledky neodpovídají očekávanému výsledku. Za očekávané výsledky si lze představit podobné výsledky jako z měření MySQL. V případě PostgreSQL jsou na první pohled viditelné značné výkyvy ve výkonu. Do pěti připojených uživatelů TPM roste, ale zvláštní je hodnota při devíti uživatelích, kdy dochází k ráznému poklesu. Další zvláštností je hodnota TPM při 25 uživatelích, kdy v druhém měření TPM (2) opět roste.

Podobné chování lze pozorovat i při měření NOPM, viz. graf níže.



	2	3	5	9	13	17	21	25
NOPM (2)	20490	32700	43618	28703	45075	41634	33394	44208
NOPM (1)	18570	31488	37046	31806	38078	36204	35169	32178

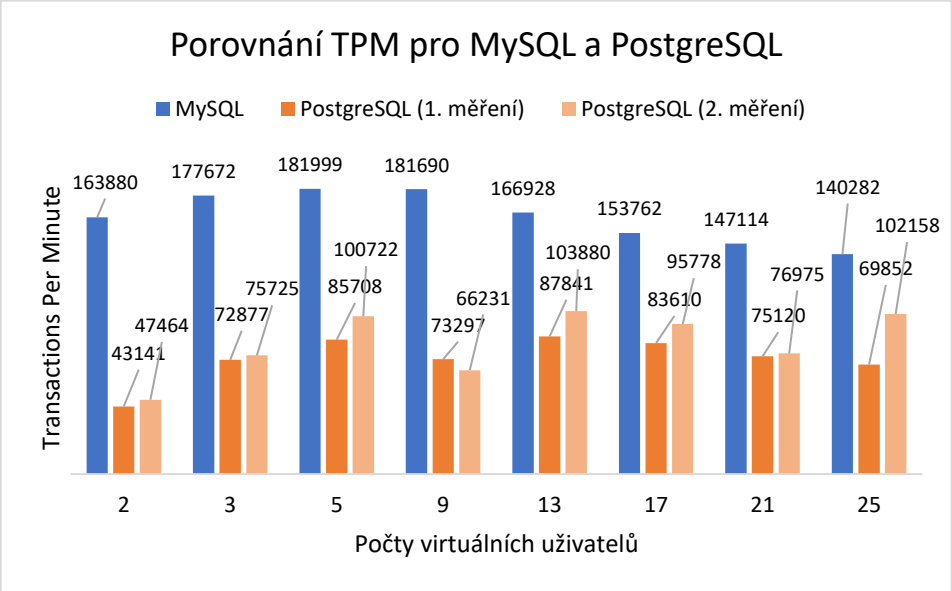
Obrázek 23 Graf výsledků měřením NOPM pro PostgreSQL (vlastní tvorba)

Při pokusu o zjištění důvodu, proč PostgreSQL vykazuje toto zvláštní chování s dokumentací HammerDB, je zjištěno, že důvodem může být nesprávná konfigurace operačního systému a hardwaru. Z hardwaru to může být konkrétně konfigurace CPU, paměti, I/O nebo sítě. (HammerDB, 2017c)

Měření probíhalo celkem 1h 20min 49s.

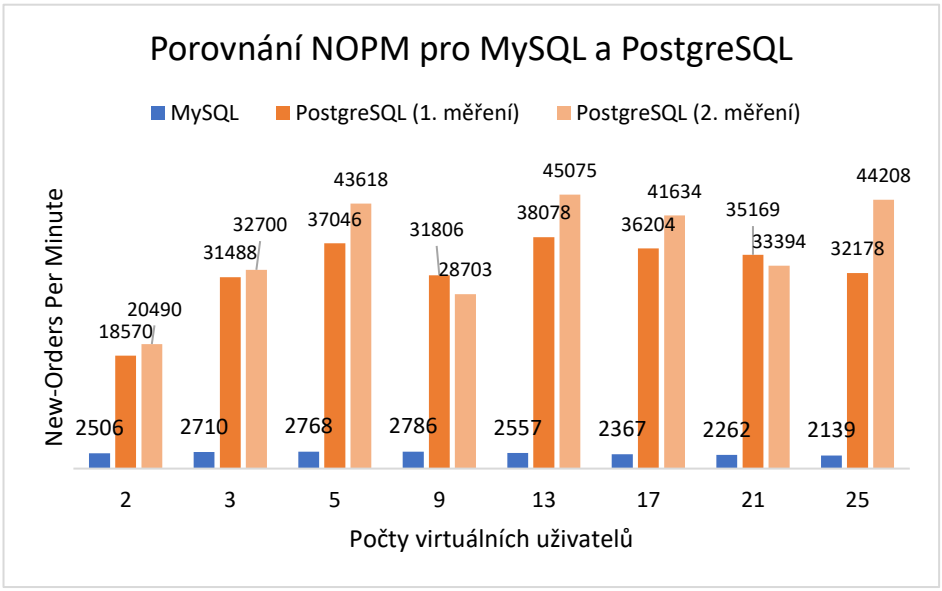
4.3.11 Porovnání výsledků měření pro MySQL a PostgreSQL

