EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 17300/B/2012/1400115131

DATABÁZA ROZVRHU HODÍN NA EUBA A JEJ AUTOMATICKÉ NAPĹŇANIE Z TEXTOVÉHO SÚBORU PROGRAMU ROZVRHY

Bakalárska práca

2011/2012 **Lukáš Petráš**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

DATABÁZA ROZVRHU HODÍN NA EUBA A JEJ AUTOMATICKÉ NAPĹŇANIE Z TEXTOVÉHO SÚBORU PROGRAMU ROZVRHY

Bakalárska práca

Študijný program: Hospodárska informatika

Študijný odbor: 6292 Hospodárska informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: Ing. Daniel Kastl, PhD.

Bratislava 2012 **Lukáš Petráš**

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

Dátum: **.....**

Poďakovanie

Týmto by som chcel v krátkosti poďakovať vedúcemu mojej záverečnej práce, pánovi Ing. Danielovi Kastlovi, PhD. za jeho ochotu a obetu jeho vzácneho času a takisto za to, že mi bol vždy nápomocný, v situáciách, keď som si nevedel dať rady.

Abstrakt

PETRÁŠ Lukáš: *Databáza rozvrhu hodín na EUBA a jej automatické napĺňanie z textového súboru programu Rozvrhy*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúci záverečnej práce: Ing. Daniel Kastl, PhD. – Bratislava: FHI EU, 2012, počet strán: 53

Cieľom záverečnej práce je navrhnúť databázu rozvrhu hodín na Ekonomickej Univerzite a programovú aplikáciu, ktorá ju bude automaticky napĺňať po zadaní zdrojového súboru databázy používaného aj v programe Rozvrhy. Práca je rozdelená do troch kapitol. Obsahuje 0 grafov, 2 tabuľky, 9 diagramov a 1 prílohu. Prvá kapitola je venovaná súčasnému stavu problematiky spracovávania rozvrhu hodín. V ďalšej časti sa charakterizujú ciele práce a metódy ich vypracovania. Záverečná kapitola sa zaoberá vývojom databázy v jej postupných krokoch a metodikou vytvorenia databázovej a používateľskej aplikácie. Výsledkom riešenia danej problematiky je navrhnutá databáza rozvrhu hodín, databázová aplikácia, umožňujúca automatické napĺňanie tejto databázy a používateľská aplikácia slúžiaca k prezeraniu rozvrhu hodín.

Kľúčové slová: rozvrh, databáza, parsing

Abstract

PETRÁŠ Lukáš: *Database of course schedule on EUBA and its automatic filling from text file of the program Rozvrhy*. – Economic university in Bratislava. Faculty of Economic informatics; Department of Applied Informatics. – thesis supervisor: Ing. Daniel Kastl, PhD. – Bratislava: FHI EU, 2012, number of pages: 53

Objective of this thesis is to design a database of course schedule on Economic university and program application, which will automatically fill the database after asking the source file used in application Rozvrhy. The work is divided into three chapters. It contains 0 graphs, 2 tables, 9 diagrams and 1 addition. The first chapter is devoted to current issue condition of processing the course schedule. In next part are defined the objectives and methods of their elaboration. The last chapter deals with development of database in its onward steps and manner of creating the database and user application. The result of solution is a designed database of course schedule, database application allowing automatic filling and user application, which serve to viewing of course schedule.

Key words: schedule, database, parsing.

Obsah **str.**

Úvod

Úlohou tejto práce je vytvorenie dynamickej databázy, ktorá slúži na prezeranie údajov, týkajúcich sa rozvrhu hodín pre rôzne typy používateľov na Ekonomickej univerzite. Naplnenie databázy je realizované zo zdrojového textového súboru vo formáte .txt alebo .par, ktorý slúžil aj ako základ pre samotné vytvorenie štruktúry tabuliek. Tento súbor treba pomocou vhodného programovacieho jazyka načítať, rozkúskovať na samostatné textové reťazce, ktoré podľa vhodného kľúča uložiť do tabuľky v databáze. Uloženie je trvalé a jednorazové, pričom do tých istých tabuliek sa v budúcnosti budú ukladať aj ďalšie rozparsované zdrojové súbory, odlíšené pomocou školského roku a semestra.

Údaje v databáze obsahujú informácie dôležité pre študentov i učiteľov, prístup k nim, však je možný aj z pohľadu miestnosti. Používateľ tejto aplikácie si dokáže prezrieť aký rozvrh hodín, ktorý ho čaká v konkrétny semester, pre konkrétny deň, či hodinu. Používateľom sa zobrazujú informácie o názve predmetu, vyučujúcom, miestnosti, času a o tom, či ide o prednášku alebo cvičenie. Stačí mu len zadať odbor, prípadne krúžok, o ktorého rozvrh sa používateľ zaujíma a aplikácia mu v databáze vyhľadá požadované informácie. Rovnako môžu učitelia prezerať svoj rozvrh hodín, ktorý sa im zobrazí po zadaní mena a poskytne informácie, ktorý deň, či hodinu majú aký predmet vyučovať, v akej miestnosti a pre aký odbor, respektíve krúžok a či ide o prednášku alebo cvičenie. Ak by používateľ chcel prezerať rozvrh hodín pre konkrétnu miestnosť, po zadaní označenia tejto miestnosti, by našiel aký predmet sa v nej bude vyučovať, prednášajúceho alebo cvičiaceho a odbor alebo krúžok, ktorý má v daný deň a danú hodinu vyučovanie.

1 Súčasný stav problematiky

Súčasný postup tvorby rozvrhu na Ekonomickej univerzite prechádza mnohými náročnými procedúrami. Podľa informácií dodaných vedúcim práce vyzerá na vybranej katedre nasledovne.

Na katedru prídu z dekanátu plánované počty študentov a počty krúžkov na nasledujúci semester na každý ročník a každý odbor. Katedrový koordinátor rozvrhu podľa študijných plánov a počtov študentov zostaví zoznam povinných predmetov a povinne voliteľných predmetov určených katedrou aj s počtami krúžkov určených na dané predmety. Predmety rozdelí medzi učiteľov podľa odbornosti a podľa toho koľko majú podľa zmluvy odučiť hodín. Používa na to tabuľkové zobrazenie v MS Excel. Tento proces sa zopakuje aj pre externé štúdium.

Navrhnuté povinné a povinne voliteľné predmety následne koordinátor pošle naspäť na dekanát, ktorý ich pošle do rozvrhárne, kde sa z nich vytvorí prvá verzia rozvrhu. Táto verzia rozvrhu je v papierovej podobe doručená na katedry. Katedrový rozvrh, teda rozvrh zamestnancov katedier, je celý ručne vkladaný do tabuľky v Exceli. Ten sa následne mailom rozpošle všetkým učiteľom na pripomienkovanie. Učitelia odpovedajú mailom koordinátorovi, pričom obsahom správ sú ich pripomienky.

Koordinátor prijíma aj požiadavky iných katedier na učebne katedry a overuje ich s vedúcou katedry. Zaznačuje ich do špeciálneho listu, aby nefigurovali v rozvrhu domovskej katedry a na piatich hárkoch rozvrhu v Exceli musí prácne zisťovať, či je dané miesto v rozvrhu ešte voľné. Koordinátor pripomienky zanesie do rozvrhárne, zaznamená si ktoré sa podarilo a ktoré sa nepodarilo uplatniť. Proces pripomienkovania sa opakuje aj 4 až 5 krát, kým sa dosiahne jeho finálna podoba. Výsledný rozvrh sa odosiela učiteľom, tlačí sa a vytavuje sa na nástenke. Takisto sa uverejňuje na stránkach univerzity vo formáte .bin, kde je možné si ho stiahnuť a otvoriť v programe LookIn.

Na konci semestra, školského roka a kalendárneho roka sa robia na katedre výkazy skutočne odučených hodín vyučujúcich (doktoranti, externí pracovníci a interní pracovníci) a odosielajú sa na dekanát spolu s porovnaním s minulým rokom, vyjadreným v percentách.

1.1 Problémy súčasného prístupu

Ručné vkladanie zoznamu predmetov a počtu študentov zo študijných plánov nesie veľké riziko vzniku chýb z manuálneho vkladania.

Zložitý systém zostavovania zmluvných úväzkov učiteľov je vykonávaný manuálne, pričom chýba informácia, kto učí ktorý odbor pri predmetoch, ktoré učia viacerí učitelia.

Ručné prepisovania celého rozvrhu hodín do Excelovského formátu je veľmi zdĺhavé a riziko vzniku chýb je vysoké.

Pripomienky učiteľov si koordinátor musí ručne kopírovať do samostatného súboru, alebo zaznačovať do návrhu rozvrhu v Exceli šípkami a poznámkami, z čoho vzniká značná neprehľadnosť v pripomienkach.

Zaznačovanie obsadenosti katedrových miestností inými katedrami je veľmi neprehľadné, čo spôsobuje zdĺhavé vyhľadávanie obsadenosti jednotlivých okien v Excelovskom rozvrhu.

Pri mailovej komunikácii katedry s učiteľom môže dôjsť k prehliadnutiu jednej požiadavky popri zozname všetkých požiadaviek, čo opäť predlžuje komunikáciu oboch strán.

Zaznamenávanie toho, či sa realizácia podarila, alebo nie, je náročné. Spôsobuje to zdĺhavú komunikáciu s jednotlivými žiadateľmi – písanie mailov každému jednému, osobné alebo telefonické predanie informácie.

Výsledný rozvrh pre katedru je organizovaný iba podľa jedného kritéria – ročníku. Nie je teda možné rýchlo vyhľadať rozvrh iba jedného krúžku, odboru, alebo učiteľa alebo viacnásobných výberových kritérií.

Pre prezeranie rozvrhu hodín v programe LookIn je potrebné vykonať na dnešnú dobu absurdne veľa krokov, ktoré sa nedajú zrealizovať dnešnými modernými technológiami - mobilný prístup, prístup z internetovej kaviarne, prístup z PC kde nie je povolené inštalovať programy.

2 Ciele a metódy

Skúmaním súčasného stavu tvorby, pripomienkovania, zobrazovania a vykazovania rozvrhu sme stanovili nasledujúce ciele bakalárskej práce.

2.1 Ciele

Hlavným cieľom tejto práce je návrh a vytvorenie databázy, ktorá bude schopná ukladať všetky informácie o rozvrhu na Ekonomickej univerzite v Bratislave a vytvorenie aplikácie, ktorá bude schopná automaticky načítavať zdroj údajov v podobe textového reťazca, rozložiť ho na menšie reťazce a uložiť do databázy. Druhou časťou bude používateľské rozhranie, v ktorom môže používateľ interaktívne vyberať údaje na základe požiadaviek, ktoré sám definuje. Rozvrh hodín sa mu zobrazí v konzolovom výstupe aplikácie, formátovaný do mriežky.

2.1.1 Návrh databázy

Databáza musí byť navrhnutá tak, aby spĺňala všetky požiadavky zdrojového súboru, teda jeho štruktúru, čím sa myslí poradie jednotlivých položiek, respektíve entít, no netreba zabúdať na výnimky, vyskytujúce sa v tomto textovom súbore nezávisle na jeho časti. V spomínanom poradí treba ukladať údaje do databázy, kvôli prepojenosti jednotlivých entít, pomocou primárnych a cudzích kľúčov. Druhým dôležitým predpokladom pri návrhu je používateľský pohľad, ktorý definuje hlavné požiadavky pri výbere údajov. Tento výber bude ľahšie realizovateľný pri čím vhodnejšie navrhnutej štruktúre tabuliek, primárnych a cudzích kľúčov.

2.1.2 Načítanie zdrojového súboru

Ak chceme pracovať so zdrojovým súborom, musíme ho načítať pomocou zvoleného programovacieho jazyka, v ktorom je napísaná aplikácia. Kvôli uľahčeniu práce databázového administrátora, ktorý sa chystá uložiť databázu, by bolo ideálnym riešením načítanie cesty ku súboru priamo z konzoly vo vývojovom prostredí. Pri realizácii tohto riešenia by administrátor nemusel zasahovať do zdrojové textu aplikácie, ale len interaktívne napísať cestu k zdrojovému súboru databázy.

2.1.3 Parsing zdrojového súboru

Parsing je najdôležitejšou časťou v činnosti aplikácie. Môžeme mať načítaný zdrojový súbor, ale čo z toho, keby neexistoval algoritmus, ktorý by ho vhodne spracoval? Parsovanie (z angl. parse: rozoberať, rozložiť) znamená rozdelenie jedného textového reťazca, ktorým je zdrojový súbor, obsahujúci niekoľko tisíc riadkov, na menšie textové reťazce, alebo číselné dátové typy. Hlavným cieľom tejto práce je teda návrh robustného mechanizmu, schopného rozparsovať súbor na niekoľko menších častí a tie následne parsovať po riadkoch, z ktorých sa vyčlenia dôležité údaje, potrebné na vloženie do databázy. Táto aplikácia by mala zároveň dbať aj na nedostatky zdrojového súboru, či prípadné výnimky.

2.1.4 Ukladanie údajov do databázy

Na túto oblasť aplikácie treba takisto vyčleniť samostatné funkčné jednotky, pričom ukladanie údajov je nemenej dôležité ako parsovanie zdrojového súboru. Ukladanie dát by malo prebiehať ihneď po rozparsovaní konkrétnej informácie, čiže nie vytvárať pomocné úložiská v operačnej pamäti, ale pracovať takmer súbežne. Chod programu by teda mal vyzerať nasledovne: *načítanie – parsing – uloženie – parsing – uloženie* a tak ďalej až po uloženie posledného údaju do databázy. Uloženie údajov sa bude vykonávať pomocou aplikovania databázového programovacieho jazyka v prostredí zdrojového kódu programu.

2.1.5 Výber požadovaných údajov z databázy

Po naplnení databázy dátami o rozvrhu hodín na Ekonomickej univerzite (pre konkrétny školský rok a semester) je potrebné, aby tieto informácie ostali prístupné na prezeranie. Je zrejmé, že výber týchto údajov sa bude vykonávať pomocou databázového programovacieho jazyka. Ten je teda nutné implementovať aj do druhej časti aplikácie, tej, ktorá je určená koncovému používateľovi, ktorý si bude môcť vybrať aký typ rozvrhu chce vidieť. K dispozícii mu budú rozvrhy pre študenta, učiteľa a miestnosť. Ak je používateľom študent, bude si môcť vybrať rozvrh pre svoj odbor (prípadne pododbor), krúžok a prehliadať taktiež aj voliteľné predmety, určené pre jeho ročník a odbor.

