Mendelova univerzita v Brně
Provozně ekonomická fakulta

Administračný modul pre podporu marketingových procesov a starostlivosti o zákazníkov

Diplomová práca

Vedúci práce:
Ing. Pavel Turčínek, Ph.D.

Bc. Jakub Kušnier

Brno 2016

Zadanie

Rád by som sa najskôr poďakoval vedúcemu diplomovej práce, Ing. Pavlovi Turčínkovi, Ph.D., za ochotu, rady a trpezlivosť, ktorú so mnou mal. Ďalej by som rád poďakoval svojmu vedúcemu v práci a architektovi systému Effortix, Ing. Václavovi Hodkovi, za nápad práce a poskytnutie všetkých potrebných informácii, prístupov a dát. V neposledním rade by som sa rád poďakoval svojej rodine za podporu a svojím priateľom za nezabudnuteľné roky v Brne a semester v estónskom Tartu.

Čestné prehlásenie

Prehlasujem, že som prácu: **Administračný modul pre podporu marketingových procesov a starostlivosti o zákazníkov** vypracoval samostatne a všetky použité zdroje a informácie uvádzam v zozname použitej literatúry. Súhlasím, aby moja práca bola zverejnená v súlade § 47b zákona č. 111/1998 Sb., o vysokých školách v znení neskorších predpisov a v súlade s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací*.

Som si vedomý, že sa na moju prácu vzťahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzatvorenie licenčnej zmluvy a použitie tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

Ďalej sa zaväzujem, že pred spísaním licenčnej zmluvy o použití diela inou osobou (subjektom) si vyžiadam písomné stanovisko univerzity, že predmetná licenčná zmluva nie je v rozpore s oprávnenými záujmami univerzity a zaväzujem sa uhradiť prípadný príspevok na úhradu nákladov spojených so vznikom diela, a to až do ich skutočnej výšky.

V Brne dňa 20. mája 2016 _____

Abstract

Kušnier, J. Administration module for marketing process suppport and customer care service. Master's Thesis. Brno: MENDELU, 2016.
Main topic of this thesis is analysis of Effortix system, with business intelligence system development, which can cluster data, show statistics and other results from given data.

Keywords

Business Intelligence, Data mining, Weka, marketing, web, Java, Jetty, PHP, MySQL, Effortix.

Abstrakt

Kušnier, J. Administračný modul pre podporu marketingových procesov a starostlivosti o zákazníkov. Diplomová práca. Brno: MENDELU v Brne, 2016.
Diplomová práca sa zaoberá analýzou existujúceho Effortix systému a vytvorením business intelligence systému, vďaka ktorému je možné dáta zhlukovať podľa rôznych kritérii, zobrazovať štatistiky, periodické dáta a iné údaje.

Klíčová slova

Business intelligence, dolovanie dát, Weka, marketing, web, Java, Jetty, PHP, MySQL, Effortix.

Obsah

Zoznam obrázkov

1 Úvod a cieľ práce

1.1 Úvod

"*If you torture the data long enough, it will confess.*" – Ronald Coase (Wikiquote)

Známy výrok britského ekonóma vystihuje podstatu súčasnej práce s dátami a ich spracovaním. Odvtedy však doba pokročila – množstvo dát sa zniekoľkonásobilo, vďaka rozšírenému pripojeniu na internet sa zlepšil prístup k nim, pokročili metódy, akými tieto dáta môžeme spracovať a samotné množstvo dát, ktoré existuje a naďalej pribúda, dosahuje objemy, ktoré nie je možné spracovávať a zrozumiteľne zobraziť bez použitia výpočtovej techniky. K tomu práve slúžia vyvinuté algoritmy dolovania dát. Ich rozsah je široký, od sociálneho, kde jeho použitie a výsledky priamo prenikajú do spoločnosti a dokážu ovplyvňovať správanie firiem a tým aj zákazníkov, cez vedecký dopad, až po skryté použitie, kedy napríklad internetové systémy na základe správania návštevníkov dokážu podsúvať informácie a meniť ich celkové správanie.

Filozoficky sa môže vyskytovať problém so zneužitím týchto dát, táto práca sa však uberá iným smerom.

Do jednej z týcho oblastí patria business intelligence systémy, ktorých firemné použitie stojí medzi zákazníkmi, trhom, zdrojmi a konkurenciou, a snaží sa o efektívnu analýzu pre čo najlepšie plánovanie ďalších krokov.

Príkladom tohto môže byť spoločnosť Best buy, ktorá podľa Kotlera (2013) zhromaždila cez 15 terabytov dát od 75 miliónoch domácností a pomocou analytických metód dokáže upravovať svoje marketingové aktivity až na úrovni jednotlivých zákazníkov.

Systém Effortix od spoločnosti LWi s.r.o., ktorý bude bližšie popísaný v ďalších kapitolách, ako začínajúci projekt tiež bude postupne zbierať väčšie a väčšie množstvá dát, ktorých analýza a následného vytvorenie jednotného BI môžu dopomôcť k rastu a využitia všetkých "priznaní od dát", ktoré bude možné získať.

1.2 Cieľ práce

Cieľom práce je vytvoriť administratívno správcovský modul na prácu s dátami k existujúceho webovému systému. Práca bude vytvorená pre internetovú službu Effiortix, ktorý sa zaoberá vytváraním mobilných aplikácii. Samotné dáta bude možné pomocou výsledného modulu zo systému získavať, spracovávať a poskytovať ďalším systémom za účelom marketingového a strategického rozhodovania v prípade samotné systému, alebo ako nadštandartná služba pre koncových zákazníkov.

Medzi funkcie pre rozhodovanie vedenia patri:

- základné štatistiky,
- periodické dáta (denné, mesačné, ročné) o príjmoch a pomeroch zakúpených tarifoch pre jednotlivé aplikácie,
- zhlukovanie existujúcich aplikácií (ďalej len klusterizácia) podľa rôznych kritérii (podľa počtu uzlov v aplikácií, poslednej úpravy, jazykovej klusterizácií, miesta objednávky – GPS klusterizácia, alebo multidimenzionálna medzi klusterizácia),
- prehľadné zobrazenie existujúcich aplikácií, ich príslušnosť k jednotlivým klusterom, filtrovanie a priamy prístup k aplikáciam,
- doplnkové funkcie, ktoré budú aktivované pre koncových zákazníkov s jedným z platených tarifov,
- základné štatistiky objednávok,
- analýza objednávok na hľadanie výrazných medziobjednávkových asociácií, ako spôsob zvýraznenia najčastejších kombinácií, ktoré zákazníci kupovali za účelom ďalšej práce s týmito dátami,
- periodické údaje o objednávkach (denné, mesačné, ročné) so zobrazením pre konkrétne meny a s možnosťou geofiltrovania pre určitú oblasť na mape.

V závere práce bude taktiež pre rozhrania vytvorená dokumentácia, kde budú popísané vstupné parametre pre jednotlivé funkcie a formát vrátených dát.

1.3 Metódika práce

Samotnej tvorbe práce predchádza schôdzka s vedúcim pracovníkom systému Effortix s krátkou inštruktážou jeho fungovania, prejednaním nápadov a poskytnutí prístupov k dátam spolu so štruktúrami fungujúceho systému. Na základe týchto požiadavkov, nápadov, informácii a štruktúr boli zformulované požiadavky na nový business intelligence systém.

Analýza je popísaná v kapitole 6 a jej výsledkom je návrh popísaný v kapitole 7, ktorej predchádzalo štúdium potencionálnych technológii a ich možná aplikácia pre daný návrh, o ktorých je možné si prečítať v časti 5.

Implementačná časť prebiehala s testovaním na rôznych typoch dát a po konečnom vyladení bola aplikácia nasadená na interné servery systému, kde sú pripravené na použitie.

Po implementácii následovalo testovanie funkčnosti, za kterým následuje ďalší vývoj tak, ako sa bude vyvíjať hlavný Effortix systém do budúcnosti.

2 Business intelligence

Spoločnosti organizujú svoje informácie do rôznych databází, ukladajú dáta o klientoch, produktoch a podobne. Tieto obsahujú mená, adresy, elektronické kontakty a iné sociologické a demografické informácie. Namiesto "bombardovania" všetkých klientov novými ponukami dokážu firmy zaraďovať klientov do určitých logických celkov na základe predchádzajúcich dát o ních a s nimi potom pracovať. Vďaka tomu je možné zvýšenie efektivity kampaní až o dvojciferné čísla (Keller, Kottler, 2013).

Business intelligence (ďalej BI) je definovaný ako sada procesov, know-how, aplikácií a technológii, ktorých cieľom je účinne a účelne podporovať riadiace aktivity vo firme. Podporujú analytické, plánovacie a rozhodovacie činnosti organizácií na všetkých úrovniach a vo všetkých úrovniach podnikového riadenia (predaj, nákup, marketing, finančné riadenie, kontrola, majetku, riadenia ľudských zdrojov, výroby a iných) (Maryška, Novotný, Pour, 2012).

2.1 Práca s dátami v BI

Aplikácie BI môžeme rozdeliť podľa ich charakteru na dva základné typy úloh, a to transakčné úlohy a analytické a plánovacie úlohy. Rozdiel medzi nimi spočíva v tom, že transakčné úlohy vytvárajú a sprístupňujú nové dáta, analyticko-plánovacie úlohy nové dáta nevytvárajú, ale využívajú už existujúce databázy a formujú ich pre potreby analýz a plánovania (Maryška, Novotný, Pour, 2012).

2.1.1 Transakčné úlohy

Medzi nároky na transakčné úlohy patrí zaistenie čo najrýchlejšieho prístupu k dátam na objednávkach, faktúrach, tovaroch a úprava nad nimi (oprava, výpočet a pod.), ďalej realizovať aktualizáciu týchto dát, ich stavy a to všetko podľa určitých skutočností (prijatá faktúra, zmena osobných údajov zákazníka). V neposlednom rade tiež zaistiť vytvorenie nových dát (nový zákazník, objednávka, faktúra a pod.).

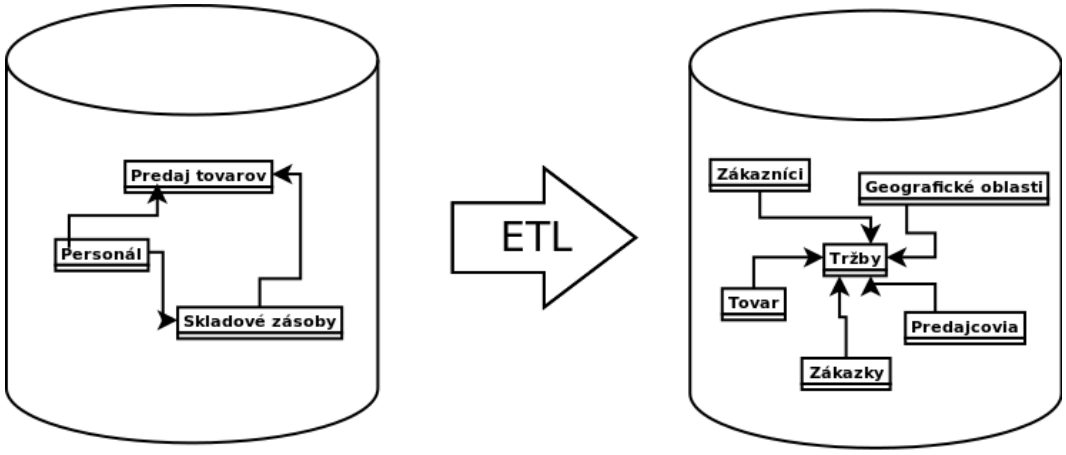
2.1.2 Analytické a plánovacie úlohy

Analytické a plánovacie úlohy zaisťujú sledovanie a hodnotenie podnikových ukazateľov(objem, tržieb, počtu reklamácií, pracovníkov a pod.), poskytujú sledovanie ukazateľov podľa určitých hľadísk, dimenzií s podporou rozhodovacích úkonov aktivít podniku (objem tržieb aj na zákazníka, kategórie tovaru, miesta objednávok, typ zákaziek a ich kombinácií) a taktiež analyzovať vývoj podstatných ukazateľov podniku v čase, a to na rôznych úroviach detailu. Existuje veľké množstvo variant BI, takže stojí za zmienku spomenúť spoločné princípy analyticko plánovacích úloh, teda BI:

- pre jeho použitie pre analyticko-plánovacie aplikácie, takže má prispôsobenú organizáciu dát v databázach
- dáta sú detailné a agregované pre rôzne úrovne detailu
- dáta má uložené multidimenzionálne pre potrebu ich vyhodnocovania podľa rôznych dimenzií
- dáta sú uložené v časových dimenziách a proces ukladania je periodický, v určitých časových intervaloch
- má vyššie nároky na kvalitu dát

2.2 Organizácia dát

Ako bolo spomenuté, BI systémy nové dáta nevytvárajú, ale využívajú už existujúce rôzne zdroje, ktoré po získaní spracujú a optimalizujú, aby mohli byť transformované do dimenzionálneho tvaru do konkrétnej analytickej databázy. Toto prebieha pomocou ETL (Extract, Transform, Load), ktoré však bude popísané nižšie, v časť Komponentny BI.



Obr. 1 Transformácia dát (Maryška, Novotný, Pour, 2012)

2.3 Dimenzia dát

Maryška, Novotný a Pour (2012) definujú dimenziu ako analytické hľadisko pre hodnotenie sledovaných ukazateľov, alebo ako štruktúra dát, prípadne databázová tabuľka obsahujúca záznamy s jednotlivými prvkami dimenzie (príkladom môže byť položky tovarov, zákazníci a podobne). Sú usporiadané v hierarchickej štruktúre s rozdelením na tovary, skupiny tovarov, alebo jednotlivé položky. Do týchto databází sa súčasne ukladajú aj agregované a iné vypočítané hodnoty na rôznych úrovniach detailu. Dáta sú uložené tzv. multidimenzionálne, podľa požiadaviek užívateľa na pohlad na dáta, tj. z viacerých hľadisk.

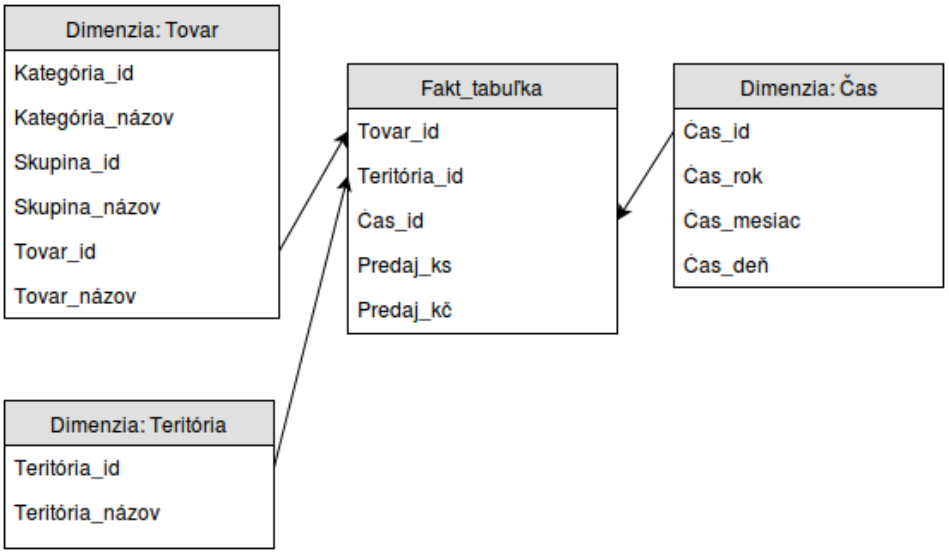
Podľa spôsobu realizácie multidimenzionality v dátach poskytuje dve základné jej vyjadrenia, a to multidimenzionalita vyjadrenia v relačných databázach a multidimenzionalita pomocou OLAP (On Line Analytical Processing).

2.3.1 Multidimenzionalita v relačných databázach

V relačných databázach bývajú multidimenzionálne dáta uložené v 3. normálnej forme, ktoré je pre tento typ použitia najviac vhodné za účelom rýchleho vkladanie dát a optimalizovanej veľkosti databází. Pre zložitejšie schémy produkčných databází sú prítomné mnohé problémy prevedenia vyplývajúce z ich podstaty, ako nesúrodosť tabuliek, neexistencia možnosti prevedenia zložitejších príkazov nad databázami, neprehľadnosť, alebo nutnosť vytvárať prepojovacie mostíky. V snahe tieto problémy obmedziť, sa objavili zjednodušenia dát ERD diagramov, ktoré sú prispôsobené pre tvorbu dátových skladov, a to dimenzionálnymi modelmi; konkrétne schéma hviezdy a schéma snehovej vločky (SNOWFLAKE scheme) (Novotný, Pour a Slánsky, 2005).

2.3.1.1 Schéma hviezdy

Schéma hviezdy, anglicky STAR scheme, je dimenzionálny model, pre ktoré je charakteristické uloženie celej hierarchickej štruktúry dimenzie v jednej tabuľke (Maryška, Novotný, Pour, 2012).

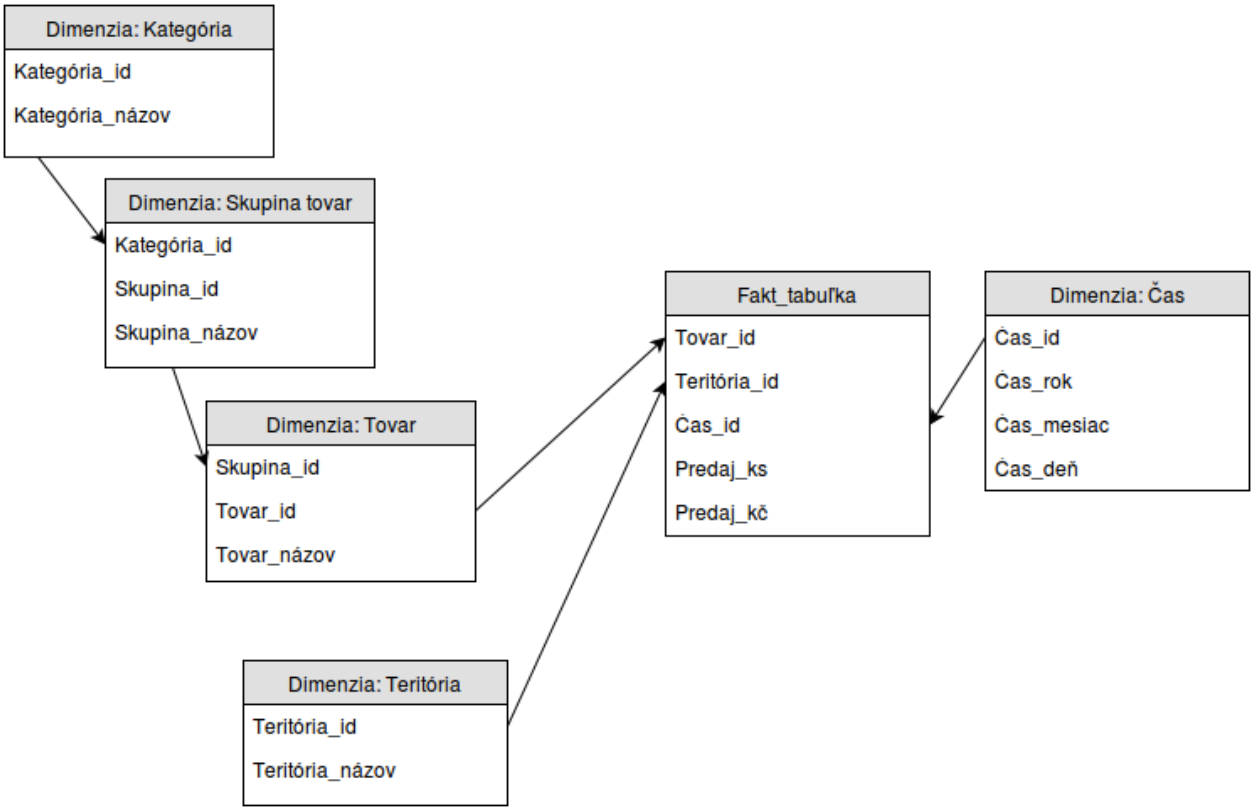


Obr. 2 Dátový model STAR schéma (Maryška, Novotný, Pour, 2012)

Dotazovanie prebieha vždy pomocu SQL príkazov (Novotný, Pour, Slánsky, 2005), ktoré budú popísané v ďalších kapitolách.

2.3.1.2 Schéma snehovej vločky

V niektorých prípadoch je riešenie schémy hviezdy nevýhodné, preto sa v týchto prípadoch dimenzionálne tabuľky normalizujú, a to tak, že sa podľa hierarchickej úrovne dimenzie rozdelí do viacerých tabuliek. Tým je tiež zaistené, že sa jednotlivé atribúty v tabuľke neopakujú. Táto schéma sa nazýva snehová vločka, teda SNOWFLAKE (Maryška, Novotný, Pour, 2012).