Na následujícím grafu, lze vidět porovnání změřených hodnot TPM z předchozích kapitol. Lze vidět, že hodnoty TPM jsou pro MySQL znatelně vyšší, než je tomu u PostgreSQL. Pro PostgreSQL bylo TPM měřeno dvakrát z důvodů, které již byly popsány dříve.



Obrázek 24 Porovnání TPM pro MySQL a PostgreSQL (vlastní tvorba)

Na dalším grafu, lze vidět opět porovnání změřených hodnot z předchozích kapitol, ovšem tentokrát pro hodnotu NOPM.



Obrázek 25 Porovnání NOPM pro MySQL a PostgreSQL (vlastní tvorba)

Hodnoty NOPM jsou v případě MySQL znatelně menší, než je tomu u PostgreSQL.

4.4 Výběr optimálního databázového systému

Na základě výsledků měření a získaných poznatků z předchozích kapitol lze určit optimální databázový systém. K tomu je použita vícekriteriální analýza variant (dále jen VAV). Varianty jsou zde dvě, a to MySQL a PostgreSQL. Předpokladem pro použití VAV je určení kritérií, což jsou hlediska, ze kterých jsou varianty posuzovány. Kritériím se poté přiřadí váhy, které lze určit pomocí Saatyho metody.

Jsou vybrána tato kritéria:

- NOPM (New-Orders Per Minute)
- TPM (Transactions Per Minute)
- Velikost (náročnost vygenerované databáze „tpcc“ na uložiště)
- Doba (doba měření)

Hodnoty kritéria NOPM a TPM jsou vyjádřeny jako aritmetický průměr ze souboru získaných dat. Pro PostgreSQL jsou tyto soubory dva, z důvodů dvou měření.

Velikost, respektive náročnost vygenerované databáze tpcc na uložiště, byla zjištěna z administrátorských nástrojů MySQL Workbench a pgAdmin 4 (pro PostgreSQL). Hodnota je vyjádřena v jednotkách GB.

Doba měření byla zjištěna z HammerDB, hodnoty jsou převedeny na jednotku sekund.

Všechny kritéria a jejich hodnoty jsou zpřehledněny v následující tabulce.

	NOPM	**TPM**	**Velikost [GB]**	**Doba [s]**
MySQL	2512	164166	18,8	5040
PostgreSQL	34398	78774	25	4849

Tabulka 5 Přehled kritérií a jejich hodnot (zdroj: autor)

TPM i NOPM jsou maximalizační kritéria – nejlepší hodnoty mají nejvyšší hodnoty. Velikost a Doba jsou minimalizační hodnoty – nejlepší hodnoty mají nejmenší hodnoty. Před hodnocením je potřeba všechny kritéria převést na jeden typ – maximalizační. To se provede tak, že se od největšího čísla ve sloupci odečtou ostatní hodnoty ve sloupci. Následující tabulka má převedená kritéria na maximalizační z tabulky 5.

	NOPM	**TPM**	**Velikost [GB]**	**Doba [s]**
MySQL	2512	164166	6,2	0
PostgreSQL	34398	78774	0	191

Tabulka 6 Převod na maximalizační kritéria (zdroj: autor)

K určení vah jednotlivých kritérií je použita metoda kvantitativního párového srovnávání neboli Saatyho metoda. Pro každou dvojici kritérií se určuje velikost preference podle Saatyho bodové stupnice. Výběr kritérií i určení preferencí, je založen na konzultaci s databázovým odborníkem z praxe.