2.1.6 Zobrazenie vybraných údajov do konzoly

Pretože náplňou mojej práce nie je vytvorenie aplikácie s grafickým používateľským rozhraním, ale iba konzolovým, bolo by vhodné vytvoriť prostredie pre zobrazenie rozvrhu v konzole, ktoré by zodpovedalo tabuľke, alebo mriežke, pre ľahšiu prehľadnosť údajov.

2.1.7 Ukladanie nových rozvrhov do databázy

Ďalším plánom je ukladať nové semestre do tej istej databázy, kde budú ponechané staré semestre, kedykoľvek pripravené k prehliadaniu. Návrh databázy teda treba prispôsobiť aj tejto vízii, aby bola teda pripravená ponúkať súčasné údaje, rovnako ako aj staré niekoľko rokov.

2.2 Metódy

Na dosiahnutie požadovaných cieľov sme zvolili nasledovnú metodiku postupov.

2.2.1 Návrh databázy

Tento proces je založený na viacerých podprocesoch, ktoré treba zodpovedne vykonať pre správnu funkčnosť databázy. Nejedná sa teda len o napísanie zdrojového kódu databázy, ale o komplexné zostavenie podmienok, ktoré treba zohľadniť pri vytváraní.

Definovanie používateľských požiadaviek

V Prvom rade treba vedieť čo má databáza zobrazovať vo výstupe, ak má totiž niekomu slúžiť, musí si vopred určiť, čo chce vidieť. Až potom možno uvažovať ďalej.

Analýza zdrojového súboru

Zdrojový textový súbor obsahuje základnú množinu informácií definujúcich podmienky na návrh databázy, druhou časťou sú požiadavky používateľov. V tomto súbore existuje určitá postupnosť údajov, z nich treba vybrať tie dôležité, pre chod bázy dát. Nástrojom analýzy môže byť takisto manuál k prekladaču, čo je dokument v MS Word,

určený na čítanie, ktorý vysvetľuje všetky označenia údajov a popisuje dáta, ktoré sa pod týmito označeniami skrývajú.

Výber potrebných entít a ich atribútov

Tieto entity sú definované v zdrojovom súbore a v manuáli k prekladaču. Jednotlivé atribúty treba vybrať podľa dôležitosti v dátovom modely. V tejto časti treba zohľadniť aj používateľské oko, ktoré bude mať rôzne požiadavky na výber údajov z databázy. Po tejto fáze vzniká *konceptuálny dátový model*.

Normalizácia

Definícia normalizácie tvrdí, že ide o proces implementácie analytických pravidiel pri testovaní atribútov relácií a ich transformácia do formy, ktorá minimalizuje. Úlohou normalizácie je odstránenie vzťahov N:M, duplicít, redundancie a dosiahnutie minimálneho počtu potrebných atribútov. Skladá sa z troch častí, pričom každú treba vykonať.

V prvej časti, teda v prvej normálnej forme, je relácia vtedy, keď ku zvolenému primárnemu kľúču nemôže žiadny atribút obsahovať viac ako jednu hodnotu.

V druhej časti, teda v druhej normálnej forme, je relácia vtedy, keď je v 1.NF a zároveň každý nekľúčový atribút je funkčne závislý na celom primárnom kľúči, teda ak entita obsahuje viacprvkový primárny kľúč, atribúty nie sú závislé iba na niektorej jeho časti.

V tretej časti, teda v tretej normálnej forme, je relácia vtedy, keď je v 2.NHF a zároveň neobsahuje tranzitívne závislosti. To znamená, že atribút A je závislý na atribúte B a až atribút B je závislý na primárnom kľúči C.

Výsledkom tohto procesu je *logický dátový model*.

Zostavenie tabuľky primárnych a cudzích kľúčov

Tabuľka ponúka prehľadnejšiu formu závislostí jednotlivých relácií pomocou primárnych a cudzích kľúčov. Následne bude jednoduchšie určiť polohu entít v entitno-relačnom diagrame, ak ich budeme mať takto predostreté v tabuľke aj s priradenými konektormi.

Vytvorenie entitno–relačného diagramu

Diagram je akési grafické znázornenie štruktúry databázy, hoci z neho nevyplývajú názvy primárnych a cudzích kľúčov, orientácia konektorov je tam viditeľná. Akékoľvek riešenia problémov, vzniknutých v databáze bude potom najľahšie sledovať práve na diagrame.

Vytvorenie tabuliek databázy

Až po všetkých predchádzajúcich krokoch, ktorých výsledkom musí byť kvalitne štruktúrovaná databáza, možno prejsť k tej poslednej a teda k vytvoreniu *fyzického modelu dát*. Vo vybranom databázovom prostredí sa pomocou jazyka SQL vytvoria tabuľky, kde domény atribútov budú predstavovať stĺpce, pričom jeden, alebo viac z nich je definovaný ako primárny kľúč a ďalšie zase ako cudzí kľúč, čo je rovnako dôležité.

2.2.2 Testovanie databázy

Testuje sa schopnosť databázy naplno uspokojovať potreby používateľov, teda či databáza dokáže vyberať údaje na základe nimi definovaných požiadaviek ešte vo fáze návrhu konceptuálneho modelu dát.

Naplnenie tabuliek testovacími údajmi

Na otestovanie rozvrhu treba najskôr vložiť nejaké dáta do tabuliek. Testovacie údaje by mali byť rôznorodé a zachytávať aj prípadné výnimky, ktoré musí databáza zvládať.

Testovanie pomocou výberových dotazov

Výberové dotazy sú realizované spájaním tabuliek vo výberových dotazoch select, ktoré sú definované nad databázou rozvrhu. V týchto dotazoch sa zároveň aj určujú podmienky zobrazenia jednotlivých údajov a výber konkrétnych atribútov určených na zobrazenie v tabuľke.

2.2.3 *Parsing zdrojového súboru*

Parsovanie je proces drobenia jedného bloku textu na malé časti v textovej, znakovej alebo číselnej podobe.

Vytvorenie vhodného algoritmu pre automatické parsovanie zdrojového súboru

Algoritmus je prenesene povedané recept na riešenie určitého problému. Podľa môjho názoru je to silný nástroj programátora, bez ktorého by si počínal akoby bez plánu výroby. Pre vytvorenie algoritmu treba na začiatku vedieť definovať čo by malo byť jeho cieľom a akou cestou to dosiahnuť. Cesta sa môže rôzne vetviť, preto je dobré ošetriť každý prípad. Algoritmus môže obsahovať aj cyklické prvky, ktoré úzko súvisia so spĺňaním podmienok vetvenia. Algoritmus možno nakresliť ako vývojový diagram tvorený procesnými blokmi, rozhodovacími, dátami pripravenými na uloženie a samozrejme môže zahŕňať aj cykly. Nákres tohto diagramu môže byť učinený na papieri, alebo kvôli estetickej stránke v špecializovanom softvéry.

Testovanie parsovacieho algoritmu

Testovanie však už musí mať fyzický základ toho, čo ideme testovať. Ide teda o fyzicky vytvorené fragmenty kódu, napísané vo vybranom programovacom jazyku a vybranom vývojovom prostredí. Zvyčajne každé vývojové prostredie umožňuje spustenie uloženého kódu a vypísanie výsledku ako konzolovú aplikáciu. Takto bude testovaný aj parsovací algoritmus, pretože výsledok, bude okamžite zrejmý z grafického výstupu.

2.2.4 *Vytvorenie používateľskej časti aplikácie*

Používateľská časť tejto aplikácie sa neskladá z grafického používateľského rozhrania, k dispozícii teda nebudú žiadne okná, ani webové rozhranie, ale iba výstup v konzole.

Spôsob výberu údajov z databázy

Výber údajov treba naprogramovať tak, aby sa prispôsobil požiadavkám užívateľom. Bolo by teda vhodné vypracovať systém dopytovania sa užívateľovi s tým, čo

chce zobraziť do konzoly, teda či to bude rozvrh pre študenta, učiteľa, alebo miestnosť, či iné požiadavky pre konkrétneho používateľa

Spôsob zobrazenia potrebných údajov

Zobrazenie údajov by mal byť proces nadväzujúci na ich výber. Bolo definované, že ideálnym zobrazením by bolo do mriežky. Mriežka by mala obsahovať rozmery 5x7 okien, teda 5 dní a 7 vyučovacích hodín na Ekonomickej univerzite. Mriežka by mala byť lemovaná označením jednotlivých dní a hodín ich skratkami, resp. číslami. Podľa vhodného kľúča sa budú predmety zobrazovať na správnych pozíciách v mriežke v závislosti od dňa a hodiny prednášky či cvičenia.

- Pre študenta sa budú zobrazovať v okienku nasledovné parametre: skratka predmetu, typ predmetu, údaj značiaci, či sa jedná o cvičenie alebo prednášku, priezvisko učiteľa a údaj o miestnosti.
- Učiteľ by mal vo svojom rozvrhu nájsť: skratku predmetu, ktorý vyučuje, či sa jedná o prednášku alebo cvičenie, takisto údaj o miestnosti a označenie paralelky, pre ktorú bude vyučovať. Paralelkou je myslený odbor v kombinácií s ročníkom.
- Pri prehliadaní rozvrhu hodín pre miestnosť musí okienko v mriežke obsahovať: skratku predmetu, ktorý sa vyučuje v danej miestnosti, či sa jedná o prednášku alebo cvičenie, priezvisko vyučujúceho a označenie paralelky.

2.3 Prostriedky na vypracovanie

Na splnenie stanovených cieľov potrebujeme softvérové vybavenie, ktoré nám umožní zhotoviť výsledný systém a takisto informačné vybavenie v podobe zdrojových dát pre prácu systému.

2.3.1 Softwarové vybavenie

1. Java virtual machine: virtuálny stroj, ktorý umožňuje vykonávať v tomto operačnom systéme príkazy v jazyku Java a takisto poskytuje vývojárske prostredie pre jazyk Java

2. Vývojové prostredie Eclipse: komplexný software, ktorý uľahčuje prácu s programovacím jazykom Java, umožňuje pripojenie mnohých plug-inov, ktoré zabezpečujú pripojenie na servery, do databázy atď.
3. Databáza Derby: jeden z funkčných pluginov v programe Eclipse, ktoré predstavuje úložisko informácii na trvalo, uložených na pevnom disku počítača. Pomocou jazyka SQL som v tejto databáze vytváral štruktúru mojej konkrétnej databázy so všetkými vzájomnými prepojeniami
4. MS Visio: program z balíka Office, ktorý slúži na vytváranie diagramov, schém, štruktúr a pod. pomocou zápisu v blokoch a konektormi medzi jednotlivými blokmi. Bol ideálnym na tvorbu vývojového diagramu pre parsovací algoritmus. Takisto som v ňom vytvoril konečnú podobu ERD diagramu pre databázu.

2.3.2 Informačné vybavenie

A. Zdrojový súbor: textový súbor, v ktorom je v rámci prísnych syntaktických pravidiel, zapísaný kompletný rozvrh hodín pre celú Ekonomickú Univerzitu v Bratislave. Tento text je predmetom spracovania bakalárskej práce. Pre mňa tvorí tento súbor cvičný súbor, ktorý mi otvára možnosti testovania na konkrétnom súbore. Môj konkrétny zdrojový súbor je rozvrh hodín pre zimný semester 2010/2011 a letný semester 2010/2011. Tento súbor zároveň poskytuje zdrojové informácie aj pre klasický program na prehliadanie rozvrhu, Look in.
B. Manuál k prekladaču: prekladačom je nazvaný zdrojový súbor. Manuál poskytuje vysvetlivky a popis ku každému atribútu textového súboru. Zároveň predstavuje súhrn pravidiel syntaxe, ktorými sa bezpodmienečne musím riadiť pri spracovávaní zdrojového súboru, jeho parsovaní a ukladaní do databázy

3 Výsledky

Vypracovaním spomínaných metód som dosiahol nasledujúce výsledky, opísané v rovnakej postupnosti.

3.1 Návrh databázy

Pracoval som podľa všeobecných princípov navrhovania databáz, vytvoreného E. F. Coddom. Najskôr som dostal definované používateľské požiadavky a nároky na databázu. Základom bolo získať zdrojový súbor, ktorý mi zabezpečil vedúci práce, spolu s manuálom k tomuto súboru. Podarilo sa mu získať 2 súbory, pre zimný a letný semester minulého školského roku 2010/2011.

3.1.1 Definovanie používateľských požiadaviek

Zoznam používateľských požiadaviek na databázu rozvrhu:

- rátame úplne všeobecný model rozvrhu t.j. rozvrh celej univerzity
- v databáze chceme evidovať
    a. učiteľov
    b. katedry
    c. fakulty
    d. miestnosti
    e. predmety
    f. PP, PVP (niektoré môžu byť aj v rôznych študijných programoch)
    g. študijné odbory
    h. krúžky v odboroch v danom ročníku
    i. študijné plány (predmety na odboroch)
    j. hodiny daného krúžku a predmetu v konkrétnom semestri, a miestnosti
    k. hodiny a ich časový rámec (1...7, 7:30-9:00, 9:15-10:45,.....)
- archivovať všetky rozvrhy zo starších ročníkov (tak aby nevznikali príliš veľké tabuľky)
- generovať zoznamy možných PP a PVP na LS aj ZS pre jednotlivé odbory (zo študijných plánov) + kto ich učí a aký počet cvičení bude nasadených

- načítať aktuálny rozvrh zo systému WinRozvrhy (predísť ručnému vkladaniu)
- zamedziť prekrývaniu viacerých cvičení pre jedného učiteľa, krúžok, miestnosť...
- zobrazovať týždenný rozvrh podľa
    a. učiteľa
    b. miestnosti
    c. ročníka
    d. odboru
    e. krúžku
    f. semestra
    g. ... a ich kombinácií

3.1.2 Analýza zdrojového súboru

Jedná sa o zdrojový súbor v textovej podobe určený pre program LookIn, používaný na Ekonomickej Univerzite pre prehliadanie rozvrhu hodín. Do toho programu si však používatelia sťahujú jeho binárnu podobu. Zdrojový súbor je koncipovaný tak, že každý odsek je oddelený vo formáte: ".NAZOV" vždy samostatne v novom riadku. Skladá sa z 13 takýchto častí v tomto poradí: .DATUM, .FAKULTA, .OBDOBIE, .ZADAVATEL, .ROZSAH, .HODINY, .KATEDRY, .MIESTNOSTI, .UCITELIA, .PARALELKY, .PREKAZKY, .PREDMETY, .KONIEC. Pod týmito „nadpismi" sú zapísané dáta, ktoré potrebujeme vybrať. Niektoré takéto celky však nie sú potrebné. Celky preto, lebo tak budem nazývať celú časť od názvu celku až po názov ďalšieho celku, kde končí prvý celok a začína druhý:

