VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

MOŽNOSTI NÁVRHU DATABÁZE SQL V MATLABU A REALIZACE DATABÁZOVÉ STRUKTURY PRO UKLÁDÁNI VÝSLEDKŮ MĚŘENÍ

POSSIBILITIES OF SQL DATABASE DESIGN IN MATLAB AND REALIZATION OF DATABASE STRUCTURE FOR STORING MEASUREMENT RESULTS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Petr Michálek**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Martin Appel**

SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Petr Michálek**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Martin Appel**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Možnosti návrhu databáze SQL v MATLABU a realizace databázové struktury pro ukládáni výsledků měření

Stručná charakteristika problematiky úkolu:

Motivací práce bude ověřit možnosti dostupných nástrojů v programu MATLAB pro práci s databázovým systémem SQL. Databázový systém SQL je dlouholetým standardem pro práci s daty, avšak v rámci nakládání s daty z měření ho v našem výzkumném týmu nepoužíváme a riskujeme tak ztrátu nebo únik drahocenných dat vlivem poškození či ztráty datového disku.
Program MATLAB/Simulink je hojně používaným nástrojem pro vytváření či řízení simulací. Datové soubory nejsou malé a mají rozdílnou strukturu. Vyřešení vhodné cesty pro přenesení a ukládání dat do databázového systému je netriviální a značně přínosné.
Výsledkem práce bude příkladová studie a sada nástrojů pro správné a bezpečné zacházení s daty z měření pro snadné zálohování, vytváření verzí a sdílení dat v rámci výzkumného týmu.

Cíle bakalářské práce:

1) Proveďte průzkum trhu v oblasti nástrojů pro práci s databázovým systémem SQL.
2) Porovnejte různé databázové systémy a jejich implementaci do programu MATLAB.
3) Prozkoumejte možnosti práce s databázovým systémem SQL pomocí Database Toolbox a pomocí využití Phython nebo Java v Matlabu.
4) Vytvořte sadu nástrojů pro práci s databází v programu MATLAB.
5) Demonstrujte vytvořené nástroje na ukázkových příkladech.

Seznam doporučené literatury:

CORKE, Peter I. Robotics, vision and control: fundamental algorithms in MATLAB. Berlin: Springer, 2011. Springer tracts in advanced robotics, v. 73. ISBN 9783642201431.

VALÁŠEK, Michael. Mechatronika. Praha: České vysoké učení technické, 1995. ISBN 80-01-01276-X.

GREPL, Robert. Kinematika a dynamika mechatronických systémů. Brno: Akademické nakladatelství CERM, 2007. ISBN 978-80-214-3530-8.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato práce se zabývá možností vyžití databázových systémů SQL s programem MATLAB. Úkolem bylo provést průzkum na trhu v oblasti SQL databází, porovnat různé systémy a jejich implementaci. Cílem praktické části je potom vytvořit nástroje, které umožní pracovat s SQL databázemi prostřednictvím programu MATLAB, bez použití placeného Database Toolbox rozšíření. A tyto nástroje potom demonstrovat na ukázkovém příkladu.

KLÍČOVÁ SLOVA

Matlab, databáze, SQL, JDBC, připojení, data, Java, MySQL, Microsoft SQL Server, PostgreSQL

ABSTRACT

This thesis is describing the usage of SQL database systems with MATLAB. The objective is to do market research considering the use of SQL databases, compare those systems and their implementation. The aim is to create tools that would allow the SQL database to be operated using MATLAB without the usage of a paid Database Toolbox. The last goal is to showcase these tools.

KEYWORDS

Matlab, database, SQL, JDBC, connection, data, Java, MySQL, Microsoft SQL Server, PostgreSQL

MICHÁLEK, Petr. Možnosti návrhu databáze SQL v MATLABU a realizace databázové struktury pro ukládáni výsledků měření. Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/125190. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Martin Appel.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci na téma „Možnosti návrhu databáze SQL v Matlabu a realizace databázové struktury pro ukládání výsledků měření“ vypracoval samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který tvoří přílohu této práce.

V Brně 24. června 2020

Petr Michálek

PODĚKOVÁNÍ

Děkuji vedoucímu práce Ing. Martinu Appelovi za pomoc při vytváření této práce. Také děkuji své rodině za pomoc při gramatické kontrole práce.

Obsah

1 Úvod

S databází se určitě za svůj život setkal již každý. Ať už s jejím papírovým předchůdcem dnešních počítačových databází – kartotékou např. u svého lékaře, které počítačové databáze předčily svými mnohem menšími nároky na fyzické místo, rychlostí vyhledávání a možností k nim přistupovat odkudkoliv. Nebo jste si díky ní mohli objednat zboží z internetového obchodu. Případně může být, díky skoro 20 000 nových nabídek pracovních pozic měsíčně, které vyžadují znalost SQL, práce s nimi přímo náplní vašeho zaměstnání [1].

Program Matlab je na naší škole velmi často používaným nástrojem pro simulace a modelování, práci s daty a programování. Z hlediska nakládaní se získanými daty ze simulací nebo měření se SQL databáze jeví jako vhodný kandidát pro řešení, které umožní tato data ukládat, zálohovat a sdílet.

Jedním z cílů této práce je seznámit čitatele s trhem v oblasti nástrojů pro práci s SQL databázovými systémy. Databázových systémů je nepřeberné množství a liší se např. licencemi, pod kterými jsou vydány, podporou platforem a programovacích jazyků nebo maximální velikostí databáze a prvků v nich obsažených.

Společnost MathWorks, která vyvíjí program Matlab, nabízí své řešení pro interakci s databázemi v podobě „Database Toolbox“. Jedná se o placené rozšíření, které umožňuje s databázemi jednoduše pracovat i bez znalosti jazyka SQL. Matlab však také podporuje knihovny napsané v jiných programovacích jazycích jako je C/C++, Java, Python a .NET. Tato vlastnost by měla umožnit obejít nutnost dodatečných nákladů spojených s pořízením Database Toolbox.

V následujících kapitolách popíši situaci na trhu SQL systémů a jejich implementaci do programu Matlab. Dále práci s databázemi pomocí Database Toolbox a nástroje, které databáze zpřístupní i bez tohoto rozšíření.

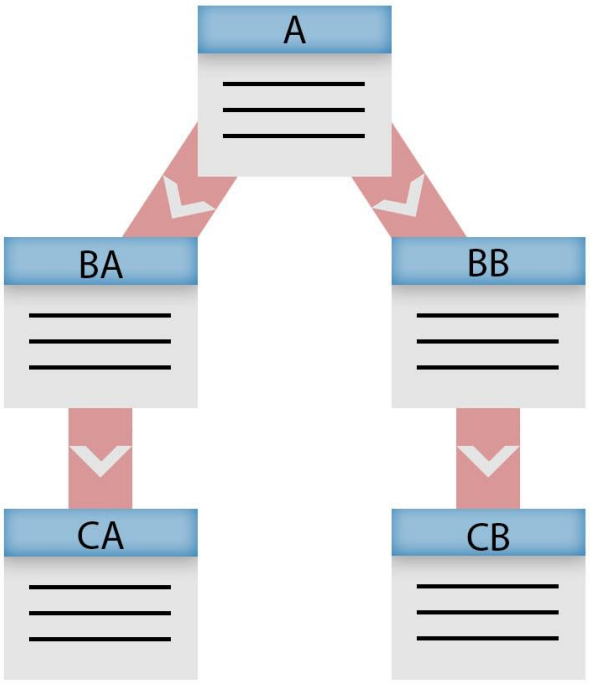
2 Průzkum trhu v oblasti databázových systémů

2.1 Historie databází

Databázové systémy hrály a hrají důležitou roli pro vývojáře softwaru i vlastníky firem. Díky nim je možné jednoduše sdílet data mezi mnoha uživateli a aplikacemi nebo psát poměrně složité programy, aniž by bylo nutné pro každou aplikaci znovu řešit, jak bude přistupovat k datům. V následujících odstavcích budou představeny typy databází, které byly zásadními v jejich vývoji [2].

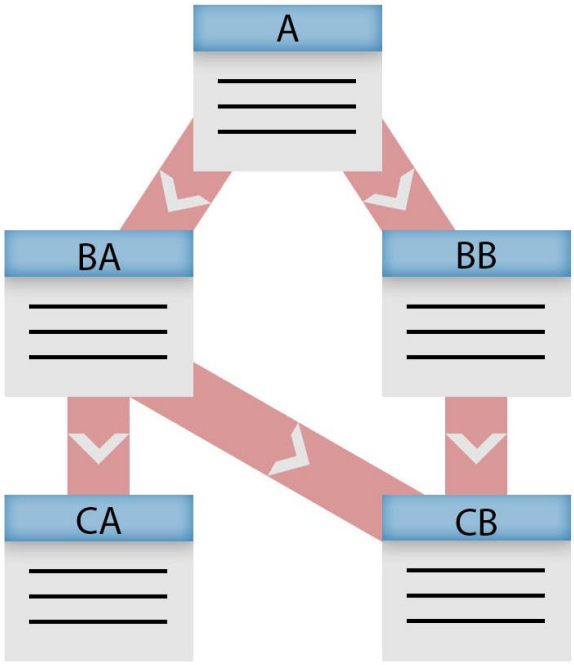
Éře relačních databází, kterým se bude práce věnovat, předcházely databáze hierarchické a síťové, které se začaly používat během 60. a 70. let 20. století. DBMS (Database Management System) je software, který umožňuje vytvářet, spravovat a udržovat databáze. DBMS používající hierarchické a síťové databáze se ve velkém prodávaly i v 90. letech a někde jsou používány dodnes [2] [3] [4].

V případě hierarchického modelu jsou data ve stromové struktuře a záznamy v ní jsou uzly, které mají mezi sebou vztah rodič-dítě. Podobně jako složky a soubory v počítači. Každý uzel může mít několik podřazených, ale žádný uzel nesmí mít více nadřazených uzlů. Výhodou takového řešení je vyšší rychlost při přistupování k informacím, protože cesty k datům jsou jasně definované. Nevýhodou je nízká flexibilita kvůli limitaci na právě jeden nadřazený uzel. Ilustrace hierarchické databáze je na obr. 1 [5].



Obrázek 1: Hierarchická databáze [5]

Síťový model má vylepšení v podobě možnosti přiřadit jednomu záznamu více „rodičů“. Oproti hierarchickým databázím tedy dokáže o něco lépe modelovat skutečné vztahy mezi daty. Výhodou je opět rychlost při čtení dat z databáze. Ilustrace síťové databáze je na obr. 2 [6].

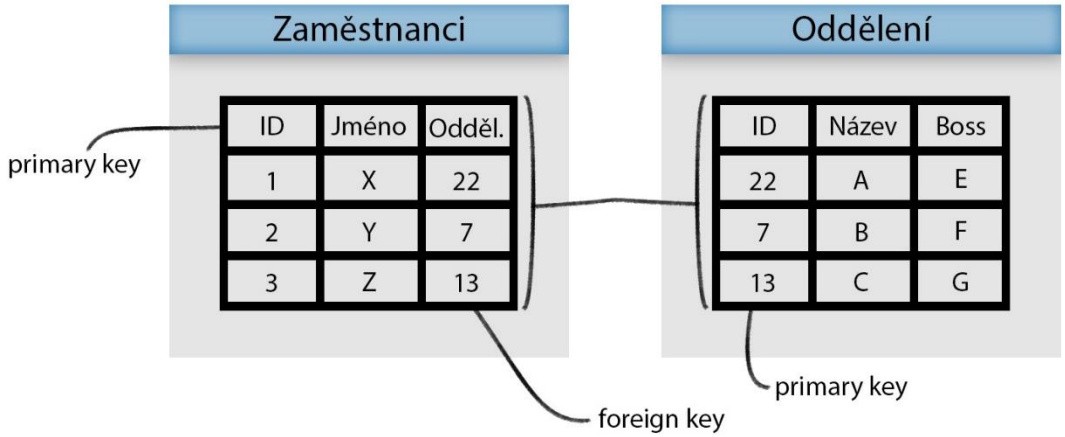


Obrázek 2: Síťová databáze [5]

Z tohoto období stojí za zmínku produkt IMS (Information Management System) od firmy IBM, který ve své dřívější verzi pojmenované ICS asistoval v NASA jako inventární systém pro Program Apollo. Jedná se právě o hierarchickou databázi. A dále sdružení CODASYL (Conference/Comittee on Data Systems Languages), které bylo založeno s cílem standardizovat databázové systémy a dosáhnout tak efektivnějšího zpracování dat. Vedlo také vývoj standardního programovacího jazyka. Výsledkem jejich snažení je programovací jazyk COBOL. Byl navržen tak, aby mohl fungoval na široké škále tehdejších sálových počítačů. Byl používán bankami a státními organizacemi. Systémy na něm založené jsou do dnešní doby v provozu, protože migrovat na novější řešení bývá nákladné. Aktuální situace vytvořila novou poptávku po lidech, kteří tento jazyk umí. Používá se totiž např. pro evidenci lidí bez zaměstnání žádajících o podporu a jejich počet prudce vzrostl kvůli pandemii koronaviru [7] [8] [9] [10] [11].

Relační model je přístup k uspořádání dat, kde jsou data reprezentována jako uspořádané n-tice seskupené do relací (vizuální reprezentací je tabulka). V relační databázi může být mnoho tabulek. Ty obsahují řádky s daty a předdefinované sloupce, které určují formát dat. Další charakteristikou relačního modelu jsou „klíče“. Jsou to speciální sloupce v tabulce, umožňující identifikovat řádky a vztahovat je k ostatním. Primární klíč (primary key) je sloupec, obsahující hodnotu, která je unikátní pro všechny řádky v tabulce, a lze je tak podle něj identifikovat. Na primární klíče se lze odkazovat s pomocí cizích klíčů (foreign key). Relační model definuje krom struktury také pravidla

pro manipulaci s daty a zajištění integrity dat. Ilustrace relační databáze je na obr. 3 [12] [13] [14] [15].

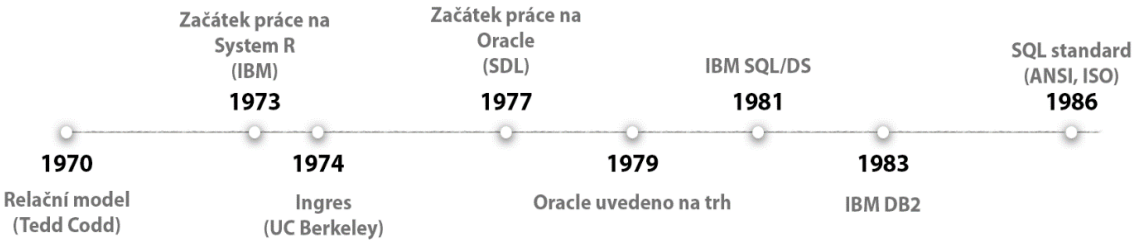


Obrázek 3: Relační databáze [16]

Relační databáze mají původ v americké firmě IBM. Donald Chamberlin a Raymond Boyce, dva noví zaměstnanci, kteří zrovna dosáhli Ph.D., se zúčastnili firemního setkání v roce 1972. Tam se poprvé dozvěděli o relačním modelu uložení dat přímo od jeho autora. Edgar Codd, který ho prezentoval, s ním přišel v roce 1969. Pro zmiňované zaměstnance IBM byl tento koncept fascinující, a začali pro sebe navzájem vymýšlet výzvy týkající se databází. Jedna z takových byla například zjistit, kteří zaměstnanci ve firmě vydělávají více než jejich manažer. Řešení takového úkolu by vyžadovalo složitý program v jazyce COBOL, ale s použitím konceptu Edgara Codda bylo řešení velmi jednoduché. Začali tedy pracovat na zpřístupnění tohoto konceptu pro běžné uživatele. Strávili rok experimentováním s návrhy jazyků. Cílem bylo navrhnout jazyk, který bude srozumitelný i bez pokročilých znalostí matematiky a programování. Prvního pokusu s názvem Square zanechali po tom, co se v roce 1973 přesunuli na projekt IBM s názvem System R. Jednalo se o experimentální databázový systém, který byl vytvořen za účelem předvedení výhod relačního modelu dat. Zároveň měl ale zachovat důležité vlastnosti a funkce jeho nerelačních předchůdců, vyžadované pro skutečné každodenní použití. System R byl zkušebně nasazen u několika zákazníků IBM. Název nového jazyka určeného pro tento projekt byl Sequel (Structured English Query Languge), ovšem kvůli problémům s ochrannou známkou a snaze ho odlišit od jeho raných verzí byl v roce 1977 zkrácen pouze na SQL (Structured Query Language). Jazyk SQL byl přijat jako standard organizacemi ISO a ANSI v roce 1986 [3] [12] [13] [17] [18] [19] [20] [21] [22].

Mezi další významná jména z historie relačních databází patří Ingres a Oracle. První z jmenovaných začal vznikat na univerzitě UC Berkeley po tom, co někteří zaměstnanci IBM publikovali dokumenty popisující System R. Funkční prototyp byl hotový v roce 1974. Ingres byl vydán pod BSD licencí a ovlivnil celou řadu dalších produktů, zejména Microsoft SQL Server, Sybase a Postgres. Oracle byl první komerční databázový systém používající jazyk SQL. Společnost, která ho vyvíjela, nesla jméno SDL a byla založena v roce 1977, o dva roky později byl Oracle vydán. Vzhledem k tomu, že IBM na trhu již mělo úspěšný databázový software (IMS), si s vydáním relačního

databázového systému dávalo na čas. Jejich první komerční databázový software používající relační model byl IBM SQL/DS, který vyšel až v roce 1981. Oracle tento časový náskok zvýhodnil a v roce 1987 se stalo největší společností v oblasti databázových systémů. Časová osa s významnými událostmi je na obr. 4 [19] [21] [22] [23] [24].



Obrázek 4: Časová osa vývoje relačních databází

Ještě novějším typem databází jsou databáze objektové. Data a kód byly běžně oddělené, ale v objektově orientovaném modelu jsou spojeny do jednoho nerozdělitelného objektu. Data jsou zde tedy reprezentována jako objekty, jejichž aplikace by měla být konzistentní s používáním objektů v objektově orientovaném programování. Tento koncept se ve světě databázových systémů vynořil během 80. let minulého století. Jejich koncem následovala i první vlna komerčních produktů. Ovšem opravdový růst tohoto segmentu trhu následoval až v letech 90 [5] [25] [26].

Podobně, jako byla pro relační databáze velmi důležitá publikace E. Codda, mají i objektově orientované databáze svůj stěžejní kus literatury. Ten byl vydán v roce 1995. Podílel se na něm Malcolm Atkinson a několik dalších autorů z různých univerzit. Definuje objektově orientovaný databázový systém. A popisuje důležité charakteristiky, které onen systém musí mít, aby se tak mohl nazývat, ale i nepovinné tipy pro zlepšení takového systému. Do historie objektových databází se zapsal také rok 2004. Ten byl důležitým vydáním systému Db4o, protože se jedná o open source projekt, který tak přispěje k rozšíření objektových databází. Oproti relačním se objektově orientované databáze používají pro nakládání s daleko složitějšími typy dat. Aplikují se např. pro CAD, CAM a CASE systémy. Kde v případě CAD systémů umožňují kontrolu nad revizemi výkresů složitých součástí (letecký průmysl, elektronika) a jednodušší kooperaci více pracovníků. Mezi další výhody se řadí lepší výkon a modelování vztahů z reálného světa mezi daty. Velkou nevýhodou je absence standardů stejného kalibru jako nabízí relační systémy. Např. neexistuje standardní jazyk jako je SQL pro relační databáze [25] [26] [27] [28].

