VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

GENERÁTOR TESTOVACÍCH DAT PRO RELAČNÍ DATABÁZE

TESTING DATA GENERATOR FOR RELATIONAL DATABASES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ BAŠTA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19230/2016/xbasta03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Bašta Jiří**
Obor: Informační technologie
Téma: **Generátor testovacích dat pro relační databáze**
Testing Data Generator for Relational Databases
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou tvorby testovacích dat pro SQL databáze a s existujícími nástroji.
2. Prostudujte problematiku efektivního vkládání velkého množství dat do reálných databází.
3. Navrhněte nástroj pro automatické generování a vkládání dat pro testovaný systém. Navrhněte formát šablon, podle kterých bude nástroj data generovat.
4. Po dohodě s vedoucím implementujte navržený nástroj na zvolené platformě. Implementujte rovněž podporu integritních omezení při vkládání dat.
5. Proveďte testování nástroje na vhodné testovací sadě a diskutujte dosaženou rychlost generování a vkládání dat.
6. Zhodnoťte dosažené výsledky a navrhněte možná rozšíření.

Literatura:

- Lacko, L.: PHP 5 a MySQL 5, Computer Press, 2007
- Chung, I., Bieman, J.M.: Automated Test Data Generation Using a Relational Approach, APIS 2007

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Burget Radek, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Testování systémů a aplikací je součástí jejich vývoje, a proto je důležité vkládat testovací data. Generátor testovacích dat dle předem vytvořené šablony slouží jako nástroj pro naplnění databázových tabulek testovacími daty. Vytvořená aplikace dovolí vývojářům nebo administrátorům efektivně testovat různé systémy jež využívají databázové úložiště. Generování dat probíhá dle předem vytvořená šablony jež definuje strukturu databázových tabulek. Následující text popisuje databázové systémy jako takové, různé způsoby tvorby a vkládání dat a postup implementace nástroje, který by tuto tvorbu testovacích dat umožňoval.

Abstract

Testing of systems and applications is part of their development and therefore, testing data generation is very important. Test data generator according to a previously created template is as a tool for inserting test data to the database tables. This software allows developers or administrators to effectively test various systems that use the database storage. Data generation is carried out according to a pre-created template that defines the structure of database tables. The thesis describes the related topics of database systems and the process of the apllication implementation.

Klíčová slova

geneorávní dat, generátor, SQL, MySQL, databázové systémy, databáze, testování, náhodná data, vývoj

Keywords

data generation, generator, SQL, MySQL, database systems, database, testing, random data, developing

Citace

BAŠTA, Jiří. *Generátor testovacích dat pro relační databáze*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Burget Radek.

Generátor testovacích dat pro relační databáze

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením vedoucího bakalářské práce pana Ing. Radka Burgeta, Ph.D. Další důležité informace mi poskytl můj bývalý vedoucí bakalářské práce Ing. Pavel Vampola. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Bašta
11. května 2017

Poděkování

Zde bych moc rád poděkoval mému vedoucímu, kterým je pan Ing. Radek Burget, Ph.D. a panu Ing. Pavlovi Vampolovi, za všechny odborné rady a konzultace, které mi velice pomohli s dokončení této práce.

Obsah

Kapitola 1

Úvod

Tato zpráva popisuje okruh ve kterém se dané téma bakalářské práce nachází, což jsou databázové systémy, různé možnosti vkládání a generování dat do databází, jejich výhody a nevýhody. Dále se zpráva zabývá návrhem a implementací nástroje, který by sloužil jako generátor testovacích dat do relační databáze pomocí předem vytvořené šablony.

Nástroj pro tvorbu testovacích dat, jehož návrh a implementaci naleznete v této práci bude generovat testovací data do databáze MySQL, dle předem vytvořené šablony. Nástroj je ve skutečnosti složen ze dvou aplikací. Kde první z nich je aplikace pro tvorbu šablony a druhou samotný nástroj pro vygenerování a vložení dat do databáze. Spolu s popisem nástroje se zde nachází kapitola o testování rychlosti generování dat pomocí různých způsobů vkládání dat.

Oproti konkurenční nástrojům, které nabízí tvorbu testovacích dat přímo ze svého rozhraní a nevyužívají žádné šablony, má nástroj, popsaný v této zprávě několik význačných výhod. Samotné vytváření šablony může být oproti přímému neplnění databáze možná poněkud zdlouhavější, avšak pro opakované využití stejné šablony pro naplnění rozdílnými daty, je to mnohem efektivnější. Stěžejní výhoda nástroje, se týká především jeho multiplatformní podpory, která v kombinaci se šablonou, dokáže vytvořit testovací data do databáze bez problémů i z konzole.

Uchovávání dat pro jejich následné využívání je velice běžná věc a v dnešní době existují výkonné a rozsáhle databázové systémy, které slouží pro uchování a správu dat. Tyto databázové systémy jsou velice často spojovány s aplikacemi jež vyžadují uchování nějakého rozsáhlého množství strukturovaných dat a jejich následnou správu. Při vývoj aplikací, ať už webových nebo počítačových, které využívají databázové systémy, je důležité jejich testování. Pro otestování aplikace, která využívá data z databáze, je nutné nejprve tato data vytvořit. Manuální tvorba testovacích dat ve větším měřítku je časově velice náročná. Kvůli tomuto důvodu existuje řada nástrojů, které slouží pro generování testovacích dat do různých druhů databází. Testovací data jsou generována náhodně, a proto jejich případná shoda s reálnými údaji je čistě náhodná. Díky tomuto bezpečnostnímu opatření se užívají k testování aplikace pouze testovací data, nikoliv reálná. Generování testovacích dat probíhá u většiny případů automatizovaně, záleží ovšem na konkrétním typu generátoru.

Kapitola 2

Databáze a databázové systémy

Pro jasnější představu o problematice tvorby testovacích dat je důležité si představit co jsou vlastně databázové systémy, jak fungují a z čeho se skládají. Také je velice užitečné vědět jak se vůbec vkládají samotná data do databází. V této kapitole lze nalézt teoretické pojednání o databázích obecně, jejich jednotlivých prvcích, k čemu se používají a jak fungují. Jsou zde vysvětleny například termíny jako entita a vztah. Dále se tu rozebírají do podrobnosti samotné relační databáze a s nimi úzce spjaté také relační diagramy tzv. ER diagramy. V neposlední řadě je zde zmínka o jazyku SQL, některé jeho konstrukce a příkazy.

2.1 Databáze

Databázi lze definovat jako uspořádanou množinu dat, která jsou uložena na nějakém médiu. Tato data jsou skladována ve strukturované a organizované formě, aby je bylo možné snadno a rychle využívat. Do databáze nepatří jenom samotná data, ale také prostředky, díky kterým s daty lze manipulovat. Spojením prostředků pro manipulaci s daty a se samotnými datovými informacemi vznikne celek, zvaný jako systém. Tento systém je nazýván jako *Systém řízení báze dat* (SŘBD) nebo anglicky též DBMS.

Data, jež jsou uložena v databázi, jsou údaje, která mají určitou vypovídací schopnost. Tato data mohou být určitým způsobem uspořádána (seřazena, například dle velikosti) a jsou k dispozici v různých formách (tabulky, grafy, zvukové signálky, aj.). Samotná data jsou obvykle rozdělena na dílčí údaje (atributy) a jsou uložena v jednotlivých tabulkách [1].

2.2 Entity a vztahy

Dalším velice důležitým pojmem je *entita*. Je to typ objektů ať už konkrétních (člověk, věc, místo), nebo abstraktních (různé katalogové položky). Základním faktem o entitě je, že je označena nějakým názvem (výrobky, zákazníci). Dalším neméně důležitým rysem je, že jednotlivé objekty dané kategorie, jsou navzájem rozlišitelné. Z toho důvodu je prováděna analýza, pomocí níž lze zjistit, co činí konkrétní objekty unikátní (identifikační číslo, název, unikátní výrobní kód, aj.) Jednotlivé entity většinou odpovídají prvkům z reálného světa které mezi sebou mohou mít určitý vztah.

Vazby (vztahy) jsou rozděleny do třech kategorií a to na vazbu typu **1:1**, kde jedna entita je svázána pouze s jednou další entitou. Dále vazbu **1:N**, kterou si lze představit například na reálné situaci, kde jeden zákazník může mít více kreditních karet, ale jedna karta nemůže být vlastněna více zákazníky. Poslední vazbu typu **M:N** lze demonstrovat například na studentech vysoké školy, kde jeden student může být zapsaný na více předmětech a zároveň jeden předmět může být navštěvován více studenty [**?**].

Existuje řada různých databází, které se mohou lišit například dle daného modelu databáze, což se odráží ve způsobu ukládání dat a vazeb mezi nimi. Pojem databázový model byl zaveden jako prostředek pro popis databáze. Dříve se velice často používaly modely typu **hierarchický** (založen na modelování hierarchie mezi entitami a to se vztahy podřazenost a nadřazenost) a **síťový** (založen na teorii grafů, jednotlivé uzly v grafu odpovídají entitám a orientované hrany mezi nimi definují vztahy). Později se tyto dva databázové modely ukázaly být nedostačující, a proto vznikl *relační model*, který se stal standardem a je dodnes velice hojně využíván [**?**].

2.3 Relační databáze

Tento typ databáze je založen na relačním modelu. Základní částí relačního modelu jsou tabulky, které reprezentují objekty reálného světa o kterých chceme ukládat nějaká data. Jednotlivé řádky tabulky jsou následně chápány jako záznamy. Objekty, reprezentované tabulkami, jsou logicky propojeny pomocí vztahů a vazeb, tzv. relací.

Entitu si lze představit jako libovolný objekt reálného světa, který je zachycen v datovém modelu. V tomto případě je například entita reprezentována jako záznam jedné nebo více tabulek.

Atribut, neboli také položka, je samostatný sloupec v tabulce. Může to být například jméno nebo příjmení dané osoby, nebo název města v případě adresy. Obecně se jedná o dílčí složku nějakého většího obnosu dat nebo objektu.

Záznamem je označován samotný řádek tabulky. V tomto záznamu jsou následně atributy (sloupce) a dohromady tvoří kompletní informace o dané entitě.

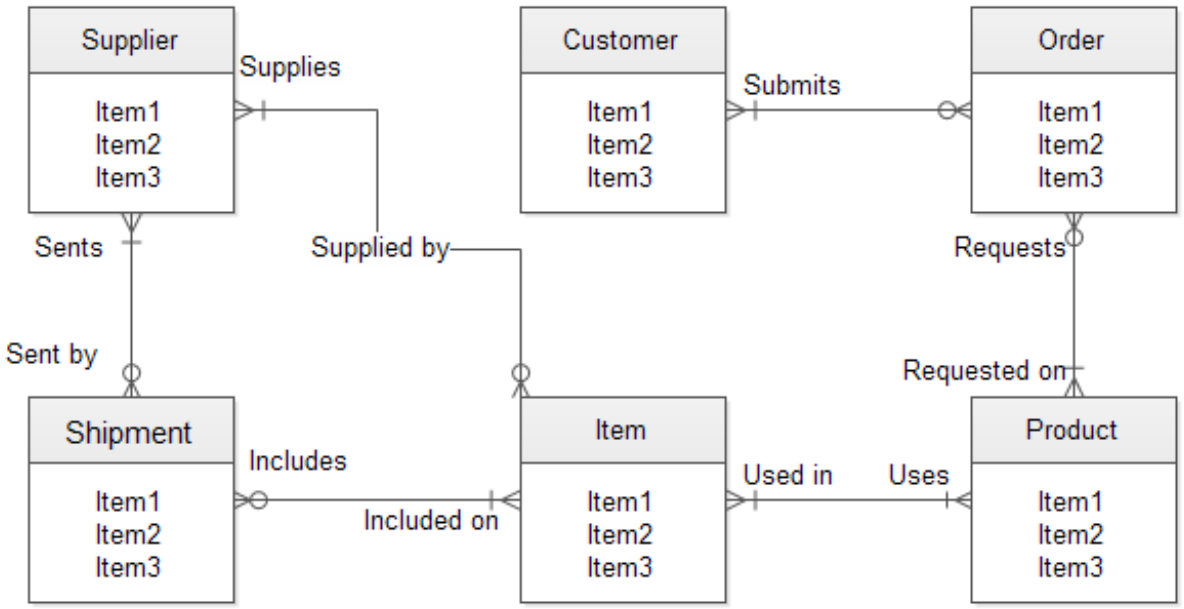
Primární klíč je atribut, jehož hodnota musí být v dané tabulce jedinečná. Dle hodnoty primárního klíče může probíhat vyhledávání nebo vytváření vztahů. Primárním klíčem může být například rodné číslo, nebo identifikační číslo nějakého pracovníka ve firmě.

Cizí klíč slouží pro vyjádření vztahů(relací) mezi databázovými tabulkami. Umožňuje identifikovat, které záznamy z různých tabulek spolu navzájem souvisí.

Databáze byly dříve tvořeny na míru v nějakém vyšším programovacím jazyce, jako například COBOL. Tyto databáze měly ovšem zásadní nedostatek - nebyly univerzální. Během vývoje relačních databází vzniklo několik jazyku jako byl QUEL a SEQUEL z čehož nakonec vznikl dnešní jazyk SQL. Relační databáze jsou proto dnes velice často spojovány s jazykem SQL, který je v těchto typech databází využíván [17].

2.4 Relační diagramy (ER diagram)

Před samotnou tvorbou databáze je důležité mít určitý návrh a analýzu. Pro návrh relační databáze se využívají relační diagramy, konkrétně se může jednat o *ER diagram* (Entity-relationship diagram). V těchto diagramech lze znázornit jednotlivé vztahy mezi entitami a vytvořit si tak obrázek o finální podobě databáze. Díky těmto diagramům v rané fázi vývoje lze zjistit a vyřešit problémy a logické chyby, kterou mohou nastat, ještě před samotnou tvorbou. Příklad relačního diagramu.



Obrázek 2.1: Příklad ER diagramu

2.5 Jazyk SQL

SQL (Strucuted Query Language) - standardizovaný strukturovaný dotazovací jazyk, který je používán jak pro práci se samotnými daty v relačních databázích, tak i celkově se samotnou datbází. Tento jazyk je v současné době rozšířen téměř do všech relačních databází a já nástupcem jazyka SEQUEL. Slouží k manipulaci se všemi daty, nebo entitami databáze[6].

Jednotlivé příkazy jazyka SQL lze rozdělit do čtyř základních skupin pro práci s databází:

- Příkazy pro manipulaci s daty, které slouží pro vybírání dat (SELECT) v databází, vkládání (INSERT), mazání (DELETE) dat z databáze, nebo pouhé upravování dat (UPDATE).
- Definice dat, jako vytváření nové tabulky (CREATE), rušení (DROP) nebo upravování (ALTER) stávající tabulky.
- Přístupová práva, slouží k udělování dané úrovně přístupu pro konkrétní uživatele, nebo celé skupiny uživatelů.

