VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

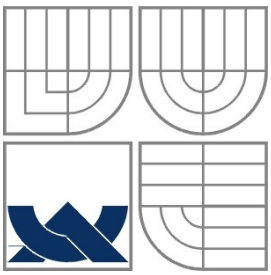
NÁSTROJ PRO GENEROVÁNÍ OBSAHU DATABÁZE PRO ÚČELY TESTOVÁNÍ SOFTWARU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

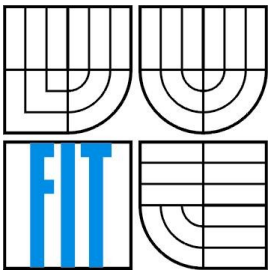
AUTOR PRÁCE
AUTHOR

DUŠAN ŽELIAR

BRNO 2016



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁSTROJ PRO GENEROVÁNÍ OBSAHU DATABÁZE PRO ÚČELY TESTOVÁNÍ SOFTWARU

TOOL FOR GENERATING DATABASE CONTENT FOR TESTING PURPOSES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DUŠAN ŽELIAR

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ALEŠ SMRČKA, Ph.D.

BRNO 2016

Abstrakt

Cieľom bakalárskej práce je vytvoriť nástroj na generovanie obsahu databázy pre účely testovania. Podstatou nástroja je vytvorenie náhodných dát, ktoré spĺňajú zadané obmedzenia. Obmedzenia popisujú štrukturálne väzby v relačnej databáze a sémantické závislosti medzi hodnotami jednotlivých stĺpcov tabuliek. Výsledkom tejto práce je nástroj, ktorý podľa definovaných obmedzení postupne generuje a vkladá dáta do databázy. Záver práce obsahuje demonštráciu funkcionality na databáze reálneho charakteru.

Abstract

The aim of this bachelor's thesis is to create a tool for generating database content for testing purposes. The main function of the generator is to create random data which meets defined constraints. Constraints describe structural relationships in a database as well as semantic dependencies among columns in tables. The result of the thesis is a tool which generates test and inserts data into database based on defined constraints. The last part of the thesis contains demonstration of the functionality on a real database.

Klíčová slova

testovanie, testovacie dáta, generátor, generovanie obsahu databázy, relačné databázy, python

Keywords

testing, test data, generator, generating database content, relational databases, python

Citace

Dušan Želiar: Nástroj pro generování obsahu databáze pro účely testování softwaru, bakalářská práce, Brno, FIT VUT v Brně, 2016

Nástroj pro generování obsahu databáze pro účely testování softwaru

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Aleša Smrčku, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Dušan Želiar
18. května 2016

Poděkování

Tímto by som sa rád poďakoval svojmu vedúcemu práce Ing. Alešovi Smrčkovi, Ph.D. za cenné rady, odborné konzultácie a ochotu pri tvorbe práce.

Obsah

Kapitola 1

1 Úvod

Testovanie softwaru je proces overujúci správnosť, úplnosť a kvalitu vyvíjaného programu. Všeobecne sa tento proces skladá z nastavenia prostredia, vykonania testu, overenia dosiahnutých výsledkov a odstránenia vedľajších účinkov testu. Programy, ktoré komunikujú s databázami, sú hlavnou zložkou v systémoch, ktoré vyžadujú prístup k persistentným dátam. Pri testovaní daných systémov sa pristupuje k dátam uložených v databáze, čo vyžaduje nastavenie vhodného testovacieho prostredia databázy. Pre tento účel existuje množstvo nástrojov, ktoré dokážu vytvoriť prostredie databázy, nekomerčné nástroje majú ale obvykle obmedzenú funkcionalitu.

1.1 Motivácia

Moja práca sa zaoberá popisom problematiky, návrhu a implementácie nástroja, ktorý slúži ako generátor náhodných dát pre účely vytvorenia testovacieho prostredia, konkrétne pre testovanie programov pracujúcich s relačnými databázami. Dáta v databáze musí tester obvykle ručne vytvárať a vkladať, čo obmedzuje ich množstvo a rôznorodosť. Nástroj by pomohol testerom s počiatočným naplnením databázy dátami a tým vytvoril základ testovacieho prostredia.

Hlavná požiadavka na tento nástroj je generovanie náhodných dát, ktoré zároveň musia zachovať vnútorné obmedzenia v relačnej databáze. Obmedzenia sa týkajú schémy databázy, ale aj sémantiky hodnôt jednotlivých stĺpcov tabuliek.

Cieľom práce je vytvoriť nástroj, ktorý pomôže programátorovi vytvoriť testovacie prostredie databázy. Vstupom nástroja je sada obmedzení popisujúca cieľovú databázu a jej vnútorné vzťahy. Charakter reálnych dát je zabezpečený formou datasetov. Datasety môžu byť vytvorené pre konkrétny test alebo všeobecne vložené medzi základné datasety, ktoré sú ďalej prístupné. Tester špecifikuje množstvo a charakter generovaných dát. Nástroj automaticky vytvorí dáta a vkladá ich do databázy v poradí, ktoré rešpektuje obmedzenia v databáze.

Kapitola 2 popisuje požiadavky na generovanie dát, charakter relačných databáz a poskytne pohľad na existujúce nástroje slúžiace pre účely generovania dát. V kapitole 3 je vysvetlený princíp fungovania nástroja a popis jazyka obmedzení. Kapitola 4 popisuje zaujímavé implementačné detaily a obsahuje testy funkcionality nástroja.

Kapitola 2

2 Vytváranie obsahu databázy

Pri testovaní systémov s databázami sa predpokladá práca so systémom riadenia bázy dát. Dané systémy tvoria rozhranie medzi uloženými dátami a testovaným programom a nie sú súčasťou testov. Pri testovaní systémov s databázami vytvoríme pre časť schémy databázy, ktorú použijeme pri testovaní, reálnu kópiu s dátami. Obsah testovanej databázy môžeme vytvoriť pomocou náhodných dát, ktoré vychádzajú z pripravených datasetov alebo generátorov. Použitie reálnych dát uľahčuje prácu testera. Umožňuje lepšie pochopiť zmysel dát a priblíži testovaný systém k reálnym podmienkam, v ktorých bude nasadený.

2.1 Relačné databázy

Databáza je organizovaný register perzistentných dát v elektronickej podobe. Relačné databázové systémy sú založené na relačnom modely. Dáta sú na logickej úrovni štruktúrované do tabuliek. Tabuľka sa skladá z riadkov a stĺpcov. Riadky nazývame jednoucho riadky alebo záznamy a stĺpce atribúty.

Pre relačné databázové systémy sú definované integritné pravidlá, ktoré obmedzujú dáta a musia platiť v každej relačnej databáze. Integrita databázy je dodržaná, pokiaľ sú dáta v nej uložené konzistentne voči definovaným obmedzeniam. Obmedzenia sa môžu týkať jednotlivých hodnôt atribútov, kombinácií hodnôt v jednom zázname alebo celej množiny záznamov vybraného typu. Integritné obmedzenia sa rozdeľujú do kategórií. Entitná integrita sa zaoberá jednoznačnou identifikáciou záznamu v tabuľke. Primárny kľúč predstavuje hodnotu atribútov tabuľky, ktorá identifikuje záznam. Častým integritným obmedzením je referenčná integrita. Hodnota atribútu záznamu tabuľky závisí od hodnoty inej tabuľky, ktorá musí byť dostupná prostredníctvom referencie, čím medzi nimi definuje vzťah. Doménová integrita obmedzuje atribúty tabuľky prostredníctvom dátového typu alebo obmedzenia rozsahu hodnôt.

2.2 Testovanie systémov pracujúcich s relačnými databázami

Pri testovaní systému pracujúceho s databázami môže mať tester rôzne požiadavky na obsah testovanej databázy. Pri použití generátora sú to nasledovné:

- vytvorenie hraničných hodnôt dát
- vytvorenie špeciálnych hodnôt dát
- vytvorenie náhodných hodnôt dát
- vytvorenie reálnych hodnôt dát
- vytvorenie zmysluplných hodnôt dát
- zachovanie vnútorných väzieb a integrity databázy
- špecifikácia štrukturálnych väzieb v databáze
- špecifikácia sémantických väzieb v databáze
- špecifikácia množstva dát
- špecifikácia poradia generovania dát
- preddefinované generátory a datasety
- vytváranie nových generátorov a datasetov
- grafické užívateľské rozhranie

- rozhranie pre príkazový riadok
- automatická detekcia väzieb v databáze
- manuálne zadanie väzieb v databáze

2.2.1 Existujúce nástroje pre generovanie dát

Táto bakalárska práca naväzuje na myšlienku pôvodne implementovanú v bakalárskej práci Alici Minářovej.[5] Nástroj preskúma textový vstup databázy, podľa ktorého zistí jej schému. Na základe schémy potom generuje náhodné dáta pre konkrétne dátové typy databázy, ktoré následne do nej vloží.
Výhody: Nástroj zachováva väzby primárnych a cudzích kľúčov a pracuje plne automaticky.
Nevýhody: Dáta sú úplne náhodné, neumožňujú špecifikáciu ich charakteru.

MockJSON je webový nástroj disponujúci základnými generátormi. Na základe JSON vstupného reťazca obsahujúceho šablóny, vytvorí v rovnakom jazyku požadované náhodné dáta.
Výhody: Jednoduchá práca s nástrojom, umožňuje vytvárať vnorené dátové typy, voľne prístupný.
Nevýhody: Štruktúra vytvorených dát nie je vhodná pre ich priame vloženie do databázy, neumožňuje vytváranie nových datasetov.[3]

Generatedata je grafický webový nástroj pre vytváranie testovacích dát. Umožňuje špecifikovať hodnoty stĺpca tabuľky prostredníctvom generátorov.
Výhody: Podporuje referenciu hodnôt stĺpcov v rámci jednej tabuľky a ich využitie v definovaných výrazoch. Výstup dát ponúka vo forme programovacích jazykov, jazykov pre serializáciu a jazyka SQL.
Nevýhody: Nekomerčná verzia nedovoľuje pridávanie nových datasetov, absentuje možnosť vytvorenia viacerých tabuliek.[1]

Mockaroo je grafický webový nástroj generujúci náhodné dáta. Disponuje množstvom základných datasetov a generátorov. Výstup nástroja je formátovaný v podobe rôznych jazykov pre serializáciu dát.
Výhody: Podporuje regulárne výrazy a výrazy podporujúce jednoduchú logiku.
Nevýhody: Nekomerčná verzia má obmedzený maximálny počet vytvorených riadkov tabuľky. Absencia generovania viacerých tabuliek súčasne a referencií hodnôt medzi nimi.[2]

DbForge Data Generator je grafická aplikácia generujúca obsah databázy. Aplikácia je oproti ostatným nástrojom oveľa komplexnejšia a poskytuje väčšiu funkcionalitu.
Výhody: Umožňuje generovať viac tabuliek naraz, vytvárať nové datasety a generátory kombináciou základných generátorov, referovať hodnotu mimo aktuálnej tabuľky a disponuje veľkou množinou základných generátorov.
Nevýhody: Komerčná aplikácia, cena produktu.[2]

Kapitola 3

3 Návrh nástroja pre generovanie náhodných dát

Táto kapitola sa venuje návrhu fungovania nástroja a popisuje jazyk vytvorený pre popis obmedzení a nastavení generátora.

3.1 Špecifikácia požiadaviek

Požiadavky na výsledný nástroj čiastočne pokrývajú vlastnosti vypísané v kapitole 2.2, konkrétne sú to tieto:

- vytvorenie zmysluplných hodnôt dát
- vytvorenie náhodných hodnôt dát
- vytvorenie reálnych hodnôt dát
- zachovanie vnútorných väzieb a integrity databázy
- špecifikácia štrukturálnych väzieb v databáze
- špecifikácia sémantických väzieb v databáze
- umožňuje väzby medzi stĺpcami v rámci rozdielnych tabuliek
- špecifikácia množstva dát
- automatické vytvorenie poradia generovania dát
- preddefinované generátory a datasety
- vytváranie nových generátorov a datasetov
- rozhranie pre príkazový riadok

3.2 Jazyk popisujúci obmedzenia a nastavenia generátora

Vytvorený jazyk umožňuje popis obmedzení a nastavení generátora. Obmedzenia popisujú štrukturálne väzby v relačnej databáze a sémantické závislosti medzi jednoduchými dátovými typmi. Nastavenia zahŕňajú špecifikáciu počtu dátových typov, ktoré sú popísané v podkapitole 3.3 a ich mapovanie do databázy. Pre tieto účely ja vhodné využiť existujúci jazyk, ktorý serializuje dáta. Sú preň vytvorené vhodné knižnice pre spracovanie hodnôt a kontrolu validity ich zápisu. Pre účely generátora je použitý jazyk YAML.[8] Konkrétny vstup pre generátor v tomto jazyku obsahuje štyri až päť častí, ktoré má každý vstupný súbor. Nasledujúce názvy predstavujú kľúče slovníka vo vstupnom súbore.