- .DATUM neobsahuje nič
- .FAKULTA (1 riadok) – názov a adresu školy
- .OBDOBIE (1 riadok) – informácia o semestri a školskom roku rozvrhu
- .ZADAVATEL (1 riadok) – názov organizačnej zložky, tvoriacej rozvrh
- .ROZSAH (1 riadok) – obsahuje 5 čísiel v nemennom poradí (tvar 13 5 7 : 3 7):
    1. Číslo – počet týždňov v semestri
    2. Číslo – počet vyučovaných dní v týždni
    3. Číslo – počet hodín v rámci jedného dňa
    4. Číslo – počet hodín dopoludňajšej výuky

5. Číslo – počet hodín popoludňajšej výuky

- .HODINY (7 riadkov) – rozpis začiatkov a koncov jednotlivých vyučovacích hodín s číselným označením hodiny

(počet riadkov bude ďalej variabilný)

- .KATEDRY– jeden riadok obsahuje 4 údaje:
  1. Názov katedry
  2. Skratka katedry
  3. Farba katedry – číslo od 0 po 728
  4. Farba textu – číslo od 0 po 728
- .MIESTNOSTI – jeden riadok obsahuje 3 údaje:
  1. Typ miestnosti – jedno z písmen U , L, S, I, (učebňa, laboratórium, špeciálna, informatická)
  2. Označenie miestnosti
  3. Kapacita miestnosti
- .UCITELIA – 5 údajov v jednom riadku
  1. Priezvisko učiteľa
  2. Meno učiteľa
  3. Katedra, na ktorej učiteľ pôsobí
  4. Titul učiteľa
  5. Typ učiteľa - vždy písmeno U
- .PARALELKY – štruktúra tohto bloku je odlišná.
  1. V jednom riadku sa nachádza označenie ročníka
  2. V ďalšom meno fakulty a skratka fakulty
  3. V ďalšom meno odboru, skratka odboru, počet študentov, počet krúžkov
  4. V ďalšom riadku sa môže nachádzať meno pododboru, skratka pododboru, počet študentov a počet krúžkov

Na odlíšenie riadku s fakultou sa na začiatku riadku uvádza symbol $. Pre odlíšenie odboru sa na začiatku riadku používa $$ a pre pododbor $$$. Pododbor sa však vyskytuje iba v prípade fakulty NHF v 4. a 5. ročníku. Treba však s ním ďalej uvažovať.

- .PREKAZKY – v jednom riadku je uvedených 7 údajov
    1. Typ prekážky – vždy písmeno M, teda prekážka sa týka miestnosti
    2. Špecifikácia typu – označenie miestnosti
    3. Kvalita – písmeno I (interná prekážka), alebo E (externá prekážka)
    4. Meno prekážky – text na 8 znakov
    5. Informácie o prekážke – text na 8 znakov
    6. Deň prekážky
    7. Hodina prekážky, alebo rozsah hodín prekážky
- .PREDMETY – najnáročnejší celok so špecifickou štruktúrou

Prvý riadok vždy obsahuje:

1. Označenie ročníka, fakulty, odboru, (prípadne pododboru), pre ktorý sa vzťahuje daný predmet
2. Kód predmetu – číselné označenie predmetu
3. Skratka katedry, pod ktorou sa daný predmet vyučuje
4. Typ prednášky (predmetu) – P (povinný), v (voliteľný), alebo S (špeciálny)
5. Typ cvičenia – V (všeobecné), L (laboratórne), S (špeciálne)
6. Názov predmetu
7. Skratka predmetu
8. Meno učiteľa – celé meno
9. Počet hodín prednášky
10. Počet hodín cvičení
11. Počet študentov na predmet
12. Poradové číslo predmetu v zozname

Druhý riadok obsahuje:

1. Zoznam krúžkov pre ktoré je prednáška určená – nepovinný riadok, rozlíšenie pomocou znaku # na začiatku riadku

Tretí riadok obsahuje:

1. Farba predmetu – číslo od 0 po 728
2. Farba textu – číslo od 0 po 728

Štvrtý riadok obsahuje:

1. Znak $ na odlíšenie prednášky
2. Počet hodín prednášky
3. Miestnosť prednášky – označenie miestnosti
4. Deň prednášky – číslo od 1 po 5
5. Hodina prednášky – číslo od 1 po 7
6. Poradové číslo prednášky, ak ich je v týždni viac – vždy hodnota 0
7. Meno učiteľa – celé meno (nepovinné)
8. Fixovanie – písmeno F, ak je prednáška fixovaná (nepovinné)

Piaty riadok obsahuje:

1. Znak & na odlíšenie cvičenia
2. Počet hodín cvičenia
3. Miestnosť cvičenia – označenie miestnosti
4. Deň cvičenia – číslo od 1 po 5
5. Hodina cvičenia – číslo od 1 po 7
6. Číslo krúžku, ktorý je cvičenie ručené
7. Poradové číslo cvičenia, ak ich je v týždni viac – vždy hodnota 0
8. Meno učiteľa – celé meno (nepovinné)
9. Fixovanie – písmeno F, ak je cvičenie fixované (nepovinné)

Jeden predmet môže obsahovať aj 2 riadky prednášky, a samozrejme niekoľko riadkov cvičenia, v závislosti od počtu krúžkov v danom odbore a ročníku.

3.1.3 Výber potrebných entít a atribútov

Túto fázu som robil opakovane z toho dôvodu, že v entitách som v mnohých prípadoch nachádzal nezhody s pôvodným modelom.

Analýzou zdrojového súboru dostávame prehľad o možných entitách a reláciách v budúcej databáze. V tomto procese treba vybrať tie, ktoré sú potrebné do modelu databázy a k nim aj potrebné atribúty. Nie všetky entity a atribúty sú potrebné na zobrazenie rozvrhu a stávajú sa teda zbytočnými dátami.

Celky .DATUM, .FAKULTA, .ZADAVATEL sú zbytočné, netreba ich na prehliadanie rozvrhu, .ROZSAH a .HODINY sú na Ekonomickej univerzite nemenné, preto ich netreba parsovať – v prípade potreby ich vieme dopísať jednorazovo a trvalo.

Celok .OBDOBIE s atribútmi semester a ročník, sa neskôr ukázal, že je potrebný v každej tabuľke – nevytvára teda vlastnú tabuľku, no tieto atribúty sa nachádzajú v každej existujúcej.

Z celku .KATEDRY je potrebné uložiť do tabuľky všetky atribúty. Farba katedry a farba textu sa ukladajú iba pre prípadné budúce použitie v aplikácii s grafickým rozhraním.

Z celku .MIESTNOSTI je vhodné uložiť všetky atribúty a vytvoriť tak z miestnosti samostatnú entitu

Ďalšou tabuľkou v databáze bude entita z .UCITELIA, kde sa ako atribúty použijú meno, priezvisko, titul a skratka katedry.

Z celku .PARALELKY sa vyberie ročník, názov fakulty, skratka fakulty, názov odboru, skratka odboru, názov pododboru, skratka pododboru a počet krúžkov.

Celok .PREKAZKY je špecifický, pretože tieto údaje sa okrem údaju „špecifikácia typu“ neobjavujú nikde inde, použijeme teda všetky atribúty.

.PREDMETY – potrebné bude určite označenie ročníka, fakulty, odboru, pododboru, skratka katedry, typ predmetu, názov predmetu, skratka predmetu, meno a priezvisko učiteľa, počet hodín prednášok a cvičení, číslo krúžku na prednáške, farba predmetu, farba textu, miestnosť prednášky, deň prednášky, hodina prednášky, miestnosť cvičenia, deň cvičenia, hodina cvičenia, číslo krúžku, meno cvičiaceho. Špecifickým záznamom bude jeho primárny kľúč – číslo predmetu, ktoré bude aplikácia generovať.

Po vybratí týchto entít a atribútov som dostal konceptuálny model dát.

3.1.4 Normalizácia

Rovnako ako výber entít a atribútov, aj normalizáciu som vykonal dva krát. Druhá verzia logického modelu síce nebola prijatá, no vychádzam z normalizácie, použitej práve pri tomto pokuse, pretože pri tretej konečnej podobe logického modelu som musel len upraviť niekoľko vecí z druhého modelu.

V konečnej verzii logického modelu pracujem v porovnaní s predchádzajúcimi modelmi len s atribútmi nevyhnutne potrebnými pre potreby zobrazovania rozvrhu.

Normalizácia

Zoznam atribútov	1.NF	2.NF	3.NF	Logický model dát
meno_katedry skratka_katedry farba_katedry farba_textu typ_miestnosti nazov_miestnosti kapacita_miestnosti priezvisko_ucitela meno_ucitela skratka_katedry Titul typ_prekazky specifikacia_typu kvalita_prekazky meno_prekazky info_prekazky den_prekazky hodina_prekazky nazov_fakulty skratka_fakulty nazov_odboru skratka_odboru nazov_pododboru skratka_pododboru c_kruzku c_predmetu rocnik typ_predmetu nazov_predmetu skratka_predmetu priezvisko_ucitela meno_ucitela pocet_hodin_prednasok pocet_hodin_cvicenia farba_predmetu farba_textu miestnost_prednasky den_prednasky hodina_prednasky miestnost_cvicenia den_cvicenia hodina_cvicenia semester rok	skratka_katedry meno_katedry farba_katedry farba_textu *Skratka_fakulty* semester rok nazov_miestnosti typ_miestnosti kapacita_miestnosti semester rok priezvisko_ucitela meno_ucitela skratka_katedry Titul *skratka_katedry* semester rok *specifikacia_typu* den_prekazky hodina_prekazky kvalita_prekazky meno_prekazky info_prekazky typ_prekazky semester rok skratka_odboru nazov_odboru nazov_fakulty skratka_fakulty semester rok skratka_odboru skratka_pododboru nazov_pododboru nazov_fakulty skratka_fakulty semester rok skratka_odboru skratka_pododboru nazov_fakulty skratka_fakulty Rocnik semester rok	skratka_katedry meno_katedry farba_katedry farba_textu *Skratka_fakulty* semester rok nazov_miestnosti typ_miestnosti kapacita_miestnosti semester rok priezvisko_ucitela meno_ucitela skratka_katedry Titul *skratka_katedry* semester rok *specifikacia_typu* den_prekazky hodina_prekazky kvalita_prekazky meno_prekazky info_prekazky typ_prekazky semester rok skratka_odboru nazov_odboru nazov_fakulty skratka_fakulty semester rok skratka_odboru skratka_pododboru nazov_pododboru nazov_fakulty skratka_fakulty semester rok skratka_odboru skratka_pododboru nazov_fakulty skratka_fakulty Rocnik semester rok semester rok	skratka_katedry meno_katedry farba_katedry farba_textu *Skratka_fakulty* semester rok nazov_miestnosti typ_miestnosti kapacita_miestnosti semester rok priezvisko_ucitela meno_ucitela skratka_katedry Titul *skratka_katedry* semester rok *specifikacia_typu* den_prekazky hodina_prekazky kvalita_prekazky meno_prekazky info_prekazky typ_prekazky semester rok skratka_odboru nazov_odboru *skratka_fakulty* semester rok skratka_odboru skratka_pododboru nazov_pododboru nazov_fakulty skratka_fakulty semester rok *skratka_odboru* skratka_pododboru *skratka_fakulty* Rocnik semester rok	KATEDRA MIESTNOST UCITEL PREKAZKY ODBOR PODODBOR PARALELKA

		skratka_odboru skratka_pododboru nazov_fakulty skratka_fakulty Rocnik c_kruzku semester rok c_predmetu typ_predmetu nazov_predmetu skratka_predmetu pocet_hodin_prednasok pocet_hodin_cvicenia farba_predmetu farba_textu *Skratka_katedry* semester rok c_predmetu miestnost_prednasky den_prednasky hodina_prednasky *Rocnik* *skratka_fakulty* *skratka_odboru* *skratka_pododboru* *priezvisko_ucitela* *meno_ucitela* semester rok c_predmetu Rocnik skratka_fakulty skratka_odboru skratka_pododboru c_kruzku miestnost_cvicenia den_cvicenia hodina_cvicenia priezvisko_ucitela meno_ucitela semester Rok *c_predmetu* *miestnost_prednasky* *den_prednasky* *hodina_prednasky* *Rocnik* *skratka_fakulty* *skratka_odboru* *skratka_pododboru* *c_kruzku* Semester rok	skratka_odboru skratka_pododboru nazov_fakulty skratka_fakulty Rocnik c_kruzku semester rok c_predmetu typ_predmetu nazov_predmetu skratka_predmetu pocet_hodin_prednasok pocet_hodin_cvicenia farba_predmetu farba_textu *Skratka_katedry* semester rok c_predmetu miestnost_prednasky den_prednasky hodina_prednasky *Rocnik* *skratka_fakulty* *skratka_odboru* *skratka_pododboru* *priezvisko_ucitela* *meno_ucitela* semester rok c_predmetu Rocnik skratka_fakulty skratka_odboru skratka_pododboru c_kruzku miestnost_cvicenia den_cvicenia hodina_cvicenia priezvisko_ucitela meno_ucitela semester Rok *c_predmetu* *miestnost_prednasky* *den_prednasky* *hodina_prednasky* *Rocnik* *skratka_fakulty* *skratka_odboru* *skratka_pododboru* *c_kruzku* Semester rok	*skratka_odboru* *skratka_pododboru* *skratka_fakulty* *Rocnik* c_kruzku semester rok c_predmetu typ_predmetu nazov_predmetu skratka_predmetu pocet_hodin_prednasok pocet_hodin_cvicenia farba_predmetu farba_textu *Skratka_katedry* semester rok c_predmetu miestnost_prednasky den_prednasky hodina_prednasky *Rocnik* *skratka_fakulty* *skratka_odboru* *skratka_pododboru* *priezvisko_ucitela* *meno_ucitela* semester rok c_predmetu Rocnik skratka_fakulty skratka_odboru skratka_pododboru c_kruzku miestnost_cvicenia den_cvicenia hodina_cvicenia priezvisko_ucitela meno_ucitela semester Rok *c_predmetu* *miestnost_prednasky* *den_prednasky* *hodina_prednasky* *Rocnik* *skratka_fakulty* *skratka_odboru* *skratka_pododboru* *c_kruzku* semester rok	KRUZOK PREDMET PREDNASKA CVICENIE KRUZOK_NA_ PREDNASKE

Obrázok 1 – normalizácia

3.1.5 *Vytvorenie tabuľky primárnych a cudzích kľúčov*

Viacero entít má veľmi zložitý primárny kľúč, je to spôsobené množstvom problémov, s ktorými som sa stýkal počas riešenia ďalších úloh, kedy som kvôli konzistencii musel pridať primárny kľúč do určitej tabuľky, inak vznikali duplicity. Toto je konečná podoba celého pripomienkovania zo strany programu.