- Řízení transakcí slouží k ovládání obsluhy[1]. (START TRANSACTION, COMMIT, ROLLBACK)

2.6 Databáze SQL

Jak bylo uvedeno výše, relační databáze jsou v dnešní době velice rozšířené a to především pro jejich modulárnost díky využití jazyka SQL. Mezi nejznámější zástupce relačních databázových systémů využívající jazyk SQL patří MySQL, Oracle, Postgress, MS SQL a další. Pro webové použití databázových systému se nejčastěji využívá MySQL ve spojení s jazykem PHP. Především díky velkému využití databázového systému MySQL v praxi a jeho přenositelnosti, je tento systém zvolen k demonstrování nástroje pro generování testovacích dat do databáze.

2.7 Databáze noSQL

V tomto databázovém konceptu se pro ukládání a zpracování dat využívají jiné prostředky než tabulková schémata tradiční relační databáze. Databáze, jež nevyužívají SQL jsou často vysoce optimalizované. Výběr typu databázového systému se orientuje především dle řešeného problému, proto nelze říct, zdali jsou lepší databázové systémy SQL nebo noSQL . V současné době využití noSQL databází významně roste avšak existuje spousta překážek, kvůli kterým se tento typ databáze zatím plně nerozšířil, např. nepřítomnost plnohodnotné podpory transakčního modelu ACID[2] [14].

[1] Databázové transkace - http://lucie.zolta.cz/index.php/iformacni-systemy-databaze/70-zamykani
[2] Vlastnosti, jež musí databáze splňovat, aby byla konzistentní.

Kapitola 3

Tvorba testovacích dat

Jak již bylo zmíněno dříve, tvorba testovacích dat je nedílnou součástí při vývoji jakéhokoliv systému, jež využívá databázi. Problematika, jež souvisí s tvorbou testovacích dat se týká především generování vhodných dat a ve větším počtu jež by zároveň nezabralo příliš mnoho času. Využívání produkčních dat pro testovací účely je velice nevhodné a to především z důvodu bezpečnosti a ochrany osobních údajů. Kvůli tomuto problému se databáze při vývoji naplňují pouze testovacími daty. Testovací data mohou být v případě menších systémů zadávána manuálně. Při větších rozměrech systému, kde je nutné vytvořit větší množství záznamů, pro dokonalé otestování, je vhodné využít automatizovanou tvorbu dat.

Data pro testovací záznamy lze získat ze spousty různých oficiálních generátorů, jež mohou být volně ke stažení nebo umístěny online na internetu. Tyto generátory využívají různé, volně dostupné seznamy jmen, příjmení, měst, apod., ze kterých náhodně generují data. Způsob a forma vygenerovaných dat se liší s daným generátorem a zvoleným formátem dat dle druhu databáze pro nějž jsou určena [8].

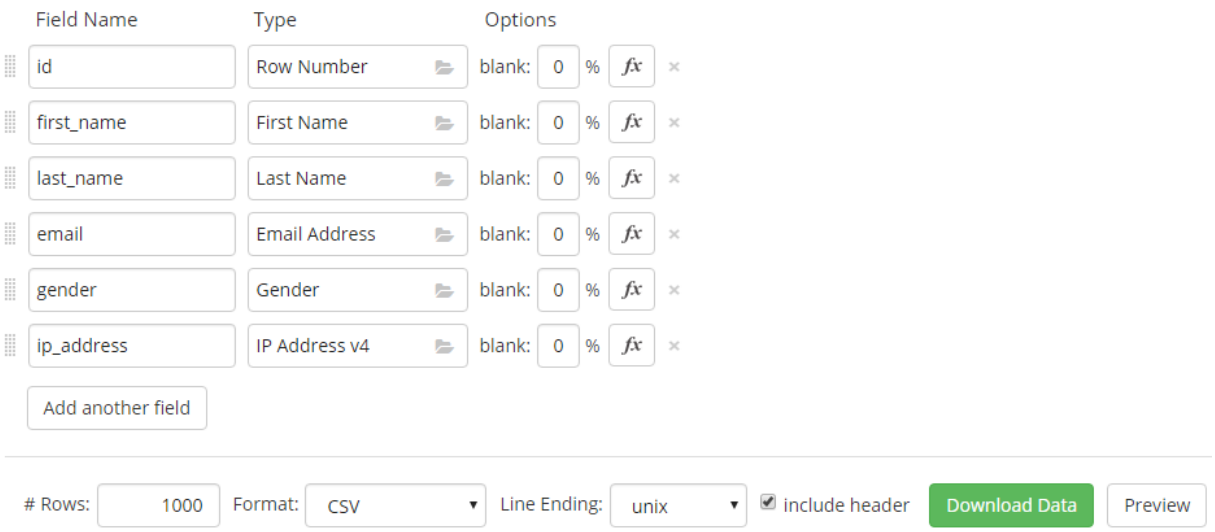
3.1 Dostupné generátory

Většina generátorů, jež mohou být volně dostupné nebo i placené, pracují způsobem, kde si uživatel manuálně vyplní formulář se strukturou jeho databáze (tabulek a atributů). Zvolí si typ generovaných testovacích dat ať už přímou kategorii, jakou může být náhodné jméno nebo příjmení nebo použije masku. Maskou je myšleno například regulární výraz, kterým specifikuje rozsah daného slova, konkrétní pozice, na kterých se mohou vyskytovat zvolené znaky, apod. Následně generátor vygeneruje testovací data. Výsledná forma, jež určuje způsob použití těchto dat může být u konkrétních generátorů velice odlišná. Některé generátory vytvoří pouze textové údaje, které je nutné ještě nějakým způsobem upravit a následně mohou být připravena pro import. Jiné generátory vytvoří přímo úseky kódů, například v jazyce SQL, které poté stačí vložit do databáze (do manuálního importu) a ta se automatizovaně naplní.

Existují ovšem i generátory, které se nainstalují na lokální počítač s databázovým systémem a následně generují testovací data do zvolené databáze. Tyto aplikace mají přímý přístup do databáze a jsou schopny analyzovat celou její strukturu a podle toho předvyplnit volby pro generování dat.

3.1.1 Mockaroo

Tento online webový generátor dokáže vygenerovat testovací data dle vyplněného formuláře se strukturou tabulky. Patří mezi nejlepší online generátory a to především pro jeho jednoduchost a rychlost. Vygenerovaná data lze uložit v různých formátech jako je CSV, XML, JSON, SQL, aj. Bohužel lze vygenerovat data pouze pro jednu tabulku současně. Data, vygenerovaná například ve formátu SQL jsou připravená pro import do databáze.[1]



Obrázek 3.1: Grafické rozhraní webového nástroje Mockaroo

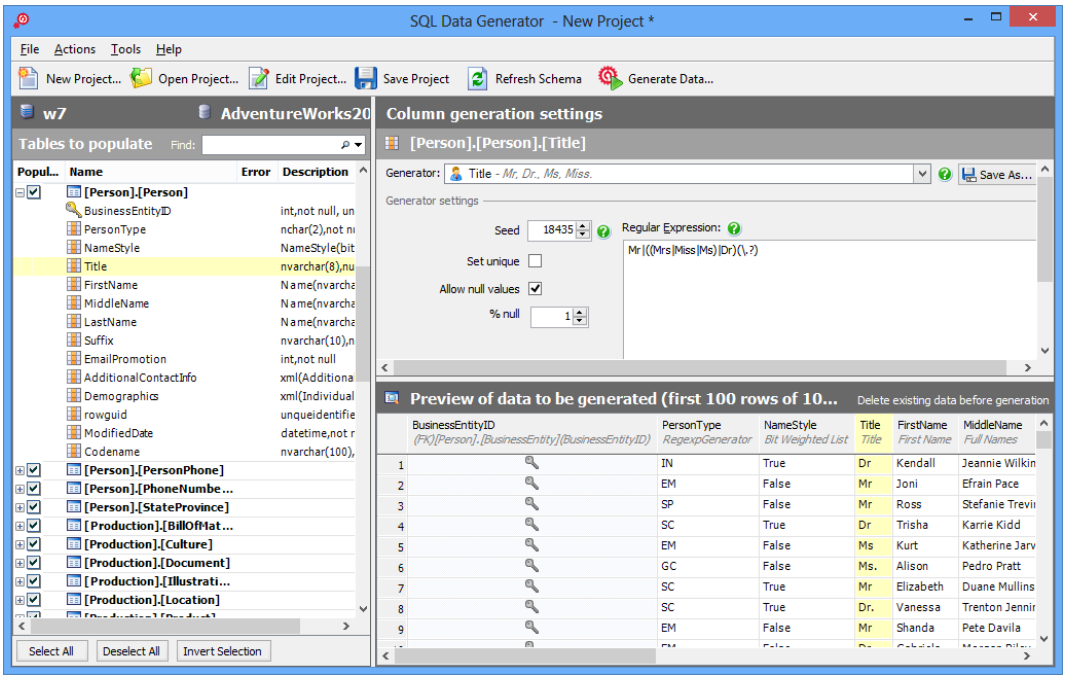
Online generátorů podobných tomuto existuje celá řada, například *generatedata.com* nebo *databasetestdata.com*. Všechny tyto generátory mají jedno společné a to je, že generují pouze surová data a uloží je do souboru. Tato data se pak vždy musí pomocí nějakého způsobu vložit do databáze. Způsob zpracování dat se liší od formátu v jakém jsou vytvořena. V případě SQL lze data vložit pomocí importu nebo manuálně vyjmout obsah a umístit do konzole databázového systému a vložit. Výhodou těchto online nástrojů je to, že jsou velice univerzální a to především pro jejich širokou škálu formátů do nichž data generovat.

Podstatnou nevýhodou těchto generátorů je, že pokud se použijí například na generování velkého množství dat do databázového serveru využívající SQL, tak bude dotaz na vložení těchto informací velice neefektivní a jeho provedení zabere dlouhou dobu. To je způsobeno minimální, nebo respektive žádnou, optimalizací dotazů sloužících pro vložení dat do databáze.

[1] Mockaroo - https://mockaroo.com/

3.1.2 SQL Data Generator

Aplikace pro generování testovacích dat pro SQL databázové systémy. Tento program je nutné nainstalovat na klientský počítač a připojit se do některé z vytvořených databází. Aplikace zobrazí celou strukturu databáze včetně všech tabulek a jednotlivých atributů. Pomocí grafického rozhraní lze zvolit jaké typy dat generovat do jednotlivých sloupečků tabulek. Lze využívat i regulární výrazy k upřesnění konkrétních testovacích dat. Aplikace následně automatizovaně naplní databázi bez jakékoliv nutnosti manuálního zadávání nebo pomocí importu.[2]



Obrázek 3.2: Grafické rozhraní aplikace SQL Data Generator

Tato aplikace je na dva týdny zdarma, poté je nutné si ji zakoupit. Aplikace je výborná pro komerční využití a větší firmy, které využijí její výkonnost a rozsáhlé možnosti konfigurace.

3.1.3 Shrnutí

Dostupné generátory testovacích dat, jež se nachází na trhu, ať už placené nebo zdarma, mají své výhody i nevýhody. Dvě společné a důležité vlastnosti, jež všechny tyto generátory mají jsou, že se skládají pouze z jedné jediné aplikace a také to, že mají uživatelské grafické rozhraní, které je nutné pro vytvoření nebo respektive "naklikání"jednotlivých informací o datech, jež se budou generovat. Uživatel jež bude chtít generovat testovací data, si musí zvolit typ dat, které chce generovat, kategorii, z jaké se budou data vybírat, množství dat, apod.

[2]SQL Data Generator - http://www.red-gate.com/products/sql-development/sql-data-generator/

Oproti tomu, generátor dat, popsaný v této zprávě je sestaven ze dvou různých aplikací, které fungují paralelně. To přináší značnou výhodou v jeho použitelnosti. Pro tvorbu šablony je grafické rozhraní nutné, jelikož je třeba si zvolit ze všech dostupných typů a kategorií, které chce mít uživatel vygenerované, jiný způsob tvorby šablony by byl velice neefektivní. Ovšem samotná aplikace pro tvorbu dat, již nevyžaduje grafické rozhraní, avšak i to se v aplikaci vyskytuje. Díky její multiplatformnosti, je možné ji bez problémů spustit například z konzole na Linuxu a naplnit tak velice snadno zvolenou databázi s předem vytvořenou šablonou. Stejně tak není problém aplikaci spustit pod systémem Windows a využít grafické rozhraní.

3.2 Způsoby tvorby dat a jejich optimalizace

Jednotlivé druhy databázových dotazů byly uvedeny výše. Pro zopakování, do téhle kategorie spadají dotazy na výběr dat z databáze, mazání, různé úpravy tabulky a především vkládání dat do databáze.

Existují různé způsoby a metody pro vkládání dat do zvolené databáze, ty se liší především v rychlosti vložení dat a míře optimalizace. V této kapitole budou popsány rozdíly mezi jednotlivými způsoby, pomocí nichž lze vkládat data do databáze.

3.2.1 Grafické rozhraní

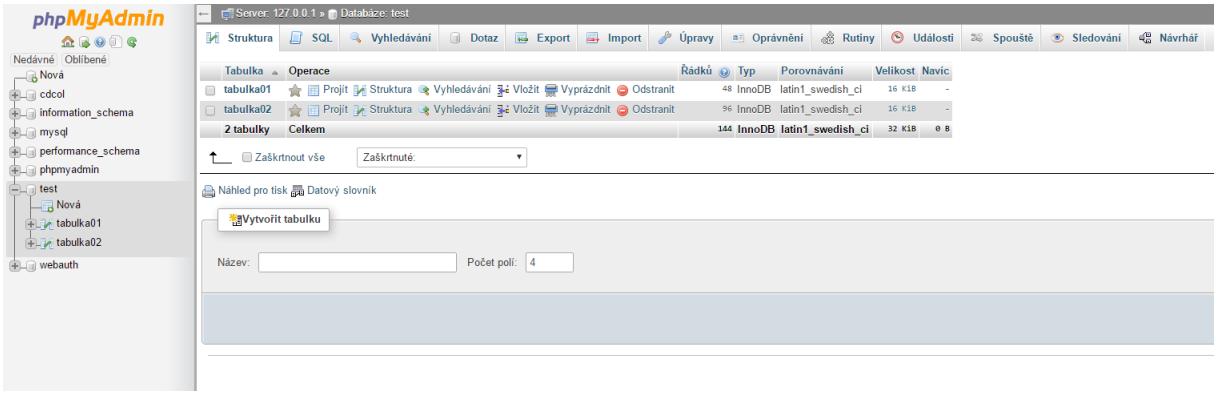
První a nejsnazší způsob vkládání dat do databáze je za pomoci grafického rozhraní. Ovšem ne každý databázový systém má tuto možnost integrovanou v základu. Kupříkladu u MySQL, která je základním testovacím systémem pro zde vyvíjenou aplikaci, není grafické rozhraní součástí instalace. Tento databázový systém ovšem nabízí různé druhy uživatelských rozhraní jako je například *PhpMyAdmin*[3]. Tento nástroj, jak již název napovídá, je naprogramován v jazyce PHP a nabízí velice rychlou a snadnou interakci s databází. Pomocí něj lze manuálně velice snadno spravovat data v databázi. Tvorba testovacích dat je tímto způsobem realizovatelná, avšak při větším množství dat velice neefektivní a pomalá.