- `types` – definícia nových dátových typov, ktoré predstavujú schému databázy
- `schema` – označuje väzby medzi dátovými typmi a stĺpcami tabuliek
- `constraints` – označuje sekciu dokumentu definujúcu sémantické obmedzenia jednoduchých dátových typov
- `generate` – špecifikuje množstvo dátových typov
- `include` – voliteľná časť, obsahuje názvy súborov uchovávajúcich preddefinované dátové typy, ktoré budú sprístupnené

3.2.1 Jazyk YAML

YAML je jazyk definujúci štandard pre serializáciu dát. Je navrhnutý tak, aby bol ľahko čitateľný človekom. Využíva formát v tvare *<kľúč> : <hodnota>*. Udržuje hierarchiu štruktúrovaných dát a podporuje unicode kódovanie. Základné komponenty jazyka, ktoré sú využité v nástroji sú:

- Komentár – začína bielym znakom nasledovaným “ # “, platí do konca riadku
- Štruktúra – názov nasledovaný “: “, obsiahnuté dáta sú v novom riadku a oddelené odsadením od ľavého okraja
- List – typ štruktúry, začína znakom “ – “ nasledovaný hodnotou
- Mapa – typ štruktúry, obsahuje kľúče s názvom oddelené “: “ a nasleduje hodnota
- Úvodzovky – rušia význam špeciálnych znakov

Príklad YAML dokumentu ktorý obsahuje štruktúry typu slovník (záznam, adresa) a typu list (študent) môže vyzerať takto:

```
záznam:
    #komentár
    adresa:
        ulica: Božetěchova
        číslo: 1 / 2
        psč: 612 00
        mesto: Brno
    študent:
        - Václav Novák
        - Martin Svoboda
    typ: “A4[|#]fg“
```

3.2.2 Popis štrukturálnych obmedzení

Pojem štrukturálne obmedzenie je vymedzený pre túto prácu ako dátový typ predstavujúci časť schémy databázy využívanej pri generovaní. Každá tabuľka databázy je prevedená na zložený dátový typ. Zložené dátové typy budeme ďalej nazývať štruktúry. Štruktúry predstavujú schému testovanej databázy, jej hierarchiu a relačné vzťahy jednotlivých tabuliek. Všetky štruktúry dokopy vytvárajú strom zložený z jednoduchých a zložených dátových typov.

Pri vytvorení nového dátového typu je počet hodnôt, ktoré nadobudne implicitne jedna, môže sa však špecifikovať pomocou kľúčového slova *one_of(cieľ)<počet>* alebo skrátene “*[cieľ]<počet>*“. Tvar špecifikácie počtu dátových typov:

- `?` - odpovedá intervalu nula až jedna
- `*` - odpovedá intervalu nula až globálne maximum
- `+` - odpovedá intervalu jedna až globálne maximum
- `{n}` - odpovedá počtu n
- `{m,n}` - odpovedá intervalu m až n

Uvedené symboly majú v jazyku YAML špeciálny význam. Preto výraz, ktorý ich používa, býva ohraničený úvodzovkami. Príklad vytvorenia nového dátového typu z dátového typu *int*, ktorý pri generovaní nadobudne postupne osem hodnôt: `new_datatype: “[int]{8}“`
Každá štruktúra dodržuje nasledovný formát, počet vnorených hodnôt je neobmedzený:

```
<dátový typ>:
    <hodnota>: <dátový typ/generátor><počet>
```

3.2.3 Popis sémantických obmedzení

Pojem sémantické obmedzenie je vymedzený pre túto prácu ako obmedzenie hodnoty dátového typu. Sémantické obmedzenia sa týkajú jednoduchých dátových typov. Ovplyvňujú ich hodnoty a

umožňujú definovať závislosť jednoduchých dátových typov naprieč štruktúrami. Obmedzenia sa definujú v tvare *<cieľový dátový typ> <operátor> <výraz>*. Podporované operátory sú `<`, `>`, `>=`, `<=` a `=`. Špeciálny operátor in definuje naraz hodnoty maxima a minima, čím vykonáva funkciu operátorov `>=` a `<=`. Tvar tohto výrazu je *<cieľový dátový typ> in* `(`*<výraz_pre_minimum>*`,` *<výraz_pre_minimum>*`)`. Povolené hodnoty operátorov a tvarov výrazov záležia na type generátora, z ktorého je dátový typ vytvorený.

3.2.4 Obmedzenia vytvorené ručne

Pri vytváraní obmedzení vychádzame zo schémy databázy. Najprv identifikujeme časti databázy, ktoré môžu byť samostatne generované. Sú to tabuľky v relačnom vzťahu 1:1 a 1:M. Stromy týchto tabuliek sú jasne definované a pri generovaní ich hodnôt sa môžu navzájom referovať. Tabuľky vzťahu M:N môžu byť pre jednoduchosť prevedené na vzťah 1:1, alebo môžu využiť špeciálne referovanie na hodnoty vygenerovanej tabuľky. V prípade špeciálnej referencie si určíme zdrojovú tabuľku a tabuľku, ktorá má na zdrojovú referenciu. Najskôr sa vygenerujú dáta zdrojovej tabuľky. Pri generovaní tabuľky s referenciou sa náhodne vyberie jeden z vygenerovaných záznamov zdrojovej tabuľky. Jej hodnoty sú skopírované do hodnôt tabuľky s referenciou.
Príklad vytvorenia vzťahu 1:N :

```
tableN:
    column1:generator
    column2:generator

table1:
    column1:generator
    substructure: “[tableN]{n}“
```

Príklad ukazuje dve štruktúry priamo reprezentujúce tabuľky, kde sa štruktúra *tableN* nachádza ako podštruktúra v *table1*. Pri každom vygenerovaní štruktúry *table1* sa vygeneruje n štruktúr *tableN*. Pokiaľ sa následne tieto hodnoty mapujú do tabuľky, generované hodnoty si tento vzťah zachovajú.

3.2.5 Obmedzenia vytvorené poloautomaticky

Sada štruktúr schémy databázy môže byť automaticky predpripravená pre testera pre jej ďalšie úpravy. Touto problematikou sa zaoberá bakalárka práca Natálie Loginovej.[]

3.3 Dátové typy v popise databázy

Dátový typ v tejto práci označuje objekt s definovanými hodnotami, ktoré môže nadobúdať. V nami vytvorenom jazyku sú to práve dátové typy, pomocou ktorých je popísaná štruktúra databázy.
Dátové typy môžeme rozdeliť na jednoduché a zložené. Jednoduché dátové typy sú buď preddefinované alebo vytvorené z generátora či iného jednoduchého dátového typu. Zložené dátové typy nazveme štruktúry, ktoré obsahujú v sebe jednoduché a zložené dátové typy. Prístup k hodnotám zložených dátových typov je prostredníctvom operátora bodka. Formát prístupu je `<zložený_dátový_typ>.<dátový_typ_v_štruktúre>`. Zložený dátový typ, ktorý nie je vnorený v žiadnom zloženom dátovom type, teda sa dá adresovať bez operátora bodka, nazveme koreňový dátový typ. Maximálna hĺbka zanorenia dátových typov nie je obmedzená.

3.3.1 Preddefinované dátové typy

Preddefinované dátové typy majú jednoznačne určený charakter dát, ktoré obsahujú. Obmedzenia nad nimi predstavujú špecifikáciu hodnôt, ktoré môžu nadobúdať. Každý preddefinovaný dátový typ vychádza z generátora, ktorý overuje validitu ich hodnôt a obmedzení, podľa ktorých následne generuje hodnoty. Medzi jednoduché preddefinované dátové typy patria:

- `int` – celé čísla so znamienkom v rozsahu od -2147483648 do 2147483647

- `uint` – celé nezáporné čísla v rozsahu od 0 do 2147483647
- `string` – reťazec maximálnej dĺžky 100 znakov, obsahuje malé a veľké znaky abecedy a čísiel
- `float` – desatinné čísla so znamienkom v rozsahu od -3.402823466E+38 do 3.402823466E+38
- `datetime` – dátum a čas v rozsahu od 1-1-1 00:00 do 9999-1-1 00:00, umožňuje prístup ku konkrétnym hodnotám zložiek času tohto dátového typu. Zložky času sú rok(year), mesiac(month), deň(day), hodina(hour), minúta(minute) a sekunda(second). Prístup k týmto zložkám je možný prostredníctvom operátora bodka. Formát prístupu je <dátový typ>.<časová zložka>
- `xdigit` – jednoznakový reťazec hexadecimálnych znakov
- `alnum` – jednoznakový reťazec alfanumerických znakov

Medzi preddefinované dátové typy môžeme zaradiť aj dátové typy získané z uložených datasetov. Tieto datasety sú uložené v súboroch, ktorých mená špecifikujeme v časti include.

3.3.2 Vytváranie nových dátových typov

Nové jednoduché dátové typy vychádzajú z preddefinovaných alebo užívateľom vytvorených dátových typov. Môžu byť rovnako priamo vytvorené z generátora. Nový typ má stanovený interval hodnôt, ktoré nadobúda. Ak je vytvorený z existujúceho dátového typu, tieto hodnoty majú rovnaké. Pri vytvorení priamo z generátora je obor hodnôt definovaný inicializačnými hodnotami generátora. Pri vytváraní zložených dátových typov sa vytvorí štruktúra, ktorej hodnoty sú jednoduché alebo zložené. Všeobecne platí formát pre jednoduchý dátový typ:

```
<názov dátového typu>: <dátový typ alebo generátor><počet>
```

Formát pre zložený dátový typ s jedným až N dátovými typmi:

```
<názov dátového typu>:
    <názov dátového typu>: <dátový typ alebo generátor><počet> +
```

Príklad vytvorenia dátových typov:

```
kľúč_a:
    kľúč_b: “[float]{3}“

kľúč_1:
    kľúč_2: “[int]{4,20}“
    kľúč_3: bool
    kľúč_4: interval(1,5)
    kľúč_5: kľúč_a
```

Kľúč_a je nový zložený dátový typ obsahujúci jeden dátový typ *kľúč_b* vytvorený z dátového typu float. *Kľúč_1* predstavuje nový zložený dátový typ, ktorý sa skladá zo štyroch dátových typov. Dátové typy *kľúč_b* a *kľúč*_2 majú zadaný počet hodnôt, ktorý obsahuje. Počet ostatných je implicitne jedna.

3.3.3 Mapovanie dátových typov do tabuliek databázy

Dátové typy ktoré figurujú aj ako dátové typy v databáze, sa musia spolu prepojiť. Toto prepojenie nazveme mapovanie. Mapovanie prebieha v sekcii vstupného dokumentu označeného *schema*. Najprv je adresovaná tabuľka databázy. V nej sú kľúče názov stĺpcov s adresou jednoduchých dátových typov. Adresa predstavuje cestu od koreňa dátových typov cez zložené dátové typy až po konkrétny jednoduchý. Príklad formátu mapovania:

```
schema:
    <názov tabuľky>:
        <názov stĺpca> : <zložený_dátový_typ.jednoduchý_dátový_typ>
        <názov stĺpca> : <zložený_dátový_typ.jednoduchý_dátový_typ>
        <názov stĺpca> : <zložený_dátový_typ.jednoduchý_dátový_typ>
```

```
<názov tabuľky>:
    <názov stĺpca> : <zložený_dátový_typ.jednoduchý_dátový_typ>
    <názov stĺpca> : FROM <tabuľka.stĺpec>

<názov tabuľky>:
    <názov stĺpca> : <zložený_dátový_typ.jednoduchý_dátový_typ>
    <názov stĺpca> : <zložený_dátový_typ.jednoduchý_dátový_typ>
```

3.4 Generátory náhodných dát jednoduchých dátových typov

Každý jednoduchý dátový typ obsahuje generátor dát, ktorý využíva pre účely získania konkrétnej hodnoty dátového typu. Generátor je objekt, ktorý má určený vlastný obor hodnôt a proces generovania dát. Generovanie je proces, ktorého výsledkom je konkrétna hodnota, ktorá patrí do oboru hodnôt generátora. Obor generovaných hodnôt je možné dynamicky meniť podľa sémantických obmedzení dátového typu v ktorom sa generátor nachádza. V danej sekcii sú popísané procesy generovania a podporované sémantické obmedzenia jednotlivých implementovaných generátorov.