Na následující tabulce lze vidět Saatyho matici s vypočítaným geometrickým průměrem a dopočítanými váhami.

	NOPM	TPM	Velikost	Doba	Geometrický průměr	Vážený GP	Váha
NOPM	1	2	5	8	2,990697562	0,508916109	**50,89 %**
TPM	1/2	1	4	7	1,93433642	0,329158982	**32,92 %**
Velikost	1/5	1/4	1	5	0,707106781	0,120325785	**12,03 %**
Doba	1/8	1/7	1/5	1	0,244461511	0,041599125	**4,16 %**
					Suma: 5,876602275	1	100 %

Tabulka 7 Saatyho matice (zdroj: autor)

Pro stanovení pořadí variant je použita Metoda váženého součtu (WSA). Definičním oborem funkce je interval od nejhorší hodnoty po nejlepší hodnotu. Pro každou hodnotu z tabulky č.6 se vypočítá dílčí užitek. Pro dílčí užitek u_{ij} hodnoty y_{ij} platí

$$u_{ij} = \frac{y_{ij} - d_j}{h_j - d_j}$$

Nejhorší hodnota se značí d_j a nejlepší hodnota h_j. V následující tabulce lze vidět výsledky jednotlivých dílčích užitků.

	NOPM	TPM	Velikost	Doba
MySQL	0	1	1	0
PostgreSQL	1	0	0	1

Tabulka 8 Výsledky dílčích užitků (zdroj: autor)

Nyní se pro jednotlivé varianty dopočítají celková užitky u podle vztahu

$$u = \sum_{j=1}^{n} w_j u_{ij}$$

Kde w_j jsou vážené geometrické průměry z tabulky č.7 a u_{ij} jsou jednotlivé dílčí užitky z tabulky č.8.

	NOPM	TPM	Velikost	Doba	Užitek	Pořadí
MySQL	0	1	1	0	0,449	**2.**
PostgreSQL	1	0	0	1	0,551	**1.**
Vážený GP:	0,508916109	0,329158982	0,120325785	0,041599125		

Tabulka 9 Výpočet celkového užitku variant a seřazení (zdroj: autor)

Výpočtem je identifikovaný optimální databázový systém. Tím je v tomto případě PostgreSQL, který má největší hodnotu užitku ***u***.

5 Zhodnocení výsledků

V praktické části práce bylo provedeno měření databázových systému MySQL 5.7 a PostgreSQL 10.2 v měřícím nástroji HammerDB. Oba databázové systémy i měřící nástroj jsou dostupné zdarma pod licencí open source.

Na začátku praktické části bylo specifikováno prostředí, ve kterém jsou později oba systémy nainstalovány a měřeny. Použité prostředí lze chápat jako osobní počítač v kategorii herních strojů, proto nelze toto prostřední srovnávat s výkonnými servery, které by byly jistě vhodnější pro oba databázové systémy, při použití v praxi. Ovšem pro zobecnění postupů měření databázových systémů je použitý stroj zcela dostačující.

V dalších krocích byly, pomocí nástroje HammerDB v obou systémech, vygenerovaná stejná schémata dle metodiky TPC-C, která představují abstrakci objednávkového systému společnosti, který zpracovává a dodává objednávky od zákazníkům. Společnost prodává 100 000 produktů, které uchovává ve svých 225 skladech. Každý sklad je propojený s 10 prodejnami a každá prodejna obsluhuje 3000 zákazníků. Zákazníci si objednávají produkty pomocí objednávek o různém počtu kusů. Tyto schémata měla různou náročnost na uložiště. V případě MySQL schéma zabralo 18,8 GB paměti SSD disku a v případě PostgreSQL 25 GB.

Poté bylo provedeno otestování funkčnosti měření, které proběhlo bez problému v obou případech. Když bylo otestování úspěšné, bylo provedeno měření pomocí funkce autopilot. Tato funkce připojovala virtuální uživatele v sekvenci počtu 2, 3, 5, 9, 13, 17, 21 a 25. Každý virtuální uživatel spustil 1 000 000 procedur (transakcí) po dobu 10 minut, něž se přešlo na další větší počet uživatelů ze sekvence.