3.2.2 Jednořádkové vkládání

Druhý způsob vkládání dat, již lze využít pro automatické generování testovacích dat, se nazývá jednořádkové vkládání a používá se k tomu jazyk SQL a některé jeho konstrukce. Tvorba dat pomocí konstrukcí jazyka SQL je mnohem rychlejší a efektivnější než za použití uživatelského rozhraní. Za pomoci této metody lze postavit funkční generátor testovacích dat. Příklad použití jazyka SQL pro vložení dat je demonstrován na úseku kódu níže 3.1

```
INSERT INTO table_name (col01, col02) VALUES (val01, val02)
INSERT INTO table_name (col01, col02) VALUES (val03, val04)
```

[3] PhpMyAdmin - nástroj napsaný v jazyce PHP umožňující jednoduchou správu databáze MySQL pomocí webového grafického rozhraní



Obrázek 3.3: PhpMyAdmin - Grafické rozhraní pro systém MySQL

Listing 3.1: Ukázka kódu jednořádkového vkládání

Využití tohoto způsobu vkládání testovacích dat je mnohem efektivnější a rychlejší než při manuálním vkládání. Pokud se kód 3.1 replikuje, vytvoří se tak několik záznamů do databáze. Ovšem s rostoucím počtem vkládaných dat rapidně klesá rychlost generování. Ta je sama o sobě závislá na počtu sloupečků v tabulce a na množství záznamů, jež se generují.

3.2.3 Víceřádkové vkládání

Dalším způsobem automatizovaného generování dat do databáze je tzv. víceřádkové vkládání tzv. *Bulk insert*, které je zobrazeno v kódu níže 3.2. Tento způsob vychází z klasického řádkového vkládání, ale vylepšuje ho o jednu zásadní věc. Tou věcí je, že provedení příkazu **INSERT** se vykoná pouze jednou oproti pomalejší metodě, kde se tento příkaz vykoná tolikrát, kolik se vkládá záznamů [4].

```
INSERT INTO table_name (col01, col02) VALUES (val01, val02),
(val03, val04), (val05, val06)
```

Listing 3.2: Ukázka kódu víceřádkového vkládání

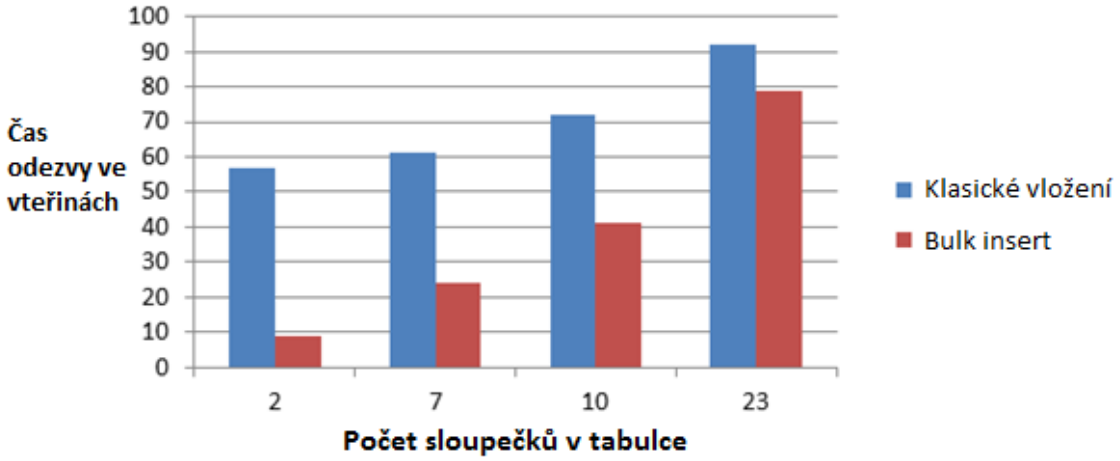
Díky použití této metody lze dosáhnout výkonu a efektivnosti vhodné profesionálních generátorů testovacích dat. Tento způsob tvorby dat je několikanásobně rychlejší než jednořádkové vkládání. Bohužel zde ale hraje velice velkou roli počet sloupečků v tabulce. S přibývajícím počtem atributů tabulky se bude přímo úměrně zpomalovat rychlost víceřádkového vkládání až bude téměř stejně rychlá jako jednořádkové viz graf 3.4.

Automatizované generování dat souvisí velice často i s některým dalším jazykem, který tyto metody zapouzdřuje a implementuje. SQL často poskytuje pouze struktury, ve kterých je možné data vkládat do databáze. Ovšem databázové systémy, jako je například Microsoft SQL Server, mají několik vlastních SQL příkazů. Tyto příkazy nabízí například, dříve zmíněné, víceřádkové vkládání (Bulk insert) [10]. Tato metoda, zabudovaná v Microsoft SQL

Serveru, pracuje poněkud jiným způsobem než klasické víceřádkové vkládání. Dokáže načítat nejenom surová data, ale také celé soubory v daném formátu a sama je dokáže rozdělit na data a vložit je [4].

3.2.4 Porovnání metod pro vkládání záznamů

Jak již bylo zmíněno v předešlém odstavci, rychlost vytváření záznamů já závislá jak na zvoleném způsobu tvorby dat, tak na počtu sloupečků v tabulce, které je nutno naplnit. Dalším neméně důležitým aspektem pro rychlost vkládání je samozřejmě výpočetní výkon daného zařízení, což ve zde popsaných testech nebude hrát žádnou roli. Na grafu 3.4 lze vidět, že při generování jednoho milionu záznamů do tabulky o 7 sloupcích je časová odezva více než dvakrát rychlejší při víceřádkovém vkládání, než při jednořádkovém. Pokud se počet sloupečků, jež je nutné naplnit, zvyšuje, roste tak přímo úměrně i časová odezva [4].

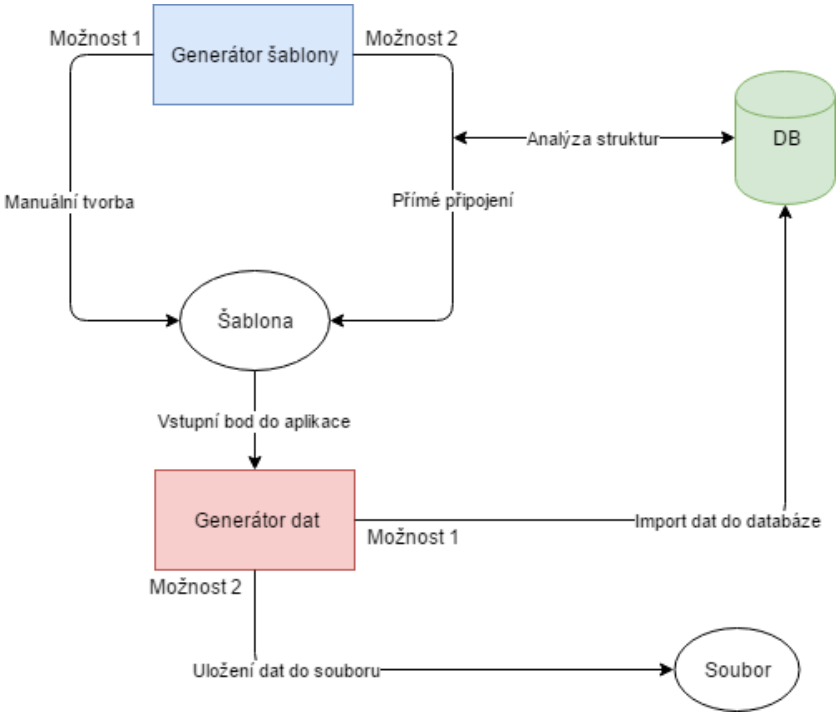


Obrázek 3.4: Rostoucí doba odezvy při zvyšování počtu sloupečků (1 milion záznamů)

Kapitola 4

Návrh generátoru

Před vývojem každé aplikace je velice důležitý mít přesně specifikovaný návrh nebo alespoň určitý směr, kterým bude vývoj aplikace směřovat. V následující kapitole bude teda popsán návrh aplikace, která by měla generovat testovací data, dle předem vytvořené šablony, do databáze. Celý nástroj se skládá ze dvou částí (aplikací). První částí je webová aplikace nebo respektive nástroj, kterým se generuje šablona, pomocí níž se budou následně generovat testovací data. Druhou částí je samotná aplikace do níž se nahraje předem vygenerovaná šablona a díky ní se vytvoří a vloží testovací data do databáze. Celý nástroj by měl být multiplatformní, proto se klade velký důraz na zvolený jazyk i na způsob používání. Mimo jiné by hlavní aplikace, běžící na zvolené stanici, měla podporovat uložení vygenerovaných dat v případě nedostupnosti databáze nebo nezadání přihlašovacích údajů.



Obrázek 4.1: Diagram aplikace

4.1 Generátor šablony

Jedna z hlavních částí nástroje je generátor šablony. Ten by měl sloužit, jak už název napovídá, pro vygenerování šablony, dle předem daných specifikací. Tento generátor je umístěn externě (online), mimo hlavní aplikaci. Nástroj pro tvorbu šablony je také možno nainstalovat na danou stanici, kde se bude nacházet také cílová databáze. Pro bezproblémový běh generátoru šablony je třeba splnit určité podmínky, které budou zobrazeny při instalaci.

4.1.1 Rozhraní

Tvorba šablony by měla fungovat oddělení od hlavní aplikace a jako celý nástroj by neměla být nijak závislá na zvolené platformě. Ideální způsob umístění aplikace by byl online na internet. Vyvstávají ovšem zde ale dva problémy. Jedním je možná absence internetového připojení, která by uživateli znemožnila tvorbu šablony. Dalším problém by mohla být nedůvěryhodnost aplikace, kde by uživatel musel na internet umisťovat přihlašovací údaje do jeho databáze a musel by mít otevřené porty pro externí připojení. Po zvážení veškerých možností by bylo nejideálnější umístit webovou aplikaci na lokální počítač, kde by ovšem musel být nainstalován také lokální server.

4.1.2 Šablona

Vygenerovaná šablona bude pravděpodobně ve formátu XML[1]. Tento formát byl zvolen především díky jeho jednoduchosti a podpory využívat jej v mnoha programovacích jazycích. Díky dobré čitelnosti jazyka XML dokáže i obyčejný uživatel upravovat vygenerovanou šablonu pomocí obyčejného textového editoru [18].

Obsahem šablony budou především informace o tabulce a jejich jednotlivých atributech. Dále nepovinným obsahem mohou být základní informace o databázi, pomocí nichž se lze připojit, viz 4.1. Informace o připojení k databázi není nutné z bezpečnostního hlediska zadávat, jelikož by to mohlo působit velice nedůvěryhodně. Z tohoto důvodu lze přihlašovací údaje vyplnit přímo v aplikaci nebo vygenerovaná data exportovat do souboru.

```
<?xml version="1.0" encoding="utf-8"?>
<database>

<connection>
    <login>admin</login>
    <password>password</login>
    <host>localhost</login>
    <dbname>databaze</dbname>
</connection>

<table name="tabulka01" rows="20">
    <col name="cl1" type="t1" cat="c1" cat02="c2" cat03="c3"
    foreign="null" unique="0"/>
    <col name="cl2" type="t2" cat="c1" cat02="c2" cat03="c3"
```

[1]XML - obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C

```
    foreign="null" unique="1"/>
</table>

<table name="tabulka02" rows="10">
    <col name="cl1" type="t1" cat="c1" cat02="c2" cat03="c3"
    foreign="null" unique="0"/>
    <col name="cl2" type="t2" cat="c1" cat02="c2" cat03="c3"
    foreign="tabulka01;cl2" unique="0"/>
</table>

</database>
```

Listing 4.1: Příklad šablony v jazyce XML

4.1.3 Tvorba šablony

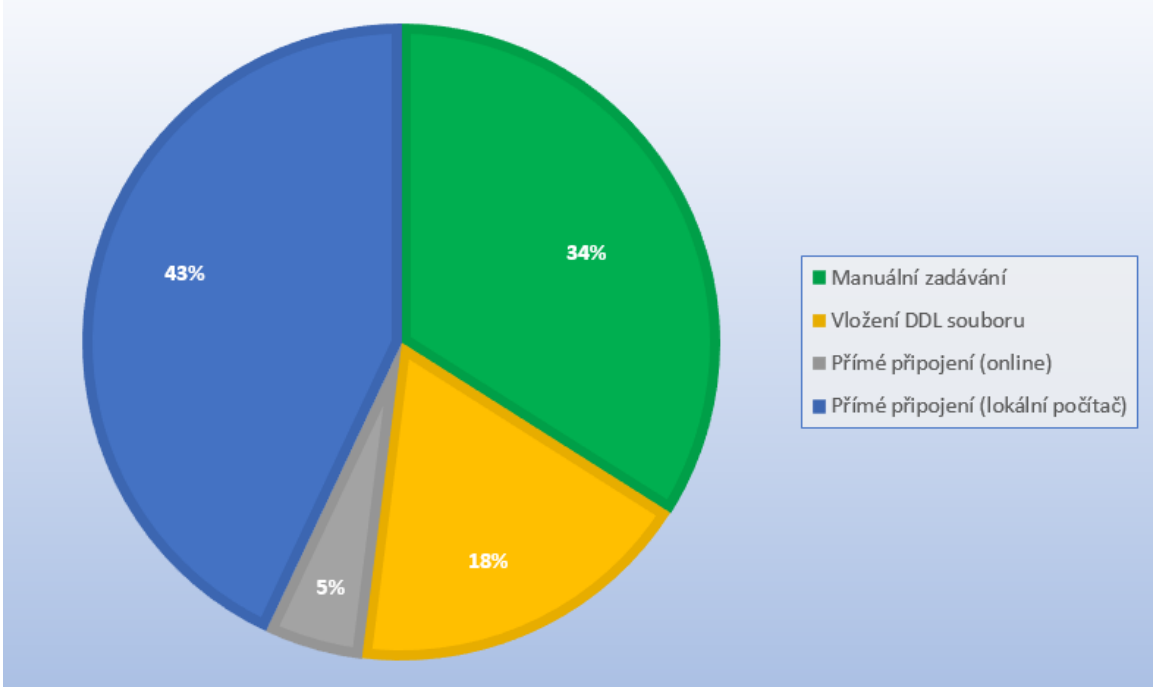
Vytvoření šablony, pomocí níž by se generovaly testovací data, lze rozdělit na manuální tvorbu a automatizovanou. Pomocí manuálního vytvoření šablony se předpokládá že uživatel ručně vyplní veškeré povinné hodnoty ve formuláři. Těchto hodnot může být při větších databázích relativně velké množství. Vyplňování formulářových dat v takovémto měřítku může přijít velice nepříjemné a uživatelsky nepřívětivé.