3.4.1 Generátor typu konštanta

Generátor typu konštanta generuje zadanú konštantu v podobe reťazca. Generovanie hodnoty vždy vráti rovnakú konštantu. Nad dátovým typom s týmto generátorom nemôžeme vykonať žiadne sémantické úpravy hodnôt. Formát zápisu generátora: *constant(konštanta)*

3.4.2 Generátor typu integer v intervale

Generátor typu integer v intervale generuje celé číslo v určitom intervale. Interval je zadaný konštantami min a max pri vytvorení dátového typu z generátora. Ak je dátový typ vytvorený z iného dátového typu, konštanty minimálnej a maximálnej hodnoty z neho preberie. Formát zápisu generátora je *interval(min,max)*.

Dátový typ vytvorený z takéhoto generátora môže byť ďalej špecifikovaný v sémantických obmedzeniach, ktoré nastavujú nové minimálne a maximálne hranice intervalu generovanej hodnoty. Povolené operátory pre dátový typ s daným generátorom sú <, >, >=, <=, = a *in*. Výraz sa skladá z celých čísiel, referencií na jednoduchý dátový typ generujúci celé čísla a operátorov +, -, /, *. Počet sémantických obmedzení nie je limitovaný.

Proces generovania začne vyhodnotením výrazov sémantických obmedzení intervalu, ktorých výsledkom sú konkrétne hodnoty. Nasleduje výber najmenšej hodnoty z definovaných maximálnych hodnôt a najväčšej hodnoty z minimálnych hodnôt. Všeobecne musí byť minimálna hodnota menšia alebo rovná maximálnej, inak pri zadaných podmienkach neexistuje hodnota ktorá by ich všetky spĺňala.

3.4.3 Generátor postupnosti typu integer v intervale

Generátor postupnosti typu integer v intervale generuje celé číslo v určitom intervale. Pri generovaní novej hodnoty vychádza z naposledy vytvorenej hodnoty, prípadne z minimálnej hodnoty, ktorá sa zvýši o hodnotu prírastku. Interval je zadaný konštantami *min*, *max* a prírastok konštantou *increment* pri vytvorení dátového typu z generátora. Ak je dátový typ vytvorený z iného dátové typu, konštanty z neho preberie. Formát zápisu generátora je *seq(min, max, increment)*. Dátový typ vytvorený z takéhoto generátora sa nedá ďalej špecifikovať v sémantických obmedzeniach.

3.4.4 Generátor typu float v intervale

Generátor typu float v intervale generuje reálne číslo v určitom intervale. Interval je zadaný konštantami reálnych čísiel *min* a *max* pri vytvorení dátového typu z generátora. Ak je dátový typ vytvorený z iného dátové typu, konštanty minimálnej a maximálnej hodnoty z neho preberie. Formát zápisu generátora je *finterval(min,max)*.

Dátový typ vytvorený z takéhoto generátora môže byť ďalej špecifikovaný v sémantických obmedzeniach. Proces generovania hodnoty a sémantické obmedzenia pre tento generátor sú rovnaké ako pre generátor typu integer v intervale popísané v podkapitole 3.4.2. Výraz v sémantických obmedzeniach naviac môže obsahovať reálne čísla vo formáte <*celá časť*>.<*desatinná časť*>.

3.4.5 Generátor typu datetime v intervale

Generátor typu datetime v intervale generuje dátum v určitom intervale. Interval je zadaný konštantami dátumu a času *min* a *max* pri vytvorení dátového typu z generátora. Formát zadania dátumu a času je v tvare <*rok*>-<*mesiac*>-<*deň*> <*hodina*>:<*minúta*>, pričom zadanie času je voliteľné. Ak je dátový typ vytvorený z iného dátové typu, konštanty minimálnej a maximálnej hodnoty z neho preberie. Formát zápisu generátora je *dinterval(min,max)*.

Dátový typ vytvorený z takéhoto generátora môže byť ďalej špecifikovaný v sémantických obmedzeniach, ktoré nastavujú nové minimálne a maximálne hranice intervalu generovanej hodnoty. Povolené operátory pre dátový typ s daným generátorom sú < a >. Výraz môže obsahovať jednu hodnotu dátumu a času, alebo celý názov jednoduchého dátového typu s generátorom typu datetime. Počet sémantických obmedzení nie je limitovaný.

Sémantické obmedzenia sa môžu týkať aj konkrétnej časovej zložky, z ktorých sa dátum a čas skladá. V takomto prípade je cieľ obmedzenia špecifikovaný. Formát adresovania konkrétnej zložky času má tvar <*dátový typ*>.<*časová zložka*>. Povolené operátory sú <, >, >=, <=, = a *in*. Samotný výraz sa musí skladať len z celých čísiel, referencií na jednoduchý dátový typ generujúci celé čísla, referencií na konkrétne časové zložky dátového typu a operátorov +, -, /, *.

Proces generovania začne vyhodnotením výrazov sémantických obmedzení intervalu celého dátumu a času ktorých výsledkom sú konkrétne hodnoty. Nasleduje výber najmenšej hodnoty z definovaných maximálnych hodnôt a najväčšej hodnoty z minimálnych hodnôt. Všeobecne musí byť minimálna hodnota menšia alebo rovná maximálnej, inak pri zadaných podmienkach neexistuje hodnota ktorá by ich všetky spĺňala. Pokiaľ boli definované sémantické obmedzenia na jednotlivé zložky dátumu a času, vyhodnotia sa zadané výrazy, čím dostaneme konkrétne hodnoty. Každá zložka času má vlastné maximálne a minimálne hodnoty. Pre rok to je 1 až 9999, mesiac 1 až 12, deň 1 až maximálna hodnota dňa v danom dátume, hodina 0 až 23, minúta a sekunda 0 až 59. Nasleduje výber najmenšej hodnoty z definovaných maximálnych hodnôt a najväčšej hodnoty z minimálnych hodnôt pre každú zložku času.

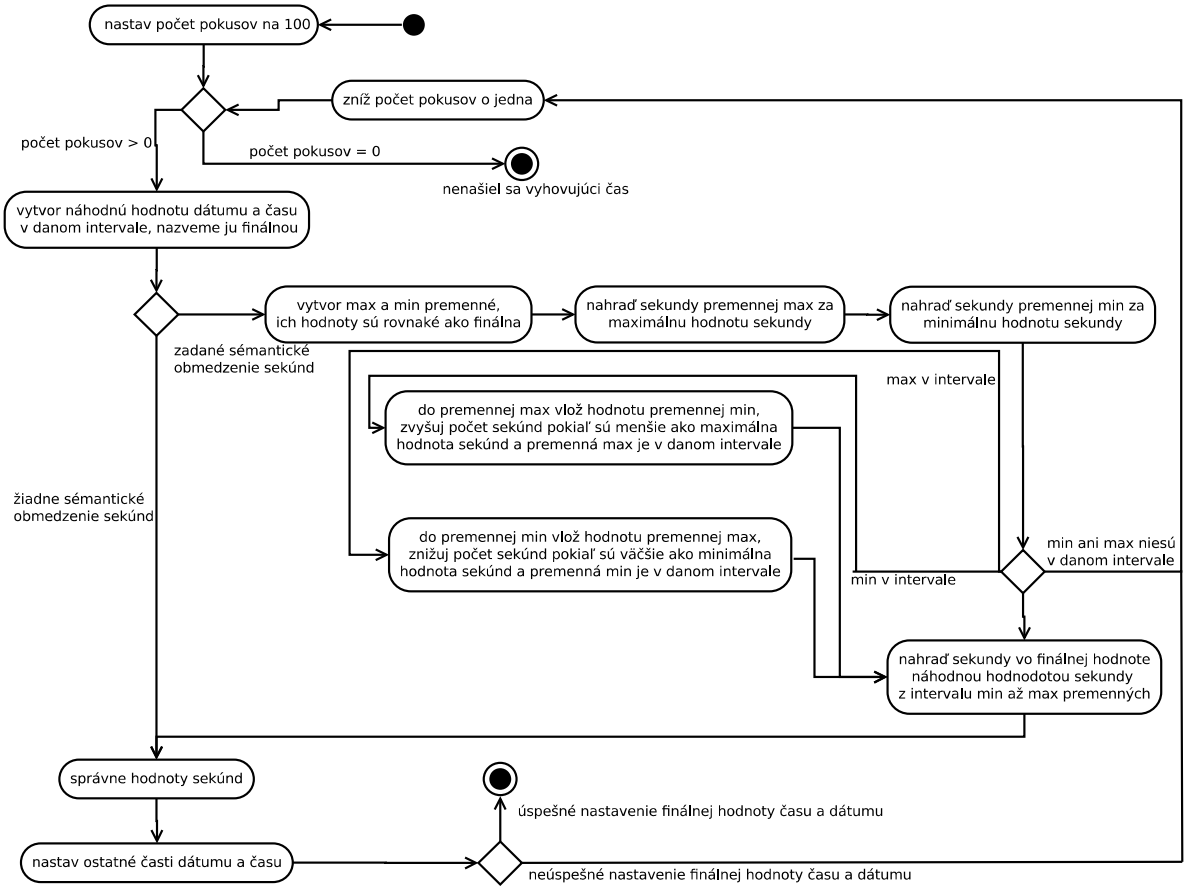
Po získaní maximálnych a minimálnych hodnôt jednotlivých intervalov nasleduje vyhľadanie takého dátumu a času, ktorého zložky patria do definovaných intervalov a zároveň je v stanovenom intervale aj ako celok. Takáto hodnota sa nemusí dať nájsť vždy, preto je počet pokusov o jej nájdenie obmedzený na sto. Ako prvá sa vytvorí náhodná hodnota dátumu a času v intervale obmedzujúcom dátový typ ako celok. Nazveme ju finálna hodnota. Nasleduje prispôsobenie jednotlivých zložiek finálnej hodnoty. Začína sa postupne od hodnoty zložky sekúnd až po roky.

Zložku finálnej hodnoty, ktorú upravujeme nazveme cieľová. Ak cieľová zložka finálnej hodnoty nepatrí do jej intervalu zložky, musí sa zmeniť. Zmena hodnoty cieľovej zložky finálnej hodnoty na hodnotu danú intervalom cieľovej zložky začína vytvorením premenných dátumu a času min a max, ktoré majú rovnakú hodnotu ako je finálna. Následne sa cieľové zložky hodnoty min a max nahradia minimálnou a maximálnou hodnotou intervalu, ktorý ju obmedzuje. Ak žiadna z premenných min a max nepatrí do celkového intervalu dátumu a času, preruší sa proces zmeny zložiek času a počet pokusov nájdenia vhodnej hodnoty sa zníži. Ak do intervalu patria obe, zmení sa hodnota cieľovej zložky finálneho času na náhodnú v intervale zložky. Ak do celkového intervalu patrí len jedna premenná, druhá sa upraví. Pokiaľ do intervalu patrila premenná max, premenná min

nadobudne hodnotu max. Potom sa postupne zmenšuje cieľová zložka min pokiaľ je jej hodnota v celkovom intervale a min je v celkovom intervale. Podobne sa postupuje, ak do intervalu patrila premenná min. Cieľová hodnota max preberie hodnotu min a následne sa zvyšuje pokiaľ patrí do celkového intervalu a do intervalu cieľovej zložky. Výsledkom sú premenné min a max, ktoré obe patria do celkového intervalu a hodnoty ich cieľových zložiek sú tiež v intervale zložky. Posledný krok je úprava cieľovej zložky finálnej hodnoty na náhodnú hodnotu v intervale cieľových zložiek premenných min a max. Pseudokód príkladu vytvorenia dátového typu s obmedzením na mesiac:

cieľový interval: 2000-4-5 00:00 , 2000-10-7 00:00
obmedzenie cieľovej zložky mesiac: 3, 6
náhodná finálna hodnota v celkovom intervalue: 2000-8-2 00:00, mesiac nepatrí do intervalu
premenná min má hodnotu 2000-3-2 00:00
premenná max má hodnotu 2000-6-2 00:00
max patrí do celkového intervalu, min nepatrí, upravíme min
min prevezme hodnotu max(2000-6-2 00:00), pokúsi sa zmenší sa jej hodnotu a skontroluje či patrí do intervalov.
min je 2000-5-2 00:00 , hodnota patrí do cieľového intervalu mesiaca a celkového intervalu
min je 2000-4-2 00:00 , hodnota patrí do cieľového intervalu mesiaca ale nepatrí do celkového intervalu, min sa nastaví na predchádzajúcu hodnotu 2000-5-2 00:00
vyberie sa náhodná hodnota medzi hodnotou mesiaca min a max (5,6) a zmení sa finálna hodnota.

