VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ALGORITMUS PRO CÍLENÉ DOPORUČOVÁNÍ PRODUKTŮ

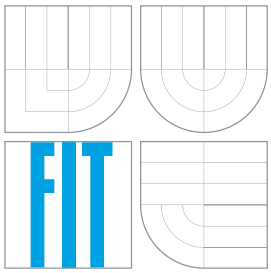
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MIROSLAV BODEČEK

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ALGORITMUS PRO CÍLENÉ DOPORUČOVÁNÍ PRODUKTŮ

ALGORITHM FOR PRODUCT RECOMMENDATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MIROSLAV BODEČEK
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JAROSLAV ZENDULKA, CSc.
SUPERVISOR

BRNO 2011

Abstrakt

Tato diplomová práce se zabývá prozkoumáním problematiky doporučování produktů v internetovém obchodování, zhodnocením dostupných technik, detailním návrhem systému doporučování produktů pro existující internetový obchod a implementací tohoto systému včetně otestování. V technické zprávě je nejprve prezentován úvod do problematiky, představen současný stav v internetovém obchodování a specifikovány požadavky na implementaci nadstavby nad internetovým obchodem. Dále zpráva obsahuje úvod do dolování dat. Následuje detailní návrh systému a zpráva o provedeném testování. Závěr obsahuje zhodnocení dosažených výsledků a diskuzi o možném dalším vývoji.

Klíčová slova

internetový obchod, elektronický obchod, doporučování produktů, asociační pravidla, data mining, dolování dat, kolaborativní filtrování

Abstract

The goal of this project is to explore the problem of product recommendations in the area of e-commerce and to evaluate known techniques, design product recommendation system for an existing e-commerce site, implement it and test it. This report introduces the problem, briefly examines current state of affairs in this area and defines requirements for a product recommendation module. The concept of data mining in general is introduced. The report proceeds to present detailed design corresponding to defined requirements and summarizes data gathered during testing phase. It concludes with evaluation and with discussion of the remaining goals for this thesis.

Keywords

e-commerce, internet commerce, product recommendation, association rules, data mining, collaborative filtering, apriori

Citace

Miroslav Bodeček: Algoritmus pro cílené doporučování produktů, diplomová práce, Brno, FIT VUT v Brně, 2011

Algoritmus pro cílené doporučování produktů

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením doc. Jaroslava Zendulky. Jako konzultant byla k projektu přizvána společnost MSPS s.r.o, která zajišťovala zejména prvotní koncept projektu a dále konzultace při volbě vhodných algoritmů,. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Miroslav Bodeček
24. dubna 2011

Poděkování

Děkuji doc. Jaroslavu Zendulkovi za cenné připomínky, které mi pomohly k vypracování této práce.

Obsah

1 Úvod **4**
- 1.1 Analýza nákupních košíků . . . 4
  - 1.1.1 Příklad analýzy nákupních košíků . . . 5
- 1.2 Doporučování produktů v internetovém obchodě . . . 5
  - 1.2.1 Amazon.com . . . 6
  - 1.2.2 Barnes & Noble . . . 6
  - 1.2.3 Internetové obchody v ČR . . . 6
- 1.3 Požadavky na nadstavbu realizující doporučení produktů . . . 7
  - 1.3.1 Doporučení na stránce detailu produktu . . . 7
  - 1.3.2 Doporučení na stránce nákupního košíku . . . 7
- 1.4 Dostupná data . . . 7

2 Dolování dat **9**
- 2.1 Co je dolování dat . . . 9
- 2.2 Zdroje dat . . . 9
- 2.3 Proces získávání znalostí . . . 10
- 2.4 Předzpracování dat . . . 10
  - 2.4.1 Čištění dat . . . 11
  - 2.4.2 Integrace dat . . . 12
  - 2.4.3 Transformace dat . . . 13
  - 2.4.4 Redukce dat . . . 14
- 2.5 Asociační pravidla . . . 15
  - 2.5.1 Definice asociačního pravidla . . . 15
  - 2.5.2 Atributy pravidla . . . 16
  - 2.5.3 Frekventovaná množina, silné asociační pravidlo . . . 16
  - 2.5.4 Lift pravidla . . . 16
  - 2.5.5 Algoritmus Apriori . . . 17
  - 2.5.6 Jiné typy asociačních pravidel . . . 18
- 2.6 Práce společnosti Amazon v oblasti analýzy nákupních košíků . . . 18
  - 2.6.1 Index podobnosti . . . 18
- 2.7 Další dolovací úlohy . . . 19
  - 2.7.1 Klasifikace . . . 19
  - 2.7.2 Odhadování . . . 19
  - 2.7.3 Predikce . . . 19
  - 2.7.4 Seskupování podle afinity . . . 19
  - 2.7.5 Shluková analýza . . . 19
  - 2.7.6 Profilování . . . 20

3 Návrh a implementace **21**
3.1 Současný stav aplikace Czechcomputer.cz . . . 21
3.2 Generování doporučení . . . 22
3.2.1 Zjednodušení Apriori pro k = 2 . . . 22
3.2.2 Časová složka . . . 23
3.2.3 Váha pravidla . . . 24
3.2.4 Kombinované doporučení . . . 24
3.3 Dvě varianty implementace . . . 25
3.4 Předzpracování dat . . . 26
3.4.1 Čištění dat . . . 26
3.4.2 Integrace dat . . . 26
3.4.3 Redukce dat . . . 27
3.5 Vymezení pojmů pro účely této implementace . . . 28
3.5.1 Transakce . . . 28
3.5.2 Položka . . . 28
3.5.3 Transakční prostor . . . 29
3.5.4 Asociační pravidlo . . . 29
3.5.5 Doporučení . . . 29
3.5.6 Parametry algoritmu . . . 29
3.6 Návrh databázové implementace . . . 29
3.6.1 Databázové schéma . . . 30
3.6.2 Průběh výpočtu . . . 31
3.7 Návrh paměťové implementace . . . 32
3.7.1 Proces generování doporučení . . . 32
3.7.2 Návrh jednotlivých tříd . . . 32
3.7.3 Příklad použití . . . 34
3.7.4 Automatické testy korektnosti . . . 34
3.7.5 Poznámky k paměťové implementaci . . . 36

4 Testování a výkon **39**
4.1 Použité parametry pro algoritmus Apriori . . . 39
4.2 Vzorky dat . . . 39
4.3 Statistiky vygenerovaných pravidel . . . 40
4.4 Testování vzájemné ekvivalence implementací . . . 40
4.5 Výkonnostní test generování pravidel . . . 40
4.5.1 Metodika . . . 41
4.5.2 Výsledky a zhodnocení . . . 41
4.6 Výkonnostní test vyhledávání pravidel . . . 41
4.6.1 Metodika . . . 41
4.6.2 Výsledky a zhodnocení . . . 42
4.7 Test paměťové náročnosti . . . 42
4.7.1 Metodika . . . 43
4.7.2 Výsledky a zhodnocení . . . 44

5 Závěr **45**
5.1 Shrnutí práce . . . 45
5.2 Pokračování této práce . . . 45
5.2.1 Úkony nad rámec této práce . . . 45

5.2.2 Další náměty pro rozšíření práce 46
5.3 Zhodnocení výsledků 46

A Návod k přeložení a spuštění demonstračního programu **48**
A.1 Požadavky 48
A.2 Přeložení 48
A.3 Spuštění 48
A.4 Parametry příkazové řádky 49
A.4.1 Základní parametry 49
A.4.2 Parametry spouštějící akce 49
A.4.3 Parametry algoritmu Apriori 49

Kapitola 1

Úvod

V této kapitole uvedu čtenáře do problematiky analýzy nákupních košíků v kontextu řízení vztahů se zákazníky (CRM), speciálně pak v oblasti internetových obchodů. Stručně přiblížím současný stav v této oblasti na poli internetových obchodů v ČR a v zahraničí. Následně popíši východiska a cíle této diplomové práce a zdroje, které jsou mi při řešení k dispozici.

1.1 Analýza nákupních košíků

Obchodování orientované na zákazníky je v dnešním světě převážně řízeno poptávkou. Proto vzniká logická snaha obchodníků nejen přizpůsobit nabídku, ale ovlivnit i zákazníky v jejich výběru. Odtud je krok k nabízení zboží zákazníkovi, o které bude mít pravděpodobně největší zájem. Kupříkladu všechny řetězce rychlého občerstvení mají vytvořeny pravidla, jaký typ nápoje má personál nabízet k objednávce. Pokud si zákazník objedná hranolky, obsluhující mu nabídne limonádu, pokud si objedná salát, měl by mu obsluhující nabídnout dietní nápoj, pokud si objedná koláč, je mu nabídnuta káva nebo čaj, a tak podobně.

Analýzou nákupních košíků nazýváme řadu technik, které vedou k pochopení vzorů v chování zákazníků při nákupu a aplikaci těchto vzorů. [1] Cílem je obvykle zvýšit zisk z prodeje a zvýšit uživatelský komfort.