2.2 Současný stav trhu

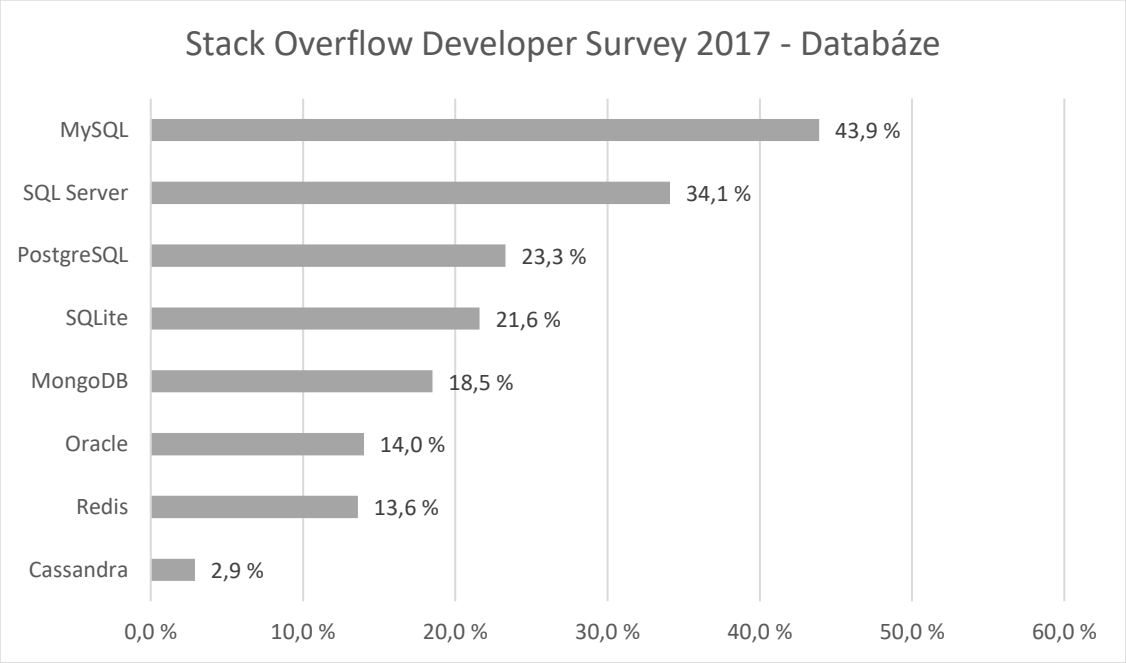
V současnosti trhu dominují SQL databáze, kterými byly až na pár výjimek nahrazeny zastaralé databáze hierarchické. Přestože bylo předpokládáno, že objektově orientované databáze začnou nahrazovat relační, neděje se to z důvodu vysokých nákladů a vývoje stávajících relačních systémů, který přinesl objektově-relační mapování. Objektově orientované databáze tak jako produkty spíše pouze doplňují relační [4] [29].

Na trhu existuje velké množství relačního SW prodávaných s různými obchodními modely. Přímý podíl jednotlivých hráčů na trhu lze pouze odhadovat. Ne všichni publikují počty prodaných licencí nebo předplatných. K tomu, jak relevantní jednotlivé SW jsou, se tedy lze dopracovat pomocí různých průzkumů. Budou vybrány některé zdroje a bude popsána jejich metodika. Zdroje jsou seřazené sestupně podle jejich spolehlivosti.

2.2.1 Stack Overflow

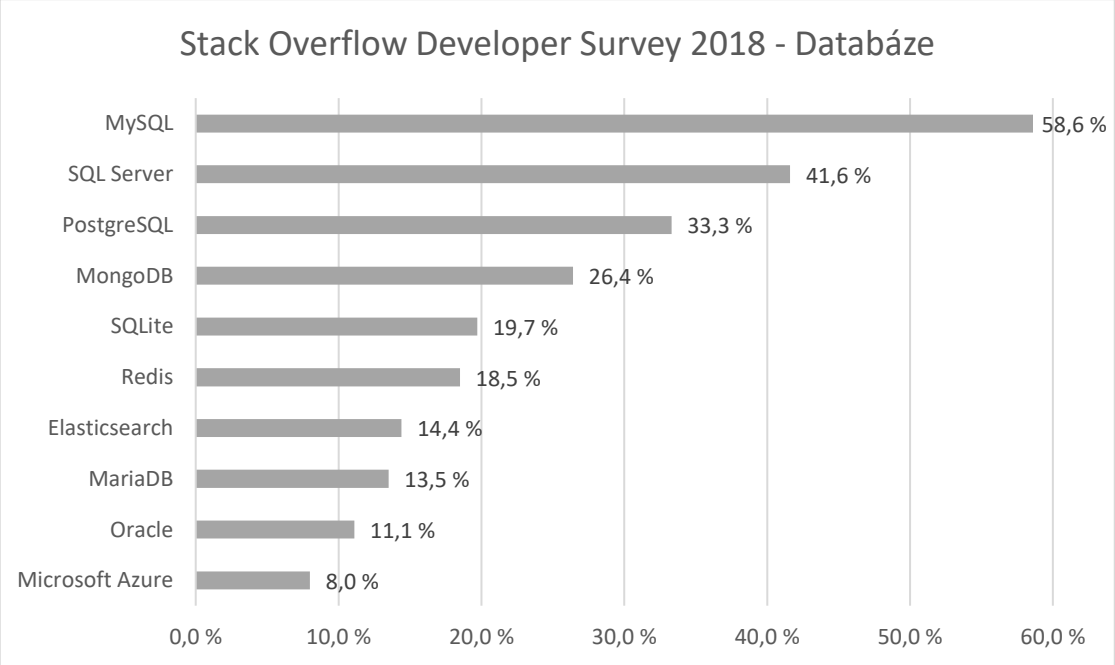
Stack Overflow je webová stránka určená pro programátory. Je na ní možné pokládat dotazy, které se místní komunita pokusí zodpovědět. Každý rok jsou prováděny průzkumy týkající se oblíbených technologií a pracovních příležitostí jejich komunity. Průzkum probíhá formou dotazníku, který vyplňují desetitisíce profesionálních vývojářů po celém světě. Od roku 2017 je součástí dotazníku i otázka zaměřená na konkrétní databázový systém, se kterým tazatelé pracují. Data ze Stack Overflow se jeví jako důvěryhodná. Je dobře popsána metodika získávání výsledků a jsou vyřazeny odpovědi od účastníků, nevěnujících dotazníku dostatek času. A také díky možnosti zobrazit si výsledky pouze od profesionálních programátorů místo celé komunity by měla dobře odrážet skutečnost [30] [31].

V roce 2017 byla do průzkumu otázka na používané databázové systémy zařazena poprvé. Tazatelé měli vybrat všechny systémy, které používají. Odpovídalo 27 612 programátorů z povolání. Výsledky jsou v grafu 1 [31].



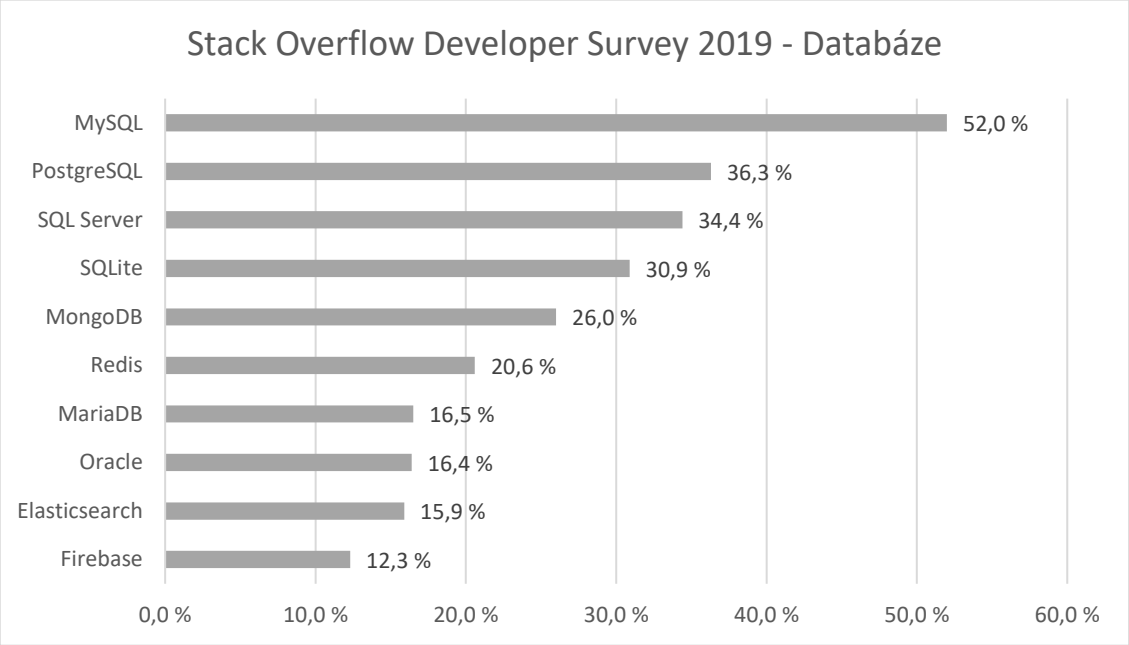
Graf 1: Stack Overflow 2017 [31]

O rok později odpovídalo na stejnou otázku hned 62 299 programátorů. Výsledky jsou omezeny na 10 nejpoužívanějších systémů a jsou v grafu 2. MySQL a PostgreSQL zaznamenaly růst, to ovšem může být způsobeno i výrazně větším počtem odpovědí [32].



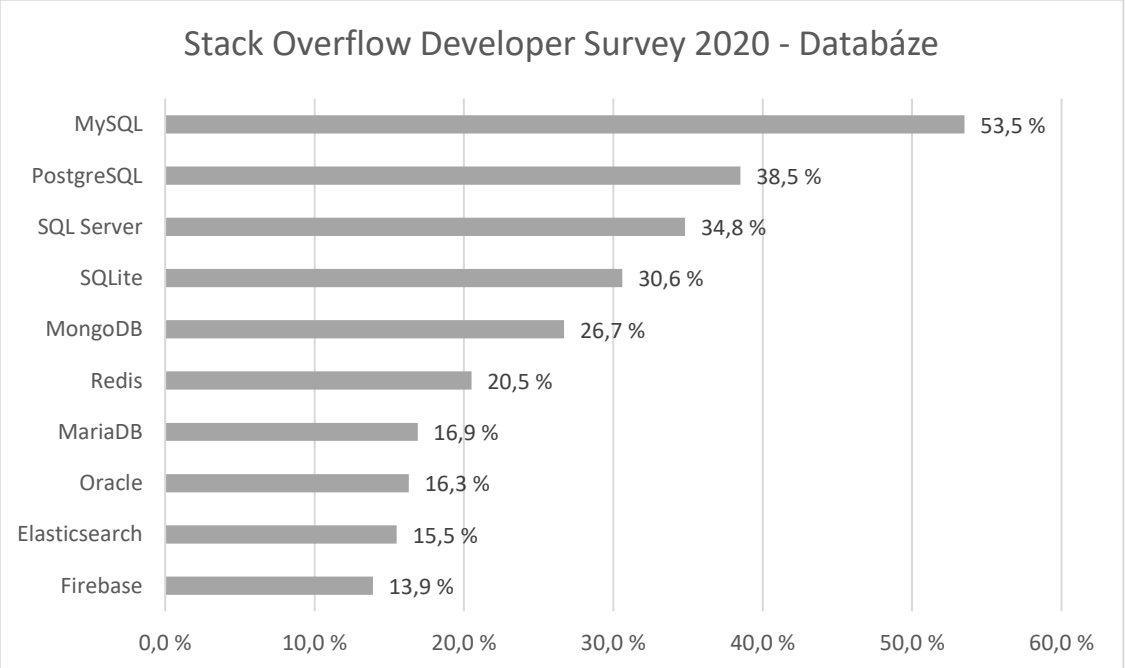
Graf 2: Stack Overflow 2018 [32]

Výsledky z roku 2019 (graf 3) tvoří 64 077 odpovědí. PostgreSQL zaznamenalo další růst a sesadilo Microsoft SQL Server z druhého místa. SQLite se také těší výrazně vyšší popularitě [33].



Graf 3: Stack Overflow 2019 [33]

Rok 2020 (graf 4**)** se nenese ve znamení velkých změn a výsledky vypadají skoro totožně jako v roce předešlém. Tento rok otázku zodpovědělo 41 811 vývojářů [34].



Graf 4: Stack Overflow 2020 [34]

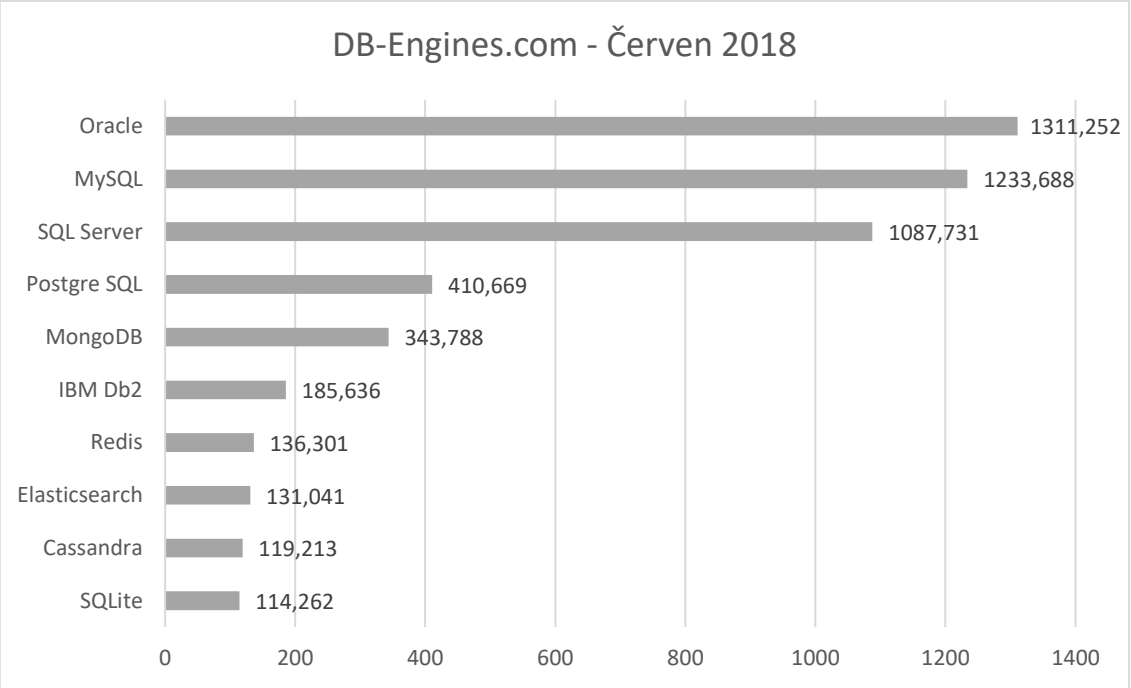
Dle výsledků ze Stack Overflow jsou mezi vývojáři tři konzistentně nejpoužívanější databázové systémy: MySQL, PostgreSQL, Microsoft SQL Server. Získaná data jsou důvěryhodná – pracují se vzorkem čítající desetitisíce profesionálních vývojářů.

2.2.2 DB-Engines.com

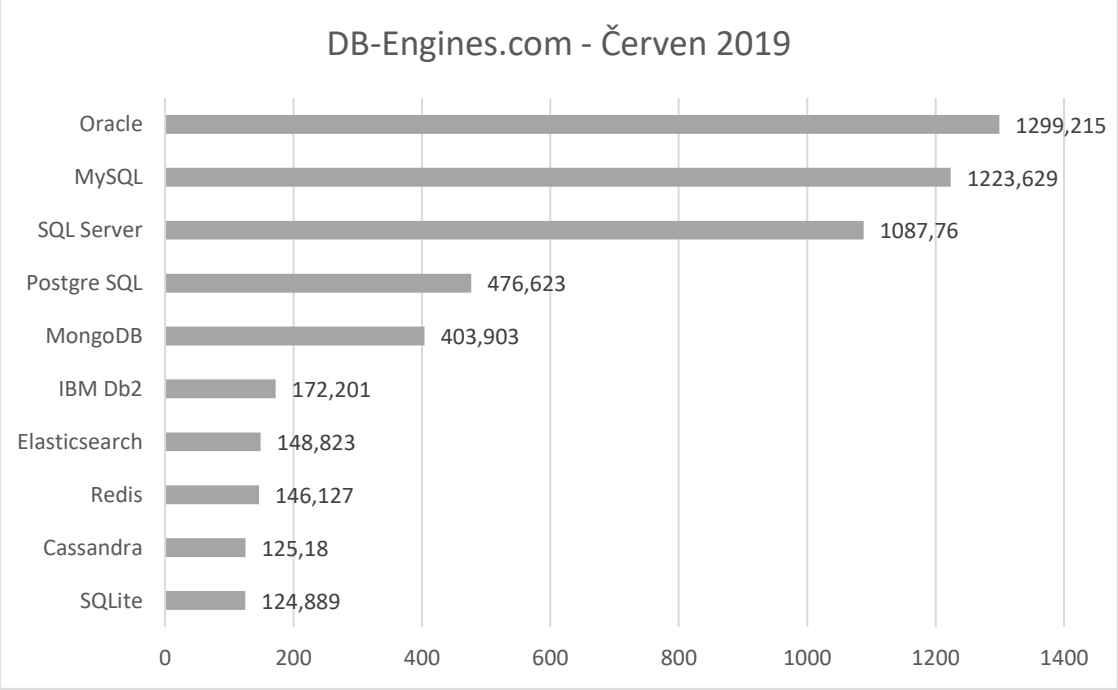
Dalším zajímavým zdrojem dat o popularitě databází je stránka DB-Engines. Jedná se o seznam nejpopulárnějších databází. Každá jednotlivá databáze zde obdrží skóre, vypočítané na základě několika parametrů. Jedná se o následující charakteristiky [35]:

- Počet výsledků na daný systém při vyhledávání pomocí Google, Bing a Yandex vyhledávačů. Počítá se tak, kolikrát byla databáze zmíněna. Aby bylo zajištěno, že se budou počítat pouze relevantní výsledky, je vyhledáváno jméno databáze společně se slovem „database“.
- Zájem o daný systém dle frekvence vyhledávání v Google Trends. Data z předchozího bodu sice vypovídají o tom, kolikrát je na webu některý databázový systém zmíněn, ovšem to nám neposkytuje náhled do toho, jak často jsou tyto informace vyhledávány. Pomocí Google Trends, kde je analyzována popularita hledaných termínů a souvisejících témat naskrz různými jazyky a regiony, je možné tato data získat.
- Množství nabídek pracovních pozic, které zmiňují daný systém. Jsou použita data z pracovních portálů Indeed a Simply Hired.
- Frekvence diskuse spjaté s danými systémy. Pro tuto kategorii jsou data získána ze stránek Stack Overflow a DBA Stack Exchange. Zde můžou uživatelé klást dotazy týkající se programování nebo v případě DBA Stack Exchange přímo databází.
- Počet profesionálů, kteří zmiňují dané systémy na svých profilech/portfoliích. Týká se to profesních sociálních sítí LinkedIn a Upwork.
- Relevance na sociálních médiích dle počtu příspěvků zmiňujících daný systém na sociální síti Twitter.