Obrázok 3.1 popisuje postup pri generovaní hodnoty, ktorá patrí do celkového intervalu a intervalu zložky. Konkrétne podrobne ukazuje nastavenia hodnoty sekundy.



Obrázok 3.1: Algoritmus generovania datetime hodnoty

3.4.6 Generátor hodnoty z datasetu

Generátor hodnoty z datasetu generuje náhodné hodnoty definované v datasete. Dataset je vytvorený pri definícii generátora formou zoznamu hodnôt ktorý je následne uložený v generátore. Hodnoty datasetu sú reťazce neobmedzenej dĺžky. Generátor implicitne generuje náhodné hodnoty z datasetu. Výber hodnôt v poradí v ktorom sú v datasete je možné zaistiť prvou hodnotou v datasete. Ak sa táto hodnota rovná reťazcu *ORDERED_DATASET*, hodnoty budú cyklicky generované v poradí v akom boli definované v datasete od začiatku po koniec. Formát vytvorenia dátového typu z generátora hodnoty z datasetu:

<názov dátového typu> :

- *“<hodnota v datasete>“ +*

3.4.7 Generátor formátovacieho reťazca

Generátor formátovacieho reťazca generuje reťazec podľa formátovacieho reťazca jazyka python. Popis fungovania formátovacieho reťazca sa dá nájsť na oficiálnych stránkach jazyka python.[9] Formátovací reťazec je zadaný reťazcom uloženom v položke format. Hodnoty, ktoré budú doň dosadené, sú v položke values. *Format* predstavuje formátovací reťazec jazyka python. Formátovací reťazec obsahuje konštanty znakov a špeciálne znaky *{ }*, ktoré sú nahradené hodnotami z values a prípadne formátované to zadaného tvaru. Dátový typ vytvorený z tohto generátora nemôže byť ďalej špecifikovaný v sémantických obmedzenia. Formát vytvorenia takéhoto generátora:

```
<názov dátového typu>:
    format: “{<formát>} <konštanta>“
    values:
        -  <dátový typ / generátor / konštanta><špecifikácia počtu>
```

Výsledný vygenerovaný reťazec bude mať tvar: *“<konštanty formátovacie reťazca> <hodnota získaná z referovaných hodnôt>“*. Maximálny počet hodnôt nie je obmedzený ale musí byť zhodný s počtom *{}* vo formátovacom reťazci.

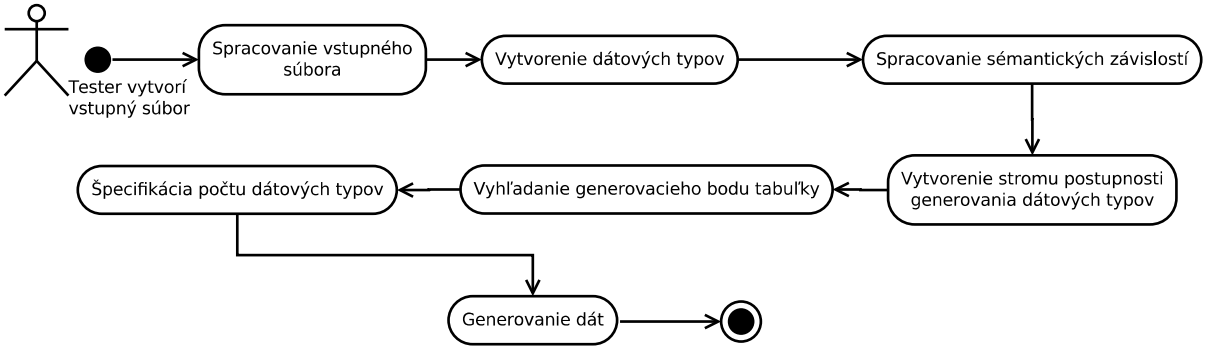
Proces vytvárania dátového typu z generátora formátovacieho reťazca začne uložením formátovacieho reťazca. Následne sa vyhodnotia hodnoty, ktoré budú do neho dosadené. Hodnoty, ktoré sú konštanty alebo generátory vytvoria príslušné generátory a uložia sa do hodnôt generátora.

Hodnota môže predstavovať aj dátový typ. Vykoná sa kontrola, či má adresovaný dátový typ rovnaký koreňový dátový typ ako ten, ktorý bude vytvorený týmto generátorom. Pokiaľ má spomenutý dátový typ rovnaký koreňový dátový typ, uloží sa do hodnôt generátora takzvaný generátor referencie ktorý bude vracať poslednú hodnotu dátového typu, na ktorý sa odkazuje. Vytvorením referencie vznikla závislosť medzi týmito dátovými typmi. Závislosť nového dátového typu na adresovanom pridáme medzi záznamy o všetkých závislostiach. Zároveň sa ešte skontrolujú existujúce závislosti adresovaného dátového typu. Pokiaľ nejaké existujú, vytvoria sa rovnaké aj pre nový dátový typ. Ak má spomenutý dátový typ iný koreňový dátový typ, vytvorí sa kópia generátora adresovaného dátového typu a je pridaná do hodnôt generátora formátovacieho reťazca. Pred pridaním sa skontroluje, či má adresovaný dátový typ závislosti. Ak má, nesmie byť skopírovaný, pretože by došlo k nekonzistentnosti referencií a nastane chybový stav.

Proces generovania novej hodnoty postupne prechádza jednotlivé uložené hodnoty. Každá hodnota obsahuje generátor, ktorý je zavolaný a vráti vygenerovanú hodnotu, ktorá sa uloží pre potreby formátovacieho reťazca. Konečná vygenerovaná hodnota je vytvorená formátovacím reťazcom jazyka python.

3.5 Fázy behu programu

Fáza behu predstavuje ucelenú časť programu. Sekvenciou vykonávania jednotlivých fáz sa vykoná celý beh programu. Diagram na obrázku 3.2 zobrazuje sekvenciu jednotlivých fáz.



Obrázok 3.2: Fázy behu programu

3.5.1 Spracovanie vstupného súboru

Zdrojový súbor v jazyku YAML spracujeme do dátových typov jazyka python, konkrétne do typov slovník, list a reťazec. Jazyk vytvorený pre tento nástroj vymedzuje, ktoré hodnoty budú spomínané dátové typy nadobúdať. Konkrétne hodnoty vyzerajú nasledovne. Dátový typ slovník môžeme prirovnať k forme asociatívneho poľa. Pomocou kľúča, v tomto prípade reťazca, pristupujeme k jeho hodnotám, ktoré sú jednými zo spomenutých typov. Dátový typ list uchováva zoradenú sekvenciu reťazcov. Reťazec predstavuje sekvenciu znakov. Príklad vstupného súbora je v príklade 3.3.

```
include:
    - "names"
    - "towns"
types:
    prevádzka:
        - "ORDERED_DATASET"
        - "BB Tel a.s"
        - "Bolit s.r.o"
        - "Ditel s.r.o "
    predajňa:
        majiteľ: prevádzka
        mesto:
            format: "{}, {}"
            values:
                - town
                - predajňa.majiteľ
    práca:
        id_pracovného_úkonu: seq(0,1000000,1)
        začiatok: dinterval(2015-1-1,2015-12-31 23:59)
        koniec: dinterval(2015-1-1,2015-12-31 23:59)

    zamestnanec:
        id: seq(0,100,1)
        meno: full_name
        pracovný_deň: "[práca]+"
```

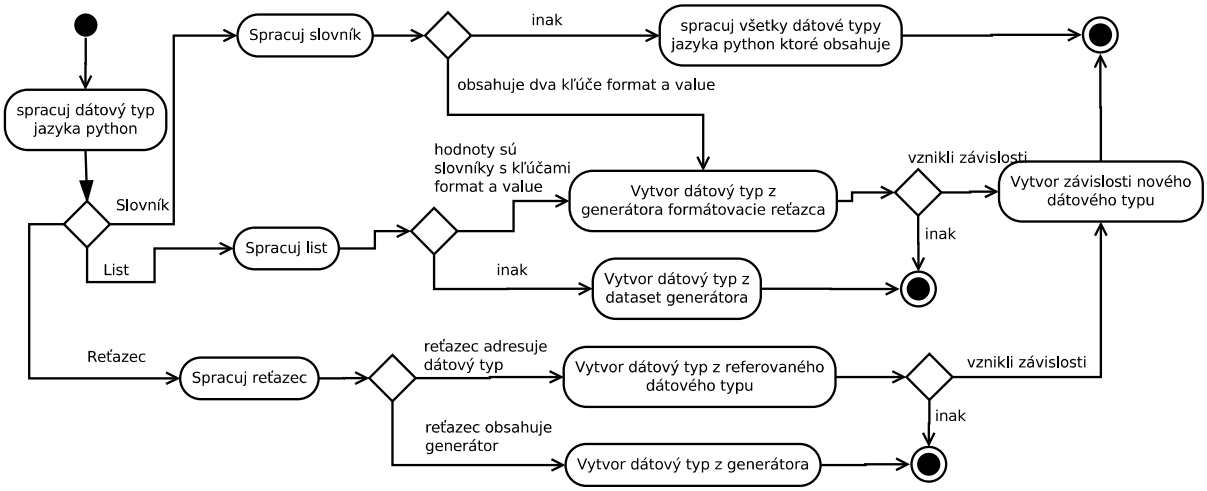
```
schema:
    table_predajňa:
        vlastník: predajňa.majiteľ
        adresa_predajne: predajňa.mesto

    table_zamestnanci:
        id: zamestnanec.id
        meno: zamestnanec.meno
    table_pracovné_úkony:
        id_zamestnanec: zamestnanec.id
        id_úkon: zamestnanec.pracovný_deň.id_pracovného_úkonu
        začiatok: zamestnanec.pracovný_deň.začiatok
        koniec: zamestnanec.pracovný_deň.koniec
        miesto: FROM table_predajňa.vlastník

constraints:
    - zamestnanec.pracovný_deň.koniec.hour in (16,20)
    - zamestnanec.pracovný_deň.začiatok.hour =
zamestnanec.pracovný_deň.koniec.hour - 8
    - zamestnanec.pracovný_deň.začiatok.year =
zamestnanec.pracovný_deň.koniec.year
    - zamestnanec.pracovný_deň.začiatok.month =
zamestnanec.pracovný_deň.koniec.month
    - zamestnanec.pracovný_deň.začiatok.day =
zamestnanec.pracovný_deň.koniec.day

generate:
    - predajňa = 3
    - zamestnanec = 8
    - zamestnanec.pracovný_deň <= 30
```

Príklad 3.3: Príklad vstupného súboru

3.5.2 Vytvorenie dátových typov

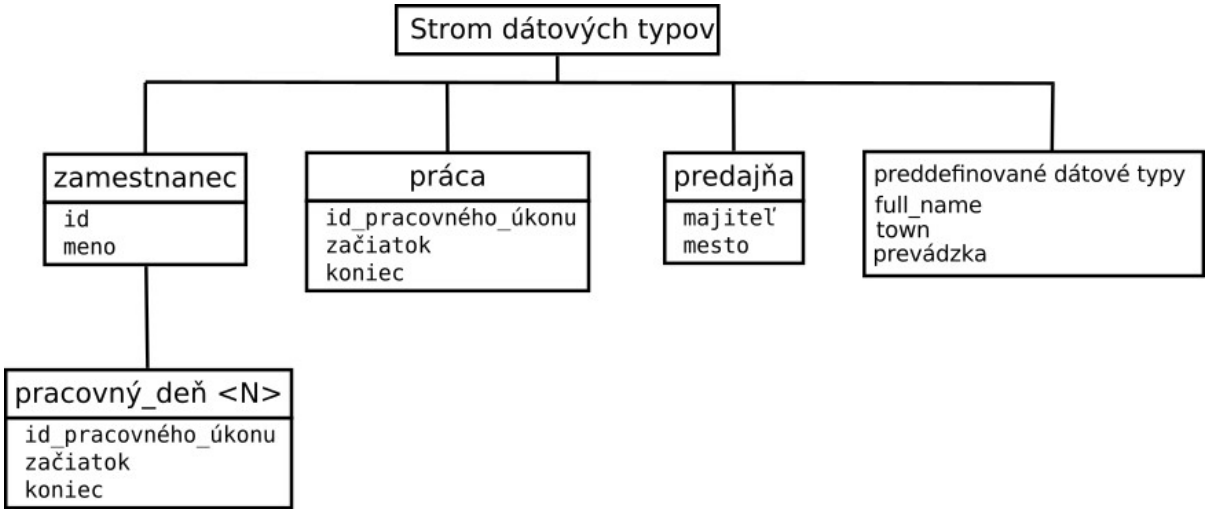
Vytvorenie dátových typov je fáza, v ktorej sa z vytvorených dátových typov jazyka python, konkrétne časti *schema* a *include* postupne vytvoria dátové typy generátora. Ako prvé sa vytvoria preddefinované dátové typy. Nasledujú dátové typy špecifikované v časti *include* a nakoniec v časti *types*.