Obchodní řetězce například používají tyto techniky pro umístění zboží na ploše obchodu. V závislosti na druhu zboží a kontextu se pak zboží prodávané společně buď umístí blízko sebe, aby zákazník na jedno z nich nezapomněl, nebo naopak co nejdále od sebe, aby zákazník musel při nákupu projít co největší část obchodu a nakoupil i další zboží.

V prodeji služeb se při telefonickém a letákovém marketingu používá znalosti o potenciálním zákazníkovi a na základě známých vzorů v chování doporučuje konkrétní produkty, argumentuje konkrétním způsobem nebo nabízí cílené slevy.

Internetové obchody využívají známých nákupních vzorů k nabízení konkrétního zboží nebo skupiny zboží v reálném čase, přímo na internetové stránce, kterou si uživatel zobrazil. Může to být například na stránce detailu konkrétního zboží nebo na stránce vedoucí k objednání.

1.1.1 Příklad analýzy nákupních košíků

Řekněme, že chceme provést analýzu nákupních košíků pro obchodní řetězec prodávající potraviny. Následující příklad byl převzat z [1]. Tabulka 1.1 ukazuje historii nákupu pěti různých zákazníků.

Tabulka 1.1: Nákupy jednotlivých zákazníků

	Zboží
Zákazník A	džus, limonáda
Zákazník B	mléko, džus, čistič oken
Zákazník C	džus
Zákazník D	džus, saponát, limonáda
Zákazník E	čistič oken, limonáda

Na základě těchto nákupu je možné sestavit matici společného výskytu 1.2. Tato matice ukazuje, kolikrát byla daná dvojice produktů prodána společně.

Tabulka 1.2: Společný výskyt zboží

	Džus	**Čistič oken**	**Mléko**	**Limonáda**	**Saponát**
Džus	4	1	1	2	1
Čistič oken	1	2	1	1	0
Mléko	1	1	1	0	0
Limonáda	2	1	0	3	1
Saponát	1	0	0	1	1

Už prostým pohledem na tuto tabulku je možné si všimnout několika potenciálních vzorů:

- ve dvou třetinách případů, kdy si zákazník koupil **limonádu**, si také koupil **džus**
- **saponát** není nikdy kupován společně s **čističem oken** nebo **mlékem**
- **mléko** není nikdy kupováno s **limonádou** nebo **saponátem**

Tyto vzory pravděpodobně nejsou z důvodu malé velikosti vstupních dat statisticky významné, ale dobře ukazují, jakými problémy se analýza nákupních košíků zabývá. V kapitole 2 k tomuto problému přistoupím formálněji a popíši *asociační pravidla*, která se používají k definování podobných vzorů.

1.2 Doporučování produktů v internetovém obchodě

Internetový obchod může mít k dispozici kompletní historii transakcí vykonaných nejen daným zákazníkem, ale i všemi ostatními zákazníky. Má na rozdíl od člověka k dispozici výpočetní výkon pro plné využití těchto dat.

Stále ale existuje řada věcí, ve kterých počítačový doporučovací systém člověka nezastoupí. Nemůže například bez rozsáhlých dat o uživateli odhadnout věk, sociální a ekonomické postavení ani celou řadu dalších proměnných, které zkušení obchodníci dokáží rozpoznat při osobním kontaktu. Chybí mu také schopnost aplikovat kontext a normy,

které jsou pro člověka zažité. Takto může vznikat řada situací, kde stroj selhává. V minulosti se například internetovému obchodu Amazon stalo, že zákazníkům, kteří si objednali počítačovou hru určenou pro předškolní děti, se zobrazilo na jejím základě doporučení počítačové hry, která je známá svými násilnými scénami a je určena pouze pro dospělé. I když se může jednat o reálný vzor v chování zákazníků, lidé obvykle na podobné asociace reagují negativně. To je příklad ukazující, že automatický systém má i své nevýhody.

1.2.1 Amazon.com

Amazon je jeden z nejúspěšnějších a zároveň největších zahraničních internetových obchodů. Obrázek 1.1 ukazuje příklad doporučování produktů ze serveru Amazon.com.

Obrázek 1.1: Doporučování na Amazon.com

Amazon používá pro doporučování produktů dva základní mechanismy [7]:

1. *Doporučování na základě podobnosti produktů*: algoritmus bere v potaz produkty, které uživatel zakoupil nebo kladně ohodnotil. Na jejich základě vytvoří seznam podobných produktů, které jsou pak uživateli doporučovány.

2. *Doporučování na základě nákupních košíků*: základem je historie objednávek od všech zákazníků. Z té se tvoří pravidla pro doporučování produktů podle položek, které má uživatel aktuálně v nákupním košíku.

Systém doporučování na Amazon.com funguje už téměř deset let. Neexistují veřejně dostupné statistiky o úspěšnosti tohoto systému. Nicméně vzhledem k prominentnímu umístění grafického prvku zobrazujícího doporučení a množství prostoru, kterého se mu už řadu let dostává, je možné předpokládat, že se jedná o velmi úspěšný kanál zvýšení objemu prodeje.

1.2.2 Barnes & Noble

Velký prodejce knih, který má většinu zisku z prodeje v USA a ve Spojeném království. Má systém doporučování, který je na pohled až překvapivě podobný tomu od Amazon.com.

1.2.3 Internetové obchody v ČR

V době psaní tohoto textu internetové obchody ze středního a vyššího segmentu působících v České republice doporučování produktů většinou nemají (např. Vltava.cz, Mall, Czechcomputer.cz a další). Jedním z obchodů z vyššího segmentu, který podobný systém má, je server Alza.cz. Tento internetový obchod integruje doporučování produktů do obchodního procesu,

kdy jsou vám po vložení do košíku doporučeny produkty, které Alza.cz „doporučuje koupit společně“. Bez podrobnějších znalostí ovšem není možné ověřit, zda se skutečně jedná o autonomní systém, nebo zda vyžaduje manuální definici pravidel pro doporučování.

1.3 Požadavky na nadstavbu realizující doporučení produktů

Server Czechcomputer.cz je jedním z největších internetových prodejců hardwarového vybavení, drobné elektroniky a příslušenství v České republice. Vzniká logická snaha pro takto rozsáhlý internetový obchod maximalizovat prodej technikami doporučování produktů. Vzhledem k rozsahu produktové databáze není možné doporučování cestou manuální definice doporučovacích pravidel. Doporučování tedy musí být autonomní. Systém by měl fungovat bez ručních zásahů ze strany administrátora.

Výpočet obou typů doporučení musí probíhat v reálném čase. Výpočet by měl být dostatečně rychlý, aby znatelně nezvyšoval odezvy při přístupu k internetovému obchodu. Nežádoucí je i nadměrné zatěžování serverů (databázového a aplikačního), a to zejména v době s běžnou návštěvností. Déle trvající výpočty je možné provádět v době s nízkou návštěvností (pozdní noc a brzké ráno), pokud budou výpočty neblokující a nijak neomezí používání internetového obchodu.

Pochopitelným požadavkem je účinnost doporučování. Toto hledisko je subjektivní, ovšem na základě dodatečné analýzy po nasazení systému bude možné získat informace o zvýšení prodeje, k jakému došlo na základě doporučování zboží.

Doporučení by se mělo objevit zejména na dvou stránkách: na stránce detailu produktu a na stránce nákupního košíku. Cílem této diplomové práce je tedy vytvořit programátorské rozhraní, pomocí kterého bude možné získat doporučení produktů pro tyto dva případy.

1.3.1 Doporučení na stránce detailu produktu

Stránka detailu produktu zobrazuje všechny základní údaje o produktu, fotogalerii, uživatelskou diskuzi a další informace. Nově bude obsahovat také doporučené produkty vybrané podle vhodných pravidel. Obrázek 1.2 ilustruje, jak by mohlo být takové doporučení zobrazeno na stránkách Czechcomputer u jednoho z produktů.

1.3.2 Doporučení na stránce nákupního košíku

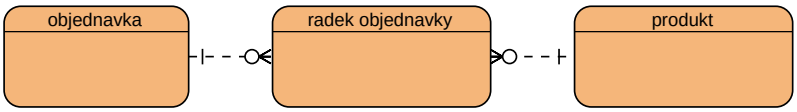
Na stránce nákupního košíku se běžně zobrazuje seznam produktů v košíku, který je možné upravovat. Tato stránka také umožňuje přistoupit k samotnému objednání zboží, jedná se tedy o první krok v objednávkovém procesu. Nově bude obsahovat produkty, které jsou doporučeny na základě produktů, které se už v košíku nachází. Programátorské rozhraní tedy musí umožňovat generování doporučení nejen podle jednotlivých produktů, ale i podle množin produktů (nákupních košíků).

