



Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Tímový projekt

Elektronická prihláška - štatistika

Implementácia a overenie riešenia

Tím 10: Bc. Martin Donoval
 Bc. Ladislav Gažo
 Bc. Marek Gregor
 Bc. Michal Grosoš
 Bc. Peter Šimún

Vedúci projektu: Ing. Vladimír Marko
Školský rok: 2004 / 2005

Obsah

1	Úvod	1
2	Implementácia aplikácie FleXtat Browser	2
3	Implementačné úpravy v aplikácii iReport a JasperReport	8
3.1	FleXtat Report plugin	8
3.1.1	Ako vytvoriť novú agregáčnú funkciu	9
3.1.2	Ako pridať fakt	9
3.2	Záplata do JasperReports	11
4	Vytváranie špecializovaných scriptletov	12
4.1	Metódy a objekty použiteľné pri tvorbe scriptletu	13
4.2	Obslužné udalosti scriptletu pri tvorbe zostavy	14
4.3	Vzor scriptletu	14
5	Implementácia fleXPump	17
6	Databázová časť systému FleXtat	18
6.1	Databázový server	18
6.2	Fyzický model databáz	18
6.3	Správanie databázovej vrstvy	20
6.4	Popis vložených funkcií	24
6.4.1	get_next_id	25
6.4.2	cis_new_version	25
6.4.3	clear_database	25
6.5	Vytvorenie databázy	26
6.6	Importovanie údajov	26
6.7	Pridávanie faktov do databázy	26
7	Metodika pridávania faktov do systému FleXtat	27
7.1	Pojmy	27
7.2	Vytvorenie potrebných objektov, a zmena existujúcich, pre pridanie nového faktu v systéme FleXtat	27
7.2.1	Vytvorenie nového číselníka v databáze aplikácie	27
7.2.2	Pridanie faktu do tabuľky údajov	28
7.2.3	Pridanie nového faktu do náhľadu pre zobrazovanie dát	29
7.2.4	Pridanie faktu do náhľadu pre vkladanie dát	30
7.2.5	Pridanie faktu do tabuľky mapovania polí pre štatistiky	32
7.2.6	Úprava nástroja fleXPump	33
8	Akceptačné testy	34
8.1	Všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat	34
8.2	Testovanie prehliadania zostáv	34

8.3	Testovanie úprav vlastností zostáv.....	35
8.4	Testovanie tlače zostáv	36
8.5	Testovanie exportu zostáv	37
8.6	Testovanie vytvárania zostáv	37
8.6.1	Vytváranie zostáv pomocou modulu fleXtat report wizard.....	37
8.6.2	Vytváranie špecializovaných zostáv pomocou scriptletov	38
8.7	Testovanie správy zostáv.....	39
8.8	Testovanie administrácie databázy.....	40
8.8.1	Testovanie aktualizácie databázy aplikáciou fleXPump	40
8.8.2	Testovanie zálohovania a obnovy databázy	41
8.9	Testovanie preddefinovaných zostáv	42
8.9.1	Rozdelenie študentov/uchádzačov podľa veku	42
8.9.2	Počet prihlásených a počet prijatých študentov.....	43
8.9.3	Počet prichádzajúcich prihlášok v závislosti od dátumu	43
8.9.4	Vplyv strednej školy na úspešnosť prijímacej skúšky	43
8.9.5	Rozdelenie študentov na základe národnosti a miesta bydliska	44
8.9.6	Vplyv súťaží na úspešnosť prijímacej skúšky.....	45
8.9.7	Úspešnosť prihlásených študentov v závislosti od úspešnosti maturitnej skúšky z určitého predmetu.....	45
9	Záver	46

1 Úvod

Tento dokument obsahuje výstupnú dokumentáciu k systému FlexTat v rámci predmetu tímový projekt počas letného semestra 2004/2005.

Systém FlexTat (FLEXible sTATistics system) je určený pre štatistické vyhodnocovanie elektronickej prihlášky. Umožňuje jednoduchým a pohodlným spôsobom vygenerovať štatistiku založenú na informáciách o elektronických prihláškach. Vygenerovaná štatistika prezentuje údaje vo forme tabuliek a grafov. Tieto údaje je možné v ucelenej forme ďalej exportovať do rôznych formátov, a tým zabezpečiť publikovanie týchto výsledkov rôznymi spôsobmi.

Počas letného semestra bol implementovaný systém FlexTat, návrh systému sa oproti zimnému semestru nezmenil ale iba rozšíril. Jednotlivé rozšírenia návrhu sú popísané v jednotlivých častiach spolu s opisom implementácie.

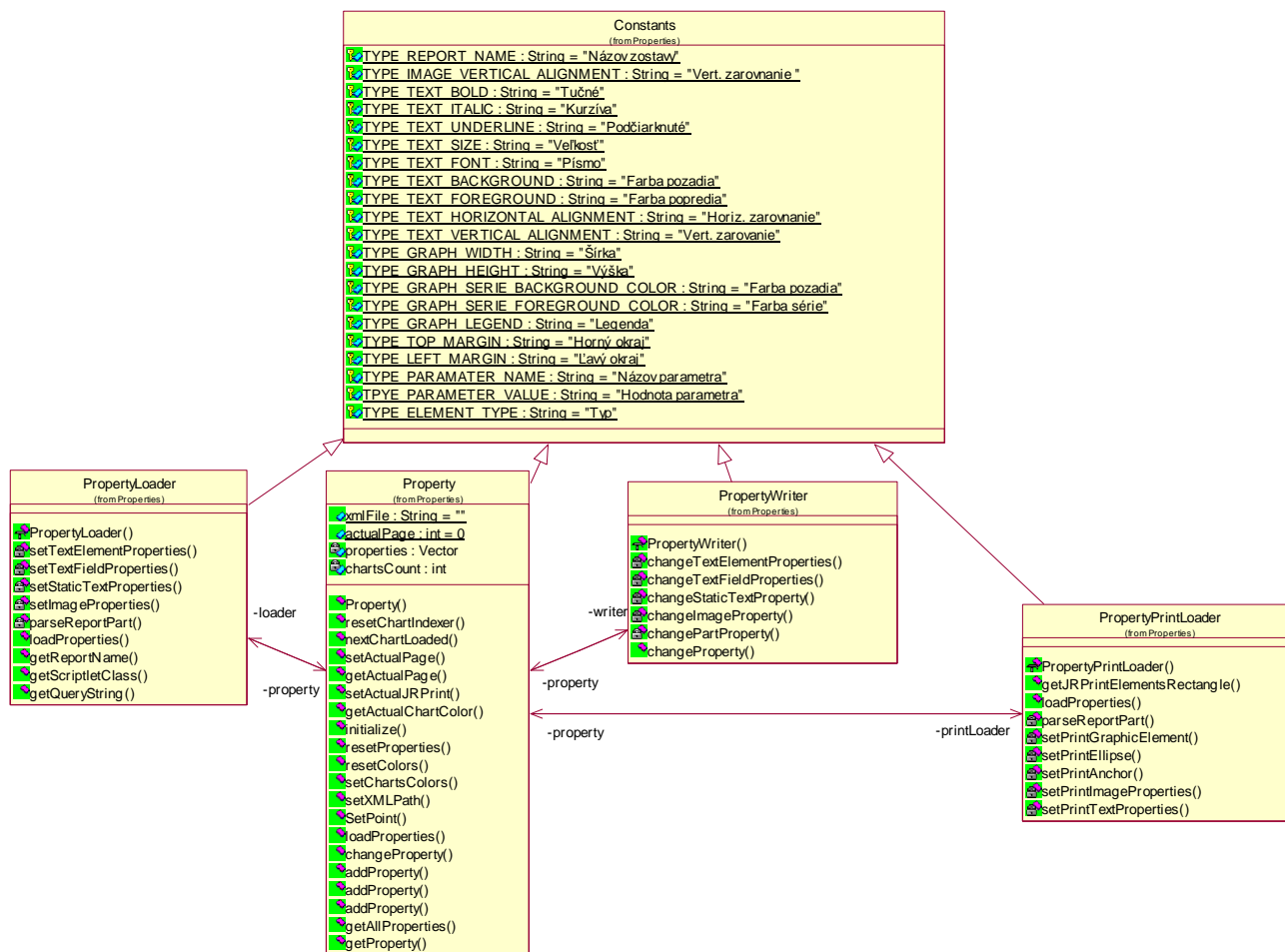
2 Implementácia aplikácie FlexTat Browser

Spúšťacím miestom flexTat browseru je trieda FXBrowser v balíčku org.fellas.flextat.ui. Na začiatku sa načítava úvodná obrazovka, ktorá informuje o stave načítavania aplikácie (trieda FXSplash). Nasleduje otvorenie prihlasovacieho okna reprezentované triedou FXLogin. FXLogin vracia vyplnené údaje v triede DBUser s názvom returnedObject na overenie funkciou login(). Po správnom overení prihlasovacích údajov sa otvorí hlavné okno. FXBrowser obsahuje menu, panel nástrojov, ľavý a pravý toolbox a prehliadací komponent zostáv. V ľavom toolboxe je strom zostáv zotriedených v kategóriách. Využíva sa na to trieda JTree doplnená o algoritmus načítavania z databázy. Algoritmus načítavania je zabezpečený priradením modelu reprezentovaného triedou DBTree, ktorý zároveň obsluhuje aj databázu z hľadiska manažmentu kategórií. Prehliadací komponent reprezentuje trieda FXView, ktorá je prerobená z triedy dostupnej v knižnici JasperReports. FXView má zmenené obslužné metódy tlačidiel a iných ovládacích prvkov z private na public. Tiež umožňuje skryť štandardný panel nástrojov, resp. pridať štandardný panel do iného panelu.

Browser má dialóg na pridávanie, resp. úpravu zostáv (FXAddReportDialog). Tento dialóg má za úlohu vyplniť návratový objekt daného typu DBReport, aby sa potom údaje mohli zapísať do databázy. K databáze sa pristupuje jednotným spôsobom cez triedu FXConnect, ktorá sa inicializuje pri prihlasovaní, na základe používateľskej role. Ostatné nastavenia sa načítavajú v triede FXAppSettings, ako napr. lokalizácia, úložisko zostáv, či umiestnenie adresára so zostavami. Na správu používateľov Browser na základe role umožní prístup k manažmentu užívateľov, reprezentovaného triedou FXManageUsers. Táto trieda má na pridávanie, resp. úpravu používateľov vyhradený dialóg (FXAddUserDialog).

Lokalizácia je zabezpečená pomocou vstavaného mechanizmu jazyku Java. V triede Messages sa nachádza metóda na načítanie správneho prekladu z prislúchajúceho jazykového súboru. Jazykový súbor má formát messages_<jazyk>_<KRAJINA>.properties. Obdobne v triede Mnemonic je metóda, ktorá definuje klávesové skratky v menu na základe lokalizácie. K tej sú priradené jazykové súbory vo formáte mnemonic__<jazyk>_<KRAJINA>.properties. V prostredí Eclipse sa dá dodatočné lokalizovanie (ale aj odlokalizovanie) kódu dosiahnuť cez menu Source -> Externalize Strings...

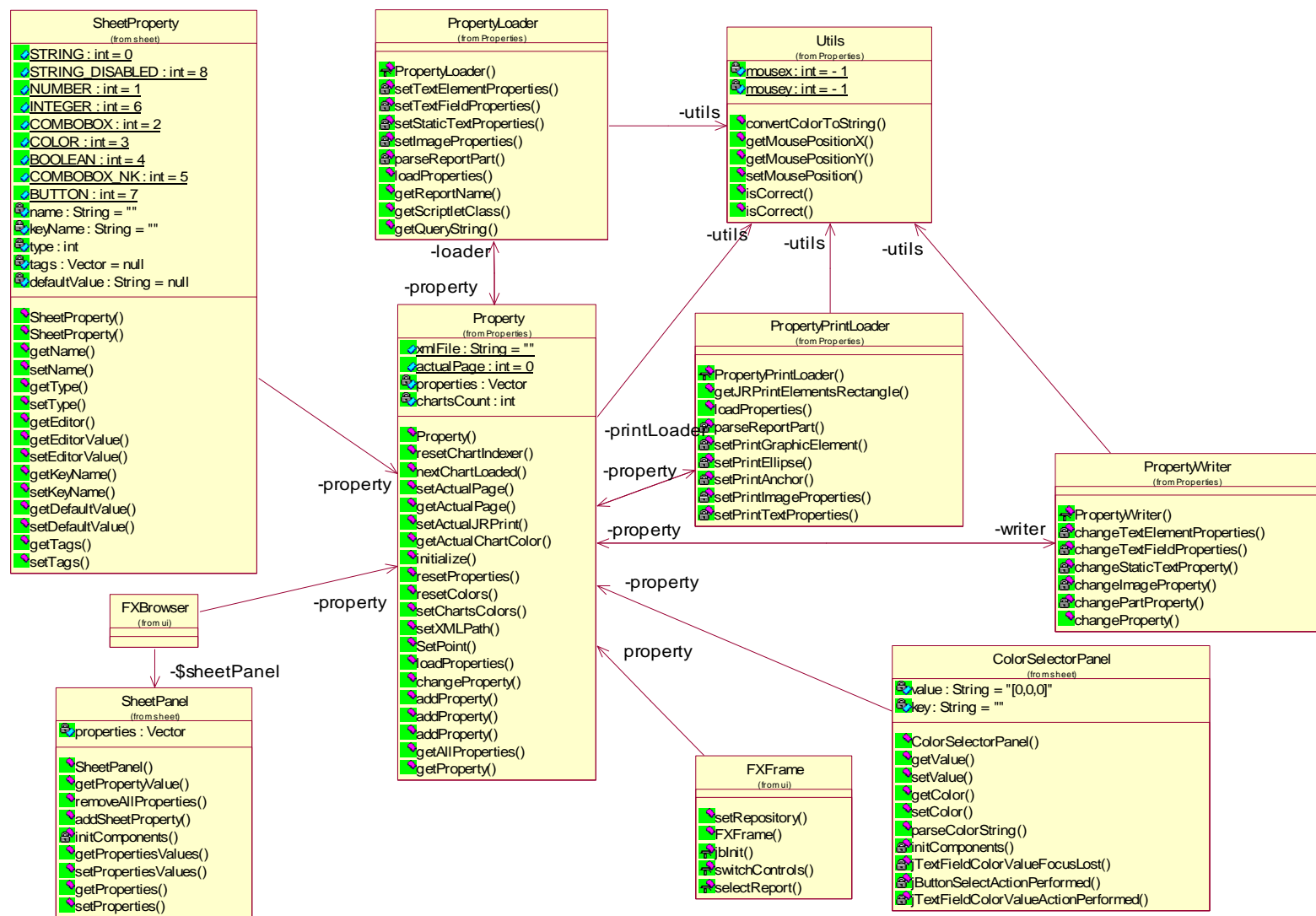
Na reprezentáciu vlastností boli implementované triedy zobrazené na nasledovných diagramoch.