Z tohoto důvodu bude možno zvolit automatizovanou tvorbu šablony. Při návrhu aplikace byly provedeny průzkumy mezi administrátory, žáky a programátory, jež by tento nástroj využívali. Z výsledků průzkumu, jež lze vidět na grafu 4.2, je vidět velká popularita především v automatizované možnosti předvyplněné formuláře pro tvorbu šablony. Samozřejmě tato forma je uživatelsky nejvíce přívětivá, ale ne všechny její možnosti jsou ideální a bezpečné. Mimo samotnou manuální tvorbu šablony, jež byla do průzkumu zahrnuta, jsou zde ještě další tři způsoby, které patří do kategorie automatizované tvorby. Využití DDL[2] souboru by bylo za jiných okolností ideální, avšak zpracování tohoto typu souboru by vyžadovalo velice komplexní a rozsáhlý parser [3]. Další dvě možnosti jsou přímé připojení do databáze ať už pomocí online způsobu nebo za využití instalátoru na cílovou stanici. Z výsledků průzkumu vyplynulo, že instalace webové aplikace je mnohem výhodnější ať už z hlediska bezpečnosti nebo dalšího využití aplikace. Naopak online způsob této možnosti, kde by docházelo k připojení do cizí databáze pomocí otevřeného portu, za využití přihlašovacích údajů, bylo vyhodnoceno jako vysoce riskantní a nepoužitelné [5].

Poslední možností, která ne tak úplně zapadá do samotné tvorby šablony, je načtení již hotové a vygenerované šablony a její následná editace. Načtením hotové šablony započne vlastně tvorba nové, s tím rozdílem, že již bude mít předvyplněné veškeré hodnoty, které v sobě uchovává hotová šablona, lze tyto hodnoty změnit a tím vygenerovat novou šablonu.

[2] DDL - https://en.wikipedia.org/wiki/Data_definition_language

[3] Parser - může se jednat o program nebo script, který provede kontrolu daného souboru (například ve formátu XML) a zpracuje ho podle daných požadavků, například rozdělení dat a jejich výpis.



Obrázek 4.2: Průzkumy ukazující odezvu uživatelů na volbu tvorby dat

4.2 Generátor dat

Samotný nástroj, jež umožní generování dat do zvolené databáze pomocí předem vytvořené šablony. Tento nástroj bude využívat seznamy jmen, příjmení, ulic, měst, aj. z nichž bude možno vytvořit celé záznamy. Výsledná data bude možno uložit přímo do databáze nebo je exportovat do souboru. Nástroj bude spuštěn na klientském počítači, kde bude umístěna také daná databáze. Typ databázového systému, pro nějž je nástroj stavěn jsou relační databáze využívající jazyk SQL, konkrétně MySQL. Mimo samotného generování aplikace dokáže exportovat data spolu s příkazy jazyka SQL do souboru, pokud uživatel nezadal nebo nechce zadávat přihlašovací údaje do jeho databáze.

4.2.1 Rozhraní

Návrh rozhraní generátoru vychází opět z předpokladu multiplatformního využití aplikace. Generátor samotný se bude vytvářet v jazyku, který bude poskytovat přenositelnost napříč operačními systémy. Základní testovací systémy budou Windows a Linux. Nástroj, jež bude spuštěn pod platformou Windows vykreslí jednoduché grafické uživatelské rozhraní, které nabídne ovládání aplikace pomocí myši. Naopak pod Linuxem, kde se klade větší důraz na ovládání pomocí terminálu, bude moci uživatel spustit aplikace pouze s využitím konzole.

4.3 Zvolené technologie

Technologie, jež mohou být použity pro vývoj nástroje popisovaného v této práci. Jedná se o samostatné programovací jazyky včetně různých knihoven a frameworků, jež mohou tyto jazyky využívat.

4.3.1 HTML a CSS

HTML je značkovací jazyk. Je to zkratka ze slov Hypertext markup Language (hypertextový značkovací jazyk). Je to hlavní a základní jazyk pro vytváření webových stránek. Umožňuje publikaci dokumentů na internetu. Využívá tzv. tagy pomocí nichž sestavuje různé struktury na webových stránkách jako jsou tabulky, různé rámce, formátování textu aj. Samotný jazyk HTML nenabízí nikterak rozsáhlé grafické možnosti. Proto je pro grafické prvky a celkový design využito CSS (Cascading Style Sheets). Obecně řečeno se jedná o určitý zápis, který definuje vzhled webové stránky, ať už se jedná o barvy textu, pozadí, velikosti písma nebo různé rozložení prvků na stránce [11].

4.3.2 PHP

Je skriptovací programovací jazyk využívající se především pro dynamické webové stránky. PHP je v dnešní době jedním z nejvíce rozšířených a nejoblíbenějších programovacích jazyků využívajících se pro tvorbu dynamické části webové stránky. PHP se používá na straně serveru, to znamená, že každá interakci s uživatelem odešle určitý požadavek na server, ten se tam zpracuje a odešle zpět klientovi do prohlížeče. Jednou z výhody PHP je, že je možno využívat objektově orientované programování, díky němuž je vývoj a následná úprava projektů mnohem snadnější. Umožňuje pomocí různých způsobů spolupracovat databázemi a konkrétně velice širokou podporu pro databáze MySQL [3].

4.3.3 JavaScript

JavaScript je skriptovací programovací jazyk, který se využívá na webových stránkách. Zapisuje se buď přímo do kódu HTML nebo může být zapsán do vlastních souborů, které jsou následně externě vloženy. JavaScript je klientský programovací jazyk, takže se vykonává na straně klienta přímo v prohlížeči. Nevýhoda klientského vykonávání scriptu je ta, že se dá relativně snadno editovat, případně úplně vypnout a může se tak stát, že některé funkce, které jsou na tomto jazyku závislé, nebudou korektně fungovat. Tento jazyk je využit především pro dynamické prvky na stránce, kterými může například být kontrola formulářových dat v reálném čase, ještě před odeslání, různé informační nebo chybové hlášení, které se mohou zobrazovat opět v reálném čase na obrazovce aj. [2].

4.3.4 jQuery

Knihovna napsané v jazyce JavaScript. Jednoduchá knihovna, pomocí které se dají napsat dynamické kontroly formulářů, dynamické zprávy. Nabízí mnoho vizuálních efektů pro zkrášlení vzhledu a zjednodušení interakci uživatele se webovou stránkou [7].

4.3.5 Java

Java je jazyk tzv. 3. generace. Jedná se o imperativní jazyk vysoké úrovně. Je univerzální a tudíž není specifikován pro konkrétní oblast. Je možné jej využít pro vývoj webových aplikací nebo také pro desktopové aplikace. Je to multiplatformní jazyk. Díky tomu, že je objektově orientovaný, tak nese podobné výhody, jaké byly zmíněny u PHP. Projekty programované v tomto jazyce budou v mnohem snadnější udržovatelné a případné rozšíření zde není příliš velký problém. Jazyk Java nabízí nespočet knihoven ať už pro grafické prostředí nebo pro práci s různými typy a formáty dokumentů nebo databází. Poradí si hravě s připojením k relační databázi MySQL, stejně tak dokáže velice snadno zpracovat soubory ve formátu XML [13].

4.3.6 Swing

Swing je framework pro programovací jazyk Java. Umožňuje velice jednoduchou tvorbu formulářových aplikací. Lze zde nalézt několik již hotových komponent, pomocí kterých lze vytvořit aplikaci se základním grafickým rozhraním. Tato knihovna je postavena na starším frameworku *AWT*. Existují zde dva způsoby návrhu formuláře. Prvním je tzv. *grafický návrhář*, což je rozhraní, které umožňuje "naklikat"a ručně nastavit pozice všem prvků jednoduše pomocí myši. Druhým způsobem je manuální návrh, kde je třeba veškeré prvky naprogramovat ve zdrojovém souboru [9].

4.3.7 XML

Zkratka XML (eXtensive Markup Language) znamená, že se jedná o rozšiřitelný značkovací jazyk. V jednoduchosti je to jazyk velice podobný například HTML, který byl popsán výše. Využívá značky, tzv. tagy a atributy. S tím rozdílem, že zde je to plně v roli uživatele nebo programátora, jaké názvy atributů a tagů použije. Soubor XML může například obsahovat rozsáhlou databázi nebo informace o pracovnicích nebo záznamy z knihovny, které díky robustnosti tohoto jazyka mohou být velice snadno strukturované. Dalším stěžejním bodem je, že dokument ve formátu XML dokáže velice snadno upravit i obyčejný uživatel, kterému stačí základní znalosti o tom jak má daný dokument vypadat. Díky těmto výhodám by se tento jazyk mohl hodit pro definici šablony, pomocí níž by se generovaly testovací data do databáze. V neposlední řadě je podpora pro dokumenty ve formátu XML implementována do téměř všech nejpoužívanějších programovacích jazyků do kterých patří například Java nebo PHP [12].

Kapitola 5

Implementace generátoru

Samotná implementace nástroje je důležitá k jeho zhotovení a dovedení do finální podoby a vychází především z návrhu nástroje. Nachází se zde konkrétní postup implementace a použitých způsobů při vývoji nástroje popisovaného v této práci. Jak již bylo uvedené v návrhu, celý nástroj je složen ze dvou částí. Každá část je vyvíjena paralelně a v jiném jazyce. Jako rozhraní mezi těmito dvěma aplikacemi slouží univerzální šablona ve formátu XML, která je výstupem z jedné aplikace a vstupem do druhé.

5.1 Generátor šablony

Aplikace, jež slouží pro generování šablony, je vyvíjena jako webový nástroj v jazyce *PHP*. Díky výběru tohoto jazyka je aplikace multiplatformní a lze ji využít jakožto online nástroj na internetu nebo si ji lze umístit na lokální počítač a využívat ji tam. Pro vývoj této části nástroje byl využit způsob objektově orientovaného programován[1] (OOP), který jazyk PHP nabízí. Díky tomu způsobu, je aplikace modulární a lze ji do budoucna velice jednoduše spravovat nebo rozšiřovat o další funkce.

5.1.1 Adresářová struktura a soubory

Jak již bylo uvedeno, aplikace je modulární a využívá objektově orientováný způsob programování. Je rozdělená do základních adresářů pomocí logické struktury.

```
/
├── classes
│   ├── Engine.php
│   └── Template.php
├── CSS
│   └── layout.css
├── js
│   └── jquery.js
├── images
├── pages
└── templates
```

[1] OOP - https://www.interval.cz/stitek/oop-php/

Adresář `classes` v sobě ukrývá implementaci tříd `Engine` a `Template`. První třída obstarává běh celé webové stránky včetně načítání jednotlivých podstránek a základní funkcionality. Druhá třída je více specifičtější pro daný nástroj a ukrývá v sobě implementaci generátoru samotného.

V adresáři `CSS` lze nalézt soubor `layout.css` ve kterém je nadefinován vzhled webové stránky pomocí jazyka CSS.

Veškerý JavaScript, včetně knihovny jQuery[2] jež je použit na stránce se nachází v adresáři `js`, konkrétněji v souboru `jquery.js`. Pomocí něj lze využít různé dynamické metody například pro vykreslování komponent do formuláře, aj.

Nakonec adresář `images` ve kterém se, jak už název napovídá, nachází obrázky použité na webovém nástroji. Veškeré stránky a podstránky jsou uloženy v adresáři `pages`. A na závěr, jelikož generátor šablony má jako výstup požadovanou šablonu, tak adresář `templates` v sobě uchovává veškeré vygenerované šablony.

5.1.2 Možnosti generování

Základem pro generování šablony je výběr metody, pomocí které se bude šablona vytvářet. Jak bylo uvedeno v návrhu generování šablony, je tu více možností pro tvorbu šablon. První volbou je manuální vytváření. Tento způsob tvorby šablony vychází z klasického ručního vyplnění formuláře, který lze vidět na obrázku 5.3. Formulář se skládá například z údaje o počtu tabulek v databázi jež bude chtít uživatel naplnit, ze jména tabulky, jednotlivých sloupců v tabulce a jejich typů. Kromě popisu struktury tabulek a sloupečků, lze vyplnit také informace o připojení k databázi. Vyplnění přihlašovacích údajů pro manuální tvorbu je volitelné, jelikož pro některé uživatele nemusí být příjemné psát přístupové údaje k databázi na různé webové stránky (jedná se především o online verzi).

Jednotlivé kroky, jako je zadání počtu tabulek a přihlašovacích údajů, výběr a vyplnění informací o tabulkách, jsou implementovány jako samostatné a znovupoužitelné metody. Díky tomuto rozdělení, lze jednoduše spravovat jednotlivé kroky a nezasahovat tak do celkové funkcionality nástroje. Po validním zadání veškerých požadovaných údajů o databázi se vygeneruje samotná šablona ve formátu XML.

Nástroj také nabízí automatické generování šablony za pomoci přímého připojení k databázi. Tato funkce je využitelní především v případě, kde je nástroj nainstalován na klientském počítači. Možnost s využitím DDL souboru, jež byla v návrhu popisována byla zrušena kvůli velice složité implementaci.

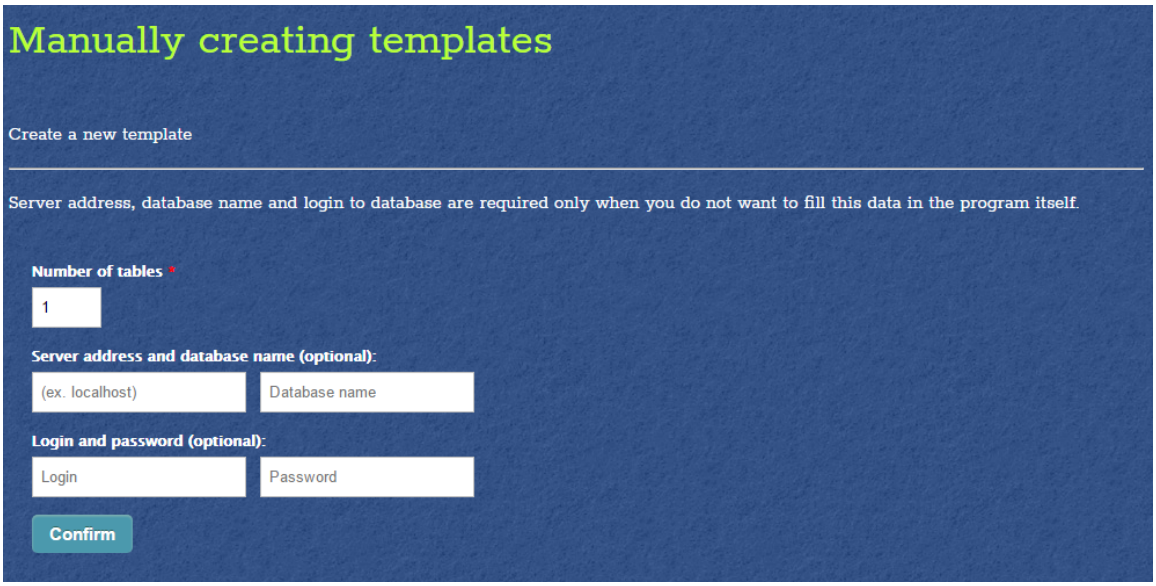
Každá tvorba šablony je rozdělena na jednotlivé kroky, něco jako nákupní košík v elektronickém obchodě. Počet kroků se liší dle zvoleného typu tvorby šablony.