1.4 Dostupná data

Jednoduchý diagram dostupných dat ukazuje obrázek 1.3. V produktové databázi se nachází asi 60 000 produktů. Pouze zlomek z těchto produktů je ale označen jako aktivní, tj. dostupných k objednání. Produkty postupem času přestávají být aktivní po ručním zásahu nebo po prošlé lhůtě pro publikaci. Toto nastává například v případě, kdy výrobek přestane být dodáván výrobcem. V současnosti je aktivních asi 9 000 produktů.

Obrázek 1.2: Možný grafický návrh detailu produktu

K dispozici je anonymizovaná historie všech objednávek v systému. Tyto objednávky obsahují položky objednávky, které reprezentují objednaný produkt. K dispozici je celkem asi 140 000 objednávek, které reprezentují půlroční historii obchodování koncových zákazníků na serveru Czechcomputer.cz. Tyto objednávky obsahují celkem asi 550 000 položek. Celkový počet odkazovaných aktivních produktů z půlroční historie objednávek je asi 7 000, tedy asi 2 000 produktů nebylo v posledním půlroce objednáno.



Obrázek 1.3: Objednávky

Kapitola 2

Dolování dat

V této kapitole vysvětlím pojem dolování dat v kontextu získávání znalostí z databáze, popíší používané techniky a zaměřím se na dolování dat vzhledem k analýze nákupních košíků.

2.1 Co je dolování dat

Dolování dat (anglicky data mining) je pojem zastřešující rozsáhlou skupinu technik, procesů a nástrojů, které slouží k analýze velkého množství dat a ke hledání skrytých, netriviálních a potenciálně užitečných vzorů v těchto datech. Ve velké části případu se dolování dat používá k vylepšení marketingových procesů, prodeje a podpory zákazníkům. Stejně tak je ale možné použít dolování dat v jiných oblastech, například v astronomii, v medicíně nebo v průmyslových odvětvích. [9]

Striktně vzato je dolování pouze jednou z fází rozsáhlejšího procesu získávání znalostí z databáze, i když se tyto pojmy běžně používají jako synonyma.

2.2 Zdroje dat

Dolovat můžeme v různých typech zdrojů dat, ať už se jedná o perzistentní data (databáze, soubory, apod.) nebo transientní data (proudy). Tři běžné druhy těchto zdrojů jsou:

- *Relační databáze* je pravděpodobně nejběžnější zdroj dat dneška. Výhodou je univerzální znalost těchto databází a dotazovacího jazyka SQL.
- *Datový sklad* - datové sklady obsahují zpravidla sumarizované údaje (např. celkové součty prodejů místo jednotlivých objednávek) a bývají modelovány jako tzv. *multidimenzionální datové kostky*. Jednotlivé dimenze odpovídají atributům (např. pobočka, kvartál, apod.), hodnoty jsou pak agregované údaje pro dané atributy (celkový prodej v Kč v daném kvartálu na dané pobočce). Datové sklady podporují OLAP operace, které zajišťují větší podporu pro analýzu a dolování dat, než relační databáze.
- *Transakční databáze* jsou specializované databáze, kde záznamy reprezentují transakce. Ty se obvykle skládají z jedinečného identifikátoru a množiny položek, které transakce obsahuje. Příkladem takových transakcí můžou být objednávky v internetovém obchodě nebo návštěvy webového serveru složené z jednotlivých HTTP požadavků.

Kromě těchto často užívaných zdrojů dat existuje samozřejmě celá řada dalších, jako jsou textové databáze (např. XML), specializované databáze (temporální, multimediální, apod.). Za zdroj dat můžeme také považovat web, kde můžeme automaticky analyzovat například shlukování a klasifikaci webových stránek nebo vývoj komunit a sociálních skupin.

2.3 Proces získávání znalostí

V této části popíši získávání znalostí jako proces složený z po sobě následujících fázích. Jsou to:

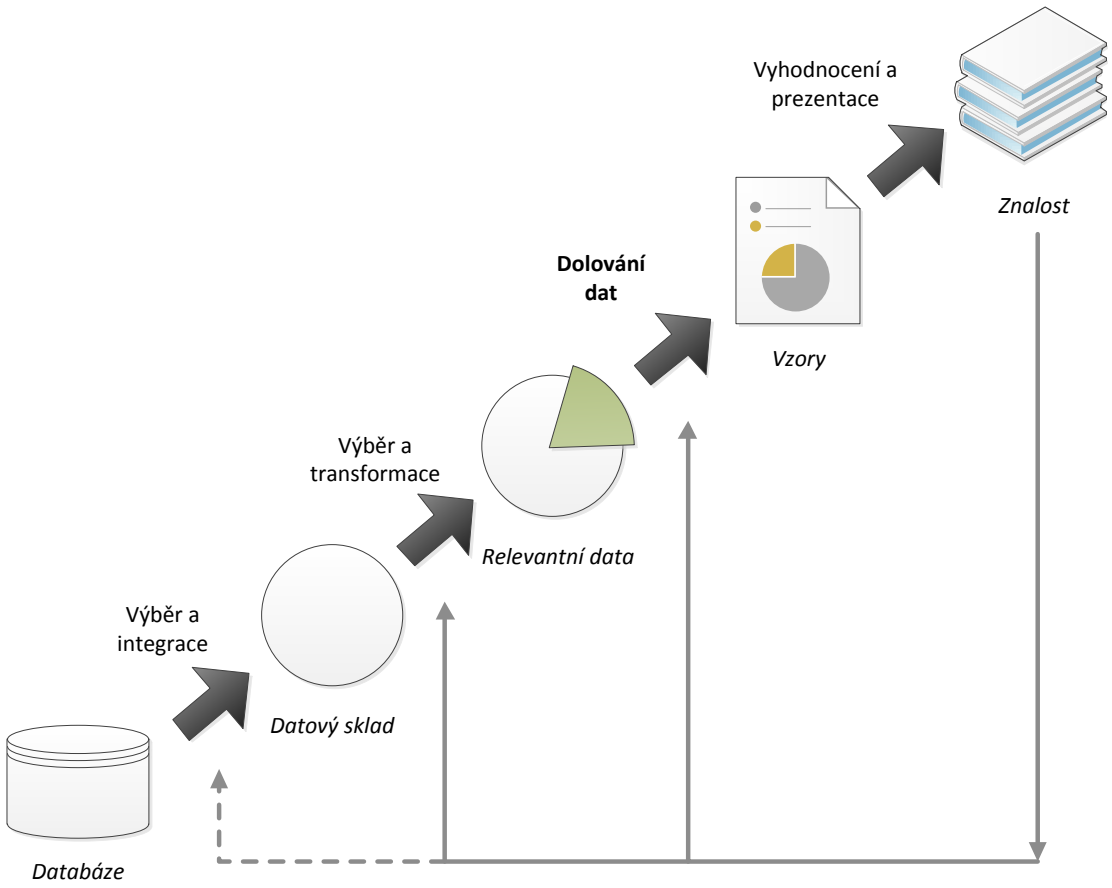
1. *Předzpracování dat*
   (a) *Čištění dat* - v této fázi jde o vyřešení problému chybných nebo chybějících dat různými technikami.
   (b) *Integrace dat* se snaží integrovat data pocházející z různých datových zdrojů. Může být prováděno společně s procesem čištění dat.
   (c) *Výběr dat* se snaží zvolit relevantní data v kontextu dané úlohy. Typicky může jít například o výběr sloupců z tabulky v relační databázi.
   (d) *Transformace dat* - v této fázi jsou data převedena do podoby vhodné pro dolování dat. Může se jednat o například o sumarizaci nebo agregaci.
2. *Dolování dat* - je to samotná aplikace konkrétní metody pro extrakci vzorů nebo vytvoření modelu dat.
3. *Hodnocení modelů a vzorů* - hodnotí vytvořené vzory různými způsoby a snaží se identifikovat zajímavé vzory.
4. *Prezentace znalostí* je důležitá pro správné pochopení vzorů uživatelem.

Tyto kroky se opakují v několika po sobě následujících iteracích, jak ukazuje obrázek 2.1.

2.4 Předzpracování dat

Předzpracování dat slouží k převedení dat do takové podoby umožňující dolování dat. Původní data mohou být příliš rozsáhlá, mohou být obsaženy ve více různých datových zdrojích, případně mohou mít problémy, které bez předzpracování znemožňují dolování. Mezi tyto problémy patří:

- *Nekompletnost dat* - může být způsobena například chybějícími atributy, faktem že data, která bychom potřebovali detailní, máme pouze agregovaná a podobně.
- *Šum* - data mohou obsahovat nesprávné hodnoty. Chybné hodnoty je částečně možné odhalit, pokud jsou například příliš odlehlé nebo se nachází mimo přirozený rozsah pro danou doménu (např. stáří vozu v databázi vozidel nastaveno na 160 let). Data mohou být také v nesprávném formátu (např. formátování datumů) nebo odpovídat jiné konvenci, než se pro aplikaci předpokládá (např. uživatelské hodnocení „jako ve škole“ oproti hodnocení „čím vyšší, tím lepší“).
- *Nekonzistence dat* - data mohou být vůči sobě nekonzistentní. Zdrojem je často přílišná redundance dat.