Tabuľka 1 - znázornenie schémy primárnych a cudzích kľúčov

	KATEDRA	MIESTNOST	UCITELIA	FAKULTA	ODBOR	PODODBOR	PARALELKA	KRUZOK	PREDMET	PREDNASKA	CVICENIE	KRUZOK_NA_PREDNASKE	PREKAZKY
Skratka_katedry	K		C						C				
Názov_miestnost[1]		K								K	K	K	K
Priezvisko_ucitela			K							C	C		
Meno_ucitela			K							C	C		
Specifikacia_typu													K
Info_prekazky													K
Den_prekazky													K
Hodina_prekazky													K
Skratka_fakulty	C			K	K	K	K	K		C	C	K	
Skratka_odboru					K	K	K	K		C	C	K	
Skratka_pododboru						K	K	K		C	C	K	
Rocnik							K	K		C	C	K	
C_kruzku				K				K			K	K	
C_predmetu									K	K	K	K	
Den_prednasky										K		K	
Hodina_prednasky										K		K	
Den_cvicenia											K		
Hodina_cvicenia											K		
Semester	K	K	K		K	K	K	K	K	K	K	K	K
Rok	K	K	K		K	K	K	K	K	K	K	K	K

Atribúty *semester* a *rok* tvoria primárny kľúč v každej tabuľke z dôvodu ukladania každého ďalšieho zdrojového súboru do tých istých tabuliek, aby nevznikali duplicity. Tento krok spĺňa požiadavku archivácie starých semestrov a tie v prípade potreby budú pripravené na prehliadanie, bez akéhokoľvek zásahu do zdrojového kódu.

Tabuľka KRUZOK obsahuje primárny kľúč zložený z 5 častí (bez *semestra* a *roku*) z dôvodu, že hierarchia skupinovania študentov musí byť dodržaná. Niektoré časti primárneho kľúča v tejto tabuľke vyzerajú zbytočne, ale nakoniec som ich musel uviesť.

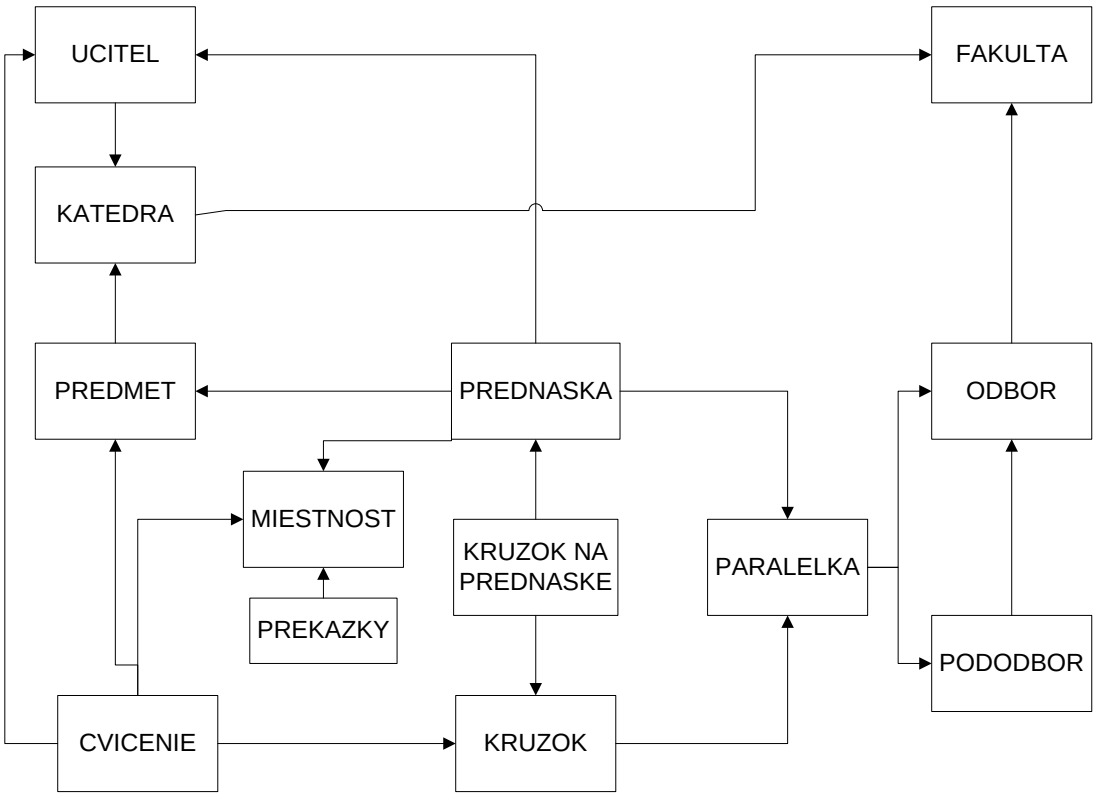
[1] Názov miestnosti je primárnym kľúčom aj pre tabuľky PREDNASKA (pod označením *miestnost_prednasky*),

Pôvodný primárny kľúč obsahoval iba *skratka_odboru, rocnik, c_kruzku*, no hneď prvý záznam o predmete v zdrojovom súbore zimného semestra obsahoval označenie paralelky, ktorá sa skladá iba zo skratky fakulty a ročníka. Pri vtedajšom primárnom kľúči to bolo teda nepostačujúce. Rovnaká situácia sa stala pri tých predmetov, ktoré sa týkajú pododborov, hoci ich nie je veľa, preto podoba primárneho kľúča vyzerá takto.

Tabuľka KRUZOK_NA_PREDNASKE bola vytvorená len kvôli jednému prípadu. Zvyčajne je prednáška určená pre celú paralelku (odbor v ročníku), v niektorých prípadoch sa však stalo, že prednáška bola určená len pre konkrétnu množinu krúžkov z danej paralelky. Preto slúži táto tabuľka na rozbitie vzťahu M:N medzi tabuľkami PREDNASKA a KRUZOK.

3.1.6 Entitno – relačný diagram

ERD diagram odzrkadľuje logický dátový model vytvorený normalizáciou atribútov. Šípky zobrazujú závislosti na vyšších entitách.



Obrázok 1 - Entitno-relačný diagram

3.1.7 Vytvorenie tabuliek databázy

Fyzický model dát vychádza z toho logického, ktorý je naznačený v ERD diagrame a tabuľke primárnych a cudzích kľúčov. Vytvorenie schémy databázy som realizoval priamo v prostredí Eclipse v SQL súbore, keďže tento editor umožňuje aj samotné pripojenie do databázy, ktorá je fyzicky vytvorená na pevnom disku. Na hornom panely tohto editoru možno vidieť stav pripojenia a databázu, ktorej sa to týka. V jazyku SQL som vytvoril tabuľky pomocou príkazu CREATE TABLE, s definovaním mien tabuliek totožných s tými v ERD modely. Za ním nasledujú mená stĺpcov s ich dátovým typom a na konci sú definované constrainty – primárne kľúče a cudzie kľúče. Použil som aj klauzulu ON DELETE CASCADE, ktorá pri vymazaní údajov vo vyššej tabuľke v hierarchii spôsobí kaskádové vymazanie údajov v nižších tabuľkách, ak sú prepojené cudzím kľúčom. Takto bolo vytvorených všetkých 13 tabuliek.

```
CREATE TABLE UCITEL
 (PRIEZVISKO_UCITELA VARCHAR(30) NOT NULL,
  MENO_UCITELA VARCHAR(20) NOT NULL,
  KATEDRA_UCITELA VARCHAR(5) NOT NULL,
  TITUL_UCITELA VARCHAR(10),
  SEMESTER VARCHAR(17) NOT NULL,
  ROK VARCHAR(11) NOT NULL,
  PRIMARY KEY(MENO_UCITELA, PRIEZVISKO_UCITELA, SEMESTER, ROK),
  FOREIGN KEY(KATEDRA_UCITELA, SEMESTER, ROK) REFERENCES KATEDRA
(SKRATKA_KATEDRY, SEMESTER, ROK) ON DELETE CASCADE);
```

3.2 Testovanie databázy

Pri testovaní databázy som postupoval podľa požiadaviek na databázu.

3.2.1 Naplnenie tabuliek testovacích údajov

Naplnenie sa realizuje vkladacími príkazmi INSERT INTO TABLE. Vybral som ľubovoľné údaje, prípadne vymyslené na otestovanie funkčnosti. Keďže fakúlt na našej univerzite je obmedzené množstvo, naplnil som tabuľku FAKULTA na pevno už pri testovaní údajmi o existujúcich fakultách. Ostatné tabuľky som napĺňal len dočasne a po testoch boli vymazané. Takisto som musel počítať s málo sa vyskytujúcimi údajmi, preto aj v nasledujúcom príklade uvádzam ako skratku pododboru ako netradičný údaj. V tabuľkách PARALELKA a KRUZOK je *skratka_pododboru* časťou primárneho kľúča.

```
INSERT INTO KRUZOK_NA_PREDNASKE (C_KRUZKU, SKRATKA_ODBORU, SKRATKA_PODODBORU, ROCNIK,
 MIESTNOST_PREDNASKY, DEN_PREDNASKY, HODINA_PREDNASKY)
VALUES (1, 'HI', '0', 2, 'B1_10', 3, 4),
       (2, 'HI', '0', 2, 'B1_10', 3, 4),
       (1, 'HI', '0', 2, 'A7_12', 3, 2),
       (2, 'HI', '0', 2, 'A7_12', 3, 2),
       (6, 'UC', '0', 2, 'B1_04', 4, 5),
       (1, 'EMP', 'VMZ', 4, 'B1_02', 2, 7),
       (3, 'EMP', 'VMZ', 4, 'B1_02', 2, 7);
```

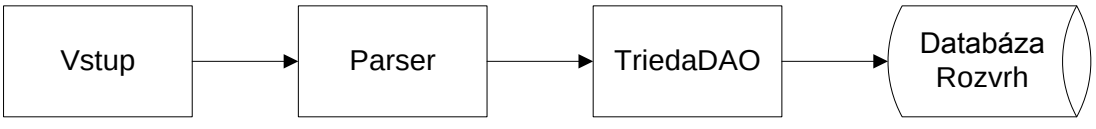
3.2.2 Testovanie pomocou výberových dotazov

Po naplnení databázy testovacími údajmi som použil výberové dotazy SELECT FROM. Vyberal som údaje podľa požiadaviek používateľa, teda rozvrh pre paralelku, krúžok, učiteľa, miestnosť a takisto aj voliteľné predmety. Na výber všetkých potrebných atribútov som tabuľky spájal pomocou equijoinových spojení. Testovanie bolo zamerané predovšetkým na schopnosť konektorov pristupovať k jednotlivým tabuľkám. Testovanie dopadlo úspešne, databáza teda bola pripravená na použitie. Ako príklad uvádzam testovanie výberu údajov pre paralelku.

```
SELECT  PAR.SKRATKA_ODBORU, P.NAZOV_PREDMETU, PR.MIESTNOST_PREDNASKY,
PR.DEN_PREDNASKY, PR.HODINA_PREDNASKY, U.MENO_UCITELA, U.PRIEZVISKO_UCITELA
FROM PARALELKA AS PAR, PREDNASKA AS PR, PREDMET AS P, UCITEL AS U
WHERE PAR.SKRATKA_ODBORU = PR.SKRATKA_ODBORU
  AND PAR.SKRATKA_PODODBORU = PR.SKRATKA_PODODBORU
  AND PAR.ROCNIK = PR.ROCNIK
  AND PR.C_PREDMETU = P.C_PREDMETU
```

3.3 Načítanie zdrojového súboru

Od tohto procesu je úloha riešená pomocou aplikácie v jazyku Java. Táto aplikácia má nasledovnú štruktúru tried - táto podoba je však zovšeobecnená. Šípky znamenajú smer exekúcie programu.



Obrázok 2 - zovšeobecnená podoba štruktúry tried databázovej aplikácie

Načítanie zdrojového súboru sa realizuje v bloku Parser znázornenom na diagrame, ktorý je všeobecným označením pre množinu tried slúžiacich na parsing zdrojového

súboru. Rovnako blok TriedaDAO je zovšeobecneným zobrazením množiny tried, pristupujúcich do databázy. V skutočnosti Parser predstavuje rozhranie medzi vstupnou triedou a parsovacími triedami, ktoré sú určené pre konkrétnu entitu zvlášť. Týchto tried je 6 rovnako ako celkov v zdrojovom súbore určených na spracovanie. Z toho som odvodil aj názvy pre tieto triedy: ParserKatedry, ParserMiestnosti, ParserUcitela, ParserParalelky, ParserPrekazky a ParserPredmetu.

Načítanie je realizované týmto spôsobom: vstupná trieda vyzve používateľa na zadanie cesty ku zdrojovému súboru, uloženému na hard disku. Túto cestu uloží ako textový reťazec a pošle tento String v argumente metódy *parsuj* definovanej v rozhraní Parser. Táto metóda podľa cesty načíta súbor pomocou triedy Scanner, alebo LineNumberReader, ktorá umožňovala získavať čísla riadkov v súbore. Čísla riadkov som však napokon nepotreboval vedieť, ale trieda slúži na čítanie súboru spoľahlivo, tak som ho v troch triedach ponechal, pričom vo zvyšných som aplikoval triedu Scanner.