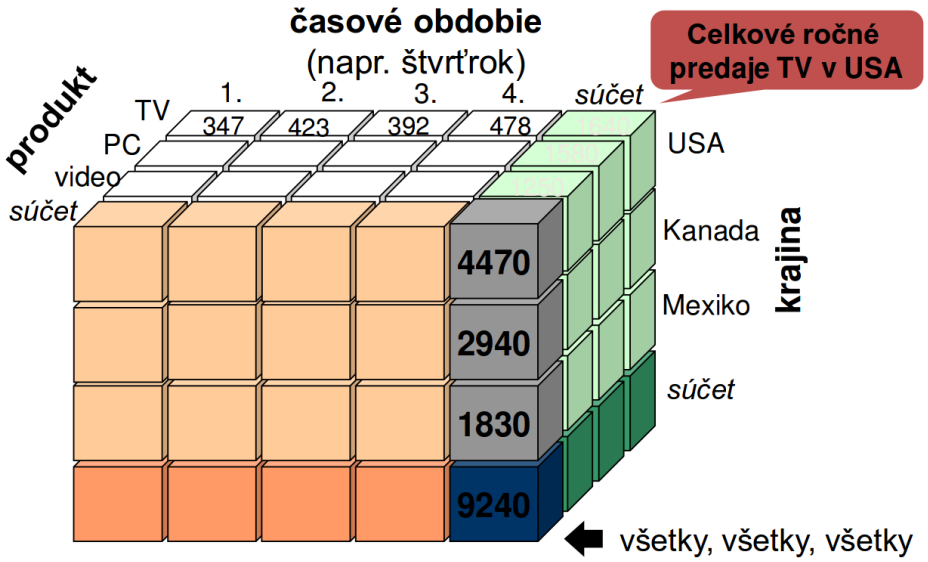


Obr. 3 Dátový model STAR schéma (Maryška, Novotný, Pour, 2012)

V tomto prípade bola dimenzia Tovar hierarchicky rozdelená na zvláštne tabuľky; oddelená bola schématicky tabuľka Skupina tovar a Kategória, ktoré sú v tomto prípade medzi sebou zviazané.

2.3.1.3 Multidimenzionalita v OLAP technológiách

Maryška, Novotný a Pour (2012) definujú OLAP (On Line Analytical Processing) technológie ako informačnú technológiu založenú na koncepcií multidimenzionálnych databází, ktorej princípom je niekoľkodimenzionálna tabuľka, ktorá umožňuje rýchlo a pružne meniť jednotlivé dimenzie a meniť tak pohľady na modelovanú ekonomickú realitu. Tie zahrňujú predspracované agregácie dát podľa historických štruktúr dimenzií a ich kombinácii. Obsah dimenzií a ich premietnutie do jedného bodu tvorí prvok OLAP kocky a jej veľkou výhodou je hlavne rýchlosť práce s nimi a krátka odozva.



Obr. 4 Príklad dátovej kocky (Paralič, 2016)

Tie vytvárajú OLAP databázu a realizujú sa v niekoľkých variantách. Príkladom môžu byť:

- MOLAP – Multidimensional OLAP. Dáta sú uložené v multidimenzionálnych, binárnych, OLAP kockách
- ROLAP – Relational OLAP. Multidimenzionalita pomocou relačné databázy
- HOLAP – Hybrid OLAP. Kombinácia MOLAP A ROLAP

Rozdiely medzi multidimenzionalitou v relačných databázach a OLAP systémami zhrnuli Novotný, Pour a Slánsky (2005) do nasledovných bodov:

- Relačné databázy sú určené na získavanie dát a tomu má aj uspôsobené architektúru, najmä databázový model. Charakteristický pre neho sú tabuľky v 3. normálnej forme, veľkým množstvom tabuliek a ich spojením a snahu o nulovú redundanciu dát.
- Dáta sú v relačných databázach aktualizované v reálnom čase, prebiehajú až desiatky tisíc tranzakcii za minutu; systém je zaťažovaný rovnomerne.
- Analytické systémy sú primárne určené na dotazovanie, a tým majú prispôsobenú architektúru, modely, nižšiu normalizáciu, vyššia frekvencia duplicít a pod.
- Väčšina analytických systémom aktualizuje svoje dáta periodicky, v denných alebo mesačných intervaloch.
- Analytické databázy si nedržia takú veľkú detailnosť dát a sú relevantne agregované a majú obmedzené atribúty.

2.4 Komponenty BI

Neexistuje jednotná koncepcia usporiadania a zloženia komponent v jednotlivých riešeniach BI a priamo závisí od konkrétneho podniku a jeho potrieb.

Medzi nástroje a aplikácie BI podľa (Novotný, Pour, Slánsky, 2005) patria:

- produkčné, zdrojové systémy,
- dočasné úložiská dát (DSA – Data Staging Areas),
- operatívne úložiská dát (ODS – Operational Data Store),
- transformačné nástroje (ETL – Extraction, Transformation, Loading),
- integračné nástroje (EAI – Enterprise Application Integration),
- dátové sklady (DWH – Data Warhouse),
- dátové trhoviská (DMA – Data Marts),
- OLAP,
- reporting,
- manažérske aplikácie (EIS),
- data mining,
- nástroje pre zaistenie kvality dát,
- nástroje pre správu metadát,
- iné.

Niektoré jej dôležité komponenty, ktoré budú využité v praktickej časti, budú popísané v nasledujúcich podkapitolách (Maryška, Novotný, Pour, 2012).

Produkčné databázy

Tiež označované ako zdrojové databázy sú databázy aplikácii, z ktorých analytické časti BI získavajú svoje dáta. Príkladom sú rôzne databázové aplikácie realizované rôznymi databázovými systémami, malé databázy, bežné tabuľkové kalkulátory, súbory oddelené čiarkami, alebo súbory s pevnou štruktúrou. Zdroje môžu byť interné, externé a úlohou BI je tieto zdroje zanalyzovať a vybrať z nich dáta relevantné pre konkrétnu aplikáciu BI.

2.4.1 ETL

ETL (teda Extract, Transform, Load) sa tiež označuje aj ako dátová pumpa, zdrojové dáta vyberá (extract), upravuje do požadovanej formy (transformation) a nahráva ich do konkrétnych dátových štruktúr na to určených (load). Prebieha pomocou prenosu medzi dvoma a viac databázami, alebo dátovými súbormi a obvykle prebieha v dávkach, naraz v určitých časových intervaloch (denne, týždenne).

Zdrojové dáta musia byť:

- vybrané účelne, určené pre konkrétne analytické a rozhodovacie procesy podniku,
- transformované do štruktúr analytických databází, ktoré sú navrhnuté, aby zodpovedali potrebám podniku,

- konsolidované, keďže prichádzajú z rôznych zdrojov databází, môžu pochádzať od rôznych dodávateľov, klientov a rovnaké dáta môžu byť uložené na viacerých miestach.

2.4.2 Dátový sklad

Dátový sklad je súhrn dát, usporiadaný pre podporu potrieb marketingu, pre ktoré platia následovné vlastnosti:

- **integrovaný** – dáta sú ukladané v rámci celého podniku,
- **konsolidovaný** – dáta sú konsolidované z rôznych zdrojov a foriem do jedného zdroja a formy,
- **subjektovo orientovaný** – dáta nie sú rozdelené podľa aplikácií, v ktorých vznikli, ale podľa ich typu,
- **stály** – dáta sú určené ako iba pre čítanie; nové dáta nevytvárajú a neaktualizujú,
- **časovo rozlíšený** – dátový sklad obsahuje aj dimenziu času pre zobrazenie jej histórie.

2.4.3 Dátové trhovisko

Je definovaný ako problémovo orientovaný dátový sklad, určený na pokrytie konkrétnej problematiky vymedzeného okruhu užívateľov a umožňujúci flexibilné analýzy dát. Na rozdiel od dátového skladu je teda určený pre obmedzený okruh užívateľov a podstatou trhoviska sú decentralizované dátové sklady, ktoré sa môžu integrovať do celopodnikového riešenia.

2.4.4 OLAP databázy

Je označenie pre niekoľko súvisejúcich a vzájomne prepojených OLAP kociek. Z dátových skladov majú predspracované agregácie dát podľa definovaných hierarchických štruktúr dimenzií.

2.4.5 Dočasné úložisko dát

Dáta, ktoré boli získane z rôznych zdrojov nie sú ešte pripravené pre ich uloženie (sú neagregované, nekonzistentné a pod.). Jej úlohou je teda tieto dáta pripraviť pred ich vstupom do skladu a po spracovaní jednoducho vymazať.

2.4.6 Reporting

Reporting je označenie činnosti spojenej s dotazovaním sa do databází pomocou štandartných rozhraní, napr. SQL príkazov. Jedná sa prehľady realizované na základe dát z dátových skladov, alebo multidimenzionálnych databází.

2.4.7 Analytické aplikácie

Označenie analytické aplikácie označuje typ aplikácií, pre ktoré platí:

- sú navrhnuté na poskytovanie manažérských informácii, sledovať firemné procesy a pod.,
- poskytujú nástroje pre on-line analýzy,
- pre zjednodušenie poskytujú vysokú vypovedajúcu hodnotu informácií pomocu grafického užívateľského prostredia.

Sú vytvárané pomocou rôznych prostriedkov – špecializovaných programov (Business objects), pomocou kancelárských aplikácií (Excel, Access), alebo pomocou špecializovaných programovacích jazykov (MDX).

2.4.8 Dolovanie dát

Dolovaniu dát (Data miningu) sa venuje jedna z ďalších kapitol tejto práce.

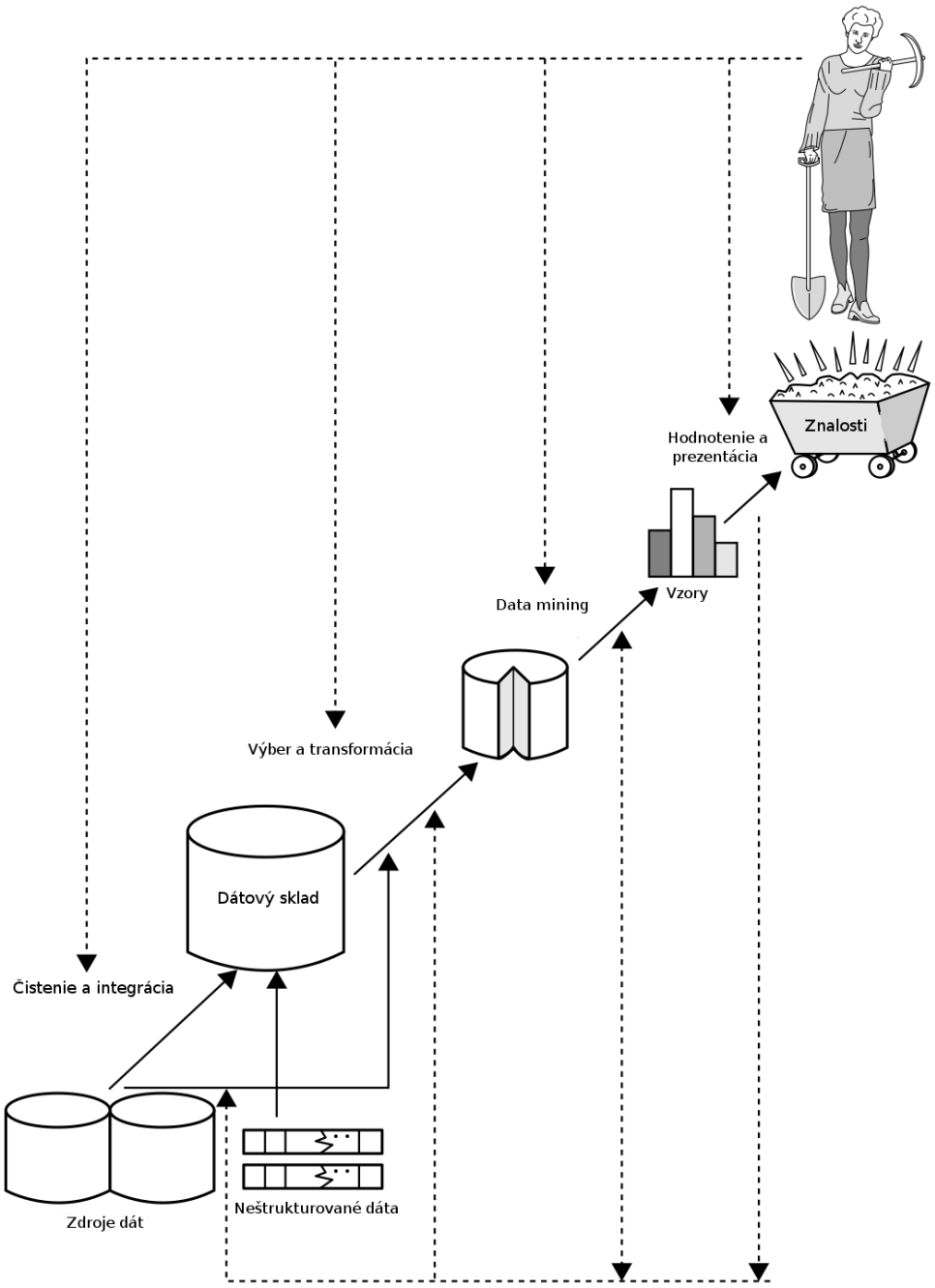
3 Data mining

V súčasnom svete sme doslova zavalení informáciami a to tak, že toto množstvo už nie je možné spracovávať takým spôsobom, ako to bolo možné v minulosti a odhadom sa toto množstvo zdvojnásouje každých 20 mesiacov. Dáta z rôznych odvetví, ako ekonómia, štatistika, predpovede počasia, alebo komunikačné technológie je možné identifikované, validované a použité pre ďalšiu predikciu (Witten, Frank, Hall, 2011).

Data mining je podľa (Han, Kamber, Pei, 2012) definovaný ako proces objavovania zaujímavých vzorov, vzorcov, a znalostí z veľkého množstva dát. Witten, Frank a Hall (2011) dopĺňajú , že sa jedná o automatický, alebo samiautomatický proces. Objavený vzor musí byť zmysluplný v tom zmysle, že vedie k nejakej, väčšinou ekonomickej výhode. V oblasti biznisu sa jedná o identifikáciu príležitostí, teda vzorov, ktoré vedú k ekonomickej výhode a jej využitiu. Úlohou jeho použitia je teda získať z dát určitý zmysel, získať vzory ako funguje reálny svet a obaliť tieto poznatky do použiteľných teórií. Tieto vychádzajú z už existujúcich dát a databází a umožňuje zistiť segment trhu, na ktorý sa má obchodník najpravdepodobnejšie zamerať a na ktorý najskôr naláka zákazníka.

Data mining je súčasť iného procesu, získavania znalostí z dát, ako jeden z jeho podprocesov (Han, Kamber, Pei, 2012):

- čistenie dát (odstránenie dátového šumu a inkonzistentných dát)
- integrácia dát (kombinácia niekoľkých zdrojov)
- výber dát (relevantné dáta k analýze sú vybrané zo zdroja)
- transformácia dát (dáta sú transformované a konsolidované to formy vhodnej na data mining)
- data mining (proces aplikácie metód na získavanie dátových vzorcov)
- hodnotenie vzorcov (identifikácia zaujímavých a reprezentatívnych znalostí)
- prezentácia znalostí (vizualizačné techniky a techniky reprezentácie znalostí za účelom prezentovania získaných znalostí pre užívateľov)



Obr. 5 Kroky procesu získavania znalostí z dát (Han, Kamber, Pei, 2012) str.7

3.1 Predspracovanie dát

Dáta, ktoré sú získavané z rôznych zdrojov na následné spracovanie vo väčšine prípadov nie sú priamo vhodné na ďalšie spracovanie. Sú zašumené, chýbné, chýbajúce. Kvalitatívne slabé dáta na vstupu môžu viesť k nepresným, alebo

úplne nesprávnym výsledkom. Kvalita dát sa vyznačuje rôznymi ukazateľmi, ktoré môžu byť:

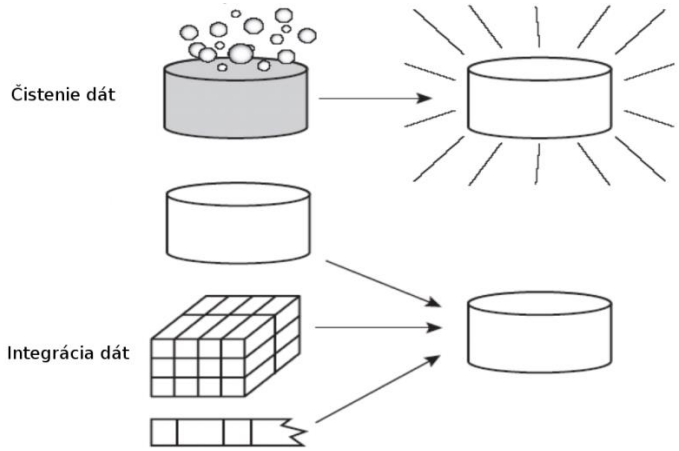
- **presnosť –** mali byť čo najlepšie reprezentovať realitu,
- **úplnosť –** do šírky (dostupnosť potrebných atribútov) aj do hĺbky (dostatočný počet hodnôt),
- **konzistencia,**
- **aktuálnosť** – v prípade rozhodovania o budúcnosti musia byť aktuálne,
- **dôveryhodnosť,**
- **pridaná hodnota** – miera prospešnosti pre riešenie danej úlohy,
- **interpretovateľnosť** – hodnoty by sa mali dať ľahko interpretovať (vek, ceny, spokojnosť zákazníkov a pod.),
- **dostupnosť.**

Existujú tri hlavné prípady nekvality dát a to:

- **nekompletné dáta** – môže chýbať hodnota nejakého atribútu (nezadaná hodnota pri nepovinnom atribúte alebo chyba v zbere), chýba samotný atribút (pri návrhu sa nepredpokladala jeho potrebnosť), alebo sú prítomné len agregované dáta (namiesto detailných)
- **zašumené dáta** – obsahujú nesprávne, alebo odľahlé hodnoty
- **nekonzistentné dáta** – redundancia dát alebo nekonzistentné pomenovanie pri použití rôznych zdrojov dát

Tieto nedostatky je potrebné odstrániť a to je možné pomocou následovných úkonov:

- **Čistenie dát** – odstránenie chýbajúcich hodnôt, identifikovanie a odstránenie odľahlých hodnôt, riešenie nekonzistencií
- **Integrácia dát** – integrovanie dát z rôznych dátových zdrojov (databází, dátových kociek, dátového skladu, alebo súborov)
- **Transformácia dát** – transformácia do tvaru vhodného pre ďalšie spracovanie
- **Redukcia dát** – výber podmnožiny atribútov, redukcia dimenzionality, redukcia počtu hodnôt
- **Diskretizácia dát** – jedná sa o špeciálny prípad redukcie dát a týka sa väčšinou numerických hodnôt, kedy je možné ich nahradiť napr. intervalom



Obr. 6 Schématické znázornenie predspracovania (Zendulka, 2009) str. 34

3.2 Základná terminológia

3.2.1 Deskriptívne a prediktívne úlohy data miningu

Funcie dolovania dát sú použité na špecifické popísanie vzorov. Podľa použitia konkrétnej úlohy data miningu sa úlohy delia na (Han, Kamber a Pei, 2012):

- **deskriptívne úlohy** – charakterizujú vlastnosť dát v cieľových dátových sadách
- **prediktívne úlohy** – slúžia na predikciu hodnôt a vlastností z existujúcich

3.2.2 Inštancia

Vstupom dát do procesu data miningu je sada inštancií. Inštanciou sa označuje ako individuálny nezávislý príklad konceptu, ktorý má byť daným algoritmom naučený, ináč povedané sa jedná o "veci", ktoré majú byť klasifikované, asociované, alebo zhlukovane[1].

3.2.3 Atribúty

Každá inštancia je určená ako presne definovaná sada vlastností, teda atribútov. Ako reprezentácia v tabuľkách, atribúty udávajú stĺpce a inštancie riadky v databázach. Každý atribút by mal mať definované svoje vlastnosti obmedzujúce podmienky. Nominálne hodnoty majú presne definovanú sadu možností a štatisticky sa delia na nominálne, ordinálne, intervaly a pomery, avšak môže byť aj vo forme zástupných symbolov, napr. slnečno, zamračené, daždivo. Nad týmito atribútmi však napr. nedávajú zmysel číselné operácie (aritmetika, radenie, ...).

[1] Tieto termíny sú popísané v práci v ďalších kapitolách

Každý dataset je reprezentovaný ako matica inštancií verzus atribútov, čo je v teórií databázových systémov označené ako jedna relácia (Witten, Frank a Hall, 2011).

3.2.4 Typy algoritmov podľa dohľadu

Výhodou data miningových aplikácií je automatické učenie, hľadanie a rozpoznávanie komplexných vzorov a prevádzať inteligentné rozhodovanie na základe dát. Podľa spôsobu rozpoznávania týchto vzorov delíme algoritmy na (Han, Kamber a Pei, 2012):

- **supervised learning** – učenie "s učiteľom", synonymum pre klasifikáciu[2]. Hlavným princípom je označovanie inštancií a odhadovanie týchto značení pre neoznačené inštancie
- **nnsupervised learning** – učenie "bez učiteľa", synonymum pre klusterizáciu[3]. Algoritmus je bez označovania a jeho značenie je dorátavané automaticky
- **semi-supervised learning** – kombinácia prechádzajúcich dvoch spôsobov. Označovanie je použité na určenie modelov tried a neoznačené hodnoty na určenie hraníc medzi triedami
- **active learning** – aktívne učenie. Užívateľ sa priamo zúčastňuje na procese a umožňuje pomocou názoru, väčšinou experta v odbore, dáta označovať a tým zlepšovať výsledok

3.2.5 Vizualizácia dát

Cieľom vizualizácie dát je vyobraziť dáta jasne a efektívne cez grafickú reprezentáciu. Použitie je už po samotnom procese dolovania dát a používa sa často v reportingu, správy obchodných operácií a podobne. Jej výhodou je jednoduchšie nájdenie a zobrazenie vzorov v dátach, ako na čisto číselných hodnotách, alebo na prvotných dátach. Medzi hlavné techniky vizualizácie patria pixelovo orientovaná vizualizácia technika, technika vizualizácie geometrickou projekciou, technika vizualizácie založenej na ikonách, hierarchická vizualizácia a vizualizácia komplexných dát a relácii (Han, Kamber a Pei, 2012). Pre potreby tejto práce budú ďalej popísané len vybrané techniky.