Proces spracovania vstupných dát sa odlišuje pre každý dátový typ jazyka python. Pri vytváraní nových dátových typov môžu vznikať závislosti. Ak sa vytvára dátový typ z generátora formátovacieho reťazca, ktorého hodnoty sa odkazujú na dátové typy v aktuálnom zloženom dátovom type, generátor použije referenciu namiesto vytvorenia kópie generátora adresovaného dátového typu a vytvorí si naň závislosť. Druhý prípad môže nastať pri vytvorení dátového typu z iného dátového typu. Musia sa skontrolovať existujúce závislosti adresovaného dátového typu a prípadne sa vytvoria rovnaké aj pre nový dátový typ. Princíp spracovania jednotlivých dátových typov jazyka python popisuje diagram na obrázku 3.4.



Obrázok 3.4: Princíp spracovania dátových typov

Vytvorené dátové typy generátora sú uložené v objekte, ktorý nazveme strom dátových typov. Názov bol zvolený na základe charakteristiky zložených dátových typov, ktoré obsahujú ďalšie vnorené typy. Zložený dátový typ, ktorý sa nenachádza v inom dátovom type a je možné ho adresovať priamo, nazveme koreňovým. Obrázok 3.5 obsahuje diagram, ktorý popisuje definované dátové typy vstupného súbora v príklade 3.3. Diagram znázorňuje zložené dátové typy, ktoré v sebe obsahujú jednoduché a zložené dátové typy. Zložené sú zobrazené samostatne a k dátovému typu, v ktorom sa nachádzajú, sú pripojené čiarou . Dátové typy, ktoré majú špecifikovaný počet iný ako jedna, obsahujú značku <*N*>.



Obrázok 3.5: Príklad stromu dátových typov

3.5.3 Spracovanie sémantických závislostí

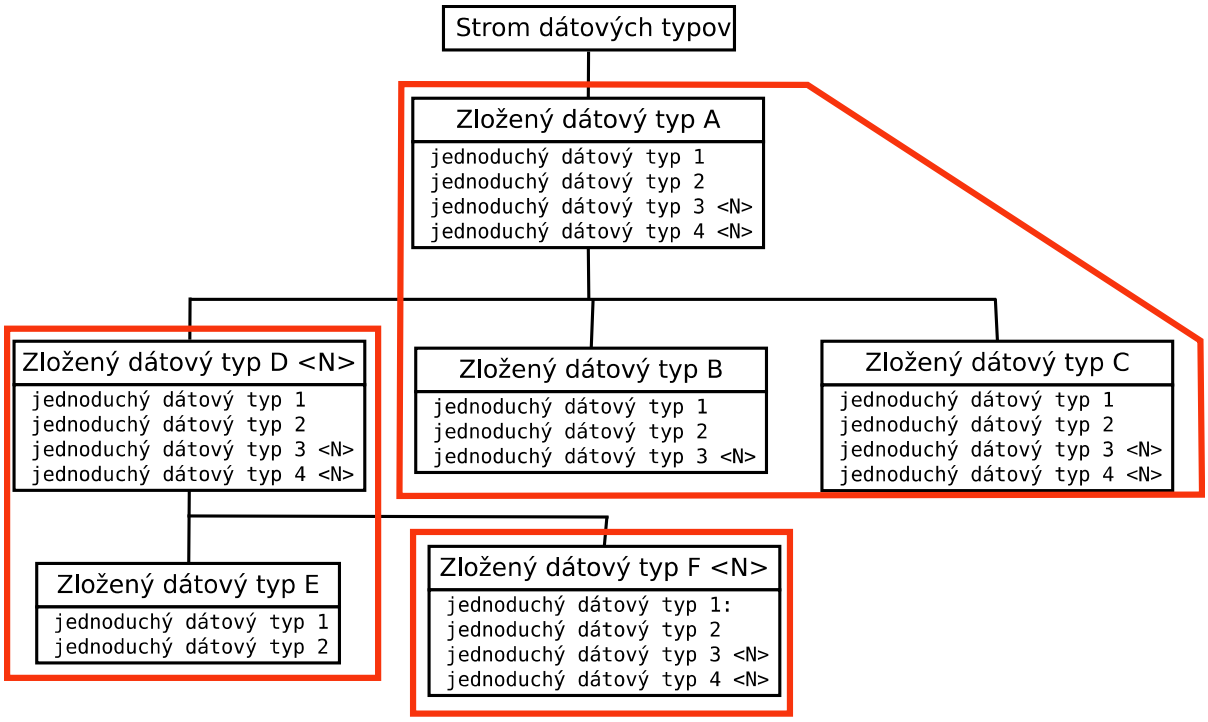
Charakteristika sémantických obmedzení inak nazvaných závislostí, je popísaná v podkapitole 3.2.3. Spracovanie obmedzenia závisí od generátora dátového typu ktorý obmedzuje. Jednotlivé obmedzenia generátorov sú popísané v podkapitole 3.4. Najprv je identifikovaný obmedzovaný jednoduchý dátový typ. Po určení typu jeho generátora sa vyhodnotí použité znamienko. Nasleduje spracovanie výrazu definujúce jeho závislosť. Výraz môže byť konštantný, pokiaľ neobsahuje žiadnu referenciu na iný dátový typ. Výraz, ktorý obsahuje názvy iných dátových typov, čím ich adresuje,

nazveme zloženým. Adresovať môžeme len jednoduché dátové typy nachádzajúce sa v rovnakom koreňovom dátovom type. Zložené výrazy spracujeme do formy, ktorá umožňuje vyhodnotenie formátovacím reťazcom jazyka python. Pri spracovaní zloženého výrazu nahradíme všetky názvy špeciálnym reťazcom *{}*, čím vznikne formátovací reťazec. Ďalej vytvoríme pole hodnôt, ktoré obsahuje referencie na generátory adresovaných dátových typov. Pri vytvorení zloženého výrazu vznikne závislosť obmedzeného dátového typu na adresovaných. Vzniknuté obmedzenie sa predá generátoru obmedzeného dátového typu.

3.5.4 Vytvorenie stromu postupnosti generovania dátových typov

Jadrom celého nástroja je vytvorenie poradia generovania všetkých dátových typov, ktoré rešpektuje definované závislosti. Výsledné poradie je uložené v objekte, ktorý nazveme strom postupnosti.

Uložené informácie o jednotlivých dátových typoch sú rozdelené do skupín, ktoré sa generujú oddelene. Každá skupina pozostáva z hlavného zloženého dátového typu, zložených dátových typov s určeným počtom hodnôt jedna a všetkých jednoduchých dátových typov, ktoré sa v nich nachádzajú. Skupina rozdeľuje dátové typy do troch častí, konkrétne sú to *one*, *multiple* a *datatype*. Časť *one* uchováva všetky jednoduché dátové typy s počtom hodnôt jedna. Časť *multiple* uchováva všetky jednoduché dátové typy s iným počtom hodnôt. Časť *datatype* uchováva všetky skupiny vytvorené priamo zo zložených dátových typov s iným počtom hodnôt. Obrázok 3.6 zobrazuje príklad vytvorenia hlavných skupín postupnosti dátových typov, skupiny sú vyznačené červenou čiarou.



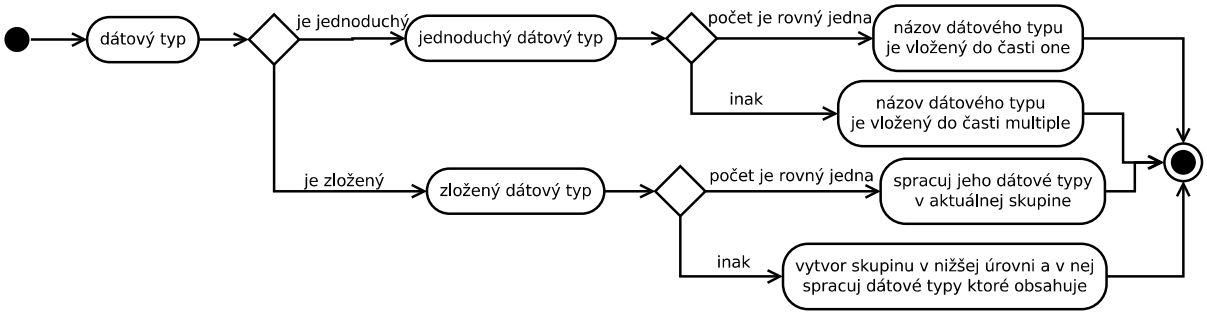
Obrázok 3.6: Príklad vytvorenia oddelených skupín dátových typov

Dáta skupiny sú uložené v jej častiach. Časti one a multiple sú zoznamy celých názvov dátových typov. Časť datatype je mapa uchovávajúca skupiny, ktoré sa nachádzajú priamo v skupine, ktorú vytvárame. Formát uloženia dátových typov z príkladu 3.6 apísaný v dátových typoch jazyka python je zobrazený v príklade 3.7. Názov dátových typov je skrátený.

```
{'multiple': ['dtA.3','dtA.4','dtA.dtB.3','dtA.dtC.3','dtA.dtC.4'],
 'one': ['dtA.1','dtA.2','dtA.dtB.1','dtA.dtB.2','dtA.dtC.1','dtA.dtC.2'],
 'datatype': {
   'dtA.dtD': {
     'multiple': ['dtA.dtD.3','dtA.dtD.4'],
     'one': ['dtA.dtD.1','dtA.dtD.2','dtA.dtD.dtE.1','dtA.dtD.dtE.2'],
     'datatype': {
       'dtA.dtD.dtF': {
         'multiple': ['dtA.dtD.dtF.3','dtA.dtD.dtF.4'],
         'one': ['dtA.dtD.dtF.1','dtA.dtD.dtF.2'],
         'datatype': {}
       }
     }
   }
 }
}
```

Príklad 3.7: Príklad formátu uloženia dát postupnosti

Proces vytvorenia stromu postupnosti generovania začína výberom všetkých zložených dátových typov. Pre každý z nich vytvorí prázdnu skupinu. Následne cyklicky spracováva všetky dátové typy ktoré obsahuje. Obrázok 3.8 ukazuje spôsoby spracovania konkrétneho dátového typu.



Obrázok 3.8: Princíp spracovania dátových typov pri tvorbe skupiny

Po vytvorení stromu postupnosti sa musia dátové typy v jednotlivých skupinách usporiadať. Usporiadanie skupiny sa týka častí *one* a *multiple*. Obe časti sa usporiadajú nezávisle na sebe. V definovaných obmedzeniach sa vyhľadajú tie, ktoré sú medzi dvoma dátovými typmi v časti, ktorú chceme usporiadať. Výsledkom je poradie dátových typov, ktoré dodržuje obmedzenia.

3.5.5 Vytvorenie generovacieho bodu tabuľky

Generovací bod tabuľky predstavuje miesto v strome postupnosti generovania dátových typov. Konkrétne označuje bod, v ktorom z posledných vygenerovaných hodnôt dátových typov adresovaných v stĺpcoch tabuľky vytvorí príkaz jazyka SQL pre vloženie dát do databázy. Samotné tabuľky sa nachádzajú v stupnom súbore v sekcii *schema*. Každá tabuľka má zadané meno a stĺpce, ktoré obsahuje. Stĺpec nadobúda hodnotu dátového typu, ktorý adresuje.

Aby sme mohli vytvoriť generovací bod, musíme vyhľadať najhlbšie adresovanú skupinu v strome postupnosti. Na začiatku procesu hľadania nastavíme generovací bod na skupinu v strome postupnosti, ktorá je rovnaká ako koreňový dátový typ prvého adresovaného dátového typu. Skupinu, na ktorú ukazuje generovací bod, nazveme generovacou. Potom sa každý adresovaný dátový typ spracuje nasledovným spôsobom. Vyhľadáme skupinu v strome postupnosti, do ktorej patrí. Nazveme ju adresovaná skupina. Porovnajú sa cesty k adresovanej a generovacej skupine. Konkrétne ide o

jednotlivé skupiny, cez ktoré pristupujeme k cieľovej skupine. Pokiaľ sa prístupové cesty rovnajú alebo je cesta adresovanej skupiny kratšia, ale celá sa zhoduje so začiatkom generovacej cesty, pokračujeme na ďalší stĺpec. Pokiaľ je cesta k adresovanej skupine dlhšia, ale jej začiatok sa zhoduje s celou cestou generovacej skupiny, zmeníme generovaciu skupinu na skupinu adresovanú dátovým typom. Ak nenastala žiadna z popísaných možností dôjde k chybe adresovania.