Obr. 1 : Diagram tried zachytávajúci triedy potrebné na načítavanie a ukladanie vlastností

Základom uvedených tried je trieda **Constants**, ktorá obsahuje konštanty, potrebné pri správe vlastností. Táto trieda obsahuje konštanty definujúce jedinečné typy vlastností. Ďalšou triedou je trieda **PropertyLoader**, ktorá zabezpečuje načítanie vlastností zo súboru JRXML pridruženého k samotnému reportu. Všetky načítané vlastnosti sa ukladajú, s výnimkou pozície elementu, ktorému dané vlastnosti patria. Táto pozícia je neskôr pridružená danému elementu pomocou triedy **PrintPropertyLoader**, ktorá reprezentuje elementy tak, ako sú zobrazované na obrazovke používateľa. Všetky elementy sú synchronizované pomocou kľúča, ktorý je jedinečný pre každý element (Na tieto účely muselo byť vytvorené rozšírenie modulu **JasperReport**, vid'. Kapitola 3.2). Na základe takto určenej pozície je možné identifikovať element, nad ktorým používateľ vykonal stlačenie ľavého tlačítka myši. Udržiavané vlastnosti je potrebné ukladať späť do súboru JRXML, na čo slúži trieda **PropertyWriter**. Poslednou triedou je trieda **Property**, ktorá vykonáva načítavanie, zmenu a ukladanie vlastností. Táto trieda spolupracuje so všetkými uvedenými triedami a predstavuje rozhranie medzi vlastnosťami a aplikáciou flexstat, resp. samotným používateľom.

Tieto uvedené triedy zabezpečovali správu vlastností. Ďalší diagram zachytáva triedy, vykonávajúce zobrazovanie vlastností na obrazovke a ich obsluhu.



Obr. 2: Celkový pohľad na triedy reprezentujúce vlastnosti

Trieda `FXBrowser` predstavuje jadro samotnej aplikácie, ktorá obsahuje celé grafické rozhranie, preto obsahuje aj rozhranie pre zoznam vlastností (zoznam zobrazený v pravej časti aplikácie). Preto je potrebné prepojenie medzi triedou `FXBrowser` a triedou `Property`. Trieda `Property` v tomto prípade vykonáva prepojenie na triedy umožňujúce načítavanie, zmenu a ukladanie vlastností reportov. Takto získané vlastnosti sa zobrazujú pomocou triedy `SheetPanel` na obrazovke používateľa (panel v pravej časti obrazovky).

Uvedené vlastnosti môžu byť rôzneho druhu :

- Editovateľná textová položka
- Needitovateľná textová položka
- Celočíselná položka
- Položka obsahujúca reálne číslo
- Zoznam hodnôt (combo box)
- Zoznam číselných hodnôt (numeric combo box)
- Farba

V konečnom dôsledku sa rozdeľujú uvedené vlastnosti do 3 skupín :

- Vlastnosti reprezentujúce farbu
- Vlastnosti reprezentujúce zoznam (či už `ComboBox` alebo číselný `ComboBox`)
- Vlastnosti s textovou položkou (či už editovateľnú alebo nie alebo či už numerickú alebo nie).

Prvú uvedenú položku zastrešuje trieda `ColorSelectorPanel`, ktorá umožňuje zvoliť ľubovoľnú farbu. Ostatné dve skupiny zastrešuje trieda `SheetProperty`. Pomocou tejto triedy sa zobrazuje konkrétna vlastnosť na obrazovku používateľa.

Posledná trieda, trieda `FXFrame`, udržiava aktuálnu inštanciu triedy `JasperPrint`, ktorá slúži na synchronizáciu rozmiestnenia elementov v reporte.

Prácu so zostavami na aplikačnej úrovni medzi `flexTat` browserom a knižnicou `JasperReports` zabezpečujú triedy `Factory`, `Repository` a `Report` nachádzajúce sa v balíčku `org.fellas.flexTat.reportrepository`. Trieda `Report` reprezentuje jednu zostavu s definíciou mena, opisu a priradeného skriptletu. V triede sa nachádza aj obsluha dátového zdroja, z ktorého sa načítavajú údaje pre výsledok do zobrazovacieho komponentu `FXView`. Metóda `getDataSource()` sa stará o správne rozhodnutie, o aký dátový zdroj ide (žiadny, skriptlet, XML alebo DB). Tomu prislúchajú hodnoty v interných vlastnostiach zostavy v kľúči `dsType` (v poradí `empty`, `scriptlet`, `XML` a `DB` – sú case-sensitive). Interné vlastnosti sú uložené v hash tabuľke `inProperties`. Na základe typu dátového zdroja môžu v tejto tabuľke existovať aj ďalšie špecifické hodnoty. Napr. pre XML dátový zdroj sa zadávajú ďalšie dve vlastnosti – `xmlData` a `xmlRootNode`. Tie definujú zdroj načítavania a koreňový element v XML súbore. Pre typ DB sa musí definovať vlastnosť v kľúči `dbSelect`, kde sa vkladá SQL príkaz pre zostavu, väčšinou zapísaný priamo v súbore zostavy.

Repository uchováva všetky zostavy a kategórie reprezentované inštanciami triedy DBTree v stromovej reprezentácii. Pomocou metódy fillRepository() sa všetky potrebné údaje načítajú z databázy a vytvorí sa stromová štruktúra. Ku konkrétnym zostavám sa dá pristupovať metódou getReport(). Trieda Repository zároveň slúži ako model pre triedu JTree, ktorá zobrazuje strom.

Trieda Factory má za úlohu vytvoriť výstupnú formu zostavy, vhodnú pre knižnicu JasperReports. Vstupným bodom je metóda createView(), ktorá z triedy Report načíta údaje a vytvorí object JasperPrint, ktorý sa podsúva triede FXView, ktorá ho zobrazuje. Metóda createView() volá metódu getReport(), ktorá buď vytvorí .jasper súbor, uchovávajúci rozloženie zostavy alebo toto rozloženie načíta z existujúceho .jasper súboru. K načítanému .jasper súboru sa doplní dátový zdroj zostavy a vygenerujú sa vyplnené stránky pomocou statickej metódy triedy JasperFillManager fillReport().

O prístup na databázovú vrstvu sa stará balíček org.fellas.flextat.database. Balíček obsahuje tri triedy na obsluhu výnimiek odvodené od triedy DBException a to: DBInsertException, DBUpdateException a DBRemoveException. Výnimky vyjadrujú chybu, vznikajúcu pri vkladaní, úprave a odstraňovaní záznamov v databáze. Obsahuje triedy výnimiek ImportException a ExportException, ktoré vznikajú pri importovaní do a exportovaní súboru z databázy.

Ďalej existuje nadradená trieda DBTable, ktorá reprezentuje triedu na prístup k databáze. Triedy DBUser, DBReport a DBTree sú konkrétne podtriedy, implementujúce prístup k tabuľkám používateľov, zostáv a stromu zostáv a kategórií. Každá z nich obsahuje metódy insert, update a delete, ktoré pre danú inštanciu triedy s vyplnenými poľami vykonávajú dotazy na SQL databázu. Ďalej tieto triedy obsahujú get metódy, ktoré získavajú údaje z SQL databázy. DBTree reprezentuje jednu položku v strome, ktorá môže byť či už kategória alebo odkaz na zostavu. Trieda DTree obsahuje navyše metódy cut a copy, ktoré sú zviazané s požiadavkou vyplývajúcou z práce so schránkou a dokážu premiestňovať, resp. kopírovať zostavy a kategórie. Na DBTree naväzuje trieda DBReport, ktorá reprezentuje parametre zostavy, súbor zostavy a skriptlet. Trieda DBReport zabezpečuje aj importovanie a exportovanie súborov, ktoré ukladá do adresára reports v hlavnom adresári aplikácie. K tejto ceste sa dá dostať z triedy FXAppSettings. Pri každom novom naštartovaní aplikácie sa zostavy načítavajú z databázy. Trieda DBUser reprezentuje jedného používateľa v databáze. Obsahuje štandardné metódy na manipuláciu s údajmi používateľa. Aktuálne prihlásený používateľ je zaznamenaný v triede FXAppSettings.

Na pohodlné úpravy stromu kategórií a zostáv bola vytvorená trieda FXClipboard. Nad schránkou možno vykonávať dve operácie – vyňatie a kopírovanie. Na identifikovanie akcie sú definované dve konštanty – CUT_ACTION a COPY_ACTION. Schránku je možné vymazať metódou clear(). Prvok v schránke sa dá vybrať metódou getItem() a vložiť metódou setItem().

3 Implementačné úpravy v aplikácii iReport a JasperReport

3.1 *FlexTat Report plugin*

Na uľahčenie vytvárania zostáv v aplikácii iReport slúži flexTat report plugin. Funguje ako sprievodca. Plugin je implementovaný pomocou štandardného rozhrania na vytváranie pluginov, dodávané so zdrojovými kódmi aplikácie iReport. Ide o abstraktnú triedu `IReportPlugin`, od ktorej je odvodená trieda `FlexTatReportPlugin`, reprezentujúca štartovací bod flexTat pluginu (Obr. 3)

Zo štartovacieho bodu sa vytvára inštancia dialógového okna sprievodcu `WizardDialog`. `WizardDialog` obsahuje šesť panelov, reprezentujúcich šesť krokov sprievodcu. Medzi jednotlivými krokmi sa dá pohybovať pomocou tlačidiel `JButtonNext` a `JButtonPrev` . V obslužných metódach týchto tlačidiel sa volá funkcia `setStep` , ktorá nastavuje aktuálny krok. V prechode medzi prvým a druhým krokom sa načítava zoznam faktov z databázy – konkrétne z tabuľky `cis_fields_mapping` . Na konektivitu na databázu sa využíva zadefinovaný konektor v prostredí aplikácie iReport s potrebnými parametrami, reprezentovaný triedou `IReportConnection` . Správnu inštanciu tejto triedy poskytuje trieda `Query` metódou `getIReportConnection()` . Metóda prehľadáva zoznam konektorov definovaných v iReporte a hľadá ten, ktorý má názov „flexdb“.

V nasledujúcom kroku sa vyberajú fakty. Zvolené fakty sa pridávajú do tabuľky, v ktorej je možné definovať aj funkciu nad daným faktom. Fakt je reprezentovaný triedou `DBField` . V triede sa nachádzajú informácie o type faktu, jeho mapovaní na tabuľku v databáze, názve ktorý sa zobrazí v zostave a priradenej funkcii. Funkciu reprezentuje abstraktná trieda `AgregateFunction` . Funkcia má pridelený návratový typ v poli `classType` , ktorý hovorí o tom, aký typ funkcia po aplikovaní vracia (napr. v prípade návratového typu `integer` sa dosadí reťazec `java.lang.Integer`). Niektoré funkcie vyžadujú špecifický vstupný typ faktu – napr. číselnú hodnotu. Preto sa typ faktu, ktorý vstupuje do funkcie, validuje funkciou `validate()` , či spadá do stromu dedenia, v ktorom sa nachádza aj potrebný rodičovský typ definovaný v poli `requiredParentClass` . Špeciálnym typom funkcie je `NoneFunction` , ktorá reprezentuje, že sa neaplikuje žiadna funkcia na fakt.

Všetky triedy `DBField` sa vytvárajú v továrni `DBFieldFactory` , pomocou metódy `loadFactsList()` . Trieda pristupuje pomocou SQL príkazov priamo k databáze a oddeľuje vytváranie faktov do centralizovaného miesta.

V ďalších krokoch sa nastavuje zoskupovanie, rozvrhnutie zostáv a voľba grafu. Typy grafov v zozname sa načítavajú z dostupných grafov v iReporte. Zoznam grafov udržiava trieda `AvailableCharts` . Po prejdení na posledný krok sa začne generovať zostava. Generovanie prebieha v jednej z dvoch metód v závislosti na zvolenom rozvrhnutí: `createColumnarReport()` alebo `createTabularReport()` . Metódy sú veľmi podobné, pozostávajú z dvoch častí. Prvá časť

vytvára zo šablóny rozvrhnutie tabuľky a druhá časť vytvára graf. Na záver sa pomocou metódy `createQuery()` vytvorí výsledný dotaz na databázu.

Výsledkom je zostava s definovaným rozložením. Šablóny sú vlastne zostavy s neasociovanými poľami. Šablóny v tvare `<meno>T.jrxml` sú tabuľkové a `<meno>C.jrxml` sú stĺpcové. Sú umiestnené v adresári `templates`.