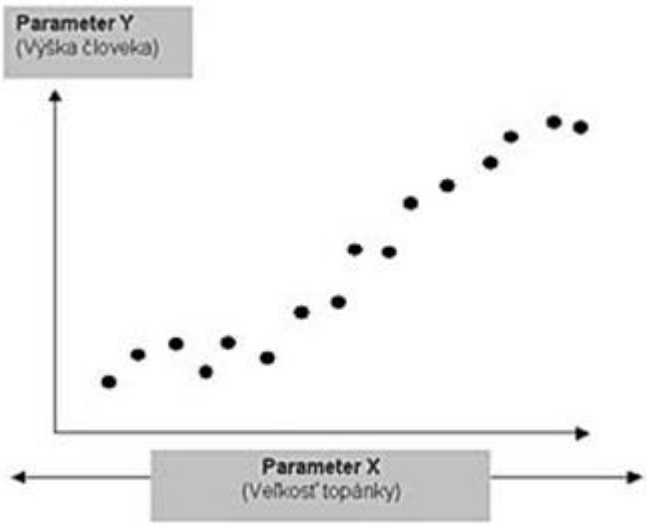
3.2.5.1 Technika vizualizácie geometrickou projekciou

Táto technika má veľkú výhodu vo vyobrazovaní hustoty dát v multidimenzionálnych priestoroch. Medzi základné typy patrí korelačný diagram (zobrazuje 2D dáta v karteziánskom súradnicovom systéme), matica korelačného diagramu (rozšírenie korelačného diagramu v prípade 3 a viac súradnicového systému,

[2] Klasifikáciou sa venujú ďalšie kapitoly

[3] Klusterizáciou sa venujú ďalšie kapitoly

kde je možné zobrazovať jednotlivé rozmery v matici) a technika paralelných rozmerov.



Obr. 7 Ukážka korelačného diagramu (msys.sk, 2016)

3.2.5.2 Vizualizácia komplexných dát a relácii

Väčšina vyššie uvedených techník je vhodná na dáta, ktoré majú hodnoty nominálneho charakteru. Aktuálne čoraz viac dát je rozmanitejšej štruktúry a na zobrazenie dát, ktoré pochádzajú z textu, sociálnych sietí, alebo iných webov, je potrebné použiť rôzne techniky na ich vizualizáciu. Najznámejším je tag cloud (doslova oblak značiek/tagov, používa sa na užívateľsky prívetivé zobrazenie tém z označených webov a ich stránok).

3.3 Zhlukovanie

Taktiež nazývané aj klusterizácia, je proces rozdeľovania objektov dát do podmnožín na základe ich podobných, alebo spoločných vlastností. Daná podmnožina sa nazýva kluster, alebo zhluk a objekty v danom zhluku sú medzi sebou podobné, ale zároveň sú ich vlastnosti odlišné od objektov, ktoré sú v iný zhlukoch (Han, Kamber, Pei, 2012). Klusterizácia sa tiež nazýva aj automatická klasifikácia v zmysle, že ku klusterom je možné pristupovať ako ku triede a to vďaka rozdielom, ktoré sú medzi triedami a podobnými vlastnosťami v rámci triedy a tiež o zhlukovaní platí, že sa jedná o učenie bez učiteľa, pretože nie je prítomné označovanie a zaraďovanie medzi zhluky prebieha len na základe vlastností dát.

Samotné zhlukovanie má široké použitie nielen v oblasti business intelligence, ale aj v oblasti rozpoznávanie obrázkov, internetové vyhľadávanie, biológia a podobne. V oblasti BI je klusterizácia používaná napríklad na organizáciu

veľkého množstva zákazníkov, kde zákazníci v rámci zhluku zdieľajú spoločné znaky a umožňuje ďalej pracovať na zlepšení správy vzťahu so zákazníkmi (customer relationship). Použitie môže mať aj v hľadaní vzdialených hodnôt (outliners) a to napríklad na vyhľadávaní neobvyklých správaní, hľadaní podvodov a podobne.

Typické požiadavky na zhlukovanie v data miningu sú:

- **Škálovateľnosť** – použitie algoritmu na malom množstve dát rovnako ako aj na dátach rozmerovo v milión položkách.
- **Schopnosť pracovať s rôznymi typmi atribútov** – algoritmy je možné aplikovať nielen na číselné hodnoty, ale aj iné typy.
- **Schopnosť objaviť zhluky aj v nepravidelných tvaroch** – schopnosť hľadať zhluky aj v dátach umiestnených v nepravideľných tvaroch, nielen v sférických klusteroch.
- **Možnosť určiť vstupné parametre –** možnosť určiť napr. počet zhlukov.
- **Schopnosť sa vysporiadať sa s nevyčistenými dátami** – súvisí všeobecne s čistením dát v data miningu, ktoré bolo spomenuté v jednej z predchádzajúcich kapitol.
- **Inkrementálne zhlukovanie a "necitlivosť" na poradie vstupných dát** – niektoré algoritmy zhlukovania sú schopné nové dáta začleniť už do existujúcich výsledkov bez nutnosti ich opätovného spracovania
- **Schopnosť zhlukovať multidimenzionálne dáta** – dáta môžu obsahovať mnoho dimenzií a atribútov.
- **Zhlukovanie s obmedzením** – použitie dát z reálneho sveta môžu byť zhlukované s rôznymi opbmedzeniami, ako napr. hľadanie najbližších bodov na mape s rôznymi překážkami.
- **Interpretovateľnosť a použiteľnosť –** súvisí s jeho konečným použitím; užívatelia by mali byť schopní dáta správne vybrať a interpretovať.

3.3.1 Zhlukovacie metódy

Samostatné zhlukovacie algoritmy sú založené na mnohých zhlukovacích metódach a konkrétne použitie závisí od konkrétnych dát a konkrétnych očakávaných výsledkov. Medzi hlavné metódy zhlukovania patria metódy predeľovaním, metódy založené na hustote, hierarchické metódy a metody založené na mriežke; pre potreby tejto práce budú popísané iba prvé dve z daných metód (Han, Kamber, Pei, 2012).

3.3.1.1 Metódy predeľovaním

Anglicky partitioning methods, je metóda, ktorá má danú sadu n objektov a vytvára k klusterov s tým, že každý zhluk musí obsahovať aspoň 1 objekt; vytvára delenie dát na úrovni jedného levelu. Väčšina metód je založená na vzdialenosti medzi objektami.

3.3.1.2 Metódy založené na hustote

Na rozdiel od väčšine metód, ktoré sú založené na vzialenosti medzi objektami a obsahujú s tým súvisiací problém s hľadaním zhlukov v nepravideľných tvaroch, sú tieto metódy, ako už z názvu vyplýva, založené na hustote (angl. density-based methods). Základnou myšlienkou tvorenia zhlukov je ich rozširovanie až do momentu, kedy dosiahnú nejakú danú prahovú hodnotu; lepšie povedané počet objektov v zhluku musí obsahovať určitý počet objektov. Výhodou je možnosť detekcie šumu v dátach a hľadanie klusterov nepravideľných, nielen sférických tvarov.

3.3.2 Zhlukovacie algoritmy

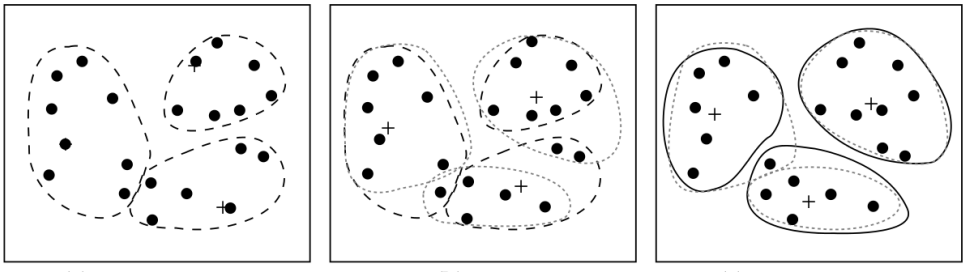
Rôzne zhlukovacie metódy môžu vytvárať rôzne zhluky na rovnakých dátach, nasledujúcich pre potreby práce budú popísané vybrané z nich.

3.3.2.1 k-means

Je asi najznámejší zhlukovací algoritmus, ktorý patrí medzi metódy predeľovaním. Jeho základom je parameter k, ktorý určuje na vstupe výsledný počet zhlukov a jeho algoritmus je implementovaný v 4 krokoch:

1. Predelenie objektov do neprázdnej podmnožiny.
2. Vypočítanie centrálneho bodu pre každý kluster – centroidu. Centroid je vlastne priemer daného klusteru.
3. Každému centroidu sú priradené najbližšie body
4. Pokračovať na bod 4 až do momentu, kedy sa v bode 3 nezmení priradenie bodov k centroidom

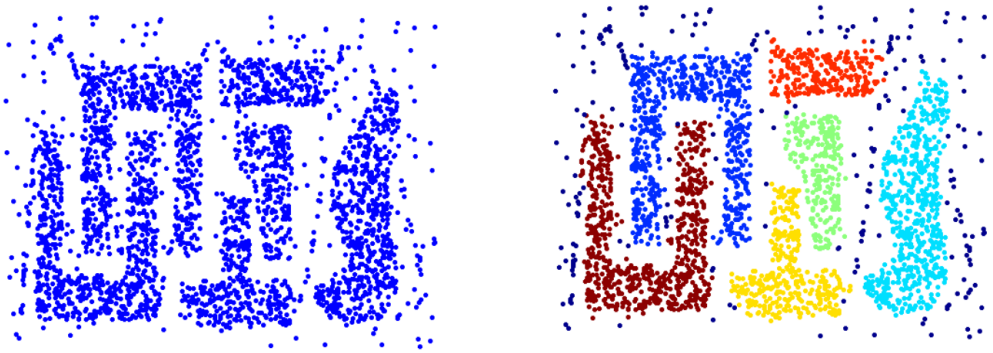
Jeho výhodou je relatívna efektívnosť algoritmu a jednoduchá použiteľnosť. Algoritmus je ďalej dobre preskúmaný, takže je jednoduché pracovať s jeho výstupmi. Na druhej strane je algoritmus nevýhodný pre kategorické dáta, je nutné špecifikovať počet výsledných zhlukov už na začiatku, ako vstup algoritmu. Algoritmus reaguje aj na odľahlé dáta, ktoré sú ako šum a nie je vhodný pre nekonvexné tvary a v neposlednom rade jeho nesprávna inicializácia a umiestnenie počiatočných bodov môžu smerovať k nesprávnym až chybným výsledkom (Vilo, 2014).



Obr. 8 Proces hľadania zhlukov zľava: 1. inicializácia, 2. iterácia, 3. finálne zloženie zhlukov (Han, Kamber, Pei, 2012)

3.3.2.2 DBSCAN

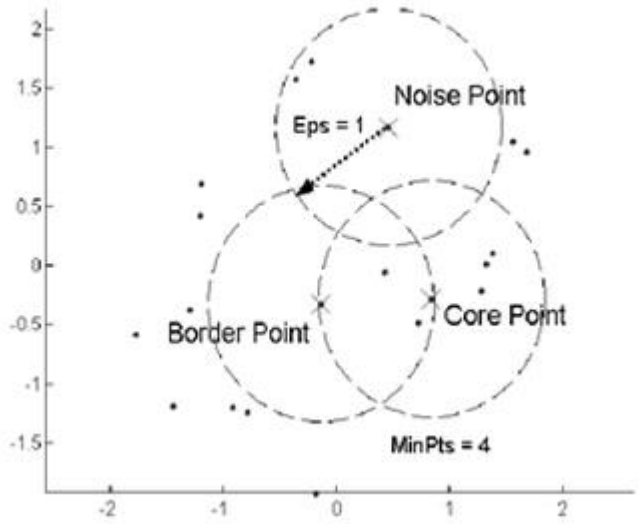
DBSCAN je algoritmus založený na hustote dát, využíva euklidovskú metriku vzdialenosti na určenie vzdialenosti medzi bodmi a na rozdiel od k-means algoritmu umožňuje automaticky určiť počet zhlukov, nájsť zhluky nepravidelných tvarov a taktiež pracovať s odľahlými miestami, ktoré sú definované ako šum v dátach (Witten, Frank, Hall, 2011).



Obr. 9 Ukážka pôvodných dát a ich zaradenie do zhlukov aj s okrajovými bodmi (Vilo, 2014)

Algoritmus má na vstupe parameter ε (epsylon), čo je hustota okolia (vzdialenosť bodov) a minPts, je minimálny počet objektov v zhluku. Pre prácu s algoritmom je potrebné definovať 3 typy bodov:

1. **Hlavný bod** (core point) – bod, ktorý má vo svojom okolí určeným vzdialenosťou ε aspoň minPts bodov.
2. **Okrajový bod** (border point) – bod, ktorý nie je označený ako hlavný bod, má vo svojom okolí aspoň 1 hlavný bod.
3. **Odľahlý bod**, alebo tiež aj šum v dátach (noise point) – bod, ktorý nie je ani hlavný, ani okrajový bod.



Obr. 10 Vizualizácia troch typov bodov (Eick, 2011)

Skrátene sa DBSCAN algoritmus dá popísať v nasledujúcich bodoch:

1. Algoritmus označí všetky body podľa parametrov na hlavné, okrajové a odľahlé.
2. Algoritmus vyberie nepriradený hlavný bod a vykoná vyhľadávanie okolitých bodov podľa parametrov, so začiatkom v danom bode a dané body priradí do rovnakého zhluku, ako bol počiatočný bod.
3. Vyberie ďalší nepriradený bod, ktorý patrí do daného zhluku a opäť pokračuje ako v predchádzajúcom bode.
4. Pokračovať pre všetky hlavné body, pokiaľ nie sú zaradené do nejakého zhluku.

3.3.3 Špeciálne použitia

Algoritmus DBSCAN sa využíva pomerne často na spracovanie priestorových dát a jeho rôzne varianty (VDBSCAN, LDBSCAN, ST-DBSCAN, MDBSCAN a pod.), odstraňujú nedostatky základnej varianty, ako problém s rôznou hustotou; pre potreby práce však postačuje aj jeho základná implementácia (Ahmed, Razak, 2016). DBSCAN pomerne dobre funguje na viacdimenzionálne dáta (Duraiswamy, Mumtaz, 2012).

3.4 Asociačné pravidlá

Algoritmy asociačných pravidel sú založené na frekventovaných vzoroch, čo sú vzory (sady položiek, subsekvencie, alebo subštruktúry), ktoré sa často vyskytujú v nejakej sade dát. Ich využitie je široké, pre účely tejto práce uvediem príklad s analýzou nákupného košíka, ktorý je aj najčastejším použitím tejto

technológie. zákazník v predajni (kamennej, e-shope, ...) nakupuje isté druhy tovaru a otázka je, ktoré kombinácie tovarov sa predávajú najčastejšie, ktorý ďalší tovar (ďalšie tovary) sú najčastejšie nakupované s inou položkou. Ďalšia analýza a aplikácia vhodných nástrojov a krokov môže viesť k zvýšenie predaja oboch typov tovarov odporúčacími algoritmami, výhodnými balíčkami zmiešaných z týchto tovary a podobne (Han, Kamber, Pei, 2012).

Formálne sú asociačné algoritmy podľa Agrawala, Imelinského a Swama definované ako: nech $I = \{i_1, i_2, \dots, i_3\}$ je sada binárnych atribútov nazývané položky. Nech $D = \{t_1, t_2, \dots, t_3\}$ je sada transakcií nazývaná databáza. Každá transakcia v D má unikátne označenie a obsahuje podmnožinu položiek I. Pravidlo je definované ako $X \Rightarrow Y$, kde $X, Y \subseteq I$. Sady položiek X a Y sú jednotlivo predpokladané a vyplývajúce od seba.

S asociačnými pravidlami súvisia pojmy podpora a spoľahlivosť.

Podpora:

$$Podpora\ (A \Rightarrow B) = \frac{(Frekvencia\ výskytu\ A\ a\ B\ v\ I)}{(Počet\ I)}$$

Spoľahlivosť:

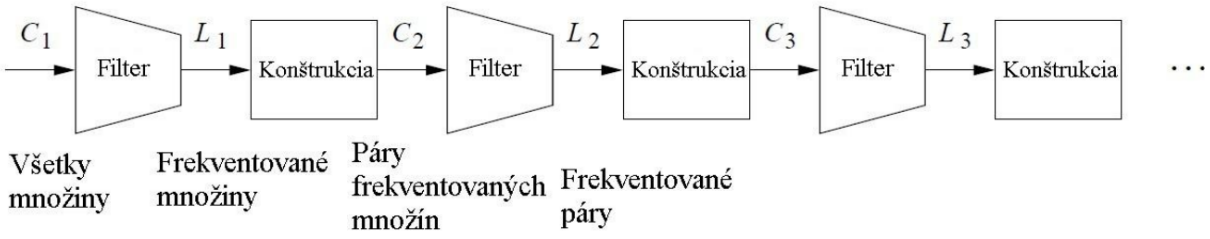
$$Spoľahlivosť\ (A \Rightarrow B) = \frac{(Frekvencia\ výskytu\ A\ a\ B\ v\ I)}{(Počet\ A\ v\ I)}$$

kde `A` a `B` sú položky (tovary) a `I` je celá sada dát, ktorú algoitmus spracováva a oba pojmy súvisia s pojmom minimálna prahová hodnota s ktorým algoritmy pracujú.

3.4.1 Apriori algoritmus

Najznámejší algoritmus asociačných pravidel apriori je založený na fakte, že ak neexistuje kombinácia n tovarov, ktoré dosiahli na vstupný prah podpory a spoľahlivosti, pridaním ďalšej položky pri iterácií spracovania tento prah celkom určite prekročený nebude (Leskovec, Rajaraman, Ullman, 2013).
Algoritmus pracuje v dvoch krokoch (Han, Kamber, Pei, 2012):

- Spojovací krok – vytvorenie kandidátov $n + 1$ pripojením $k + p$, kde n je počet vytvorených predchádzajúcich n-tíc, k je poradie posledne pridanej položky a p je relatívne poradie ďalšej pridanej položky. Na začiatku sú vytvorení kandidáti o veľkosti 1 zo všetkých atribútov.
- Vylučovací krok – z vytvorených kandidátov sa vyradia asociácie, ktoré nespĺňajú minimálnu podporu, alebo spoľahlivosť; ako bolo vyššie spomenuté, ak nájdeme asociácie s určitou podporou n položiek, tak ak k n položkám pridáme ďalšiu položku na $n + 1$, podpora tejto dátovej sady nemôže byť určite vyššia, celú vetvu je teda možné vyradiť zo spracovania.



Obr. 11 Kroky apriori algoritmu (Leskovec, Rajaraman, Ullman, 2013)

3.5 Text mining

Text mining, na rozdiel od data miningu, ktorý sa zaoberá hľadaním vzorov v dátach, sa venuje hľadaním v texte. Ďalším rozdielom je, že zdrojové dáta z textu nie je možné extrahovať tak jednoducho, ako zo štruktúrovaných dát a text musí byť najskôr predspracovaný na formu vhodnú pre existujúce algoritmy. Samotné "spracovanie textu" čítaním človekom je totiž časovo náročné a z pohľadu sémantiky dokáže text miningu nájsť potenciálne využiteľné informácie. Text mining algoritmy sa používajú na klasifikáciu a klusterizáciu dokumentov, pre potreby tejto práce bude popísaná klusterizácia textových dokumentov.

3.5.1 Stop words

Stop words (stop slová) sú slová funkčné, bez podstatného významu pre obsah textu. Vyskytujú sa oproti ostatným slovám veľmi frekventovane a v rámci predspracovania by sa mali odstrániť, lebo nie sú pre účely použitelné. Pre každý jazyk existuje zvláštna sadu stop slov (Witten, Frank, Hall, 2011).

3.5.2 Model reprezentácie dokumentov

Model priestorového vektoru sa stal široko používaným modelom na reprezentáciu textových dát na pre spracovanie textových dát. Dokumenty sú reprezentované ako m-dimenzionálne vektory, kde m je počet unikátnych slov, ktoré sú už po predspracovaní. Každý dokument je reprezentovaný vektorom vlastnosti $d = (w_{d1}, \dots, w_{dm})$. Každá komponenta tohto vektoru určuje stupeň väzby asociovanými výrazmi a dokumentom, čo sa tiež aj nazýva váha výrazu (Vector space model – VSM).