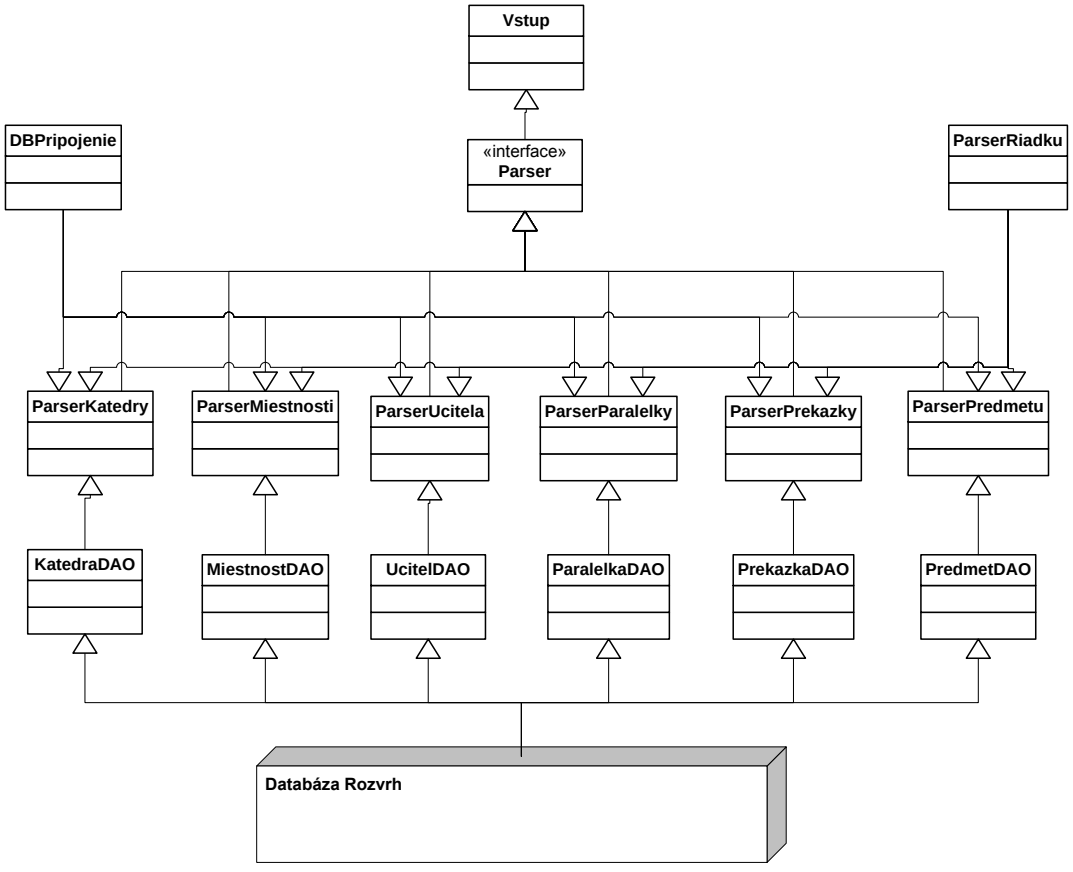
3.4 Parsing zdrojového súboru

Parsovanie sa deje v tej istej metóde ako načítanie zdrojového súboru. Trieda Scanner, ako aj LineNumberReader obsahuje metódy (*nextLine, resp. readLine*), ktoré vracajú celý jeden riadok vo forme Stringu, čo zásadne uľahčuje prácu, pretože takto sa dá dostať ďaleko menší textový reťazec, určený na spracovanie. Obe triedy však pracujú od začiatku súboru po koniec a nemožno znovu načítať predchádzajúci riadok. Vo všetkých šiestich triedach sa najskôr vyčlení riadok, ktorý obsahuje celok ".OBDOBIE", pretože nasledujúci riadok obsahuje údaj o semestri a školskom roku, ktoré sú primárnym kľúčom vo všetkých tabuľkách. Skenovacia metóda sa k nemu dostane pomocou cyklu while. Vykoná sa to porovnaním aktuálneho riadku v podmienkovej časti cyklu, pomocou metódy *contains* alebo *equals* so Stringovým argumentom, ktoré vracajú hodnotu true alebo false.

Medzi týmito metódami je malý rozdiel: *contains* zistí, či zadaný argument metódy sa nachádza v danom porovnávanom reťazci (v tomto prípade riadku zdrojového súboru). Metóda *equals* porovnáva celý riadok s argumentom. Dôležitou je najmä pri porovnávaní krátkych Stringov – používal som ju na zistenie, či je aktuálny riadok prázdny, čo by v prípade *contains* nebolo možné, pretože každý riadok končí „ničím" a teda vždy by vracala hodnotu true.

Riadok s obdobím sa posiela ako argument v konštruktore DAO triedy, kde sa rozdelí na semester a rok pomocou metódy *substring* z triedy String, kde stačí definovať poradové číslo symbolu, od ktorého chceme rozdeliť String, prípadne aj konečného symbolu, ak nechceme posledné symboly reťazca. V zdrojovom súbore býva vždy uvedené v tomto riadku "zimný semester 2010/2011" alebo "letný semester 2010/2011". Oba výrazy majú rovnaký počet znakov, čo zjednodušuje situáciu a môžem teda spoľahlivo nastaviť poradové čísla symbolov pre semester na (0,15), kde 0 je číslo začiatočného symbolu a 15 číslo konečného symbolu. Rok teda bude obsahovať iba začiatočné číslo, odkiaľ chcem rozdeliť reťazec teda 15.

Keďže sa zdrojový súbor načíta vždy celý naraz, parsing sa vykoná takisto celý pre jeden blok v súbore. Nie je potrebné volať metódu parsuj viackrát, preto stačí aby bola vytvorená jedna inštancia tried Parser, čo znamená, že objekt tejto triedy je **návrhový vzor Singleton**.



Obrázok 3 - diagram tried databázovej časti aplikácie

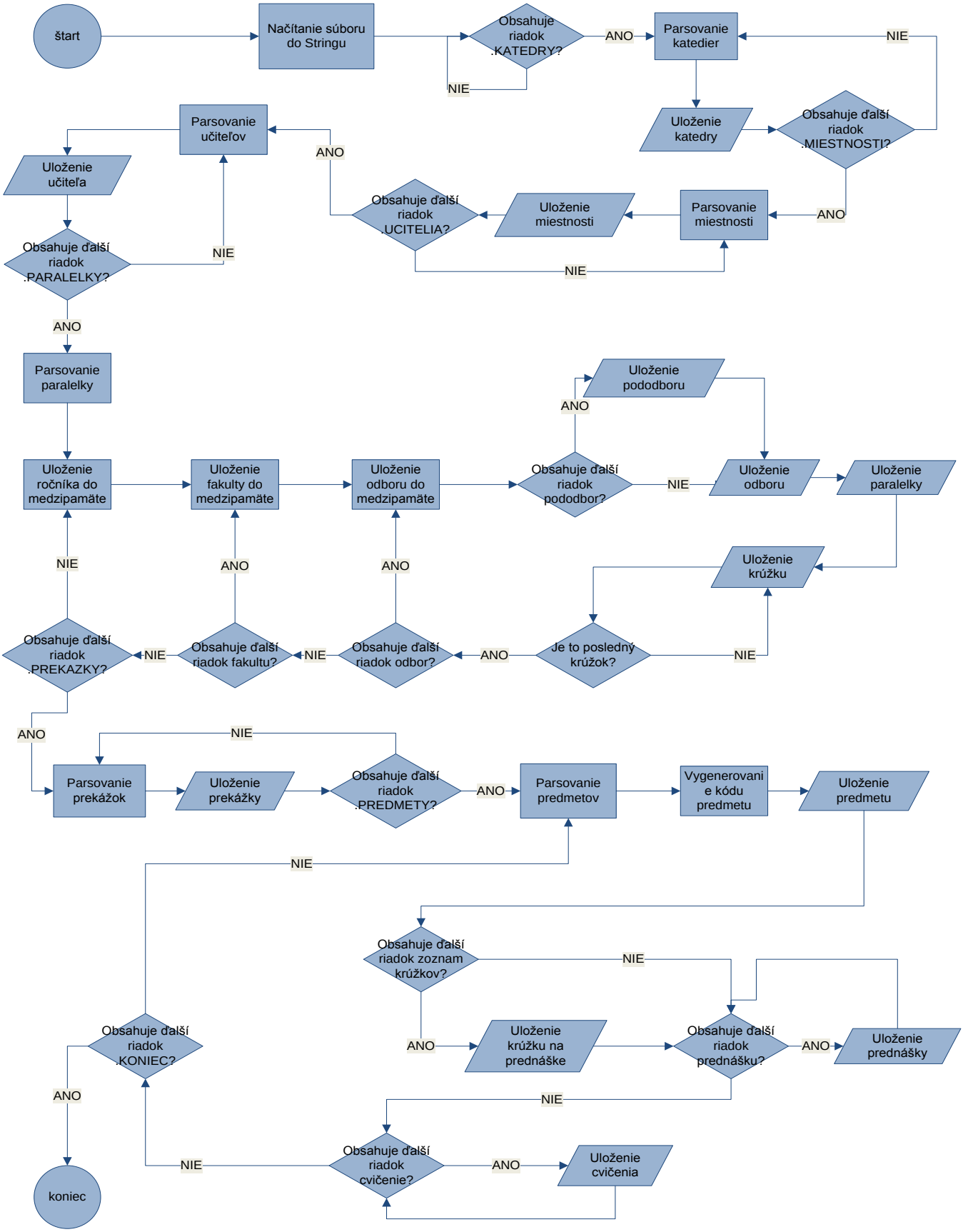
3.4.1 Vytvorenie vhodného algoritmu pre automatické parsovanie zdrojového súboru

Aby sa mohli údaje ukladať automaticky, treba postupovať systematicky. Bolo by chybou pokúšať sa ukladať do databázy cvičenie s cudzím kľúčom *c_kruzku* a nemať pritom uložený krúžok. Aj zdrojový súbor je štruktúrovaný tak, aby sa ukladali najskôr vyššie entity. Poradie volania tried parsera je teda totožné s poradím blokov v zdrojovom súbore:

```
Parser katedra = ParserKatedry.dajInstanciu();
Parser miestnost = ParserMiestnosti.dajInstanciu();
Parser ucitel = ParserUcitela.dajInstanciu();
Parser paralelka = ParserParalelky.dajInstanciu();
Parser prekazka = ParserPrekazky.dajInstanciu();
Parser predmet = ParserPredmetu.dajInstanciu();
```

Každej inštancii rozhrania Parser je pridelená cesta ku zdrojovému súboru, avšak metóda *parsuj* si v danej triede implementujúcej rozhranie vždy nájde v texte svoj príslušný blok, ktorý musí rozparsovať. Toto hľadania sa uskutočňuje porovnávaním riadkov s ".ENTITA" pomocou metódy *contains*. Až narazí na tento riadok, dovtedy ostatné riadky jednoducho preskakuje. Porovnanie sa uskutočňuje v podmienkovej časti cyklu while. Po splnení podmienky na ukončenie preskakovacieho cyklu prejde chod programu k podobnému cyklu while, ktorý však v podmienke obsahuje porovnanie riadku s nasledujúcim názvom bloku. Napríklad ak parsujeme .KATEDRY a nasledujúcim blokom sú .MIESTNOSTI, cyklus sa vykonáva dovtedy, kým sa v riadku objaví názov bloku miestností.

V nasledujúcom vývojovom diagrame je zobrazená postupnosť s akou dochádza v programe k parsingu. Obdĺžnikový rámec popisuje proces, kosoštvorec rozhodovanie, kruh začiatok alebo koniec úlohy a kosodĺžnik znamená ukladanie dát do databázy.



Obrázok 4 - algoritmus spracovania zdrojového súboru v databázovej aplikácii

3.4.2 *Pomocná trieda ParserRiadku*

Má za úlohu odľahčiť beztak robustné triedy Parser, ktoré potrebujú vybrať z riadku špecifikované údaje. ParserRiadku obsahuje 5 metód:

```
• public String parsujRiadok(String riadok, int cisloPrvehoSymbolu, char zaciatok, char koniec)
• public String parsujRiadok(String riadok, int cisloPrvehoSymbolu, char koniec)
• public int parsujCislo(String riadok, int cisloPrvehoSymbolu, char zaciatok, char koniec)
• public int parsujCislo(String riadok, int cisloPrvehoSymbolu, char koniec)
• public int parsujKruzok(String riadok)
```

porovnáva jednotlivé znaky. Deje sa to pomocou cyklu for a metódy *charAt* z triedy String, ktorá porovná znak v reťazci s integerom zadaným v argumente. Keď sa dostane k začiatočnému znaku, uloží ho do pomocného Stringu a každé ďalšie ukladá až po konečný znak. Toto by nebolo možné pomocou preddefinovanej metódy *substring*, pretože v nej sa parsuje len pomocou poradových čísel znakov, ale v tomto prípade je počet znakov v jednom riadku vysoko variabilný, preto treba porovnávať znaky s určitosťou sa nachádzajúce v riadkoch. Môže sa stať, že sa takýto znak nachádza v riadku viackrát, tento prípad je preto ošetrený číslom začiatočného symbolu, odkiaľ začne porovnávací cyklus v riadku. Kvôli variabilnosti počtu znakov v riadku sa často uvádza číslo získané metódou *length* z triedy String, ktorá vracia dĺžku reťazca.

Obe metódy *parsujCislo* pracujú rovnako, ale nakoniec sa údaj pomocou metódy *parseInt* z triedy Integer prekonvertuje na celočíselný dátový typ.

Posledná metóda *parsujKruzok* je určená len pre parsovaciu triedu ParserParalelky, ktorá spracováva riadky typu:

```
$$ Hospodarska_informatika HI 50/1-2
```

kde krúžkom je údaj za lomenom. Ak sa za lomenom nachádzajú čísla oddelené pomlčkou, ide o rozsah krúžkov od 1 po n a vracia sa vždy to druhé číslo ako počet krúžkov v danom ročníku na danom odbore.

3.4.3 Trieda ParserKatedry

Všetky parsujúce triedy môžu vytvoriť iba jednu inštanciu, čo je zabezpečené metódou *dajInstanciu*, ktorej návratová hodnota je typu ParserKatedry. Trieda načíta zdrojový súbor pomocou triedy LineNumberReader.

Ako už bolo spomenuté, trieda ParserKatedry, ako aj všetky ostatné triedy implementujúce rozhranie Parser, najskôr nájde .OBDOBIE a pošle ho konštruktoru triedy KatedraDAO aj so Statementom získaným pripojením do databázy z vytvorenej triedy DBPripojenie, ktorá má za úlohu pripojiť sa do databázy pomocou drivera, prihlasovacieho mena a hesla a vráti Statement, ktorý umožňuje implementovať SQL príkazy v javovskej triede. Následne pokračuje hľadaním kľúčového riadku .KATEDRY, odkiaľ začína parsing.

Priamo v podmienkovej časti ďalšieho cyklu while sa vkladá do pomocného Stringu nový riadok, ktorý sa vzápätí porovnáva s hodnotou .MIESTNOSTI. Pokým nový riadok neobsahuje miestnosti, bude sa vykonávať cyklus.

V prípade objavenia prázdneho riadku pomocou porovnania v príkaze if, pokračuje cyklus načítaním ďalšieho riadku.