Priebeh v plugine sa monitoruje a zapisuje do logového súboru `pluginlog.html` a `pluginlog.xml`. HTML súbor v prehľadnej forme ukazuje aktivitu. XML súbor slúži na jednoduchú výmenu a spracovanie záznamov. Na zabezpečenie logovania sa využíva knižnica `log4j`, ktorú používa aj `iReport`. Miera logovania sa nastavuje v súbore `log4j.properties` v nasledovnom riadku:

```
log4j.logger.org.fellax.flextat.plugin=DEBUG, Console, XML, HTML
```

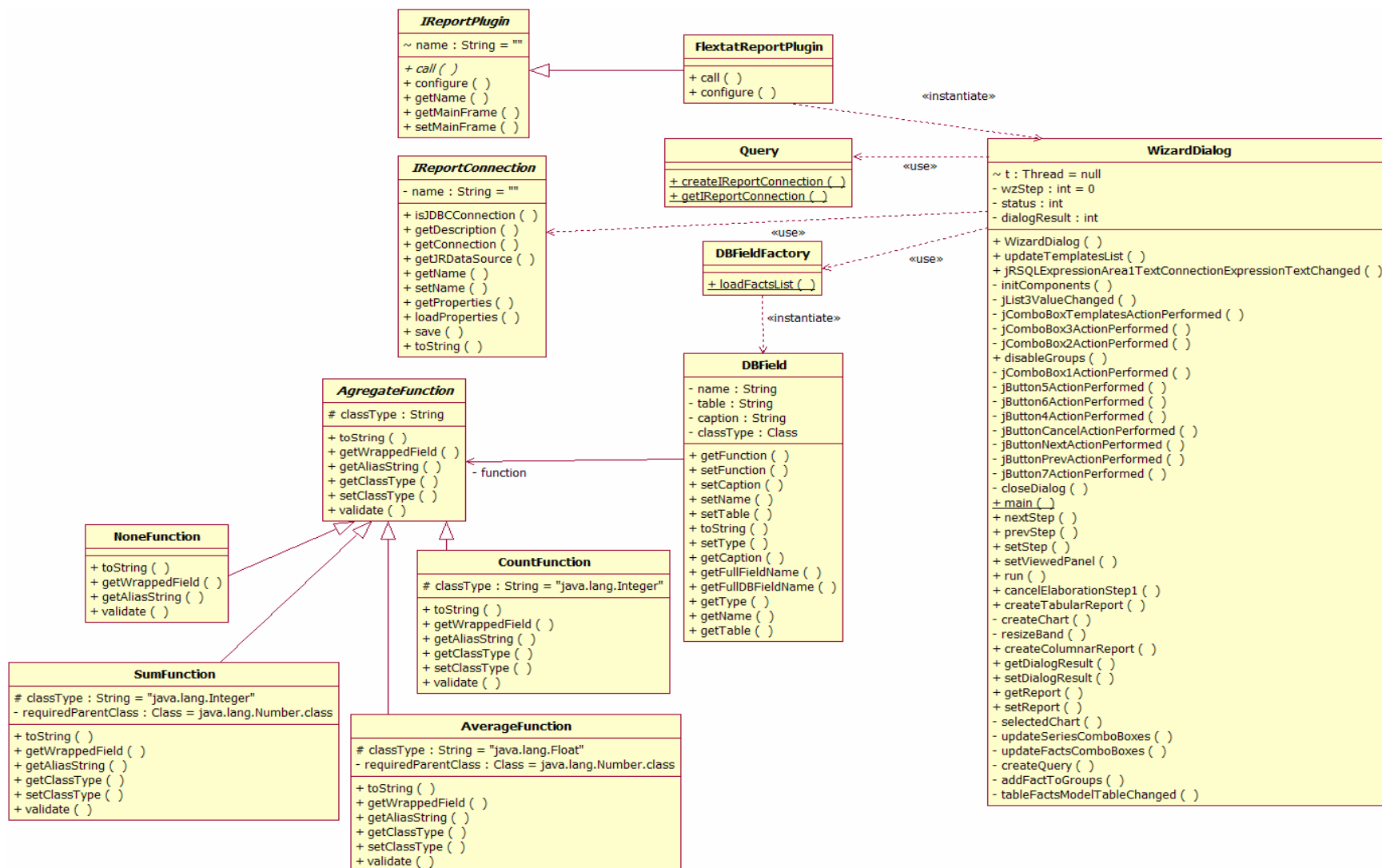
Pre zníženie množstva logovaných informácií treba nastaviť niektorú z nasledovných možných hodnôt: `INFO`, `WARN`, `ERROR`, `FATAL`. V poradí sa množstvo informácií znižuje. `FATAL` zaznamenáva len kritické informácie, pri ktorých plugin zlyhá a sa zrúti.

3.1.1 Ako vytvoriť novú agregáčnú funkciu

Novú agregáčnú funkciu je možné pridať do pluginu odvodením od abstraktnej triedy `AgregateFunction`. Rovnako je treba preťažiť všetky abstraktné metódy v triede a v prípade potreby zdefinovať typ návratovej hodnoty v poli `classType`. Ak sa kladú na funkciu špeciálne nároky na vstupný hodnotový typ, treba zdefinovať aj pole `requiredParentClass`. Vzorovou ukážkou je trieda `AverageFunction`, ktorá implementuje aj overovanie vo funkcii `validate()`.

3.1.2 Ako pridať fakt

Fakt sa pridáva do tabuľky `cis_fields_mapping`. Vyplnia sa tri potrebné stĺpce: `caption`, `name` a `type`. `Caption` je hodnota, ktorá sa vypíše používateľovi v zozname faktov. `Name` je meno stĺpca v tabuľke `dat_stat_application`. `Type` je dátový typ stĺpca, reprezentovaný niektorou triedou v jazyku Java. Zadáva sa ako textový reťazec, napr. pre typ `integer` to je `java.lang.Integer`, pre typ dátum je to `java.sql.Date` a pre reťazcový typ zas `java.lang.String`.



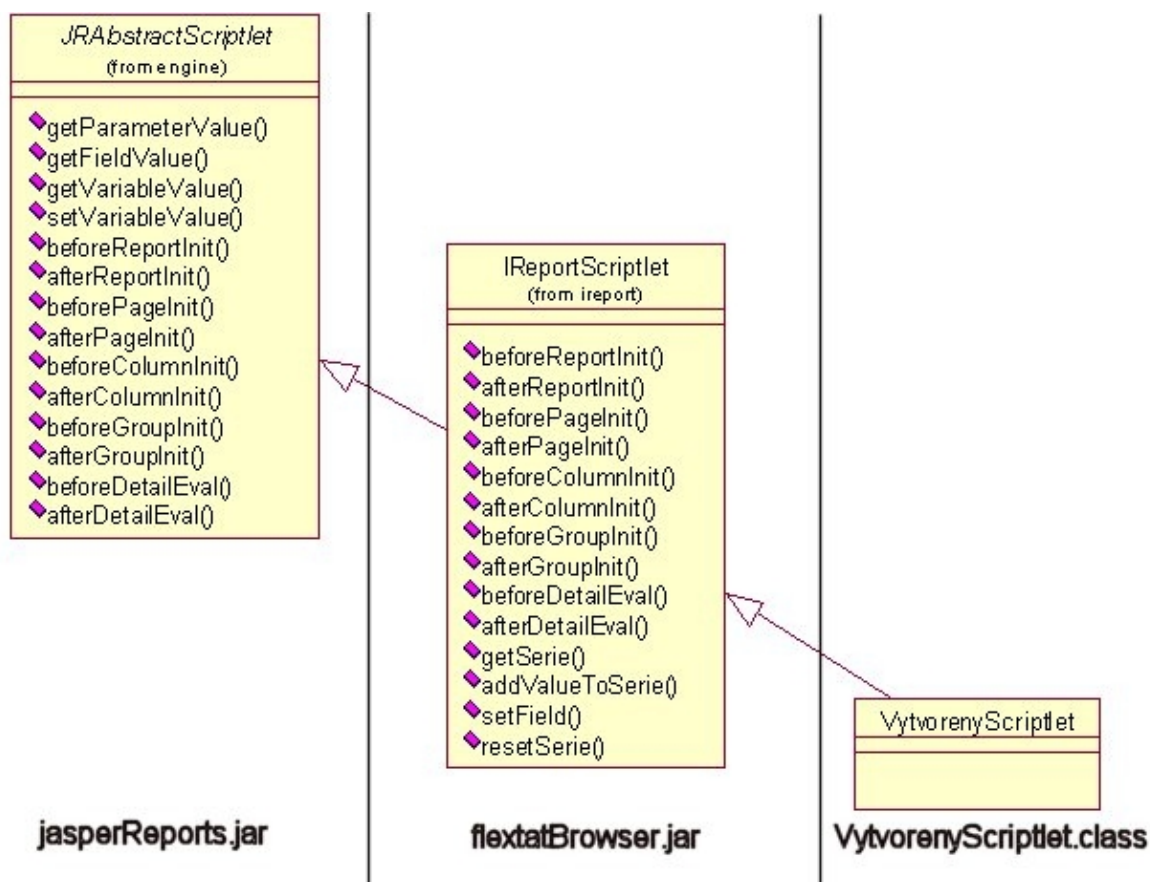
Obr. 3: Dátový model zobrazujúci hierarchiu tried slúžiacich na tvorbu pluginov

3.2 Záplata do JasperReports

Do knižnice JasperReports 0.6.4 bola dorobená dodatočná funkcionálna, ktorá umožňuje správne určiť pozíciu elementov v zostave. Jedná sa o publikovanie poľa „key“ v triedach, reprezentujúcich výsledné tlačéné elementy s predponou JRFill a JRPrint. Ďalej boli dorobené prístupové get a set metódy k tomuto poľu. Záplata je umiestnená v RCS súbore key_field_patch. Aplikovanie záplaty je možné v prostredí Eclipse pomocou funkcie Apply patch v kontextovom menu projektu v položke Team alebo pomocou iného záplatovacieho nástroja.

4 Vytváranie špecializovaných scriptletov

Pre vytváranie špecializovaných scriptletov ktoré realizujú operácie ktoré nie je možné jednoduchým spôsobom zadať pomocou FlexTat Report pluginu je potrebné mať naštudovanú používateľskú príručku k systému FlexTat, predovšetkým kapitolu o manuálnej tvorbe zostáv pomocou nástroja iReport, kde sú popísané jednotlivé dátové elementy popisujúce prenos údajov informácii medzi zostavou a scriptletom. Jedná sa o premenné, parametre, polia a série. Spôsob vytvárania odkazov na tieto elementy v zostavách je popísaný v spomínanej používateľskej príručke. Táto kapitola opisuje okrem práce s týmito dátovými entitami zo scriptletu aj tvorbu scriptletov ako takých.



Obr. 4: Hierarchia tried pri tvorbe scriptletu

Pri vytváraní zostavy v nástroji iReport sa automaticky definuje typ scriptletu ako trieda `IReportScriptlet`. Každý nový scriptlet teda musí byť odvodený od triedy `it.businesslogic.iReport.IReportScriptlet`. Táto trieda implementuje alebo dedí

všetky dôležité metódy na prácu s premennými, parametrami, sériami a poliami. Na Obr. 4 je znázornená hierarchia tried pri vytváraní scriptletu a ich zaradenie do jednotlivých knižníc.

4.1 Metódy a objekty použiteľné pri tvorbe scriptletu

Počas práce na projekte sme rozšírili funkčnosť triedy `IreportScriptlet` o prácu so sériami a nastavovanie polí. Tieto funkcionality neboli pred tým implementované. Na prácu s premennými, sériami a poliami je teda možné použiť nasledovné metódy. Ide `get` a `set` metódy pre jednotlivé dátové elementy, kde reťazcový parameter špecifikuje názov dátového elementu (premennej, parametra, série, pol'a).

- pre prácu s premennými je možné použiť metódy:
 - `getVariableValue(variableName : String) : Object`
 - `setVariableValue(variableName : String, value : Object) : void`
- pre prácu s parametrami je možné použiť metódy:
 - `getParameterValue(parameterName : String) : Object`
- pre prácu s poliami je možné použiť metódy:
 - `getFieldValue(fieldName : String) : Object`
 - `setField(name : String, values : Vector) : void`
- pre prácu so sériami je možné použiť metódy:
 - `getSerie(serieName : String) : Vector`
 - `addValueToSerie(serieName : String, value : Object) : void`
 - `resetSerie(serieName : String) : void`

Funkcie na prácu s poliami a sériami nie je možné volať na všetkých miestach v scriptlete, upresnenie možností volania týchto funkcií bude opísané ďalej. Výstupom `get` metód je trieda `Object` ktorú je potrebné pretypovať na typ deklarovaný v súbore definícii zostavy.

Okrem týchto metód je možné použiť atribút triedy `IReportScriptlet` „`isLoaded`“ ktorý indikuje fakt, že scriptlet bol už v rámci behu aplikácie spustený a uchováva si všetky nastavené hodnoty premenných, sérii a polí. V takom prípade nie je potrebné znova nastavovať ich hodnoty spomenutými funkciami. Atribút `isLoaded` je typu boolovská hodnota.

Okrem samotného nastavovania jednotlivých elementov scriptletu musí byť možné aj získavať dáta z databázy ku ktorej je aplikácia `FeXtat Browser` pripojená. Získavanie týchto dát je možné realizovať pomocou špeciálnej statickej metódy `org.fellas.flextat.main.FXConnect.getReference()`, ktorá vráti inštanciu triedy `FXConnect`. Volaním metódy `select(String query)` tejto inštancie je možné získať resultset daného SQL príkazu nad práve používanou databázou `FeXtat Browsera`. `ResultSet` je typu `java.sql.ResultSet`.

4.2 Obslužné udalosti scriptletu pri tvorbe zostavy

Okrem týchto metód sú v triede `IReportScriptlet` definované metódy začínajúce predponou „before“ alebo „after“. Tieto metódy sú volané ako obsluha pred alebo po špecifických udalostiach počas tvorby a napĺňania konečnej zostavy. Jedná sa o inicializáciu zostavy (`beforeReportInit`, `afterReportInit`), stránky (`beforePageInit`, `afterPageInit`), stĺpca tabuľky (`beforeColumnInit`, `afterColumnInit`), skupiny (`beforeGroupInit`, `afterGroupInit`) a vyhodnotenia konkrétneho riadku tabuľky (`beforeDetailEval`, `afterDetailEval`).

Pokiaľ chce vývojár implementovať scriptlet ktorý vykonáva špecifickú funkčnosť pri tvorbe zostáv je potrebné preťažiť minimálne jednu zo spomenutých obslužných metód. Spravidla sa používa iba preťažená metóda `beforeReportInit`, v ktorej sa naplnia všetky potrebné premenné, série a polia. Pri preťažovaní týchto metód je potrebné zachovať jedno pravidlo, že na začiatku každej preťaženej metódy „before“ je potrebné zavolať tú istú metódu predka a to isté je potrebné na konci každej preťaženej metódy „after“. Teda kód bude vypadať napríklad nasledovne:

```
void beforeReportInit()
{
    super.beforeReportInit();
    // vlastný kód
}

void afterReportInit()
{
    // vlastný kód
    super.afterReportInit();
}
```

Metódy pracujúce so sériami a metóda `setField` sú špeciálne metódy, ktoré je možné volať iba v rámci metódy `beforeReportInit`, pre ich volania mimo tejto metódy nie je zaručená korektnosť vykonaných operácií. Metódu `getFieldValue` sa podľa tvorcov `JasperReportu` odporúča používať pri metódach `beforeDetailEval` a `afterDetailEval`, kedy táto metóda sprístupní hodnotu položky aktuálneho riadka.