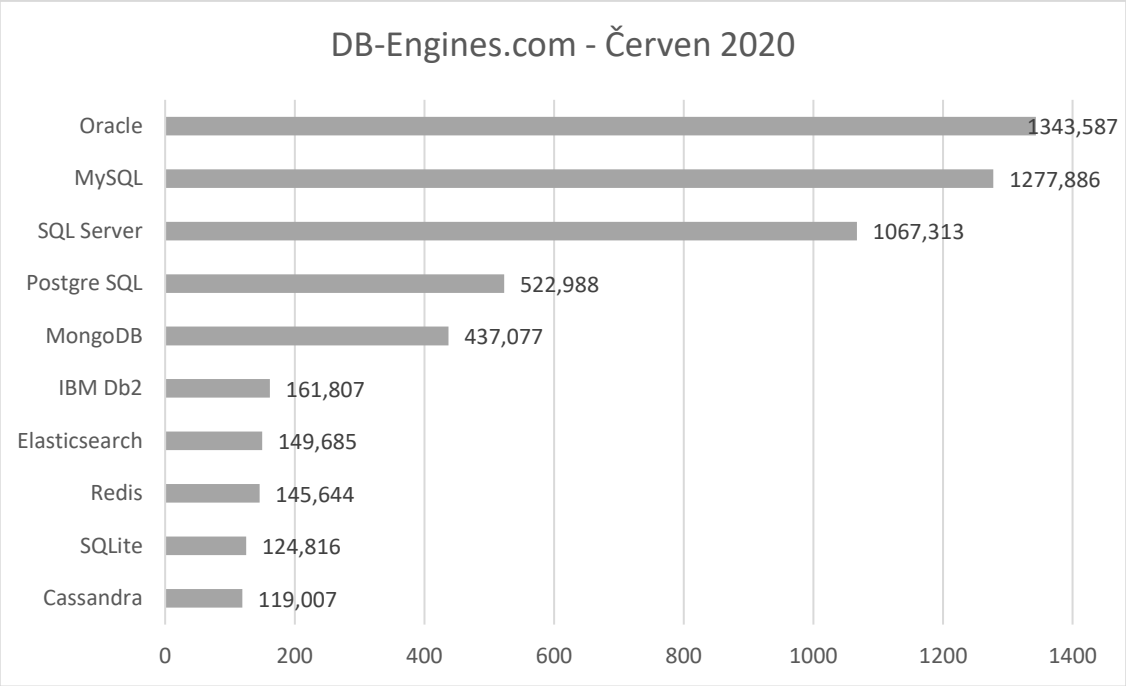
Na základě těchto faktorů je potom vypočteno skóre, které vypovídá o popularitě jednotlivých systémů. Jednotlivé parametry jsou standardizovány a průměrovány. Když je výsledné skóre jednoho systému např. dvakrát vyšší než systému druhého, znamená to, že je v průměru dvakrát populárnější. Hodnocení probíhá každý měsíc [35].



Graf 5: DB-Engines 6/2018 [36] [37]



Graf 6: DB-Engines 6/2019 [36] [37]

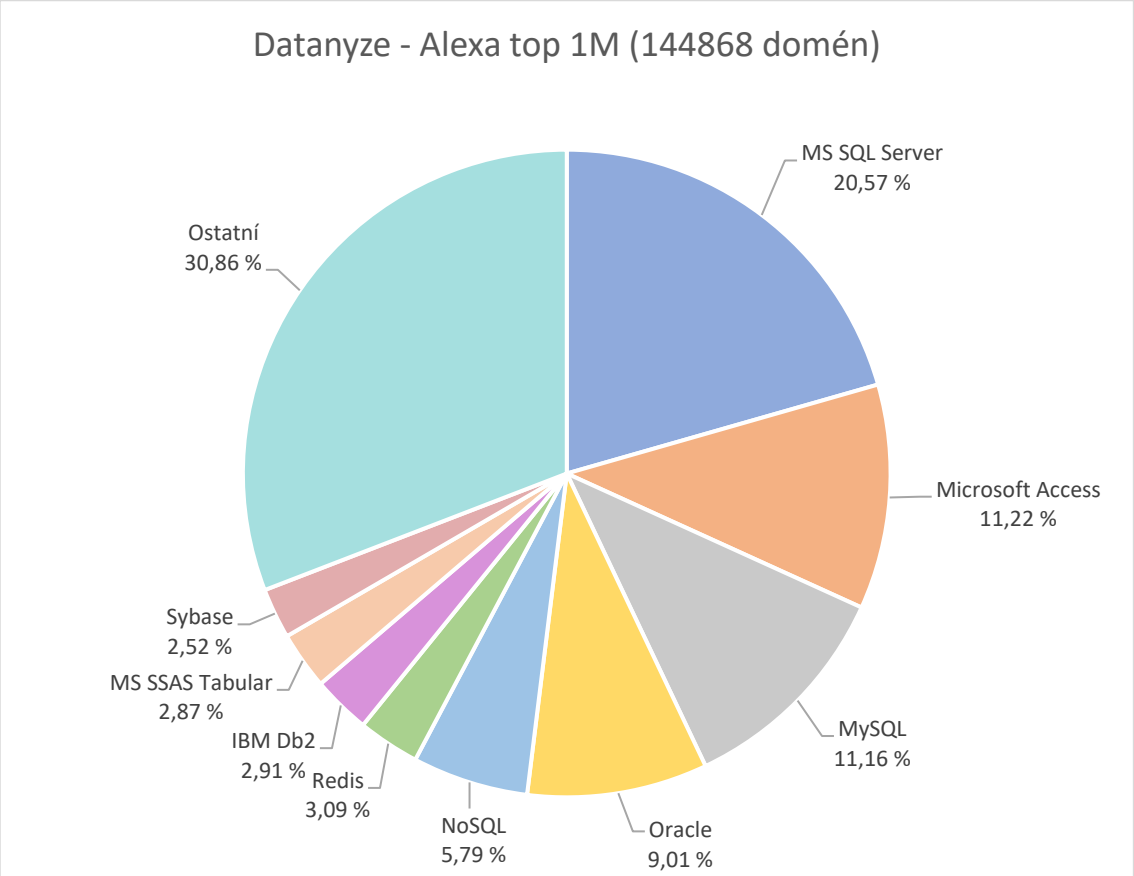


Graf 7: DB-Engines 6/2020 [36] [37]

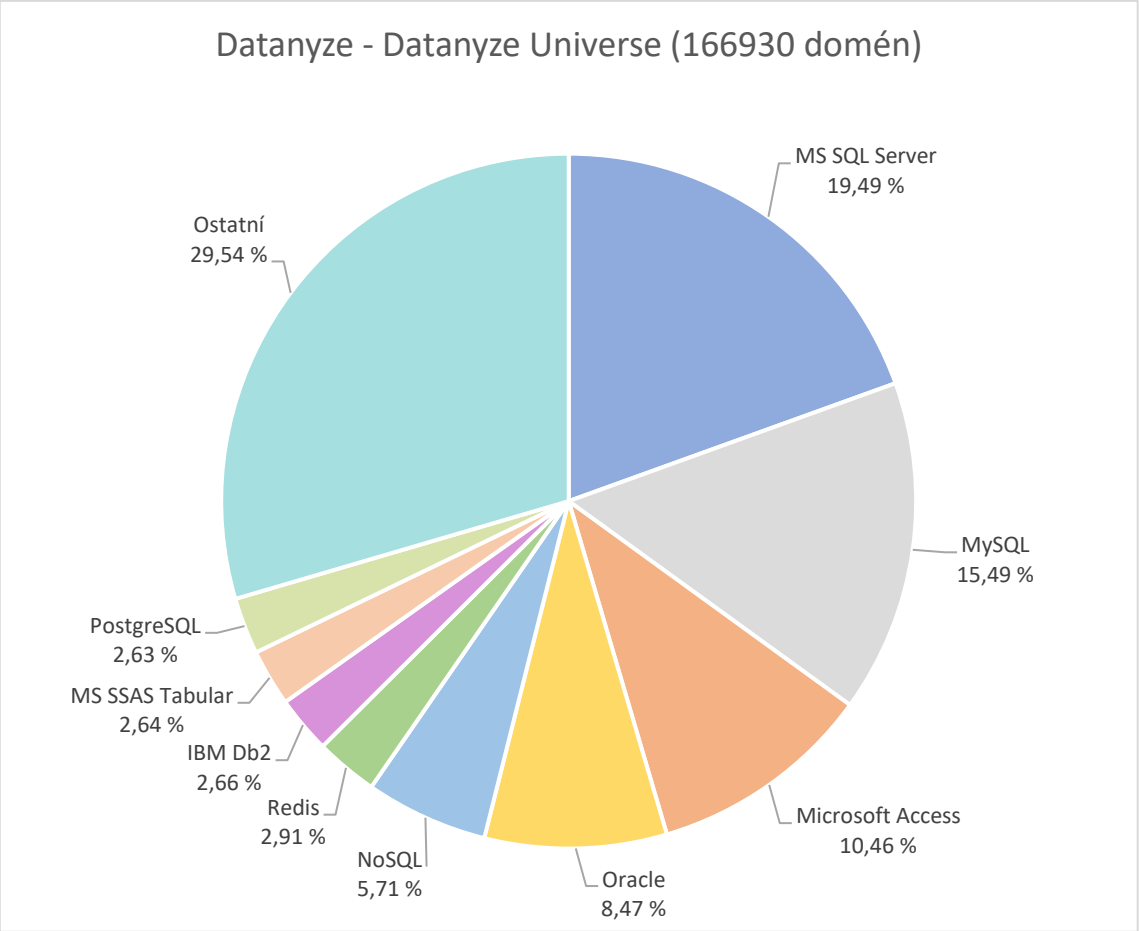
Výsledky z června roku 2018 (graf 5), 2019 (graf 6), 2020 (graf 7) si jsou velmi podobné. Jako nejpopulárnější databázové systémy ukazují: Oracle, MySQL a SQL Server. To, že má databáze vysoké skóre, nutně nemusí znamenat, že má stejně vysoký počet instalací. Vzhledem k tomu, z jakých dat DB-Engines vychází, ale rozhodně bude daný systém na trhu relevantní. Při srovnání výsledků s daty ze Stack Overflow průzkumu figurují mezi nejpopulárnějšími systémy stejná jména. Prvních deset míst obsadilo osm stejných systémů, jejich umístění se ale vždy neshoduje. Největší rozdíl v umístění se týká Oracle, které podle DB-Engines suverénně vede, ale u Stack Overflow obsazuje nižší příčky. Za zmínku stojí také rozdíl u SQLite, které je naopak používanější než nasvědčují data z DB-Engines. Vzhledem ke způsobu hodnocení byly rozdíly v pořadí systémů očekávány. Protože Stack Overflow konalo průzkum přímo mezi vývojáři, zatímco DB-Engines odhaduje popularitu na základě ukazatelů, které nemusí souviset pouze s rozšířením systému, jeví se data ze Stack Overflow jako lepší ukazatel toho, jaké systémy se aktuálně používají pro vývoj nových aplikací.

2.2.3 Datanyze

Datanyze je společnost nabízející řešení pro analýzu trhu skrze tzv. technografickou segmentaci. Technografická data jsou podobně jako demografická data používána pro porozumění trhu a lepšímu cílení marketingu na potenciální zákazníky. Jen je pozornost soustředěna na technologie, které používají, místo indikátorů jako je věk nebo příjem. Datanyze používá strojové učení společně s blíže nespecifikovanými metodami k tomu, aby získala data o více než 35 milionech společností po celém světě. Data je možné filtrovat podle země nebo Alexa žebříčku. Alexa Internet je dceřiná společnost obchodního gigantu Amazon. Ta sbírá statistická data od uživatelů jejich rozšíření do webového prohlížeče. Tato data jsou analyzována a je publikován žebříček nejnavštěvovanějších stránek. Z Datanyze můžeme získat data pro některé stránky z tohoto žebříčku. Data zde prezentovaná jsou omezena na ty z nejnavštěvovanějšího milionu stránek podle Alexa žebříčku (graf 8) a všechna data dostupná na Datanyze (graf 9) [38] [39] [40] [41] [42].



Graf 8: Datanyze Alexa TOP 1M [40]



Graf 9: Datanyze Universe [41]

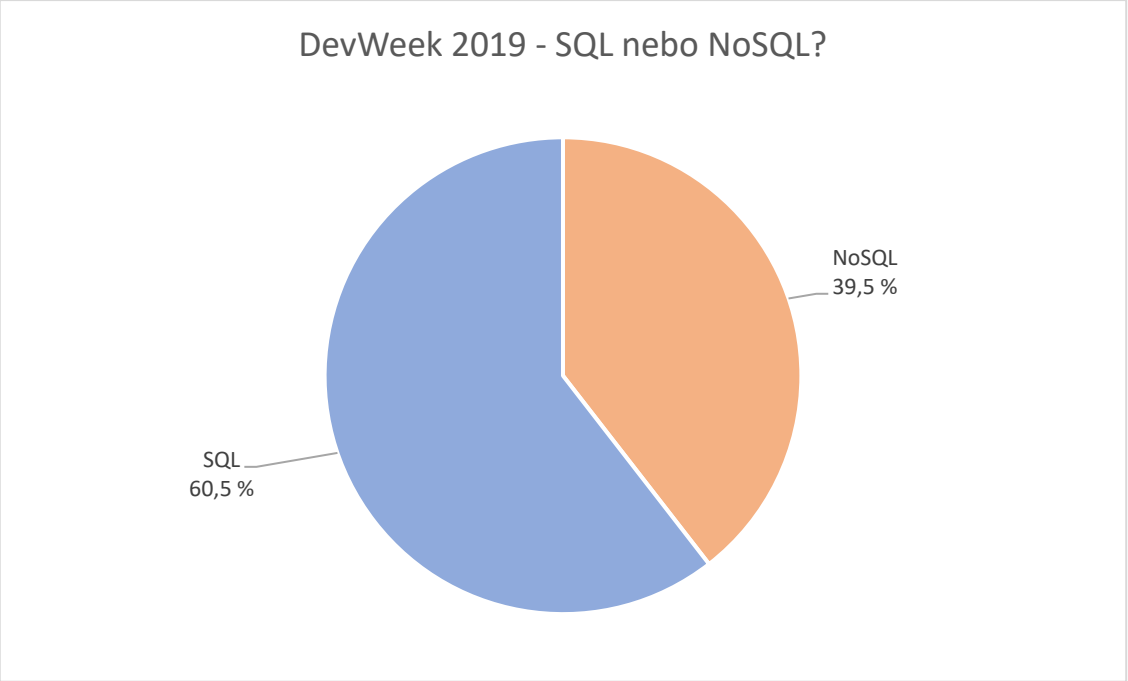
Data podle Alexa žebříčku a všechna data, co má Datanyze k dispozici, se příliš neliší. Největší rozdíl je u podílu MySQL. To není překvapivé, vzhledem k tomu, že zbytek dat zahrnuje i menší společnosti, než jsou ty v prvním milionu Alexa žebříčku. MySQL je open source software, takže je jeho vyšší popularita mezi o něco menšími společnostmi opodstatněná. Přestože jsou data z Datanyze zajímavá, nebude jim přikládána taková váha jako předchozím dvěma zdrojům. Z prvního milionu nejnavštěvovanějších stránek se podařilo získat informace pouze o zlomku domén a jedná se o větší společnosti, které nutně nereprezentují zbytek trhu. Získaná data však působí důvěryhodně, Datanyze má silnou incentivu poskytovat přesná data kvůli službám, které nabízí.

2.2.4 DeveloperWeek

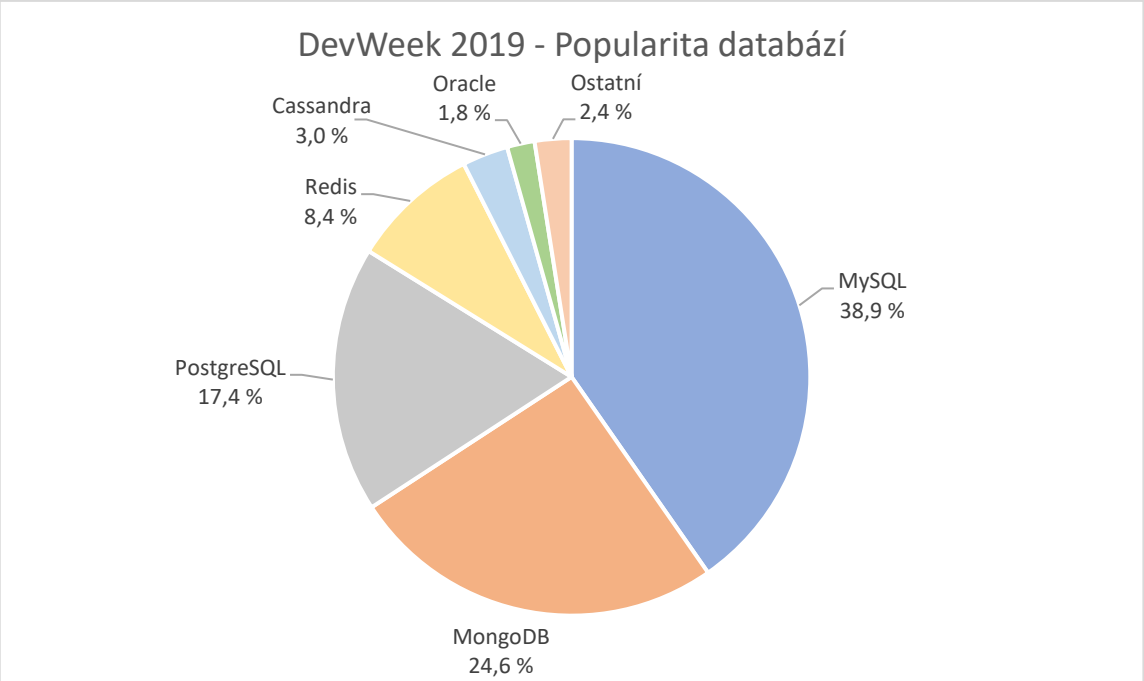
DeveloperWeek se koná v San Franciscu. Je to příležitost, při které se schází více než 8 000 vývojářů a lidí jiných profesí z oboru. Mohou se zde dozvědět novinky ze světa programování, získat nabídku práce nebo se účastnit hackathonu. Cestu si tam udělají lidé z více než 70 různých zemí, proto je to dobrá příležitost zjistit, s jakými technologiemi pracují. Toho využila společnost ScaleGrid, která se věnuje databázím. Konkrétně nabízí provoz databází jako předplacenou službu pro průmyslové použití. Jejich cloudové řešení umožňuje spravovat MySQL, PostgreSQL, Redis nebo MongoDB databáze na Amazon Web Services, Microsoft Azure nebo jejich vlastních dedikovaných serverech [43] [44] [45].

Na DeveloperWeek, který se konal v roce 2019, se tedy zástupci ScaleGrid vyptávali stovek vývojářů na to, jaké technologie používají.

První otázka zjišťuje, jaké modely databází jsou využívány (graf 10). Většinu získaly relační databáze.



Graf 10: DevWeek 2019 SQL/NoSQL [45]



Graf 11: DevWeek 2019 [45]

Druhá otázka již byla zaměřena na jednotlivé databázové systémy (graf 11). První tři místa obsadila: MySQL, MongoDB, PostgreSQL.

Vzhledem k nízkému počtu tázaných a povaze služeb, které ScaleGrid nabízí, tato data působí nejméně důvěryhodně. Obzvlášť zarážející je absence Microsoft SQL Server mezi nejrozšířenějšími systémy. Podle všech předchozích zdrojů se jedná o jeden z nejpopulárnějších systémů, zde je však schován až v kategorii „ostatní“. Za zmínku stojí také fakt, že se nejvýše umístily právě ty systémy, pro které ScaleGrid nabízí služby.

2.2.5 Vyhodnocení

Byly prezentovány různé zdroje dat o popularitě databázových systémů. Po srovnání výsledků a uvážení důvěryhodnosti jednotlivých zdrojů se jako systémy, kterým se vyplatí věnovat nejvíce pozornosti, jeví: MySQL, Microsoft SQL Server a PostgreSQL.