5.1.3 Manuální generování

Základem tohoto způsobu je zvolení počtu tabulek a vyplnění údajů k připojení (volitelné) víz 5.1. Pomocí metody `ParseBasicValues()` se provede první krok a tím se inicializuje

[2] jQuery - http://api.jquery.com/

počáteční *SESSION*[3], do které se následně ukládá celá struktura tabulek a informací o databázi. V následujícím kroce se vyplňují názvy atributů, tabulek, jejich typy, kategorie dat a další informace o datech, jež se budou do daných sloupečků generovat. Po tomto výběru se provede metoda `ConvertTables()`. Tato metoda jako vstupní argument přijímá veškerá uživatelem vyplněná data z formuláře, kde byla definována struktura tabulky a generovaných dat. Tato data budou přerozdělena a naformátována do obecné struktury, aby byla připravena pro export. Hned po provedené této metody, je zavolána metoda `MakeXML()`, která z již hotových a strukturovaných dat vygeneruje XML soubor se šablonou. K vytvoření XML souboru slouží knihovna *SimpleXML* [16]. A pro úpravu formátování ve výsledném souboru je využita *DOMDocument* [15]. Soubor je uložen do adresáře `templates`. V posledním kroku, je šablona již hotová a je nabídnuta uživateli ke stažení. K tomu poslouží metoda `DownloadFile()`.



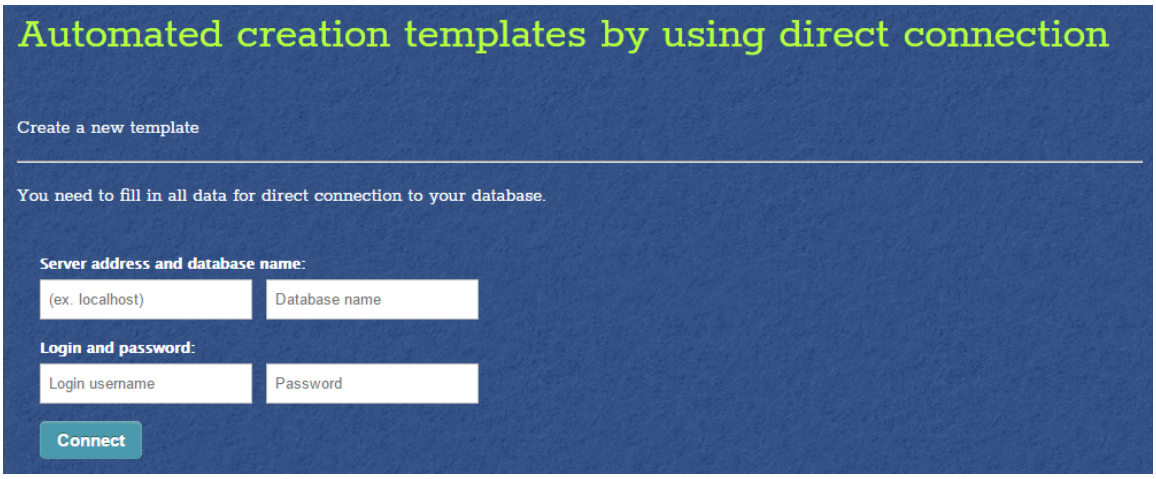
Obrázek 5.1: Rozhraní - Manuální tvorba šablony

5.1.4 Automatizované generování

Velká část automatického generování šablon je podobná jako u manuálního. Při tomto způsobu tvorby šablony je využito přímé připojení k databázi. Je nutné zde vyplnit veškeré povinné přihlašovací údaje včetně názvu databáze, viz 5.2. Pro využití této možnosti se doporučuje, aby se nástroj pro generování šablony nacházel na počítači, kde je provozována také databáze a je možné se do ní připojit. Další možností by bylo se připojit z online webové aplikace pomocí otevřeného portu na určitou databázi. Připojení do MySQL databáze je implementováno v metodě `ConnectToDatabase()`, která využívá funkce knihovny *MySQLi*[4]. Připojení probíhá pomocí zadání hostitelské adresy, vyplnění přihlašovacího jména, hesla a zvolení databáze. Po úspěšném připojení do databáze se

[3]SESSION - http://www.w3schools.com/php/php_sessions.asp
[4]MySQLi - http://php.net/manual/en/book.mysqli.php

pomocí SQL příkazu `SHOW TABLES` zobrazí veškeré tabulky databáze. Ty jsou vykresleny v cyklu spolu se zaškrtávacími políčky. Uživatel zvolí, které tabulky mají být zahrnuty v šabloně a potvrdí aktuální krok. Spolu se samotnými tabulkami, jsou zde zobrazeny i cizí klíče a jejich vztahy napříč hierarchií databáze. Následně se provede metoda `ProcAutotables()` do které se pošle seznam zvolených tabulek. Zde se pomocí příkazu `SHOW COLUMNS` postupně analyzují všechny zvolené sloupečky daných tabulek a informace o nich, jako je název a typ. Všechny tyto informace se uloží do struktury. Pomocí výsledné struktury se předvyplní velká část formuláře o struktuře tabulek. Je nutné ještě vyplnit kategorie dat, jež mají být generovány. V následujícím kroku je opět využita metoda `ConvertTables()` pro převedený struktury do jednotného formátu a metodou `MakeXML()` je vytvořena samotná šablona. V posledním kroku si již uživatel může šablonu stáhnout opět pomocí metody `DownloadFile()`.



Obrázek 5.2: Rozhraní - Automatizovaný tvorba šablony

```
foreach ($tables as $table) {
    $t=$this->t->FindRawTable($table, $relations);
     if (is_array($t)) {
        foreach ($t as $new_t) {
            if (!in_array($new_t, $newTables)) $newTables[]=$new_t;
        }
    }else{
        if (!in_array($t, $newTables)) $newTables[]=$t;
    }
}
```

Listing 5.1: Hlavní cyklus pro procházení zvolených tabulek

```
public function FindRawTable($table, $relations) {
   // If selected tables is not in relations
   if (!isset($relations[$table])) {
      return $table;
   }else{
      $ret_tabs=array();
      foreach ($relations[$table] as $t) {
         if (isset($relations[$t[0]])) {
            $ret_t=$this->FindRawTable($t[0], $relations);
            if (is_array($ret_t)) {
               $ret_tabs=array_merge($ret_tabs, $ret_t);
            }else{
               $ret_tabs[]=$ret_t;
            }
         }else{
            $ret_tabs[]=$t[0];
         }

         $end_t=end($relations[$table]);
         if ($t[0]==$end_t[0]) {
            $ret_tabs[]=$table;
         }
      }
      return $ret_tabs;
   }
}
```

Listing 5.2: Rekurzivní metoda pro seřazení tabulek podle závislosti na vyplnění

5.1.5 Integritní omezení

Pod pojmem integritní omezení si lze představit jistá pravidla, která musí data v databázi splňovat pro to, aby databáze byla konzistentní a validní. Obecně se této problematice říká *databázová integrita*[5]. Integritní omezení je zavedeno především pro automatizovanou tvorbu šablony, kdy lze analyzovat celou databázi a určit všechny vztahy mezi všemi tabulkami. V případě manuálního generování, lze pouze omezit data na unikátní hodnoty. Při generování šablony je to tak další možnost jak vytvořit co nejspecifičtější testovací data a především jak zachovat konzistenci databáze a všech tabulek. Tabulky v databázi mohou mít mezi sebou určité vztahy, jak již bylo řečeno, tyto vztahy jsou vytvořeny a vázány pomocí cizích klíčů (viz výše). Vztah mezi primárním klíčem jedné tabulky a cizím klíčem v druhé tabulce, je zobrazen při výběru tabulek v případě automatizovaného generování šablony. K tomu je využita metoda `GetAllRelations()`, která pomocí SQL dotazu vybere všechny vztahy z celé databáze včetně informací o nich (tabulka, cizí klíč, primární klíč). Při výběru tabulek pro generování dat je nutné zvolit všechny tabulky, jež spadají do daného vztahu.

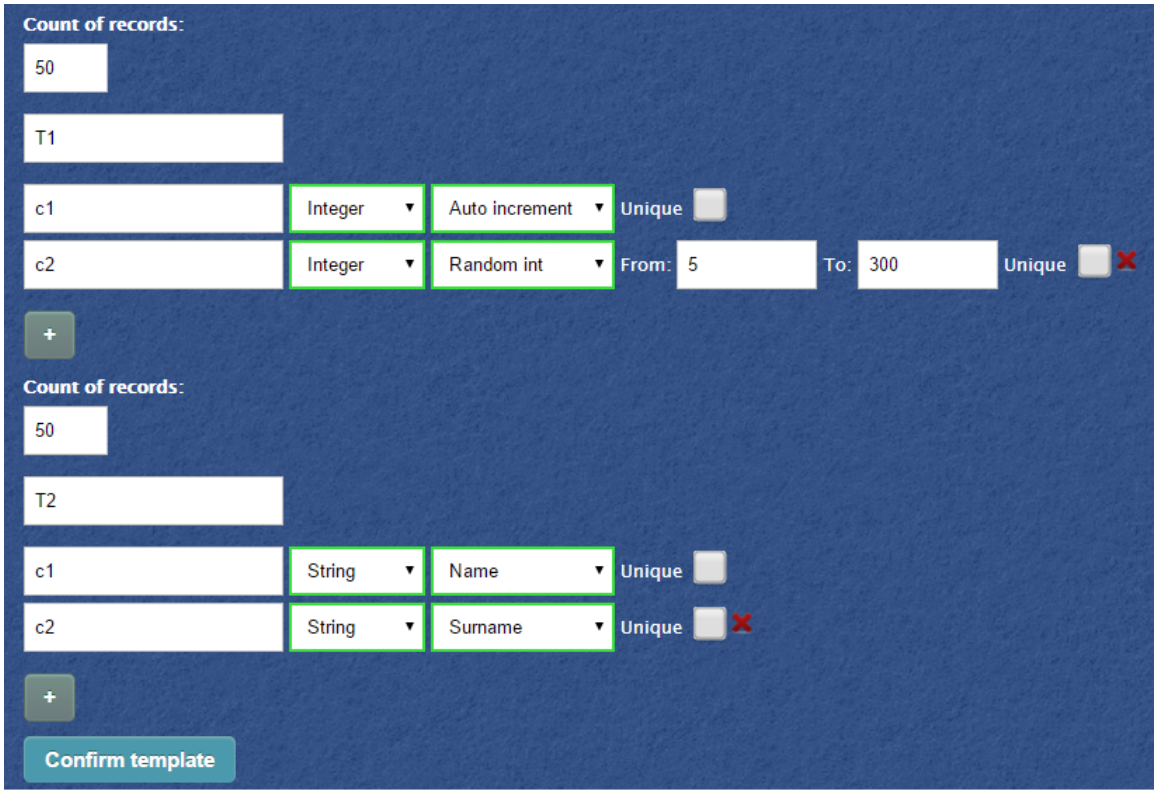
Vyhledání veškerých vztahů a tabulek, které je nutno naplnit daty, pro zachování konzistence databáze, je pouze malou částí problematiky integritních omezení. Podstatnou částí je seřazení a přefiltrování těch tabulek, jež mají nějaké vztahy s jinými tabulkami a jejich data jsou tak nepostradatelné pro generování dalších dat. Filtrace a řazení tabulek je prováděna rekurzivní metodou `FindRawTable()`, ukázka kódu: 5.2. Tato metoda se používá cyklicky na všechny zvolené tabulky, ukázka kódu: 5.1. Přijímá dva argumenty, kde prvním je aktuální tabulka a druhým je seznam vztahů databáze. Rekurzivním procházením jsou tabulky roztříděny tak, aby tabulky, na jejíchž existenci dat jsou závislé jiné, byly vyplněny prioritně před nimi. Seřazení je nejdůležitější a víceméně jedinou důležitou věcí o zbytek se postará metoda `ProcAutoTables()` (viz výše).

Další částí integritních omezení je specifikace unikátních hodnot do konkrétních sloupečků. Tato specifikace je možná jak pro automatizovanou tvorbu tak i pro manuální. Jedná se o zaškrtávací políčko *UNIQUE*. Jeho funkcí je, že generované data, budou unikátní, co se týče daného sloupce pro který je tato funkce aktivována.

5.1.6 Rozhraní

Rozhraní generátoru šablony je postaveno vzhledově velice podobně jako různé internetové generátory testovacích dat. Základem je formulář, který se dynamicky duplikuje podle počtu tabulek. Pro každou tabulku existuje vstupní pole pro zadání kolik záznamů se má generovat. Následně lze vytvořit jednotlivá pole, která reprezentují sloupečky tabulky. Lze dynamicky přidávat a odebírat další sloupce pomocí knihovny jQuery. Pro každý sloupec je nutno zvolit jeho datový typ a následně podle volby datového typu je nutno zvolit kategorii nebo respektive druh dat, které se mají do daného sloupce generovat.

[5] Databázová integrita - https://www.tutorialcup.com/dbms/integrity.htm



Obrázek 5.3: Ukázka rozhraní generátoru šablon

5.1.7 Podporované datové typy a kategorie

- **Integer** - Datový typ pro celá čísla. Nabízí generování celočíselných hodnot a to buď náhodná čísla definovaná rozsahem nebo automatické zvyšování hodnoty tzv. auto increment.
- **Float** - Datový typ pro desetinná čísla. Nabízí generování desetinných hodnot a to náhodná čísla definovaná rozsahem.
- **String** - Datový typ pro řetězce. Nabízí generování řetězců za využití databázových hodnot. Je možné zvolit například jméno, příjmení, město, ulici nebo náhodné slovo.
- **Text** - Datový typ, též pro řetězce, ale mnohem větší délky. Jedná se o vygenerování například celých odstavců. Nabízí generování odstavců pomocí *Lorem ipsum*[6].
- **Date** - Datový typ pro datum. Nabízí generování náhodného data dle zvoleného rozmezí.
- **Time** - Datový typ pro čas. Nabízí generování náhodného času dle zvoleného rozmezí.
- **Boolean** - Datový typ boolean. Nabízí pevné generování hodnoty true/false nebo náhodnou volbu mezi těmito možnostmi.

[6]Lorem ipsum - http://www.lorem-ipsum.cz/

5.1.8 Tvorba instalátoru

Instalátor pro webovou aplikace, kterou je generátor šablony, byl vytvořen pomocí externí aplikace *BitRock Install Builder*[7]. Instalátor byl vydán pro platformy Windows, Linux a Mac. Pomocí něj lze nainstalovat generátor šablony na klientský počítač a využívat jej bez nutnosti připojení k internetu. Instalátor zahrnuje všechny zdrojové soubory samotné webové aplikace, díky instalaci na lokální počítač by neměl být nejmenší problém s využitím automatizovaného generování šablony s využitím přímého připojení do databáze. Pro bezproblémový běh aplikace je nutné dodržet umístění aplikace při instalaci, jelikož aplikace využívá lokálního serveru pro připojení do MySQL a Apache[8].

5.2 Generátor dat

Aplikace, sloužící pro generování samotných dat a následné vkládání do databáze. Jako vstup zde slouží předem vygenerovaná šablona ve formátu XML. Tato část nástroje je vyvíjena v jazyce Java. Velký důraz je kladen především na multiplatformnost aplikace, na její údržbu a případné rozšiřování do budoucna. Díky těmto důvod byl vybrán jazyk, který nabízí objektově orientovaný způsob programování a multiplatformní využití. Aplikace je implementována jako modulární. Pro zobrazení grafického rozhraní je využita grafická knihovna swing.