4.3 Vzor scriptletu

Na nasledujúcich riadkoch je uvedený vzor scriptletu.

```
package org.fellas.flextat.reportrepository.scriptlets;

import it.businesslogic.ireport.IReportScriptlet;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Vector;

public class ExampleScriptlet extends IReportScriptlet {

    public void beforeReportInit() throws JRScriptletException {
        super.beforeReportInit();
        if( !isLoading())
            getAllValues();
    }

    private void getAllValues() {
        Vector values = new Vector();
        Vector labels = new Vector();
        FXConnect connector = FXConnect.getReference();
        ResultSet resultSet;

        resultSet=connector.select("select ....");

        try {
            while(resultSet.next()) {
                // filling values and labels
            }
        } catch (SQLException e) {

            e.printStackTrace();
        }

        series.put("SERIE_values", values);
        series.put("SERIE_labels", labels);
    }
}
```

```
fields.put("Counts", values);  
fields.put("Schools", labels);  
}
```

5 Implementácia fleXPump

Aplikácia fleXPump je jednoduchá konzolová Java aplikácia pracujúca v dvoch fázach. V prvej fáze sa vykonáva presun dát z jednej databázy do druhej, konkrétne dáta z databázy aplikácie elektronickej prihlášky do databázy FlexDB. V niekoľkých opakujúcich blokoch sa vykonáva *SELECT* nad databázou elektronickej prihlášky. Hodnoty v *ResultSete* sa zapisujú štandardným *INSERTom* do tabuliek databázy FlexDB. V prvej časti napĺňania sa naplnia číselníky databázy, nakoniec sa napĺňa centrálna tabuľka *dat_stat*. Nasledujúci kód demonštruje príklad bloku vykonávajúceho *SELECT* a následný *INSERT*.

```
rs = executeQuery( "SELECT countries.id, countries.legacy_id FROM
                    countries ORDER BY countries.id");
while (rs.next())
{
    stmt2 = connection2.prepareStatement("INSERT INTO
                                         cis_citizenship VALUES(?,?)");
    stmt2.setInt(1,rs.getInt(1));
    stmt2.setString(2,rs.getString(2));
    stmt2.executeUpdate();
}
```

Operácie nad databázou neprebiehajú v transakcii z dôvodu problémov s implementáciou. Pred začiatkom procesu pumpovania sa volá funkcia, ktorá vykonáva vyprázdnenie všetkých tabuliek v databáze, aby sa predišlo možným duplikáciám.

V druhej fáze sa parsuje XML súbor jazykovej verzie aplikácie, pričom sa do tabuľky *cis_messages* v databáze FlexDB zapisujú hodnoty všetkých atribútov *key* z tohto XML súboru.

Inštrukcie k spusteniu a testovaniu aplikácie fleXPump je možné nájsť v kapitole 8.8.

6 Databázová časť systému FlexTat

6.1 Databázový server

Ako databázový server je využitý voľne dostupný SQL server PostgreSQL vo verzii 8.0.2, ktorá je funkčná aj na operačných systémoch Windows. Pretože podporuje aj také funkcie SQL servera ako triggre, pravidlá a vstavané funkcie, bola to logická voľba pre implementáciu databázovej vrstvy.

6.2 Fyzický model databáz

Na Obr. 5 je znázornený fyzický model databázy. Avšak ako je vidieť zo spomenutého modelu, je tento značne odlišný od navrhovaného fyzického modelu. Model je minimalizovaný pre potreby aplikácie flexTat a je v ňom pridaná funkcionálna verzionovania číselníkov v báze dát. Ústredným objektom je aj naďalej tabuľka „dat_stat“ kde sú vlastne uložené všetky dáta. Avšak na prístup k hodnotám z číselníkov sa nevyužívajú fyzické tabuľky, ale iba náhľady, ktoré sú všetky presmerovávané do tabuľky „cis_dat“. V predmetnej tabuľke sú uložené všetky verzie všetkých číselníkov (ako jednoznačný identifikátor slúži pole „id“, a ako identifikátor v náhľade slúži pole „iid“). Funkcia a význam jednotlivých polí tabuliek „cis_cis“ a „cis_dat“ je popísaný v časti Správanie databázovej vrstvy.

Tabuľka „cis_fields_mapping“ definuje mapovanie stĺpcov v tabuľke „cis_dat“ (čiže jednotlivých faktov) do aplikácie flexTat. Sú v nej definované názvy stĺpcov – faktov v tabuľke „cis_dat“, ako aj ich názvy tak ako sa zobrazujú používateľovi (polia name a caption). V poli type je uložený dátový typ, tak ako je reprezentovaný v jazyku Java a obsahom tohto pola je plné znenie názvu dátového typu (napríklad java.lang.String). To jednak slúži ako definícia názvov faktov v štatistikách, ale typovanie zároveň umožňuje aj ďalšie spracovávanie faktov (funkciami nad daným dátovým typom) v štatistikách, kde nie je presne uvedený typ faktu.

Tabuľka „cis_messages“ slúži na ukladanie hodnôt atribútov z parsovaného XML súboru. Mená atribútov z tohto súboru sú ukladané do pola id a slúžia ako jednoznačný identifikátor a zároveň kľúč pre hodnoty číselníkov s viacjazyčnou podporou, a ich hodnoty sú ukladané do pola „description“.

Do databázového modelu bola pridaná tabuľka „cis_county_cis_couty“, ktorá zabezpečuje previazanie krajov na jednotlivé okresy (takže je to jednoduchá väzobná tabuľka). Táto tabuľka obsahuje identifikátor okresu (v stĺpci „county_id“) a identifikátor kraja (v stĺpci „couty_id“), pričom obsahom týchto polí sú hodnoty stĺpcov „id“ z náhľadov príslušného číselníka. Číselník krajov je vytváraný rovnako ako ostatné pri inicializácii bázy dát, avšak na rozdiel od ostatných je tu aj naplňovaný dátami. Rovnako je dátami naplnená aj spomínaná väzobná tabuľka „cis_county_cis_couty“, pričom ako východzí bod bol braný obsah číselníka „cis_county“ v dobe odovzdávania aplikácie flexTat do prevádzky. Keďže kraje a ani

väzby okresov na kraje nie sú predmetom aplikácie STUDAPP, tak nie je možné importovať tieto údaje do aplikácie fleXtat. Z týchto dôvodov je nutné udržiavať číselník krajov „cis_couty“ (podporuje verzionovanie) a aj jeho väzobnú tabuľku (nepodporuje verzionovanie) v aktuálnom stave ručne administrátorom systému.

Poslednou opisovanou štruktúrou je spojenie dvoch tabuliek – „cis_tree“ a „cis_report“. V prvej zmienenej je uložená stromová štruktúra zoznamu štatistík (to čo poskytuje aplikácia v ľavej časti používateľského rozhrania). Stĺpec „id“ predstavuje jednoznačný identifikátor každého záznamu. Stromová štruktúra je zabezpečená previazaním stĺpca „upper_id“ na stĺpec „id“ v rovnakej tabuľke. Týmto sa docieli vytvorenie stromovej štruktúry. Názvy jednotlivých uzlov v strome sú načítané zo stĺpca „popis“. V prípade, že v danom uzle sa nachádza aj niektorá zo štatistík, tak jej jednoznačný identifikátor je zapísaný vo väzobnom stĺpci „report_id“, ktorý je spojený so stĺcom „id“ v tabuľke „cis_report“. Touto štruktúrou je zabezpečené, že jeden uzol môže obsahovať viacero štatistík, ale zároveň jedna štatistika sa môže nachádzať vo viacerých uzloch.

V tabuľke „cis_report“ sú teda uložené všetky potrebné údaje k samotnej štatistike. Okrem názvu (stĺpec „detail“) a jednoznačného mena štatistiky (stĺpec „report_name“) je tu uložených niekoľko špecifických dát potrebných pre konkrétnu štatistiku. Je tu samozrejme pre centralizovaný prístup uložená aj samotná štatistika (stĺpec „report_file“), ako aj potrebné scriptlety a ich definície pre konkrétnu štatistiku (aby boli scriptlety rovnako centralizovane prístupné z bázy dát). Nakoniec v stĺpci „ds_type“ je uložený typ zdroja dát štatistiky pre aplikáciu fleXtat.

K databázovému modelu treba poznamenať, že na spravovanie jednoznačných identifikátorov boli v databáze vytvorené sekvencie. Tieto sekvencie sú jedinečné pre každú tabuľku, ktorá vyžaduje ako svoj jednoznačný identifikátor hodnotu typu integer. Sekvencia zabezpečuje pridelenie jednoznačného identifikátora aj v prípade prístupu z viacerých bodov (simultánneho vyžiadania novej hodnoty), čo sa napríklad autoinkrementálnym číslom v definícii stĺpca nedá dosiahnuť.

Všetky vytvorené trigger funkcie sú obsahom elektronického média a inštalačného skriptu na vytvorenie databázy a preto, ak si chce používateľ naštudovať ich funkcionality, ich tam treba hľadať. Všetky funkcie pripravené ako databázová vrstva aplikácie fleXtat sú písané v jazyku PL/SQL (jeho mutácii pre SQL server PostgreSQL). Trigger funkcie pracujú ako nad tabuľkami ako riadkové funkcie, to znamená, že každý zo zmenených alebo spracovávaných riadkov definovaných SQL príkazom je spracovaný trigger funkciou samostatne. Preto je napríklad v databáze k dispozícii funkcia clear_database, ktorá vyčistí obsah číselníkov, miesto toho aby bolo vytvorených toľko verzií číselníka koľko je v ňom záznamov, pri vymazávaní všetkých záznamov funkciou „DELETE FROM cis_degree“.

Za pošímnutie stojí fakt, že v pravidlách, ale aj vo vložených procedúrach, sú prístupné dva databázové záznamy s názvami new a old. Pri vkladaní záznamov je v zázname new

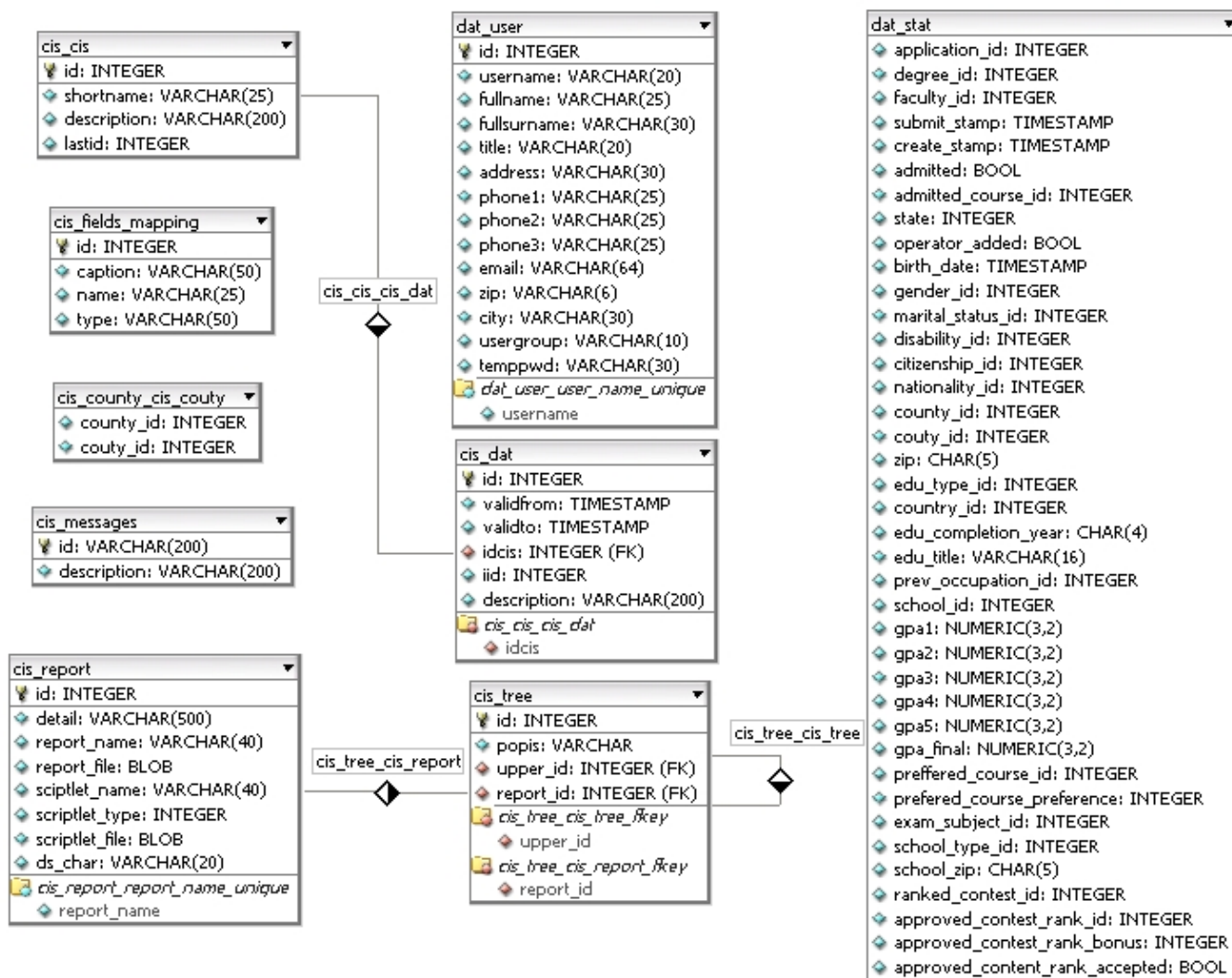
uložený nový záznam. Pri vymazávaní je mazaný záznam k dispozícii v zázname old, a v prípade zmeny je predchádzajúci stav uložený v old a nový v zázname new. Oba tieto záznamy majú rovnakú štruktúru ako tabuľka nad ktorou sa táto operácia vykonáva, takže vo vložených procedúrach sa pokojne môžeme odvolávať napríklad na pole new.id.