3 Představení vybraných databázových systémů

Existence tak velkého množství různých databázových systémů dává smysl, protože poskytují různé funkce a mají různé obchodní modely. V této kapitole budou krátce představeny systémy vybrané ve vyhodnocení minulé kapitoly.

3.1 MySQL

MySQL je relační databázový systém, který byl vyvíjen ve švédské firmě MySQL AB, založené v roce 1995. Ve stejném roce byla hotová i první interní verze MySQL, její vývoj však začal už v roce 1994. Projekt stavěl na předchozí práci jednoho ze tří zakladatelů. Jednalo se o databázový systém Unireg z roku 1982 vyvíjený pro švédskou společnost TcX. Do něj byl implementován SQL jazyk. První veřejná verze byla vydána v létě 1996. Verze pro operační systémy Windows následovala v roce 1998. Důležitým milníkem byl rok 2000, kdy byl systém vydaný jako open source software pod GPL licencí. To mělo za následek prudký pokles příjmů a nějaký čas trvalo, než se firma ze ztrát zotavila. MySQL bylo i nadále nabízeno i pod placenou komerční licencí pro zákazníky, kteří nechtějí svůj kód dále sdílet kvůli GPL licenci. V roce 2001 dosáhl systém dvou milionů aktivních instalací a implementoval podporu pro databázové transakce – tento trend přidávání funkcí důležitých pro nasazení ve velkých firmách pokračoval i v dalších letech. Dále získal podporu investorů, ale také čelil první žalobě. V roce 2002 se společnost žalobě bránila podáním protižaloby pro porušení GPL licence a obě strany se nakonec dohodly na mimosoudním vyrovnání. Ve stejném roce bylo také otevřeno nové sídlo v USA a uživatelská základna se rozrostla až na tři miliony. Počet uživatelů dále rostl i v roce 2003, kdy se dostal na čtyři miliony. V roce 2004 byla přehodnocena dosavadní obchodní strategie a firma se začala soustředit na získání zákazníků z řad firem. Cílem bylo si zajistit opakovaný příjem místo jednorázového poplatku za licenci. Dalším zdrojem příjmů byly služby týkající se technické podpory, konzultací a certifikačních procesů. V následujících letech Oracle odkoupilo Innobase a Sleepycat – dvě společnosti, které s MySQL AB spolupracovaly. V roce 2006 byl ze strany Oracle učiněn pokus o odkoupení MySQL, který byl neúspěšný. O dva roky později však MySQL AB odkoupila společnost Sun Microsystems za cca jednu miliardu dolarů. Po krátké době opouští MySQL AB dva z jeho tří zakladatelů. V roce 2009 firmu již opustili všichni tři zakladatelé, a ve stejném roce se podařilo Oracle dohodnout se Sun Microsystems o vykoupení jejich společnosti včetně práv na MySQL [46] [47] [48] [49] [50] [51] [52].

MySQL je stále ve vývoji a je nabízeno v rámci ročních předplacených služeb a nadále i v open source variantě s GPL licencí. Je považováno za nejoblíbenější open source relační databázový systém. Mezi jejich zákazníky se řadí jména jako YouTube, PayPal, Facebook, LinkedIn, Twitter, Netflix, NASA, Tesla, Toyota, Nokia nebo Tencent [53] [54].

3.2 Microsoft SQL Server

Microsoft SQL Server začal jako společný projekt mezi společnostmi Microsoft, Ashton-Tate a Sybase. Cílem bylo vytvořit konkurenci na trhu, který v té době dominovalo Oracle a IBM. Výsledný produkt, vydaný v roce 1989, byl v podstatě pouhou konverzí již existujícího databázového systému na operační systém OS/2. Základem této konverze byl Sybase SQL Server pro Unixové systémy. V roce 1991 následovala jedna další verze SQL Serveru pro OS/2, ale protože začal Microsoft dominovat trh s operačními systémy, bylo rozhodnuto další verzi přizpůsobit pro jeho Windows NT platformu. Toto rozhodnutí nebylo v Sybase kladně přijato a vztah obou společností se začal zhoršovat. Microsoft SQL Server 4.21 pro Windows NT byl vydán v roce 1993. O rok později bylo partnerství mezi Sybase a Microsoftem formálně ukončeno. Práva na všechny verze SQL Server pro operační systémy Windows získal Microsoft. Další verze přinesly razantní změny v kódu, výrazně rozšířily nabídku funkcí a zlepšily optimalizaci. Jednotlivé části systému byly postupně přepracovány a v roce 2005 již nebyl přítomný žádný původní kód od Sybase [55] [56] [57] [58].

Microsoft nabízí svůj SQL Server v několika edicích, které se liší funkcemi, limitacemi pro využitelný HW a hlavně cenou. Pro vývojáře je však k dispozici SQL Server Developer, který je ke stažení zdarma a lze ho legálně použít pro vývoj a testování aplikací. Standardní edici SQL Serveru je možné licencovat pod dvěma modely. První je „core-based“, kde každý server, na kterém MS SQL Server běží, musí mít licenci na každé jádro procesoru. Připojit se pak může neomezené množství klientů. To platí i v případě virtualizace, kde je třeba licence na každé virtuální jádro. A pro oba případy platí, že na každý procesor nebo VM je třeba koupit licenci na alespoň čtyři jádra i když jimi systém nedisponuje. Druhý model je „Server + CAL“, kde každý server, na kterém MS SQL Server nebo jeho libovolná komponenta běží, musí mít licenci. A každé zařízení nebo uživatel, které se chce připojit, musí mít licenci CAL (Client Access License). Enterprise edice mířená na velké společnosti je potom dostupná pouze s core-based licencí. Microsoft SQL Server používají společnosti jako Dell, Stack Overflow, Malaysia Airlines nebo město Olomouc [59] [60].

3.3 PostgreSQL

PostgreSQL původně neslo název Postgres (POST inGRES). Ten se odkazuje na databázový systém Ingres, ze kterého vychází. Práce na novém databázovém systému začala kvůli problémům dále rozšiřovat Ingres. Jeho tvůrce měl jasné cíle pro tento nový databázový systém, které kvůli stavu, v jakém se Ingres nacházel, nebylo možné uskutečnit. V oné době už byl Ingres několik let po dokončení a po rozšíření několika dalšími prototypy nebyl, dle slov jeho autora, v dobrém stavu. Zároveň implementaci nových funkcí bránila některá předchozí rozhodnutí při návrhu celého systému [61] [62].

Cílem nového projektu bylo:

- Poskytnout lepší podporu pro komplexní objekty – např. objekty složené z polygonů, čar a kružnic.

- Umožnit práci s větším množství datových typů např. pro integraci s CAD nebo CAM systémy.
- Implementovat principy aktivních databází (Událost – Podmínka – Akce).
- Redukovat množství kódu potřebného k obnovení databáze při chybě.
- Efektivně využít nové technologie (pracovní stanice s více procesory, úložiště s optickými disky).
- Zachovat relační model beze změn [62].

Vývoj projektu Ingres na univerzitě v Berkeley oficiálně skončil v roce 1985. O rok později byl spuštěn projekt Postgres. Tato snaha byla sponzorována několika organizacemi, včetně armádní agentury DARPA a ARO. V roce 1987 byla připravena první předváděcí verze systému, která byla o rok později prezentována na konferenci SIGMOD (Special Interest Group on Management of Data). Vývoj pokračoval a rostla i uživatelská základna, což mělo za následek oficiální ukončení projektu v roce 1994. S rostoucím počtem uživatelů totiž stoupaly i časové nároky na údržbu kódu a technickou podporu. To ale nezastavilo dva studenty, kteří ve stejném roce do projektu implementovali jazyk SQL, místo dosud používaného POSTQUEL. Tato verze nesla název Postgres95 a zahrnovala řadu dalších vylepšení, včetně o 30-50 % lepší výsledky v testu Wisconsin Benchmark. V roce 1996 byl název projektu změněn na PostgreSQL. Jasně tak dává najevo použití jazyka SQL [61] [63] [64].

PostgreSQL je stále vyvíjen a je nabízen pod Open Source PostgreSQL licencí, která je podobná BSD nebo MIT licencím. Mezi společnosti používající PostgreSQL se řadí např. Uber, Instagram, Spotify nebo Reddit [65].

3.4 Srovnání

Vybrané databázové systémy budou porovnány na základě jejich vlastností a vlastního měření rychlosti.

3.4.1 Vlastnosti

Na stránce DB-Engines.com je ke každé hodnocené databázi k dispozici i rychlý přehled jejich vlastností. Databáze porovnává také stránka mssqltips.com. Z těchto dat byla sestavena tabulka 1, kde jsou především údaje, ve kterých se databáze nějakým způsobem liší nebo můžou hrát velkou roli při výběru systému. MS SQL Server jako jediný nemá open source licenci. Všechny systémy jsou napsány v C nebo C++. PostgreSQL podporuje největší spektrum operačních systémů, zatímco MySQL si rozumí s největším počtem programovacích jazyků. Při práci s velmi rozměrnými tabulkami potom bude hrát roli metoda dělení dat. Při replikaci dat se data ukládají na více serverů, pro zajištění lepší přístupnosti dat pro více uživatelů a lepší spolehlivost databáze [66] [67] [68].

	MySQL	MS SQL Server	PostgreSQL
Vývojář	Oracle	Microsoft	PostgreSQL Global Development Group
Rok vydání	1995	1989	1989
Poslední verze	8.0.20 (2020)	2019 (2019)	12.3 (2020)
Licence	Open Source (GNU GPL), komerční	Komerční	Open Source (MIT like)
Napsáno v	C a C++	C++	C
Podporované OS	Windows, Linux, OS X, FreeBSD, Solaris	Windows, Linux	Windows, Linux, OS X, Unix, FreeBSD, Solaris, OpenBSD, NetBSD, HP-UX
Podporované programovací jazyky	C, C#, C++, Java, Python, PHP, JavaScript, Delphi, Objective-C, D, Eiffel, Erlang, Haskell, Ocaml, Perl, Ruby, Scheme, Tcl, Ada	C#, C++, Java, Python, Visual Basic, PHP, JavaScript, Ruby, R, Go	C, C++, Java, Python, .NET, JavaScript, PHP, Delphi, Perl, Tcl
Podpora API	ODBC, JDBC, ADO.NET, vlastní API	ODBC, JDBC, ADO.NET, OLE DB, TDS	ODBC, JDBC, ADO.NET, nativní C knihovna, streaming API
Metody dělení dat	dělení podle rozsahu, seznamu hodnot, hash funkce	dělení podle rozsahu	dělení podle rozsahu, seznamu hodnot, hash funkce
Metody replikace	master-master, master-slave	dle edice	master-slave
Paralelní provedení příkazu	1 CPU na příkaz	Možnost více CPU na příkaz	Možnost více CPU na příkaz
Cache pro příkazy	Lze nastavit velikost, rozdělit na více částí	Připojení = proces, každé má přiřazenou svoji paměť	Lze nastavit, nelze rozdělit

Tabulka 1: Vlastnosti databázových systémů [67] [68]

Velikost tabulek u MS SQL Server je omezena pouze dostupným úložištěm. Teoretická maximální velikost databáze je stanovena na 524 272 TB. Limity se však mohou lišit dle edice [69].

U MySQL nejsou žádné interní limity na velikost tabulky nebo databáze. Limitujícím faktorem tedy bude použitý souborový systém. Používat FAT32, kde je maximální velikost souboru necelé 4 GB, se tedy silně nedoporučuje [70] [71].

V případě PostgreSQL závisí maximální velikost tabulky na velikosti segmentů, po kterých je ukládá. Defaultně jsou velké 8 192 B, přičemž maximální velikost tabulky je 32 TB. Maximální velikost databáze není omezena, takže bude limitována souborovým systémem [72].

3.4.2 Měření rychlosti

Pro porovnání rychlosti vybraných databází byl proveden test pomocí Apache JMeter 5.3. Test byl prováděn na nejnovějších dostupných verzích databázových systémů a jedné starší verzi MySQL (MySQL 5.7.20). K databázím se přistupovalo skrze JDBC API. Pro eliminování bottlenecku v připojení byl test prováděn na lokální síti. Pro test byla zvolena tabulka s náhodně generovanými čísly o deseti sloupcích a 10 000 řádcích. Pro generování testovací tabulky byl napsán Matlab skript, který sloupce naplnil požadovaným počtem řádků. Zároveň sloužil jako test připojení k různým typům databází. Každý z testů se skládal ze 100 měření po 10 nepřetržitých opakováních, mezi kterými byly 10sekundové prodlevy.

Testovací sestava:

- Procesor: AMD Ryzen 3600 při 4,15 GHz, 6 jader, 12 vláken
- RAM: 2x8 GB, 3200 MHz, CL16, dual-channel
- SSD: Intel 600p NVMe PCIe M.2
- OS: Windows 10 Pro (1909), Build: 18363.900

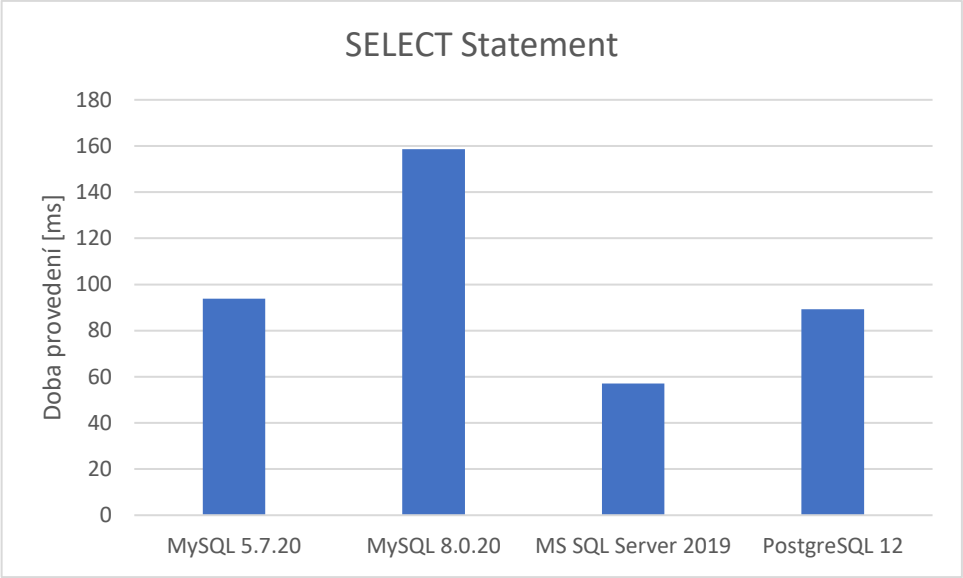
Testované databáze:

- MySQL Community Server 5.7.20
- MySQL Community Server 8.0.20
- Microsoft SQL Server 2019 Developer Edition
- PostgreSQL 12

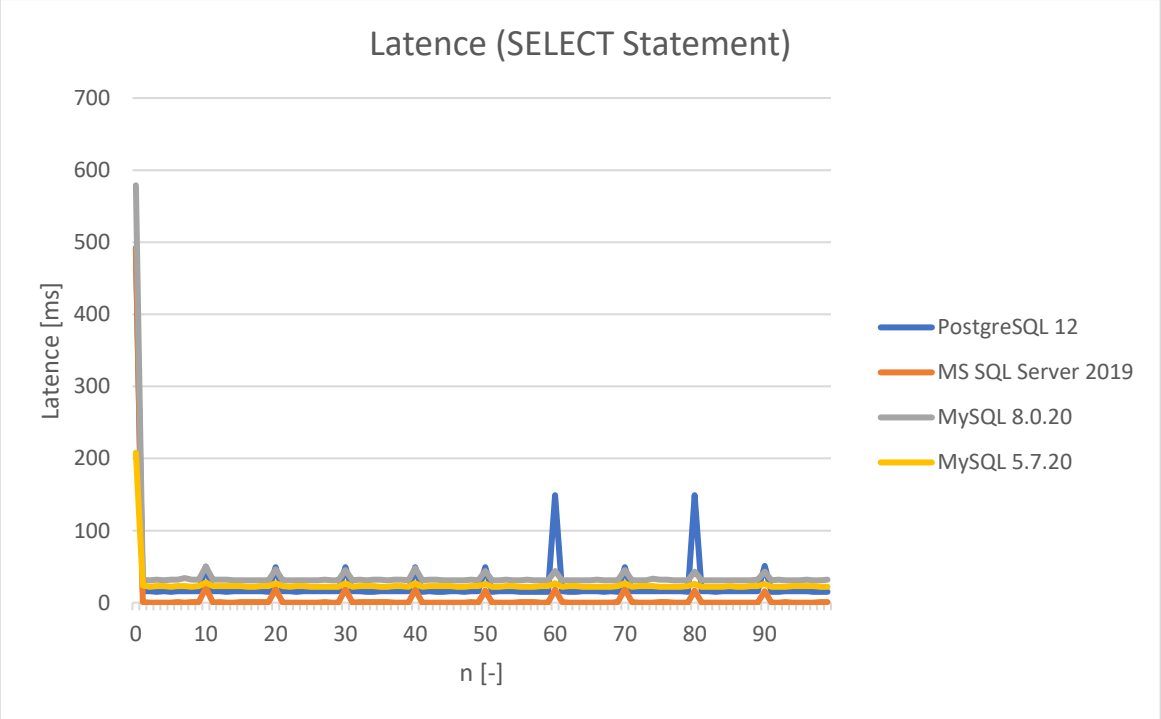
Prvním testem byl příkaz SELECT, který je používán pro vybrání dat z databáze. Vybraná data jsou vrácena ve formě tabulky. Příkaz (obrázek 5) byl aplikován na všechna data v tabulce. V grafu 12 je průměrná doba provedení jednoho příkazu. V grafu 13 je latence pro každé provedení příkazu [73].

```
1 SELECT * FROM bench;
2
```

Obrázek 5: První testovaný příkaz



Graf 12: SELECT



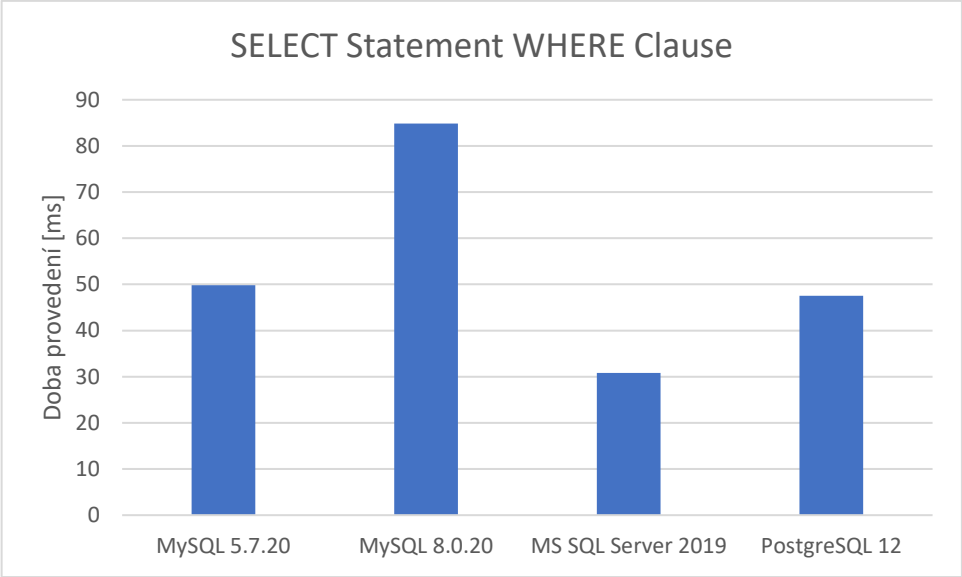
Graf 13: Latence

Nejlépe si vedl MS SQL Server 2019, následuje PostgreSQL 12 a MySQL 5.7, nejpomalejší bylo MySQL 8.0. Co se týče latence, opět vyhrává MS SQL Server 2019. Ostatní systémy jsou srovnatelné. Za zmínku stojí jen výkyvy u PostgreSQL 12 při prodlevě mezi přístupy.