Obrázek 2.1: Proces získávání znalostí

2.4.1 Čištění dat

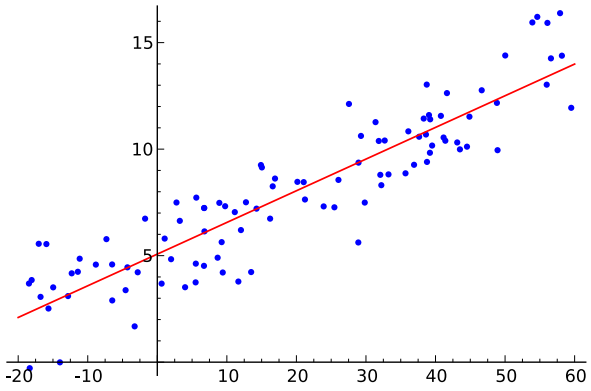
Čištění slouží zejména k odstranění nekompletních dat, vyhlazení šumu a vyřešení nekonzistence dat. Při odstranění problému chybějících atributy, můžeme použít některá z těchto řešení:

- *Ignorování n-tice* - tento způsob není příliš užitečný kromě případů, kdy je chybějících atributů více, nebo je atribut cílovou třídou v kontextu klasifikace dat.
- *Manuální nahrazení* - je v praxi stěží použitelný přístup kvůli velkému množství dat. Nicméně v případě že uživatel má znalost problematiky a jedná se o malé množství dat, je možné předpokládat dobré výsledky.
- *Automatické nahrazení* má několik variant:
  - nahrazení konstantou
  - nahrazení průměrnou hodnotou atributu
  - nahrazení průměrem v rámci stejné třídy jako je daná n-tice
  - nahrazení nejpravděpodobnější hodnotou - zde je možné použít hodnot ostatních atributů. Lze řešit jako samostatnou úlohu pro dolování dat, ke které je možné použít regresi, Bayesovskou klasifikaci nebo rozhodovací strom

Pro vyřešení šumu v datech je možné použít řadu technik vyhlazování. Šum v datech může vzniknout např. poruchou zařízení, lidskou chybou při zadávání dat, apod.

- *Plnění (binning)* vyhlazuje numerická data. Nejprve se seřazená data rozdělí do tzv. košů, každý o přibližně stejném počtu prvků. Pak se atributy v každém koši upravují podle koše samotného. Můžou být například změněny na průměr všech hodnot v koši nebo jejich medián. Plnění tedy provádí lokální vyhlazování.

- *Regrese* - data se nahrazují hodnotami danými regresní křivkou. Může se jednat o lineární regresi, v tom případě se hledá přímka, která aproximuje hodnoty dvou atributů, jak je znázorněno na obrázku 2.2



Obrázek 2.2: Lineární regrese

- *Shlukování* se používá pro hledání odlehlých hodnot. Jsou to potenciálně ty hodnoty, které nejsou zařazeny v žádném shluku.

- *Diskretizace* nahrazuje numerický atribut intervalem hodnot. Používají se dva způsoby:

  - *Rozčlenění na intervaly stejné šířky* - rozsah se rozdělí na N disjunktních intervalů stejné šířky. Šířka každého intervalu bude $\frac{max-min}{N}$, kde max a min je maximální, resp. minimální hodnota ze všech hodnot. Problémem této metody je, že odlehlé hodnoty se můžou stát dominantní a řada zajímavých hodnot může padnout do stejného rozsahu. Tento problém je možné řešit vyloučením určitého procenta nejnižších a nejvyšších hodnot.
  - *Rozčlenění na intervaly stejné hloubky* - vytvoří se intervaly, kde každý interval má přibližně stejný počet položek. Tento způsob je vhodný pro práci s vychýlenými daty.

2.4.2 Integrace dat

Data často pochází z více než jednoho zdroje. V takovém případě je nutné data integrovat do jediného zdroje, se kterým můžou úlohy dolování dat pracovat. Mezi hlavní problémy, které integrace dat musí řešit, patří:

- *Konflikt schématu* - pokud jsou metadata dvou nebo více zdrojů navzájem nepřevoditelná, mluvíme o konfliktu schématu. Například pokud v jednom datovém zdroji jsou čtyři

pole pro adresu trvalého bydliště pojmenované *ulice*, *číslo domu*, *PSČ* a *město* a ve druhém zdroji je pouze jedno pole - *adresa*.

- *Konflikty hodnot* - pokud jsou hodnoty z různých zdrojů navzájem odlišné i když označují stejnou skutečnost, mluvíme o konfliktu hodnot. Například pokud je v jednom datovém zdroji rok narození uvedena hodnota 1986 a ve druhém je to 86.
- *Konflikty identifikace* - ne vždy je možné jednoduše zjistit, které dva záznamy jsou identické. Může se jednat o problém různých identifikátorů v různých databázích, absenci identifikátoru, atp. Proto je třeba data navzájem namapovat a identifikaci sjednotit.
- *Redundance* - hodnoty z různých zdrojů můžou být navzájem redundantní. Navíc je některé hodnoty možné odvodit z jiných, takže se výskyt odvozených hodnot také stává redundantní. Například jeden zdroj může uvádět, že v dotazníku provedeném v roce 2008 uvedl dotazovaný věk 28 let. V jiném zdroji může být uveden jeho rok narození jako 1980, proto se hodnota věku stává redundantní.

2.4.3 Transformace dat

Úkolem transformace je převést data do podoby vhodné pro dolování dat. Mezi operace, které transformace může zahrnovat, patří:

- *vyhlazení šumu*, jak bylo popsáno v části 2.4.1
- *agregace* - detailní data jsou agregovány pro zmenšení rozsahu dat a zrychlení výpočtu
- *generalizace* - nahrazení dat koncepty, například kategorický atribut ulice mohou být na vyšší úrovni nahrazeny názvem města nebo kraje
- *normalizace* - cílem je normalizovat numerické hodnoty do určitého rozsahu hodnot, typicky $\langle 0, 1\rangle$ nebo $\langle -1, 1\rangle$
- *konstrukce* - vytvoření nových atributů za použití jiných, typicky pro zrychlení nebo zkvalitnění procesu dolování

Normalizaci je třeba provádět obvykle proto, že to vyžaduje příslušný dolovací algoritmus. Obvykle jsou to některé metody klasifikace, metody založené na vzdálenosti, jako je klasifikace metodou nejbližšího souseda a shlukování. Popíši zde tři často používané způsoby normalizace.

Min-max normalizace spočívá v lineární transformaci původních dat. Pokud jsou min_A a max_A minimální a maximální hodnoty atributu A a $nmin_A$ a $nmax_A$ jsou minimální a maximální hodnoty požadovaného rozsahu, potom hodnota v atributu A se transformuje pomocí vztahu

$$v' = \frac{v - min_A}{max_A - min_A}(nmax_A - nmin_A) + nmin_A$$

Z-skóre normalizace je vhodná, když není předem známý rozsah normalizovaného atributu nebo když existují odlehlé hodnoty, kvůli kterým nelze použít min-max normalizace. V případě z-skóre normalizace jsou hodnoty atributuA transformovány na základě průměru a standardní odchylky A. Hodnota se transformuje podle vztahu

$$v' = \frac{v - \bar{A}}{\sigma_A}$$

kde $\bar{A}$ je průměr a σ_A je směrodatná odchylka atributu A.

Normalizaci dekadickou změnou měřítka spočívá pouze v posunutí desetinné tečky tak, aby výsledný rozsah byl $(-1.0, 1.0)$. Hodnota v atributu A se transformuje na hodnotu v' podle vztahu:

$$v' = \frac{v}{10^j}$$

kde j je nejmenší celé číslo takové, že $max(|v'|) < 1$.

Mějme například rozsah atributu A od -897 do 152. Normalizace bude podle vztahu

$$\begin{aligned} v'_{min} &= -897/10^4 = -0,897 \\ v'_{max} &= 152/10^4 = 0,152 \end{aligned}$$

spočívat pouze ve vydělení transformovaného čísla hodnotou 1000. Normalizovaný rozsah tedy bude $\langle -0,897, 0,152 \rangle$.

2.4.4 Redukce dat

Redukce se snaží zmenšit rozsah dat při zachování stejných výsledků dolování. Redukci dat můžeme provést různými způsoby:

- *agregace datové kostky* - redukce agregováním detailních údajů
- *výběr podmnožiny atributů* se snaží eliminovat nerelevantní nebo málo relevantní atributy a ponechat jen tu podmnožinu atributů, která se pro dolování skutečně použije
- *redukce dimenzionality* - data se zakódují takovým způsobem, že dojde k redukci, ale bude možné provádět s daty potřebné operace
- *redukce počtu hodnot* - kompletní data jsou nahrazena nějakým modelem a reprezentována jeho parametry nebo jsou reprezentována v nějaké redukované podobě
- *diskretizace a generace konceptuální hierarchie* - hodnoty atributů jsou nahrazeny intervaly nebo pojmy z nějaké konceptuální hierarchie.