Ak riadok nie je prázdny, volá sa metóda *parsujRiadok* pre získanie mena katedry a následne aj skratky katedry. Katedru treba podľa štruktúry databázy priradiť pod určitú fakultu. Prepojenie medzi fakultami a katedrami však v zdrojovom súbore nie je definované, preto som vytvoril akýsi porovnávací blok, ktorý nájde skratku katedry uloženú v premennej, v zozname skratiek a priradí jej príslušnú skratku fakulty. Môže sa však stať, že vznikne nová katedra, ktorá nemá skratku na zozname – program vyzve používateľa ku zadaniu mena a skratky katedry a skratky fakulty ručne z klávesnice.

Keď už sú všetky tri údaje kompletné, chod programu volá metódu *vlozKatedru* z triedy KatedraDAO. Vykonávanie cyklu sa skončí po nájdení riadku, ktorý obsahuje .MIESTNOSTI a chod programu sa vráti späť do vstupnej triedy.

3.4.4 Trieda ParserMiestnosti

Správa sa podobne ako trieda ParserKatedry. Vytvorí inštanciu, načíta zdrojový súbor, získa semester a rok, pošle ho so Statementom do konštruktora triedy MiestnostDAO, spustí ďalší cyklus while, ktorý sa bude vykonávať až kým, nenarazí na riadok obsahujúci .UCITELIA.

Z riadku sa pomocou metódy *charAt* získa typ miestnosti (jedná sa o prvý znak v riadku, preto je možné použiť *charAt*), názov miestnosti sa získa metódou *parsujRiadok* a kapacita miestnosti pomocou *parsujCislo*.

Následne sa pošlú všetky údaje do triedy MiestnostDAO metódou *vlozMiestnost* a chod programu sa vráti do vstupnej triedy.

3.4.5 Trieda ParserUcitela

Podobne ako trieda ParserKatedry a ParserMiestnosti. Na získanie mena, priezviska, katedry a titulu učiteľa použije štyrikrát metódu *parsujRiadok.* Následne volá metódu *vlozUcitela* z triedy UcitelDAO.

3.4.6 Trieda ParserParalelky

Na parsovanie fakulty, odboru, pododobru, paralelky a krúžku bolo potrebné použiť zložitejší mechanizmus. Štruktúra bloku .PARALELKY je takáto:

```
4_rocnik 4
 $ Narodohospodarska_fakulta NHF
  $$ Bankovnictvo BANK 110/1-5
```

Parsovať ročník nie je potrebné, pretože ich počet je jasne stanovený, stačí iba zistiť, v ktorom riadku sa nachádza ročník. O vytváranie čísiel ročníkov sa stará cyklus for, ktorý sa vykoná presne 5 krát. Údaj o ročníku musí byť uložený do premennej, kvôli nižším entitám.

Ak sa v riadku nachádza znak "$", znamená to označenie fakulty. Najskôr sa zozparsuje skratka fakulty pomocou *parsujRiadok* a kvôli paralelkám obsahujúcim iba údaj fakultu a ročník, sa do tabuľky ODBOR pomocou metódy *vlozNulovyOdbor* vloží odbor so skratkou a názvom rovným 0. Rovnako sa vloží aj nulový pododbor pomocou *vlozNulovyPododobor*. Bez toho by nebolo možné vložiť paralelku, ktorá obsahuje iba fakultu a ročník. To sa vykoná pomocou *vlozParalelku*. Všetky vkladacie metódy sú z triedy ParalelkaDAO. Vkladať fakultu netreba, pretože všetky fakulty boli jednorazovo uložené pri testovaní databázy. Je však potrebná pre vkladanie menších entít, preto musí byť uložená v premennej.

Ak riadok obsahuje znak "$$", znamená to označenie odboru. Najskôr sa jednoducho rozparsuje názov a skratka odboru a tento odbor sa vloží do tabuľky ODBOR metódou *vlozOdbor*. Ak však súčasný riadok obsahuje aj znak "/" (čo sa zistí porovnaním

metódou *contains*), odbor neobsahuje pododbor a rozparsuje sa krúžok pomocou metódy *parsujKruzok,* ktorá vráti počet krúžkov v danom ročníku a odbore. Keďže neobsahuje pododbor, uloží sa nulový pododbor a následne paralelka. Pomocou cyklu for sa uložia všetky krúžky metódou *vlozKruzok.*

Ak riadok obsahuje znak "$$$", znamená to označenie pododboru. Rozparsuje sa názov pododboru, skratka pododboru a počet krúžkov. Vloží sa pododbor, paralelka a rovnakým spôsobom aj všetky krúžky.

3.4.7 Trieda ParserPrekazky

Pracuje podobne ako prvé tri parsovacie triedy. Rozparsuje sa typ prekážky, špecifikácia typu, meno prekážky, info prekážky, kvalita prekážky, deň prekážky a hodina prekážky, prípadne rozsah prekážky, ak sa v riadku nachádza znak "-", čo uloží danú prekážku pomocou *vlozPrekazku*. viackrát v závislosti od rozsahu. Spomínaná metóda sa nachádza v triede PrekazkaDAO.

3.4.8 Trieda ParserPredmetu

Je to trieda najmohutnejšia zo všetkých parsovacích tried, ktorá okrem zložitej štruktúry predmetov v zdrojovom súbore, musí akceptovať rôzne výnimky nachádzajúce sa v súbore. V prílohe je uvedený zoznam možných prípadov, ktoré sa môžu objaviť v zdrojovom súbore. Tu je príklad typického záznamu o predmete:

```
1_NHF_FBI 32001/1: KM: P V | Matematika_A MATA Fecenko_Jozef 1/1 160 ;3
~ 725 0
$1 B1_05 2 1 0 Fecenko_Jozef :F
&1 C1_07 1 2 1 0
&1 B1_06 3 1 2 0
&1 B1_08 3 4 3 0
&1 C1_07 1 2 4 0
&1 B1_06 3 5 5 0
&1 B1_06 3 5 6 0
&1 B1_06 3 1 7 0
&1 B1_08 3 4 8 0
```

Je rozdelený do viacerých riadkov, preto som aj kód upravil do blokov, pričom, jeden blok spracováva jeden riadok. Vkladanie nových riadkov sa realizuje dynamicky – vždy po dokončení prvého riadku. Čo sa týka cvičení, tie sú definované spôsobom jedno cvičenie – jeden riadok. Keď sa vloží riadok do Stringu, treba zistiť, čo sa v ňom nachádza.

Ak sa v ňom nenachádza znak “&“, ktorý symbolizuje cvičenie, prechádza cyklus, ktorý sa končí až riadkom s textom “.KONIEC“, k parsovaniu ďalšieho predmetu.

Z prvého riadku sa získajú tieto údaje: ak program zistí, že sa jedná o ďalší predmet, vygeneruje mu číslo predmetu, ďalej sa vyparsuje označenie paralelky (ročník, fakulta, odbor, pododbor), skratka katedry, typ predmetu, názov predmetu, skratka predmetu, učiteľ a počet hodín prednášok a cvičení.

Ak druhý riadok obsahuje znak “#“, k danej prednáške prislúcha zoznam krúžkov. Tu sa potom zisťuje prvé a posledné číslo krúžku a týmto rozsahom sa naplní celočíselné pole, ktoré sa bude neskôr vkladať do tabuľky KRUZOK_NA_PREDNASKE. Ak sa toto pole pri jednom predmete naplní, treba ho do parsingu ďalšieho predmetu vyprázdniť, pretože by program ukladal krúžky na prednáške aj vtedy, keď prednáška nie je delená.

Z tretieho riadku sa získava farba predmetu a farba textu. Údaje na vloženie predmetu do tabuľky PREDMET sú kompletné. Volá sa preto metóda *vlozPredmet*.

Štvrtý riadok zistí, či sa v ňom nachádza znak “$“ symbolizujúci prednášku a následne parsuje miestnosť, deň a hodinu prednášky. Tieto údaje pošle triede PredmetDAO metódou *vlozPrednasku* do tabuľky PREDNASKA. V prípade, ak je vytvorený zoznam krúžkov na danú prednášku, volá sa preťažená metóda *vlozPrednasku,* ktorá vkladá údaje do tabuľky KRUZOK_NA_PREDNASKE.

Piaty riadok zistí, či sa v ňom nachádza znak “&“ symbolizujúci cvičenie čo sa vykonáva v podmienkovej časti cyklu while. Takto sa rozparsuje miestnosť, deň a hodina cvičenia a krúžok, pre ktorý je cvičenie určené. Následne sa cvičenie pošle triede PredmetDAO metódou *vlozCvicenie.* V prípade, že sa to nepodarí, došlo ku situácii, kedy sa program snaží vložiť do tabuľky CVICENIE cvičenie, ktoré sa odkazuje na neexistujúci krúžok. Táto situácia sa môže stať ak sa jedná o voliteľný predmet, kde je rozdelenie krúžkov iné ako je uložené pre danú paralelku. V tom prípade sa priamo v triede PredmetDAO odchytí výnimka a zavolá metóda *vlozKruzok* z triedy ParalelkaDAO a krúžok sa vloží. Následne možno dodatočne vložiť cvičenie.

3.4.9 Testovanie parsovacieho algoritmu

Všetky triedy som priebežne testoval, čo mi napĺňalo tabuľky databázy priebežne. Údaje som pri testovaní celého parsingu vymazal a spustil celý parsovací algoritmus. Test dopadol úspešne na tretí pokus a jedným spustením bola naplnená celá databáza. Pri

neúspešných pokusoch som najskôr opravil chybu a vymazal údaje, ktoré sa stihli vložiť a až potom som testoval znovu.

3.5 Trvalé jednorazové uloženie zdrojových údajov

Ukladanie dát zabezpečujú triedy pristupujúce do databázy (DAO – database access object). Je ich 6 a každá ukladá iné záznamy, pričom sú členené podľa parsovacích tried, či štruktúry zdrojového súboru. Každá z týchto tried dostane v konštruktore objekt triedy Statement a obdobie, ktoré sa priamo v tele konštruktora rozparsuje na semester a rok. Pomocou objektu triedy Statement môžeme metódou *execute* implementovať SQL dotazy a vložiť tak údaje, ktoré sú zatiaľ uložené v premenných.

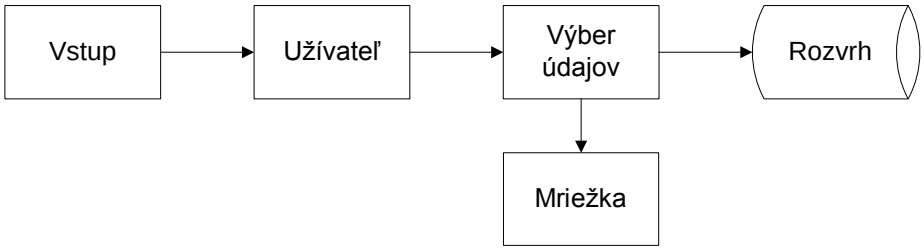
V nasledujúcej tabuľke je znázornený prehľad DAO tried, ktoré vkladajú definované entity do ktorých tabuliek.

Tabuľka 2 - zobrazenie príslušnosti entít k ukladajúcim DAO triedam

DAO trieda	Entita na vloženie	Tabuľka
KatedraDAO	Katedra	KATEDRA
MiestnostDAO	Miestnosť	MIESTNOST
UcitelDAO	Učiteľ	UCITEL
ParalelkaDAO	Odbor Pododbor Paralelka Krúžok	ODBOR PODODBOR PARALELKA KRUZOK
PrekazkaDAO	Prekážka	PREKAZKY
PredmetDAO	Predmet Prednáška Cvičenie Krúžok na prednáške	PREDMET PREDNASKA CVICENIE KRUZOK_NA_PREDNASKE

3.6 Vytvorenie používateľskej časti aplikácie

Používateľskú časť som štruktúroval do 4 častí:

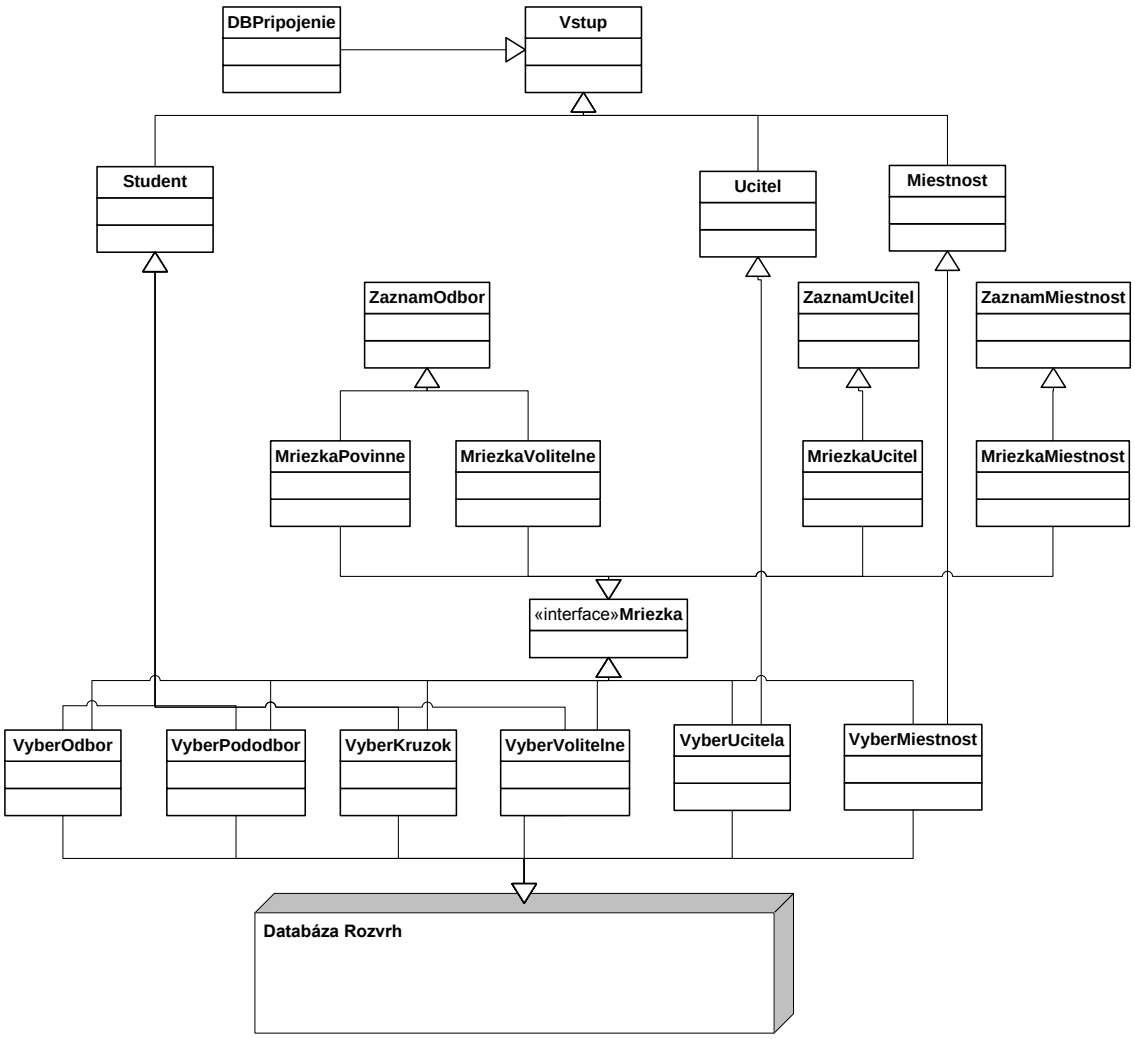


Obrázok 5 - beh používateľskej časti aplikácie

Kde Vstup je samostatná trieda, ktorá dotazuje užívateľa aplikácie, aký semester a rok by chcel prehliadať a aký rozvrh chce zobraziť. Ten má možnosť si vybrať z troch druhov: *student, ucitel, miestnost*, ktoré následne volajú objekty tried s rovnakými menami, ktoré som v diagrame zovšeobecnil pod pojmom Užívateľ. Vybratie užívateľa sa realizuje načítaním jednej z týchto troch možností z klávesnice.