Další test byl opět SELECT příkaz, ale s podmínkou WHERE (obr. 6). Takže budou vybrána jen některá data z testovací tabulky. V grafu 14 jsou průměrné časy provedení jednoho příkazu [74].

```
1  SELECT * FROM bench
2  WHERE a BETWEEN 5 AND 30;
3
```

Obrázek 6: Druhý testovaný příkaz



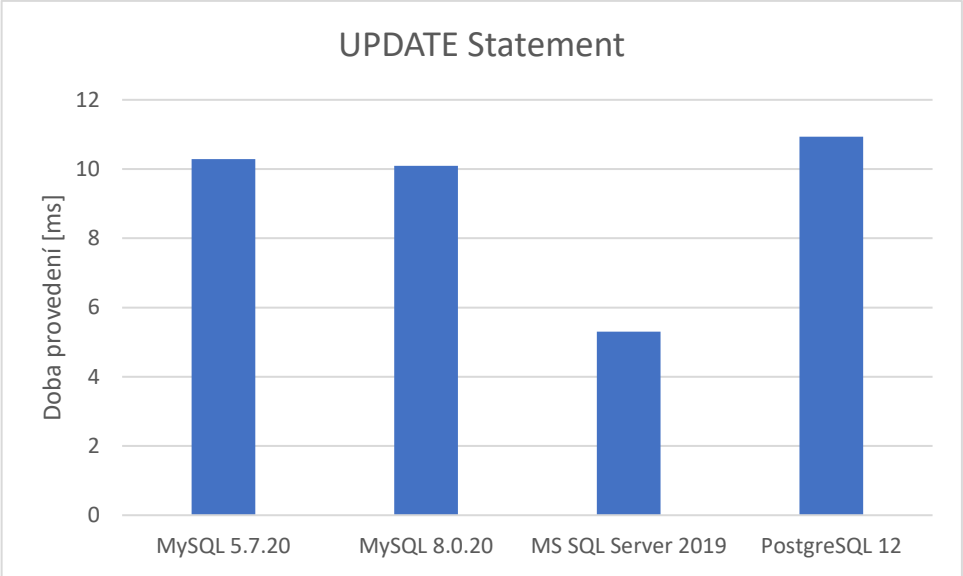
Graf 14: SELECT WHERE

Výsledky jsou téměř totožné s minulým testem, jen se zkrátila doba provedení testu u všech systémů – kvůli přenosu menšího objemu dat.

Posledním testem byl příkaz UPDATE, který slouží k úpravě existujících dat. Byly provedeny tři různé úpravy, každá dle určité podmínky s pomocí WHERE (obr. 7). Průměrná doba provedení jednoho příkazu je v grafu 15 [73] [75].

```
1  UPDATE bench SET a = 5 WHERE a < 2;
2  UPDATE bench SET b = 10 WHERE b BETWEEN 2 AND 15;
3  UPDATE bench SET c = 1000 WHERE c BETWEEN 0 AND 50;
4
```

Obrázek 7: Třetí testovaný příkaz



Graf 15: UPDATE

Výsledky posledního testu ukazují, že nejrychlejší je MS SQL Server 2019, všechny ostatní systémy jsou srovnatelné.

Po vyhodnocení všech testů se jeví Microsoft SQL Server 2019 jako konzistentně nejrychlejší databáze. Rychlost MySQL se mezi dvěma testovanými verzemi výrazně změnila. Starší verze 5.7.20 je srovnatelná s PostgreSQL 12, zatímco novější MySQL 8.0.20 zaostává. U latence PostgreSQL se při prodlevě mezi příkazy někdy vyskytovaly výrazné výkyvy. To by mohlo hrát roli při výběru systému tam, kde je nízká latence prioritou. Také je třeba brát v potaz relativní jednoduchost testovaných příkazů. Časy se totiž mohou výrazně lišit v závislosti na specifickém workloadu. Pro tyto testy byly zvoleny příklady, které by měly reprezentovat využití databáze, např. při zapisování a čtení výsledků měření. Všechny systémy poskytují dostatečnou rychlost pro využití v praktické části práce.

4 Implementace databází

Jak již bylo zmíněno, velkou výhodou relačních databází je jejich vysoká míra standardizace. Krom jazyka SQL existuje i několik dalších důležitých standardů, které s databázovými systémy usnadňují práci. Právě díky nim je možné s databází efektivně komunikovat. Před nástupem těchto standardů zněla možnost použít jednu aplikaci pro přístup k více různým databázovým systémům, bez nutnosti měnit kód při jejich implementaci, nereálně. Nyní je to běžné.

4.1 ODBC

Application Programming Interface (API) definuje, jak spolu budou interagovat různé softwarové komponenty. Je to soubor procedur, protokolů a nástrojů, určený pro tvorbu aplikací. Open Database Connectivity (ODBC) je API, které slouží pro přístup k datům v databázích. S použitím ODBC tak mohou aplikace přistupovat k datům bez ohledu na to, o jaký databázový systém se jedná, jaký operační sytém počítač používá nebo na jakém typu počítače běží. ODBC je založeno na standardu (ISO/IEC 9075-3:2003) Call Level Interface (CLI). Ten definuje, jakým způsobem bude program používat SQL příkazy při komunikaci s databázovým systémem [76].

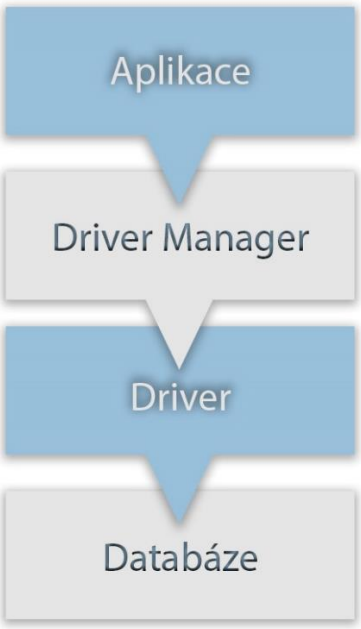
Při standardizaci přístupu k databázím pomocí ODBC bylo úkolem vyřešit následující výzvy:

- Přístup k různým databázovým systémům bez nutnosti změny zdrojového kódu aplikace.
- Přístup k několika různým databázovým systémům současně.
- Jaký rozsah funkcí má ODBC podporovat. Různé databázové systémy implementují různé funkce, které ostatní nemusí nabízet [77].

První problém je vyřešen použitím knihovny nebo ovladače pro každý databázový systém, který ODBC podporuje. Ovladač se stará o správnou implementaci funkcí definovaných v ODBC. Pro použití aplikace s jiným databázovým systémem tedy není nutno měnit její zdrojový kód. Je pouze načten jiný ovladač / knihovna. Druhý problém je vyřešen načtením potřebného počtu příslušných ovladačů a poskytnutím driver manageru. Driver manager implementuje všechny funkce v ODBC, aplikace tak funkce volá dle jejich jména v driver manageru místo použití pointeru pro každý driver. Ke třetímu problému se ODBC staví tak, že podporuje větší výběr funkcí, než nabízí většina databázových systémů. Kdyby byly podporovány pouze funkce, co jsou společné pro všechny databázové systémy, tak by ODBC postrádalo smysl. Je vyžadováno, aby driver podporoval určité základní funkce, ale podpora všech není podmínkou. Výhodou ODBC je, že poskytuje společné rozhraní pro všechny funkce, které podporuje. Aplikace tedy obsahují kód, který se vztahuje k těmto funkcím v ODBC, ten tak není specifický pouze pro jeden databázový systém. Kód tedy není nutné měnit, stačí mít příslušný driver pro daný systém. To platí i v případě, že je do nějakého databázového systému přidána nová

funkce a nově s ním chceme použít naši aplikaci. Ilustrace ODBC architektury je na obr. 8. Obsahuje tyto komponenty [77] [78] [79] [80]:

- Aplikace – volá ODBC funkce pro odeslání SQL příkazů a získání výsledků.
- Driver Manager – předává výstup z aplikace správnému driveru.
- Driver – zpracovává požadavky na ODBC funkce, odesílá SQL příkazy a přijímá zpět výsledky.
- Databázový systém – zpracovává požadavky z driveru a vrací mu výsledky.



Obrázek 8: ODBC [80]

ODBC podporuje většina rozšířených relačních databázových systémů včetně Microsoft SQL Server, Oracle, PostgreSQL, MySQL a DB2.

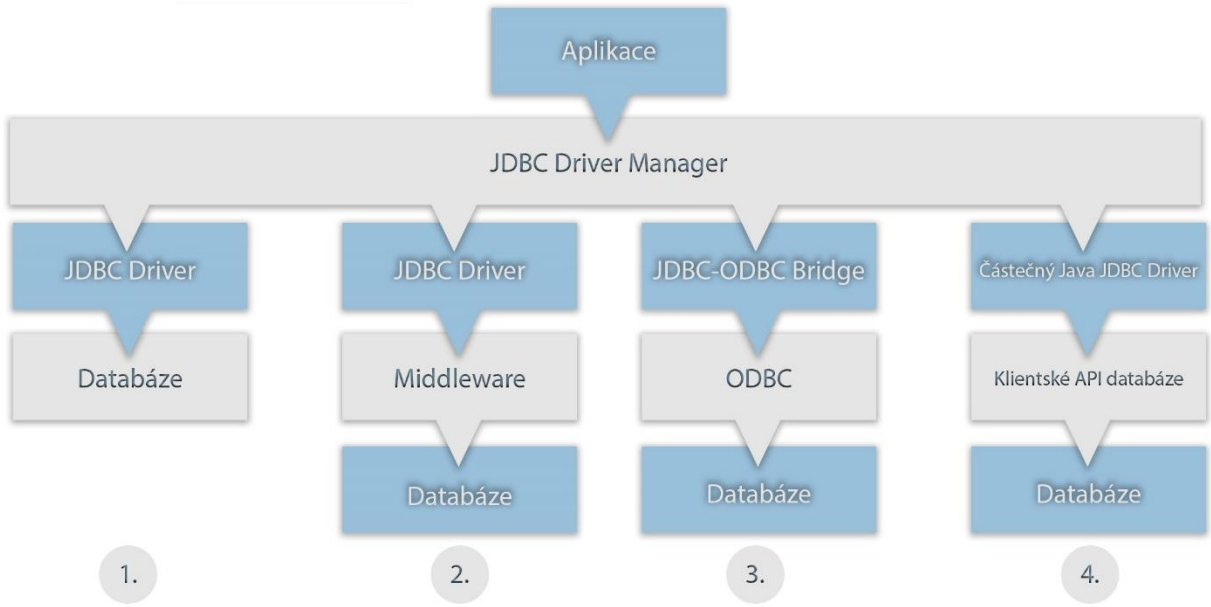
4.2 JDBC

Java Database Connectivity (JDBC) je API pro připojení a práci s různými databázemi, určené pro programovací jazyk Java. Umožňuje používat Java aplikace s různými databázemi bez nutnosti aplikace. Jeho vývoj začal v roce 1996 a dbalo se při něm na zpětnou vazbu od tvůrců databázových systémů. Stejně jako ODBC je JDBC založeno na CLI standardu. JDBC je součástí Java platformy od roku 1997, kdy bylo přidáno do nástrojů pro vývojáře Java Development Kit 1.1. Od té doby je s ním aktualizováno a bylo také přidáno do programu Java Community Process, kde můžou členové komunity ovlivňovat jeho vývoj [81] [82] [83].

Existují čtyři typy JDBC rozhraní (obr. 9) dělící se do dvou kategorií:

- Pouze JDBC ovladače:
  1. Direct-to-Database Pure Java Driver
     - Komunikuje přímo s databázovým serverem.
  2. Pure Java Driver for Database Middleware
     - Komunikuje s databázovým middleware, které komunikuje se serverem.

- JDBC spolu s ODBC ovladačem nebo knihovnami databáze:
  3. JDBC-ODBC Bridge a ODBC ovladač (odebráno v JDK8)
     - Umožňuje používat JDBC API s ODBC ovladačem.
  4. Nativní API databáze s (částečně) JDBC ovladačem
     - Ovladač překládá JDBC komunikaci pro klientské API databáze.



Obrázek 9: JDBC

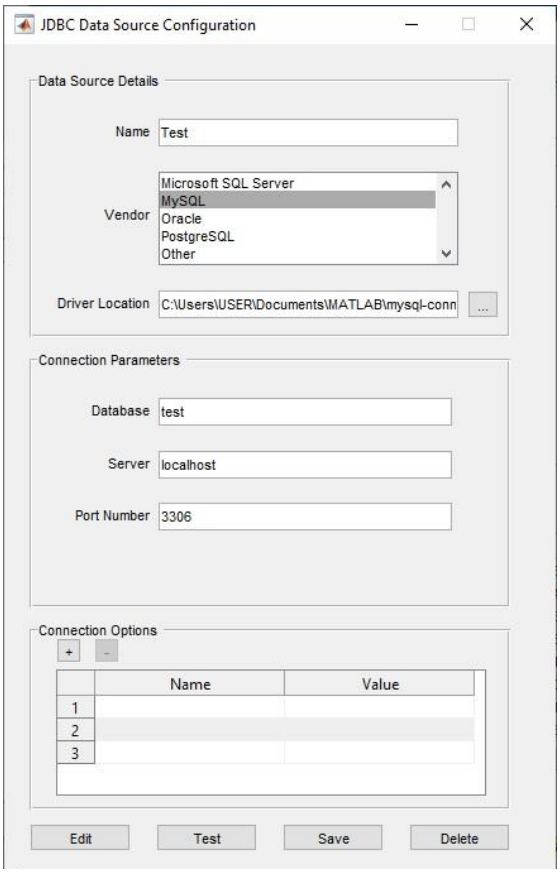
Díky ODBC a JDBC standardům a faktu, že Matlab podporuje některé programovací jazyky, je možné různé databázové systémy integrovat skoro totožně. Stačí poskytnout odpovídající driver. Kód samotných skriptů může zůstat nezměněn. Při další práci s databází a programem Matlab bude použit standard JDBC.

5 Matlab Database Toolbox

Mathworks nabízí toolbox pro svůj software Matlab, který umožňuje pracovat s relačními a některými ostatními databázemi. Dokáže data z databáze importovat, provádět s nimi libovolné operace a také je exportovat do databází. Nabízí také aplikaci Database Explorer, která má grafické rozhraní pro práci s databázemi i bez nutnosti ovládat jazyk SQL [84].

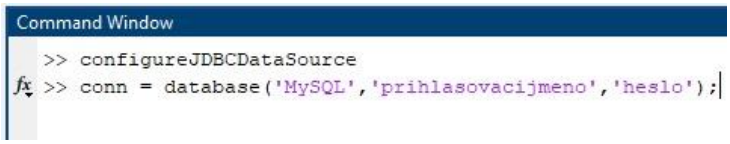
5.1 Připojení k databázi

Připojení k relačním databázím probíhá pomocí ODBC nebo JDBC driveru. ODBC nabízí vyšší rychlost při importu a exportu dat, ale vyžaduje větší množství paměti, není dostupný mimo Windows a není s ním dostupná funkce *runstoredprocedure*. JDBC je kompatibilní s větším množstvím operačních systémů a verzí ovladačů, podporuje všechny funkce DB Toolboxu. Pro případ, kdy chce uživatel pracovat s relační databází bez nutnosti instalovat databázový systém nebo ovladač, umí Matlab pracovat i s SQLite databázemi. SQLite je relační databázový systém, který je celý obsažený v C knihovně a ukládá celou databázi v jednom souboru [84] [85] [86]. Po přidání databáze v

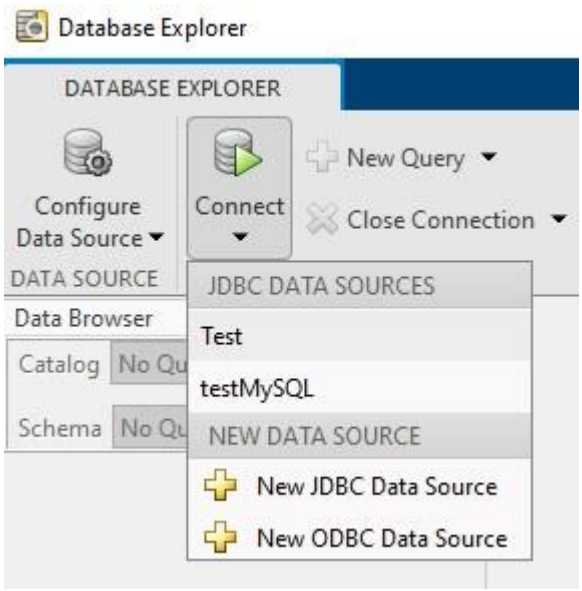


Obrázek 10: Přidání Data Source

Matlabu jako „Data Source“ (obr. 10) je možné se k databázi připojit s pomocí příkazového řádku (obr. 11) nebo aplikace Database Explorer (obr. 12). Před přidáním je třeba se ujistit, zda je přítomen příslušný ODBC nebo JDBC driver. Krom ovladače je třeba poskytnout adresu serveru, port a přihlašovací údaje. Pro Microsoft SQL Server je také k dispozici přihlášení pomocí uživatelského účtu Windows. Je možné se připojit k více databázím nebo vytvořit více připojení pro jednu databázi [84] [87] [88].



Obrázek 11: Připojení pomocí příkazové řádky



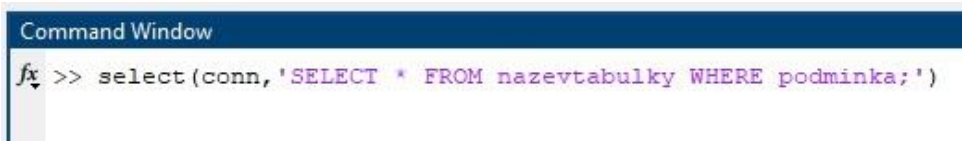
Obrázek 12: Připojení pomocí DB Explorer

5.2 Import dat

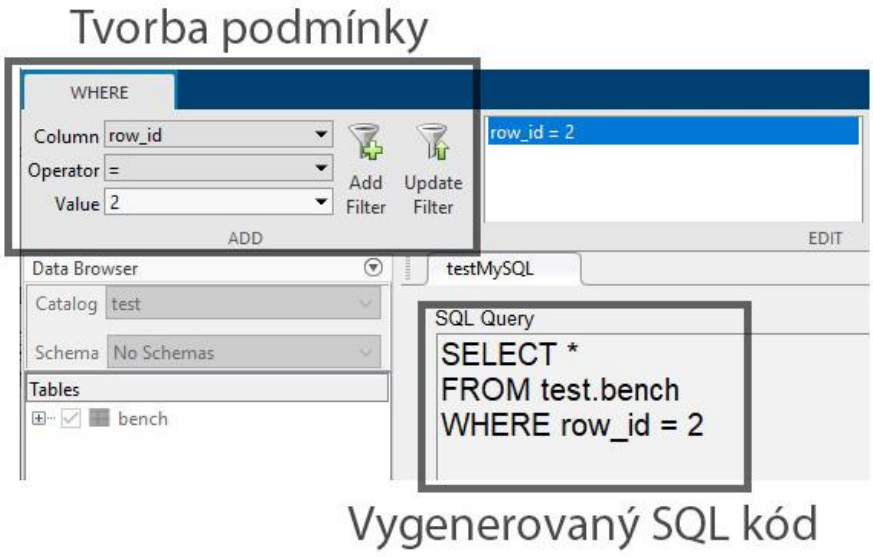
Import dat je opět možný jak pomocí příkazového řádku, tak pomocí Database Explorer aplikace. Pro import dat z databáze pomocí příkazového řádku slouží několik funkcí. První je *sqlread*. Jejím argumentem je pouze proměnná s připojením k databázi a název tabulky, kterou importuje celou. Další funkcí je *select* (obr. 13), kde je argumentem

připojení a SQL SELECT příkaz pro získání žádaných dat. Funkce *fetch* se používá téměř totožně, ale dovoluje větší kontrolu nad importem jednotlivých sloupců, za cenu pomalejšího přístupu k datům a větší náročnosti na paměť. Poslední možností je použít funkci *excecuteSQLScript*, která umožňuje import dat podle SQL skriptu. Další nastavení importu jako vynechání duplikátních dat, doplnění hodnot pro chybějící data a typy proměnných, jsou dostupná pomocí funkce *setoptions* [84] [89] [90] [91] [92] [93].