Cílem výběru podmnožiny atributů je vybrat jen některé z mnoha atributů ve zdrojových datech a přitom zachovat stejné výsledky, jako bychom pracovali s původní množinou atributů. Výběr atributů je někdy možné provést se znalostí sémantiky ručně, ale při velkém množství atributů je nutné proces automatizovat.

Uvažujme množinu N atributů. Prostor řešení je tedy 2^N. Není možné prohledávat celý prostor z důvodu časové náročnosti, takže se používají heuristické metody, které nezaručují nalezení globální optima za každých okolností. Heuristické metody, které je možné použít, zahrnují:

- *postupný dopředný výběr* - atributy se přidávají iterativně do výsledné množiny podle hodnocení (při každé iteraci se vybere atribut s nejvyšším hodnocením a přidá se)
- *postupná zpětná eliminace* - začíná se s množinou všech atributů a v každé iteraci se atributy hodnotí a odstraňuje se ten nejhorší atribut
- *indukce rozhodovacího stromu* - zde se používají algoritmy pro generování rozhodovacího stromu, jako se používají v klasifikačních úlohách. Redukovanou množinu atributů pak tvoří ty atributy, které jsou v rozhodovacím stromu obsaženy

2.5 Asociační pravidla

Dolování asociačních pravidel obecně hledá zajímavé souvislosti a asociace mezi prvky rozsáhlých datových množin. [10, 277 s.] Existuje řada využití pro dolování asociačních pravidel, příklady takových zahrnují:

- *Analýza nákupních košíků* - základní myšlenkou je, že určité zboží zákazníci často kupují společně. Mezi typické otázky pro analýzu nákupních košíků patří „jak často si zákaznící kupují X ke svému Y?“ či „pokud si zákazník kupuje X, co si s největší pravděpodobností koupí také?“ Analýza nákupních košíků může být užitečná například pro přípravu reklamních kampaní, rozmístění zboží v supermarketu nebo doporučování produktů v internetovém obchodě.
- *Web mining* se zabývá hledáním vzorů v chování uživatelů webových stránek. To se provádí na základě záznamů o přístupech, které bývají asociovány s uživatelským sezením. Web mining odpovídá na otázky typu „pokud uživatel navštívil stránku www.idnes.cz, jaké další stránky s pravděpodobností nejméně X tento týden navštíví?“ V praxi se používá k vylepšení informační architektury webů, k lepšímu rozmístění odkazů ve stránce nebo v internetových reklamních kampaních.
- *Detekce podvodů*, typicky v oblasti pojišťovnictví. Jako vstupní data slouží hlášené pojistné události. Dolování asociačních pravidel se zde používá k prvotnímu výběru subjektů, které jsou pak podrobeny hlubším analýzám.

2.5.1 Definice asociačního pravidla

Nechť $I = \{i_1, i_2, i_3, \dots\}$ je množina položek. Dále nechť D je množina transakcí, kde každá transakce T je množina položek taková, že $T \subseteq I$. Ke každé transakci přísluší unikátní identifikátor `TID`. Nechť A je množina položek. Říkáme, že transakce T obsahuje A, pokud $A \subseteq T$. Asociační pravidlo je implikace tvaru $A \Rightarrow B$, kde $A \subset T$, $B \subset T$ a $A \cap B = \emptyset$. [1]

Množiny A, resp. B se někdy také označují jako *tělo* (body), resp. *hlava* (head) pravidla.[10] *Frekvence* množiny A je pak počet transakcí T v množině transakcí D, které obsahují A. Tedy

$$f(A) = card\{T_i : A \subseteq T_i, i = 1, 2, \dots m\}$$

Například v případě analýzy nákupních košíků obvykle položky reprezentují zboží nebo produkty a transakce reprezentují nákupy či objednávky.

[1] Definice převzata z [11]

2.5.2 Atributy pravidla

Jsou definovány dva základní atributy každého pravidla:

- *podpora* (support) pravidla $A \Rightarrow B$ je poměr počtu transakcí obsahujících tělo i hlavu pravidla k počtu všech transakcí. Formálněji můžeme psát

$$support(A \Rightarrow B) = \frac{f(A \cup B)}{|D|}$$

- *spolehlivost* (confidence) pravidla $A \Rightarrow B$ je poměr počtu transakcí obsahujících tělo i hlavu pravidla k počtu transakcí obsahujících tělo.

$$confidence(A \Rightarrow B) = \frac{f(A \cup B)}{f(A)}$$

2.5.3 Frekventovaná množina, silné asociační pravidlo

Při získávání asociačních pravidel se snažíme získat tzv. *silná asociační pravidla* složená z *frekventovaných množin.*

- *frekventovaná množina* je každá množina položek A, která má podporu

$$support(A) = \frac{f(A)}{|D|}$$

  vyšší nebo rovnou definované minimální podpoře

- *silné asociační pravidlo* je každé, které má podporu i spolehlivost vyšší nebo rovnou definované minimální podpoře a spolehlivosti

2.5.4 Lift pravidla

I když se silné asociační pravidla mohou používat pro reprezentaci zajímavosti z obchodního hlediska, není to vždy nejvhodnější kritérium.[10, 2]

Uvažujme prodejce elektroniky, který z celkovém objemu 100 transakcí měl 60 transakcí obsahující fotoaparát, 75 transakcí obsahující tiskárnu a 40 transakcí obsahujících fotoaparát i tiskárnu. Pokud bychom uvažovali například minimální podporu $0,3$ a minimální spolehlivost $0,6$ bez ohledu na okolnosti, zjistíme že pravidlo $\{fotoaparat\} \Rightarrow \{tiskarna\}$ bude mezi silnými pravidly ($support = 40/100 = 0,4$ a $confidence = 40/60 = 0,666\ldots$). Toto pravidlo naznačuje, že fotoaparát a tiskárna se často prodávají společně.

Při hlubším zamyšlení ale zjistíme, že pravděpodobnost koupě tiskárny a fotoaparátu zároveň, je pouze $40/60 = 0,666\ldots$, což je méně než samotná pravděpodobnost koupě tiskárny $75/100 = 0,75$. Tedy zákazník si naopak spíše koupí tiskárnu, pokud si *nekupuje* fotoaparát.

Další atribut, který se u pravidla uvádí, a která tento nedostatek řeší, je *lift.*

$$lift(A \Rightarrow B) = \frac{confidence(A \Rightarrow B)}{support(B)} = \frac{f(A \cup B)|D|}{f(A)f(B)}$$

Lift nám říká, kolikrát větší je pravděpodobnost, že se hlava pravidla vyskytne v transakci zároveň s tělem, než že se hlava vyskytne bez ohledu na přítomnost těla pravidla. Pokud je lift větší než 1, znamená to, že pravidlo vykazuje pozitivní korelaci výskytu těla a hlavy zároveň. Lift menší než 1 značí negativní korelaci.

2.5.5 Algoritmus Apriori

K nalezení frekventovaných množin se velmi často používá algoritmus Apriori. K vyšší efektivitě se využívá tzv. Apriori vlastnost. Tato vlastnost říká, že každá podmnožina frekventované množiny musí být také frekventovaná. To vychází z faktu, že přidání prvku k množině nemůže způsobit, že by její podpora vzrostla.

Algoritmus Apriori pro nalezení množiny L_k, která se skládá ze všech frekventovaných k-množin, má tyto kroky:

1. Nalezení množiny L_{k-1} pomocí algoritmu Apriori
2. Spojovací krok: množina C_k je vytvořena jako $C_k \leftarrow \{l_1 \cup l_2;\ l_1, l_2 \in L_{k-1}\}$
3. Vylučovací krok: z množiny C_k jsou odebrány ty prvky, které mají alespoň jednu podmnožinu, která není frekventovaná

Zde C_k označuje kandidátní množinu k-prvkových množin prvků.

Program 1 ukazuje v pseudokódu iterativní podobu algoritmu $Apriori(D, \varepsilon)$, kde D je množina transakcí a ε je minimální podpora.