Najjednoduchšie by sa jednalo o binárne hodnoty (`0`,`1`). Hodnota `1` pre prítomnosť a `0` pre neprítomnosť výrazu v dokumente. Častejšie využívaná je však metóda váhy výrazu. Táto metóda prináša lepšie výsledky pre výrazy, ktoré sa často vyskytnú vo viac špecifických dokumentoch, ale objavujú sa zriedkavo v celej kolekcii, a dáva sa im väčšia váha – táto metóda sa nazýva Term Frequency-Inverse Document Frequency (`TFIDF`) a dôležitosť výrazu proporcionálne stúpne množstvu výskytu v danom dokumente, ale je ovplyvnený celkovým vý-

skytom vo všetkých dokumentoch (TUNALI, BILGIN, ÇAMURCU, 2016) TFIDF hodnota výrazu w_{dt} sa počíta pomocou:

$$w_{dt} = (1 + \log(f_{dt}) \times \log(1 + \frac{n}{f_t}))$$

kde f_{dt}je frekvencia výskytu výrazu t v dokumente d, n je počet dokumentov v kolekcii D a f_t je počet dokumentov, ktoré obsahujú výraz t.

	doc1	doc2	doc3
I	1	0	0
like	1	0	0
football	1	1	0
John	0	1	1
likes	0	1	1
football	1	1	0
basketball	0	0	1

Obr. 12 Ukážka binárneho VSM; stĺpce vyjadrujú dokument, riadky výrazy a obsah tabuľky udáva prítomnosť výrazu *X* v dokumente *Y* (Molino)

3.5.3 Použitie k-means v oblasti text miningu

S použitím správnej reprezentácie, konkrétne vyššie uvedený vektorový zápis, je možné použiť k-means algoritmus na zhlukovanie týchto dát na určenie kategórie, do ktorej môže textový dokument patriť (TUNALI, BILGIN, ÇAMURCU, 2016). V jednej z ďalších častí práce je taktiež popísané, že implementácia k-means vo Weka algoritmoch obsahuje taktiež TFIDF transformáciu, čo je ďalší dôvod na použitie tohto algoritmu v práci.

4 Štatistika a intervalové a skupinové triedenie

Štatistika, ako odbor, sa dá podľa Hana, Kambera a Peia (2012) v určitej miere zaradiť do data miningu, pretože sa jedná o získavanie, analýzu, interpretáciou a prezentáciou dát a samotné algoritmy data miningu od klasifikácie cez klusterizácie používajú štatistické metódy. Táto práca sa však na štatistické metódy nezameriava, pre jej potreby však budú popísané 2 termíny a to aritmetický priemer[4] a intervalové a skupinové triedenie.

V prípade väčšieho počtu znakov je možné ich rozdeliť do intervalov (v prípade diskrétnych hodnôt). Na ich určenie neexistujú jednoznačné pravidlo. Blažková a kol. (2015) uvádzajú, že interval by nemal byť príliš malý, pre stratu podstatných informácií, ale ani príliš veľký, aby nebola znížená prehľadnosť tried. Počet tried je možné určiť vzťahom

$$k \doteq \sqrt{n}$$

Pre určenie počtu tried je jeho vyrátanie jeho šírky už jednoduché

$$h \doteq \frac{x_{max} - x_{min}}{k}$$

h je označenie šírky, x_{max} je maximálna a x_{min} minimálna hodnota znakov. Samotný prvý interval je obvykle zdola neohraničený a taktiež posledný je zhora neohraničený a zodpovedá tomu tvar

$$\left\langle x_i - \frac{h}{2}, x_i + \frac{h}{2}\right), i = 1,2, \dots k.$$

[4] Aritmetický priemer je pojem tak známy, že jeho popis je zbytočný

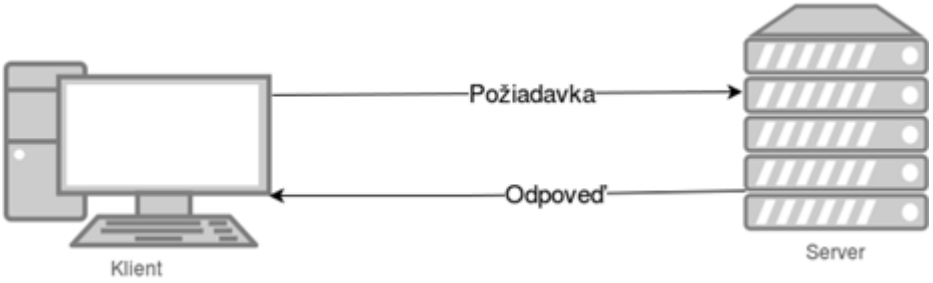
5 Použité technológie

5.1 Webové technológie

Aplikácia dostupné online sú v súčasnej dobe založené na rôznych technológiách a platformách, najpoužívanejšiou kombináciou technológií sú však s veľkou väčšinou založené na PHP (Web technology surveys, 2016) a to kombinácia technológii LAMP – Linux, Apache MySQL, PHP.

5.2 Klient – server architektúra

Najpoužívanejším spôsobom prenosu a spracovania dát na internete je v súčastnosti architektúra klient – server, ktorá podľa Chaudhuryho a Raa (1996) rozdeľuje spracovanie dát na 2 časti – klient si dáta vyžiada a server ich poskytne. Klient posiela na spracovanie serveru jeden, alebo viac požiadaviek a server dokáže spracovať mnoho požiadavkov od veľkého množstva klientov. Medzi široké využitie tohto konceptu patrí zdieľanie dát v databázach, distribuované spracovanie dát.



Obr. 13 Vizualizácia architektúry klient – server

5.3 HTML

Niederst (2012) definuje HTML (HyperText Markup Language) ako jazyk na vytváranie webových dokumentov. Nie programovací, ale značkovací jazyk, pretože funguje ako systém na identifikáciu a popis rôznych komponent dokumentov (ako nadpisy, odstavce, zoznamy) na vloženie rôznych médií, obrázkov, videí a nástrojov, ako formulárových prvkov. HTML značky indikujú zátvorky < a > a každý element je nimi určený. Najnovšia verzia HTML5, okrem už existujúcej funkčnosti predchádzajúcej verzie, obsahuje nové prvky pre súčasné webové trendy, ako articles, sections, headers, footers a prvky multimédii video, audio, canvas (Goldstein, Lazaris, Weyl, 2011). Ku prvkom HTML sa na strane client – side spracovania pristupuje pomocou DOM[5] a umožňuje tak jeho manipuláciu po dynamickej stránke.

[5] Document Object Model

5.4 CSS

Kaskádové štýly (CSS) sú neoddeliteľnou súčasťou tvorby internetových stránok. Goldstein, Lazaris a Weyl (2011) definujú CSS ako štýlovací jazyk na popis, ako sa HTML prezentuje a štylizuje; príkladom je rozloženie, farby, fonty. V novej verzii CSS3 zavádza nové prvky a funkcie, ako tiene, farebné prechody, alebo okrúhle rohy prvkov. Tieto nové funkcie bolo v minulosti nutné dosahovať rôznymi technikami. Na určenie časti HTML, ktorá je ovplyvnená deklaráciou vlastností CSS, môže byť príkladom `ID` značky, `CLASS` triedy aj s rôznymi pseudodeklarácie, napr. `:hover`, ktorá aplikuje danú deklaráciu na elementy, nad ktorou je zrovna kurzor myši.
Vloženie CSS deklarácii prebieha vložením do prvku, definovaním pravidiel priamo v HTML dokumente, alebo priložením externého súboru s definíciami.

5.5 Apache

Apache je projekt za účelom vytvárania robustných, komerčných, funkčných a široko využívaných kódov na implementáciu HTTP (Web) serveru (Apache HTTP server project, 2016). Z dlhodobého hľadiska je tiež najpoužívanejší z najznámejších serverových riešení (Netcraft, 2016).

Medzi jeho zásadné vlastnosti a funkcie patrí (Apache HTTP server project, 2016):

- jeho rozšírenosť,
- podpora širokého rozpätia programovacích jazykov (Perl, Python, PHP),
- podpora šifrovania HTTP spojenia,
- široká modulovateľnosť (napr. *mod_rewrite* na prepisovanie URL adries, alebo *mod_proxy* na preposielanie HTTP požiadavkov).

5.6 API

API je skratka pre Application Programming Interface, v preklade rozhranie pre programovacie aplikácie. Goldstein, Lazaris a Weyl (2011) uvádzajú porovnanie s grafickým rozhraním. Tak, ako grafické rozhranie je rozhranie pre nás, ľudí, API je rozhranie pre počítače a poskytuje sadu preddefinovaných funkcií, pomocou ktorého je možné dosiahnúť očakávané správanie; dostať sa k informáciam.

5.7 Databázy

Databázy (Sheldon, Moes, 2005) sa stali jednou z hlavných súčastí webových aplikácií a na ukladanie dát a spoliehajú na nich aplikácie rôzneho druhu. Databázami sa teoreticky môže označiť rôzne médium na ukladanie dát od kolekcie súborov až po komplexné štruktúry a definuje ju objem uložených dát, komplexicita dát, počet elementov a podobne. Súčasné databázy sú väčšinou založené

na hierarchickom modeli, ktorý poskytuje jednoduchú štruktúru, kde jednotlivé záznamy sú organizované do vzťahov rodič-potomok, čím napomáha logickej kategorizácií a podkategorizácií dát do logických prvkov dát, konkrétne v relačnom modeli. Na menšie projekty a jednoduhšie štruktúry sú tiež používané súborové dazabázy, kde celá databáza je uložená v jednom súbore; najznámejším predstaviteľom je *SQLite* (SQLite, 2016). Databázové systémy sú ako také iba koncepty, funkčnosť im dáva tzv. systém riadenia bázy dát (angl. database management system – DBMS), ktorý poskytuje softwarové nástroje na prácu s dátami. Príkladom DBMS sú *Oracle*, *DB2*, *SQL Server*, *PostgreSQL* a *MySQL*, ktorý je v práci popísaný ďalej. Medzi hlavné funkcie DBMS patria:

- správa úložiska
- správa bezpečnosti
- správa pomocných dát
- poskytovanie konektivity/pripojenia
- optimalizácia výkonu
- poskytovať zálohu a obnovu dát
- spracovávať príkazy na získavanie a modifikáciu dát

Medzi funkcie MySQL, ako DBMS použitého v tejto práci patrí:

- **škálovateľnosť** – využívaná veľkými spoločnosťami, dokáže ukladať milióny až miliardy záznamov,
- **prenosnosť** – MySQL funguje na širokej škále operačných systémov založených na rozdielnej architektúre,
- **konektivita** – podporuje TCP/IP sokety a obsahuje API na rôzne programovacie jazyky (PHP, C, C++, Perl, Java, ...),
- **bezpečnosť** – systém riadi prístupy k dátam a taktiež podporuje SSL[6] pre šifrované pripojenia,
- **rýchlosť** – systém bol vytvorený s prihliadnutím na rýchlosť práce s ním,
- **jednoduchosť použitia** – jednoduchá inštalácia a práca s MySQL umožňuje pracovať s MySQL v minútach,
- **Opensource licencia** – kód je možné si voľne stiahnúť.

5.7.1 SQL

Na komunikáciu s DBMS systémami sa používa jazyk SQL, pomocou ktorého je možné manipulovať s dátami, získavať dáta, kontrolovať prístup k ním a zabezpečovať integritu dát. Príkazy sa v SQL sa delia na DDL, DML a DCL[7].

DDL príkazy slúžia na prácu so štruktúrou dát, napr. vytváranie databází (príkaz `CREATE DATABASE`), tabuliek (`CREATE TABLE`), úprava tabuliek (`ALTER TABLE`) alebo ich mazanie (`DROP TABLE`).

[6] Secure socket layer

[7] Data definition language, Data manipulation language, Data control language

Pomocou DML príkazov, na rozdiel od DDL, sa pracuje priamo s dátami – výber (`SELECT`), pridanie (`INSERT INTO`), úprava (`UPDATE`), mazanie (`DELETE FROM`). Podrobným popisovaním všetkých funkcii a ich použitím sa táto práca nezaoberá, špeciálne by som spomenul však kľúčové klauzule a funkcie.

- klauzula `GROUP BY` – umožňuje zoskupovať hodnoty spolu vzhľadom na spoločné hodnoty. Výsledkom sú skupiny záznamov unikátných hodnôt na strane jednej a sumárne informácie na strane druhej
- funkcia `SUM` – vráti súčet hodnôt
- funkcia `AVG` – vráti aritmetický priemer hodnôt
- funkcia `COUNT` – vráti počet hodnôt
- funkcia `IF` – podmienená funkcia v MySQL
- funkcia `CONCAT` – hodnoty v argumente spojí do jedného reťazca
- funkcia `CURDATE` – vráti aktuálny dátum
- funkcia `DATE` – prevedie čas na dátum
- funckia `LAST_DAY` – vráti posledný deň mesiaca v argumente
- funkcia `DATE_ADD` – k danému dátumu pridá časové obdobie

Príkaz `SELECT DISTINCT` je najčastejším príkazom pri dotazovaní z dátového skladu. Nad týmito dátami je možné pomcou SQL príkazov vytvárať rôzne analýzy a reporty, vrátane požadovaných agregovaných hodnôt (Novotný, Pour a Slánsky, 2005).

5.8 Návrhové vzory

Pecinovský (2007) prirovnáva návrhové vzory k matematickým vzorcom, ale na rozdiel od nich sa do vzorov nedosadzujú čísla, ale triedy, rozhrania a objekty. Jedná sa o odporúčané postupy riešenia časti sa opakujúcich úloh. V rámci práce budú použité 3 vzory:

- **singleton (jedináčik)** – trieda, ktorá bude maximálne jednu inštanciu
- **container (prepravka)** – zlučuje niekoľko samostatných objektov do jedného, vďaka čomu je možné tieto informácie ľahšie ukladať a prenášať
- **factory (továreň)** – definuje statickú metódu nahradzujúcu konštruktor a jej použitie je prí získavaní odkazu na objekt, kde ale priame použitie konštruktoru nie je z rôznych príčin optimálne

5.9 PHP

Hypertext Preprocessor (skrátené PHP) je, ako uvádza oficiálny manuál (PHP, 2016), je široko využívaný, opensource skriptovací jazyk, ktorý je špecializovaný na vývoj webových aplikácii a môže byť vložený do HTML. Samotný kód je založený na sytnaxi jazyku C a je spúšťaný na strane serveru, kde spracováva požiadavky a vracia výsledný (väčšinou HTML) kód. Môže sa využívať aj ako skripto-

vací jazyk, alebo jazyk na vývoj desktopových aplikácii; táto práca sa však týmto použitím nezaoberá. Silnou stránkou je podpora širokého spektra rôznych databází na vývoj databázovo orientovaných aplikácií, podpora e-mailových protokolov a veľkého množstva iných knižníc.

V súčasnosti sa však pomaly upúšťa od písaní aplikácií v čistom PHP a viac sa využívajú na vývoj frameworky, ktoré z veľkej časti urýchľujú vývoj, poskytujú predpripravené funkcie a štruktúry a podporujú "dobre návyky" programátorov pri programovaní (Niederst, 2012).

5.10 Javascript

JavaScript popisuje Niederst (2012) ako skriptovací jazyk, ktorý slúži na pridanie interaktivity na webových stránkach na strane klienta, menovite niektoré, patria medzi nich:

- kontrola validity formulárov,
- úpravy vzhľadov a štýlov elementov na stránke,
- ukladanie informácii o užívateľoch pre ich opätovné navštívenie,
- budovanie interaktívnych prvkov.

Prístup k HTML elementov je možný cez DOM štruktúru a odkazuje sa k nemu ako DHTML (Dynamic HTML). Taktiež, ako aj pri PHP, aj pri práci s JavaScriptom sa v praxi neprogramuje v jeho "čistej" forme, ale sú používané JavaScriptové knižnice a frameworky; taktiež úplne neplatí pravidlo, že JavaScript je využívaný len čisto na programovanie na strane klienta, ale v poslednej dobe vzniká aj široké využitie Javascriptu, ako programovacieho jazyku v iných oblastiach (napr. programovanie serverových aplikácii).

5.11 jQuery

Knižnica napísaná v roku 2005, jQuery je bezplatná a v súčasnosti úplne najpopulárnejšia javascriptová knižnica vydávaná pod licenicou open source. Umožňuje zjednodušenie práce s JavaScriptom pre vývojárov z pohľadu kódu, CSS a DOM štruktúre HTML.

Jeho nadstavba, jQuery UI, pridáva do funkčnosti nové prvky (Niederst, 2012)., napr. (jQuery UI, 2016):

- **nástroje** – autocomplete, button, datepicker, dialog, menu, slider, ...
- **efekty** – pridanie triedy, farebné animácie, schovávanie, ...
- **interakcie** – draggable, droppable, resizable, selectable, sortable

5.12 Grafové knižnice

Pre uľahčenie vizualizácie dát, či už jednotlivých, alebo periodických, existujú na vykreslenie grafov na internetových stránkach k dispozícii rôzne nástroje. Ich výhodou je jednoduchosť implementácie (pracujú na pomerne priamočarých

a pochopiteľných vstupoch) a šírka ponúkaných vizualizácii (neobsahujú len základné grafy ale pomerne širokú paletu rôznych vizualizácii). Medzi známe javascriptové knižnice patria *Google Charts*, *ChartJS*, alebo *Highcharts JS*, no v rámci tejto práce bola použitá knižnice *RGraph*.

5.13 Smarty

Smarty je šablónovací nástroj napísaný v jazyku PHP a umožňuje uložiť vizuálny obsah internetových stránok (HTML, CSS) od jeho backendu. Veľkými výhodami použitia ľubovolného šablónovacieho nástroju sú (What is Smarty, 2016):

- Premenné výstupov sú skladané medzi HTML značky, čo zlepšuje prehľadnosť oddelenia čistého PHP kódu od HTML,
- umožňuje editovať obsahovú stránku oddelene a nezávisle od zvyšku aplikácie,
- poskytuje rôzne funkcie na celkové zlepšenie práce s obsahom stránok.

Šablónovacie nástroje delia obsah do blokov, ktoré obsahujú svoj identifikátor a tie sú volané nezávisle za účelom poskladania celého obsahu stránky (Brampton, 2008).

5.14 MVC

Architektúra Model-View-Controller je podľa Bramptona (2008) spôsob vývoja softwaru, kedy je výsledný systém rozdelený na 3 samostatne oddelené.

View predstavuje vrstvu, pomocou ktorej komunikuje program s používateľom; zobrazenie je prevádzané cez HTML a CSS súbory, vo väčšine prípadov predspracované cez šablóny.

Model dodáva na základe parametrov dáta; tie môžu byť z databázy, súborov, alebo API.

Controller má na starosti komunikáciu medzi oboma zvyšnými vrstvami; spracováva požiadavky pre model a predpripravuje výstupy do view. Medzi hlavné výhody tejto štruktúry patria nezávislosť jeho častí, a tým jednoduchá obmena, prehľadnejší vývoj a celková oddelená logika fungovania systému.

5.15 PHP frameworky

Ako bolo v jednej z vyššie uvedených kapitol popísané, programovanie v čistom PHP v súčastnosti prekrývajú veľké možnosti vývoja pomocou frameworkov. Pri otázke prečo ich uprednostňovať pred čistým PHP existuje niekoľko dôvodov (Hongkiat, 2015):

- rýchlosť vývoja,
- dobre organizovaný kód a jeho znovupoužiteľné časti,
- lepšia škálovateľnosť webových aplikácii,
- zlepšenie bezpečnosti aplikácií na najnižšej úrovni,

- aplikácie vytvorené cez framework podporujú MVC model,
- prítomnosť moderných programovacích praktík,
- veľké množstvo pluginov a predpripravených funkcii.

Funkcie, o ktoré sa kvalitný framework postará (Brampton, 2008):

- práca s formulármi,
- autentikácia a autorizácia užívateľov,
- ladenie chýb,
- práca s URL adresami (routovanie),
- lepšia práca s databázemi,
- jazykové mutácie aplikácie,
- šablónovací systém pre prácu s vizuálnou stránkou.

Príklady súčasných PHP frameworkov (Hongkiat, 2015):

- **Laravel** – v súčasnosti celosvetovo najpoužívanejší famework. Obsahuje svoj vlastný templatovací systém Blade, má elegantnú syntax, ktorá poskytuje veľké množstvo často používaných funkcii.
- **Symfony** – na tomto frameworku sú založené známa projekty, ako Drupal, alebo phpBB a obsahuje veľké množstvo použiteľných knižníc.
- **CodeIgniter –** je veľmi jednoduchý nástroj s minimálnymi požiadavkami na inštaláciu a konfiguráciu. MVC architektúra nie je tak striktne dodržiavaná, takže poskytuje vývojárovi voľnosť, ale zároveň obsahuje veľké množstvo modulov pre ďalšie požadované funkcie.
- **Yii 2** – zameraný na rýchlosť, objektovo založený, s logickým kódom a pokročilými nástrojmi na tvorbu front-endu.
- **Phalcon** – taktiež zameraný na rýchlosť a s množstvom nástrojov ako auto-loader, nástroje na preklady, bezpečnosť a podobne.
- **CakePHP** – starší, ale v najnovšej verzii stále aktuálny framework, využívaný veľkými spoločnosťami a s pokročilými bezpečnostnými prvkami.
- **Zend** – jeho robustnosť spôsobila to, že nie je vhodný na menšie projekty. Obsahuje zaujímavé nástroje pre kryptografiu, front-end a celkovo je stavaný pre klientov väčšieho rozmeru.
- **Slim** – minimalistický framework vhodný pre malé projekty. Obsahuje routovanie, flash správy a podobne.
- **Fuel PHP** – založený na modulovateľnosti a rozšíriteľnosti.
- **PHPixie** – podľa oficiálnej dokumentácie sa ho môžete naučiť za 30 minút. Programátor v ňom nájde okrem iného nástroje ORM[8], validáciu formulárových prvkov, autentikáciu, autorizáciu a routovací systém.
- **Nette** – je český projekt programátora Davida Grudla (Nette framework, 2016). V tuzemských podmienkach hojne využívaný, obsahuje veľké množstvo nástrojov pre prácu s databázami, formulármi, vlastný šablónovací systém a to všetko s podporou širokej komunity.