V Database Explorer je možné i bez znalosti jazyka sestavit SQL příkaz pomocí grafického rozhraní (obr. 14). Po vygenerování SQL příkazu je možné vytvořit Matlab skript pro automatizaci tohoto procesu. SQL kód je možné napsat i manuálně. Při práci s velkým objemem dat je doporučováno použít příkazový řádek. Podporované datové typy pro import a export jsou uvedeny v tabulce 2, včetně těch, specifických pro jednotlivé databáze. Database Toolbox automaticky provádí převod mezi datovými typy databází a datovými typy programu Matlab [84] [94].



Obrázek 13: Funkce *select*



Obrázek 14: Tvorba SQL příkazu

Databáze	Podporované datové typy
Všechny	BOOLEAN
	CHAR
	DATE
	DECIMAL
	DOUBLE
	FLOAT
	INTEGER
	NUMERIC
	REAL
	SMALLINT
	TIME
	TIMESTAMP
MS SQL Server	NTEXT
	TEXT
	VARCHAR(MAX)
	CHAR(8000)
	NCHAR(4000)
	NVARCHAR(MAX)
	TINYINT
MySQL	MEDIUMTEXT
	LONGTEXT
	TINYINT
Oracle	LONG
	VARCHAR2(4000)
	NCHAR(2000)
	NVARCHAR2(4000)

Tabulka 2: Datové typy podporované v DB Toolbox [84]

Při získávání dat z databází je třeba si dát pozor na následující limitace [94]:

- V názvu tabulky nebo sloupce nesmí být použity uvozovky.
- V názvu sloupce nesmí být použity rezervované výrazy jako DATE a TABLE.
- Při importu z Microsoft Access je ideální se vyhnout názvům sloupců obsahující mezery. Import je možný, ale každý název sloupce obsahující mezeru je třeba dát do uvozovek.

K dispozici je také několik funkcí pro práci s velkými objemy dat. Funkce *databaseDatastore* funguje stejně jako *fetch*, ale je díky ní možné používat analytické nástroje Matlabu i na velké množství dat, která by se jinak celá nevešla do paměti. Jsou použity „tall arrays“, tedy pole, která nejsou omezena počtem řádků, protože Matlab na pozadí automaticky nakládá pouze s menšími úseky těchto dat. Jakmile uživatel použije funkci *gather* nebo *write*, která výsledek načte do paměti, popř. zapíše na disk, jsou

požadované operace provedeny. Do té doby jsou výpočty zdrženy, aby byla práce s těmito poli svižná. Kdyby byly všechny operace prováděny okamžitě, jejich dokončení by mohlo zabrat několikanásobek času, protože by byly prováděny po jedné. Matlab se automaticky snaží minimalizovat počet průchodů daty. Pro import velkých dat se také hodí funkce *createConnectionForPool*, která umožňuje několik paralelních připojení ke stejné databázi. Pomocí funkce *splitsqlquery* je potom možné rozdělit SQL příkaz, který by zahrnoval import velkého množství dat a provádět tyto pod úkoly paralelně [84] [95] [96].

5.3 Práce s daty a export

Aplikace Database Explorer nenabízí funkce pro úpravu dat. Pro vložení dat do databáze však slouží několik funkcí. Funkce *sqlwrite* přidá požadovaná data do vybrané tabulky připojené databáze. Jejím argumentem je připojení, jméno tabulky a data. V případě, že tabulka neexistuje, bude automaticky vytvořena. Pro vložení většího množství dat může být výhodné vytvořit soubor obsahující všechna data a vkládat je z něj pomocí funkce *execute*. Tam je možné využít specifických příkazů, které jsou pro dané databázové systémy při zapisování velkých dat nejrychlejší. V případě *sqlwrite* se provádí SQL příkaz INSERT INTO. U *execute* lze s výhodou použít T-SQL příkaz BULK INSERT v případě MS SQL Server, příkaz LOAD DATA INFILE v případě MySQL nebo COPY FROM v případě PostgreSQL. Funkce *datainsert* a *fastinsert* jsou další funkce DB Toolboxu sloužící k vkládání dat do databází, jejich používání ale není doporučeno a v některé z následujících verzí budou odstraněny [84] [97] [98] [99] [100] [101] [102] [103].

Pro přepsání stávajících dat slouží funkce *update*. Jako argument přijímá připojení, název tabulky a sloupců, data samotná a podmínku WHERE. Je možné upravit více záznamů i sloupců zároveň za použití více podmínek. V případě, že je vypnutý *AutoCommit* pro dané připojení, lze tyto změny vrátit pomocí funkce *rollback*. Nebo je možné změny permanentně potvrdit funkcí *commit*. Argumentem obou funkcí je pouze připojení [104] [105].

Další práci s daty lze provést několika způsoby. S daty importovanými do workspace je samozřejmě možné provádět libovolné operace, které pro ně Matlab nabízí. A potom tato data opět zapsat do databáze. Krom toho je možné skrze Matlab provádět SQL příkazy pomocí funkce *execute*. Je možné použít i připravené příkazy (prepared statements). To jsou SQL příkazy, předem připravené pro opětovné použití. Místo parametrů, které se budou při jejich používání měnit v nich je znak „?“, který bude nahrazen požadovanou hodnotou. Použití těchto příkazů je efektivnější v případě, kdy se často opakují, protože se po prvním použití mezi serverem a klientem přenáší už jen tyto parametry. Zároveň jejich použití zvyšuje bezpečnost databáze proti jednomu z nejrozšířenějších útoků. Brání proti pokusům o SQL injection, kdy je do SQL příkazů vložen škodlivý kód. Uložené procedury jsou jim podobné. K jejich implementaci je však potřeba znát programovací jazyk, který je používán databázovým serverem. Jejich výhodou oproti prepared statements je možnost je okamžitě použít i po opětovném připojení k databázi. Prepared statements se totiž musí po každém připojení znovu kompilovat, to může zabrat delší dobu než jedno provedení tohoto příkazu, a nemusí se

tak vždy vyplatit. Volat uloženou proceduru je možné pomocí funkce *runstoredprocedure* [101] [106] [107] [108] [109] [110] .

V Matlabu je též možné použít funkce z Database Toolboxu při vytváření samostatně spustitelných aplikací v Matlab Application Compiler nebo s funkcí *mcc*. Takto zkompilovaný program je potom možné spustit i na počítačích, kde není Matlab nainstalovaný. Při práci s JDBC driverem je nutné ho ke zkompilované aplikaci přidat [84].

Licenci k Database Toolbox je možné zakoupit buď na jeden rok nebo trvale. Přehled edicí a cen, které jsou dostupné na stránkách MathWorks, je v tabulce 3. Ta zahrnuje pouze licence přímo k Database Toolbox. Licence k němu může být i součástí některých edic Matlabu [111].

Licence	**Trvalá**	**Roční**
Standard	1000 €	400 €
Education	200 €	100 €

Tabulka 3: Cena DB Toobox [111]

6 Matlab a JDBC

Database Toolbox využívá již existující standardy pro komunikaci s databázovými systémy – JDBC a ODBC. Vzhledem k tomu, že Matlab podporuje různé programovací jazyky včetně C, Python a Java, by mělo být možné nástroje pro práci s databázemi vytvořit vlastnoručně. Ideální pro tento účel bude JDBC – podporuje ho velké množství databází a jazyk Java je mezi podporovanými jazyky [112].

6.1 Připojení k databázi pomocí JDBC

Připojení k databázi pouze skrze JDBC není o mnoho složitější než za pomoci Database Toolbox. Pro úspěšné připojení stačí importovat Java třídy potřebné pro práci s SQL a mít odpovídající JDBC driver. U verzí JDBC starších než JDBC 4.0 bylo třeba ho explicitně registrovat, nyní ho Driver Manager detekuje a načte automaticky. Pro samotné připojení slouží metoda *getConnection* třídy DriverManager. Stejně jako při využití toolboxu, je třeba specifikovat parametry připojení jako URL adresu serveru, port, uživatelské jméno a heslo [113] [114] [115] [116] [117].

Při použití JDBC v Matlabu je ovšem výhodnější se použití třídy DriverManager vyhnout. Usnadňuje sice práci s větším množstvím ovladačů/typů databází, ale při jeho vynechání se předejde problémům s dynamickou cestou *classpath*, která specifikuje umístění uživatelských tříd. Některé JDBC ovladače s dynamickou cestou nefungují. Při přímém použití ovladače ho stačí umístit do aktuální složky Matlab projektu. Po úspěšném připojení zůstane v Matlab workspace objekt *JDBC4Connection*, který reprezentuje připojení k dané databázi [118].

6.2 Import dat pomocí JDBC

Po úspěšném připojení je možné s databází dále pracovat. Pro získání dat lze použít následující postup. Nejdříve se pomocí metody *createStatement* vytvoří objekt *Statement*, což je rozhraní, které reprezentuje SQL příkaz. Poté je příkaz pomocí metody *executeQuery* proveden. Tím je vygenerován objekt *ResultSet*, ve kterém je tabulka vybraných dat. Pro orientaci v tabulce slouží kurzor ve formě pointeru, který ukazuje na konkrétní řádek v *ResultSet* objektu. Jeho pohyb je umožněn metodami v něm definovanými. Jeho počáteční pozice je před prvním řádkem tabulky. Pro pohyb kurzorem o řádek dál slouží metoda *next*, která při posunu vrátí logickou 1 a v případě, že se posune za poslední řádek, vrátí logickou 0. Pro posun na (za) začátek slouží (*before)First*, pro posun na (za) konec (*after)Last*. Kurzor je možné posunout také relativně k jeho současné pozici s *relative(int row)* nebo na konkrétní řádek s *absolute(int row)*. Pro získání hodnot bude nyní využito metod definovaných v *ResultSet*. Metoda *getObject(int columnIndex)* vrátí hodnotu konkrétní buňky v aktuálním řádku a sloupci. Hodnota je vrácena ve formě Java objektu, jehož datový typ byl převeden

z odpovídajícího datového typu databáze. Argumentem je index sloupce (číslováno od 1) nebo jeho jméno. Použití indexu sloupce je většinou efektivnější a umožní použití kódu na různé tabulky. Aby byly hodnoty dostupné v Matlab workspace, je třeba aby byly převedeny do jeho datových typů a umístěny do odpovídající struktury. Ta je připravena s pomocí informací z metadat, získaných metodou *getMetaData*. Nesou informace o počtu a vlastnostech sloupců. Díky kurzoru je znám počet řádků, z metadat je zjištěn počet sloupců a jejich typy. Může tedy být vytvořena struktura o stejných rozměrech připravená pro zápis dat z *ResultSet.* V Matlabu jsou potom po řádcích vyplňovány jednotlivé sloupce, dokud kurzor nedojde za poslední řádek. Výsledná struktura je převedena na tabulku, která je dostupná ve workspace. Když už není objekt *Statement* používán, je dobrou praxí ho pomocí metody *close* uzavřít a uvolnit tak systémové prostředky, které využíval [119] [120] [121] [122] [123] [124] [125] [126].

Pro tento proces byla vytvořena Matlab funkce *sqlfetch(sqlprikaz)*. Jejím argumentem je SQL SELECT příkaz.

6.3 Práce s daty a export pomocí JDBC

Protože JDBC umožňuje provádět SQL příkazy, je možné ho použít i pro úpravu dat. První krok po připojení k databázi je stejný jako u čtení dat. Opět je použita metoda *createStatement* pro vytvoření *Statement* objektu. Následně je použita metoda *executeUpdate*, která slouží pro provedení SQL příkazů INSERT, UPDATE, DELETE. S nimi je možné vkládat nová data, měnit existující záznamy nebo vytvářet a mazat celé tabulky. Pro úpravu dat v databází pomocí JDBC je tedy třeba změny vyjádřit v jazyce SQL. Metoda *executeUpdate* vrátí celé číslo, je to počet řádků změněných provedeným příkazem. S daty importovanými do Matlab workspace je opět možné nakládat jako s libovolnými jinými daty a provádět s nimi požadované operace. Pro export dat z workspace je ale třeba vytvořit funkci, která změny vyjádří v SQL, nebo příkazy napsat manuálně. Příkazy je možné provádět po skupinách. Po připojení je třeba nastavit vlastnost *setAutoCommit* na false a použít metodu *createStatement.* Poté je možné přidávat příkazy do fronty pomocí metody *addBatch*. Skupina příkazů je provedena metodou *executeBatch*. Pro opakované příkazy je také možné použít prepared statement. Díky metodě p*repareStatement* může být příkaz předem zkompilován a uložen v objektu *PreparedStatement*, potom opakovaně proveden s vyšší efektivitou. Volání uložených procedur je možné díky rozhraní *CallableStatements*. Nejdříve se připraví pomocí metody *prepareCall*, poté je možné ji provést s pomocí metody *execute* a potvrdit metodou *commit* [125] [127] .

Pro úpravu dat byla vytvořena Matlab funkce *sqledit(sqlprikaz)*. Jejím argumentem je SQL příkaz.

7 Návrh databáze

7.1 Obecný přístup k návrhu

Fáze návrhu databáze je extrémně důležitá pro to, aby systém správně plnil svůj účel a práce s ním byla rychlá a uživatelsky přívětivá. Před realizací je výhodné věnovat dostatek času plánování, později ve vývoji můžou být zásadní změny nemožné nebo velmi časově náročné [128].

Proces návrhu databáze by měl zahrnovat tyto činnosti [129]:

- Analýza
- Organizace a návrh
- Implementace

Při analýze je třeba zjistit, jaké jsou potřeby projektu a jaká všechna data zahrnuje. Dále je třeba zjistit, které skupiny uživatelů budou s databází pracovat a návrh přizpůsobit jejich potřebám. Např. zákazníci a zaměstnanci téže společnosti budou mít na systém velmi odlišné nároky. Také je třeba brát v úvahu již existující systémy a jejich případnou integraci do nového projektu. Když je známo, jaká data budou v novém systému a jak s nimi bude nakládáno, je možné se posunout do dalšího stádia – samotného návrhu. Zde dojde na definici uspořádaní dat do tabulek a záznamů v nich. Jedná se o tzv. logický model databáze. Ten se ještě neimplementuje a může proběhnout i na papíře. Pro zjednodušení této fáze jsou k dispozici i nástroje jako sqldbm.com, kde je možné databázi navrhnout přímo v prohlížeči pomocí grafického rozhraní. Databázi je pak možné s tímto modelem vytvořit bez nutnosti napsat jediný řádek kódu. Tato fáze návrhu je nejkomplikovanější a zužitkují se informace z předchozí analýzy. Určení databáze by mělo mít zásadní vliv na to, jakým způsobem budou v databázi data uspořádána. Při návrhu tabulek je možné použít principy normalizace dat, které sníží počet duplicitních dat a předejdou problémům při jejich editaci. Ovšem v případě, kdy je prioritou co nejrychlejší přístup k datům, se mohou redundantní data stát žádoucí. V této fázi je také třeba vyjádřit vztahy mezi daty pomocí primárních klíčů. Potom je na řadě implementace tohoto modelu v konkrétním databázovém systému. Tato část zahrnuje přípravu hardwaru, instalaci systému, případnou migraci dat z ostatních systémů a testování. Fáze testování bývá dle Louise Davidsona (2016) podceňovaná a často se na ní šetří. Je třeba systém otestovat a porovnat ho s požadavky, které vznikly na začátku jeho návrhu. V opačném případě jeho nedostatky odhalí až jeho koncoví uživatelé a změny na již používaném systému budou opět obtížnější [130] [131] [132] [129] [133].

7.2 Návrh databázové struktury pro automatizaci simulací

Jedním z cílů praktické části práce bylo automatizovat provádění a ukládání výsledků simulace. Struktura databáze pro toto využití je přímočará. Byla vytvořena primární tabulka a pro každou proběhlou simulaci bude vytvořena jedna tabulka s výsledky. V primární tabulce je obsaženo identifikační číslo simulace, které je zároveň primárním klíčem tabulky. Primární tabulka dále obsahuje vstupní parametry simulací, údaje o tom, zda byla simulace řešena a dokončena. Pokud byla dokončena, obsahuje i název tabulky s výsledky dané simulace. Schéma primární tabulky je znázorněno v tabulce 4. Tabulky s výsledky potom obsahují index řádku, výsledky simulace a identifikační číslo simulace. Schéma tabulky s výsledky je v tabulce 5. Datové typy jsou platné pro systém MySQL.

Sloupec	sim_id	n	parametry simulace	taken	done	results
Datový typ	INT		DOUBLE	TINYINT(1)		VARCHAR(30)

Tabulka 4: Primární tabulka simulací v MySQL

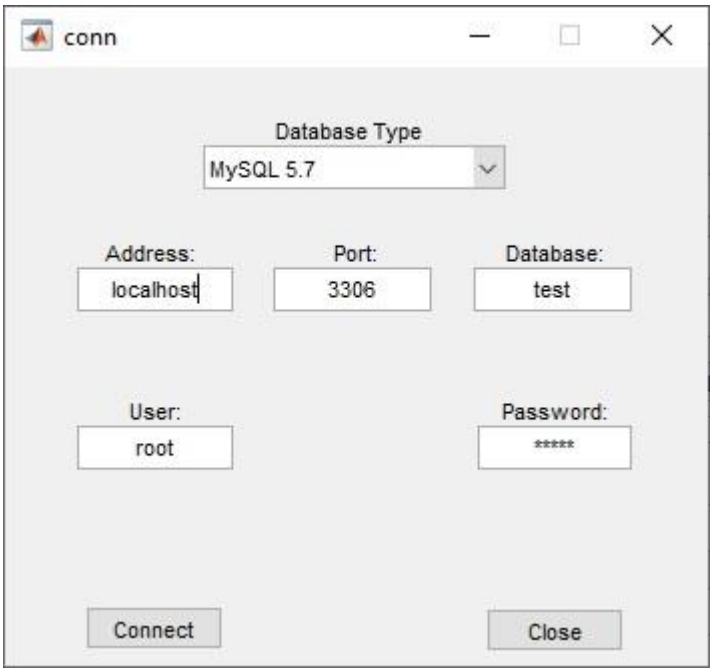
Sloupec	row_id	výsledky	sim_id
Datový typ	INT	DOUBLE	INT

Tabulka 5: Tabulka pro výsledky simulace v MySQL

8 Demonstrace výsledků

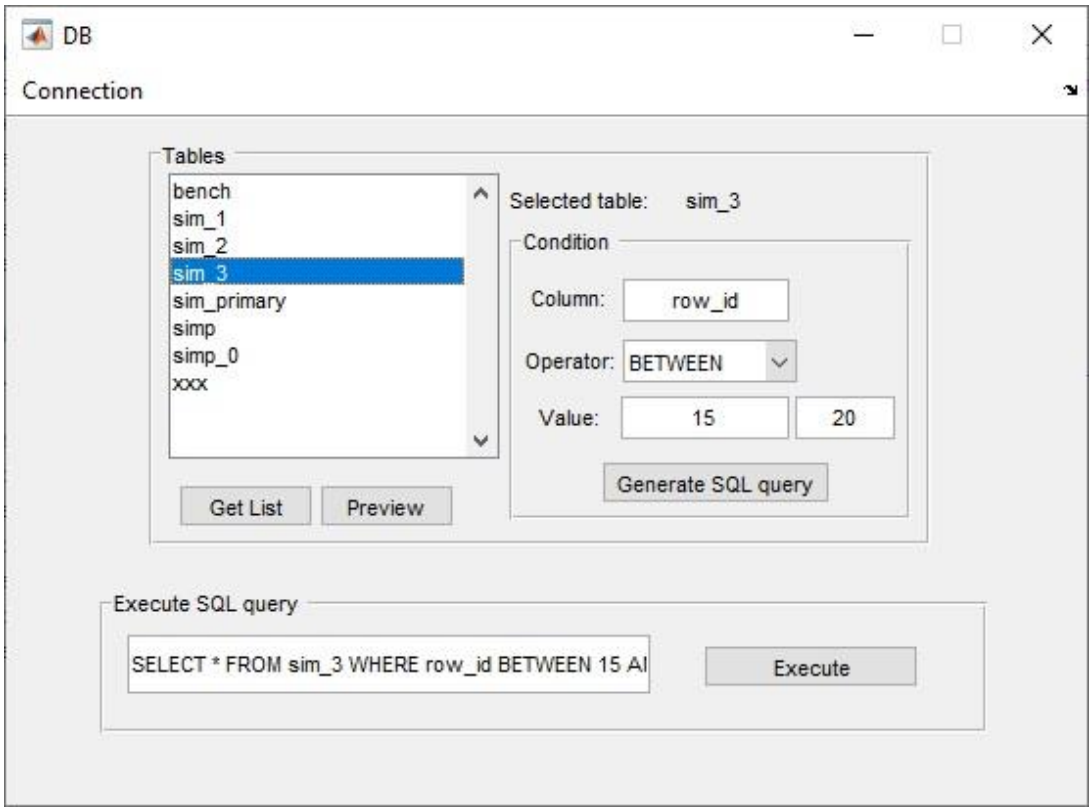
8.1 GUI

Pro zjednodušení práce s databázemi bylo vytvořeno uživatelské rozhraní v programu Matlab (GUIDE). Dialogové okno pro připojení k databázovému serveru je na obr. 15. Při připojování je na výběr z několika různých databázových systémů. V případě, že uživatel nezná název databáze, ke které se chce připojit, je možné toto pole nechat prázdné a zobrazit si seznam dostupných databází na serveru. Po vybrání konkrétního systému je automaticky doplněn jeho defaultní port. Po vyplnění všech parametrů a úspěšném připojení je objekt s připojením uložen do workspace.