Program 1 Algoritmus Apriori

```
L[1] :=  všechny položky v~D s~výskytem >= ε
k~:= 2
while L[k - 1] is not empty:
  // spojovací krok
  C[k] := join(L[k - 1])

  // vylučovací krok
  for c in C[k]:
    for s~in subsets(c):
      if s~is not in L[k - 1]:
        remove c from C[k]
        continue outer cycle

  // aktualizace count
  for t in T:
    for c in C[k]:
      if c subset of t:
        count[c]++

  // vytvoření L[k]
  for c in C[k] with count[c] >= ε:
    add(L[k], c)

result := union of L[1..k]
```

Na základě znalosti množiny L_k je pak možné sestavit asociační pravidla následujícím způsobem:

- pro každou frekventovanou podmnožinu $l \in L_k$ se vygenerují její neprázdné podmnožiny $s_1 \dots s_n$

- pro každou takovou podmnožinu s_i, kde $i \in \{1, \ldots n\}$, se vytvoří pravidlo $s_i \Rightarrow l \setminus s_i$
- pro pravidlo se vypočítá spolehlivost podle vzorce uvedeného v 2.5.2 (pokud je jeho spolehlivost vyšší než minimální, pak je pravidlo silné)
- podmínka na minimální podporu je u všech pravidel splněna vždy, protože pravidla jsou generována z frekventovaných množin

2.5.6 Jiné typy asociačních pravidel

Dosud uváděná asociační pravidla jsou striktně vzato jen podmnožinou obecných asociačních pravidel. Ta mohou být klasifikována podle následujících kritérií:

- Obecná asociační pravidla je možné rozdělit podle typu zpracovávaných hodnot:
  1. *booleovské* (např. $zakaznik\ je\ muz \Rightarrow nabidni\ "pivo"$)
  2. *kvantitativní* (např. $vek(zakaznik) \geq 18 \Rightarrow nabidni\ "pivo"$)
- Podle použitých proměnných:
  1. *jednodimenzionální* (např. $zakaznik\ kupuje\ "hranolky" \wedge zakaznik\ kupuje\ "steak" \Rightarrow zakaznik\ kupuje\ "pivo"$)
  2. *vícedimenzionální* (např. $zakaznik\ kupuje\ "salat" \wedge zakaznik\ je\ zena \Rightarrow zakaznik\ kupuje\ "dietni\ kola"$)
- Podle úrovně:
  1. *jednoúrovňová* (založeny na konkrétních položkách, např. $zakaznik\ koupil\ "caj" \Rightarrow nabidni\ "cukr"$)
  2. *víceúrovňová* (založeny na skupinách položek, např. $zakaznik\ koupil\ "horky\ napoj" \Rightarrow nabidni\ "cukr"$)

2.6 Práce společnosti Amazon v oblasti analýzy nákupních košíků

Společnost Amazon se této oblasti dolování dat od roku 2001 intenzivně zabývá, a přišla za tuto dobu s několika metodami pro doporučování produktů v internetovém obchodě Amazon.com. [7, 4, 5, 6]

Uvedená dokumentace sice nikde nepoužívá název „asociační pravidla“, ale použitý koncept je stejný. Jedná se o booleovské jednodimenzionální jednoúrovňová asociační pravidla. Ta mají navíc z důvodu velkých rozsahů dat pouze jednoprvkové množiny v hlavě i těle pravidla.

2.6.1 Index podobnosti

Je zde uvažován nový atribut pro hodnocení zajímavosti pravidla, tzv. *index podobnosti* (commonality index). Ten je definován následovně:

$$CI(A \Rightarrow B) = \frac{f(A \cup B)}{\sqrt{f(A)f(B)}}$$

Podle dokumentace tento atribut udává intuitivní podobnost zboží. [2]

2.7 Další dolovací úlohy

Dolovacích úloh je velké množství a mohou sloužit k mnoha úkolům. Zde jsou popsané některé úlohy, které se často používají. [1]

2.7.1 Klasifikace

Klasifikace spočívá ve zkoumání vlastností dříve neznámého objektu a přiřazení do jedné z předdefinovaných tříd. Charakteristické pro klasifikaci je existence předem známých definicí jednotlivých tříd a znalost tříd v trénovacím vzorku dat. Úkolem je vybudovat model, který je možné aplikovat na nezatříděná data a tato data zařadit co nejpřesněji do tříd.

2.7.2 Odhadování

Zatímco klasifikace se zabývá generováním diskrétních výstupů, tedy zařazením do jedné třídy, odhadování generuje spojité hodnoty. Může sloužit také jako forma klasifikace, kde příslušnost do třídy není jistá, ale je definována pravděpodobností příslušnosti do dané skupiny.

2.7.3 Predikce

Predikce je velmi podobná klasifikaci a odhadování. Liší se pouze v charakteru výstupní hodnoty. Pro predikci platí, že výstupní hodnota je veličina, která je známá v budoucnu. Většinou je závislá na budoucí události a není možné tedy ověřit její správnost.

2.7.4 Seskupování podle afinity

Účelem je rozpoznat, které objekty z množiny mají určitou logickou souvislost. Může se jednat například o rozpoznání produktů, které jsou často objednávány společně.

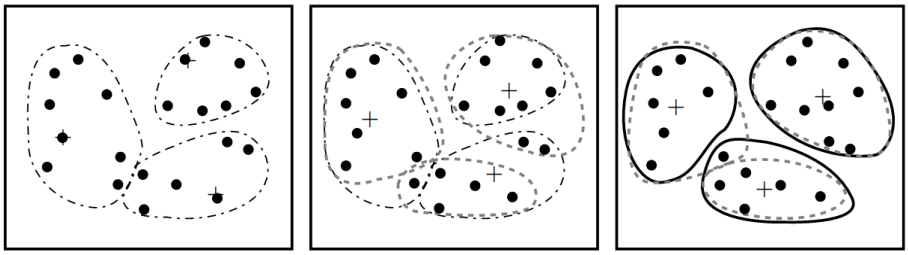
2.7.5 Shluková analýza

Spočívá v rozdělení množiny heterogenních položek do více homogenních podmnožin neboli shluků. Na rozdíl od klasifikace nespoléhá na předem definované třídy, ale na třídy definované v modelu algoritmicky. Třídy jsou samy o sobě anonymní a pouze reflektují vnitřní podobnost jednotlivých položek. Je na uživateli, aby určil, jaký význam náleží jednotlivým skupinám.

Obrázek 2.3 ukazuje jednu z metod používanou pro shlukování, k-means clustering. Tato metoda postupuje následujícím způsobem:

1. libovolně se vybere k prvků jako počáteční prvky shluků
2. každý prvek je zařazen do svého nejbližšího shluku podle vzdálenosti k jeho středu
3. je vypočten střed každého shluku jako průměr prvků, které do něj náleží
4. pokud se změnilo zařazení prvků při poslední iteraci, pokračuje se od bodu 2

[2] Jedná se o název z doby, kdy se Amazon zaměřoval zejména na prodej knih (později hudby a filmů). Zákazník na základě svého vkusu spíše zároveň objednává knihy, které jsou si obsahově a stylově podobné. Odtud název „index podobnosti“



Obrázek 2.3: K-means clustering, převzato z [3]. Kříže znázorňují středy shluků.

2.7.6 Profilování

Cílem je v tomto případě popsání vzorů a chování v datech, které vede k jejich vysvětlení.

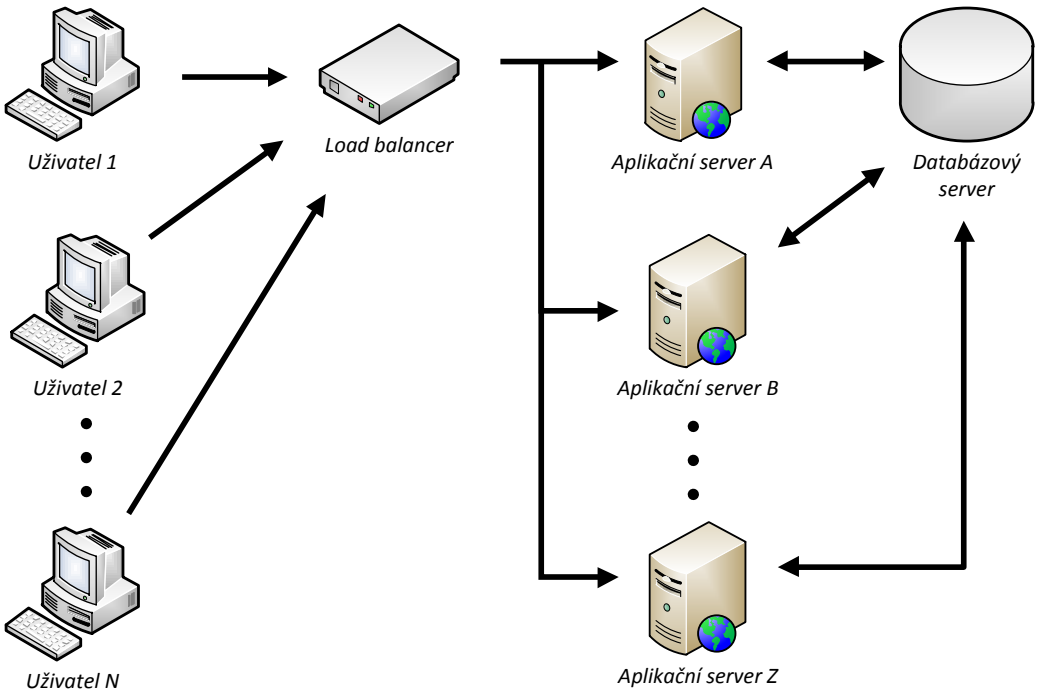
Kapitola 3

Návrh a implementace

V této kapitole stručně popíši současnou architekturu internetového obchodu Czechcomputer.cz a shrnu teoretické východiska pro implementaci. Dále popíši dvě varianty implementace, jejich výhody a nevýhody a provedu detailní návrh obou variant.