[8] Object-related mapping

5.16 Google maps API

Google Maps je internetová služba, ktorá poskytuje mapy, satelitné snímky a zábery ulíc. Ako službu poskytujú API rozhranie pre aplikácie tretích strán, ktoré môžu byužívať nielen mapy, ale aj rôzne iné informácie z oblasti geografických informácii. Uvedené služby poskytujú pre weby, webové služby, Androidové a iOSové aplikácie do určitého počtu požiadaviek denne zdarma, preto sú vhodným riešením pre menšie alebo súkromné aplikácie, ktoré nevyužíva mnoho ľudí (Google Developers, 2016).

5.17 Java

Java je objektovo orientovaný multiplatformový programovací jazyk. Kompilácia prebieha do tzv. byte-code a ten je následne možné spustiť na mnohých platformách nezmenený (Herout, 2008). Aktuálne patrí medzi najpoužívanejšie programovacie jazyky a fungujú na ňom rôzne služby a platformy, ako napr. servery alebo Android zariadenia. Do vyvíjaných programov je možné vložiť už existujúce baličky kódov – knižnice, tzv. JAR súbory (Oracle 2016).

5.17.1 Vláknové programovanie

Java, ako programovací jazyk, podporuje multivláknové programovanie; kód nemusí bežať sekvenčne ako jeden celok, ale niektoré jeho procesy, ktoré sú nezávislé, a priestorovo/časovo náročné, môžu byť spustené paralelne. Použitie má v časovo náročných operáciach, pri čakaní na vstup od užívateľa, pri opakujúcich sa výpočtoch, alebo pri úlohách typu producent-konzument. Základ tvorí trieda `Thread`, ktorá poskytuje základ pre viacvláknové programy. V rámci komunkácie medzi vláknami a ich paralelného pristupovania k zdrojom, programátor musí zabezpečiť medzi nimi synchronizáciu, väčšinou pomocou synchronizovaných metód; toto však nie je pravidlom a existujú rôzne prístupy k viacvláknovým programom, táto práca sa im však nevenuje.

5.17.2 Kolekcie

Použitie klasických polí nielen v Jave prináša komplikácie s indexovaním, ich deklaráciou a v niektorých prípadoch je nutné použiť kolekcie. To je typ premennej, ktoré združujú mnohonásobné a mnohodimenzionálne štruktúry do jedného objektu. Príkladom kolekcií sú `ArrayList`, `LinkedList`, `Stack`, `Queue`, `HashSet`, alebo `HasMap` (Burd, 2014).

5.18 Jetty

Pomocou projektu Jetty je možné vytvárať HTTP servery. Jeho výhodou je jednoduchosť, malá veľkosť a rýchlosť vývoja (Jetty, 2016).

5.19 Weka

Je v Jave napísaný framework, ktorý obsahuje kolekciu data mining, machine learning algoritmov a nástrojov na predspracovanie dát na akademickej úrovni. Poskytuje nástroje na celý proces dolovania dát, vrátane prípravu dát, vitualizáciu dát aj výsledkov. Bola vyvinutá na University of Wakaito na Novom Zélande, napísaná v Jave, je multiplatformová a vydávaná pod licenciou GNU-GPL. Pomocou Weky je možné riešiť hlavné problémy dolovania dát – regresiu, klasifikáciu, zhlukovanie aj asociačné algoritmy, aplikovať predprípravné procesy, dáta predikovať, algoritmy parametrizovať a na vstupe dát sú možné rôzne formáty, napr. vlastný formát ARFF alebo CSV.

Nástroj poskytuje pomerne intuitívne GUI pomocou troch rozhraní – Explorer, the Knowledge Flow a Experimeter. V práci však uvedené algoritmy využijem pomocou ďalšej možnosti, ktorý Weka poskytuje, a to je priamo ako Java knižnicu vo vlastnom kóde (Witten, Frank, Hall, 2011).

5.20 JSON

Komunikácia medzi systémom a API prebieha pomocou štruktúrovaných dát, väčšinou JSON. JSON (JavaScript Object Notation) umožňuje vytvárať vlastné štruktúry na prenos dát (Vrána, 2010).

6 Súčasný stav

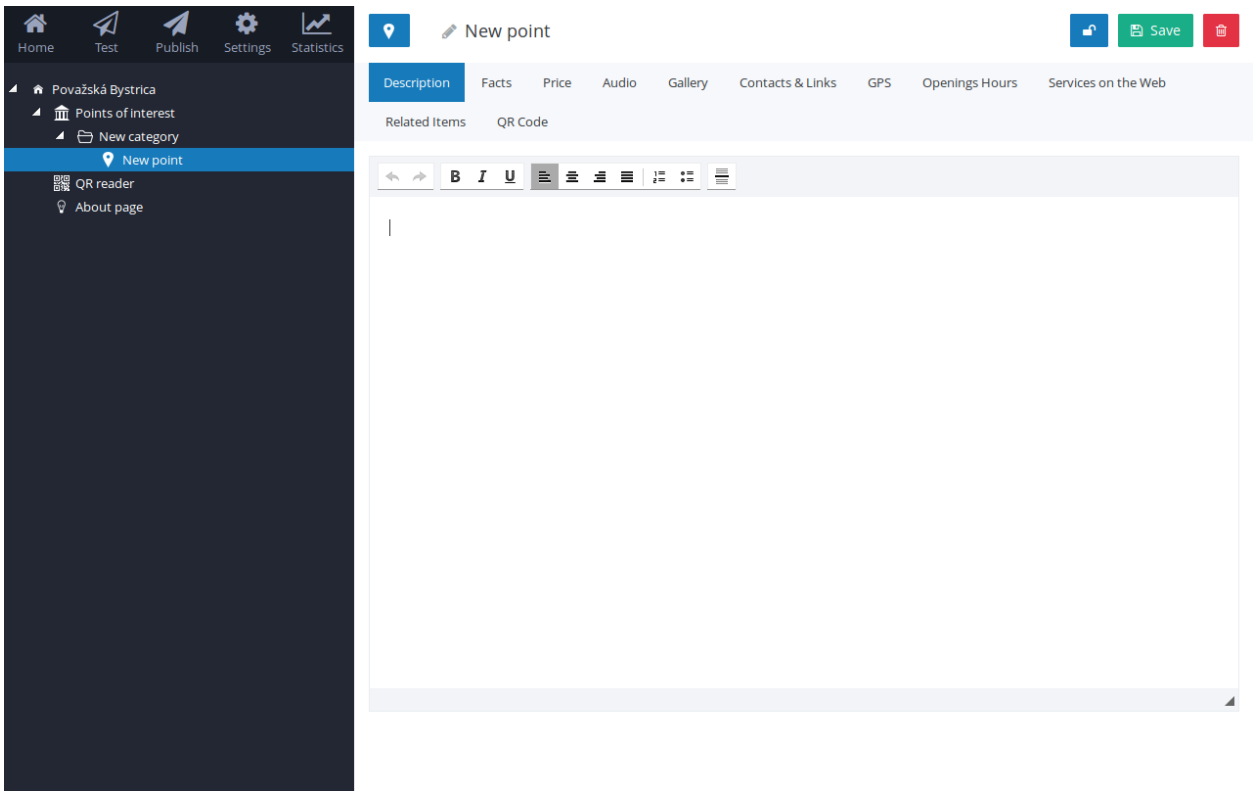
6.1 Effortix

Effortix[9] je komplexný SaaS[10] systém od brnenskej spoločnosti LWi s.r.o. pre koncových užívateľov na vytváranie mobilných aplikácii bez nutnosti znalosti programovania. Samotný proces vytvárania prebieha vo webovej administrácii a výsledkom sú identické aplikácie pripravené na použitie na Android a Apple mobilných zariadeniach (Effortix, 2016). Medzi základné aj pokročilé funkcie systému patria:

- nákupný košík s katalógom a objednávkovým systémom,
- dokumenty, súbory,
- možnosť vytvárať vlastné fomuláre,
- podpora mnohojazyčnosti a niekoľkých svetových mien,
- vytváranie rôznych základných stránok, odkazov, textov, noviniek, ...,
- pokročilé nástroje na tvorbu turistických aplikácii (mapy, služby, pamiatky, ...),
- prehľadná administrácia na správu obsahu, objednávok, prijatých vyplnených formulárov,
- testovanie aplikácie cez testovacie mobilné aplikácie zvane TestDrive skôr, ako sa ich uživateľ rozhodne publikovať.

[9] http://www.effortix.com

[10] Software as a service

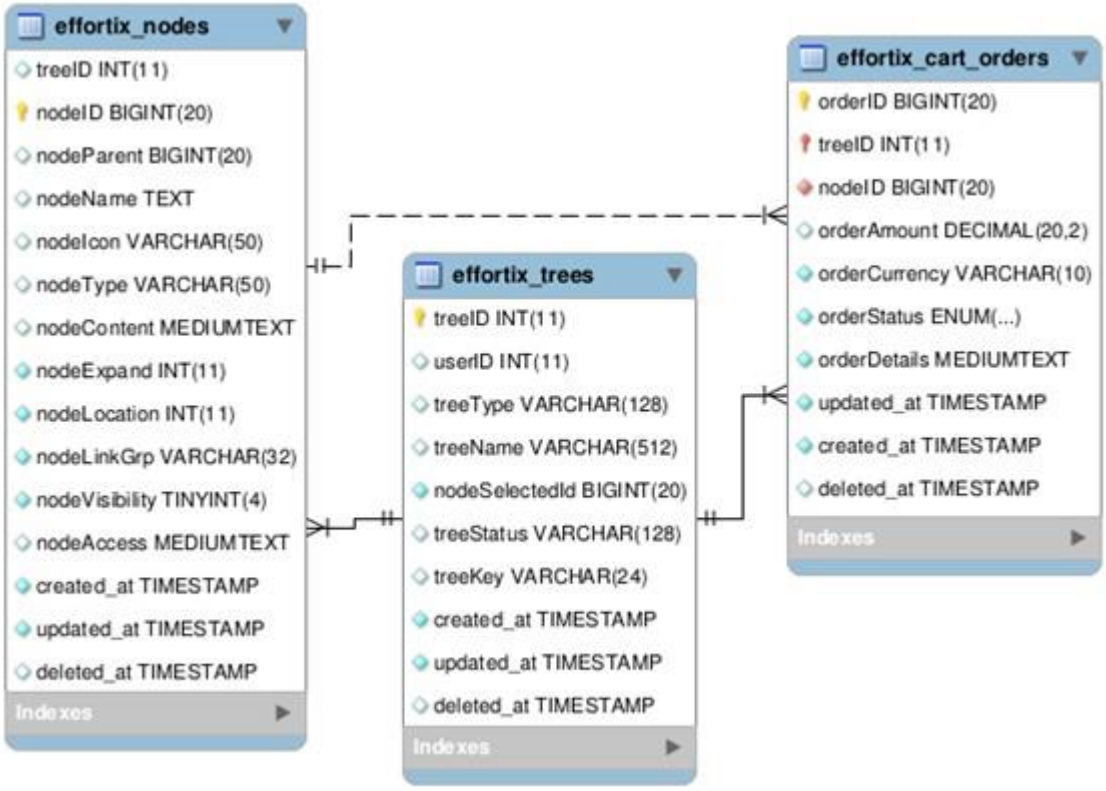


Obr. 14 Ukážka administrácie Effortix v internetovom prehliadači

Systém poskytuje 3 typy tarifov pre klientov – *Free* (základné funkcie), *Pro* (oproti Free má k dispozícii funkcie na vytváranie aplikácii pre cestovný ruch a vlastné formuláre), *Business* (obsahuje aj nástroje na objednávky a košík), čo je kľúčové pre ďalšiu analýzu.

6.2 Analýza systému

Všetky dáta celého systému sú uložené v dvoch databázach. Do prvej databázy sa ukladajú všetky dáta súvisiace s obsahom systému – stromy aplikácií, položky, užívatelia, obsahy košíku, objednávky, vyplnené formuláre a pod. Databáza má 13 tabuliek, pre potreby práce však popíšem len 3 z nich, a to `effortix_cart_orders`, `effortix_nodes` a `effortix_trees`. Druhá databáza ukladá informácie o integrovanom platobnom systéme, publikáciach aplikácii a iné, pomocné údaje mimo obsahu aplikácii. Táto databáza obsahuje 47 tabuliek, v práci budú použité 3 z nich, konkrétne `server_bin_data_dates`, `server_invoice_orders` a `server_invoice_pay_progs`.



Obr. 15 Výber z databázy obsahu, ktorý bude v práci použitý

6.2.1 Tabuľka effortix_trees

Môže byť označená ako hlavná/centrálna tabuľka systému. Jednotlivé záznamy predstavujú vytvorené aplikácie systémom a ich základné údaje (názov, užívateľa, ...).

6.2.2 Tabuľka effortix_cart_orders

Tabuľka v sebe obsahuje informácie o objednávkach, ktoré užívatelia odoslali. Kľúčovým atribútom je `orderDetails`, kde je v JSON štruktúre uložená kompletná informácia o objednávke (ceny v rôznych menách, položky vo všetkých jazykoch, počet kusov) a o odosielateľovi (meno, adresa, ...). Samozrejmosťou je nezahrňanie objednávok, ktoré boli zmazané, teda hodnotu `deleted_at` nie je *`NULL`*.

6.2.3 Tabuľka effortix_nodes

V tejto tabuľke sú uložené štruktúry aplikácii, informácie o nich a celková hierarchia položiek v aplikáciách. Podstatný je atribút `nodeContent`, ktorý obsahuje informáciu o uzle v JSON formáte. Každý uzol je však iného typu a tak jeho

spracovanie závisí od obsahu. Týmto je v rámci systému docielená ohromná flexibilita obsahu, možnosť pridávania nového a typov uzlov.

6.2.4 Tabuľka server_bin_data_dates

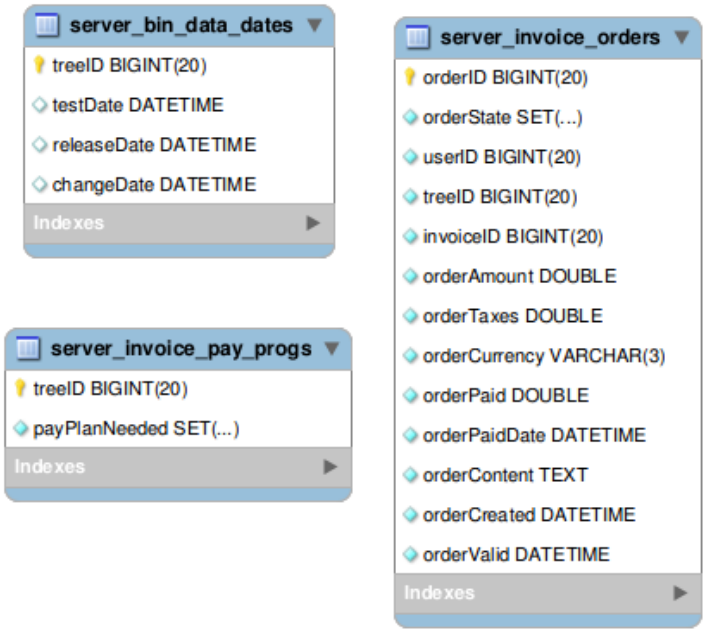
Tabuľka obsahuje časové informácie o základných aktivitách aplikácie, ako čas publikácie alebo testovania.

6.2.5 Tabuľka server_invoice_orders

Tabuľka ukladá informácie o serverových platbách, cenách, stave, alebo časoch, ktoré sa vzťahujú k objednávke. Z tejto tabuľke je podstatný atribút `orderContent`, ktorý obsahuje najmä informácie o objednávateľovi – meno a adresu.

6.2.6 Tabuľka server_invoice_pay_progs

Tabuľka obsahuje informácie o tarifoch, ktoré jednotlivé aplikácie majú. Podstatné hodnoty v atribúte `payPlanNeeded` sú free (tarif zdarma), plan2 (tarif Pro) a plan3 (tarif business).



Obr. 16 Výber z platobnej databázy, ktorý bude v práci použitý

6.2.7 Uloženie textov

Texty a stringy z aplikácie sú uložené väčšinou v tabuľke `effortix_nodes`. Problémom však je, že rôzne typy uzlov obsahujú veľké množstvo variant štruktúr, ako sú texty uložené a písať pre každý typ uzlu iný parser je zdĺhavé a zároveň zbytočné, vďaka jazykovým súborom. Po otestovaní/publikovaní aplikácie

sa na diskové pole systému ukladajú kompletné informácie o danej aplikácii, aby bola následne publikovaná, alebo poslaná na testovanie do testovacej aplikácie *TestDrive*. Tá obsahuje obrázky, štruktúry a jazykové súbory so všetkými textami a to vo formáte SQLite. Tie obsahujú tabuľku strings s veľmi jednoduchou štruktúrou (`CREATE TABLE strings (id TEXT, value TEXT)`).

Výhodou použitia tejto databázy sú všetky reťazce, ktoré sa v aplikácii nachádzajú sú pohromade v jednom súbore, nevýhodou je, že sa vytvárajú v momente publikovania, alebo testovania aplikácie, preto môžu byť do spracovania zaradené len tie aplikácie, ktoré obsahujú o sebe časový záznam.

7 Návrh

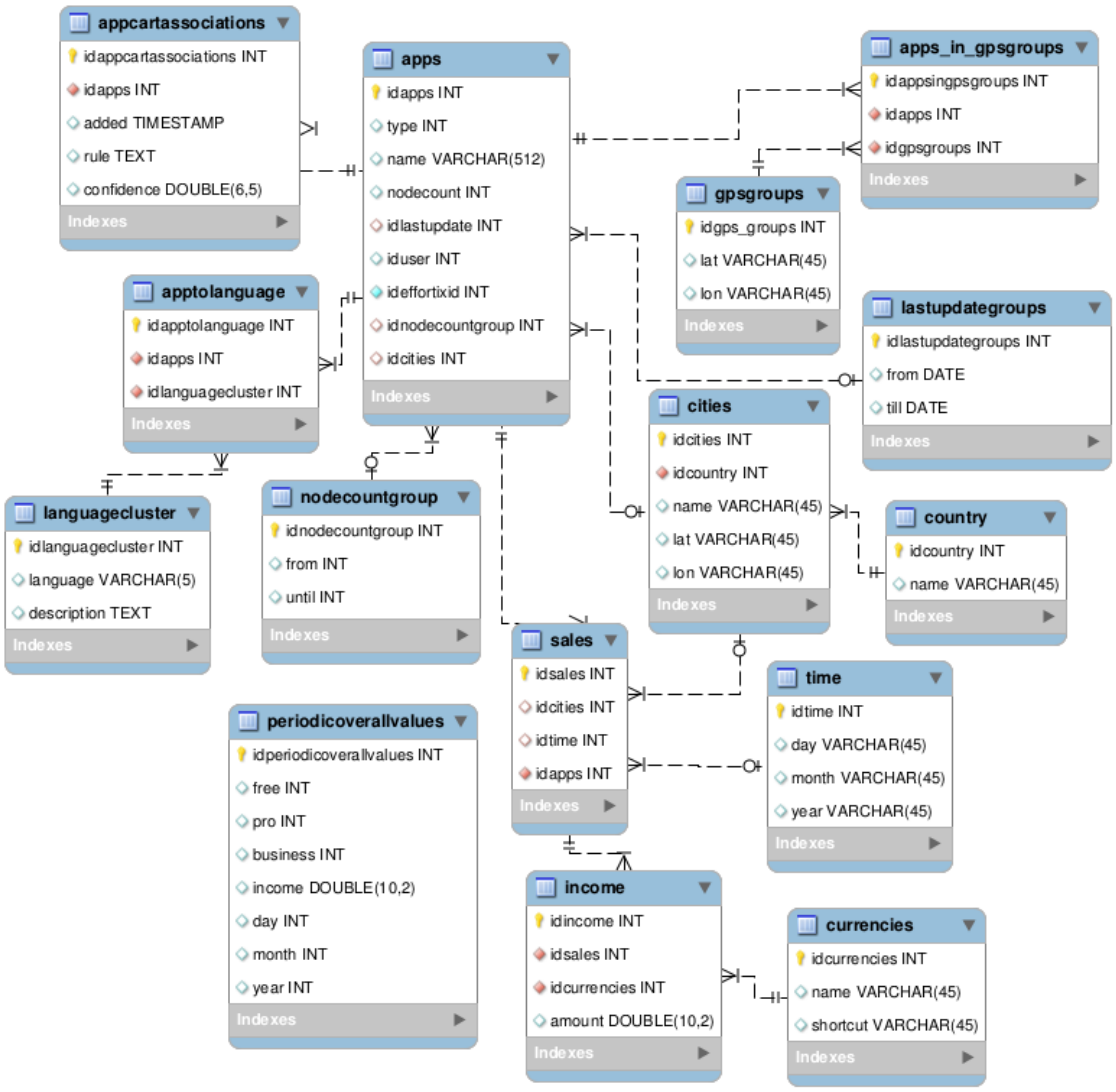
Systém funguje v dvoch rovinách. Prvá, hlavná časť, slúži len pre vedúceho pracovníka Effortixu.

V tejto časti sú kľúčovými dátami zoznamy aplikácii, ich názov, užívateľa, tarif aplikácie, atď. Z nich plynú základné štatistické funkcie. Ďalšou funkciou je zobrazenie priebehu periodických dát – príjem (denný, mesačný, ročný) a pomery tarifov aplikácii (denný). Poslednou časťou analýzy sú zhlukovacie analýzy.