Výsledkem tohoto měření byly hodnoty TPM (Transactions Per Minute), NOPM (New-Orders Per Minute) a byla zde i zjištěna doba měření – 1 h 24 min pro MySQL a 1 h 20 min 45 s pro PostgreSQL. V případě MySQL hodnoty TPM i NOPM rostly do 5 připojených virtuálních uživatelů, a poté začaly hodnoty klesat. V případě PostgreSQL hodnoty TPM i NOPM také rostly do 5 připojených virtuálních uživatelů, ale hodnoty rázně klesly při 9 uživatelích, poté hodnoty opět stouply a začaly postupně klesat až do 21 uživatelů. Byly provedeny dvě měření, právě z důvodu neočekávaného výsledku u 9 uživatelů. Ovšem v obou měřeních byl výsledek podobný, s tím rozdílem, že v druhém měření TPM i NOPM náhle vzrostlo při 25 uživatelích.

V závěru praktické části byl vybrán optimální databázový systém pomocí vícekriteriální analýzy variant. Byly zde vybrány kritéria – NOPM, TPM, náročnost databáze na uložiště (Velikost) a doba měření (Doba). Kritériím byly určeny váhy pomocí Saatyho metody. Výsledný užitek, který určil optimální systém, byl zjištěn pomocí metody váženého součtu (WSA).

Vítězem porovnání se stal PostgreSQL. Za předpokladu, že budou použity jiné metodiky měření, bude použit jiný stroj nebo jiná konfigurace hardware a databázového systému, pak se mohou výsledky lišit od výsledků této práce. Při jakékoliv změně, by mělo být provedeno nové měření, upraveno konkrétním novým podmínkám.

6 Závěr

Bakalářská práce si klade za cíl porovnat dva databázové systémy MySQL a PostgreSQL. Tento cíl byl proveden jak po stránce teoretické, tak i praktické. V teoretické části jsou popsány pojmy relační databáze a jazyk SQL, a jsou zde také charakterizovány MySQL i PostgreSQL. V praktické části bylo provedeno měření obou systémů pomocí měřícího nástroje HammerDB na operačním systému Windows 10. Měření bylo provedeno dle metodiky TPC-C, jejíž součástí je implementace databázového systému na konkrétním databázovém modelu.

Výstupy měření byly zpracovány pomocí vícekriteriální analýzy variant, konkrétně Saatyho metody pro zjištění vah jednotlivých kritérií a metody váženého součtu pro zjištění užitku obou variant. Optimální databázový systém byl identifikován podle největšího dosaženého užitku.

Přínosem této bakalářské práce je definování postupů při měření výkonnosti databázových systémů a seznámení s metodikami měření TPC. Obecně je hlavním smyslem měření databázových systémů určit, který systém je vhodnější k implementaci a k definování optimální konfigurace hardware a databáze, při které databáze v zátěži dosahuje nejvyššího výkonu.

Práci lze dále rozšířit o porovnání s dalšími databázovými systémy (Oracle, SQL Server, DB2 atd.) a odvozeninami (MariaDB, Percona atd.). Rozšiřovat a doplňovat lze zejména o další srovnání výkonu z metodik TPC, například TPC-H, TPC-E apod., které se liší metodikou testování, nebo použití zcela jiných metodik (například SPEC). Dále je možné měřit chování databázových systémů při různých konfiguracích nebo při použití výkonnějších strojů, určených pro serverové použití.