5.2.1 Adresářová struktura a soubory

Aplikace je napsána dle objektově orientovaného paradigmatu a je logicky rozčleněna do několika souborů, kde každý z nich reprezentuje jednu třídu (objekt).

```
/
├── db
└── TestGen
    ├── Database.java
    ├── GUI.java
    ├── Generator.java
    ├── Main.java
    └── MySQL.java
```

Adresář `TestGen` obsahuje veškeré soubory s implementací tříd. Hlavní třída `Main`, odkud se následně spouští samotný proces generování, ať už z grafického rozhraní nebo konzolového zadávání, spadá do souboru **Main.java**. Třída `Generator` ze souboru `Generator.java` tvoří hlavní rozhraní mezi ostatními objekty. Třídy **MySQL** a `Database` nacházející se v souborech `MySQL.java` a `Database.java` slouží jakožto pomocné třídy pro připojení do databáze a pro načtení samotných dat sloužící pro generování. Poslední soubor `GUI.java` v sobě ukrývá celé grafické rozhraní jež je ve výchozím stavu zapnuté a pomocí spouštěcího argumentu `-nogui`, lze nastavit vypnutí tohoto rozhraní.

[7] BitRock Installer - http://bitrock.com/
[8] Apache - https://httpd.apache.org/

5.2.2 Princip generátoru

Počáteční bod, odkud aplikace vychází je třída `GUI`, ve které je implementováno grafické rozhraní aplikace a počáteční inicializace dalších tříd. Také je zde ukryta implementace pro filtrování argumentů, s nimiž může být aplikace spuštěna. Jedná se především argument, který deaktivuje grafické rozhraní mezi další argumenty patří například samotná šablona, možnost exportu do souboru nebo volba režimu pro vkládání dat. Grafické rozhraní je ve výchozím stavu zapnuté. Deaktivace grafického rozhraní spolu s vložením šablony přes parametr je využito především pro spouštění z konzole, kde může být absence grafických prvků. V obou dvou případech spuštění, bude následovat vytvoření objektu, ze třídy `Main`. Argumenty při vytváření této třídy jsou soubor se šablonou, logická hodnota určující zdali je zapnuté grafické rozhraní nebo nikoliv a odkaz na soubor určený pro export, pokud takový existuje. Tento objekt je vytvořen v závislosti na zvoleném typu spuštění. V případě grafického rozhraní, je objekt `Main` vytvořen až spolu s výběrem šablony a zároveň je spuštěna metoda `LaunchGen()`. V druhém případě, kdy se jedná o konzolové spuštění aplikace, se tvorba objektu `Main` vykoná okamžitě po spuštění aplikace spolu se zavoláním metody `LaunchGen()`, která jako vstupní argument přijímá typ zvoleného režimu pro vkládání dat. Na výběr je mezi výchozím víceřádkovým vkládání a mezi jednořádkovým. Další dva argumenty, jež tato metoda přijímá jsou logické hodnoty, určující zdali se má aktivovat počítání časových hodnot pro generátor a zdali se mají vymazat předchozí obsah tabulek v databázi před vložením dat. Následuje vytvoření instance z třídy `Generator`, která jako dva vstupní argumenty přebírá odkaz na vstupní šablonu a odkaz na případný soubor pro exportování dat. Následuje kontrola validity samotné šablony, jež je zadána, pomocí metody `ProcessXML`.

Při analýze šablony se její jednotlivé informace, jako je připojení do databáze nebo konfigurace tabulek uloží do vnitřní struktury aplikace. Jelikož zadání přihlašovacích údajů k databázi do šablony je pouze volitelné, tak aplikace musí ověřit, zdali se tyto údaje v šabloně vyskytují nebo nikoliv. V případě, že nebyly uživatelem zadány, nebo byly zadány chybně, aplikace vyzve k zadání těchto údajů pomocí metody `CheckLogin()`. Tato metoda zároveň ověřuje existenci údajů v šabloně. Při existenci přihlašovacích údajích a jejich správnosti, nebo po úspěšném zadání těchto údajů, se pomocí metody `ConnectMySQL()`, pokusí aplikace připojit k databázi.

Při úspěšném připojení do databáze, se zavolá metoda `StartGenerate()`. Do této metody se odesílají pouze dva argumenty. Zvolený typ generování a logická hodnota určující vymazání předchozích dat ze zvolených tabulek. V této metodě se nachází vytvoření nové instance třídy `Database`. Tato třída obsahuje načtení celé lokální databáze obsahující surová data pro generování a jednotlivé metody jež těmto datům dávají finální podobu. Dále v metodě `StartGenerate()` jsou pomocí dvou resp. tří zanořených cyklů procházeny všechny tabulky a jejich jednotlivé sloupce. Podle typu daného sloupce je volána metoda `GenerateData()` která spadá pod třídu `Database`. Do této metody je zaslán typ kategorie a dodatečné informace k ní, jako je rozsah nebo různé datové omezení.

5.2.3 Integritní omezení

Nově vygenerované hodnoty, jež mají být unikátní v rozsahu daného sloupce, jsou porovnávány s hodnotami uloženými v lokální pomocné databázi, která je postupně naplňována nově vygenerovanými daty. Pokud je aktivní připojení k databázi a není nastaven export do souboru, jsou použity také již vyplněné hodnoty z databáze, v takovémto případě je garantováno zachování databázové integrity. Hodnoty jsou z databáze vybrány pomocí metody `QuerySelect()` nacházející se ve třídě `MySQL`, která přijímá dva argumenty, jméno tabulky a název sloupce. V případě, že se vygenerovala hodnota, která v tomto pomocném poli již existuje, provede se generování znovu a to do té doby, dokud nově vygenerovaná hodnota nebude v rámci pole a sloupce unikátní. Při nedostatečném množství hodnot nebo při špatně zvoleném výčtu hodnot bude generování ukončeno s chybovým hlášením. V případě, že je zvoleno vymazání stávajících dat z tabulek, provede se metoda `QueryDelete()` na zvolenou tabulku v případě aktivního připojení. V případě exportu do souboru bude před samotné generování umístěn příkaz pro vymazání stávajících dat, tím bude zaručeno zachování databázové integrity za cenu ztráty stávajících dat.

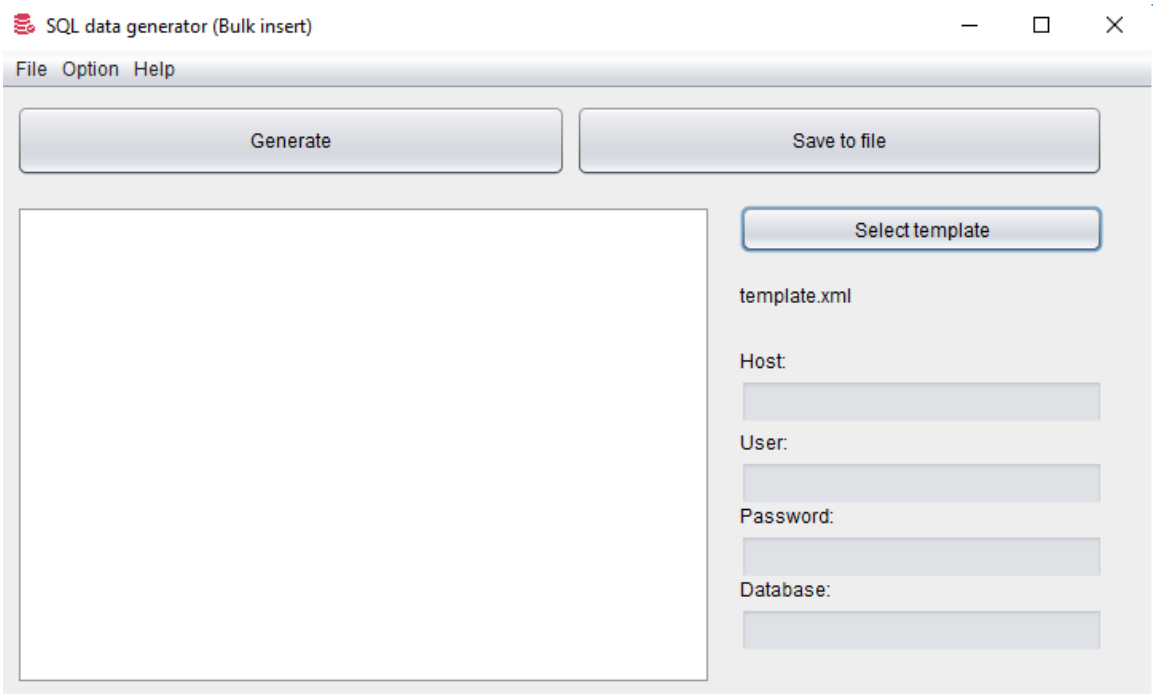
Jednotlivé datové typy mají vlastní způsoby pro generování unikátních hodnot. V případě generování například celočíselných hodnot je využíván cyklus, který v počáteční podmínce zjišťuje, zdali nově vygenerovaná hodnota existuje v poli již vygenerovaných dat. Životnost tohoto cyklus je tedy závislá na existenci vygenerované hodnoty ve výčtu porovnávaných hodnot a na počtu kombinací, které mohou být vygenerovány. Mimo počáteční podmínky je porovnáváno množství již vygenerovaných hodnot s maximálním počtem hodnot, které mohou být vygenerovány. V případě, že tyto dvě hodnoty budou identické, bude cyklus ukončen s chybou o nedostatečném množství požadovaných dat. Při generování unikátních dat z lokálních textových databází jako jsou výčty jmen, příjmení, měst, ulic, aj. je využit poněkud jiný a rychlejší způsob. Tento způsob je založen na poli, jež obsahuje výčet všech již použitých hodnot a ten je odečten od celkového výčtu surových dat, které mohou být vygenerovány.

Díky algoritmu, použitému ve webové aplikaci, při generování šablony, jsou tabulky, na jejichž hodnotách, jsou závislé některé jiné, vyplněné jako první. Tento algoritmus zajišťuje, že nebudou vyžadována data, která ještě nebyla vytvořena. Díky tomu je další implementace tvorby *cizích klíčů* relativně snadná a zachovává konzistenci databáze. Pokud generátor narazí na sloupec, do kterého je nutné přiřadit hodnotu cizího klíče, využije nejprve lokální pomocnou databázi aktuálně generovaných hodnot a zní vybere výčet hodnot z referenční tabulky. V případě, že je aktivní připojení do databáze a nejedná se o export do souboru, se do výčtu referenčních hodnot připojí také již existující hodnoty z databáze. Pomocí metody `QuerySelect()`, budou vybrány všechny hodnoty z referenční tabulky z databáze.

5.2.4 Rozhraní

Rozhraní u samotného generátoru lze rozdělit na dva typy. Prvním je klasické uživatelské rozhraní, zobrazené na obrázku 5.4, které je zobrazitelné jak pod platformou Windows tak i Linux. Toto rozhraní obsahuje veškeré standardní prvky klasických aplikací. Skládá se z jednoduchého menu, tlačítky pro výběr souboru a pro začátek generování do databáze nebo do souboru. Dále se zde nachází čtyři vstupní pole, které lze využít pro zadání

přihlašovacích údajů do databáze, v případě, že uživatel nemá tyto informace vyplněné v šabloně. Tyto vstupní pole se zpřístupní hned po tom, co byla detekována absence přihlašovacích údajů, a nebo pokud tyto údaje nejsou validní. Pro případ, že uživatel nechce zadávat své údaje pro přihlášení nebo je například nezná, je zde možnost uložit celý SQL výstup do souboru, který následně lze vložit do databáze pomocí importu.



Obrázek 5.4: Ukázka rozhraní generátoru testovacích dat

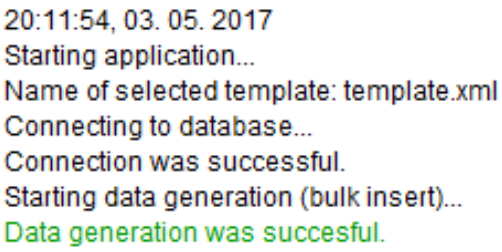
Další možností spuštění aplikace může být například pod platformou Linux, která nemusí nutně podporovat grafického rozhraní a tudíž je zde nutné spustit aplikaci s parametrem `-nogui`. Spolu s názvem programu a parametrem pro spuštění bez grafického rozhraní je nutno zadat jako další parametr také `-f` a jméno šablony. Pokud se mimo jména šablony zadá také specifický parametr `-e` spolu s vlastním názvem souboru, tak se provede generování testovacích dat pouze do zvoleného souboru, který se v případě, že neexistuje, vytvoří. V opačném případě započne generování dat přímo do databáze. Dalším, opět volitelným parametrem, je `-single`. Tento parametr změní metodu vkládání, na místo výchozího víceřádkového vkládání na jednořádkové. Tato metoda je využita především pro testovací účely, není nijak předpokládáno její praktické využití, jelikož je značně pomalejší. Posledním parametrem je `-t`, který na konci výpisu zobrazí jednotlivé časy pro běh samotné aplikace a pro generování dat.

5.2.5 Databáze dat

Databází dat v tomto případě je myšleno úložiště, odkud se berou surová data pro generování testovacích hodnot. V adresáři `db` jsou uloženy textové soubory, které obsahují místní databáze slov. Tyto zásobníky jsou složeny z jednotlivých jmen, příjmení, názvu ulic, apod. Každý řádek zde reprezentuje jeden záznam. Při vytváření nové instance třídy `Database` se nejdříve načtou do místního asociativního pole jednotlivé soubory, které je možno načítat také dynamicky v případě budoucího rozšíření o vlastní databáze. Následně se pomocí interní metody `LoadDB()` načtou veškerá surová data z místních databází do vnitřních struktur aplikace. Pomocí metody `GenerateData()`, která spadá pod třídu `Database` se vygenerují data. Tato metoda přijímá čtyři parametry a těmi jsou datový typ, kategorie z níž se mají testovací data vybírat a dvě dodatečné specifikace, které konkretizují podobu výsledných hodnot, jež se generovat. Může se jednat například o udání určitého rozsahu pro generování celých čísel nebo volby počty znaků pro generování textového řetězce.

5.2.6 Správa oznámení

Na první pohled relativně velkou část grafického rozhraní v aplikaci zabírá místo, kde se zobrazují zprávy, které zobrazují výsledky a aktuální dění aplikace. Zprávy jsou zobrazovány jak pro grafické rozhraní v tomto okně, tak i pro konzolovou verzi, přímo na obrazovku. Zprávy se vypisují až po dokončení daného úkonu, včetně času zahájení a veškerých kroků, které se provedly. Příklad výpisu oznámení lze vidět na obrázku 5.5. Zde je vidět úspěšný pokus o generování dat do databáze. Lze aktivovat i časový výpočet, který zobrazí čas generování a čas běhu samotné aplikace. Zprávy jako takové jsou pomocí statické metody `Message()` ukládaný do interního pole, spolu s typem této zprávy a samotným textem. Rozlišují se zde tři typy zpráv. Zprávy oznamovací, úspěšné a chybové. Rozlišení je pouze pro vizuální rozdíly v grafickém rozhraní. Kde chybové zprávy se zobrazují červeně a zprávy o úspěchu, zeleně. Při výpisu v konzoli není viditelný rozdíl. Jak bylo již dříve uvedeno, zprávy se vypisují až po dokončení veškerých úkonů generování. Pro výpis a aktualizaci grafického okna je využita metoda `PrintText()`.