Tá prebieha nad miestami objednávky (podľa vyplneného kontaktu), jazykovými súbormi (zhlukovanie textov do tématických okruhov podľa jazykov), vytváranie skupín aplikácii prebieha taktiež nad poslednou úpravou, kedy bola vykonaná v aplikácií, alebo počet uzlov, ktoré aplikácia v sebe obsahuje. Toto súvisí s automatickými e-mailovými kampaňami, kedy je možné vyselektovať aplikácie, ktoré patria do určitej skupiny a zamerať sa s ďalšími krokmi na ne. Prítomné je teda multidimenzionálne zhlukovanie na vytvorenie skupín nad skupinami.

Druhá rovina súvisí iba so zákazníkmi, ktorých aplikácia je označená tarifom Business. Jedná sa o prehľad jeho objednávok, základné štatistiky, vizualizácia periodických dát – príjmu podľa výberu dní, mesiacov, alebo rokov, a v neposlednom rade analýza položiek, ktoré koncoví uživatelia objednávali. Taktiež k základným štatistikám príjmu a objednávok patrí geografické zobrazenie. Ktoré položky objednávali spolu a ako nákupy spolu súviseli. Toto všetko s výborom meny, ktorú má daná aplikácia aj v systéme povolené.

7.1 Navrhnutý dátový model



Obr. 17 Štruktúra výslednej databázy.

7.2 Tabuľka apps

Centrálna tabuľka systému. Jeden záznam obsahuje informácie o jednej aplikácii – názov, užívateľa, napojenia na všetky zhluky, id aplikácie v hlavnom Effortix systéme, tarif a napojenie na mesto, odkiaľ bol tarif objednaný.

7.3 Tabuľka periodicoverallvalues

Do tabuľky sa pridáva každý deň nový záznam o aktuálnom stavu systému – príjem z objednávok za ten deň a aktuálny počet tarifov podľa počtu aplikácii.

Platba v Effortixe je ukladaná iba v amerických dolároch, preto je možné uložiť len jednu sumu pre jednu menu.

7.4 Tabuľka cities

Centrálna tabuľka miest. Obsahuje záznam o názvu mesta, odkaz to tabuľky krajín a GPS súradnice mesta. Tie sa každý deň kontrolujú a ak neexistujú pre dané mesto zatiaľ žiadne súradnice, systém si ich pomocou Google Maps API vyžiada a uloží. Záznamy sa ukladajú ako z objednávok aplikácii, tak aj z objednávok košíkov jednotlivých aplikácii.

7.5 Tabuľka country

Obsahuje informácie o krajinách, konkrétne ich názov.

7.6 Tabuľka gpsgroups

Pre aplikácie, ktoré majú zaplatený tarif vyšší, ako je Free, sa z miest, ktoré boli vyplnené v objednávkovom fomulári, a majú priradené GPS súradnice, vytvárajú zhluky, aby bolo možné zistiť oblasti, kde sa koncentrujú zákazníci. Zhluky obsahujú len informácie o ich centre.

7.7 Tabuľka apps_in_gpsgroups

Spojovacia tabuľka, ktorá priraďuje aplikácie k jednotlivým zhlukom.

7.8 Tabuľka languagecluster

K aplikáciam sa vytvárajú zhluky z textov, ktoré dokážu určiť určitú príslušnosť k nejakej téme. Tabuľka obsahuje skratku jazyka a jeho popis z Weka algoritmu (k-means). Jazykové zhluky sa vytvárajú pre každý jazyk zvlášť.

7.9 Tabuľka apptolanguage

Prepojenie jazykového zhluku a aplikácie.

7.10 Tabuľka lastupdategroups

Obsahuje informácie o skupinách aplikácii, konkrétne o dátume, kedy bola aplikácia naposledy upravená. Vďaka tomu je možné vidieť časové periódy úprav a kedy aplikácie upravovali najčastejšie. Tabuľka obsahuje časovú periódu OD a DO dátumu.

7.11 Tabuľka nodecountgroup

Rovnakým spôsobom je možné vidieť aj skupiny aplikácii podľa ich veľkosti, konkrétne počtu uzlov; opäť v tabuľke informácia veľkosti OD a DO počtu uzlov.

7.12 Tabuľka sales

Centrálna tabuľka analýz predajov pre koncových zákazníkov. Každý predaj v rámci aplikácie obsahuje odkaz do tabuliek (dimenzii) mesta a krajiny, času (deň, mesiac, rok) a aplikácie, v ktorej bola objednávka vykonaná.

7.13 Tabuľka currencies

V tabuľke sú uložené informácie o menách, ktoré boli v minulosti niekedy cez nejakú aplikáciu vybraté a to konkrétne názov a skratka. Aplikácie majú na výber viac mien, nielen dolár, ako hlavný systém.

7.14 Tabuľka income

Tabuľka na príjem pre danú objednávku. Obsahuje kľúč do tabuľky sales, tabuľky currencies a sumu. Údaj o sume je normalizovaný pre potrebu ukladať sumy pre všetky meny zvlášť.

7.15 Tabuľka time

Tabuľka obsahuje časový údaj, konkrétne deň, mesiac a rok.

7.16 Tabuľka appcartassociations

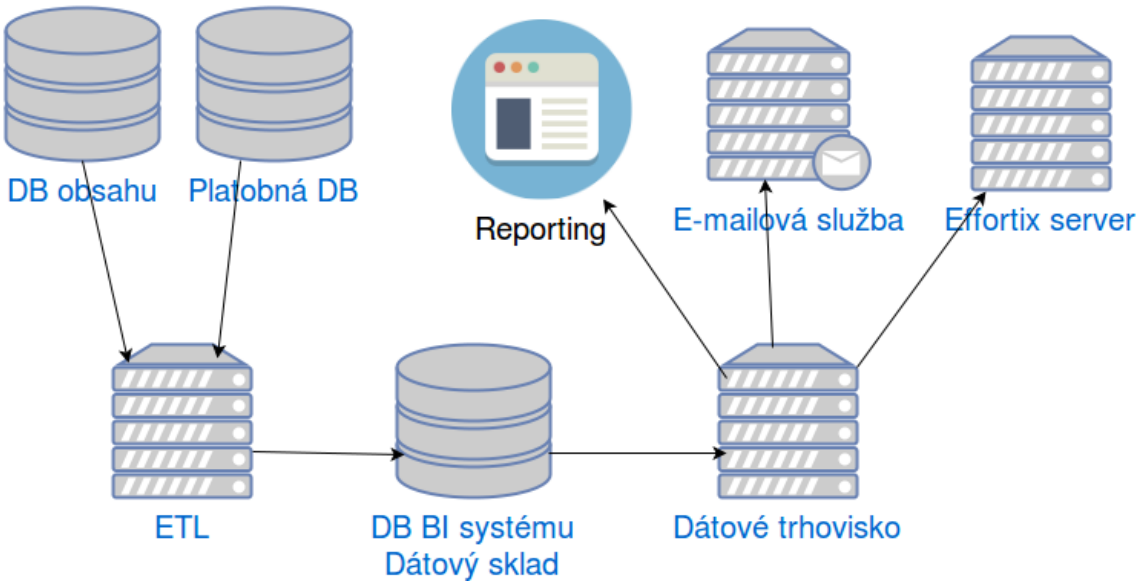
Tabuľka ukladá nájdené asociačné pravidlá pre konkrétnu aplikáciu. Obsahuje informáciu o čase pridania, významné pravidlo (názvy položiek vo všetkých jazykoch) a podporou, ktoré má dané pravidlo.

7.17 Návrh podsystémov

Technicky je celý BI systém riešený na samostatné podsystémy, ich Implementá bude popísaná v ďalšej kapitole.

- **ETL systém (EffortixETL)** – jedná sa o zberný podsystém. Program má za úlohu z oboch databází hlavného systému každý deň vybrať aktuálne dáta a vytvoriť informácie o aplikáciách, pridať nové záznamy do periodických dát, spracovať všetky skupiny, povytvárať štatistiky pre užívateľov a ich objednávkach a následne poslať uložiť do databázy dátového skladu.
- **Dátovy sklad** – databáza, ktorá v sebe obsahuje štruktúru uvedenú v predchádzajúcej kapitole

- **dátové trhovisko** – má priamy prístup k dátovému skladu a na základe modulov a ich parametrov[11] spracovávať a poskytovať všetky potrebné informácie
- **reporting** – na ukážku bude vytvorená jedna aplikácia, ktorá bude priamo komunikovať s užívateľom cez GUI rozhranie a sprostredkovávať komunikáciu s dátovým trhoviskom. Využitie je však širšie a systém je pripravený na komunikáciu s inými systémami, ako e-mailová služba na automatické rozposielanie, poprípade integrácia do hlavného Effortix systému.



Obr. 18 Schématická nákres systému a jeho podsystémov

[11] Moduly budú popísané v kapitole o implementácii a zdokumentované v dokumentácii

8 Implementácia

8.1 ETL

EffortixETL subsystém vytvorený za účelom načítania dát z hlavného systému Effortix, spracovať ich – vytvoriť aktuálne štatistiky, periodické dáta, jazykové, GPS skupiny a skupiny počtu uzlov a dátumu poslednej úpravy, Následne ich uložiť do navrhnutej databázovej štruktúry. Jednotlivé procesy a ich poradie sú zakreslené na obr. 19.
Aplikácia je rozdelená do následovných balíkov:

- **com.effortix.bi.etl** – jadro systému, obsahuje spúšťaciu funkciu, pripojenie na MySQL, konfiguráciu a funkcie,
- **com.effortix.bi.etl.containers** – singleton triedy na prenos štruktúr medzi funkciami,
- **com.effortix.bi.etl.contentprocessors** – funkcie na štatistiku, komunikáciu s Google Maps API a geografické funkcie,
- **com.effortix.bi.etl.datamining** – Data Mining algoritmy – Asociačné pravidlá, DBSCAN, k-means,
- **com.effortix.bi.etl.datamining.text** – súvisí s data miningom; obsahuje jazykové súbory pre text mining,
- **com.effortix.bi.etl.models** – modely, ktoré dodávajú dáta z externých zdrojov,
- **com.effortix.bi.etl.threads** – triedy na prácu s vláknami.

EffortixETL je napísaný v Jave a pravidelne sa spúšťa každý deň nad aktuálnymi dátami.

8.1.1 Načítanie aplikácii

V prvej časti sa po inicializácii z modelu vytiahne zoznam všetkých aplikácii. Najskôr z databázy obsahu, kde sú úplne všetky aplikácie, ktoré boli vytvorené. Druhý zdroj je načítanie z platobnej databázy, kde je možné zistiť, ktoré aplikácie boli testované, alebo publikované z toho dôvodu, že ako bolo vyššie spomenuté, nepublikované aplikácie sa pre neprítomnosť obsahových súborov na jazykové testovanie nespracovávajú.

8.1.2 Prvá iterácia

Jedna z troch iterácii, ktoré sú v procese spracovania dát prevádzané. V tomto kroku sa porovnávajú zdroje aplikácií z oboch databáz, z platobnej databázy sa vyberá tarif každej aplikácie, a ich celkové spočítavanie, prevádzajú sa prípravy na skupinové roztriedenie podľa počtu uzlov a poslednej aktualizácie a nakoniec sa vyberajú užívateľské objednávky pre jednotlivé aplikácie. To vyberá dáta pre

asociačné spracovanie a taktiež pre celkové štatistiky predaja pre koncového klienta.

8.1.3 Odstránenie nepoužitých appiek

Jedná sa len o vyčistenie appiek, ktoré neboli ešte testované, alebo publikované, za účelom ich ďalšieho nesprácovávanie.

8.1.4 Skupiny – počty uzlov a posledná úprava

V predchádzajúcom kroku sa v prvej iterácii našli najmenšie a najväčšie hodnoty počtu uzlov a dátumu poslednej úpravy. Pomocou intervalového a skupinového triedenia sú vyrátané pomocou týchto hodnôt skupiny a uložené do databázy. Keďže skupiny poslednej úpravy sú rátané podľa unix timestamp a skupín sú z týchto čísel vytvorené tisíce, pre celkovú prehľadnosť sú hodnoty delené číslom 1000000.

```
public static ArrayList<Long> getIntervalBorders(Long min,
Long max){
        ArrayList<Long> r=new ArrayList<Long>();
        Long number=Math.round(Math.sqrt(max-min));
        Integer interval=Math.round((max-min)/number);
        Long from=min;
        while(from<max){
            from=from+interval;
            r.add(from);
        }
        return r;
    }
```

Funkcia vráti z maximálnej a minimálnej hodnoty zoznam hraníc intervalov.

8.1.5 Druhá iterácia

Druhá iterácia do skupín vytvorených v predchádzajúcom kroku jednotlivé aplikácie priraďuje. Pre každú aplikáciu vyberie počet uzlov a dátum poslednej úpravy, nájde skupinu, ktorej rozmedzie im zodpovedá a priradí.
V druhej časti sa načítajú z platobnej databázy posledné objednávky a zistí sa miesto objednávky pre všetky aplikácie, ktoré vôbec objednávku mali. Hľadanie mesta prebieha zistením, že toto mesto aj s krajinou už je uložené v databáze v tabuľke cities. Ak existuje, načíta sa jej ID, ak nie, tak sa vytvorí jeho záznam a vráti sa jeho ID, ktoré je potom priradené k aplikácii v pamäti.

8.1.6 Vlákna data miningu

V rámci data miningového spracovania sú spustené 3 typy vlákien – na vytvorenia jazykových klusterov, asociačných pravidiel a GPS súradníc, ktoré systém

nechá bežať až kým sa nevytvoria všetky potrebné údaje, počká dovtedy, kým sa nespracujú a pokračuje až potom.

```
AssociateCartThread[]workers1 = new AssociateCartThread[5];
for (int j = 0; j < workers1.length; j++) {
     workers1[j]=new AssociateCartThread();
}
for(AssociationRuleData data:asociationRuleInput){
     if(data.inputData.size()>0){
          for (int j = 0; j < workers1.length; j++) {
               if(!workers1[j].isAlive()){
                    workers1[j].setParams(data);
                    workers1[j].start();

                    break;
               }
          }
     }
}
```

Príklad spustenia vlákien ku asociačným pravidlám.

Na začiatku inicializácia 5 vlákien, každému vláknu sú poslané dáta na spracovanie a vlákna sú spustené.Podobným spôsobom sú spustené aj zvyšné algoritmy. Na konci prebieha kontrola skončenia spracovania všetkých vlákien pomocou `workers.join()`.

8.1.7 Vytvorenie jazykových klusterov

Prebieha pomocou algoritmov z programu Weka, konkrétne triedou `SimpleKMeans()`. Z pohľadu vlákien sú spracovávané pre príslušné jazyky zvlášť. Prvá časť je vytvorenie vektorov dokumentov – odstránenie stop slov z textov a jeho vloženie do Weka implementácie triedy `FastVector`. Na internete existuje veľké množstvo balíčkov so stop slovami, ktoré sa dajú stiahnúť. Slová pre konkrétny jazyk sú uložené v zvlášnych triedach a pomocou návrhového vzoru Factory sa len príslušná trieda zavolá[12]. Ďalšou časťou je do algoritmu aplikovať filter `StringToWordVector` na transformáciu slov do vektoru a na neho aplikovať ďalší filter `getIDFTransform`. Po spracovaní algoritmu sa vyberie záverečný report, ktorý sa štandardne vypisuje do Weky a sparsuje sa pomocou regulárnych výrazov na výber slov pre klustery. Tie sú následne uložené do DB a sú k nim priradené aplikácie pre každý jazyk zvlášť.

```
public class StopWordsFactory {
    public static DefaultStopWords get(String shapeType) {
```

[12] https://code.google.com/archive/p/stop-words/downloads

```
        if (shapeType == null) {
            return new EN();
        }
        if(shapeType.equalsIgnoreCase("sk")) {
            return new SK();
        }
        …
        else if (shapeType.equalsIgnoreCase("ru")) {
            return new RU();
        }
        return new EN();
    }
}
```

Factory trieda na stop slová

8.1.8 Asociačné pravidlá

Dáta sú predpripravené vo formáte názvu položky vo všetkých jazykoch a do spracovávania sú posielané tiež ako vektor s ID položky binárnom zobrazení výskytu `0/1`. Po spracovaní triedou `Apriori()` sa opäť výsledky naparsujú. Vlákna spracovávajú dáta pre každú aplikáciu zvlášť.

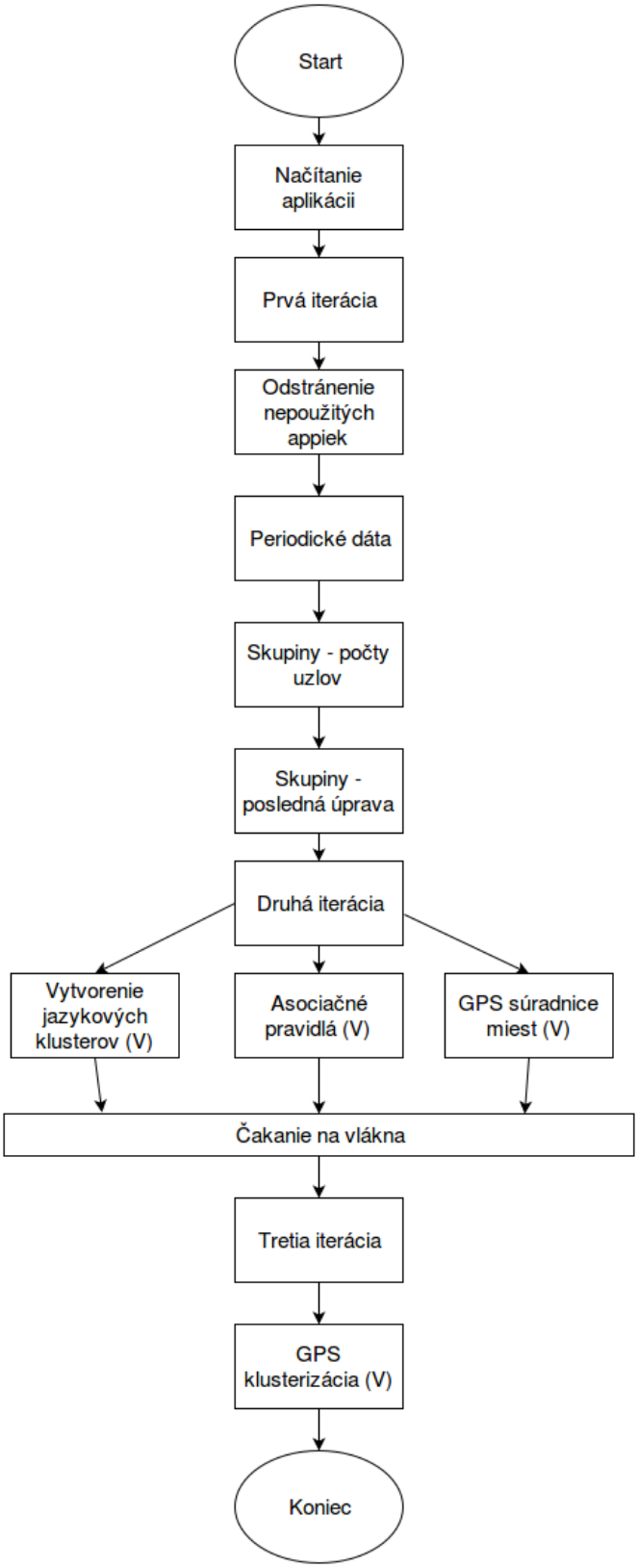
8.1.9 GPS súradnice miest

Databáza miest, ktorá bola vytvorená pre všetky objednávky aplikácii aj v aplikácii z predchádzajúcich krokov obsahujú len názvy, avšak bez GPS súradníc. Pomocou Google Maps API je však možné po odoslaní parametrov, ako je názov mesta a krajiny, získať zemepisnú šírku a výšku mesta. Po získaní API kľúču na stránke je možné posielať požiadavky a Google vráti potrebné údaje v JSON formáte[13]. Po vyparsovaní sa ukladajú súradnice ku mestu do DB. Google Maps API však obsahuje vo verzii zadarmo určité obmedzenia počtu požiadaviek, preto sa algoritmus vždy preruší po 10 požiadavkách na 1.5 sekundy. Pre potreby tohto systému je to však dostatočné.

8.1.10 Tretia iterácia

Po spracovaní vlákien v 3. iterácii prebieha priradenie získaných GPS súradníc ku aplikáciám. Vďaka nemu bude možné vytvoriť geografické skupiny, odkiaľ boli Effortix aplikácie objednávané.

[13] https://developers.google.com/maps/documentation/geocoding/intro#geocoding



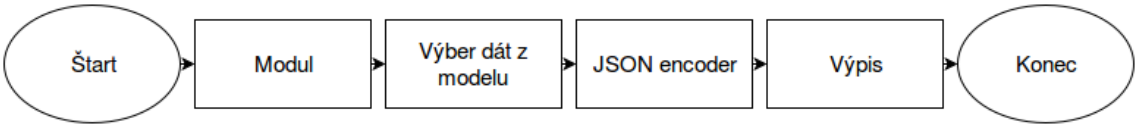
Obr. 19 Procesy v ETL podsystéme

8.1.11 GPS klusterizácia

Ako posledné spracovanie je spustené viacvláknové vytváranie GPS zhlukov. To prebieha pomocou DBSCAN algoritmu vo Weke, ktoré dostane ako parameter zoznam GPS súradníc, vložia sa do štruktúry FastVector a po vytvorení triedou `DBSCAN()` sa výsledky vyparsujú do formátu klusterov a príslušných aplikácii a vypočítaných stredov klusterov.

8.2 Dátové trhovisko

Ďalší podsystém celého systému je dátové trhovisko, ktoré má na starosti na základe volaných modulov a ich vstupných parametrov vrátiť údaje, ktoré boli predtým spracované cez ETL subsystém. Napísaný je taktiež v Jave, konkétne sa jedná o Jetty HTTP server napísaný v Jave. Na obr. 20 je zobrazený priebeh žiadosti o dáta.