Citovaná literatura

1. AMAZON WEB SERVICES, , 2017. Amazon Aurora. *Amazon Web Services* [online]. Amazon Web Services [cit. 2017-11-01]. Dostupné z: https://aws.amazon.com/rds/aurora/
2. ANSI, , 2017. ISO/IEC JTC 1/SC 32 Publishes Updated SQL Database Language Standard. *ANSI: American National Standards Institute* [online]. Washington: ANSI [cit. 2017-09-15]. Dostupné z: https://www.ansi.org/
3. BELAID, , 2015. Introduction to PostgreSQL physical storage. *Rachidbelaid.com* [online]. Belaid [cit. 2017-11-03]. Dostupné z: http://rachbelaid.com/introduction-to-postgres-physical-storage/
4. CABRAL, Sheeri a Keith MURPHY, 2009. *MySQL administrator's bible*. Indianapolis, IN: Wiley Pub. ISBN 9780470506615.
5. ČÁPKA, David, 2017. 1. díl - MySQL krok za krokem: Úvod do MySQL a příprava prostředí. *ITnetwork.cz* [online]. ITnetwork [cit. 2017-11-15]. Dostupné z: https://www.itnetwork.cz/mysql/mysql-tutorial-uvod-a-priprava-prostredi/
6. DASH, Anil, 2010. Forking is a feature. *Anil Dash: A blog about making culture* [online]. New York City [cit. 2017-11-15]. Dostupné z: http://anildash.com/2010/09/forking-is-a-feature.html
7. DATA, , c1999-2017. *W3Schools Online Web Tutorials* [online]. Norway: Data [cit. 2017-09-15]. Dostupné z: https://www.w3schools.com/
8. DATABASE FRIENDS, , b.r. History of MySQL: MySQL Forks. *Database Friends* [online]. [cit. 2017-10-17]. Dostupné z: http://databasefriends.blogspot.cz/2014/02/history-of-mysql.html
9. ENTERPRISEDB CORPORATION, , 2017. EDB Postgres Advanced Server. *EnterpriseDB* [online]. EnterpriseDB Corporation [cit. 2017-11-01]. Dostupné z: https://www.enterprisedb.com/products/edb-postgres-platform/edb-postgres-advanced-server
10. FREE SOFTWARE FOUNDATION, , 1991. GNU General Public License v2.0. *GNU Project: Free Software Foundation* [online]. Boston: Free Software Foundation [cit. 2017-10-04]. Dostupné z: http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

11. GILFILLAN, Ian., 2003. *Myslíme v MySQL 4: knihovna programátora*. 1. vyd. Praha: Grada. Myslíme v--. ISBN 80-247-0661-X.
12. HAMMERDB, , 2017a. About. *HammerDB* [online]. [cit. 2018-02-11]. Dostupné z: http://www.hammerdb.com/about.html
13. HAMMERDB, , 2017b. Documentation. *HammerDB* [online]. [cit. 2018-02-11]. Dostupné z: http://www.hammerdb.com/document.html
14. HAMMERDB, , 2017c. *Introduction to Transactional (TPC-C) Testing* [online]. [cit. 2018-02-15]. Dostupné z: http://www.hammerdb.com/hammerdb_transactionintro.pdf
15. HAMMERDB, , 2017d. *MySQL OLTP Load Testing Guide* [online]. [cit. 2018-02-18]. Dostupné z: http://www.hammerdb.com/hammerdb_mysql_oltp.pdf
16. HOWE, , 2002. Database. *FOLDOC: Computing Dictionary* [online]. London: Howe [cit. 2017-09-14]. Dostupné z: http://foldoc.org/database
17. HOWE, , 2005. Database. *FOLDOC: Computing Dictionary* [online]. London: Howe [cit. 2017-09-14]. Dostupné z: http://foldoc.org/database
18. JAY A. KREIBICH., , 2010. *Using SQLite*. 1st ed. Sebastopol, CA: O'Reilly. ISBN 9780596521189.
19. KSIAZEK, Krzysztof, 2017. Comparing Oracle MySQL, Percona Server and MariaDB. *Severalnines* [online]. Ksiazek [cit. 2017-10-17]. Dostupné z: https://severalnines.com/blog/comparing-oracle-mysql-percona-server-and-mariadb
20. MARIADB, , 2017a. MariaDB versus MySQL: Compatibility. *MariaDB Knowledge Base* [online]. MariaDB [cit. 2017-10-17]. Dostupné z: https://mariadb.com/kb/en/library/mariadb-vs-mysql-compatibility/
21. MARIADB, , 2017b. About XtraDB. *MariaDB Knowledge Base* [online]. MariaDB [cit. 2017-10-17]. Dostupné z: https://mariadb.com/kb/en/library/about-xtradb/
22. MARIADB, , 2017c. Aria FAQ. *MariaDB Knowledge Base* [online]. MariaDB [cit. 2017-10-17]. Dostupné z: https://mariadb.com/kb/en/library/aria-faq/
23. OLDANYGROUP, , 2018. OLTP. *OldanyGroup* [online]. [cit. 2018-02-11]. Dostupné z: http://www.oldanygroup.cz/index-stranek-115/oltp/
24. OPENSSL SOFTWARE FOUNDATION, , c1999-2016. Welcome to OpenSSL!. *OpenSSL: Cryptography and SSL/TLS Toolkit* [online]. OpenSSL Software Foundation [cit. 2017-09-29]. Dostupné z: https://www.openssl.org/