6.3 Správanie databázovej vrstvy

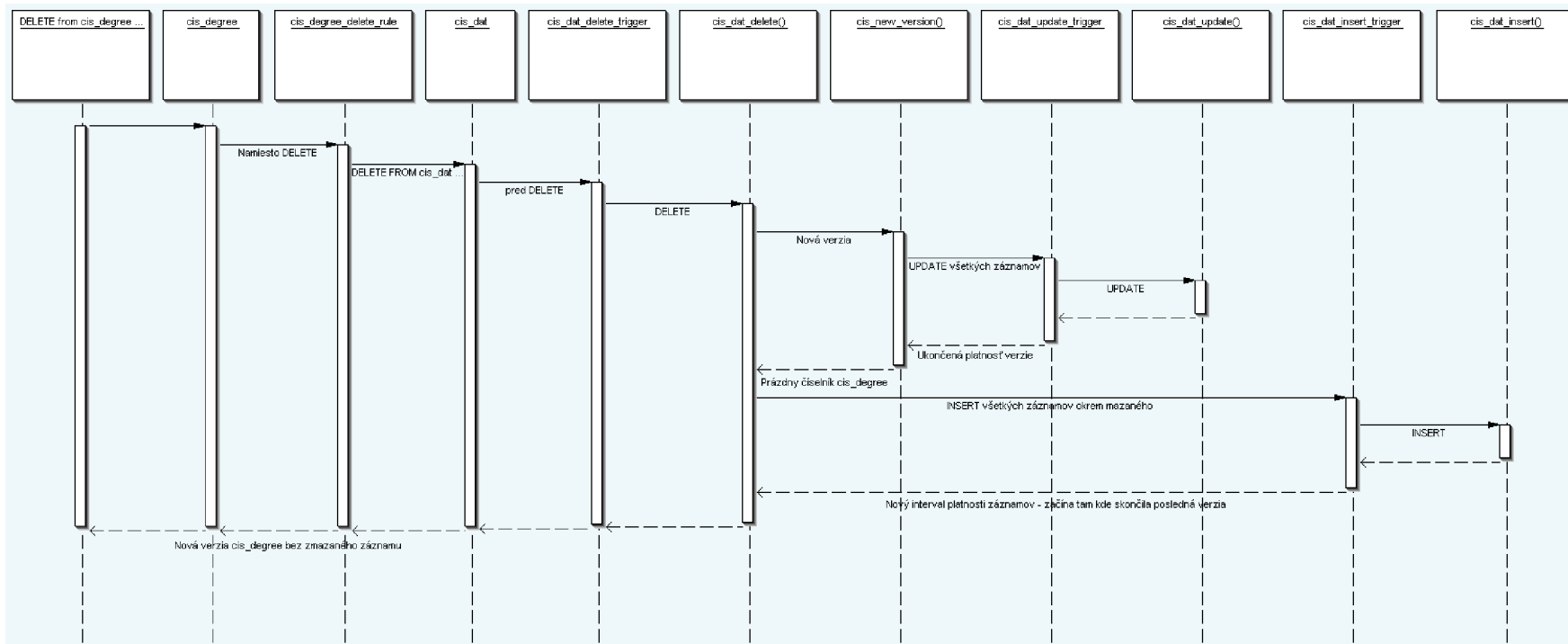
Logický model databázy sa však v prípade tabuliek dát a číselníkov v ničom nelíši od navrhovaného riešenia. Stále je ústredným prvkom databázy tabuľka (náhľad) dat_stat_applications. V nej sú uložené všetky fakty a to buď priamo, alebo ako odkaz do číselníka. Importovanie dát sa však nie je podporované samotným náhľadom „dat_stat_applications“, ale na to špeciálne určeným náhľadom „dat_stat_insert“. Náhľad „dat_stat_applications“ totižto zobrazuje priamo hodnoty všetkých faktov, to znamená aj tých, ktoré sú uložené v číselníkoch (táto funkcionality je implementovaná pre jednoduchšiu prácu s databázou zo strany aplikácie fleXtat). Preto je logickým dôsledkom vytvorenie nového náhľadu na importovanie dát do tabuľky „dat_stat_applications“ (fyzicky „dat_stat“). Tento náhľad umožňuje vkladanie záznamov s tým, že povoľuje vkladať ID hodnôt uložených v číselníku tak, ako to bolo navrhnuté v prototypu.

Teda nad fyzickou tabuľkou „dat_stat“ sú vytvorené dva náhľady, pričom náhľad „dat_stat_insert“ má priradené pravidlo na vkladanie záznamov („dat_stat_insert_insert_rule“). Spomenuté pravidlo rieši vkladanie ID hodnôt záznamov v číselníkoch, pričom automaticky ich transformuje na jedinečné identifikátory priradené tomuto ID v konkrétnom číselníku. (viac o prepojení v odstavci o tabuľke „cis_dat“). Toto pravidlo sa vykonáva nad dátami namiesto každého príkazu INSERT.

Číselníky faktov sú implementované ako náhľady nad tabuľkou „cis_dat“. Všetky takéto náhľady majú definované pravidlá pre všetky operácie nad dátami uloženými v číselníku, pričom pre lepšiu orientáciu vo verziách číselníkov, každý z náhľadov zobrazuje iba aktuálne verzie. Tak je zabezpečené, že pri vkladaní záznamu sa pracuje s najnovšou hodnotou uloženou v číselníku.



Obr. 5: Fyzický model databázy fleXtat



Obr. 6: Sekvencia volania funkcií v prípade zmazania záznamu z číselníka

Verzia číselníka je pritom časovo závislá a má stanovený interval platnosti. Predmetný interval je uložený v tabuľke „cis_dat“ v poliach validfrom a validto (definujú počiatočný čas a konečný čas intervalu platnosti verzie). Verzie jednotlivých číselníkov sú však plne nezávislé a tak každý z nich môže mať v platnosti inú verziu.

Na to, aby mohli byť všetky číselníky uložené v jednej centrálnej tabuľke, je potrebné aby táto tabuľka mala niekoľko špecifických polí navyše. Tieto polia definujú príslušnosť k danému číselníku – idcis, hodnotu id v číselníku – iid, hodnotu – description. Stĺpec id sa používa ako jednoznačný identifikátor záznamu a do dátovej tabuľky sú miesto hodnôt iid vkladané práve tieto jednoznačné identifikátory. Polia validfrom a validto definujú platnosť verzie tak ako to bolo spomenuté vyššie. Zvyšné hodnoty (všetky okrem iid a description), ktoré je treba doplniť pri vkladaní záznamu do číselníka (alebo pri identifikácii záznamu pri zmene) definujú pravidlá nad každým náhľadom na číselník, ktoré sa vykonávajú namiesto operácií (INSERT, UPDATE, DELETE). Tieto pravidlá a náhľad nad tabuľkou „cis_dat“ sú však nezávisle vytvárané pri vkladaní nového faktu do databázy.

Na zabezpečenie verzionovania číselníkov sú implementované triggre nad tabuľkou „cis_dat“. Triggre vlastne zabezpečujú vytvorenie novej verzie číselníka v prípade jeho zmeny (pod zmenou sa rozumie UPDATE, DELETE). V prípade vkladania záznamov do číselníka sú vkladané automaticky na koniec aktuálnej verzie (ID hodnoty bude vždy väčšie ako posledná vložená). Trigger pre INSERT teda vloží nový záznam do aktuálnej verzie číselníka, pričom nezávisle od danej hodnoty ID (ak je špecifikovaná v INSERT príkaze) ho vloží do číselníka. Duplicity sú v číselníkoch povolené, pretože niektoré číselníky nemajú jednoznačné hodnoty (táto skutočnosť vyplynula s testovacej vzorky dát aplikácie STUDAPP). Nový záznam tak bude mať definovaný popis a hodnotu ID podľa toho ktorá z dvoch hodnôt (ID vkladaneho záznamu, alebo ID posledného záznamu v číselníku je väčšia). Pozor, ak vložíte záznam s ID hodnotou 10000, všetky ďalšie záznamy budú mať ID vyššie a preto treba vkladať záznamy s vzostupne usporiadaným ID!

Na vkladanie nových faktov do bázy dát ako aj na vytváranie náhľadov nad tabuľkou „cis_dat“ boli vytvorené triggre nad tabuľkou „cis_cis“. V spomenutej tabuľke sú uložené parametre jednotlivých číselníkov. Stĺpec id slúži ako jednoznačný identifikátor číselníka. Jeho hodnota je v tabuľke „cis_dat“ využívaná ako referencia číselníka (definuje príslušnosť záznamu k číselníku). V ďalšom stĺpci „shortname“ je uložené skrátené meno číselníka, jednak sa toto meno používa pri vytváraní nového faktu a jednak slúži na referencovanie náhľadu. Pri vytváraní nového číselníka je toto meno použité a s prefixom „cis_“ tvorí meno náhľadu. Preto nesmie obsahovať medzery a iné znaky zakázané pri názvosloví tabuliek v SQL serveri. Description definuje popis číselníka a môže byť použitý pre referencovanie faktu (ako popis v aplikácii). Posledný stĺpec „lastid“ definuje vyššie spomenutý záznam posledne použitého ID v

číselníku. Je to preto, lebo ak sa záznam z číselníka vymaže, tak aby pri vložení nového záznamu do číselníka s rovnakým id nevznikla nekonzistencia dát.

Nad operáciami v tabuľke „cis_cis“ sú opätovne definované triggre. Tie majú za úlohu správu a vytváranie náhľadov a pravidiel nad tabuľkou „cis_dat“. Pri vložení nového záznamu do tabuľky „cis_cis“, čo vlastne znamená vloženie nového faktu, sa vytvorí nad tabuľkou „cis_dat“ náhľad s číselníkom reprezentujúcim nový fakt. Rovnako budú k tomuto náhľadu automaticky vytvorené pravidlá, ktoré budú obsluhovať dátové operácie v tomto náhľade (INSERT, UPDATE, DELETE). Pre lepšiu orientáciu v poradí volania jednotlivých funkcií a spúšťania triggrov je na obrázku Obr. 6 znázornená sekvencia volania databázových objektov v prípade vymazávania záznamu z číselníka „cis_degree“ (príkaz „DELETE FROM cis_degree WHERE id=1“). Sekvencie volaní sa v prípade iných číselníkov v ničom nelíšia, a v prípade volaní iných operácií nad dátami sú veľmi obdobné.

Rovnako sa zmien dotklo aj riešenie manažmentu používateľov. Pre väčšiu transparentnosť boli urobené zmeny v kontrole prihlasovanie používateľov. V tejto verzii aplikácie je prihlasovanie delegované na databázový server, ktorý si sám udržiava používateľov a ich prihlasovacie práva. Na vkladanie nových používateľov ako aj na ich zmenu je v báze dát použitá tabuľka „dat_user“. V tejto tabuľke sú uložené jedinečné používateľské mená a niekoľko ďalších viac, alebo menej potrebných údajov. Okrem používateľského mena sú vyžadované iba dva stĺpce a to skupina používateľov (users, editors alebo admins) a heslo používateľa (temppwd, ktoré sa však do databázy neukladá a políško slúži iba ako referencia). Tieto parametre sú využité triggrami pracujúcimi nad tabuľkou „dat_user“, ktoré vytvárajú, alebo menia, databázového používateľa v definovanej databázovej skupine používateľov. Skupiny používateľov sú pritom vytvárané v DDL skripte na inicializáciu databázových objektov. Podľa skupín používateľov sú potom aj riadené prístupové práva na jednotlivé objekty v databáze. V prípade zmeny používateľa (jeho meno, alebo heslo, prípadne zaradenie do skupiny) sa spomínané triggre postarajú o potrebný manažment používateľa na strane databázového servera. Avšak všetky uvedené operácie aplikácia nevykonáva nad spomínanou tabuľkou, ale nad náhľadom „dat_users“, ktorý transformuje stĺpec „temppwd“ z tabuľky do stĺpca „userpwd“ v náhľade.

Všetky ostatné tabuľky sú prístupné tak ako sú definované v databáze. Jedinou výnimkou z nich je tabuľka „cis_report“, kedy sa všetky referencie vymazaného záznamu vymažú aj v tabuľke „cis_tree“, aby nedochádzalo k referencovaniu vymazanej štatistiky.

6.4 Popis vložených funkcií

V tejto časti sú opísané vložené funkcie, ktoré boli vytvorené ako pomocné operácie nad databázou. Operácie sa vykonávajú atomicky a pracujú ako jedna transakcia (keďže sa spúšťajú ako jedna atomická funkcia).

6.4.1 get_next_id

Vstup: typ int4 (id číselníka)

Výstup: typ int4 (hodnota nasledujúceho id v číselníku)

Funkcia vracia logicky nasledujúcu hodnotu ID pre daný číselník. ID číselníka je vstupom pre túto funkciu, pričom ID číselníka je uložené v tabuľke „cis_cis“ v poli ID. Funkcia porovnáva dve hodnoty navzájom (hodnotu aktuálneho záznamu v číselníku s najvyšším ID a hodnotu nasledujúceho ID, ktoré je zapísané v tabuľke číselníkov). Funkcia vracia hodnotu o jednotku vyššiu ako bolo väčšie z porovnávaných čísel. Toto vrátené číslo nakoniec zapíše do tabuľky „cis_cis“ ako posledné použité ID pre daný číselník. Touto funkciou sa zabezpečuje, že nové záznamy v číselníku nebudú mať hodnotu ID rovnakú ako iný záznam v predchádzajúcej verzii číselníka.

6.4.2 cis_new_version

Vstup: typ int4 (id číselníka)

Výstup: timestamp (časová známka, od kedy je platný nový číselník)

Funkcia vytvorí v databáze nasledujúcu verziu číselníka. Táto verzia je platná od okamžiku skončenia platnosti verzie predchádzajúcej. V prípade, že práve platný číselník nemá zhora ohraničenú dobu platnosti (validto je hodnoty NULL), tak sa nastaví platnosť do okamžiku vytvorenia novej verzie. Nová verzia je teda platná od tohto okamžiku. Funkcia sa využíva hlavne pri volaní triggera spustenom pri vymazávaní údajov z číselníka, kedy je potrebné vytvoriť identickú kópiu, s ktorou sa bude pracovať. Keďže aplikácia flexTat nepodporuje zhora ohraničené číselníky (a teda doprednú históriu), aj keď je na ňu plne pripravená, v tejto funkcii by bolo nutné dopracovať vytvorenie takej novej verzie, ktorá by rozdelila platný interval, pričom okamžik delenia by sa zhodoval s okamžikom vytvorenia novej verzie.

6.4.3 clear_database

Vstup: bez parametra

Výstup: boolean (true ak bola vykonaná bez chýb)

Táto funkcia je pripravovaná pre vkladanie nových skupín záznamov do databázy. Nové skupiny sú myslené ako plné importy dát zo systému STUDAPP. Pri takomto importe je potrebné nanovo vložiť obsahy jednotlivých číselníkov a aj dátovej tabuľky. Táto funkcia preto zmení aktuálne verzie číselníkov a nastaví im rozsah platnosti do momentu spustenia tejto funkcie (okrem číselníka krajov, ktorý nie je spravovaný aplikáciou). Tak budú nové číselníky platné od tohto momentu. Rovnako nastaví posledné použité ID v číselníku na hodnotu 0, aby mohli byť importované všetky číselníky zrkadlovo oproti systému STUDAPP. Posledným

krokom je vymazanie všetkých hodnôt z tabuľky „cis_messages“ (u nej sa neudržiava história a spätná kompatibilita je vecou správcu systému STUDAPP), kde sa ukladajú hodnoty parametrov z XML súboru.