Obr. 20 Sekvencia práce s marketom

Subystém funguje modulovo a podľa výberu modulu a parametrov vyberá z BI databázy potrebné dáta. Zoznam modulov (kontextov) je následovný:

- **`applist`** – poskytuje celkové prehľady jednej, alebo viacerých aplikácii, podľa unikátnych kľúčov, tarifu, užívateľa, mesta, krajiny a podobne. Návratové hodnoty obsahujú vždy základné informácie, zhluky a ich základné informácie.
- **`gpsgroups`** – vráti buď jeden, alebo všetky skupiny GPS zhlukovania. Návratový formát obsahuje identifikácie, súradnice stredu a aplikácie, ktoré k zhluku patria.
- **`updategroups`** – vráti buď jeden, alebo všetky skupiny zhlukovania poslednej úpravy. V návratovej štruktúre sú limity skupiny, identifikačné údaje a prislúchajúce aplikácie.
- **`nodecountgroups`** – podobné fungovanie ako updategroups s tým rozdielom, že modul pracuje s počtami uzlov v aplikácii
- **`languagegroups`** – jazykové zhluky sú vrátené aj s jazykom, popisom slov a aplikáciami, ktoré do zhlukov patria.
- **`clusterization`** – modul prevádza medzizhlukovú klusterizáciu medzi počtami uzlov, poslednou úpravou, jazykovými a GPS zhlukami. Prakticky umožňuje nájsť skupiny aplikácii s podobnými vlastnosťami a to pomocou DBSCAN algoritmu, ktorý podporuje viacdimenzionálne zhlukovanie.
- **`statistics`** – modul vracia základné štatistiky celého systému: celkové počty a počty jednotlivých tarifov, priemery a pod.
- **`periodic`** – podľa vstupného parametru časového obdobia od – do a výberu denných, mesačných, alebo ročných štatistík vráti pre existujúce časové údaje štatistiky o tarifoch a príjmu.
- **`userstatistics`** – vráti základné štatistiky pre koncového užívateľa – pre užívateľa, alebo pre appku; počet objednávok, priemer objednávok na appku a vo všetkých uložených menách štatistiky o príjmu z objednávok.

- **associations** – pre užívateľa, alebo appku vráti význačné asociačné pravidla.
- **appperiodic** – vracia periodické údaje pre appku, alebo appky užívateľa. Vstupnými parametrami je časový limit od-do, typ časového údaju (dni, mesiace, roky), id mesta, id krajiny, poprípade jednoduché severné, južne, západné a východné geografické ohraničenie .

Konkrétne a kompletné parametre modulov a návratové formáty sú uvedené v dokumentácii.
Systém má nasledujúcu štruktúru balíkov:

- **com.effortix.bi.market** – základné nastavenia, funkcie, konfigurácie, inicializácie.
- **com.effortix.bi.market.containers** – návrhový vzor container triedy na prenos dátových štruktúr medzi funkciami
- **com.effortix.bi.market.contexts** – moduly vo forme kontextu
- **com.effortix.bi.market.datamining** – algoritmy na data mining spracovanie
- **com.effortix.bi.market.JSONBuilders** – triedy, ktoré dodané dáta spracovájú do JSON štruktúr na výstup
- **com.effortix.bi.market.models** – triedy na výber dát z databáz a ich predanie na zobrazenie

Výber modulu prebieha Jetty inicializáciu a do `ContextHandlerCollection` sa vkladajú kontexty pre z URL adresy a server je spúšťaný pod nastaveným portom, v aplikácii konkrétne je to port 8095.

URL má tvar `http://localhost:8095/MODUL_NAME/` a serverový kontext pre jeden z modulov MODUL_NAME napr. `nodecountgroups` vypadá následovne:

```
ContextHandler nodecountgroupContext = con-
texts.addContext("/nodecountgroups", "/");
SessionHandler nodecountgroupSessions = new SessionHan-
dler(manager);
nodecountgroupContext.setHandler(nodecountgroupSessions);
nodecountgroupSessions.setHandler(new NodeCountGroupCon-
text());
```

Každý kontext zistuje prítomnosť GET parametrov, ktoré boli modulu poslané pomocou `request.getParameterMap().containsKey("paramname")` a tým spracuje vstupné parametre modelom, ich výstupy spracuje pomocou JSONBuilderu a ten zobrazí ako návratovú štruktúru dát pre aplikácie, ktoré si dáta vyžiadajú od trhoviska. Moduly pomocou MySQL konektoru vyberajú z DB dáta cez SQL konštruované cez `PreparedStatement`. JSON enkódery (JSONBuildery) v sebe obsahujú ako container štruktúru, tak aj funkcie, ktoré tieto štruktúry parsujú do `JSONArray` a `JSONObject`, ktorý sa nakoniec spra-

cuje do výstupu zo serveru. Výsledné `JSONArray` a `JSONObject` štruktúry sa prevedú na reťazec JSON, obalia HTTP hlavičkami a pošlú sa ako výsledok.

Daný subsystém môže poskytovať spracované dáta pre rôzne iné procesy, ktorých použitie vyžaduje BI dáta.

8.3 Reporting subsystém

Slúži ako systém na finálnu vizualizáciu štatistík, výsledkov a reportov z BI. Jedná sa o príklad na použitie trhoviska – na princípy jeho fungovania, zadávania vstupov a podobne, takže systém nie je ošetrený z bezpečnostného hľadiska alebo vyladeného vzhľadu. Reporting funguje v dvoch režimoch.

Prvý režim simuluje prístup správcu Effortixu k dátam týkajúceho sa celého systému – prehľady aplikácii, zhlukov, periodických dát a štatistík. Druhý režim predstavuje prihláseného užívateľa v prehľade svojich aplikácii a zobrazenie štatistík príjmu, asociácii medzi kupovanými produktami, alebo prehľad periodických dát.

Systém je vytvorený pomocou CodeIgniter PHP frameworku. Ako bolo spomenuté v popise PHP frameworkov v kapitole 5.15, jedná sa o veľmi jednoduchý nástroj a samotným vytvorením reporting subsystému sa potvrdilo, že prostredie má strmú krivku učenia a vytvoriť subsystém bola otázka jednotiek dní aj s naštudovaním jeho fungovania. K základnému CodeIgniter kódu bol tiež pridané následovné externé nástroje:

- Šablónovací systém Smarty,
- knižnica jQuery,
- knižnica jQuery UI,
- knížnica na vykreslovanie grafov RGraph,
- knižnica Google Maps,
- knižnica MarkerClusterer na grafické zhlukovanie bodov v prostredí Google Maps.

Taktiež bola použitá na ukážku šablóna založená na knižnici bootstrap.
Samotný framework funguje na MVC modeli a jej jednotlivé časti sa nachádzajú v zložkách models, controllers a views.

8.3.1 Controllers

Ich úlohou je po ich zavolaní HTTP routingom[14] spracovať predané parametre, získať dáta z modelu a tie nasádzať do view. Systém funguje na 3 controlleroch:

- **`Adminapps`** – poskytuje rozhranie na získanie administračných údajoch.
- **`Groups`** – spracováva požiadavky na zobrazenie zhlukov zo systému.
- **`Userapps`** – poskytuje ukážku práce so štatistikou prístupnou pre koncového klienta.

[14] Umožňuje spracovávať URL a volať príslušné controllery s predaním parametrov.

```
public function nodecount($idgroup =null){
        //url na správne zobrazenie ciest
        $data['assets']=asset_url();
        $data['base']=base_url();

        $data['points'] = $this->GroupsModel-
>getNodecounty($idgroup); //získanie zhlukov z modelu
        if($idgroup ==
null)$data['content']="groupsnodeupdate"; //zobrazenie
všetkých zhlukov - použiť šablónu na ich zobrazenie
        elseif($idgroup != null){ //zobrazenie 1 zhluku
            $data['apps']=$data['points']['apps'];//1 zhluk
obsahuje informácie aj o aplikáciach, ktoré do neho patria
            $data['content']="groupnodeupdate"; //šablóna
na vykreslenie 1 zhluku
        }
        $data['menu']="adminmenu"; //šablóna menu
        $this->smarty->view('main.tpl',$data);
//vykreslenie vrátených dát do  view
    }
```

Daná ukážka zdrojového kódu spracováva zobrazenie skupín zhlukovania

8.3.2 Model

Vrstva na získavanie dát nepotrebuje na prácu pripojenie na databázu. Parametre, ktoré dostane z controlleru spracuje do URL adresy na dátové trhovisko spolu s názvom modulu a jeho parametrami. Každý controller má svoj model, konkrétne sú to modely `AdminappsModel`, `GroupsModel` a `UserappsModel`.

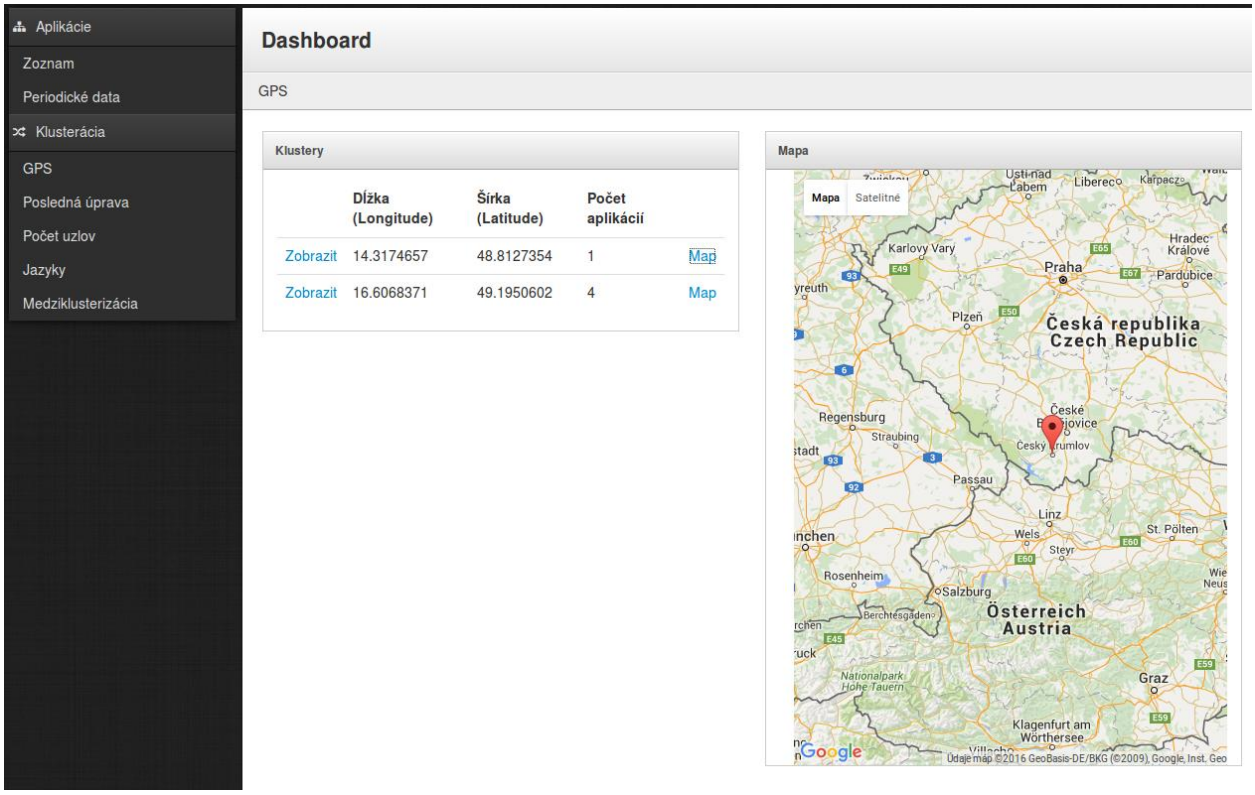
```
function getPeriod($period=null, $from=null, $to=null){
    //parametre
    $getparams=array();
    if(!empty($period))$getparams[]="period=".$period;
    if(!empty($from))$getparams[]="from=".$from;
    if(!empty($to))$getparams[]="to=".$to;
    return json_decode(file_get_contents($this->config-
>item('biurl')."periodic".((sizeof($getparams)>0) ?
"?".implode("&", $getparams) : "")), true); //získanie dát
z modulu periodic dátového trhoviska a ich dekódovanie do
asociatívneho poľa
    }
```

Ukážka modelu na získanie periodických dát hlavnej administrácie.

8.3.3 View

Dáta, ktoré boli spracované z modelu cez controller sa cez premenné zobrazia do view, konkrétne majú podobu danú Smarty šablónou. Hlavná šablóna v sebe zahŕňa HTML štruktúru na dosadenie dát a podšablón.

Čo sa týka tých, boli vytvorené šablóny na menu (v ľavej časti šablóny sa zobrazuje menu pre dáta celého systému, alebo koncového klienta), pravý horný roh (prepínače na výber aplikácie, jazyka, meny a ich kombinácie), a pre každé zobrazenie controllerov. Do nich sú podľa potreby vkladané javascriptové knižnice *RGraph* (periodické zobrazenie), *Google Maps* (zobrazenie miest nákupov a ich zhlukov), alebo *jQuery UI* (kalendár).



Obr. 21 Ukážka reporting subsystému

8.4 Ukážky repporting susbsystému

Po spustení ukážkového reporting systému, ktorý funguje nad už spracovanými ukážkovými dátami je na úvodnej obrazovke možné vidieť základné štatistické údaje.

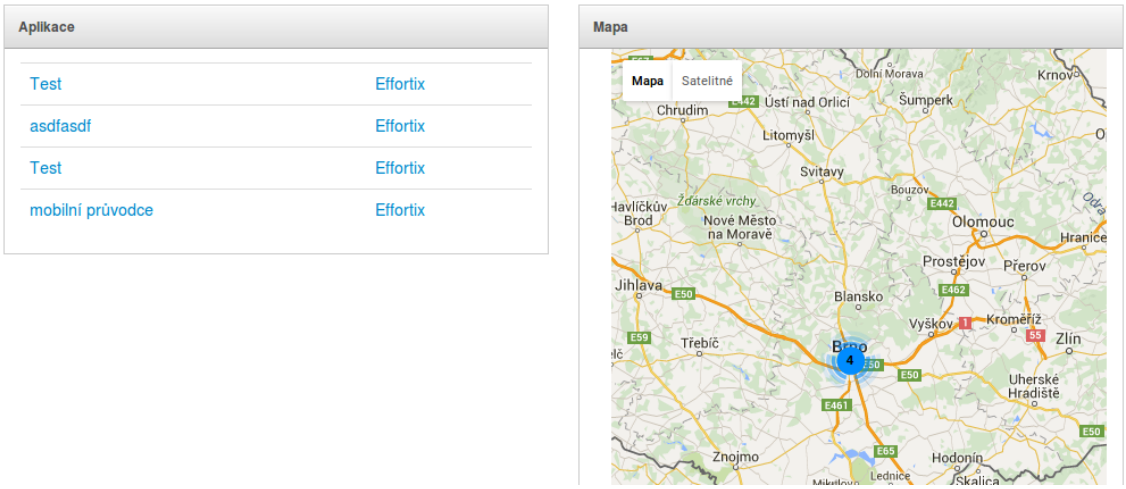
Zaujímavejšie údaje sa dajú vyčítať z klusterizácie. Jej prevedenie podľa počtu uzlov a poslednej úpravy má rovnakú vizuálnu stránku a je možné pozrieť všetky zhluky, počet aplikácii v nich a ich zobrazenie so zoznamom všetkých aplikácii. Zhlukovanie podľa taktiež poskytuje prehľadný zoznam spolu so

zobrazením slov so zoznamom jazykov s možnosťou filtrovania. Zaujímavé sú nájdene zhluky z hľadiska prehľadu aplikácii, kde našlo 4 zhluky:

1. Zhluk obsahoval len aplikáciu *Katalog Eve Style*, čo svedčí, že sa jedná o vytvorený produktový katalóg.
2. S aplikáciami ako *Tipatour*, *Stebnické bojiště*, *MS bikes 2016*, *PIZZA U HONZY*, *CentrumBrna.cz* – klasické prípady aplikácii pre miesta, ale aj s komercionalizáciou, ako cestovné kancelárie, podniky a pod.
3. aplikácie ako *Moje úžasná aplikace*, *NP Podyjí - TEST aplikace*, alebo *Test* môžu byť označené ako testovacie aplikácie, ktoré si možno vytvorili užívatelia len na skúšku.
4. *Český Krumlov*, *Průvodce Chebem* alebo *Ignis Brunensis 2015* ukazujú na turistických sprievodcov.

Rozdiely medzi klusterom 2 a 4 môže byť užitočné preskúmať vedúcim pracovníkom a menšie rozdiely a hodnoty indexov jednotlivých slov porovnať pre aplikáciu ďalších krokov.

GPS klusterizácia, konkrétne na obr.22 detail zhluku, obsahuje tiež mapu, kde je možné vidieť stred zhluku; podľa daných zhlukov je možné vidieť centrá nákupu aplikácie, kde všetky boli zakúpené klientmi z Brna.

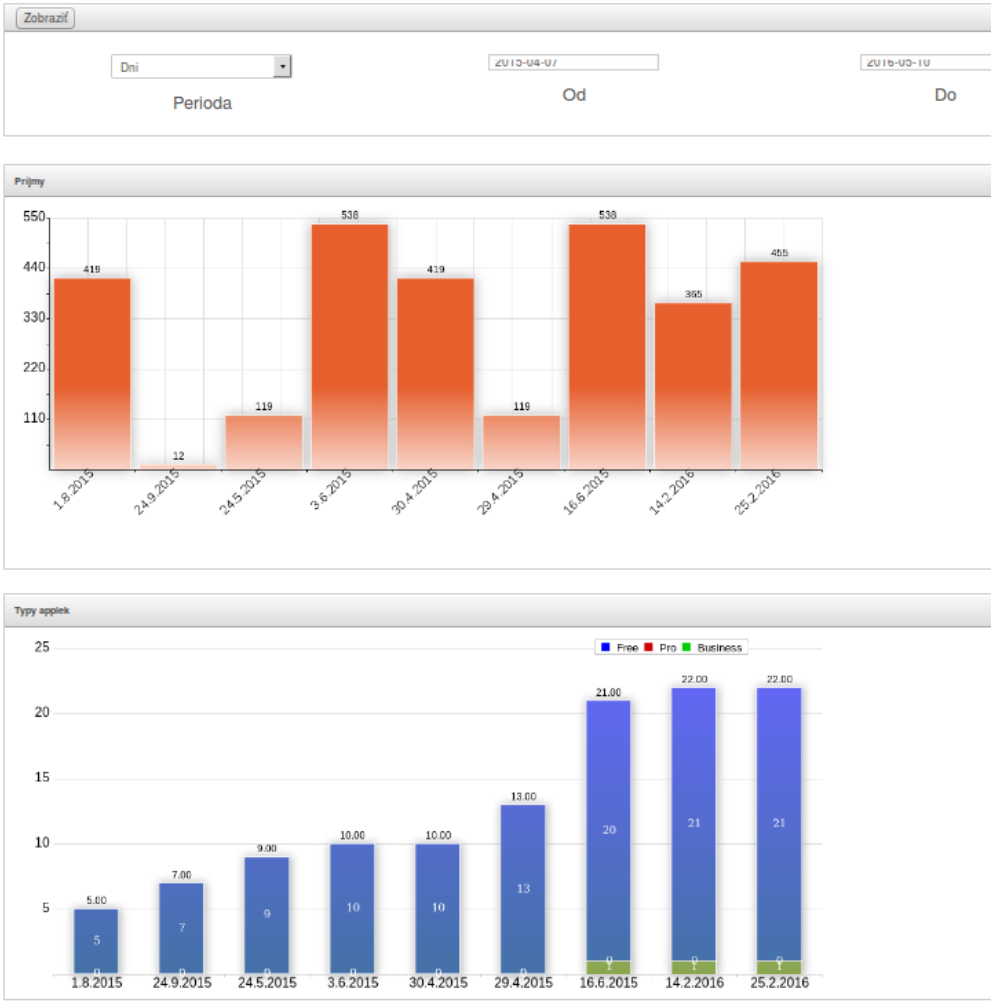


Obr. 22 Detail GPS klusteru

Medziklusterizácie po výbere počtu uzlov a poslednej aktualizácie zobrazuje zhluky aplikácii, ktoré napr. majú menší počet uzlov a boli aktualizované väčšinou v druhej polovici roka.

Zoznam aplikácii poskytuje prehľad všetkých aplikácii so štatistikami, informáciami o zhlukoch, v ktorých sú zaradené a taktiež je možné si prekliknúť aplikácie len od daného uživateľa, len z učitého miesta objednávky. Rozkliknutá aplikácia zobrazuje všetky prehľadné údaje, spolu aj s mapami miesta nákupu a GPS zhluku, do ktorého je zaradená pre porovnanie, ako sa tieto dve líšia.