25. ORACLE CORPORATION, , 2007. History of MySQL. *MySQL: MySQL 5.7 Reference Manual* [online]. Oracle Corporation [cit. 2017-09-27]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/history.html
26. ORACLE CORPORATION, , 2012. FOSS License Exception. *MySQL* [online]. Oracle Corporation [cit. 2017-10-04]. Dostupné z: https://www.mysql.com/about/legal/licensing/foss-exception/
27. ORACLE CORPORATION, , 2017a. What is MySQL?. *MySQL: MySQL 8.0 Reference Manual* [online]. Oracle Corporation [cit. 2017-09-21]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html
28. ORACLE CORPORATION, , 2017b. MySQL 8.0 FAQ: General. *MySQL: MySQL 8.0 Reference Manual* [online]. Oracle Corporation [cit. 2017-09-28]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/faqs-general.html
29. ORACLE CORPORATION, , 2017c. MySQL Enterprise Edition. *MySQL: MySQL 8.0 Reference Manual* [online]. Oracle Corporation [cit. 2017-09-29]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/mysql-enterprise.html
30. ORACLE CORPORATION, , 2017d. MySQL Editions. *MySQL* [online]. Oracle Corporation [cit. 2017-09-29]. Dostupné z: https://www.mysql.com/products/
31. ORACLE CORPORATION, , 2017e. Alternative Storage Engines. *MySQL: MySQL 5.7 Reference Manual* [online]. Oracle Corporation [cit. 2017-10-06]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/storage-engines.html
32. ORACLE CORPORATION, , 2017f. Introduction to InnoDB. *MySQL: MySQL 5.7 Reference Manual* [online]. Oracle Corporation [cit. 2017-10-11]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_consistent_read
33. ORACLE CORPORATION, , 2017g. Clustered and Secondary Indexes. *MySQL: MySQL 5.7 Reference Manual* [online]. Oracle Corporation [cit. 2017-10-11]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/innodb-index-types.html
34. ORACLE CORPORATION, , 2017h. The MyISAM Storage Engine. *MySQL: MySQL 5.7 Reference Manual* [online]. Oracle Corporation [cit. 2017-10-11]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html
35. ORACLE CORPORATION, , 2017i. MySQL Community Edition. *MySQL* [online]. [cit. 2017-11-15]. Dostupné z: https://www.mysql.com/products/community/
36. ORACLE CORPORATION, , c1993-2013. History of SQL. *Oracle® Database SQL Reference* [online]. [cit. 2017-09-14].

37. PERCONA LLC, , c2006-2017. Frequently Asked Questions. *Percona* [online]. Percona LLC [cit. 2017-10-18]. Dostupné z: https://www.percona.com/software/mysql-database/percona-server/faq
38. POSTGIS PROJECT STEERING COMMITTEE, , b.r. About PostGIS. *PostGIS* [online]. PostGIS Project Steering Committee [cit. 2017-11-01]. Dostupné z: http://postgis.net/
39. RIGGS, Simon a Hannu KROSING, 2010. *PostgreSQL 9 Admin Cookbook*. Birmingham: Packt Pub. ISBN 9781849510295.
40. SHELDON, Robert a Geoff MOES, 2005. *Beginning MySQL*. Indianapolis, IN: Wiley Pub. ISBN 9780764596575.
41. SCHNELLER, Daniel a Udo SCHWEDT, 2010. *MySQL admin cookbook 99 great recipes for mastering MySQL configuration and administration*. Birmingham, U.K: Packt Pub. ISBN 978-184-7197-979.
42. SMAGEN, , 2017. Future of storage. *PostgreSQL wiki* [online]. Smagen [cit. 2017-11-03]. Dostupné z: https://wiki.postgresql.org/wiki/Future_of_storage
43. STONEBRAKER, Michael a Lawrence ROWE, 1986. *The design of Postgres* [online]. University of California Berkeley: ACM [cit. 2017-11-20]. Dostupné z: http://db.cs.berkeley.edu/papers/ERL-M85-95.pdf
44. TAYLOR, Allen, 2010. *SQL for dummies*. 7th ed. Hoboken, NJ: Wiley Publishing. -- For dummies. ISBN 9780470593141.
45. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , 2007. Pod jakou licencí je PostgreSQL?. *PostgreSQL* [online]. PostgreSQL Global Development Group [cit. 2017-10-27]. Dostupné z: https://postgres.cz/wiki/1.03_Pod_jakou_licenc%C3%AD_je_PostgreSQL%3F
46. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , 2014. Slovník. *PostgreSQL* [online]. [cit. 2017-10-06]. Dostupné z: http://postgres.cz/wiki/Slovn%C3%ADk#ACID
47. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , 2016. Postgres-XL 9.5 R1 Released!. *PostgreSQL* [online]. The PostgreSQL Global Development Group [cit. 2017-11-01]. Dostupné z: https://www.postgresql.org/about/news/1662/
48. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , c1996-2017a. Window Functions. *PostgreSQL: Documentation: 9.1* [online]. The PostgreSQL Global