Pre prípad nesprávneho zapísania mena jednej z možností, je táto trieda ošetrená niekoľkými variáciami daného slova. Napríklad *zimný semester* obsahuje 16 spôsobov ako zapísať dané slovo a výsledok sa dostaví ("zimný semester", "zimny semester", "Zimný", "ZIMNY", "z", "zimny_semester" a pod.). Rovnako sú ošetrené aj výrazy *letný semester, študent, učiteľ, miestnosť, áno, nie.*

V časti Užívateľ sa bližšie špecifikujú požiadavky na zobrazenie. V triede Student je potrebné zadať ročník, skratku fakulty a odbor a následne sa zobrazí rozvrh hodín pre daný odbor v ročníku. Program sa následne opýta užívateľa, či chce zobraziť rozvrh pre konkrétny krúžok a následne aj či zobraziť voliteľné predmety. Trieda Ucitel vyžaduje od používateľa zadanie mena a priezviska a trieda Miestnost iba názov miestnosti. Každá z týchto troch tried, ktorých jedinou metódou je *vyber*, volá inštanciu tried z časti Výber údajov.



Obrázok 6 - diagram tried používateľskej časti aplikácie

3.6.1 *Spôsob výberu údajov z databázy*

V diagrame znázornený blok Výber údajov je zovšeobecnením pre sadu výberových tried. Trieda z tohto bloku obsahuje vždy metódu *zobrazRozvrh* a jej hlavnou úlohu a tou je výber požadovaných údajov z databázy pomocou SQL dotazu zapísaného v argumente metódy *executeQuery*. Argumenty metód *zobrazRozvrh* tvoria identifikátory odboru, ročníku, fakulty, čísla krúžku, mena a priezviska učiteľa či názov miestnosti, ktoré sú potrebné na určenie konkrétnych položiek z tabuľky. Návratovou hodnotu je objekt triedy ResultSet, ktorá umožňuje spracovať vybraté údaje. Menšou úlohou je zobrazenie názvu odboru, mena učiteľa, či miestnosti ako informáciu o aký rozvrh sa jedná, zobrazenú mimo mriežky rozvrhu. Pomocou rozhrania Mriezka sa volajú triedy implementujúce toto rozhranie.

Výber údajov podľa odboru

Trieda VyberOdbor pristupuje do databázy v metóde *zobrazRozvrh*, volajúcej metódu *executeQuery*, ktorej argumentom je SQL príkaz SELECT. Ten spája tabuľky PREDNASKA a PREDMET, pretože pre odbor potrebujeme zobrazovať údaje skratka_predmetu, priezvisko_ucitela, miestnost_prednasky, den_prednasky, hodina_prednasky, typ_predmetu a či rozlišovač prednášky od cvičenia. Vybraté údaje sa uložia do objektu triedy ResultSet a následne sa tento objekt pošle na zobrazenie do mriežky v triede MriežkaPovinne volaním metódy *zobrazMriezku*.

V príkaze sa nachádza vnorený SELECT, ktorý má za úlohu vybrať tie údaje, ktoré idú do množinového mínusu. Tabuľka KRUZOK_NA_PREDNASKE je určená len pre krúžky, avšak rovnaké prednášky sú uložené aj v tabuľke PREDNASKA, preto ich treba vypustiť z výsledkov.

```
rs = st.executeQuery("SELECT P.SKRATKA_PREDMETU, PR.PRIEZVISKO_UCITELA,
                      PR.MIESTNOST_PREDNASKY AS MIESTNOST, PR.DEN_PREDNASKY AS DEN,
                      PR.HODINA_PREDNASKY AS HODINA, P.TYP_PREDMETU, 'P' AS P_C "
                     +"FROM PREDNASKA AS PR, PREDMET AS P "
                     +"WHERE PR.C_PREDMETU = P.C_PREDMETU "
                     +"AND P.ROK = '"+rok+"' "
                     +"AND P.SEMESTER = '"+semester+"' "
                     +"AND PR.ROK = '"+rok+"' "
                     +"AND PR.SEMESTER = '"+semester+"' "
                     +"AND P.TYP_PREDMETU = 'P' "
                     +"AND PR.SKRATKA_ODBORU = '"+odbor+"' "
                     +"AND PR.ROCNIK = "+rocnik+" "
                     +"AND PR.C_PREDMETU NOT IN (SELECT K.C_PREDMETU "
                                               +"FROM KRUZOK_NA_PREDNASKE AS K "
                                               +"WHERE K.ROK = '"+rok+"
                                               +"AND K.SEMESTER = '"+semester+"' "
                                               +"AND K.SKRATKA_ODBORU = '"+odbor+"'"
                                               +"AND K.ROCNIK = "+rocnik+") ");
```

Výber údajov podľa pododboru

Postupuje sa analogicky – v triede VyberPododbor sa v metóde *zobrazRozvrh* volá metóda *executeQuery* z triedy Statement, spájajú sa tabuľky PREDNASKA a PREDMET a vyberajú tie isté údaje ako v prípade odboru, len dôležitým podmienkovým údajom je skratka_pododboru. Vybraté údaje v objekte ResultSet triedy sa pošlú do triedy MriezkaPovinne metódou *zobrazMriezku*.

Výber údajov podľa krúžku

Postup volania metód a inštancii tried je rovnaký ako v predchádzajúcich prípadoch, rozdiel však obsahuje príkaz SELECT. Ten musí pomocou dvoch príkazov UNION spojiť údaje z tabuľky CVICENIE, KRUZOK_NA_PREDNASKE a (PREDNASKA – KRUZOK_NA_PREDNASKE). Spomínaná tabuľka (KNP) obsahuje len tie prednášky pre krúžky, ktoré sú delené. Použitím iba tejto tabuľky by študent nedostal kompletný rozvrh hodín pre svoj krúžok, pretože by dostal len tie prednášky, ktoré sú delené pre jednotlivé krúžky. To teda treba doplniť o údaje z tabuľky PREDNASKA, ktorá však zároveň obsahuje ja údaje z tabuľky KNP a tak je potrebný opäť množinový mínus. Až potom je možné dostať správny rozvrh, ktorý sa pošle do mriežky na zobrazenie.

Výber údajov podľa voliteľných predmetov

Logika sa opäť príliš nelíši, rozdiel je v tom, že v príkaze SELECT sa ako podmienka uvádza typ_predmetu, ktorý môže nadobúdať len hodnoty "V" ako voliteľné. Z dôvodu, že voliteľných predmetov môže byť pre daný odbor viacero aj v ten istý čas, je potrebné volať iný typ mriežky. Trieda MriezkaVolitelne obsahuje rovnaký názov metódy *zobrazMriezku*.

Výber údajov podľa učiteľa

Pre učiteľa treba vo výberovom príkaze spojiť tabuľky PREDNASKA a PREDMET a zjednotiť ich pomocou UNION-u s výberom CVICENIE, PREDMET. Podmienkami budú vždy konkrétne meno a priezvisko učiteľa na výber z tabuliek. Výsledok sa pošle triede MriezkaUcitel, pretože zobrazenie rozvrhu pre učiteľa sa mierne líši od zobrazenia pre odbor, preto som vytvoril samostatnú mriežku pre učiteľov.

Výber údajov podľa miestnosti

Pre miestnosť treba takisto pospájať tabuľky PREDNASKA, PREDMET a CVICENIE, PREDMET a takisto aj PREKAZKY. Zjednotenie týchto troch blokov vyberie údaje pre konkrétnu miestnosť, ktoré sa následne zobrazia v triede MriezkaMiestnost.

3.6.2 Spôsob zobrazovania potrebných údajov

Na zobrazenie údajov slúžia spomínané triedy implementujúce rozhranie Mriezka. Aby boli naraz použiteľné všetky údaje, prenesené v argumente metódy *zobrazMriezku,* teda obsiahnuté v objekte triedy ResultSet, je treba použiť nejaký zoznam, zložený z týchto údajov. Tie sa pomocou metód *getString* a *getInt* dokážu zobraziť samostatne, kým existujú nejaké záznamy v ResultSete a tým pádom aj uložiť do premenných. Pretože týchto údajov je viac ako jeden alebo dva, ani jedna kontajnerová trieda (Arrays ani HashMap), nie je schopná vytvoriť zoznam z niekoľkých argumentov. Je však možné ukladať do zoznamu objekty, ktoré už tieto dátové členy môžu obsahovať. Vytvoril som preto pomocné triedy ZaznamOdbor, ZaznamUcitel a ZaznamMiestnost, ktorých jediná úloha spočíva v uložení údajov. Objekty tejto triedy už potom môžem ukladať do kontajnerovej triedy a mať pritom k dispozícii všetky potrebné údaje.

Trieda Arrays na tento účel nepostačuje, pretože je nutné rozlišovať isté číselné usporiadanie objektov uložených v zozname – aby bolo jasné, do ktorého okna v mriežke sa má zobraziť. Pretože vyučovacích dní je EUBA 5 a počet hodín v rámci dňa je 7, počet buniek teda musí byť 35. Trieda HashMap vytvorí inštanciu, ktorú program hneď po vytvorení naplní prázdnymi hodnotami. Kombináciou dňa a hodiny sa vypočíta, ktoré políčko bude zaplnené daným záznamom a kľúč v zozname pod týmto číslom uloží všetky prislúchajúce údaje.

Každá bunka je dimenzovaná na 19 miest pre znaky medzi oddeľujúcimi čiarami v riadku ||. Je to dostatok na zobrazenie všetkých údajov a celková mriežka pri tom nie je príliš veľká. Celá grafika čiar je prispôsobená na elegantné zobrazenie pomocou javovskej štandardnej metódy *System.out.println.* Prvé riadky zobrazujú číslo hodiny – umiestnené na mieste desiateho symbolu, teda presne v polovici bunky. Ľavý panel slúži na zobrazenie skratky dňa.

Trieda MriezkaPovinne

Trieda MriezkaPovinne zobrazuje rozvrh hodín pre odbor v ročníku, pododbor a krúžok. Pre tieto tri entity je potrebné zobraziť v políčku v mriežke priezvisko učiteľa, skratku predmetu, aj to či ide o povinný, alebo voliteľný, miestnosť a rozlíšenie cvičenia od prednášky. Pre odbor, pododbor či krúžok sa nemôže stať, že by sa ich hodiny prekrývali, teda na zobrazenie všetkých týchto údajov sú v políčku v mriežke potrebné 3 riadky, ktoré vypisuje metóda *System.out.println*.

Bunky mriežky sú naplnené pomocou metód *naplnBunky*, definované v tele tejto triedy ako statické, teda, môžu k nim pristupovať len objekty tejto triedy. Ide o preťažené metódy z toho dôvodu, že prvý a tretí riadok síce obsahujú len jeden údaj na zobrazenie (priezvisko učiteľa, resp. miestnosť), ale druhý riadok zobrazuje tri krátke údaje (rozlišovač cvičenia a prednášky, skratku predmetu a rozlíšenie povinných a voliteľných predmetov) a tak potrebujeme 3 údaje. Stačí však zadať len číslo bunky a potrebné údaje následne program nájde v zozname.

Zobrazenie týchto údajov v bunke je formátované do stredu riadku, čiže zo základných 19 znakov sa vypočíta koľko prázdnych znakov má byť od začiatku riadku po prvý znak zobrazovaného údaju a koľko prázdnych znakov má byť od posledného znaku zobrazovaného údaju po koniec riadku. Výsledok sa odošle spať do mriežky ako jeden reťazec pozostávajúci vždy z 19 riadkov. Metóda *naplnBunky* sa bez rozdielu volá pre každú jednu bunku, aj tie ktoré sú vyplnené nulami. V tomto prípade sa do mriežky nazad odošle riadok s 19 medzerami. Podobne v prípade, ak priezvisko učiteľa je len hodnota "AAA" (nezadaný učiteľ pre danú hodinu).

```
 ___________________
|      Krsjak       |
|P       OS       P|
|       B1_08       |
|___________________|
```

HI, 2. Ročník,

Trieda MriezkaVolitelne

Táto trieda musela byť vytvorená kvôli prípadom zhody viacerých predmetov v rovnaký deň a hodinu. Stáva sa teda, že v jednom okne by mali byť zobrazené 2 predmety a viac. Vo väčšine prípadov voliteľných predmetov sa neobjavuje priezvisko učiteľa, preto som sa rozhodil tento údaj vypustiť. Skratka predmetu sa spolu s označením miestnosti zmestia do jedného riadku a posledný potrebný údaj je rozlíšenie prednášky od cvičenia. Prednáška je označená klasicky – písmeno 'P' a cvičenia sú aj v tomto prípade priradené krúžkom, nie však klasických, ale špeciálnych, vytvorených práve pre voliteľné predmety. Čísla týchto krúžkov sú potom odlíšením cvičenia.

```
 ___________________
|4       RP       D103|
|5      HPEU      D101|
|4      HPEU      B103|
|___________________|
```

FBI, 3. Ročník, voliteľné

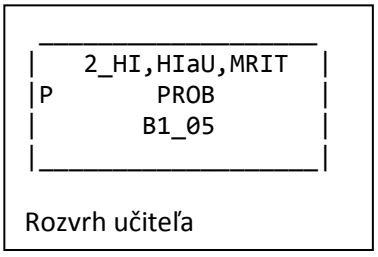
V prípade, že zhody času u dvoch predmetoch, je rovnako potrebné uviesť do mriežky oba. Toto realizuje metóda *naplnVolit*, ktorá vracia ako reťazec všetky tri údaje v jednom riadku. Touto metódou takto možno naplniť všetky tri riadky v bunke. V prípade, ak časovo zhodných predmetov je viac ako 3, členská metóda *vlozRiadok*, vloží nový

riadok do celého dňa a pre danú hodinu sa následne uloží záznam. Takto je možné vložiť až 10 riadkov, čo úplne stačí.