3.1 Současný stav aplikace Czechcomputer.cz

HTTP požadavky na internetovou adresu Czechcomputer.cz nejprve zpracuje zařízení pro rozdělení zátěže (load balancer). Load balancer předává požadavky dále na několik aplikačních serverů, jak ukazuje obrázek 3.1. Ty pak komunikují s vyhrazeným databázovým serverem, ukládají na něj data a data z něj čtou.



Obrázek 3.1: Architektura Czechcomputer.cz

Současná aplikace Czechcomputer využívá technologie:

- Java Standard Edition 6 a Java Enterprise Edition 5 na aplikačním serveru
- Oracle 11g na databázovém serveru

3.2 Generování doporučení

Při hledání vhodného způsobu pro generování doporučování produktů jsem vycházel z literatury, dokumentace společnosti Amazon a z konzultací se společností MSPS s r.o. Běžně používaný a efektivní způsob pro tvorbu doporučování bylo ve všech případech zmíněno dolování asociačních pravidel pomocí algoritmu Apriori. Vzhledem k nárokům na množství zpracovaných položek jsem zvolil booleovské jednodimenzionální jednoúrovňová pravidla ve tvaru $i \Rightarrow j$, kde i, j jsou položky. Tato forma asociačních je nenáročná na výkon a podle dostupných materiálů od společnosti Amazon dostatečně efektivní.

3.2.1 Zjednodušení Apriori pro k = 2

Algoritmus Apriori obecně generuje množiny L_k, které obsahují všechny k-prvkové frekventované množiny. Vzhledem k tomu, že námi generovaná pravidla mají tvar $i \Rightarrow j$, tedy $k = 2$, je možné obecný algoritmus zredukovat odstraněním cyklu. Jeho zápis je znázorněn v programu 2.

Program 2 Algoritmus Apriori pro k = 2 zapsaný v pseudokódu

```
L := { všechny položky v~T s~výskytem >= EPSILON }

// spojovací krok
C := join(L)

// vylučovací krok
for c in C:
  for s~in c:
    if s~is not in L:
      remove c from C
      continue outer cycle

// aktualizace count
for t in T:
  for c in C:
    if c subset of t:
      count[c]++

// vytvoření result
for c in C with count[c] >= EPSILON:
  add(result, c)
```

3.2.2 Časová složka

V analýze nákupních košíku se za transakce často považují konkrétní osamocené objednávky. To ale nemusí být ideální řešení, protože objednávky, které jsou v rychlém sledu po sobě, spolu mohou významně souviset. Vezměme si příklad, kdy si zákazník objedná digitální fotoaparát. Druhý den si pak vzpomene, že k němu potřebuje speciální kabel, a objedná si jej v samostatné objednávce. V pojetí transakcí jako objednávek se tato informace ztratí.

Ve spolupráci se společností MSPS s r.o. jsem proto navrhl rozšíření o *časovou složku*. Ta spočívá v následujících dvou změnách v pojetí dat:

- Souhrn veškerého zboží objednaného jedním zákazníkem se považuje za jednu transakci.
- Každá asociace mezi položkou a transakcí obsahuje časy objednání, které jsou rozhodující pro určení vazby obou položek na sebe. Čas objednání $time_A(i)$ udává všechny časy objednání položky i v transakci A.

Pro pravidlo $i \Rightarrow j$ je pak možné určit *nejmenší časové rozpětí* ve dnech v rámci jedné transakce

$$mintimespan_A(i \Rightarrow j) = \min_{t_i \in time_A(i), t_j \in time_A(j)} |t_i - t_j|$$

a na základě ní *průměrné nejmenší časové rozpětí* ve dnech

$$avgtimespan(i \Rightarrow j) = \sum_{A \in T} \frac{mintimestamp_A(i \Rightarrow j)}{f(\{i, j\})}$$

kde T je množina transakcí obsahujících i a j zároveň.

Je možné předpokládat, že čím je interval mezi objednáním dvou položek jedním uživatelem kratší, tím silnější je vzájemný vztah těchto dvou položek. Aby bylo možné kvantifikovat tento vztah, je třeba zavést matematickou funkci, která bude reprezentovat *koeficient časového rozpětí*. Tato funkce bude nadále označována jako TSC (time span coeficient). Platí $TSC(x) \in R$, kde x je uvažovaný počet dnů mezi oběma nákupy, $x \in Z, x \geq 0$.

Dále tato funkce musí splňovat následující podmínky:

1. $TSC(x) \in (p, 1\rangle$: hodnota $p = 1 - tscWeight$, kde tscWeight je parametr algoritmu v rozmezí $(0, 1\rangle$, určující poměrnou „důležitost“ časové složky při výpočtu
2. $TSC(0) = 1$: pro položky objednané v ten samý den je koeficient maximální
3. $\lim_{x \to \infty} TSC(x) = p$: jak se časové rozpětí blíží nekonečnu, koeficient se blíží své minimální hodnotě

Analytici společnosti oXy Online, která Czechcomputer.cz vyvíjí, navrhli další omezení, která dále vymezují průběh funkce:

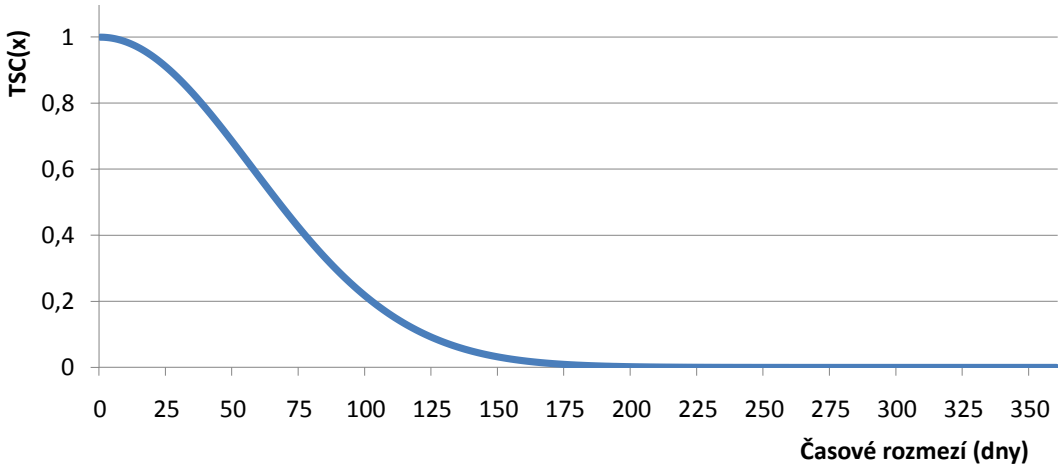
1. Dvojice produktů objednaných do 7 dnů od sebe by měla mít koeficient nejméně 99% (intuitivně je vzájemná vazba zboží objednaných v jednom týdnu od sebe téměř stejně významná, jako kdyby byli ve společné objednávce)
2. Dvojice produktů objednaných více než 100 dnů od sebe by měla mít koeficient nejvýše 20%

Na základě těchto požadavků byla zvolena funkce:

$$TSC(x) = p + e^{-6500x^2 - ln(1-p)}$$

vycházející z obecné Gaussovy funkce

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$



Obrázek 3.2: Průběh funkce TSC(x), p = 0

Tato funkce je použita v následující části pro výpočet celkové váhy pravidla.

3.2.3 Váha pravidla

Pro kvantifikaci celkové „zajímavosti“ pravidla je třeba definovat novou proměnnou, kterou jsem nazval *váha pravidla*. Podle váhy pravidla bude možné pravidla porovnávat a řadit, což umožní v elektronickém obchodu zobrazovat doporučení od nejvíce relevantního po nejméně relevantní.

Pro určení váhy pravidla jsem se po konzultací se společností MSPS a pročtením dokumentace firmy Amazon rozhodl použít index podobnosti popsaný v části 2.6.1. Pro váhu je také důležitý koeficient časového rozpětí uvažující průměrné nejmenší časové rozpětí. Pro účely implementace budu tedy za váhu považovat součin těchto dvou hodnot.

$$w(i \Rightarrow j) = CI(i \Rightarrow j) \cdot TSC(avgtimespan(i \Rightarrow j))$$

Tento způsob výpočtu váhy nemusí být ideální a může být potřeba ho na základě zkušeností z reálného provozu změnit, například za použití spolehlivosti nebo liftu. Implementace bude s touto variantou počítat a bude umožňovat jednoduchou výměnu výpočtu váhy.