Development Group [cit. 2017-10-17]. Dostupné z: https://www.postgresql.org/docs/9.1/static/tutorial-window.html

49. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , c1996-2017b. About. *PostgreSQL* [online]. The PostgreSQL Global Development Group [cit. 2017-10-20]. Dostupné z: https://www.postgresql.org/about/
50. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , c1996-2017c. History. *PostgreSQL* [online]. The PostgreSQL Global Development Group [cit. 2017-10-25]. Dostupné z: https://www.postgresql.org/about/history/
51. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , c1996-2017d. Appendix E. Release Notes. *PostgreSQL* [online]. The PostgreSQL Global Development Group [cit. 2017-10-25]. Dostupné z: https://www.postgresql.org/docs/current/static/release.html
52. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , c1996-2017e. License. *PostgreSQL* [online]. The PostgreSQL Global Development Group [cit. 2017-10-27]. Dostupné z: https://www.postgresql.org/about/licence/
53. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , c1996-2017f. Downloads. *PostgreSQL* [online]. The PostgreSQL Global Development Group [cit. 2017-10-27]. Dostupné z: https://www.postgresql.org/download/
54. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, , c1996-2017g. Arrays. *PostgreSQL: Documentation: 9.1* [online]. [cit. 2017-11-20]. Dostupné z: https://www.postgresql.org/docs/9.1/static/arrays.html
55. TPC, , c2001-2018a. Frequently Asked Questions (FAQ). *TPC* [online]. [cit. 2018-03-05]. Dostupné z: http://www.tpc.org/information/about/faq-generic.asp
56. TPC, , c2001-2018b. Who We Are. *TPC* [online]. [cit. 2018-03-05]. Dostupné z: http://www.tpc.org/information/who/whoweare.asp
57. VAN EEDEN, Daniël, 2016. History Graphs about MySQL and forks. In: *Github* [online]. [cit. 2017-10-17]. Dostupné z: https://github.com/dveeden/mysql-history-graph
58. WILTON, Paul a John COLBY, 2005. *Beginning SQL*. Indianapolis, IN: Wiley. ISBN 9780764596322.

Seznam použitých zkratek

SQL	Structured Query Language	Strukturovaný dotazovací jazyk
DBMS	database-management system	Systém řízení báze dat (SŘBD)
ANSI	American National Standards Institute	Americký národní standardizační institut
ISO	International Organization for Standardization	Mezinárodní organizace pro normalizaci
IEC	International Electrotechnical Commission	Mezinárodní elektrotechnická komise
DDL	Data Definition Language	Jazyk pro definici dat
DML	Data Manipulation Language	Jazyk manipulace s daty
DCL	Data Control Language	Jazyk kontroly dat
TCL	Transaction Control Language	Jazyk kontroly transakcí
API	Application Programming Interface	Programové rozhraní aplikace
CPU	Central Processing Unit	Centrální procesorová jednotka
I/O	Input/Output	Vstup/Výstup
FOSS	Free and Open Source Software	Svobodný a otevřený software
MVCC	Muliti Version Concurrency Control	Kontrola souběžnosti s více verzemi
ACID	Atomic, Consistent, Isolated, Durable	Atomičnost, konzistence, izolace, trvanlivost
TPC	Transaction Processing Performance Council	
RAM	Random Access Memory	Paměť s náhodným přístupem
GPU	Graphic Processing Unit	Grafický procesor
HDD	Hard Drive Disk	Pevný disk
SSD	Solid-State Drive	
TPM	Transactions Per Minute	Transakce za minutu
NOPM	New-Order Per Minute	Nových objednávek za minutu

Seznam obrázků

Seznam tabulek