Obrázek 5.5: Ukázka zobrazení oznamovací zprávy v hlavním okně aplikace.

5.2.7 Vkládání dat (databáze)

V kapitole o způsobech vkládání dat byly uvedeny různé možnosti a porovnání jejich efektivnosti. Pro tento případ byl ve výchozím nastavení využit nejefektivnější dostupný způsob vkládání testovacích dat do databáze. Tento princip se jmenuje *bulk insert* neboli víceřádkové vkládání. Oproti klasickému vkládání, po řádcích, se mnohem lépe hodí pro generování většího množství dat. V metodě `StartGenerate()` je implementován algoritmus pro tvorbu testovacích dat, ve spolupráci s generováním jednotlivých hodnot díky metodě `GenerateData()` se vytvoří řetězec. Ten obsahuje SQL příkazy a samotná data pro vložení do databáze. Na tento řetězec jsou akumulovány další hodnoty, což je smyslem víceřádkového vkládání. Následně se za pomoci metody `QueryInsert()` tento dotaz provede. Vkládání probíhá jednorázově pro každou tabulku. Základem algoritmu jsou dva resp. tři zanořené cykly. Hlavní cyklus slouží na procházení samotných tabulek. První zanořený cyklus je využíván pro vygenerování názvu sloupců. Další úsek je složen ze dvou vnořených cyklů, kde první se provede tolikrát, kolik chce uživatel vygenerovat řádků dat. Vnitřní cyklus poté prochází samotné sloupce tabulky a je využit pro generování samotných údajů odpovídajícím danému typu a kategorii.

5.2.8 Vkládání dat (soubor)

Pokud uživatel nechce vložit data do databáze, ať už z důvodu neznalosti přihlašovacích údajů nebo kvůli bezpečnosti při jejich zadávání, může je uložit do souboru. Ukládání dat do souboru využívá stejný algoritmus jako při vkládání do databáze. Využívá se zde ovšem jiná metoda, kterou je `FileInsert()`. Tato metoda vloží data do souboru, jehož název byl definován uživatelem na začátku generování. Vkládání dat do souboru je přístupné, jak už bylo zmíněno, i v případě neznalosti přihlašovacích údajů. Jedinou nevýhodou možnosti vkládání dat do souboru namísto databáze je ta, že generovaná data se berou pouze v rozsahu zadání šablony a nepočítají s existencí dalších dat v databází. Z tohoto důvodu by následný import výstupního souboru do již předvyplněné databáze nemusel fungovat. Jedná se zde především o unikátní hodnoty a cizí klíče.

5.2.9 Možná rozšíření

Jak už bylo dříve uvedeno, pro tvorbu tohoto nástroje byl pro obě dvě aplikace využit objektově orientovaný způsob programování. Díky tomu je správa aplikace a obecně její budoucí rozšiřování velice ulehčeno. Při vývoji aplikace se vyskytlo několik nápadů, které by do budoucna mohly využívání aplikace zjednodušit nebo obohatit o další možné funkce.

Jedná se například o spojení obou aplikací do jednoho, kde by tvorba šablony, pro případ častého použití a samotné generování dat bylo v jednom. Díky této úpravy by aplikace byla velice podobná ostatním nástrojům jež jsou aktuálně na trhu, s výjimkou využívání šablony.

Dalším nápadem byla editace šablony nebo editace konkrétních vkládaných dat přímo aplikaci pro generování dat. Může se jednat například i o změnu počtu dat, jež se mají generovat. Tato úprava by nikterak neovlivnila samotný paralelní běh obou dvou aplikací.

Zavedení regulárních výrazů pro definici tzv. *masky*. Tato úprava by nebyla příliš rozsáhlá, ale velice by rozšířila možnosti pro generování hodnot všech různých typů. Mohlo by se jednat například o specifikaci konkrétních formátů pro telefonní čísla nebo poštovní směrovací čísla.

Posledním návrhem pro rozšíření by bylo zavedení vlastních typů kategorií, kde by uživatel zadal vlastní kategorii, ke které by přiložil svoji vlastní databázi dat (slov, písmen, číslic) a z této databáze by se samotná data generovala.

Paralelní zpracování samotného generování. Kde každý by generační procesor vytvořil nové vlákno a mohlo by tak být zpracování například několik šablon zároveň. Problém by zde mohl ovšem být u databází, které by neumožňovaly vícenásobný přístup k datům.

Kapitola 6

Testování

Na začátku práce bylo zmíněno určité tvrzení a teorie ohledně rychlosti určitých způsobů vkládání dat do databáze. Potvrzení pravdivosti je důležitou součástí prokázání použitelnosti a důvěryhodnosti nástroje, jež je zde popisován a nyní i testován. Následující kapitola se tedy bude zabývat samotným testováním výsledné aplikace, jež je popisována v této práci. Testování se bude týkat především rychlosti samotného generátoru a rychlosti vkládání dat. Tyto dvě hodnoty jsou rozdílné, jelikož samotný generátor vyžaduje jak čas pro vygenerování testovacích dat, tak i čas pro samotné vložení. Posledním bodem testování bude především použitelnost v praxi a srovnání výsledků s již dostupnými generátory.

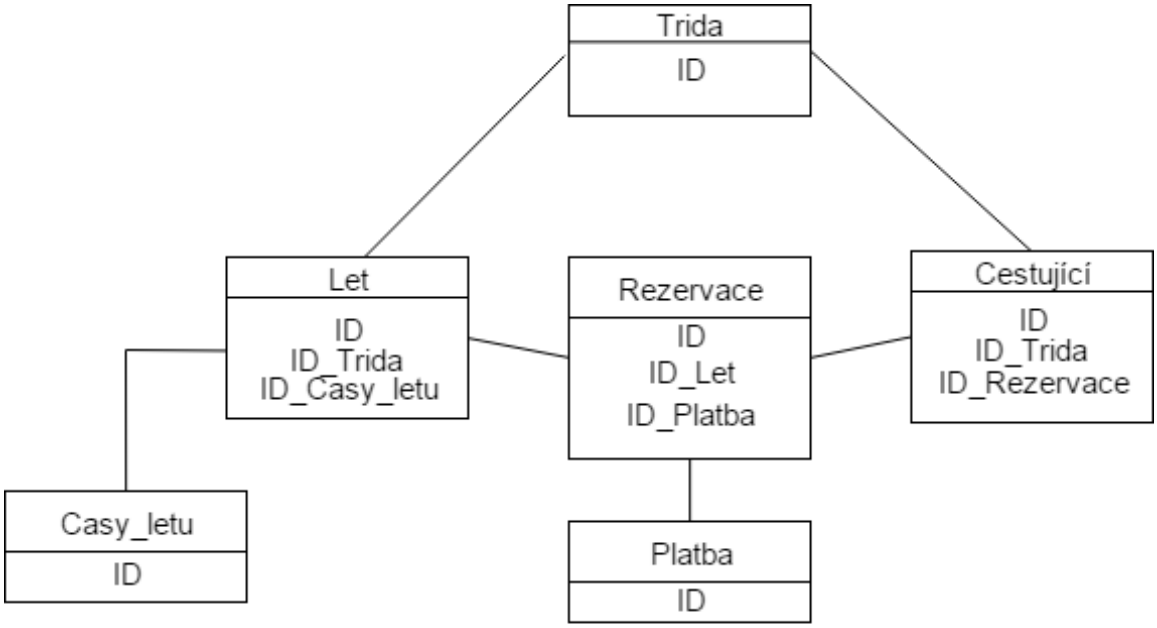
6.1 Testovací sada

Pro průkazné výsledky testování je třeba zvolit vhodnou testovací sadu. V tomto případě se bude jednat o několik různých databází. Vkládání dat bude probíhat pomocí dvou metod, kterými jsou, již dříve zmíněné víceřádkové a jednořádkové vkládání. Tyto dvě metody se pro testovací účely dají velice snadno přepínat v samotné aplikaci. Do testovacích databází se budou vkládat různé počty a typy data. Hlavním bodem testování bude porovnání rychlosti mezi oběma metodami vkládání dat. Pro všechny druhy databází byly vygenerovány šablony pomocí webové aplikace, která je opět popsána v této práci.

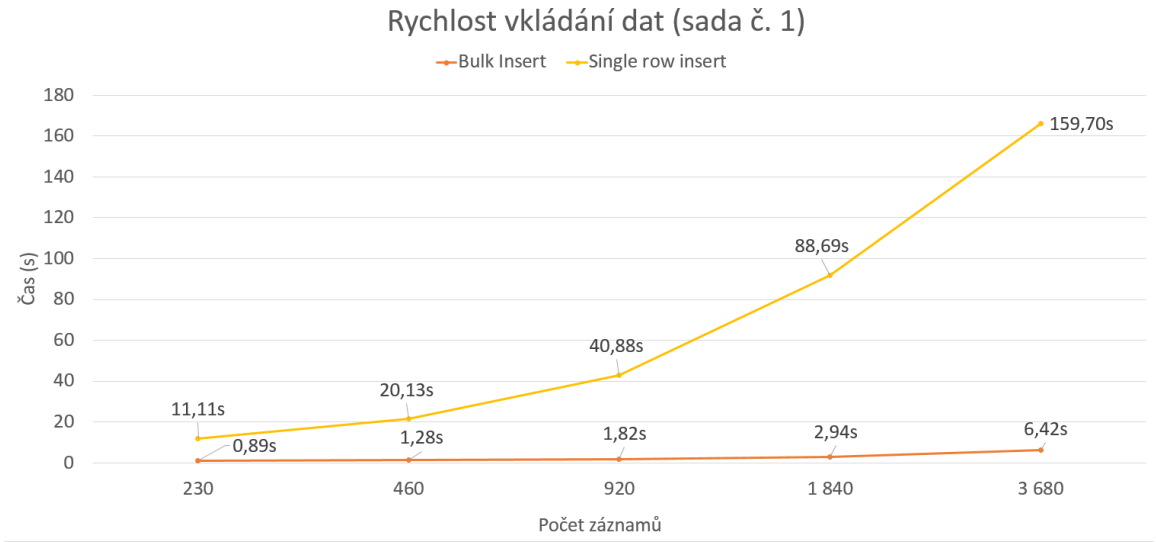
6.2 Databáze č. 1

První test využívá reálnou databázi, která byla použita v praxi. Tato databáze spadá pod systém rezervace letenek a letů pro zákazníky. Vyskytuje se zde několik různých tabulek se všemi možnými typy dat. Jsou zde využití unikátní hodnoty, primární a cizí klíče. Zjednodušený ERD diagram pro lepší představu lze vidět na obrázku 6.1. Grafické znázornění této testovací sady je pouze pro lepší znázornění komplexnosti databáze.

Graf 6.2 znázorňuje rychlost vkládání dat do databáze pro obě použité vkládací metody. Je zde názorně vidět rapidní časový rozdíl mezi metodou bulk insert, která je viditelně rychlejší, a metodou single row insert. Z grafu je také viditelné, že při rostoucím počtu záznamů se rychlost metody bulk insert zmenšuje téměř nepatrně. Pomalejší metoda vkládání má pro tento případ testovací sady několikanásobně pomalejší odezvu s rostoucím počtem vkládaných záznamů.



Obrázek 6.1: Zjednodušený ERD diagram první testovací databáze



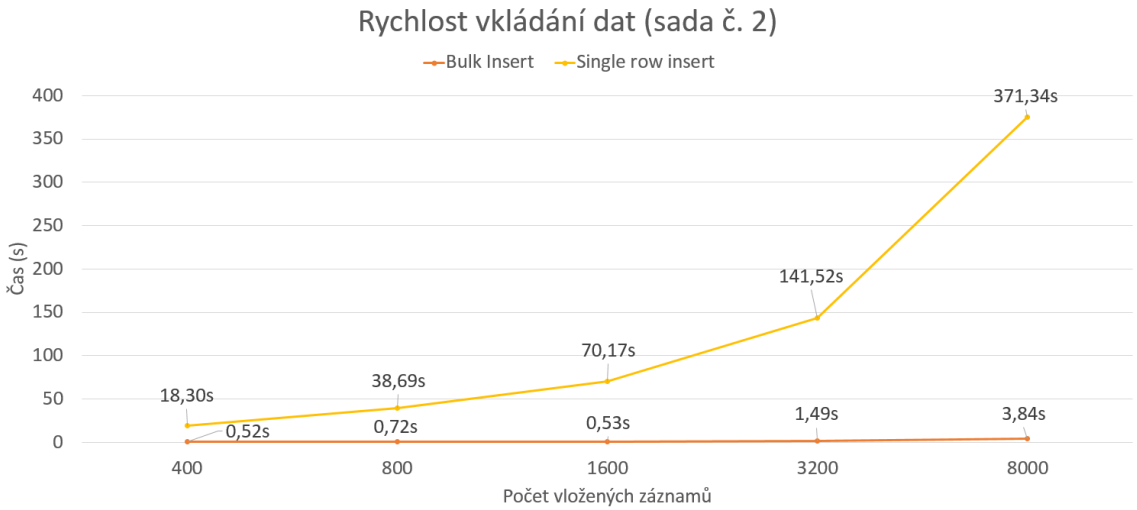
Obrázek 6.2: Graf zobrazující rychlost vkládání dat - sada č. 1

6.3 Databáze č. 2

Další test v řadě obsahuje čtyři stejně velké tabulky, kde každá má pět sloupců. Každá tabulka v základu generuje 100 záznamů pro obě dvě vkládací metody. Počet vkládaných záznamů se bude postupně zvyšovat pro každý test až do maximálních 16000 záznamů.

Tento test zjišťuje náročnost při vkládaní do více stejných tabulek při vyšších počtech záznamů.

Výsledek testu je možno vidět na grafu 6.3. Vyjímaje počet vkládaných dat je výsledek velice podobný s testovací sadou č. 1, kde je viditelně rychlejší opět metoda bulk insert. Zde je detailněji vidět, že časová odezva pro rychlejší metodu, začíná být trošku větší, v porovnání s metodou pomalejší je tento časový pokles zanedbatelný. Pro první krok se vkládá 100 záznamů do každé tabulky, což ve výsledku dělá 400 záznamů. Tyto záznamy jsou vloženy pomocí víceřádkového vkládání během 0.22 sekund. Oproti pomalejší metodě, je stejný počet záznamů vložen za 4,25 sekundy. V tomto konkrétním případu, je metoda bulk insert téměř 19 krát rychlejší. V posledním testovacím cyklu je vloženo 16000 záznamů. Pomocí rychlejší metody, je časová odezva necelé tři minuty. Pomocí pomalejší metody je to necelých 20 minut. Zde je metoda bulk insert rychlejší cca 7 krát. Jedná se zde o ideální případ, kdy databáze může obsahovat několik desítek tabulek a každá tabulka bude obsahovat optimální počet sloupců (5-12) a rychlost vkládaní bude více než dostatečná.