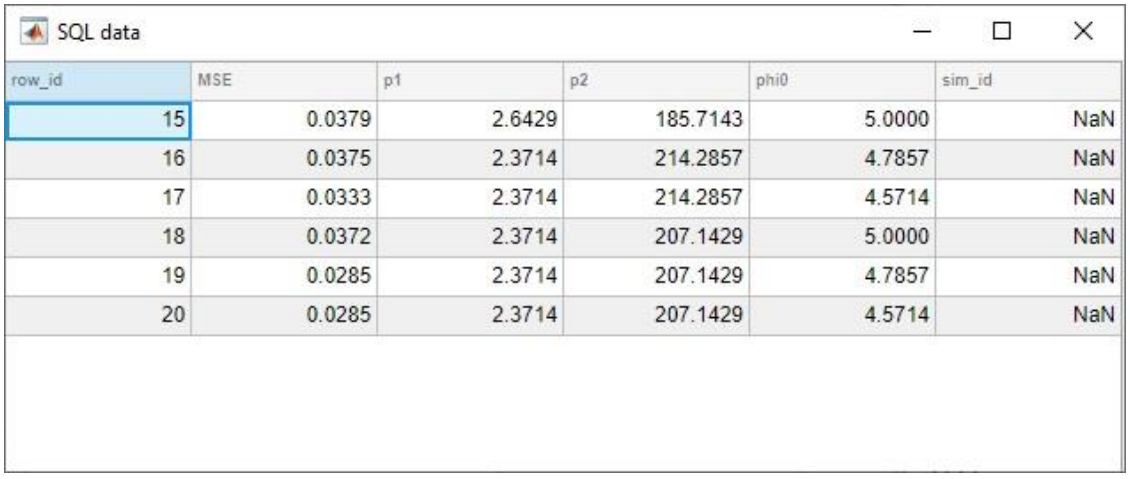


Obrázek 15: Okno pro připojení k databázi

Rozhraní hlavního okna je na obr. 16. Umožňuje vypsat seznam všech tabulek v připojené databázi. Po vybrání tabulky je možné si zobrazit náhled celé tabulky. Data v tabulce je možné filtrovat dle několika kritérií. Po zadání podmínky je automaticky vygenerován SQL příkaz, který podle ní data vybere. Rozhraní je tedy možné používat bez znalosti jazyka SQL. Z vygenerovaných příkazů je také patrné, jak SQL příkaz SELECT funguje, takže je možné se naučit jeho syntaxi. Data jsou poté automaticky importována do workspace a je možné s nimi dále pracovat. K dispozici je také řádek, kam je možné SQL příkaz zadat manuálně a poté ho provést. Uživatelské rozhraní při náhledu dat v tabulce je na obr. 17.



Obrázek 16: Hlavní okno programu



row_id	MSE	p1	p2	phi0	sim_id
15	0.0379	2.6429	185.7143	5.0000	NaN
16	0.0375	2.3714	214.2857	4.7857	NaN
17	0.0333	2.3714	214.2857	4.5714	NaN
18	0.0372	2.3714	207.1429	5.0000	NaN
19	0.0285	2.3714	207.1429	4.7857	NaN
20	0.0285	2.3714	207.1429	4.5714	NaN

Obrázek 17: Náhled vybraných dat

8.2 Automatizace simulace

Vybraná simulace se týká pohybu kyvadla s viskózním tlumením. Odhady ze vstupních parametrů jsou vyhodnocovány pomocí střední kvadratické chyby. Pouze výsledky, které se skutečné hodnotě dostatečně přiblíží, jsou považovány za platné. Cílem je umožnit nastavení těchto vstupních parametrů skrze databázi a kvůli jejich časové náročnosti rozdělit provedení simulací mezi více počítačů. Výsledky je třeba zapsat do databáze. Pro automatizaci tohoto procesu byl vytvořen Matlab skript. Před jeho spuštěním je třeba se připojit k databázi. Skript potom v daném časovém intervalu kontroluje, zda v primární tabulce nejsou zadání, která ještě nebyla řešena. Když je nalezeno volné zadání, je do příslušného řádku primární tabulky zapsáno, že je úkol řešen a z databáze jsou staženy vstupní parametry pro simulaci. Skript poté spustí samotnou simulaci. Jakmile dostane výsledky, vybere z nich pouze ty, které jsou dostatečně přesné. Pro jejich vyhodnocení je použita střední kvadratické chyba (MSE). Pro výsledky dané simulace je potom automaticky vytvořena nová tabulka a výsledky jsou do ní zapsány. Když je celý tento proces hotov, je do primární tabulky zapsáno, že je úkol hotov a název tabulky s výsledky simulace. Skript se po daném intervalu opakuje a opět vyhledává volná zadání. Náhled primární tabulky po dokončení všech úloh je na obr. 18.

sim_id	n	p1low	p1up	p2low	p2up	phi0low	phi0up	taken	done	results
1	5	0.2000	4	150	250	3.5000	6.5000	1	1	SIM_1
2	10	0.2000	4	150	250	3.5000	6.5000	1	1	SIM_2
3	15	0.2000	4	150	250	3.5000	6.5000	1	1	SIM_3

Obrázek 18: Náhled primárního okna simulací v grafickém rozhraní

9 Závěr

Hlavním cílem této práce bylo umožnit komunikaci programu Matlab s relačním databázovým systémem pro provádění simulace a rozdělení práce mezi více počítačů. Před samotnou realizací bylo třeba se s tématem databází seznámit. K tomu slouží několik prvních kapitol. Ty se věnují jak historii databázových systémů, tak současné situaci na trhu. Při průzkumu rozšíření jednotlivých systémů bylo třeba získat kvalitní zdroje dat a vybrat systémy, kterým se vyplatí věnovat větší pozornost. Na základě tohoto průzkumu byly vybrány tři systémy, které byly mezi sebou porovnány. Jedná se o MySQL, Microsoft SQL Server a PostgreSQL. Systémy byly porovnány na základě jejich vlastností a bylo provedeno měření rychlosti jednotlivých systémů. Pro toto měření byly zvoleny příkazy, které reprezentují využití databáze při ukládání výsledků ze simulací nebo měření.

Další část práce je věnována otázce implementace databází do aplikací. Jedná se o seznámení s univerzálními standardy ODBC a JDBC. Ty jsou podporovány naprostou většinou populárních databází. Umožňují pracovat s různými databázemi bez nutnosti provádět velké změny v kódu aplikace. Tyto standardy využívá i Matlab Database Toolbox, kterému je věnována samostatná kapitola. Jsou v ní shrnuty funkce, které nabízí. Poté se práce zabývá možnostmi, jak dosáhnout stejné funkčnosti jen díky integraci jazyka Java do programu Matlab a standardu JDBC. Jsou popsány postupy nutné k připojení, importu a exportu dat.

Fáze návrhu databázového systému je velmi důležitá. Obecnému přístupu k návrhu databáze je věnována jedna kapitola. V ní byla popsána i databázová struktura vytvořená pro řízení simulací a ukládání jejich výsledků.

Ve finále jsou prezentovány dva nástroje vytvořené pro práci s databázemi. Prvním je grafické rozhraní navržené pro přímočaré připojení k vybrané databázi. Umožňuje také import dat do programu Matlab, generování SQL příkazů pro výběr dat z tabulky a názorné zobrazení vybraných dat. Další nástroj slouží pro automatizaci provádění simulací. Počítač, na kterém skript běží, pravidelně databázi kontroluje. Když jsou v ní vstupní parametry, pro které simulace ještě neproběhla, je spuštěna a její výsledky jsou do databáze zapsány.

Mezi možnostmi, jak rozšířit nabídku funkcí vytvořeného grafického rozhraní, je např. přidání automatického generování více druhů SQL příkazů. Aktuálně je podporováno pouze vytváření SQL SELECT příkazů. Rozšíření o SQL příkazy UPDATE nebo INSERT INTO by umožnilo uživatelům upravovat data bez znalosti jazyka SQL.

Seznam použitých zdrojů

[1] *What Is SQL and How Is It Used?* [online]. the balance careers, 2019 [cit. 2020-04-10]. Dostupné z: https://www.thebalancecareers.com/what-is-sql-and-uses-2071909

[2] GRAD, Burton a Thomas BERGIN. Guest Editors' Introduction: History of Database Management Systems. *IEEE Annals of the History of Computing* [online]. 2009, **31**(4), 3-5 [cit. 2020-04-10]. DOI: 10.1109/MAHC.2009.99. ISSN 1058-6180. Dostupné z: http://ieeexplore.ieee.org/document/5370775/

[3] GRAD, Burton. Relational Database Management Systems: The Formative Years [Guest editor's introduction]. *IEEE Annals of the History of Computing* [online]. 2012, **34**(4), 7-8 [cit. 2020-04-10]. DOI: 10.1109/MAHC.2012.66. ISSN 1058-6180. Dostupné z: http://ieeexplore.ieee.org/document/6359704/

[4] HARRINGTON, Jan L. *Relational database design and implementation: clearly explained* [online]. Third edition. Amsterdam: Morgan Kaufmann Publishing is an imprint of Elsevier, 2009 [cit. 2020-04-10]. ISBN 978-012-3747-303.

[5] BLOOM, Madeline. *Data Integration Glossary* [online]. Washington, DC: U.S. Department of Transportation, 2001 [cit. 2020-04-10].

[6] MAURER, H. a N. SCHERBAKOV. 1. Network (CODASYL) Data Model. *Wayback Machine internet archive* [online]. [cit. 2020-04-10]. Dostupné z: https://web.archive.org/web/20060904190944/http://coronet.iicm.edu:80/wbtmaster/allcoursescontent/netlib/ndm1.htm

[7] BEYER, Kurt W. *Grace Hopper and the Invention of the Information Age*. 1. st Edition. MIT Press, 2009. ISBN 978-0262013109.

[8] *History of IMS: Beginnings at NASA: Introduction to IMS* [online]. IBM Corporation, 2010 [cit. 2020-04-11]. Dostupné z: https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.imsintro.doc.intro/ip0ind0011003710.htm

[9] *Introduction to IMS* [online]. IBM Corporation, 1990 [cit. 2020-04-11]. Dostupné z: https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.imsintro.doc.intro/intro.htm

[10] *Conference on Data Systems Languages records* [online]. University of Minnesota, Minneapolis: Charles Babbage Institute, 1959-1990 [cit. 2020-04-11]. Dostupné z: https://archives.lib.umn.edu/repositories/3/resources/35

[11] MELTZ, Dean, Rick LONG, Mark HARRINGTON, Robert HAIN a Geoff NICHOLLS. *An Introduction to IMS: Your Complete Guide to IBM's Information Management System*. IBM Press, 2005. ISBN 0131856715.

[12] CODD, E.F. *A Relational Model of Data for Large Shared Data Banks* [online]. Association for Computing Machinery, Inc. [cit. 2020-04-11]. Dostupné z: https://web.archive.org/web/20070714062917/http://www.acm.org:80/classics/nov95/toc.html

[13] E, F. CODD, E. F. Derivability, redundancy and consistency of relations stored in large data banks. *ACM SIGMOD Record* [online]. 2009, **38**(1), 17-36 [cit. 2020-04-11]. DOI: 10.1145/1558334.1558336. ISSN 0163-5808. Dostupné z: https://dl.acm.org/doi/10.1145/1558334.1558336

[14] KAUFMANN, Morgan. *Database Modeling and Design: Logical Design (The Morgan Kaufmann Series in Data Management Systems)*. Fifth edition. 2011. ISBN 0123820200.

[15] CONNOLLY, Thomas a Carolyn BEGG. *Database Systems: A Practical Approach to Design Implementation and Management*. Sixth Edition. University of the West of Scotland: Pearson, 2015. ISBN 978-1292061184.

[16] MEIER, Andreas a Michael KAUFMANN. *SQL & NoSQL Databases:: Models, Languages, Consistency Options and Architectures for Big Data Management*. 1. st Edition. Springer Vieweg, 2019. ISBN 9783658245481.

[17] GRAD, Burton. Relational Database Management Systems: The Business Explosion [Guest editor's introduction]. *IEEE Annals of the History of Computing* [online]. 2013, **35**(2), 8-9 [cit. 2020-04-11]. DOI: 10.1109/MAHC.2013.24. ISSN 1058-6180. Dostupné z: http://ieeexplore.ieee.org/document/6563082/

[18] BOYCE, R., D. CHAMBERLIN, M. HAMMER a W. KING. Specifying queries as relational expressions. *Proceedings of the 1973 meeting on Programming languages and information retrieval - SIGPLAN '73* [online]. New York, New York, USA: ACM Press, 1973, , 31-47 [cit. 2020-04-12]. DOI: 10.1145/951762.951765. Dostupné z: http://portal.acm.org/citation.cfm?doid=951762.951765

[19] CHAMBERLIN, Donald D. Early History of SQL. *IEEE Annals of the History of Computing* [online]. 2012, **34**(4), 78-82 [cit. 2020-04-12]. DOI: 10.1109/MAHC.2012.61. ISSN 1058-6180. Dostupné z: http://ieeexplore.ieee.org/document/6359709/

[20] CHAMBERLIN, Donald, Franco PUTZOLU, Patricia SELINGER et al. A history and evaluation of System R. *Communications of the ACM* [online]. **24**(10), 632-646 [cit. 2020-04-12]. DOI: 10.1145/358769.358784. ISSN 00010782. Dostupné z: http://portal.acm.org/citation.cfm?doid=358769.358784

[21] HELLERSTEIN, Joe a Anthony JOSEPH. System R & DBMS Overview. *Advanced Topics in Computer Systems* [online]. [cit. 2020-04-12]. Dostupné z: http://bnrg.cs.berkeley.edu/~adj/cs262/Lec_8_27a.html

[22] BERGIN, Thomas a Thomas HAIGH. The Commercialization of Database Management Systems, 1969–1983. *IEEE Annals of the History of Computing* [online]. 2009, **31**(4), 26-41 [cit. 2020-04-12]. DOI: 10.1109/MAHC.2009.107. ISSN 1058-6180. Dostupné z: http://ieeexplore.ieee.org/document/5370777/

[23] HALL, Mark a Erik GREGSEN. Oracle Corporation: GLOBAL CORPORATION. *ENCYCLOPÆDIA BRITANNICA* [online]. [cit. 2020-04-12]. Dostupné z: https://www.britannica.com/topic/Oracle-Corporation

[24] *Ingres* [online]. Carnegie Mellon Database Group, 2020 [cit. 2020-04-12]. Dostupné z: https://dbdb.io/db/ingres

[25] *Introduction to ODBMS: Definition* [online]. ODBMS.org [cit. 2020-04-27]. Dostupné z: http://www.odbms.org/introduction-to-odbms/definition/

[26] *Introduction to ODBMS: Short History* [online]. ODBMS.org [cit. 2020-04-27]. Dostupné z: http://www.odbms.org/introduction-to-odbms/history/

[27] AZIZ, Tariq, Ehsan-ul HAQ a Dost MUHAMMAD. Performance based Comparison between RDBMS and OODBMS. *International Journal of Computer Applications* [online]. 2018, **180**(17), 42-46 [cit. 2020-04-27]. DOI: 10.5120/ijca2018916410. ISSN 09758887. Dostupné z: http://www.ijcaonline.org/archives/volume180/number17/rao-2018-ijca-916410.pdf

[28] *The Object-Oriented Database System Manifesto* [online]. 1995 [cit. 2020-04-27]. Dostupné z: https://www.cs.cmu.edu/afs/cs/user/clamen/OODBMS/Manifesto/htManifesto/Manifesto.html

[29] BEYNON-DAVIES, Paul. *Database Systems* [online]. Third edition. Macmillan International Higher Education, 2017 [cit. 2020-04-27]. ISBN 9780230001077. Dostupné z: https://books.google.cz/books?id=TC5dDwAAQBAJ

[30] *About Stack Overflow* [online]. Stack Exchange Inc, 2020 [cit. 2020-05-10]. Dostupné z: https://stackoverflow.com/company

[31] Developer Survey Results 2017. *Stack Overflow Annual Developer Survey* [online]. Stack Exchange Inc, 2020 [cit. 2020-05-10]. Dostupné z: https://insights.stackoverflow.com/survey/2017#technology-_-databases

[32] Developer Survey Results 2018. *Stack Overflow Annual Developer Survey* [online]. Stack Exchange Inc, 2020 [cit. 2020-05-10]. Dostupné z: https://insights.stackoverflow.com/survey/2018#technology-_-databases

[33] Developer Survey Results 2019. *Stack Overflow Annual Developer Survey* [online]. Stack Exchange Inc, 2020 [cit. 2020-05-10]. Dostupné z: https://insights.stackoverflow.com/survey/2019#technology-_-databases

[34] 2020 Developer Survey. *Stack Overflow Annual Developer Survey* [online]. Stack Exchange Inc, 2020 [cit. 2020-05-10]. Dostupné z: https://insights.stackoverflow.com/survey/2020#technology-databases

[35] *Method of calculating the scores of the DB-Engines Ranking* [online]. solid IT gmbh, 2020 [cit. 2020-05-15]. Dostupné z: https://db-engines.com/en/ranking_definition

[36] *DB-Engines Ranking* [online]. solid IT gmbh, 2020 [cit. 2020-05-15]. Dostupné z: https://db-engines.com/en/ranking

[37] *DB-Engines Ranking - Trend Popularity* [online]. solid IT gmbh, 2020 [cit. 2020-05-15]. Dostupné z: https://db-engines.com/en/ranking_trend

[38] About Us. *Alexa* [online]. Alexa Internet, c1996-2020 [cit. 2020-05-15]. Dostupné z: https://www.alexa.com/about

[39] About us. *Datanyze: A zoominfo company* [online]. 2020 [cit. 2020-05-15]. Dostupné z: https://www.datanyze.com/about-us

[40] Alexa top 1M. *Datanyze: A zoominfo company* [online]. 2020 [cit. 2020-05-15]. Dostupné z: https://www.datanyze.com/market-share/databases--272/Alexa%20top%201M

[41] Database Market Share Report. *Datanyze: A zoominfo company* [online]. 2020 [cit. 2020-05-15]. Dostupné z: https://www.datanyze.com/market-share/databases--272/Datanyze%20Universe

[42] WHIFE, Chris. Technographics: A Guide to What, Why and How. *Leadiro* [online]. Leadiro™ Ltd, c2015-2020 [cit. 2020-05-15]. Dostupné z: https://www.leadiro.com/blog/technographics

[43] *DeveloperWeek: About* [online]. DeveloperWeek [cit. 2020-05-22]. Dostupné z: https://www.developerweek.com/about/

[44] *About Us: Fully managed hosting for MySQL, PostgreSQL, Redis™* and MongoDB® Database built by SQL and NoSQL database management experts.* [online]. [cit. 2020-05-22]. Dostupné z: https://scalegrid.io/about-scalegrid.html

[45] *2019 Database Trends – SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use* [online]. [cit. 2020-05-22]. Dostupné z: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/

[46] *Dries Buytaert: The history of MySQL AB* [online]. 2016 [cit. 2020-05-27]. Dostupné z: https://dri.es/the-history-of-mysql-ab

[47] Five Questions With Michael Widenius - Founder And Original Developer Of MySQL. *Wayback Machine internet archive* [online]. [cit. 2020-05-27]. Dostupné z: https://web.archive.org/web/20090313160628/http://www.opensourcereleasefeed.com/interview/show/five-questions-with-michael-widenius-founder-and-original-developer-of-mysql

[48] *GNU Operating System: Judge Saris defers GNU GPL Questions for Trial in MySQL vs. Progress Software* [online]. Free Software Foundation, Inc., 2014 [cit. 2020-05-27]. Dostupné z: https://www.gnu.org/press/2002-03-01-pi-MySQL.html

[49] ST AMANT, Kirk a Brian STILL. *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives*. 2007. DOI: 10.4018/978-1-59140-999-1. ISBN 9781591409991.