Po vykonaní tohto algoritmu získame generovací bod tabuľky. Musíme ešte zistiť, či sa adresovali dátové typy s hodnotou inou ako jedna. Ak áno, musia sa všetky nachádzať v rovnakej skupine a táto skupina musí byť generovacou, inak došlo k chybnému adresovaniu.

Do skupiny adresovanej generovacím bodom pridáme názov tabuľky a informáciu o adresovaných dátových typoch. Ak sú všetky dátové typy s jednou hodnotou, tabuľku uložíme ako typ *one*. Inak je uložená ako typ *multiple*.

Stĺpec tabuľky nemusí vždy obsahovať len hodnotu dátového typu. Nástroj ponúka možnosť referencie na náhodný vytvorený záznam inej tabuľky. Adresovanie má tvar *FROM* <tabuľka.stĺpec>. Takto adresovaná tabuľka sa musí nachádzať v inom koreňovom dátovom type a jej hodnoty budú generované pred hodnotami tabuľky využívajúcej referenciu.

3.5.6 Špecifikácia počtu dátových typov

Nástroj umožňuje dodatočne špecifikovať hodnoty koreňových dátových typov a dátových typov s počtom hodnôt iným ako jedna. Využívame ho hlavne pri koreňových dátových typoch, ktorých jednoduché dátové typy sú priamo adresované stĺpcami tabuľky. Ďalšie využitie nachádza pri určení počtu dátových typov vytvorených zo šablóny. Meniť priamo jej hodnoty by ovplyvnilo ostatné dátové typy, ktoré z nej tiež vychádzajú.

3.5.7 Generovanie dát

Generovanie dát predstavuje finálnu fázu programu. Vytvára hodnoty dátových typov a nové záznamy zadaných tabuliek. Na začiatku procesu určíme poradie generovania jednotlivých koreňových skupín stromu postupnosti generovania dát. Poradie určujú referencie hodnôt jednej tabuľky na druhú. Skupiny tabuliek na ktoré sú vytvorené referencie, musíme vygenerovať pred skupinou tabuľky s referenciou.

Proces generovania začína výberom koreňovej skupiny. Nájdeme zložený dátový typ, z ktorého bola skupina v strome postupnosti generovania dátových typov vytvorená. Vytvoríme náhodné číslo v intervale maximálneho a minimálneho počtu hodnôt tohto dátového typu. Číslo predstavuje počet cyklov procesu vytvárania hodnôt dátových typov v danej skupine.

Proces vytvárania hodnôt dátových typov v skupine postupnosti generovania začína vygenerovaním hodnôt všetkých dátových typov v časti *one*. Keďže počet hodnôt dátových typov v tejto časti je jedna, môžeme vytvoriť nové záznamy vo všetkých tabuľkách typu *one*. Ďalšie generujeme dátové typy v časti *multiple*. Vytvoríme náhodné číslo v intervale maximálneho a minimálneho počtu hodnôt každého dátového typu a uložíme ho. Nasleduje proces generovania hodnôt, ktorý opakujeme pokiaľ sme nevygenerovali všetky hodnoty jednotlivých dátových typov. Počet nových záznamov v tabuľkách časti *multiple* určuje najmenší počet hodnôt z dátových typov ktoré sú adresované tabuľkou, čím predchádzame nekonzistentným dátam v prípade vzájomných závislostí týchto dátových typov. Pre skupiny nachádzajúce sa v časti *datatype* je postup generovania rovnaký ako pre koreňové skupiny.

Kapitola 4

4 Implementácia a vyhodnotenie nástroja

Táto kapitola uvádza implementačné detaily navrhnutého nástroja pre generovanie testovacích dát a popisuje testovanie funkciionality výslednej aplikácie.

4.1 Implementačné detaily nástroja

Nástroj je implementovaný v jazyku Python 3. Implementované rozhranie má formu príkazového riadku. Každý generátor hodnôt dátového typu je implementovaný vo vlastnej triede. Každá trieda obsahuje metódu `generate()` ktorá vygeneruje novú hodnotu a tá sa uloží do atribútu `last`. Prístup k danej hodnote je prostredníctvom metódy `last()`.

4.1.1 Použité funkcie a knižnice

Pre spracovanie vstupného súbora je využívaná knižnica `pyyaml`. Štandartne knižnica spracuje vstup do dátových typov jazyka python, konkrétne do typov `list`, `dictionary` a `string`. Pre účely nástroja je dictionary nevyhovujúci dátový typ, pretože neuchováva poradie, v ktorom boli jeho položky vytvorené. Preto bola knižnica zmenená tak, aby vytvárala dátový typ `orderedDictionary`. Tento dátový typ uchováva poradie vstupného súbora, čo umožňuje pristupovať k jeho položkám v definovanom poradí ako boli vo vstupnom súbore.

Pri vyhodnotení výrazov v sémantických obmedzeniach sa využíva funkcia `eval`. Podľa typu obmedzovaného generátora sa jej výsledok zmení na typ `int`, `bool` alebo `float`. Ak počas tohto procesu došlo k chybe, daný výraz bol chybne zadaný.

Priame vkladanie dát do databázy zabezpečuje knižnica `PyMySQL`. Spojenie s databázou vyžaduje meno a heslo užívateľa, adresu a názov databázy. Pri každom vytvorení nového záznamu kontrolujeme jeho vloženie do databázy. Ak vznikla chyba pri vložení dát, nebudú do databázy vložené žiadne dáta. Ak všetky príkazy vloženia prebehli v poriadku, dáta sa uložia.

4.1.2 Adresárová štruktúra

Zdrojové súbory nástroja sú uložené v niekoľkých častiach:
`Src` – Koreňový adresár, ktorý obsahuje všetky časti zdrojového kódu
`src/parse_input.py` – zdrojový kód časti spracovávajúcej vstupný súbor a argumenty spustenia
`src/generator.py` – hlavný zdrojový súbor, ktorý sa spúšťa
`src/sql_connector.py` – zdrojový súbor funkcií nad databázou
`src/generator_classes.py` – zdrojový súbor obsahuje všetky triedy generátorov
`src/generating_algorithm.py` – obsahuje všetky fázy behu programu
`src/datasets` – adresár ktorý obsahuje súbory s vlastnými dátovými typmi
`src/examples` – adresár, obsahuje vzorové vtupné súbory

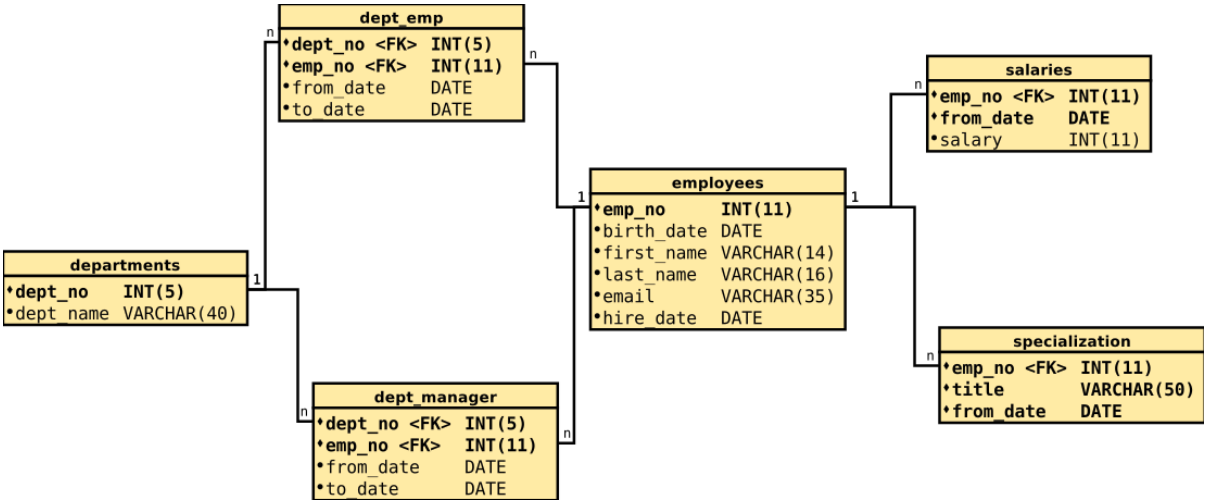
4.2 Testovanie funkcionality nástroja

Pre účel testovania funkčnosti generátora je vhodné vybrať reálne používanú databázu, prípadne databázu reálneho charakteru. Databáza ideálne obsahuje rôzne druhy štrukturálnych a sémantických

väzieb. Cieľom testu je overenie čo najväčšej funkcionality implementovaného nástroja. Pre účely testovania nástroja boli vytvorené dve testovacie databázy. Nasledujúce podkapitoly popisujú vstupné súbory pre dané databázy, generované výstupy sú uvedené v prílohe.

4.2.1 Testovacia databáza A

Testovacia databáza predstavuje databázu v systéme uchovávajúcom informácie o zamestnancoch a oddeleniach spoločnosti. Hlavný predmet testu je overenie dodržania štrukturálnych závislostí jednotlivých tabuliek ale zároveň skúma aj sémantické. Schéma databázy obsahuje sedem tabuliek. Prvý krok analýzy schémy je rozdelenie tabuliek do samostatných častí, ktoré obsahujú len vzťahy 1:1 a 1:N. Vytvoria sa tak časti, ktoré sa budú samostatne generovať. Prvá časť sa skladá z tabuliek *employees*, *salaries* a *titles*. Ďalšia časť obsahuje tabuľky *departments*. Zostali tabuľky dept_emp a dept_manager. Tabuľku dept_emp pripojíme do časti, v ktorej je tabuľka employees. Pre tabuľku dept_manager vytvoríme samostatnú časť. Obrázok 4.1 zobrazuje schému databázy.



Obrázok 4.1: Diagram schémy testovanej databázy

Použité sémantické závislosti sa vzťahujú na obmedzenia dĺžky názvu oddelenia a dňa dátumu výplaty. Dátumy výplaty, nadobudnutia špecializácie a začiatku práce v oddelení sú neskôr ako dátum zamestnania zamestnanca. Ďalej bola využitá šablóna names pre účely krstného mena a priezviska. Príloha A obsahuje výstup nástroja. Príklad 4.2 obsahuje vstupný súbor danej databázy.

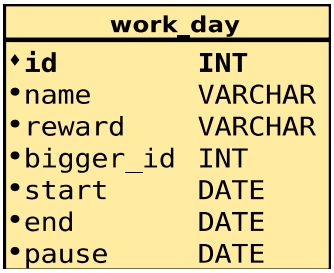
```
Include: ["names"]
types:
    job_specialization: ["ORDERED_DATASET","Application
programmer","System programmer","Web programmer","Tester"]
    departments:
        dept_no: seq(0,100000,1)
        dept_name: string
    dept_emp:
        from_date: dinterval(2000-1-1,2016-12-31)
        to_date: dinterval(2000-1-1,2016-12-31)
    salary:
        from_date: dinterval(2000-1-1,2016-12-31)
        salary: interval(1000,4000)
    specialization:
        title: job_specialization
        from_date: dinterval(2000-1-1,2016-12-31)
```

```
    employee:
        emp_no: seq(0,100000,1)
        birth_date: dinterval(1950-1-1,1997-12-31)
        first_name: first_name
        last_name: last_name
        email:
            format: "{}.{}{}@company.com"
            values:
                - employee.first_name
                - employee.last_name
                - interval(1,99)
        hire_date: dinterval(2000-1-1,2016-12-31)
        spec: "[specialization]{1,2}"
        salary: "[salary]{0,10}"
        dept_emp: "[dept_emp]{1,3}"
    dept_manager:
        active: bool
schema:
    employees:
        emp_no: employee.emp_no
        birth_date: employee.birth_date
        first_name: employee.first_name
        last_name: employee.last_name
        email: employee.email
        hire_date: employee.hire_date
    salaries:
        emp_no: employee.emp_no
        from_date: employee.salary.from_date
        salary: employee.salary.salary
    specialization:
        emp_no: employee.emp_no
        title:  employee.spec.title
        from_date: employee.spec.from_date
    departments:
        dept_no: departments.dept_no
        dept_name: departments.dept_name
    dept_emp:
        emp_no: employee.emp_no
        dept_no: FROM departments.dept_no
        from_date: employee.dept_emp.from_date
        to_date: employee.dept_emp.to_date
    dept_manager:
        emp_no: FROM dept_emp.emp_no
        dept_no: FROM dept_emp.dept_no
        from_date: FROM dept_emp.from_date
        to_date: FROM dept_emp.to_date
        active: dept_manager.active
constraints:
    - departments.dept_name in (4,10)
    - employee.salary.from_date.day = 10
    - employee.salary.from_date > employee.hire_date
    - employee.spec.from_date > employee.hire_date
    - employee.dept_emp.from_date > employee.hire_date
    - employee.dept_emp.to_date > employee.dept_emp.from_date
generate: ["departments = 2", "employee = 10", "dept_manager = 4"]
```

Príklad 4.2: Príklad podoby vstupného súboru pre testovaciu databázu

4.2.2 Testovacia databáza B

Testovacia databáza predstavuje jednoduchú databázu, ktorá sa skladá len z jednej tabuľky. Hlavný cieľ testu je overenie funkcionality všetkých generátorov, dodržania sémantických závislostí jednotlivých dátových typov a ich interné usporiadanie v ktorom sa hodnoty generujú. Obrázok 4.3 obsahuje diagram schémy databázy.