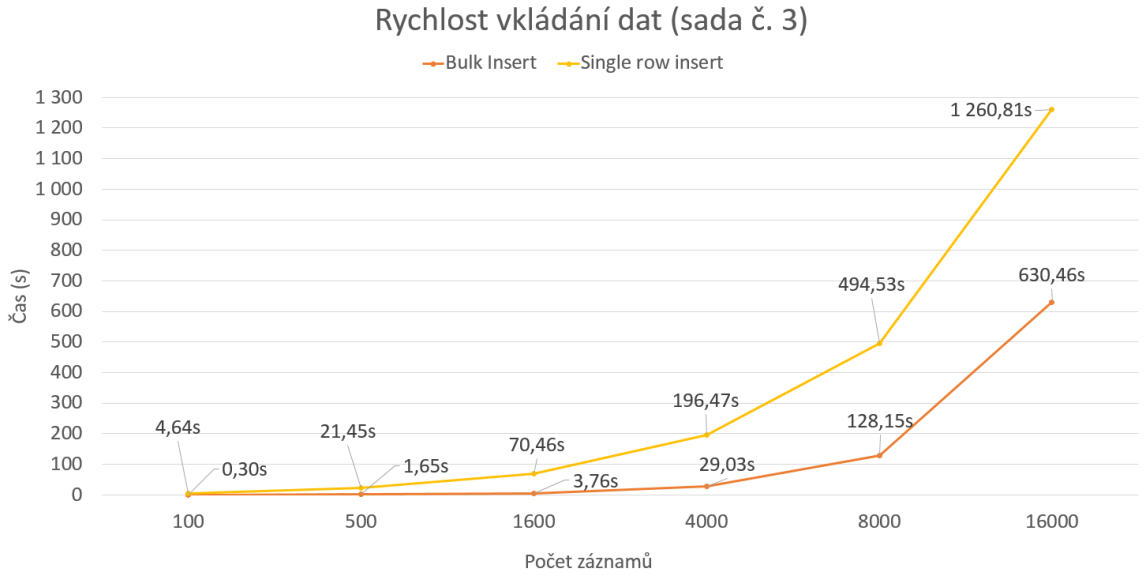


Obrázek 6.3: Graf zobrazující rychlost vkládání dat - sada č. 2

6.4 Databáze č. 3

Poslední testovací sada obsahuje jednu jedinou tabulku, která má relativně vysoký počet sloupců. V praxi se obecně nedoporučuje vytvářet takto rozsáhlé tabulky a jak již bylo zmíněno v kapitole o databázích, je důležité takto rozsáhlé informace rozdělovat do více tabulek. Pro tuto testovací databázi obsahuje tato jediná tabulka 20 sloupců. Testovat se zde bude opět časová odezva obou dvou vkládacích metod. Typy sloupců v tabulce jsou zde irelevantní.

Výsledný graf 6.4 zde zobrazuje poněkud jiné výsledky než předchozí dva grafy. Je zde velice viditelný výrazný pokles rychlost metody bulk insert, který je přímo úměrný zvyšujícímu se počtu vkládaných záznamů. Pro číselné prezentování lze vzít první a poslední hodnoty. Pro první případ je do tabulky vkládáno 100 záznamů. V tomto případě má rychlejší metoda čas 0,3 sekundy a pomalejší má 4,64 sekund. Zde je metoda **bulk insert** 15 krát rychlejší. V posledním kroku testování, je vkládáno celých 16000 záznamů. Zde je již ze samotného grafu viditelný vzestup časové křivky pro rychlejší metodu. Rychlejší metoda má zde časovou odezvu 10 a půl minuty. Což vzhledem k samotnému vkládání takto vysokého počtu dat není nikterak neobvyklé. Ovšem pomalejší metoda má zde časovou odezvu 21 minut. Což je přesně dvojnásobek rychlejší metody. Rychlostní poměr se zde velice zmenšil. V případě, že by počet hodnot rostl dál, byly by na tom obě metody časově velice podobně, ne-li stejně.



Obrázek 6.4: Graf zobrazující rychlost vkládání dat - sada č. 3

6.5 Výsledky testování

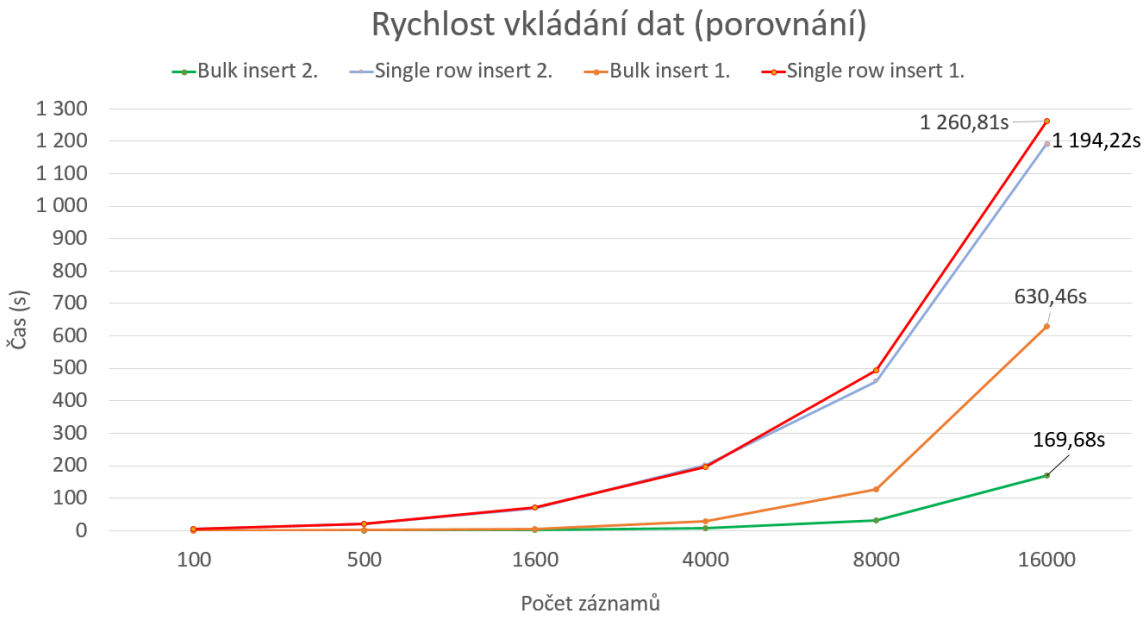
Testovací sady byly zvolené tak, aby pokryly většinu možností generování a otestování všech možných způsobů vkládání dat do různých situací. Vyskytla se zde databáze z praxe, která byla použita v informačním systému, databáze s více tabulkami a průměrným množstvím sloupečků a na závěr jediná tabulka s velkým množstvím sloupců.

Měřená byla jak doba běhu aplikace při generování dat, tak i doba samotného provedený příkazů pro vložení dat. Rozdíl těchto dvou časů byl ve většině případů okolo několika desetin vteřiny. Rychlost generování je mimo volby vkládací metody závislá také na hardwarovém vybavení počítače, na kterém je generátor spuštěn. Pro testovací účely sloužila soukromá stanice s *procesorem AMD FX-6300, 6 cores (3,5 Ghz), 16GB RAM*. Operační systém pro testovací účely byl Windows 10. Grafická náročnost aplikace je zanedbatelná. Pro přesnější výsledky testování bylo každé měření provedeno šestkrát za

sebou a výsledná hodnota byla aritmetickým průměrem naměřených hodnot.

Z dosažených výsledků je viditelné, že je metoda bulk insert výrazně rychlejší i při zvyšujícím se počtu vkládaných záznamů. Toto tvrzení bylo zmíněno v kapitole Tvorba testovacích dat do databáze a zde bylo pouze ověřeno. Dalším zjištěným důležitým faktem je, že při zvyšujícím se počtu sloupců v tabulce, se metoda bulk insert výrazně zpomaluje. Zpomalování probíhá v přímě úměře vůči rostoucímu počtu sloupců a zvyšujícímu se počtu vkládaných záznamů. V takovém případě může nastat chvíle, kdy bude rychlost vkládání u obou metod identická.

Demonstrování názorného rozdílu při vkládání stejného množství záznamů do větší tabulky o 20 sloupcích a do čtyř různých tabulek, kde každá má 5 sloupců, což ve výsledku dá stejné množství informačních hodnot, lze vidět na grafu 6.5. V obrázku je vidět rapidní rozdíly skupiny č. 1, kde jsou data vkládána do jediné rozsáhlé tabulky a skupinou č. 2 kde jsou čtyři menší tabulky. Pro metodu **single row insert** není změna nikterak viditelná, mnohem podstatnější je zde rozdíl pro metodu **bulk insert**, kde je několikanásobně rychlejší čas pro vkládání záznamů.



Obrázek 6.5: Graf zobrazující rychlost vkládání dat - sada č. 3

Kapitola 7

Závěr

Cílem bakalářské práce bylo navrhnout a vytvořit nástroj, který by sloužil pro automatické generování testovacích dat pro reálné relační databáze. Tento nástroj měl umět generovat testovací data pomocí předem vytvořené šablony. S tím úzce souvisí také samotný nástroj pro generaci této šablony. Na trhu aktuálně existuje nespočetné množství různých aplikací pro generování testovacích dat. Valná většina těchto nástrojů je ale bohužel zpoplatněna nemalými částkami. Vyjímaje poplatků za využívání, bývají tyto nástroje také velice obsáhlé a komplikované, což nikterak neusnadňuje jejich používání. Aplikace, jejíž vývoj je popsán v této práci se nesnaží nijak konkurovat ostatním nástrojům. Jde spíše vlastní cestou, což lze poznat například ve využití šablony a u samotného rozdělení nástroje do dvou menších aplikací. Jedna aplikace zde slouží jako generátor šablony a druhá figuruje jako samotný generátor dat, dle předem vytvořené šablony. Šablona je zde využita především pro opětovné generování různých dat do databáze. Díky tomu není nutné vyplňovat formulář s informacemi o generovaných datech pokaždé, kdy je třeba naplnit databázi. Generátor šablony je webová aplikace, kterou lze využívat online nebo ji lze nainstalovat na lokální počítač, kde je ovšem nutné mít vlastní lokální server a splnit některé další požadavky. Samotný generátor testovacích dat je desktopová aplikace spustitelná pod platformami Linux a Windows i bez podpory grafického uživatelského rozhraní.

První polovina této práce se věnuje převážně teoretickému pojednání o databázích obecně a o různých způsobech vkládání dat, do výše zmíněných, relačních databází. Byly zde také uvedeny dva stěžejní způsoby vkládání dat, víceřádkové vkládání (bulk insert) a jednořádkové vkládání (singel row insert). V teoretické části byly zmíněny výhody i nevýhody těchto dvou metod a také některá jejich úskalí. Pro ověření veškerých tvrzení bylo vyrobeno několik testovacích sad, na kterých se prováděly různé testy. Testy se týkaly především rychlosti vkládání testovacích dat. Testovaly se obě dvě vkládací metody, měřil se čas aplikace i čas vkládání samotných dat. Testy potvrdily, že víceřádkový způsob vkládaní dat, který byl považován za rychlejší a byl také využit pro implementaci nástroje, je značně rychlejší ve většině případů. Byla zde odhalena také jedna výjimka, kde víceřádkové vkládání ztrácí svoji rychlost. Tato výjimka se týká rapidně rozsáhlých tabulek, které mohou mít i více než 20 atributů.

Po zhodnocení výsledků testování a aplikace samotné jsme dospěli k názoru, že zde popisovaná aplikace je pro praktické využití automatizované tvorby dat pro testovací účely více než dostatečná. V rámci příslušného rozsahu se podařilo dosáhnout předem daných cílů,

ovšem je zde veliký prostor pro rozšiřování aplikace, jak bylo uvedeno i v kapitole možných rozšíření. Do budoucna je v plánu na nástroji dále pracovat.

Literatura

[1] Auer, David J. a Kroenke, David M.: *Databáze*. Computer Press, 2015, ISBN 978-80-251-4352-0.

[2] Axel Rauschmayer: *Speaking JavaScript*. O'Reilly Media, 2014, ISBN 978-1-4493-6503-5.

[3] Ed Lecky-Thomson and Steven D. Nowicki: *PHP 6 - Programujeme profesionálně*. Computer Press, 2010, ISBN 9788025131275.

[4] Eugene Philipov: *Comparing multiple rows insert vs single row insert with three data load methods*. [Online; navštíveno 3.5.2017].
URL https://www.simple-talk.com/sql/performance/comparing-multiple-rows-insert-vs-single-row-insert-with-three-data-load-methods/

[5] Grant Allen, Mike Owens: *The Definitive Guide to SQLite*. Apress, 2010, ISBN 9781430232254.

[6] Jaromír Skřivan: *Databáze a jazyk SQL*. [Online; navštíveno 3.5.2017].
URL https://www.interval.cz/clanky/databaze-a-jazyk-sql/

[7] Jonathan Chaffer, Karl Swedberg: *Learning jQuery - Fourth Edition*. Packt, 2013, ISBN 9781782163145.

[8] Martin Štrbák: *Příprava testovacích dat*. [Online; navštíveno 3.5.2017].
URL
http://cz-testing.blogspot.cz/2011/03/priprava-testovacich-dat-i.html

[9] Mary Campione: *JFC Swing Tutorial*. PEARSON, 2004, ISBN 0201914670.

[10] Microsoft: *Microsoft - Developer Network - SQL Server - BULK INSERT*. [Online; navštíveno 3.5.2016].
URL https://msdn.microsoft.com/en-us/library/ms188365.aspx

[11] Molly E. Holzschlag: *HTML a CSS jdi do toho*. Grada, 2006, ISBN 80–247–1454-X.

[12] Pavel Herout: *Java a XML*. Kopp, 2007, ISBN 8072323074.

[13] Pavel Herout: *Java - Učebnice jazyka*. Kopp, 2010, ISBN 9788072323982.

[14] SitePoint Pty. Ltd.: *SQL vs NoSQL: The Differences*. [Online; navštíveno 3.5.2017].
URL http://www.sitepoint.com/sql-vs-nosql-differences/

[15] The PHP Groupu: *PHP Manual - DOMDocument*. [Online; navštíveno 8.5.2017].
URL http://php.net/manual/en/class.domdocument.php

[16] The PHP Groupu: *PHP Manual - SimpleXML*. [Online; navštíveno 8.5.2017].
URL http://php.net/manual/en/book.simplexml.php

[17] Tomáš Horváth: *Teoretický úvod do relačních databází*. [Online; navštíveno 4.5.2017].
URL http://programujte.com/clanek/2007110801-teoreticky-uvod-do-relacnich-databazi/

[18] Tvorba-webu: *XML základy*. [Online; navštíveno 3.5.2017].
URL http://www.tvorba-webu.cz/xml/

Přílohy

Obsah přiloženého CD

- Generátor šablony
  - `TemplateGenerator.exe` - instalační soubor pro Windows
  - `TemplateGenerator.zip` - archiv s webovou aplikací (zdrojové kódy)
  - `readme.txt` - popis spuštění a běhu aplikace
- Generátor dat
  - `generator.jar` - hlavní aplikace pro generování dat
  - `codes` - archív se zdrojovými kódy aplikace
  - `readme.txt` - popis spuštění a běhu aplikace
- `doc.pdf` - obsahuje tuto technickou zprávu ve formátu PDF
- Testy - adresář obsahující všechny informace a data k testovacím sadám
- testy.xlsx - výsledky testů včetně tabulek a grafů