6.5 Vytvorenie databázy

Databáza sa vytvára priloženým skriptom na vytvorenie databázy. Skript má názov „flectat_create_database.sql“ a obsahuje DDL skripty na vytvorenie a inicializáciu prvkov databázy. Predpokladom pre sustenie tohto skriptu je existencia novej, čistej databázy s kódovaním „UNICODE“. Tento skript dá databáze inicializovaný stav, kedy sú vytvorené všetky potrebné objekty a naplnené iba tabuľky číselníkov, tabuľka mapovania stĺpcov a východzí stav tabuľky krajov. Tabuľka krajov vychádza z verzie číselníka, ktorá bola dostupná v čase vytvorenia aplikácie a jej údržba nie je predmetom žiadnych skriptov ani nástrojov. Túto tabuľku je teda potrebné udržiavať ručne.

6.6 Importovanie údajov

Údaje sa, do databázy fleXtat, importujú priloženou pumpou fleXpump. Táto pumpa transformuje údaje z aplikácie STUDAPP a prevádza ich automaticky do formátu vyžadovaného aplikáciou. Okrem toho importuje aj .XML súbor, ktorý obsahuje jazykové mutácie niektorých číselníkov. Tento XML súbor je spracovaný pumpou a všetky jeho parametre spolu s hodnotami sú uložené do príslušnej tabuľky údajov. Pri zmene jazykovej mutácie databázy je teda potrebné nainportovať nanovo súbor XML s adekvátnym prekladom. Je tu potrebné ešte zdôrazniť fakt, že importované číselníky musia byť usporiadané podľa hodnoty ID vzostupne (podmienka z funkcie get_next_id).

6.7 Pridávanie faktov do databázy

Pridávanie nových faktov do existujúcej databázy rieši metodika pre pridávanie nových faktov. V prípade, že je používateľ dostatočne zručný, môže rovnaké úpravy zapracovať aj do skriptu pre vytváranie databázy. Pred akoukoľvek zmenou v databáze však odporúčame urobiť úplnú zálohu databázy, aby bola prístupná v prípade zlyhania, alebo chyby pri prebiehajúcej zmene v báze dát.

7 Metodika pridávania faktov do systému FleXtat

Táto kapitola popisuje metodiku na pridávanie nových faktov do existujúceho systému fleXtat. V metodike opíšem potrebné kroky na vytvorenie nového faktu na voľne dostupnom a použitom databázovom serveri PostgreSQL.

7.1 Pojmy

- Náhľad – (view) definuje pohľad na dáta uložené v tabuľke
- Postgres, PostgreSQL – voľne dostupný databázový server
- Pravidlo – (RULE) špeciálna forma SQL procedúry, ktorá pozná iba príkazy na manipuláciu s dátami. Pravidlá sa vykonávajú pred alebo po vykonaní SQL príkazu nad tabuľkou.
- Riadok – jeden záznam v databázovej tabuľke
- Schéma – Náhľad nad databázou. Väčšinou sa používa na obmedzenie prístupu skupiny používateľov iba k určitým objektom databázy
- Tabuľka – databázová tabuľka
- Trigger – špeciálna akcia SQL servera vykonávaná pri volaniach SQL príkazov pracujúcich so záznamami (SELECT, INSERT, UPDATE, DELETE)
- Vložená funkcia – funkcia napísaná v jazyku SQL, PL/SQL alebo C a uložená priamo na databázovom serveri

7.2 Vytvorenie potrebných objektov, a zmena existujúcich, pre pridanie nového faktu v systéme FleXtat

V tejto kapitole je popísaný kompletný postup pridania nového faktu do databázy. Keďže sa v prípade aplikácie fleXtat jedná o ROLAP databázovú štruktúru, je potrebné do hlavnej tabuľky pridať pole na uchovávanie faktu a vytvoriť číselník s verzionovaním pre uchovávanie hodnôt faktov. Predpokladom pre použitie tohto návodu je vytvorená databáza aplikácie fleXtat a privilégia používateľa na správu danej databázy. Na prácu s databázovým serverom bol použitý dodávaný GUI nástroj pgAdmin III.

V prípade využívania PostgreSQL využívajte pri názvoch tabuliek, polí a premenných iba malé písmená. Miesto oddelovania veľkým písmenom radšej použite znak „_“. Vyhnite sa tak problémom pri spätnej práci s takýmto objektom – ak použijete iné znaky ako [a-z0-9_-] budete na odvolávanie musieť používať úvodzovky (") pred a za takýmto parametrom.

7.2.1 Vytvorenie nového číselníka v databáze aplikácie

- Vstup:** Požiadavka na vytvorenie číselníka nového faktu v databáze
- Výstup:** DB náhľad reprezentujúci číselník s verzionovaním

1. Pripojte sa do danej databázy. Pri pripájaní musíte použiť konto s administrátorskými privilégiami v databáze fleXtat. V prípade, že pri pripojení cez pgAdmin máte definovaného používateľa s nižšími právami, musíte zmeniť používateľské meno.
 - a. Pravé tlačítko na databázovú inštanciu -> odpojiť.
 - b. Pravé tlačítko na databázovú inštanciu -> vlastnosti
 - c. Zmeňte používateľské meno
 - d. Ľavý dvojklik na inštanciu (pripojí sa na inštanciu) a zadajte heslo pre daného používateľa
2. V prípade, že fakt bude mať údaje zapísané priamo v tabuľke údajov a nie v číselníku preskočte nasledujúce kroky a pokračujte na podkapitole 3.2.
3. Spustíte si dotazový nástroj pre spúšťanie SQL príkazov nad databázou.
 - a. Menu aplikácie pgAdmin III Nástroje -> Dotazový nástroj
4. Do otvoreného dotazovacieho nástroja zadajte nasledovný SQL príkaz, pričom dodržujte nasledovné zásady:
 - a. meno_faktu je skrátený názov, ktorý sa použije pri vytvorení náhľadu. Nový náhľad bude mať meno cis_meno_faktu. Preto nepoužívajte medzery a veľké písmená pre definovanie mena faktu (platia rovnaké zásady ako pri vytváraní mena tabuľky)
 - b. popis faktu je ľubovoľný popis v maximálnej dĺžke 200 znakov.

```
INSERT INTO cis_cis (shortname, description) VALUES ('meno_faktu','popis faktu');
```
5. Na vykonanie dotazu stlačte F5, alebo v menu Dotaz -> Vykonať
6. Po úspešnom vykonaní dotazu je v databáze vytvorený nový verzionovaný číselník faktu – pozor – nehľadajte ho medzi tabuľkami, ale medzi náhľadmi.

7.2.2 Pridanie faktu do tabuľky údajov

Vstup: Databázová tabuľka reprezentujúca údaje

Výstup: Databázová tabuľka reprezentujúca údaje s pridaným novým faktom

1. Na pridanie nového faktu do tabuľky údajov potrebujeme zmeniť existujúcu tabuľku údajov. Táto tabuľka sa nachádza medzi fyzickými tabuľkami a má meno dat_stat.
 - a. V ľavej časti v stromovej štruktúre rozbaľte schému public. Nájdite podstrom tabuľky a rovnako ho rozbaľte. Nájdite tabuľku dat_stat. Na tejto tabuľke pravým tlačidlom myši na Vlastnosti.
 - b. Prepnite sa na záložku Stĺpce, a stlačte tlačidlo Pridať.
 - c. iDo otvoreného okna vpíšte všetky potrebné parametre. Pre lepšiu orientáciu doporučujem použiť Meno stĺpca rovnaké, ako bolo zadané v

predchádzajúcom kroku pre meno faktu + doplniť na koniec „_id“ aby bolo vidieť, že je z číselníka. („meno_faktu_id“). Pokiaľ máte údaje uložené priamo použite iba meno faktu bez udaného suffixu („meno_faktu“).

- d. Dátový typ zadajte podľa hodnoty, ktorú potrebujete uchovávať pre tento nový fakt.
 - i. Ak budete uchovávať hodnoty v číselníku zadajte int4.
 - ii. Ak budete uchovávať hodnoty priamo definujte priamo jej dátový typ.
 - e. V prípade, že ešte nie sú v databáze naplnené žiadne údaje a ste si istý, že každý záznam v dátach bude mať definovanú nejakú hodnotu pre daný fakt (NULL sa nepovažuje za hodnotu), môžete definovať stĺpec ako Not NULL. V prípade, že už dáta v databáze sú, musíte nechať stĺpec s možnosťou Not NULL neaktivovanou.
 - f. Vytvorenie stĺpca potvrdíte tlačidlom OK.
2. Zmenu dátovej tabuľky potvrdíte stlačením tlačidla OK.
 - 3.

7.2.3 Pridanie nového faktu do náhľadu pre zobrazovanie dát

Vstup: Náhľad dat_stat_applications nad tabuľkou dat_stat

Výstup: Náhľad dat_stat_applications nad tabuľkou dat_stat s pridaným novým faktom

V tomto kroku je potrebné uvažovať s tromi variantami pridania faktu pre zobrazovanie. Pretože je v prípade niekoľkých faktov využitá viacjazyčná podpora pomocou tabuľky jazykových mutácií budeme uvažovať pridanie nového faktu aj s viacjazyčnou podporou (nutnosť definovania hodnôt vo vstupnom XML), aj priame hodnoty. Problémom pri editácii náhľadov je to, že nie sú editovateľné (čo sa týka zmeny počtu stĺpcov) a je potrebné ich vytvoriť nanovo.

1. V ľavej časti obrazovky rozbaľte podstrom Náhľady. Pravé tlačidlo myši na náhľad dat_stat_applications a zvolte Vlastnosti.
2. Prepnete sa na záložku Definícia. Celú definíciu náhľadu si odložte do schránky (Clipboard).
3. Zrušte editáciu náhľadu tlačidlom Zrušiť. Teraz je potrebné tento náhľad zmazať. Pravé tlačidlo na náhľad „dat_stat_applications“ a stlačte Zmazať/Zrušiť.

4. Po jeho úspešnom zrušení musíte vytvoriť tento náhľad nanovo. Pravé tlačidlo myši na Náhlady a potvrdíte Nový Náhľad.
5. Meno náhľadu bude „dat_stat_applications“. Prepnete sa na záložku Definícia a do nej vložte obsah schránky, ktorý ste si predtým odložili.
6. Presuňte sa kurzorom úplne na koniec definície náhľadu a začnite editovať presne pred „FROM dat_stat;“.
7. Pridanie nového faktu
 - a. s viacjazyčnou podporou. Pred „FROM“ dajte čiarku a doplňte za ňu nasledujúci príkaz:

```
(SELECT cis_messages.description FROM cis_messages WHERE cis_messages.id = ((( SELECT cis_dat.description FROM cis_dat WHERE cis_dat.id = dat_stat.meno_faktu_id)))) AS meno_faktu_id
```
 - b. s priamymi hodnotami z číselníka. Pred „FROM“ dajte čiarku a doplňte za ňu nasledujúci príkaz:

```
(SELECT cis_dat.description FROM cis_dat WHERE cis_dat.id = dat_stat.meno_faktu_id) AS meno_faktu_id
```
 - c. s priamymi hodnotami z dátovej tabuľky. Pred „FROM“ dajte čiarku a doplňte za ňu nasledujúci príkaz:

```
dat_stat.meno_faktu
```
8. Pre vytvorený náhľad nastavte nasledovné bezpečnostné nastavenia. V záložke Privilégiá nastavte:
 - a. 1. skupina: group users, privilégiá: SELECT, RULE
 - b. 2. skupina: group editors, privilégiá: SELECT, RULE
 - c. 3. skupina: group admins, privilégiá: SELECT, RULE
9. Vytvorenie náhľadu potvrdíte tlačidlom OK.

7.2.4 Pridanie faktu do náhľadu pre vkladanie dát

Vstup: Náhľad dat_stat_insert nad tabuľkou dat_stat

Výstup: Náhľad dat_stat_applications nad tabuľkou dat_stat s pridaným novým faktom

1. V ľavej časti obrazovky rozbaľte podstrom Náhlady. Rozbaľte si náhľad a vyberte pravidlá. Pravé tlačidlo myši na pravidlo „dat_stat_insert_insert_rule“ a zvolte Vlastnosti.
2. Prepnete sa na záložku Definícia. Celú definíciu pravidla si odložte do schránky (Clipboard). Preneste si túto definíciu napríklad do aplikácie Notepad. (Je potrebné si túto definíciu náhľadu bezpečne odložiť zo schránky!).
3. Práve tlačidlo myši na náhľad „dat_stat_insert“ a zvolte Vlastnosti.
4. Prepnete sa na záložku Definícia. Celú definíciu náhľadu si odložte do schránky (Clipboard).