Obrázok 4.3: Diagram schémy testovanej databázy

Obmedzenia atribútov tabuľky:

- id - primárny kľúč, zväčšuje sa o jedna v každom vytvorenom zázname databázy
- name – meno pracovníka vytvorené z preddefinovaného datasetu
- reward – reťazec obsahuje kód platby pozostávajúci z troch až šiestich čísiel šestnástkovej sústavy, sumy platby v hodnote sto až dvesto zaokrúhlené na dve desatinné miesta a znak meny euro alebo dolár
- bigger_id – náhodné číslo v intervale od hodnoty id po jej dvojnásobok
- start – náhodný dátum a čas v roku 2016, hodina je v intervale šesť až desať hodín
- end – dátum a čas rovnaký ako v atribúte start, odlišná hodnota hodiny, ktorá je minimálne o štyri až dvanásť hodín väčšia
- pause - dátum rovnaký ako dátum v atribúte start, hodnota hodiny je v intervale danom hodinami start a end

Vygenerované dáta sú v prílohe Príloha B . Príklad 4.4 obsahuje vstupný súbor so všetkými zadanými obmedzeniami.

```
include: ["names"]
types:
    currency: ["€","$"]
    work_day:
        id: seq(0,100000,1)
        name: full_name
        reward:
            format: "{}: {:.2f},- {}"
            values:
                - "[xdigit]{3,6}"
                - finterval(100,200)
                - currency
        bigger_id: int
        start: dinterval(2016-1-1,2016-12-31)
        end: datetime
        pause: datetime
schema:
    work_day:
        id: work_day.id
        name: work_day.name
        reward: work_day.reward
        bigger_id: work_day.bigger_id
        start: work_day.start
        end: work_day.end
        pause: work_day.pause
```

```
constraints:
    - work_day.bigger_id in (work_day.id, work_day.id * 2)
    - work_day.start.hour in (6,10)
    - work_day.pause.hour in (work_day.start.hour + 1, work_day.end.hour -
1)
    - work_day.end.hour in (work_day.start.hour + 4, work_day.start.hour +
12)
    - work_day.pause.year = work_day.start.year
    - work_day.pause.month = work_day.start.month
    - work_day.pause.day = work_day.start.day
    - work_day.pause.minute = work_day.start.minute
    - work_day.end.year = work_day.start.year
    - work_day.end.month = work_day.start.month
    - work_day.end.day = work_day.start.day
    - work_day.end.minute = work_day.start.minute
generate:
    - work_day = 30
```

Príklad 4.4: Príklad podoby vstupného súboru pre testovaciu databázu

5 Záver

Cieľom bakalárskej práce bolo navrhnutie a implementácia nástroja, ktorý dokáže vytvoriť náhodné dáta databázy. Hlavným cieľom nástroja je vytvorenie dát dodržiavajúcich vnútorné väzby v databáze. Dôležitou súčasťou je podpora vytvárania datasetov, ktoré obsahujú reálne hodnoty. S využitím datasetov a sémantických závislostí jednotlivých hodnôt tabuľky sa vytvorené dáta databázy výrazne podobajú na reálne dáta, ktoré by databáza obsahovala v reálnom nasadení systému s danou databázou.

Výsledný typ nástroja má stanovené požiadavky, ktoré sa od obvyklých nástrojov rovnakej oblasti líšia pohľadom na generovacie dáta. Ako hlavné požiadavky boli zvolené zachovanie štruktúry databázy a významu jej hodnôt. Na základe daných požiadaviek bol navrhnutý jazyk schopný zapísať všetky nároky na výsledné dáta. Vytvorený jazyk dodržuje formát jazyka YAML, čo uľahčuje spracovanie vstupného súboru. Potom sa zvolili dátové typy, ktoré nástroj podporuje. Ku každému z nich existuje minimálne jeden spôsob generovania jeho hodnôt. Najdôležitejšou časťou návrhu je navrhnutie architektúry a systému práce nástroja tak, aby rešpektoval všetky definované závislosti. Docielilo sa to rozdelením procesu generovania na fázy. Každá fáza zaisťuje časť celkovej funkcionality nástroja.

Nástroj je implementovaný ako aplikácia pre príkazový riadok. Pre implementáciu bol použitý jazyk Python 3. Nástroj bol testovaný na reálnej databáze a na databáze pokrývajúcej všetky typy generátorov hodnôt dátových typov.

Budúca práca a rozšírenia Možnosti rozšírenia nástroja sú veľké. Základné rozšírenie použiteľnosti nástroja predstavujú nové preddefinované generátory. Jedná sa najmä o datasety, ale môžu to byť aj iné dátové typy hodnôt, ktoré je možné vytvoriť z podporovaných generátorov. Ďalšie rozšírenie funkcionality predstavuje vytvorenie nových generátorov. Rovnako priestor pre rozšírenie nachádzame pri adresovaní vytvorených záznamov tabuliek, kde by okrem implementovaného náhodného výberu mohol byť výber špecifickejší. Osobitnou časťou rozšírenia nástroja je rozvoj automatického vytvorenia vstupného súboru.

Literatúra

[1] Ben Keen: generatedata.com. 2015 [cit. 2016-04-25].
Dostupné z: http://www.generatedata.com/

[2] Mark Brocato: mockaroo.com. 2016 [cit. 2016-04-25].
Dostupné z: http://www.mockaroo.com/

[3] Menno van Slooten: mockJSON. 2010 [cit. 2016-04-25].
Dostupné z: http://experiments.mennovanslooten.nl/2010/mockjson/tryit.html

[4] dbForge Data Generator for SQL Server, 2016 [cit. 2016-04-25]
Dostupné z: https://www.devart.com/dbforge/sql/data-generator/features.html

[5] Minářová, Alice. Automatická tvorba obsahu databáze sql pro podporu testování. Brno, 2014. Bakalářská práce, Vysoké učení technické v Brňe, Fakulta informačních technológí. Vedoucí práce Aleš Smrčka.
Dostupné z: https://www.fit.vutbr.cz/study/DP/BP.php.cs?id=15795&file=t

[6] Emmi, Michael, Rupak Majumdar a Koushik Sen. Dynamic test input generation for database applications. In: *ISSTA '07: Proceedings of the 2007 international symposium of Software testing and analysis*. Londýn: ACM, 2007, s. 151-162, ISBN 978-1-59593-734-6
Dostupné z: http://dl.acm.org/citation.cfm?doid=1273463.1273484

[7] Python v3.5.1 documentation. [online] 2016 [cit. 2016-05-13].
Dostupné z: https://docs.python.org/3/

[8] YAML Ain’t Markup Language Version 1.2. [online] 2009 [cit. 2016-05-13].
Dostupné z: http://www.yaml.org/spec/1.2/spec.html

[9] Python format strings syntax. [online] 2016 [cit. 2016-05-13].
Dostupné z: https://docs.python.org/3/library/string.html#formatstrings