[50] *History of MySQL* [online]. Database Friends, 2014 [cit. 2020-05-27]. Dostupné z: https://databasefriends.blogspot.com/2014/02/history-of-mysql.html

[51] CABRAL, Sheeri a Keith MURPHY. *MySQL Administrator's Bible*. 1st Edition. Wiley Publishing, 2009. ISBN 9780470416914.

[52] Sun to Acquire MySQL. *Wayback Machine internet archive* [online]. MySQL AB [cit. 2020-05-27]. Dostupné z: https://web.archive.org/web/20080117192218/http://www.mysql.com/news-and-events/sun-to-acquire-mysql.html

[53] *MySQL Customers* [online]. Oracle Corporation and/or its affiliates, 2020 [cit. 2020-05-28]. Dostupné z: https://www.mysql.com/customers/

[54] *MySQL Products* [online]. Oracle Corporation and/or its affiliates, 2020 [cit. 2020-5-28]. Dostupné z: https://www.mysql.com/products/

[55] *What is MSSQL (Microsoft SQL)?* [online]. Atlantic.Net, 2020 [cit. 2020-06-01]. Dostupné z: https://www.atlantic.net/what-is-mssql/

[56] PRESTON, W. Curtis. *Backup & Recovery: Inexpensive Backup Solutions For Open Systems*. O'Reilly, 2007. ISBN 9780596102463.

[57] *ByteScout: MS SQL SERVER HISTORY AND ADVANTAGES* [online]. ByteScout, 2020 [cit. 2020-06-01]. Dostupné z: https://bytescout.com/blog/2014/09/ms-sql-server-history-and-advantages.html#1

[58] *Microsoft SQL Server 2000 DBA Survival Guide*. 2nd Edition. SAMS, 2002. ISBN 978-0-672-32468-0.

[59] *Editions and supported features of SQL Server 2019 (15.x)* [online]. Microsoft, 2019 [cit. 2020-06-01]. Dostupné z: https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-version-15?view=sql-server-ver15

[60] *Microsoft Customer Stories* [online]. Microsoft, 2020 [cit. 2020-06-01]. Dostupné z: https://customers.microsoft.com/en-us/home?sq=&ff=&p=0

[61] *A Brief History of PostgreSQL* [online]. The PostgreSQL Global Development Group, c1996-2020 [cit. 2020-06-04]. Dostupné z: https://www.postgresql.org/docs/8.4/history.html

[62] STONEBRAKER, Michael a Lawrence ROWE. The design of POSTGRES. *ACM SIGMOD Record* [online]. 1986, **15**(2), 340-355 [cit. 2020-06-04]. DOI: 10.1145/16856.16888. ISSN 01635808. Dostupné z: http://portal.acm.org/citation.cfm?doid=16856.16888

[63] *PostgreSQL: License* [online]. The PostgreSQL Global Development Group, c1996-2020 [cit. 2020-06-04]. Dostupné z: https://www.postgresql.org/about/licence/

[64] *A.M.TURING AWARD: Michael Stonebraker* [online]. Association for Computing Machinery, 2019 [cit. 2020-06-04]. Dostupné z: https://amturing.acm.org/award_winners/stonebraker_1172121.cfm

[65] *PostgreSQL: What is PostgreSQL?* [online]. StackShare, Inc., 2020 [cit. 2020-06-04]. Dostupné z: https://stackshare.io/postgresql

[66] *GeeksForGeeks: Data Replication in DBMS* [online]. [cit. 2020-06-05]. Dostupné z: https://www.geeksforgeeks.org/data-replication-in-dbms/

[67] *Dbengines: System Properties Comparison Microsoft SQL Server vs. MySQL vs. PostgreSQL* [online]. solid IT gmbh, 2020 [cit. 2020-06-05]. Dostupné z: https://db-engines.com/en/system/MySQL%3BMicrosoft+SQL+Server%3BPostgreSQL

[68] *Compare SQL Server, MySQL and PostgreSQL Features* [online]. Edgewood Solutions, 2018 [cit. 2020-06-05]. Dostupné z: https://www.mssqltips.com/sqlservertip/5745/compare-sql-server-mysql-and-postgresql-features/

[69] *Maximum capacity specifications for SQL Server* [online]. Microsoft, 2020 [cit. 2020-06-05]. Dostupné z: https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?view=sql-server-ver15

[70] *8.4.6 Limits on Table Size* [online]. Oracle Corporation and/or its affiliates, 2020 [cit. 2020-06-05]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/table-size-limit.html

[71] *NTFS vs FAT vs exFAT* [online]. LSoft Technologies Inc., 2020 [cit. 2020-06-05]. Dostupné z: https://www.ntfs.com/ntfs_vs_fat.htm

[72] *Appendix K. PostgreSQL Limits* [online]. The PostgreSQL Global Development Group, c1996-2020 [cit. 2020-06-05]. Dostupné z: https://www.postgresql.org/docs/12/limits.html

[73] *W3schools.com: SQL SELECT Statement* [online]. Refsnes Data, c1999-2020 [cit. 2020-06-06]. Dostupné z: https://www.w3schools.com/sql/sql_select.asp

[74] *W3schools.com: SQL WHERE Clause* [online]. Refsnes Data, c1999-2020 [cit. 2020-06-06]. Dostupné z: https://www.w3schools.com/sql/sql_where.asp

[75] *W3schools.com: SQL UPDATE Statement* [online]. Refsnes Data, c1999-2020 [cit. 2020-06-06]. Dostupné z: https://www.w3schools.com/sql/sql_update.asp

[76] BEAL, Vangie. *API - application program interface* [online]. [cit. 2020-06-06]. ISSN Webopedia. Dostupné z: https://web.archive.org/web/20200614213602/https://www.webopedia.com/TERM/A/API.html

[77] GEIGER, Kyle. *Inside ODBC (Microsoft programming series)*. Microsoft Press, 1995. ISBN 978-1-55615-815-5.

[78] *Microsoft: What Is ODBC?* [online]. 2017 [cit. 2020-06-07]. Dostupné z: https://docs.microsoft.com/en-us/sql/odbc/reference/what-is-odbc?view=sql-server-ver15

[79] *Microsoft: The ODBC Solution* [online]. 2017 [cit. 2020-06-07]. Dostupné z: https://docs.microsoft.com/en-us/sql/odbc/reference/the-odbc-solution?view=sql-server-ver15

[80] *Microsoft: ODBC Architecture* [online]. 2017 [cit. 2020-06-07]. Dostupné z: https://docs.microsoft.com/en-us/sql/odbc/reference/odbc-architecture?view=sql-server-ver15

[81] *Introduction to JDBC* [online]. Javasoft [cit. 2020-06-07]. Dostupné z: https://condor.depaul.edu/elliott/513/projects-archive/DS513Fall98/project/Chess/JDBC.html

[82] *Java Community Process* [online]. Oracle Corporation and/or its affiliates, 2020 [cit. 2020-06-20]. Dostupné z: https://www.jcp.org/en/home/index

[83] *ORACLE: JDBC Overview* [online]. Oracle Corporation and/or its affiliates, 2018 [cit. 2020-06-07]. Dostupné z: https://web.archive.org/web/20180929051553/https://www.oracle.com/technetwork/java/overview-141217.html

[84] *MATLAB: Database Toolbox* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-07]. Dostupné z: https://www.mathworks.com/products/database.html

[85] *What Is SQLite?* [online]. [cit. 2020-06-07]. Dostupné z: https://www.sqlite.org/index.html

[86] *Choosing Between ODBC and JDBC Drivers* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-07]. Dostupné z: https://www.mathworks.com/help/database/ug/choosing-between-odbc-and-jdbc-drivers.html

[87] *Configuring Driver and Data Source* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-07]. Dostupné z: https://www.mathworks.com/help/database/ug/configuring-driver-and-data-source.html

[88] *Data Import Using Database Explorer App or Command Line* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/data-import-using-database-explorer-app-or-command-line.html

[89] *Sqlread* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.sqlread.html

[90] *Select* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/select.html

[91] *Fetch* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.fetch.html

[92] *ExecuteSQLScript* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.executesqlscript.html

[93] *Setoptions* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/database.options.sqlimportoptions.setoptions.html

[94] *Data Retrieval Restrictions* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/data-retrieval-restrictions.html

[95] *DatabaseDatastore* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/matlab.io.datastore.databasedatastore.html

[96] *CreateConnectionForPool* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-08]. Dostupné z: https://www.mathworks.com/help/database/ug/createconnectionforpool.html

[97] *14.4. Populating a Database* [online]. The PostgreSQL Global Development Group, c1996-2020 [cit. 2020-06-10]. Dostupné z: https://www.postgresql.org/docs/current/populate.html

[98] *BULK INSERT (Transact-SQL)* [online]. Microsoft, 2020 [cit. 2020-06-10]. Dostupné z: https://docs.microsoft.com/en-us/sql/t-sql/statements/bulk-insert-transact-sql?view=sql-server-ver15

[99] *LOAD DATA Statement* [online]. Oracle Corporation and/or its affiliates, 2020 [cit. 2020-06-10]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/load-data.html

[100] *Sqlwrite* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.sqlwrite.html

[101] *Execute* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.execute.html

[102] *Datainsert* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.datainsert.html

[103] *Fastinsert* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.fastinsert.html

[104] *Commit* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.commit.html

[105] *Rollback* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/help/database/ug/database.odbc.connection.rollback.html

[106] *Database Operations* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/help/database/database-operations.html

[107] *DIfference Between Stored Procedures and Prepared Statements?* [online]. Stack Exchange Inc., 2018 [cit. 2020-06-10]. Dostupné z: https://stackoverflow.com/questions/7296417/difference-between-stored-procedures-and-prepared-statements

[108] *PHP MySQL Prepared Statements* [online]. Refsnes Data, c1999-2020 [cit. 2020-06-10]. Dostupné z: https://www.w3schools.com/php/php_mysql_prepared_statements.asp

[109] *SQL Injection* [online]. Refsnes Data, c1999-2020 [cit. 2020-06-10]. Dostupné z: https://www.w3schools.com/sql/sql_injection.asp

[110] *SQL Stored Procedures for SQL Server* [online]. Refsnes Data, c1999-2020 [cit. 2020-06-10]. Dostupné z: https://www.w3schools.com/sql/sql_stored_procedures.asp

[111] *Pricing and Licensing* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/pricing-licensing.html?prodcode=DB&intendeduse=comm

[112] *MATLAB Interfaces to Other Languages* [online]. The MathWorks, Inc., c1994-2020 [cit. 2020-06-10]. Dostupné z: https://www.mathworks.com/support/requirements/language-interfaces.html

[113] *Connecting MATLAB and MySQL with the JDBC Driver* [online]. Stack Exchange Inc., 2014 [cit. 2020-06-12]. Dostupné z: https://stackoverflow.com/questions/24438359/connecting-matlab-and-mysql-with-the-jdbc-driver

[114] *Public class: JDBC4Connection* [online]. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA: MySQL AB, Sun Microsystems, c2002-2007 [cit. 2020-06-12]. Dostupné z: http://www.docjar.com/docs/api/com/mysql/jdbc/JDBC4Connection.html

[115] *Establishing a Connection* [online]. Oracle Corporation and/or its affiliates, 1995 [cit. 2020-06-12]. Dostupné z: https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html#:~:text=In%20previous%20versions%20of%20JDBC,sql.

[116] *Java Database Connectivity with 5 Steps* [online]. www.javatpoint.com, c2011-2018 [cit. 2020-06-12]. Dostupné z: https://www.javatpoint.com/steps-to-connect-to-the-database-in-java

[117] PAGE, Trevor. Java Imports. *Coders Campus* [online]. [cit. 2020-06-12]. Dostupné z: https://howtoprogramwithjava.com/java-imports/

[118] *PATH and CLASSPATH* [online]. Oracle Corporation and/or its affiliates, 1995 [cit. 2020-06-12]. Dostupné z: https://docs.oracle.com/javase/tutorial/essential/environment/paths.html

[119] *Interface ResultSetMetaData* [online]. Oracle Corporation and/or its affiliates, 2020 [cit. 2020-06-12]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/sql/ResultSetMetaData.html

[120] *Processing SQL Statements with JDBC* [online]. Oracle Corporation and/or its affiliates, 1995 [cit. 2020-06-12]. Dostupné z: https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html

[121] *Public class: ConnectionImpl* [online]. MySQL AB, Sun Microsystems, c2002-2007 [cit. 2020-06-12]. Dostupné z: http://www.docjar.com/docs/api/com/mysql/jdbc/ConnectionImpl.html#createStatement

[122] *Public class: JDBC4ResultSet* [online]. MySQL AB, Sun Microsystems, c2002-2007 [cit. 2020-06-12]. Dostupné z: http://www.docjar.com/docs/api/com/mysql/jdbc/JDBC4ResultSet.html

[123] *Public class: ResultSetImpl* [online]. MySQL AB, Sun Microsystems, c2002-2007 [cit. 2020-06-12]. Dostupné z: http://www.docjar.com/docs/api/com/mysql/jdbc/ResultSetImpl.html#getMetaData

[124] *Public class: ResultSetMetaData* [online]. MySQL AB, Sun Microsystems, c2002-2007 [cit. 2020-06-12]. Dostupné z: http://www.docjar.com/docs/api/com/mysql/jdbc/ResultSetMetaData.html

[125] *Public class: StatementImpl* [online]. MySQL AB, Sun Microsystems, c2002-2007 [cit. 2020-06-12]. Dostupné z: http://www.docjar.com/docs/api/com/mysql/jdbc/StatementImpl.html

[126] *Retrieving and Modifying Values from Result Sets* [online]. Oracle Corporation and/or its affiliates, 1995 [cit. 2020-06-12]. Dostupné z: https://docs.oracle.com/javase/tutorial/jdbc/basics/retrieving.html

[127] *Public class: ConnectionImpl* [online]. MySQL AB, Sun Microsystems, c2002-2007 [cit. 2020-06-12]. Dostupné z: http://www.docjar.com/docs/api/com/mysql/jdbc/ConnectionImpl.html#createStatement

[128] DAVIDSON, Louis. Ten Common Database Design Mistakes. *Redgate Hub* [online]. c1999-2020 [cit. 2020-06-15]. Dostupné z: https://www.red-gate.com/simple-talk/sql/database-administration/ten-common-database-design-mistakes/

[129] DAVIDSON, Louis a Jessica MOSS. *Pro SQL Server Relational Database Design and Implementation*. Apress, 2016. ISBN 978-1-4842-1973-7.

[130] *11 important database designing rules which I follow* [online]. Shivprasad koirala, 2012 [cit. 2020-06-15]. Dostupné z: https://www.codeproject.com/Articles/359654/11-important-database-designing-rules-which-I-fo-2

[131] *Database Design Tutorial: Learn Data Modeling* [online]. Guru99, 2020 [cit. 2020-06-17]. Dostupné z: https://www.guru99.com/database-design.html

[132] *Database Structure and Design Tutorial* [online]. Lucidchart, 2020 [cit. 2020-06-18]. Dostupné z: https://www.lucidchart.com/pages/database-diagram/database-design

[133] MCCALDIN, David. *Tutorial: Step by Step Database Design in SQL* [online]. In: . [cit. 2020-06-18]. Dostupné z: https://www.linkedin.com/pulse/tutorial-step-database-design-sql-david-mccaldin/

Seznam použitých zkratek

AMD	Advanced Micro Devices
ANSI	American National Standards Institute
API	Application Programming Interface
ARO	Army Research Office
BSD	Berkeley Software Distribution
CAD	Computer-Aided design
CAL	Client Access License
CAM	Computer-Aided Manufacturing
CASE	Computer-Aided Software Engineering
CL	Column Access Strobe Latency
CLI	Call Level Interface
COBOL	Common Business-Oriented Language
CODASYL	Conference/Committee on Data Systems Languages
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DB	Database
Db4o	Database For Objects
DBMS	Database Management System
FAT	File Allocation Table
GPL	General Public License
GUI	Graphical User Interface
GUIDE	Graphical User Interface Development Environment
HP-UX	Hewlett Packard Unix
IBM	International Business Machines Corporation
ICS	Information Control System
IEC	International Electrotechnical Commission
IMS	Information Management System
ISO	International Organization for Standardization

JDBC	Java Database Connectivity
JDK	Java Development Kit
MIT	Massachusetts Institute of Technology
MS	Microsoft
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NVMe	Non-Volatile Memory Express
ODBC	Open Database Connectivity
OS	Operating System
PCIe	Peripheral Component Interconnect Express
PHP	PHP: Hypertext Preprocessor
RAM	Random-Access Memory
SDL	Software Development Laboratories
SIGMOD	Special Interest Group on Management of Data
SQL	Structured Query Language
SQL/DS	Structured Query Language/Data System
SSD	Solid State Drive
SW	Software
TDS	Tactical Data Systems
T-SQL	Transact-Structured Query Language
URL	Uniform Resource Locator
USA	United States of America

Seznam příloh

Elektronické přílohy:

Příloha 1: JDBC drivery

Příloha 2: Návod pro instalaci databáze a spuštění skriptů

Příloha 3: Simulace

Příloha 4: Skripty