5. Zrušte editáciu náhľadu tlačidlom Zrušiť. Teraz je potrebné tento náhľad zmazať. Pravé tlačidlo na náhľad „dat_stat_insert“ a stlačte Zmazať/Zrušiť.
6. Po jeho úspešnom zrušení musíte vytvoriť tento náhľad nanovo. Pravé tlačidlo myši na Náhlady a potvrdíte Nový Náhľad.
7. Meno náhľadu bude „dat_stat_insert“. Prepnite sa na záložku Definícia a do nej vložte obsah schránky, ktorý ste si odložili v kroku 4.
8. Presuňte sa kurzorom úplne na koniec definície náhľadu a začnite editovať presne pred „FROM dat_stat;“.
 - a. Pridanie nového faktu s hodnotami z číselníka. Pred „FROM“ dajte čiarku a doplňte za ňu nasledujúci príkaz: (SELECT cis_dat.iid FROM cis_dat WHERE cis_dat.id = dat_stat.meno_faktu_id) AS a meno_faktu_id
 - b. Pridanie nového faktu s priamymi hodnotami z dátovej tabuľky. Pred „FROM“ dajte čiarku a doplňte za ňu nasledujúci príkaz: dat_stat.meno_faktu
9. Pre vytvorený náhľad nastavte nasledovné bezpečnostné nastavenia. V záložke Privilégiá nastavte:
 - a. 1.skupina: group users, privilégiá: SELECT, RULE
 - b. 2.skupina: group editors, privilégiá: SELECT, RULE
 - c. 3.skupina: group admins, privilégiá: SELECT, INSERT, DELETE, UPDATE, RULE
10. Vytvorenie náhľadu potvrdíte tlačidlom OK.
11. Pravé tlačidlo myši na vytvorený náhľad „dat_stat_insert“ a zvolíte Nový Objekt -> Nové Pravidlo.
12. Meno pravidla bude „dat_stat_insert_insert_rule“ a bude mať zvolené parametre „Vykonať namiesto“ na aktívne a udalosť bude INSERT.
13. Prepnite sa na záložku Definícia a do nej vložte definíciu pravidla, ktorú ste si odložili v kroku 2.
14. Presuňte sa kurzorom úplne na koniec prvého riadku definície náhľadu a začnite editovať presne pred znakom „)“.
15. Za posledný fakt dajte čiarku a vložte za ňu pridaný fakt.
 - a. V prípade, že máte údaje uložené v číselníku použite meno „meno_faktu_id“
 - b. V prípade, že máte údaje uložené priamo v tabuľke údajov použite meno „meno_faktu“
16. Presuňte sa kurzorom úplne na koniec definície náhľadu a začnite editovať presne pred poslednou zátvorkou „)“;“. Vložte ako oddeľovač čiarku a naľadovne:
 - a. Pre vloženie faktu, ktorý čerpá údaje z číselníka zadajte nasledovný príkaz: (SELECT cis_dat.id FROM cis_dat WHERE cis_dat.iid =

```
new.meno_faktu_id AND cis_dat.idcis = (( SELECT cis_cis.id FROM
cis_cis WHERE cis_cis.shortname = 'meno_faktu')) AND
cis_dat.validfrom < current_timestamp AND (cis_dat.validto IS NULL OR
cis_dat.validto > current_timestamp))
```

- b. Pre vloženie faktu, ktorý má údaje priamo v tabuľke vložte:
new.meno_faktu

17. Vytvorenie pravidla potvrdíte tlačidlom OK.

7.2.5 Pridanie faktu do tabuľky mapovania polí pre štatistiky

Vstup: Fakt pre zobrazovanie v štatistikách

Výstup: Upravená tabuľka cis_fields_mappings (pridaný fakt pre zobrazovanie v štatistikách)

Na to, aby ste mohli správne využívať pridaný fakt v štatistikách, je potrebné, aby ste ho pridali do tabuľky mapovania polí. Táto tabuľka zabezpečuje korektné prepojenie do štatistik a zároveň definuje dátový typ faktu pre aplikáciu fleXtat.

1. Spustíte si dotazový nástroj pre spúšťanie SQL príkazov nad databázou.
2. Do otvoreného dotazovacieho nástroja zadajte nasledovný SQL príkaz, pričom dodržujte nasledovné zásady:
 - a. Zobrazené_meno je meno, ktoré sa bude zobrazovať používateľovi aplikácie fleXtat pri využívaní faktu v štatistikách. Meno_faktu je názov, ktorý ste použili v dátovej tabuľke, a dátový_typ je typ údajov v Jave (plné meno typu), ktoré reprezentuje typ údajov. Ak máte dáta v číselníku dátový typ je vždy „java.lang.String“.
 - b. V prípade, že máte dáta mapované z číselníka použite tento SQL príkaz:
INSERT INTO cis_fields_mapping VALUES (default, 'zobrazené_meno', 'meno_faktu_id', 'java.lang.String');
 - c. V prípade, že máte dáta uložené priamo použite tento SQL príkaz:
INSERT INTO cis_fields_mapping VALUES (default, 'zobrazené_meno', 'meno_faktu', 'dátový_typ');
3. Na vykonanie dotazu stlačte F5, alebo v menu Dotaz -> Vykonať
4. Po úspešnom vykonaní dotazu je v databáze vytvorený záznam pre mapovanie nového faktu v aplikácii.

7.2.6 Úprava nástroja fleXPump

Na to, aby bolo zaručené aj korektné napĺňanie databázy v súlade s novým (pridaným faktom) je potrebné zmeniť aj nástroj fleXpump. Táto zmena spočíva v pridaní napĺňania hodnôt nového faktu, prípadne aj pridanie napĺňania číselníka.

8 Akceptačné testy

Táto kapitola obsahuje popis akceptačných testov s postupom a výsledkom alfa testovania systému fleXtat povereným členom tímu uskutočnenom vo finálnej fáze vývoja aplikácie. Testy boli vytvorené z návrhu hlavných funkcionality aplikácie popísaných v dokumentácii za zimný semester v kapitole funkčnej špecifikácie.

8.1 Všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat

Pre používanie a testovanie aplikácie fleXtat musí používateľ aplikácie spĺňať nasledujúce¹ podmienky:

- odporúčaná konfigurácia pre spustenie aplikácie fleXtat: Intel Pentium 3 600 MHz resp. ekvivalent od AMD, 256 MB RAM, 20 MB voľného miesta na disku, myš
- nainštalovaný Java Runtime Environment 1.4.2_07
- fyzické pripojenie do počítačovej siete s logickým prístupom k databáze fleXDB
- úspešné vykonanie krokov inštalácie popísaných v používateľskej príručke v kapitole 3.

8.2 Testovanie prehliadania zostáv

Účel testovania:

- Otestovanie funkčnosti jednej z kľúčových funkcionality aplikácie – prehliadania zostáv.

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (viď. 8.1)
- Používateľ musí mať oprávnenia na prihlásenie sa do aplikácie v roli editora alebo referenta
- V databáze musí byť uložený záznam aspoň o jednej zostave

Postup testovania:

- Prihláste sa do aplikácie
- V ľavej časti okna aplikácie v strome zostáv rozbalte² kategóriu zostáv, v ktorej sa nachádza zostava, ktorú si želáte prezrieť
- Kliknite na položku zostavy, ktorú si želáte prezrieť

Očakávaný výsledok:

¹ V prípade, že pre testovanie niektorej funkcionality nie je potrebné splniť niektorú y týchto podmienok, bude tento fakt uvedený v kapitole testovacieho scenára pre túto funkcionality.

² realizuje sa kliknutím na plošku v strome

- V strednej časti okna aplikácie sa zobrazí želaná zostava.
- V ľavej časti okna aplikácie sa v tabuľke vlastnosti zobrazia vlastnosti zostavy, ktoré je možné meniť³.
- Informácie v zostave zodpovedajú informáciám v databáze FlexDB a spôsobu ich spracovania

Výsledok testovania:

Tento testovací scenár je podmnožinou resp. konečnou fázou iných testovacích scenárov popísaných v tejto kapitole. Testovanie zobrazovania zostáv prebehlo po každom vytvorení novej zostavy resp. modifikovaní existujúcej zostavy, či už sa jednalo o zostavu zobrazovanú pomocou scripletu alebo zostavu generovanú modulom flexTat report wizard.

- Výsledok testovania sa zhoduje s očakávaným výsledkom

8.3 Testovanie úprav vlastností zostáv

Účel testovania:

- Otestovanie funkcionality zobrazenia vlasností jednotlivých polí zostavy v tabuľke v pravej časti okna aplikácie
- Otestovanie možnosti modifikácie týchto vlastností.

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie flexTat (viď. 8.1)
- Používateľ musí mať umožnené prehliadanie zostáv (viď.8.2)

Postup testovania:

- Zvoľte zostavu na prehliadanie (viď. 8.2)
- V tabuľke vlastností v pravej časti okna aplikácie kliknutím na riadok vlastnosti zvoľte vlastnosť, ktorú si želáte modifikovať.
- Upravte hodnotu vlastnosti a kliknite na tlačidlo obnovenia zostavy.
- V prípade, že chcete modifikovať vizuálnu vlastnosť zostavy (farba grafov, typ a veľkosť písma a pod.), kliknite ľavým tlačidlom myši do oblasti, ktorej vlastnosti si želáte modifikovať.
- V tabuľke vlastností sa zobrazia vlastnosti poľa, ktoré je možné upraviť.
- Kliknite na riadok vlastnosti, ktorú si želáte upraviť. V závislosti od typu vlastnosti sa zobrazia pomocné dialógové okná pre úpravu vlastností (napr. pri zmene farby písma alebo grafu sa zobrazí okno s farebnou paletou)
- Po úprave vlastnosti kliknite na tlačidlo obnovenia zostavy.

Očakávaný výsledok:

- Výberom zostavy sa naplní tabuľka vlasností zostavy, ktoré je možné modifikovať.

³ bližšie opísané v kapitole 8.3

- Kliknutím na niektorú oblasť v zostave, sa tabuľka vlastností naplní údajmi o vlastnostiach, ktoré je možné v tejto oblasti modifikovať.
- Po úprave niektorej vlastnosti ostane táto zmena zapísaná v tabuľke vlastností.
- Zapísané zmeny sa prejavia kliknutím na tlačidlo obnovenia zostavy.

Výsledok testovania:

- V zostavách sa podarilo úspešne modifikovať všetky vizuálne vlastnosti zobrazených zostáv.
- V niektorých preddefinovaných zostavách nie je doimplementovaná možnosť modifikácie špecifických vlastností zostavy⁴.

8.4 Testovanie tlače zostáv

Aplikácia fleXtat browser používa na zobrazovanie, tlač a export zostáv modul z aplikácie iReport, JasperPreview. Funkcionalita exportu a tlače zostáv v aplikácii fleXtat browser teda závisí od funkčnosti tohto modulu, ktorý nebok implementovaný tímom TheFellas.org .

Účel testovania:

- Otestovanie funkcionality vytlačenia prehľadanej zostavy priamo⁵ z aplikácie fleXtat browser.

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (vid'. 8.1)
- Používateľ musí mať umožnené prehliadanie zostáv (vid'. 8.2)
- Počítač musí byť pripojený priamo alebo cez sieť k funkčnej tlačiarni.
- Tlačiareň musí mať dostatok čistého papiera

Postup testovania:

- Zvoľte zostavu na prehliadanie (vid'. 8.2)
- Kliknite na tlačidlo tlače zostavy

Očakávaný výsledok:

- Vytlačí sa zostava identická zostave v aplikácii fleXtat browser

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom. Z aplikácie fleXtat browser je možné priamo vytlačiť prehliadané zostavy.

⁴ Konkrétnejšie v kapitolách

⁵ Zostavu je možné exportovať do formátu pdf a tlačiť v ľubovoľnej inej aplikácii schopnej zobrazovať a tlačiť súbory v tomto formáte.

8.5 Testovanie exportu zostáv

Účel testovania:

- Otestovanie funkcionality exportu zostavy do rôznych iných elektronických formátov

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (vid'. 8.1)
- Používateľ musí mať umožnené prehliadanie zostáv (vid'. 8.2)

Postup testovania:

- Zvoľte zostavu na prehliadanie (vid'. 8.2)
- Kliknite na tlačidlo exportu zostavy
- V zobrazenom okne vyberte formát, do ktorého chcete exportovať zostavu a pomenujte súbor, do ktorého sa má zostava exportovať.
- Potvrďte tlačidlom OK

Očakávaný výsledok:

- Vytvorí sa súbor s menom a typom zadaným v dialógovom okne exportu
- Obsah súboru je zostava exportovaná do tohto formátu a zodpovedá zostave zobrazenej v aplikácii fleXtat browser

Výsledok testovania:

- Export do formátov pdf a xml prebehol úspešne a výsledok testu je zhodný s očakávaným výsledkom.
- Export do ostatných formátov neprebehol úspešne v dôsledku chybné implementácie modulu JasperPreview, ktorý je časťou aplikácie iReport. Export je možné uskutočniť, no generovaná zostava sa nezhoduje so zostavou zobrazenou v aplikácii fleXtat browser

8.6 Testovanie vytvárania zostáv

Nové zostavy je oprávnený vytvárať len používateľ v roli editora. Na vytvorenie novej zostavy je možné použiť modul fleXtat report wizard spustiteľný ako plugin v rámci aplikácie iReport. Tento modul bol navrhnutý a implementovaný ako súčasť aplikácie fleXtat. V prípade, že funkcionality modulu nie je postačujúca pre vytvorenie zostavy napr. nad dátami je potrebné vykonať zložitejšie matematické operácie, je možné vytvoriť business logiku tohto procesu implementovať do scriptletov v spolupráci s programátorom ovládajúcim programovací jazyk Java.

8.6.1 Vytváranie zostáv pomocou modulu fleXtat report wizard

Účel testovania:

- Otestovanie funkcionality modulu fleXtat report wizard.

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (viď. 8.1)
- Používateľ musí mať nainštalovanú a spustenú aplikáciu iReport doplnenú o patche, ktoré sú súčasťou aplikácie fleXtat a modul fleXtat report wizard.

Postup testovania:

- Spustíte aplikáciu iReport
- Spustíte plugin fleXtat report wizard kliknutím na položky v menu *Plugins->fleXtat report plugin*
- Postupujte podľa inštrukcií v používateľskej príručke v kapitolách 6.1 až 6.5
- Novovytvorenú zostavu uložte a pridajte do databázy v aplikácii fleXtat browser (viď. používateľská príručka kapitola 4.3)
- Zostavu, ktorú ste vytvorili a vložili do databázy zvolíte na zobrazenie v aplikácii fleXtat browser (viď. 8.2)

Očakávaný výsledok:

- Po ukončení práce s modulom fleXtat report wizard sa v aplikácii iReport zobrazí náhľad na vytvorenú zostavu.
- Po pridaní novej zostavy do databázy z aplikácie fleXtat browser sa meno zostavy pridá do stromu zostáv v ľavej časti okna aplikácie
- Po vybratí zostavy na zobrazenie sa v strednej časti okna zobrazí zostava s vyplnenými tabuľkami prípadne vykreslenými grafmi
- Zobrazené dáta sú na prvý pohľad relevantné⁶

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom. Pomocou modulu je možné vytvoriť plne funkčné zostavy schopné komunikovať s databázou pomocou jazyka SQL.

8.6.2 Vytváranie špecializovaných zostáv pomocou scriptletov

Účel testovania:

- Otestovanie komunikácie zostáv s triedami implementujúcimi bussines logiku spracovania dát z databázy – scriptletmi.

Podmienky:

- Implementácia tried si vyžaduje nainštalovaný Java Development Kit a vývojové prostredie pre jazyk Java so schopnosťou kompilácie zdrojových súborov.
- Triedy scriptletov môže implementovať len osoba so znalosťami programovacieho jazyka Java.