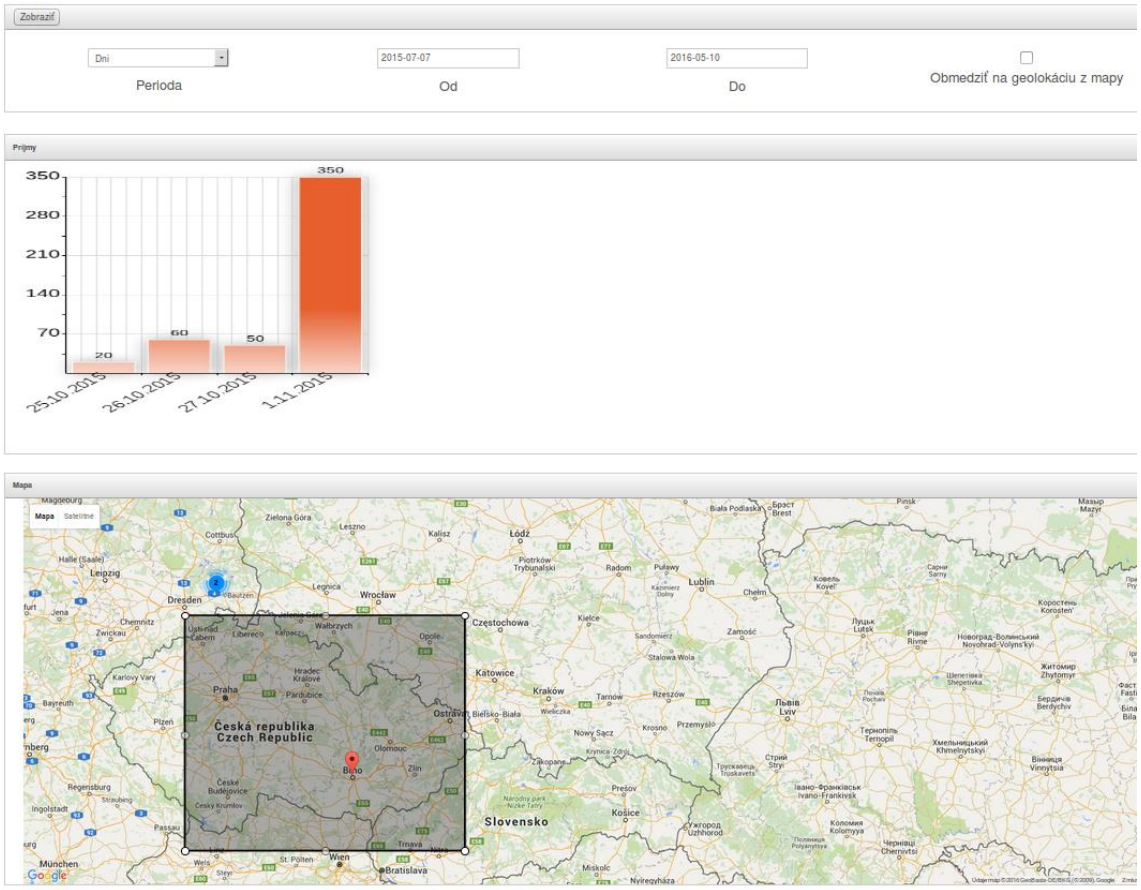
Na obrázku 23 je možné vidieť prehľadné periodické dáta vizualizované do dvoch grafov. Prvý je príjem z aplikácii podľa dní, druhý zobrazuje zloženie aplikácii, konkrétne počty jednotlivých tarifov a ich priebeh po dňoch. Vstupné parametre je možné upraviť a vizualizovať podľa dní, mesiacov a rokov.



Obr. 23 Periodické dáta administrátora

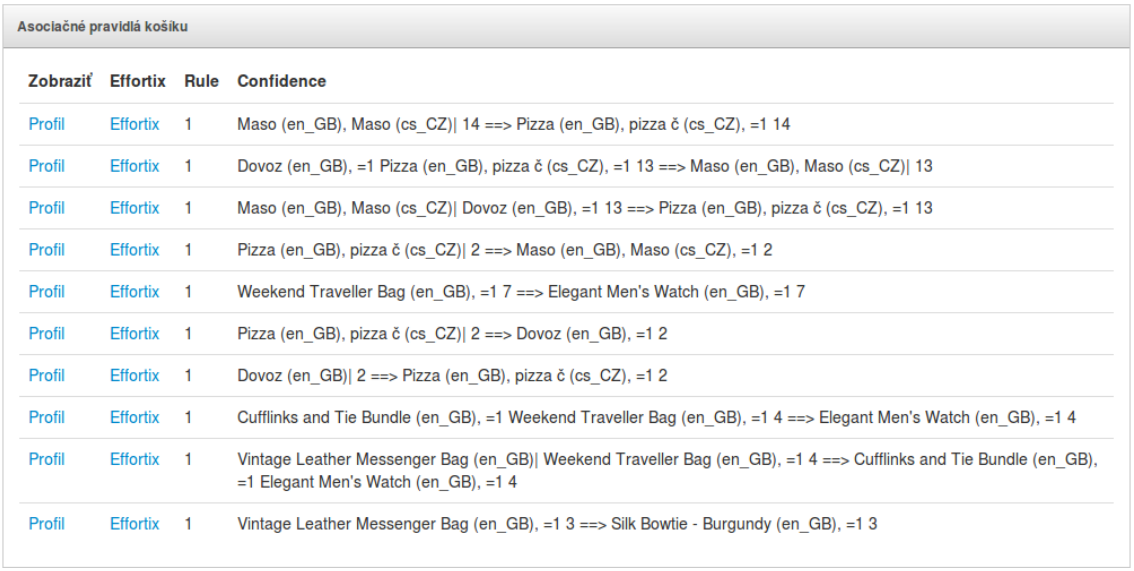
Koncový užívateľ ma v systéme možnost pozrieť informácie a spracované údaje zo svojích aplikácii. Prvá obrazovka zobrazuje štatistiky o objednávkach vo všetkých menách a podľa výberu aplikácie.

Z týchto dát je možné si prezrieť aj periodické dáta, konkrétne príjem podľa výberu daného obdobia s možnosťou výberu meny a aplikácie. Táto obrazovka tiež obsahuje užitočnú možnosť a to je špecifikovanie oblasti odkiaľ má systém filtrovať dáta. Po vybratí obdobia je možné zapnúť filtrovanie a na Google mape vybrať obdĺžnik z ktorej oblasti dáta vyberať.



Obr. 24 Periodické dáta koncového zákazníka

Asociácie košíku zobrazujú položky, ktoré boli spolu zakúpené aj s príslušnými hodnotami, pre lepšie plánovanie ďalších stratégií a plánov.



Asociačné pravidlá košíku

Zobraziť	Effortix	Rule	Confidence
Profil	Effortix	1	Maso (en_GB), Maso (cs_CZ)| 14 ==> Pizza (en_GB), pizza č (cs_CZ), =1 14
Profil	Effortix	1	Dovoz (en_GB), =1 Pizza (en_GB), pizza č (cs_CZ), =1 13 ==> Maso (en_GB), Maso (cs_CZ)| 13
Profil	Effortix	1	Maso (en_GB), Maso (cs_CZ)| Dovoz (en_GB), =1 13 ==> Pizza (en_GB), pizza č (cs_CZ), =1 13
Profil	Effortix	1	Pizza (en_GB), pizza č (cs_CZ)| 2 ==> Maso (en_GB), Maso (cs_CZ), =1 2
Profil	Effortix	1	Weekend Traveller Bag (en_GB), =1 7 ==> Elegant Men's Watch (en_GB), =1 7
Profil	Effortix	1	Pizza (en_GB), pizza č (cs_CZ)| 2 ==> Dovoz (en_GB), =1 2
Profil	Effortix	1	Dovoz (en_GB)| 2 ==> Pizza (en_GB), pizza č (cs_CZ), =1 2
Profil	Effortix	1	Cufflinks and Tie Bundle (en_GB), =1 Weekend Traveller Bag (en_GB), =1 4 ==> Elegant Men's Watch (en_GB), =1 4
Profil	Effortix	1	Vintage Leather Messenger Bag (en_GB)| Weekend Traveller Bag (en_GB), =1 4 ==> Cufflinks and Tie Bundle (en_GB), =1 Elegant Men's Watch (en_GB), =1 4
Profil	Effortix	1	Vintage Leather Messenger Bag (en_GB), =1 3 ==> Silk Bowtie - Burgundy (en_GB), =1 3

Obr. 25 Ukážka asociačních pravidel košíka

Zo všetkých aplikácií existuje hypertextový odkaz priamo na Effortix systém, kde je možné upravovať aplikácie a tiež mapy, ktoré v sebe obsahujú viac bodov, majú zapnuté vizuálne zhukovanie pre lepší prehľad.

9 Diskusia

Na základe požiadaviek a dostupných dát bol vypracovaný systém, zložený z niekoľkých podsystémov za účelom zobrazenia už existujúcich dát, ale s iným pohľadom. Vďaka nemu bude možné nahliadať na aplikácie s nadhľadom vytvorenými skupinami aplikácii, periodickými dátami a BI klientov.

Systém bol testovaný ako na testovacích dátach, tak aj dátach ostrých a výsledný ukážkový systém obsahuje dáta z oboch zdrojov. Ostrý systém totiž nie vždy obsahuje dostatečné množstvo dát (napr. objednávok koncových užívateľov pre jednotlivé aplikácie), ktoré by boli použiteľné na koncové algoritmy. Na testovanie bude vedúcemu a oponentovi práce poskytnutá URL adresa s možnosťou prehliadania výsledkov nad testovacími dátami.

V najbližších týždňoch bude celý systém použitý do ostrého behu hlavného serveru a následne budú ďalej analyzované požiadavky s následnými realizovanými úpravami.

9.1 Ekonomické zhodnotenie práce

Ekonomická stránka projektu sa dá zhrnúť do výdavkov a plánovaných ušetrených nákladoch. Na strane výdavkov sa jedná o prácu na návrh a vytvorenie projektu. Jeho vyvorenie zabralo mne, ako programátorovi cca. 3 mesiace. Podľa portálu platy.sk[15] sa hrubá mesačná mzda Business Intelligence Specialist pohybuje na úrovni 1705 €, čo je podľa aktuálneho kurzu 46077 Kč. Vo výsledku sa jedná o náklad 138231 Kč. V rámci svojej práce som však systém vytvoril v rámci svojho voľného času, takže sa jedná o celkom významnú úsporu.

Z pohľadu druhej strany, ušetrených prostriedkov, sa nejaký odhad nedá kalkulovať, lebo sa jedná o vylepšené prezeranie dát. Veľkým plusom však môže byť uvedenie BI metód ako jednu z pokročilých funkcii plateného tarifu, čo môže prilákať ďalších zákazníkov.

9.2 Možné ďalšie vylepšenia

Jednou z možností ďalšieho rozšírenia bude ukladanie historických dát a ďalšia práca s nimi. Primárne tieto dáta nie sú potrebné a účel vytvorenia BI systému tieto informácie nevyžaduje. Zaujímavým vylepšením by tiež mohlo byť upraviť spracovanie jazykovou klusterizáciou na lepšie filtrovanie slov. Do zdrojových textových súborov sa vkladajú kompletné jazykové súbory, takže nielen obsahy stránok a uzlov, ale aj pomocné texty aplikácie; je to však otázka priority, lebo napr. prítomnosť dní v týždni môžu indikovať ich špeciálne použitie v aplikácii (otváracie hodiny môže symbolizovať predajňu, alebo aplikáciu turistického sprievodcu). Pri jazykovom spracovaní tiež môžem spomenúť možné vylepšenie

[15] Cenová hladina na Slovensku je podobná ako v Českej republike a pre hrubý odhad je zdroj dát dostačujúci.

vizualizácie slov napr. word cloudom. Toto však môže byť dosiahnuté v ďalšom použití v inej aplikácii/systéme.

Ukladanie GPS dát a prácu s nimi je možné vylepšiť databázovým systémom, alebo špeciálnym modulom pre lepšiu prácu; aktuálne však funguje na klasickej MySQL databázy bez špeciálnych nástrojov pre prácu s geografickými dátami, čo je pre účely tejto práce dostatočné.

V neposlednom rade je možnosť úpravy hlavného Effortix systému – neexistuje reálne zaradenie položiek do kategórii (pre ďalší rozmer BI kocky), poprípade priradenie objednávky ku predajom (možnosť prezerať štatistiky práce predajcov).

10 Záver

Požiadavky, ktoré boli stanovené, boli splnené. Menším problémom však bolo jedine nedostatočné množstvo dát – atribútov, ktoré si systém ukladá a taktiež však môže byť aj vek samotného systému, keďže systém Effortix obsahuje zatiaľ menší objem dát, s ktorými disponuje. Systém zatiaľ nedisponuje veľkým množstvom zákazníkov, špeciálne platiacích, a lepšie použitie bude možné v ďalších fázach, kedy bude možné využiť a doladiť funkcie pre komercionalizáciu.

Z dôvodu, že sa jedná o systém pre interné účely a nebude poskytnutý širokej verejnosti, bolo možné použiť jednoduchšie nástroje na algoritmy, ktoré sú rýchlejšie na implementáciu, ale náročnejšie na prevedenie.

Vývoj systému bude pokračovať ďalej a podľa potrieb sa bude upravovať nielen ten, ale aj samotný Effortix, ako zdroj dát v prípade, že to budú ďalšie štatistiky vyžadovať. Reporting subsystém v tejto práci bol vytvorený len ako ukážka jeho použitia, skutočné použitie dát zo systému však bude zamerané na iné, už fungujúce systémy a ich využitie je zatiaľ minimálne plánované smerovať do hlavného Effortix systému a externého e-mailového serveru.

11 Literatúra

AGRAWAL, R. IMIELINSKI, T. SWAMI, A. *Mining Association Rules Between Sets of Items. in Large Databases* : SIGMOD Conference, 1993, 207-216.

AHMED, K. NAFEES A T. ABDUL RAZAK. *An Overview of Various Improvements of DBSCAN Algorithm in Clustering Spatial Databases. International Journal of Advanced Research in Computer and Communication Engineering* [online]. 2016, , 360-363 [cit. 2016-04-28]. Dostupné z: http://www.ijarcce.com/upload/2016/february-16/IJARCCE%2077.pdf

BLAŠKOVÁ, V. TIRPÁKOVÁ, A. MARKECHOVÁ, D. STEHLÍKOVÁ, B. MOLL, I. A STŘELEC, L. *Statistika I. Vydání třetí přepracované*. Brno: Mendelova univerzita v Brně, 2015. ISBN 978-80-7509-329-5.

BRAMPTON, M. *PHP5 CMS framework development: expert insight and practical guidance to creating an efficient, flexible and robust framework for a PHP5-based content management system*. Birmingham ,U.K.: Packt Pub., c2008. From technologies to solutions.

BURD, BARRY. *Java for dummies. Sixth edition*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2014. --For dummies. ISBN 978-1-118-61285-9.

CHAUDHURY, A. A RAO, H. R. *INTRODUCING CLIENT/SERVER TECHNOLOGIES IN INFORMATION SYSTEMS CURRICULA* [online]. Buffalo, 1996 [cit. 2016-04-29]. Dostupné z: http://wings.buffalo.edu/academic/department/som/isinterface/pedagogy/client-s.html

DURAISWAMY, K. A MUMTAZ, K. *An Analysis on Density Based Clustering of Multi Dimensional Spatial Data*. Indian Journal of Computer Science and Engineering [online]. 2012, 8-12 [cit. 2016-04-28]. Dostupné z: http://www.feridunozcakir.com/bnmfls/dbscan.pdf

GOLDSTEIN, A. LAZARIS, L. WEYL, E. *HTML5 & CSS3 for the real world*. 1. vyd. Collingwood: SitePoint, 2011. 342 s. ISBN 978-0-9808469-0-4.

HAN, JIAWEI, MICHELINE KAMBER A JIAN PEI. *Data mining: concepts and techniques*. 3rd ed. Boston: Elsevier, c2012. Morgan Kaufmann series in data management systems. ISBN 978-0-12-381479-1.

HEROUT, P. *Učebnice jazyka Java. 3.*, rozš. vyd. [i.e. 4. vyd.]. České Budějovice: Kopp, 2008. ISBN 978-80-7232-355-5.

KELLER, L. K. KOTLER, P. *Marketing management*. Praha: Grada, 2013. 816 s. ISBN 978-80-247-4150-5.

NIEDERST R. J., *Learning Web design: a beginner's guide to HTML, CSS, JavaScript, and web graphics*. Fourth edition. Beijing: O'Reilly, 2012. ISBN 9781449319274.

NOVOTNÝ, O., POUR, J. A SLÁNSKÝ, D. *Business intelligence: jak využít bohatství ve vašich datech*. 1. vyd. Praha: Grada, 2005. Management v informační společnosti. ISBN 80-247-1094-3.

PECINOVSKÝ, R. *Návrhové vzory : [33 vzorových postupů pro objektové programování]*. 1. vyd. Brno: Computer Press, 2007. 527 s. ISBN 978-80-251-1582-4.

POUR, J., MARYŠKA, M. NOVOTNÝ, O. *Business intelligence v podnikové praxi*. 1. vyd. Praha: Professional Publishing, 2012. 276 s. ISBN 978-80-7431-065-2.

RAJARAMAN, A. LESKOVEC, J. A ULLMAN, J. *Mining of Massive Datasets*. 2012.

SHELDON, R. A MOES, G. *Beginning MySQL*. Indianapolis, IN: Wiley Pub., 2005.

VOLKAN, T. BILGIN, T. A ÇAMURCU, A. *An Improved Clustering Algorithm for Text Mining: Multi-Cluster Spherical K-Means*. International Arab Journal of Information Technology [online]. 2016, 13(1), 12-19 [cit. 2016-04-27]. Dostupné z: https://www.researchgate.net/publication/273632084_An_Improved_Clustering_Algorithm_for_Text_Mining_Multi-Cluster_Spherical_K-Means

VRÁNA, J. *1001 tipů a triků pro PHP*. 1. vyd. Brno: Computer Press, 2010. 456 s. ISBN 978-80-251-2940-1.

WITTEN, I, FRANK, E. A Mark A H. *Data mining: practical machine learning tools and techniques*. 3rd ed. Burlington: Morgan Kaufmann, c2011. Morgan Kaufmann series in data management systems. ISBN 978-0-12-374856-0.

ZENDULKA, J. A KOL. *Získávání znalostí z databází ZZN: Studijní opora*. 2009, Brno.

12 Internetové zdroje

10 PHP FRAMEWORKS FOR DEVELOPERS – Best Of. In: Hongkiat [online]. 2015 [cit. 2016-04-30]. Dostupné z: http://www.hongkiat.com/blog/best-php-frameworks/

APRIL 2016 WEB SERVER SURVEY. In: Netcraft [online]. [cit. 2016-04-28]. Dostupné z: http://news.netcraft.com/archives/2016/04/21/april-2016-web-server-survey.html

EFFORTIX. In: Effortix [online]. 2016 [cit. 2016-04-30]. Dostupné z: https://www.effortix.com/

EICK, CH. *Density base clustering*. In: Department of Computer Science , University of Houston [online]. 2011 [cit. 2016-04-27]. Dostupné z: http://www2.cs.uh.edu/~ceick/ML/Topic9.ppt

GOOGLE DEVELOPERS. In: Google Maps APIs [online]. 2016 [cit. 2016-04-30]. Dostupné z: https://developers.google.com/maps/

JAVA. In: Oracle [online]. [cit. 2016-04-30]. Dostupné z: https://community.oracle.com/community/java

JETTY. In: Jetty – servlet engine and Http server [online]. 2016 [cit. 2016-04-30]. Dostupné z: http://www.eclipse.org/jetty/

JQUERY UI. In: JQuery user interface [online]. [cit. 2016-04-30]. Dostupné z: https://jqueryui.com/

KORELAČNÝ DIAGRAM. In: Management systems [online]. [cit. 2016-04-25]. Dostupné z: http://www.msys.sk/nastroje_korelacny_diagram.htm

MOLINO, P. *Indexing, vector spaces, search engines* [online]. In: . [cit. 2016-04-27]. Dostupné z: http://www2.cs.uh.edu/~ceick/ML/Topic9.ppt

NETTE FRAMEWORK. In: *Nette framework* [online]. 2016 [cit. 2016-04-30]. Dostupné z: https://nette.org/

PHP DOCUMENTATION. In: PHP.net [online]. [cit. 2016-04-28]. Dostupné z: http://php.net/

PLATY.SK. In: Platy.sk [online]. 2016 [cit. 2016-05-09]. Dostupné z: http://www.platy.sk/platy/informacne-technologie/business-intelligence-specialist

SQL GROUP BY CLAUSE. In: Software Testing Class [online]. 2013 [cit. 2016-04-30]. Dostupné z: http://www.softwaretestingclass.com/sql-group-by-clause/

SQLITE. In: SQLite [online]. [cit. 2016-04-30]. Dostupné z: https://www.sqlite.org/

USAGE OF SERVER-SIDE PROGRAMMING LANGUAGES FOR WEBSITES. In: W3techs [online]. [cit. 2016-04-28]. Dostupné z: http://w3techs.com/technologies/overview/programming_language/all

VILO, J. *Data mining lectures*. In: Institute of Computer science, Tartu University [online]. 2014 [cit. 2016-04-27]. Dostupné z: https://courses.cs.ut.ee/2014/dm/spring/Main/Lectures

WHAT IS THE APACHE HTTP SERVER PROJECT? In: Apache HTTP server project [online]. [cit. 2016-04-28]. Dostupné z: http://httpd.apache.org/ABOUT_APACHE.html

WHAT IS SMARTY? In: Smarty template engine [online]. [cit. 2016-04-30]. Dostupné z: http://www.smarty.net/about_smarty

WIKIQUOTE. Ronald Coase. In: Wikiquote [online]. 2016 [cit. 2016-05-09]. Dostupné z: https://en.wikiquote.org/wiki/Ronald_Coase

Prílohy

A Priložené CD

- zdrojový kód v zaheslovanom archíve (heslo bude poskytnuté vedúcemu a oponentovi práce)
- dokumentácia dátového skladiska