Aby sa nestala situácia, keď v jeden deň sú na dvoch vyučovacích hodinách 4 predmety a vytvoril by sa štvrtý a piaty riadok, je tento prípad ošetrený porovnaním čísel zoznamu hešovacej mapy s hodnotou null. Ak sa stane, že dva záznamy vyhodnotí ako true, volá sa priamo z tela tejto metódy *naplnVolit*, ktoré ako návratovú hodnotu pošlú nazad do *vlozRiadok* reťazec s 19 znakmi. Tie sa začlenia nového riadku pre daný deň. Táto metóda sa volá po každom dni, a v prípade nezhody sa nič nezobrazí (metóda je návratovej hodnoty void, prípadný nový riadok sa len zobrazí, nepošle naspäť do mriežky).

Triedy MriezkaUcitel a MriezkaMiestnost

Obe triedy pracujú totožne, rozdiel je len v jednom údaji, potrebnom na zobrazenie. MriezkaUcitel musí zobrazovať aj miestnosť, kde bude konkrétny učiteľ vyučovať a naopak MriezkaMiestnost musí obsahovať priezvisko učiteľa, ktorý bude v konkrétnej miestnosti vyučovať. Preto bolo potrebné vytvoriť aj dve pomocné triedy (ZaznamUcitel resp. ZaznamMiestnost). Ako vysvetľujúci príklad použijem MriezkaUcitel.



Rozvrh učiteľa

Rozvrh miestnosti

Druhý a tretí riadok sú podobne ako v triede MriezkaPovinne názov predmetu a rozlišovač prednášok cvičení a označenie miestnosti. Prvý riadok tvorí označenie paralelky, zložené z čísla ročníku, skratky odboru, prípadne pododboru a číslo krúžku, ak sa jedná o cvičenie, ktoré je oddelené lomenom. Tieto údaje sa vkladajú pomocou ďalšej členskej metódy *naplnParalelky*.

Týchto odborov môže byť na jednej hodine však viac, preto sú radené za sebou. V prípade, že by nestačila kapacita riadku v bunke, podobným spôsobom ako v prípade MriezkaVolitelne sa vloží nový riadok metódou *vlozRiadok*.

3.6.3 Beh používateľskej aplikácie

Na nasledujúcej ukážke je demonštrované prehliadanie rozvrhu hodín študentom. Najskôr musí zadať čo chce prezerať, zadať školský rok v predpísanom tvare, semester v danom roku. Táto časť sa vykoná pre všetky tri typy používateľa.

V ďalšej časti si program vyžiada číslo ročníku, skratku fakulty, skratku odboru a opýta sa, či zadaný odbor obsahuje aj pododbor. Ak áno, používateľ tak zadá aj skratku pododboru. Potom sa vyberú požadované údaje z databázy a zobrazia v mriežke. Následne sa program opýta či chce študent zobraziť rozvrh hodín pre konkrétny krúžok. Po zvolení možnosti "áno" môže používateľ vložiť číslo krúžku, o ktorý sa zaujíma. Predchádzajúca mriežka sa zobrazí ešte raz, len sa v nej doplnia údaje o cvičeniach.

Následne sa program opýta či chce používateľ zobraziť voliteľné predmety pre daný odbor a ročník. Pri zadaní "áno" sa zobrazí rozvrh hodín, ak nie, opýta sa program, či chce používateľ vidieť ešte nejaký rozvrh z kategórie "študent". Ak zadá "áno" cyklus programu sa vráti k bodu, kde žiada používateľa o číslo ročníka. Ak by zadal nie, program sa opýta, či už má skončiť. Odpoveď "nie" nasmeruje činnosť programu na začiatok – kde sa volí typ používateľa.

```
Dobrý deň, víta vás aplikácia Rozvrhy
_____________________________________________________________________________________________
Zadajte aký rozvrh chcete zobraziť:
(student) (ucitel) (miestnost)
student
Zadajte školský rok v tvare: 20xx/20xx:
2010/2011
Zadajte semester:
zimný
_____________________________________________________________________________________________
Zadajte ročník:
2
Zadajte skratku fakulty:
FHI
Zadajte skratku odboru:
HI
Obsahuje váš odbor pododbor? (A/N)
N
_____________________________________________________________________________________________
Zvolený rozvrh: 2. ročník   Hospodarska_informatika        Zimný semester 2010/2011
```

	1	2	3	4	5	6	7
PO	P Krsjak OS B1_08 P						
UT		P Mazikova UCTA A7_04 P					
ST		P Rakovska TEI A7_12 P	P Hanak PROA B1_08 P	P Vojtkova STAA B1_10 P			
ŠT							
PI							

```
Chcete rozvrh pre konkrétny krúžok? (A/N)
A
Zadajte číslo krúžku:
2
_____________________________________________________________________________________________
Zvolený rozvrh: 2. ročník   Hospodarska_informatika   kr. 2        Zimný semester 2010/2011
```

	1	2	3	4	5	6	7
PO	P Krsjak OS B1_08 P	C TEI A7_12 P		C STAA A8_08 P			
UT		P Mazikova UCTA A7_04 P	C UCTA A7_07 P				
ST	C PROA A8_04 P	P Rakovska TEI A7_12 P	P Hanak PROA B1_08 P	P Vojtkova STAA B1_10 P		C OS A8_14 P	
ŠT							
PI							

```
Chcete zobraziť voliteľné predmety?
N
Chcete pozrieť ďalší rozvrh z kategórie študent? (A/N)
N
koniec? (A/N)
A
```

3.7 Ukladanie nových rozvrhov do databázy

Jednou z užívateľských požiadaviek bolo aj archivovanie starých semestrov s možnosťou prezerania a ukladanie nových semestrov do databázy. Aby bola zabezpečená dynamika prezerania, najlepšou možnosťou bola voľba ukladania každého nového rozvrhu do tých istých tabuliek s rozlíšením pomocou primárneho kľúča rok a semester. Keďže všetky rozvrhy sa budú ukladať do jednej aplikácie, nie je potrebné pristupovať do inej, čo by v konečnom dôsledku spomalilo exekúciu programu, kvôli zložitému pripájaniu sa do databázy. Užívateľ pri spustení databázovej aplikácie iba napíše cestu k danému zdrojovému súboru a o ostatné sa už aplikácia postará.

4 Záver

Náplňou tejto práce bolo navrhnúť databázu rozvrhu hodín a k nej aplikáciu, ktorá ju bude automaticky napĺňať zo zdrojového súboru. Návrh databázy prešiel radou formálnych procedúr – od analýzy zdrojového súboru, vytvorením konceptuálneho, logického modelu až po vytvorenie jeho fyzickej podoby. Databáza bola vytvorená podľa požiadaviek používateľov. Primárnym cieľom tejto práce bolo vytvoriť aplikáciu, ktorá spracuje zdrojový súbor, používaný v programe Rozvrhy i keď v textovej podobe. Používateľ spustí databázovú aplikáciu, kde iba vloží do konzoly programu cestu k zdrojovému súboru, uloženému v počítači a aplikácia ho približne za 2 minúty (v závislosti od rozsahu) celý spracuje a uloží do databázy. Spracovanie zdrojového súboru prechádza zložitým parsovacím procesom, ktorý tvorí ten pomyselný motor, celej aplikácie.

Na druhej strane v používateľskej časti má prezerajúci možnosť načítať údaje z databázy a zobraziť ich do konzoly vývojového prostredia v prehľadnej mriežke dimenzovanej pre všetky prípady prezerania. Výhodou aplikácie je možnosť prístupu do starých semestrov, ktoré sú uložené v tých istých tabuľkách, bez čakania na zdĺhavé pripájanie sa do inej databázy.

Tento program však nebol vytvorený ako prezentačný nástroj a ostáva iba vývojárskym výstupom pre jednoduché zobrazovanie rozvrhu hodín pomocou jazyka HTML a následne v kalendároch – MS Outlook a pre mobilné operačné systémy.

Zoznam použitej literatúry

Knižné publikácie:

BLOCH, J. 2008 *Efective Java*. Stoughton: Courier, 2008. 327 s. ISBN-13, 978-0-321-35668-0

CODD, E. F. 2000 *The Relational Model for Database Management Version 2*. Ann Arbor: Addison Wesley Publishing Company, 2000. 538 s. ISBN 978-0201141924

DONAHOO, M.J, SPEEGLE, G.D. 2005 *SQL : Practical guide for developers*. San Francisco: Elsevier Inc, 2005. 249 s. ISBN-13 978-0-1222-0531-6

HEROUT, J. 2006 *Java : Bohatství knihoven*. České Budějovice: Kopp, 2006. 250 s. ISBN 80-7232-288-5

LIANG, D. 2011 *Introduction to Java Programming : Brief, 8/E*. Savannah: Prentice Hall, 2011. 744 s. ISBN-13: 9780132130790

Internetové zdroje:

DEVDAILY. 2007 *Java JDBC*. [online]. 2007. Dostupné na internete <http://www.devdaily.com/java/edu/pj/jdbc/jdbc0003>

MICROSOFT. 2012 *MSDN Library : SQL Server*. 2012. Dostupné na internte <http://msdn.microsoft.com/en-us/library/ms187928.aspx>

ORACLE, 1993 *Java™ Platform : Standard Edition 7 API Specification*. [online]. 2012. Dostupné na internete: <http://docs.oracle.com/javase/7/docs/api/index.html>

W3SCHOOLS. 1999 *SQL Quick Reference*. [online]. 2012. Dostupné na internete <http://www.w3schools.com/sql/sql_and_or.asp

Prílohy

Prípady, ktoré sa vyskytli pri parsingu zdrojového súboru v časti .PREDMETY. Ide o výnimky, bez ktorých by bol rozvrh hodín pre mnohé paralelky nekompletný

Štandard:

```
1_NHF_FBI 32001/1: KM: P V | Matematika_A MATA Fecenko_Jozef 1/1 160 ;3
~ 725 0
$1 B1_05 2 1 0 Fecenko_Jozef :F
&1 C1_07 1 2 1 0
&1 B1_06 3 1 2 0
&1 B1_08 3 4 3 0
&1 C1_07 1 2 4 0
&1 B1_06 3 5 5 0
&1 B1_06 3 5 6 0
&1 B1_06 3 1 7 0
&1 B1_08 3 4 8 0
```

Paralelka pozostávajúca len z fakulty:

```
1_FPM 75043/1: KMPV: V 0 V | Komparacia_politic_systemov KPS AAA_ucitel 1/0 100
;66
~ 624 1
$1 C1_07 3 6 0
```

Krúžok k paralelke pozostávajúcej len z fakulty:

```
1_NHF 19077: KAIVT: V 1 V | Informatics INFAJ Prvakova_Slavka 1/1 35 ;0
~ 502 0
$1 3B01 1 4 0 Prvakova_Slavka
&1 3B01 1 5 1 0
```

Vypísaný zoznam krúžkov:

```
1_OF_POCR 11001/4: KET: S V | Ekonomicka_teor ET AAA_ucitel 1/1 80 ;46
# 7 8 9 10
~ 675 9
$1 D113 1 3 0 :F
&1 B106 4 2 7 0
&1 B208 3 2 8 0
&1 B202 3 4 9 0
&1 B212 4 2 10 0
```

Len prednáška:

```
1_NHF_ETZ 13001: KVSRR: P V | Hospodarska_geograf_sv HGS Misunova_Ema 1/0 20
;38
~ 674 0
$1 B108 2 4 0 Misunova_Ema :F
```

Len cvičenie:

```
1_NHF_LZSM 75020/8: KMPV: V 1 V | Filozofia FIL Gregus_Peter 0/1 25 ;29
~ 624 1
&1 A6_06 3 1 1 0
```

Neznámy učiteľ:

```
1_NHF_FBI 41003: KPH: P V | Podnikove_hospod PH AAA_ucitel 1/1 160 ;2
~ 54 1
$1 B1_02 1 3 0
&1 A5_17 1 1 1 0
```

```
&1 B102 1 2 2 0
&1 B206 1 2 3 0
&1 A4_08 3 1 4 0
&1 B206 1 1 5 0
&1 A5_16 4 5 6 0
&1 A5_17 4 4 7 0
&1 A5_06 2 3 8 0
```

Učiteľ na cvičení:

```
1_OF_POCR 75002/1: KMPV: V 2 V | Filozofia FIL Marton_Milan 0/1 40 ;43
~ 624 1
&1 A5_15 1 7 1 0 Marton_Milan
&1 A4_07 3 6 2 0 Marton_Milan
```

Voliteľná prednáška:

```
1_NHF_ETZ 75017/8: KMPV: V 0 V | Politologia POL Lidak_Jan 1/0 20 ;40
~ 624 1
$1 C1_09 4 7 0 Lidak_Jan
```

Paralelka s pododborom:

```
4_FPM_EMP_VMZ 42007: KMZ: P V | Medz_manaz_a_medz_podnikan MMMP Srsnova_Jana
1/1 75 ;449
~ 67 1
$1 B1_03 5 3 0 Srsnova_Jana :F
&1 A6_07 3 2 1 0
&1 A6_14 3 1 2 0
&1 A5_16 1 4 3 0
```

Rozdielni učitelia na cvičení a prednáške:

```
4_FHI_SME 34024: KS: P V | Narodohospodarska_statistika NHS Pardelova_Ruzena
1/1 24 ;500
~ 565 1
$1 C1_09 3 5 0 Pardelova_Ruzena
&1 A7_12 2 4 1 0 Hurbankova_
```

Zakončenie riadku symbolom F:

```
4_FHI_UFR 35059: KUA: P V | Medzin_stand_uctov_vykaz IFRS Tumpach_Milos 2/1 50
;515
~ 727 1
$2 B1_04 3 4 0 Tumpach_Milos
&1 A7_04 4 3 1 0  : F
&1 A7_04 4 3 2 0  : F
```

2 hodiny prednášok:

```
2_OF_POCR 12001/6: KHP: P V | Narodohospodarska_polit NHP Lukacik_Jan 2/0 250
;149
~ 662 9
$2 B108 5 2 0 Lukacik_Jan :F
```