⁶ v prípadoch, keď je možné na prvý pohľad povedať, že zobrazené údaje dávajú zmysel

Postup testovania:

- Implementujte triedu scriptletu. Postupujte podľa inštrukcií z metodiky o vytváraní scriptletov z kapitoly 5.
- Vytvorte zostavu podľa používateľskej príručky, ktorá bude spolupracovať so scriptletom.
- Nastavte parametre, série, polia a premenné v zostave podľa používateľskej príručky.
- Novovytvorenú zostavu uložte a pridajte do databázy v aplikácii fleXtat browser, pričom je potrebné vyplniť položku *Súbor scriptletu* pri vkladaní zostavy a nastaviť položku *Typ dátového zdroja* na typ *scriptlet* (viď. používateľská príručka kapitola 4.3)
- Zostavu, ktorú ste vytvorili a vložili do databázy zvolte na zobrazenie v aplikácii fleXtat browser (viď. 8.2)

Očakávaný výsledok:

- Po pridaní novej zostavy do databázy z aplikácie fleXtat browser sa meno zostavy pridá do stromu zostáv v ľavej časti okna aplikácie
- Po vybratí zostavy na zobrazenie sa v strednej časti okna zobrazí zostava s vyplnenými tabuľkami prípadne vykreslenými grafmi
- Zobrazené dáta sú na prvý pohľad relevantné

Výsledok testovania:

- V implementačnej fáze bolo navrhnutých a implementovaných 5 scriptletov kvôli implementácii preddefinovaných zostáv. Výsledky testovania týchto zostáv je možné nájsť v kapitolách 8.9.1 až 8.9.7 .

8.7 Testovanie správy zostáv

Účel testovania:

- Otestovanie funkcionality správy – administrácie zostáv
 - Otestovanie pridávania nových zostáv
 - Otestovanie premiestňovania zostáv
 - Otestovanie mazania zostáv

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (viď. 8.1)
- Používateľ musí mať prístupové práva pre rolu editora.

Postup testovania:

- Prihláste sa do aplikácie fleXtat browser
- Postupujte podľa inštrukcií v používateľskej príručke v kapitole 4.3
- V rámci testovania vykonajte pridanie, premiestnenie a odobratie zostavy

Očakávaný výsledok:

- Po operácii pridania novej zostavy sa nová zostava zaradí do kategórie, ktorá bola zadaná ako rodičovská kategória pre novú zostavu
- Po operácii premiestnenia zostavy resp. celej kategórie zostáv sa zostava (kategória so zostavami) premietni pod zadanú kategóriu zostáv
- Po operácii zmazania zostavy zo stromu zostáv sa táto odstráni zo stromu zostáv aj z databázy⁷

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom.

8.8 Testovanie administrácie databázy

Účel testovania:

- Otestovanie funkcionality databázovej vrstvy s manažmentom verzií

Podmienky:

- Musí byť vytvorená databáza fleXDB skriptom, ktorý je súčasťou projektu fleXtat
- Vytvorená databáza musí mať nastavené kódovanie UNICODE
- Databáza musí byť naplnená údajmi pomocou aplikácie fleXPump

Postup testovania:

- Prihláste sa do aplikácie fleXtat
- Prezrite a vytlačte si štatistiku o zastúpení jednotlivých pohlaví
- Prihláste sa do aplikácie na administráciu databázového servera s administrátorskými právami
- Prepnite sa na databázu fleXDB a otvorte si okno na zadávanie SQL príkazov
- Zadajte a spustite príkaz UPDATE cis_gender SET description = 'nemuž' WHERE id = 1
- Opätovne si prezrite a vytlačte si štatistiku o zastúpení jednotlivých pohlaví

Očakávaný výsledok:

- Zmena sa neprejaví v štatistikách

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom

8.8.1 Testovanie aktualizácie databázy aplikáciou fleXPump

Účel testovania:

- Otestovanie funkcionality aplikácie fleXPump

Podmienky:

- Musí byť vytvorená databáza fleXDB skriptom, ktorý je súčasťou projektu fleXtat
- Vytvorená databáza musí mať nastavené kódovanie UNICODE

⁷ po reštarte aplikácie sa v strome nezobrazí

- Používateľ spúšťajúci aplikáciu fleXPump musí mať oprávnenie čítať z databázy aplikácie elektronickej prihlášky a musí mať oprávnenie zapisovať do databázy FleXDB
- Pre spustenie aplikácie fleXPump musí byť prístup k konfiguračnému súboru aplikácie a k súboru jazykovej mutácie aplikácie elektronickej prihlášky

Postup testovania:

- Modifikovaním konfiguračného súboru aplikácie fleXPump nastavte adresy databázy elektronickej prihlášky a databázy FleXDB.
- Spustíte aplikácie FleXDB podľa pokynov, ktoré sa vypíšu na obrazovku po spustení aplikácie príkazom *java -jar flexpump.jar*

Očakávaný výsledok:

- Databáza FleXDB sa naplní dátami z databázy elektronickej prihlášky

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom

8.8.2 Testovanie zálohovania a obnovy databázy

Účel testovania:

- Simulácia scenára obnovy databázy aplikácie po výpadku

Podmienky:

- Musí byť vytvorená databáza FleXDB skriptom, ktorý je súčasťou projektu fleXtat
- Vytvorená databáza musí mať nastavené kódovanie UNICODE
- Databáza musí byť naplnená údajmi pomocou aplikácie FleXPump
- Pred zahájením testovania musí existovať záloha aktuálnej verzie databázy
- Nie nutnou, ale doporučenou podmienkou je existencia zálohy systémových objektov databázového servera (ide o používateľov a skupiny)
- Znalosť administrácie databázového servera – testovanie vykonáva administrátor servera

Postup testovania:

- Prihláste sa do databázového servera ako administrátor systému
- V aplikácii na správu databázového servera zrušte databázu fleXDB, čím sa vlastne stratia kompletne údaje (pravé tlačidlo na databázu fleXDB a Zmazať/Zrušiť)
- Vytvorte novú databázu s názvom fleXDB a kódovaním UNICODE
- Nastavte sa na databázu fleXDB (aktívna) a spustíte obnovu databázy z existujúceho zálohového súboru
- Obnovte používateľov systému
 - Ak existuje záloha systémových objektov tak z existujúcej zálohy systémových objektov
 - Ak neexistuje:

- § Vytvorte skupiny používateľov admins, editors a users
- § Zmažte všetkých používateľov aplikácie vymazaním všetkých záznamov v tabuľke dat_user
- § Pridajte administrátora systému príkazom `INSERT INTO dat_users (username, userpwd,usergroup) VALUES ('Admin', 'admin', 'admins');`
- § Spustite aplikáciu fleXtat a prihláste sa so základným administrátorským menom a heslom
- § Manuálne pridajte všetkých predchádzajúcich používateľov

Očakávaný výsledok:

- Databáza bude v rovnakom stave ako pred zahájením testovania

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom

8.9 Testovanie preddefinovaných zostáv

V implementačnej fáze boli implementované preddefinované zostavy, ktorých testovanie je popísané v nasledujúcich kapitolách. Testovanie preddefinovaných zostáv prebehlo nad reálnymi dátami z databázy elektronickej prihlášky z roku 2004.

8.9.1 Rozdelenie študentov/uchádzačov podľa veku

Účel testovania:

- Otestovanie funkcionality navrhutej a implementovanej preddefinovanej zostavy

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (viď. 8.1)
- Používateľ musí mať oprávnenie na prihlásenie sa do databázy v roli editora alebo referenta
- Zostava musí byť uložená v databáze FleXDB

Postup testovania:

- Prihláste sa do aplikácie fleXtat browser
- V strome zostáv v ľavej časti okna aplikácie fleXtat browser kliknutím rozbalte kategóriu *Rozdelenie uchádzačov*
- Kliknite na položku *Podľa veku a pohlavia*

Očakávaný výsledok:

- V strednej časti okna aplikácie fleXtat browser sa zobrazí zostava popísaná v dokumentácii k projektu⁸ v kapitole 6.4.1

⁸ Dokument zo zimného semestra

- Dáta zobrazené v zostave sú relevantné a zhodujú sa s dátami, ktoré by bolo možné získať iným spôsobom (napr. ručným vyhodnotením prihlášok)

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom.

8.9.2 Počet prihlásených a počet prijatých študentov

Zostava nebola implementovaná z dôvodu, že v súčasnosti je aplikácie elektronickej prihlášky nasadená len na FIIT. Navrhované grafy by boli jednofarebné.

8.9.3 Počet prichádzajúcich prihlášok v závislosti od dátumu

Účel testovania:

- Otestovanie funkcionality navrhutej a implementovanej preddefinovanej zostavy

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (viď. 8.1)
- Používateľ musí mať oprávnenie na prihlásenie sa do databázy v roli editora alebo referenta
- Zostava musí byť uložená v databáze FlexDB

Postup testovania:

- Prihláste sa do aplikácie fleXtat browser
- V strome zostáv v ľavej časti okna aplikácie fleXtat browser kliknutím rozbalte kategóriu *Prihlášky*
- Kliknite na položku *Podľa dátumu*

Očakávaný výsledok:

- V strednej časti okna aplikácie fleXtat browser sa zobrazí zostava popísaná v dokumentácii k projektu zo zimného semestra v kapitole 6.4.3
- Dáta zobrazené v zostave sú relevantné a zhodujú sa s dátami, ktoré by bolo možné získať iným spôsobom (napr. ručným vyhodnotením prihlášok)

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom.

8.9.4 Vplyv strednej školy na úspešnosť prijímacej skúšky

Účel testovania:

- Otestovanie funkcionality navrhutej a implementovanej preddefinovanej zostavy

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (viď. 8.1)

- Používateľ musí mať oprávnenie na prihlásenie sa do databázy v roli editora alebo referenta
- Zostava musí byť uložená v databáze FleXDB

Postup testovania:

- Prihláste sa do aplikácie fleXtat browser
- V strome zostáv v ľavej časti okna aplikácie fleXtat browser kliknutím rozbalte kategóriu *Úspešnosť*
- Kliknite na položku *Podľa strednej školy*

Očakávaný výsledok:

- V strednej časti okna aplikácie fleXtat browser sa zobrazí zostava popísaná v dokumentácii k projektu v kapitole 6.4.4
- Dáta zobrazené v zostave sú relevantné a zhodujú sa s dátami, ktoré by bolo možné získať iným spôsobom (napr. ručným vyhodnotením prihlášok)

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom.

8.9.5 Rozdelenie študentov na základe národnosti a miesta bydliska

Táto zostava bola rozdelená na dve zostavy. V prvej sa sleduje rozdelenie uchádzačov podľa národnosti, druhá zostava popisuje vzťah medzi úspešnosťou uchádzača a krajom, z ktorého pochádza.

Účel testovania:

- Otestovanie funkcionality navrhutej a implementovanej preddefinovanej zostavy

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (viď. 8.1)
- Používateľ musí mať oprávnenie na prihlásenie sa do databázy v roli editora alebo referenta
- Zostava musí byť uložená v databáze FleXDB

Postup testovania:

- Prihláste sa do aplikácie fleXtat browser
- Pre rozdelenie podľa národnosti
 - V strome zostáv v ľavej časti okna aplikácie fleXtat browser kliknutím rozbalte kategóriu *Rozdelenie uchádzačov*
 - Kliknite na položku *Podľa národnosti*
- Pre vzťah medzi úspešnosťou a krajom bydliska
 - V strome zostáv v ľavej časti okna aplikácie fleXtat browser kliknutím rozbalte kategóriu *Úspešnosť*
 - Kliknite na položku *Podľa kraja*

Očakávaný výsledok:

- V strednej časti okna aplikácie fleXtat browser sa zobrazí zostava popísaná v dokumentácii k projektu v kapitole 6.4.5
- Dáta zobrazené v zostave sú relevantné a zhodujú sa s dátami, ktoré by bolo možné získať iným spôsobom (napr. ručným vyhodnotením prihlášok)

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom.

8.9.6 Vplyv súťaží na úspešnosť prijímacej skúšky

Účel testovania:

- Otestovanie funkcionality navrhutej a implementovanej preddefinovanej zostavy

Podmienky:

- Musia byť splnené všetky všeobecné podmienky pre spustenie a testovanie aplikácie fleXtat (viď. 8.1)
- Používateľ musí mať oprávnenie na prihlásenie sa do databázy v roli editora alebo referenta
- Zostava musí byť uložená v databáze FleXDB

Postup testovania:

- Prihláste sa do aplikácie fleXtat browser
- V strome zostáv v ľavej časti okna aplikácie fleXtat browser kliknutím rozbalte kategóriu *Úspešnosť*
- Kliknite na položku *Podľa súťaže*

Očakávaný výsledok:

- V strednej časti okna aplikácie fleXtat browser sa zobrazí zostava popísaná v dokumentácii k projektu v kapitole 6.4.6
- Dáta zobrazené v zostave sú relevantné a zhodujú sa s dátami, ktoré by bolo možné získať iným spôsobom (napr. ručným vyhodnotením prihlášok)

Výsledok testovania:

- Výsledok testovania sa zhoduje s očakávaným výsledkom.

8.9.7 Úspešnosť prihlásených študentov v závislosti od úspešnosti maturitnej skúšky z určitého predmetu

Zostava nebola implementovaná z dôvodu, že databáza elektronickej prihlášky neobsahuje informácie o výsledku maturitnej skúšky uchádzača.

9 Záver

Táto kapitola je záverom druhej časti dokumentácie k projektu Systém pre štatistické vyhodnocovanie elektronickej prihlášky z predmetu Tímový projekt.

Dokument obsahuje opis implemetácie výsledkov práce v letnom semestri pozostávajúci predovšetkým z opisu implementácie aplikácie Flextat Browser, Flextat Report pluginu, úprav do JasperReportu a IReportu, aplikácie flexXPump a databázy. Okrem opisu implementačných špecifik jednotlivých častí systému bol zdokumentovaný aj postup vytvárania špecializovaných scriptletov a pridávanie faktov do systému Flextat. V predposlednej kapitole je uvedený popis akceptačných testov s postupom a výsledkom alfa testovania systému Flextat.

Počas celého letného semestra tím implementoval všetky navrhnuté časti systému. Výsledkom práce na projekte je komplexný systém na prezentáciu uložených dát vo forme štatistických výstupov. Spomenuté výstupy sú jednoducho manažovateľné, s možnosťou jednoduchšej zmeny niektorých parametrov, okrem toho systém ponúka aj možnosť vytvorenia vlastných jednoduchých zostáv s využitím sprievodcu.