Príloha A

Výstupný súbor testovacej databázy 1

Testovacia databáza definovaná v podkapitole 4.2.1 bola zaplnená nasledujúcimi dátami. Predmetom skúmania týchto dát je poradie, v ktorom boli jednotlivé operácie *insert* vykonané.

```
INSERT INTO departments (dept_no, dept_name)
VALUES ('0', 'VFTz')
INSERT INTO departments (dept_no, dept_name)
VALUES ('1', 'spz2YFInh')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('0', '1976-04-10 13:01:08:512245', 'ROSARIO', 'MEADOWS',
'ROSARIO.MEADOWS84@company.com', '2013-03-13 19:15:59:538572')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('0', 'Application programmer', '2016-08-06 12:31:16:583744')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2015-09-10 06:56:07:436308', '3480')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2013-08-10 20:07:27:120830', '3606')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2014-10-10 06:18:44:910606', '1426')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2013-11-10 07:49:25:156640', '1132')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2014-01-10 18:26:52:870123', '2315')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2015-01-10 16:21:46:295263', '1175')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2014-11-10 13:17:37:933366', '3274')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2014-03-10 01:43:40:419359', '1015')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('0', '2015-09-10 17:01:18:338729', '2972')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('0', 1', '2016-01-11 11:51:05:655017', '2016-11-07
23:08:34:846343')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('1', '1973-05-06 10:02:24:966428', 'GIDGET', 'MCKINNEY',
'GIDGET.MCKINNEY82@company.com', '2009-08-14 09:48:36:922952')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('1', 'System programmer', '2014-01-17 14:37:39:378539')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('1', '2015-02-10 15:18:42:888471', '3506')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('1', '2013-12-10 18:57:55:316376', '2804')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('1', '2015-01-10 23:48:14:877738', '1722')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('1', '2014-12-10 21:08:42:842272', '1505')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('1', '2015-10-10 20:56:43:988255', '3623')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('1', '2015-05-10 16:19:37:269385', '1267')
INSERT INTO salaries (emp_no, from_date, salary)
```

```
VALUES ('1', '2014-08-10 18:14:38:742191', '1341')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('1', '2013-09-10 15:54:27:038259', '2556')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('1', 0', '2013-03-23 06:40:11:282671', '2016-05-14
02:05:49:975234')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('2', '1969-10-06 01:46:40:633932', 'LESLEY', 'HOLDEN',
'LESLEY.HOLDEN5@company.com', '2009-05-23 16:31:37:111929')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('2', 'Web programmer', '2015-08-02 02:34:57:902436')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('2', 'Tester', '2015-10-30 06:18:11:669212')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('2', '2014-12-10 06:43:06:796102', '1175')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('2', '2016-02-10 08:36:36:176285', '3971')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('2', '2016-07-10 14:24:48:789951', '3821')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('2', '2014-09-10 11:25:50:726857', '1143')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('2', '2013-08-10 15:46:46:526569', '2026')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('2', '2014-06-10 14:00:05:738601', '3498')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('2', '2013-10-10 21:48:34:526932', '1867')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('2', 0', '2014-06-22 21:27:30:317624', '2016-10-06
02:09:40:659984')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('2', 0', '2015-10-01 11:52:14:846669', '2016-03-14
10:30:00:308493')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('2', 1', '2014-02-20 21:26:28:266624', '2016-05-23
13:11:34:684015')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('3', '1960-08-26 09:39:38:594233', 'NATHANIEL', 'GAINES',
'NATHANIEL.GAINES19@company.com', '2006-06-26 09:54:09:340449')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('3', 'Application programmer', '2014-02-15 16:11:33:544044')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2013-09-10 08:08:05:170507', '2807')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2016-02-10 03:01:25:896444', '3806')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2013-07-10 15:49:27:960987', '3502')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2015-11-10 01:52:36:891470', '2054')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2014-03-10 16:56:53:506174', '1980')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2016-09-10 04:30:05:305990', '1399')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2015-12-10 15:47:58:535318', '1887')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2015-12-10 02:19:26:386376', '1622')
```

```
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('3', '2014-08-10 22:29:22:257992', '2266')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('3', 0', '2014-02-27 07:41:22:442728', '2016-12-11
20:58:58:043453')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('3', 1', '2014-03-24 03:43:15:659999', '2016-11-03
08:21:48:432099')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('3', 0', '2015-08-27 16:47:55:814724', '2016-10-23
12:59:48:042908')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('4', '1974-10-30 16:28:23:157321', 'ANDREAS', 'ALVAREZ',
'ANDREAS.ALVAREZ12@company.com', '2003-09-24 03:56:44:761820')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('4', 'System programmer', '2013-09-23 14:34:32:982344')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('4', 'Web programmer', '2015-10-01 15:23:45:885119')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('4', '2014-05-10 08:49:23:117912', '2155')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('4', 0', '2016-10-18 16:23:54:527656', '2016-06-25
02:42:40:452701')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('4', 1', '2016-06-13 23:04:18:849335', '2016-12-26
05:49:09:313249')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('5', '1988-02-29 21:22:26:308496', 'JASON', 'WARE',
'JASON.WARE52@company.com', '2004-01-10 19:33:11:422279')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('5', 'Tester', '2013-12-23 21:51:37:650305')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('5', '2016-08-10 05:48:13:847227', '1926')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('5', '2015-07-10 16:37:53:617543', '2153')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('5', '2014-08-10 11:49:20:131086', '2094')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('5', 1', '2016-09-09 17:11:42:744008', '2016-11-04
20:32:52:644005')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('5', 0', '2016-06-08 11:52:12:905230', '2016-05-26
02:27:58:034218')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('6', '1980-07-27 22:47:30:865597', 'KEITH', 'POPE',
'KEITH.POPE3@company.com', '2004-05-06 23:13:53:585447')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('6', 'Application programmer', '2016-07-05 03:45:55:370797')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('6', 'System programmer', '2014-01-25 05:14:13:206624')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('6', '2015-08-10 07:24:11:966593', '1293')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('6', '2014-03-10 21:43:29:558314', '2372')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('6', '2014-08-10 21:30:54:525850', '1953')
```

```
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('6', '2016-03-10 00:13:50:604379', '3463')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('6', '2014-04-10 22:02:22:463017', '3263')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('6', 0', '2015-07-27 23:03:05:747778', '2016-03-13
17:14:14:527026')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('6', 0', '2013-10-20 10:56:19:968619', '2016-06-20
07:07:43:383955')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('6', 0', '2016-12-01 02:45:19:971720', '2016-07-23
18:32:03:071285')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('7', '1978-02-10 10:40:00:365331', 'CALEB', 'VELASQUEZ',
'CALEB.VELASQUEZ11@company.com', '2004-05-03 19:15:28:478055')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('7', 'Web programmer', '2015-04-19 20:49:23:061218')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2015-09-10 13:48:14:303983', '1187')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2015-10-10 01:08:43:748566', '2189')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2013-06-10 00:37:45:689105', '1313')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2015-07-10 02:51:58:909793', '2139')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2014-04-10 15:35:30:891349', '2041')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2016-02-10 07:29:50:792280', '1436')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2016-02-10 18:02:38:192525', '3693')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2016-04-10 19:20:25:341447', '1793')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('7', '2014-09-10 04:42:28:899308', '3678')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('7', 0', '2015-11-28 06:18:57:675715', '2016-07-04
11:39:09:820965')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('7', 1', '2016-02-26 06:02:21:394377', '2016-03-01
14:49:47:949189')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('7', 0', '2013-08-15 21:50:37:477220', '2016-05-01
22:00:24:354873')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('8', '1993-10-19 09:05:51:492776', 'EUGENIO', 'DALTON',
'EUGENIO.DALTON17@company.com', '2005-07-14 06:41:53:508093')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('8', 'Tester', '2015-06-05 23:19:18:959299')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('8', 'Application programmer', '2014-03-02 03:37:26:053914')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('8', '2014-06-10 23:03:09:352835', '1501')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('8', '2014-11-10 20:22:37:576705', '1381')
INSERT INTO salaries (emp_no, from_date, salary)
```

```
VALUES ('8', '2016-06-10 23:59:46:112916', '3924')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('8', '2016-01-10 12:36:40:178861', '3266')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('8', '2015-04-10 18:04:32:056031', '1261')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('8', '2015-09-10 18:31:54:242887', '2252')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('8', '2016-10-10 03:27:34:782052', '2973')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('8', '2016-07-10 16:05:32:055251', '1885')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('8', 0', '2013-11-04 02:09:59:049434', '2016-10-12
21:08:27:021079')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('8', 0', '2016-08-09 23:43:10:712859', '2016-04-24
10:19:37:211732')
INSERT INTO employees (emp_no, birth_date, first_name, last_name, email,
hire_date)
VALUES ('9', '1970-09-12 16:56:47:677758', 'DARYL', 'HOLLOWAY',
'DARYL.HOLLOWAY63@company.com', '2000-05-16 19:40:36:016782')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('9', 'System programmer', '2015-10-30 22:59:34:902183')
INSERT INTO specialization (emp_no, title, from_date)
VALUES ('9', 'Web programmer', '2015-03-16 08:21:41:628063')
INSERT INTO salaries (emp_no, from_date, salary)
VALUES ('9', '2013-12-10 20:28:20:030498', '3545')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('9', 1', '2014-12-13 00:40:49:072202', '2016-06-02
07:24:37:288966')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('9', 1', '2014-08-03 08:26:25:799875', '2016-03-14
10:37:25:411789')
INSERT INTO dept_emp (emp_no, dept_no, from_date, to_date)
VALUES ('9', 0', '2016-03-21 04:57:21:699404', '2016-03-27
16:42:23:870840')
INSERT INTO dept_manager (emp_no, dept_no, from_date, to_date, active)
VALUES (1', 0', 2013-03-23 06:40:11:282671', 2016-05-14 02:05:49:975234',
'False')
INSERT INTO dept_manager (emp_no, dept_no, from_date, to_date, active)
VALUES (6', 0', 2013-10-20 10:56:19:968619', 2016-06-20 07:07:43:383955',
'True')
INSERT INTO dept_manager (emp_no, dept_no, from_date, to_date, active)
VALUES (3', 0', 2015-08-27 16:47:55:814724', 2016-10-23 12:59:48:042908',
'True')
INSERT INTO dept_manager (emp_no, dept_no, from_date, to_date, active)
VALUES (2', 1', 2014-02-20 21:26:28:266624', 2016-05-23 13:11:34:684015',
'False')
```

Príloha B

Výstupný súbor testovacej databázy 2

Testovacia databáza definovaná v podkapitole 4.2.2 bola zaplnená nasledujúcimi dátami. Predmetom skúmania týchto dát je dodržanie vzťahov hodnôt v stĺpcoch tabuľky.

```
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('0',   'CLAY   OLSON',   '8642:   €106.44',   '0',   '2016-11-21
09:22:30:499231',      '2016-11-21      13:22:57:635712',      '2016-11-21
12:22:16:223145')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('1',   'FARRAH   PERKINS',   '3FD:   €131.54',   '1',   '2016-10-22
07:07:41:001709',      '2016-10-22      19:07:09:072815',      '2016-10-22
10:07:22:758621')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('2',   'JACELYN   SIMS',   '932C:   $197.21',   '3',   '2016-05-14
06:28:23:823652',      '2016-05-14      12:28:03:364532',      '2016-05-14
09:28:52:953979')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('3',   'EMILIO   MCKINNEY',   'A31:   $145.94',   '4',   '2016-08-06
06:47:34:095125',      '2016-08-06      12:47:15:142578',      '2016-08-06
07:47:24:657974')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('4',   'HOSEA   WILEY',   'E34F8:   $151.63',   '6',   '2016-08-04
09:45:53:383336',      '2016-08-04      19:45:53:255096',      '2016-08-04
16:45:42:101776')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('5',   'WHITNEY   UNDERWOOD',   '729:   €153.93',   '7',   '2016-01-04
07:54:07:129152',      '2016-01-04      17:54:47:946518',      '2016-01-04
13:54:25:401764')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('6',   'WENDY   GROSS',   'F626:   $176.02',   '9',   '2016-03-31
09:45:20:008217',      '2016-03-31      19:45:34:022430',      '2016-03-31
13:45:30:404724')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('7',   'MICHAL   HICKMAN',   '7AC:   €168.14',   '12',   '2016-11-26
08:24:04:033313',      '2016-11-26      15:24:00:554810',      '2016-11-26
12:24:43:275208')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('8',   'SHANDRA   CHAPMAN',   'F2C7:   €173.86',   '9',   '2016-03-22
07:21:06:399769',      '2016-03-22      19:21:21:023834',      '2016-03-22
10:21:38:995667')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('9',   'KARLENE   KERR',   'E1DB:   €118.10',   '17',   '2016-09-06
09:19:15:857540',      '2016-09-06      14:19:09:714447',      '2016-09-06
11:19:19:934647')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('10',   'SAMMY   LANG',   'E92DF:   $114.93',   '17',   '2016-11-02
09:15:29:512221',      '2016-11-02      14:15:38:864471',      '2016-11-02
12:15:57:493195')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES   ('11',   'TOBY   NIELSEN',   'D1E64C:   $133.17',   '15',   '2016-08-12
07:19:11:857627',      '2016-08-12      11:19:36:544983',      '2016-08-12
10:19:36:704239')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
```

```
VALUES ('12', 'MAISHA GUZMAN', '279E: €157.93', '12', '2016-02-27
07:32:53:378156', '2016-02-27 14:32:01:154572', '2016-02-27
12:32:15:910736')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('13', 'ZACK FRITZ', '5F18: €169.92', '23', '2016-07-22
06:18:11:496707', '2016-07-22 16:18:18:258179', '2016-07-22
11:18:03:547180')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('14', 'BRYCE MERCER', 'F235E: €183.56', '14', '2016-07-16
09:26:04:284632', '2016-07-16 15:26:06:420227', '2016-07-16
11:26:06:503082')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('15', 'CLAUDETTE DORSEY', '84513: €176.88', '19', '2016-06-01
09:40:59:069577', '2016-06-01 13:40:51:600922', '2016-06-01
12:40:19:473816')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('16', 'NOAH BRAY', '48D: $125.90', '16', '2016-08-11
10:53:13:637506', '2016-08-11 14:53:57:116028', '2016-08-11
13:53:25:456879')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('17', 'BERTRAM POWERS', '48F5: $183.85', '21', '2016-03-17
08:56:38:598891', '2016-03-17 17:56:09:638802', '2016-03-17
10:56:30:992310')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('18', 'TRINIDAD LUTZ', '9CB: €148.58', '27', '2016-05-20
06:06:31:550997', '2016-05-20 15:06:29:108772', '2016-05-20
08:06:45:022980')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('19', 'PEGGIE WILEY', '68D: €140.63', '24', '2016-05-17
09:15:07:666450', '2016-05-17 20:15:50:802124', '2016-05-17
13:15:38:481419')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('20', 'ERVIN CALDERON', '1C2E: €146.50', '23', '2016-12-08
08:59:30:193283', '2016-12-08 12:59:29:790039', '2016-12-08
10:59:38:399612')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('21', 'CHANTELLE STARK', 'ABEDE: $110.87', '40', '2016-01-14
07:42:44:399396', '2016-01-14 13:42:14:308563', '2016-01-14
09:42:30:409462')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('22', 'TAMMARA WEST', '1D4D: €128.39', '22', '2016-10-04
08:18:10:650295', '2016-10-04 20:18:06:696625', '2016-10-04
16:18:41:160248')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('23', 'TOD LEON', 'CF53: €153.12', '42', '2016-02-28
10:24:08:389767', '2016-02-28 21:24:56:499191', '2016-02-28
19:24:09:829773')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('24', 'CHASE WALTER', '1E2AA: €180.32', '36', '2016-09-21
07:43:37:121110', '2016-09-21 15:43:41:033722', '2016-09-21
12:43:23:791443')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('25', 'REA CANTU', '6DE: $157.40', '29', '2016-06-15
06:48:55:774340', '2016-06-15 12:48:03:945984', '2016-06-15
08:48:17:359528')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES ('26', 'ERIN HANSON', 'EA713: €142.01', '28', '2016-06-11
07:04:49:515947', '2016-06-11 16:04:27:714874', '2016-06-11
15:04:31:440460')
```

```
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES  ('27',  'VAL  BERNARD',  '57F88:  €150.94',  '41',  '2016-06-16
07:58:46:667752',      '2016-06-16      16:58:04:716453',      '2016-06-16
10:58:50:125473')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES  ('28',  'LEONARD  FOSTER',  'ECFCF:  $126.49',  '29',  '2016-08-24
07:19:27:604874',      '2016-08-24      17:19:29:696198',      '2016-08-24
11:19:28:545166')
INSERT INTO work_day (id, name, reward, bigger_id, start, end, pause)
VALUES  ('29',  'HYO  CRUZ',  'CCB319:  $183.06',  '48',  '2016-04-09
07:15:26:966242',      '2016-04-09      16:15:42:600525',      '2016-04-09
15:15:18:711273')
```