3.2.4 Kombinované doporučení

Kromě doporučování na základě jednoho produktu je nutné implementovat i kombinované doporučování, na základě množiny produktů (pro stránku nákupního košíku). Proto je nutné definovat způsob, jakým se budou jednotlivá asociační pravidla kombinovat, pokud je dotazem množina produktů a ne pouze jeden produkt.

Mějme vstupní množinu produktů $A = \{i_1, i_2, \dots i_n\}$, pro kterou chceme generovat doporučené produkty. Formálně vzato je cílem z množiny známých pravidel

$$\begin{gathered} i_1 \Rightarrow j_1 \ (w_1) \\ i_2 \Rightarrow j_2 \ (w_2) \\ \vdots \\ i_n \Rightarrow j_n \ (w_n) \end{gathered}$$

s vahami $w_1, w_2, \dots w_n$ sestavit novou množinu pravidel

$$\begin{gathered} \{i_1, i_2, \dots i_n\} \Rightarrow k_1(w'_1) \\ \{i_1, i_2, \dots i_n\} \Rightarrow k_2(w'_2) \\ \vdots \\ \{i_1, i_2, \dots i_n\} \Rightarrow k_m(w'_m) \end{gathered}$$

Pochopitelně je důležité správně stanovit váhy výsledných kombinovaných pravidel. Pro účely této práce jsem zvolil jednoduchý součet vah pravidel normalizovaný podle součtu vah všech uvažovaných pravidel. Tedy pro každý *unikátní* produkt k se vytvoří právě jedno pravidlo $\{i_1, i_2, \dots i_n\} \Rightarrow k$. Váha tohoto pravidla bude

$$w(\{i_1, i_2, \dots i_n\} \Rightarrow k) = \frac{\sum\{w(y \Rightarrow k) \mid y \in A\}}{\sum\{w_1, w_2, \dots w_n\}}$$

Příklad Řekněme, že potřebujeme generovat kombinované doporučení pro množinu produktů $\{mleko, chleb, rohlik\}$ a známe pro ně tato pravidla:

$$\begin{array}{rcll} mleko & \Rightarrow & noviny & (w = 0,7) \\ chleb & \Rightarrow & pivo & (w = 0,5) \\ chleb & \Rightarrow & noviny & (w = 0,2) \end{array}$$

Výsledné kombinované pravidla pak budou

$$\begin{array}{rcll} \{mleko, chleb, rohlik\} & \Rightarrow & noviny & (w = (0,7 + 0,2)/1,4 \doteq 0,643) \\ \{mleko, chleb, rohlik\} & \Rightarrow & pivo & (w = 0,5/1,4 \doteq 0,357) \end{array}$$

3.3 Dvě varianty implementace

Pro implementaci je třeba uvažovat řadu kritérií, která jsou důležitá pro provoz v reálném prostředí. Jsou to zejména tato kritéria:

- *Rychlost výpočtu pravidel* - elektronický obchod vyžaduje, aby data pro doporučování produktů byla generována často. Aby to bylo umožněno, musí být výpočet dostatečně rychlý.
- *Rychlost vyhledávání pravidel* - protože k doporučování musí docházet v reálném čase přímo na internetových stránkách, je nutné, aby vyhledání pravidla byla záležitost maximálně desítek milisekund při nízké zátěži serveru. Tento požadavek vychází ze specifikace aplikace, která určuje maximální akceptovatelnou dobu pro zpracování jednoho HTTP požadavku při nízké zátěži.

- *Udržovatelnost* - modul pro výpočet asociačních pravidel bude dlouhodobě udržovaný dalšími programátory, proto musí splňovat určité minimální standardy návrhu, dokumentace a kvality kódu. Zároveň je také z tohoto důvodu nutné pokud možno omezit použití složitých nástrojů nebo knihoven vyžadujících znalost problematiky dolování dat.
- *Škálovatelnost výpočtu* - výpočet pravidel by měl dobře škálovat i při řádově větším množství pravidel, ovšem není to absolutní požadavek.

Po zvážení všech těchto kritérií jsem se rozhodl pro dvě varianty implementace. Která z těchto implementací bude vhodnější pro nasazení, ukáže testování, které je předmětem další kapitoly.

Paměťová implementace spočívá v realizaci algoritmu Apriori v prostředí aplikačního serveru, jak ukazuje obrázek 3.3. Algoritmus bude napsán v jazyce Java a bude využívat data kompletně načtená do operační paměti stroje, kde jsou pravidla uložena po dobu běhu aplikačního serveru. Výhoda tohoto přístupu je očekávaná vyšší rychlost při generování i vyhledávání pravidel. Nevýhoda je obtížnější implementace a paměťové nároky (bude třeba při generování všechna vstupní data načíst do paměti). Dále je třeba pravidla na každém z aplikačních serverů generovat zvlášť nebo zajistit jejich sdílení pomocí specializovaného nástroje jako Terracotta[1] nebo Hazelcast[2].

Databázová implementace principem je převedení algoritmu do jazyka SQL a využití konstruktů relační databáze pro uložení asociačních pravidel, jak ukazuje obrázek 3.4. Použitelnost tohoto postupu ukazuje [8, 62 s.]. Výhodou tohoto principu je vynechání potenciálně náročné fáze načítání dat do paměti. Je také zajištěno, že všechny aplikační servery budou za každých okolností používat ta samá asociační pravidla. Nevýhodou je předpokládaná nižší rychlost generování a vyhledávání pravidel

V rámci této práce jsou implementoval obě varianty. V kapitole 4 je pak uvedeno jejich zhodnocení.

3.4 Předzpracování dat

V této části popíšu, jaké předzpracování dat bude třeba v průběhu generování doporučení.

3.4.1 Čištění dat

Všechna obchodní data, která se používají jako vstup pro generování pravidel, by měla být z principu správná, konzistentní a kompletní. Proto není třeba provádět žádné explicitní čištění.

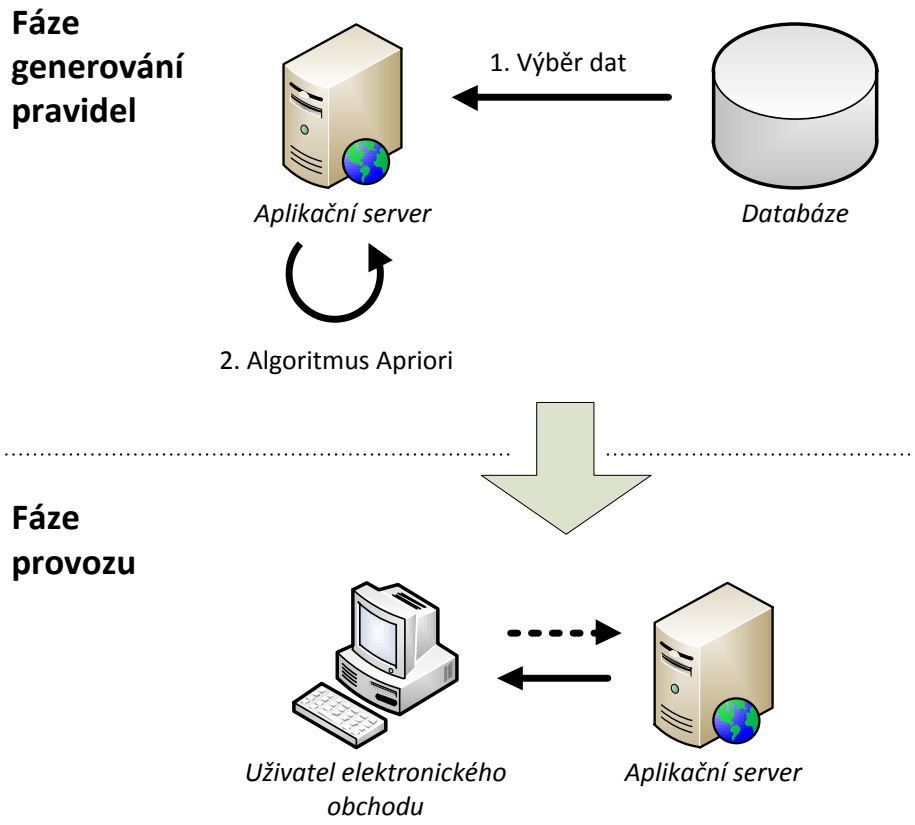
3.4.2 Integrace dat

Data o objednávkách jsou uložena ve specializovaném ERP [3] systému, který je používaný ve společnosti Czechcomputer. Internetový obchod periodicky replikuje určitou část těchto

[1] http://www.terracotta.org/

[2] http://www.hazelcast.com/

[3] Enterprise Resource Planning, informační systém pro automatizaci a řízení procesů firem týkajících se produkce



Obrázek 3.3: Ilustrace paměťové implementace

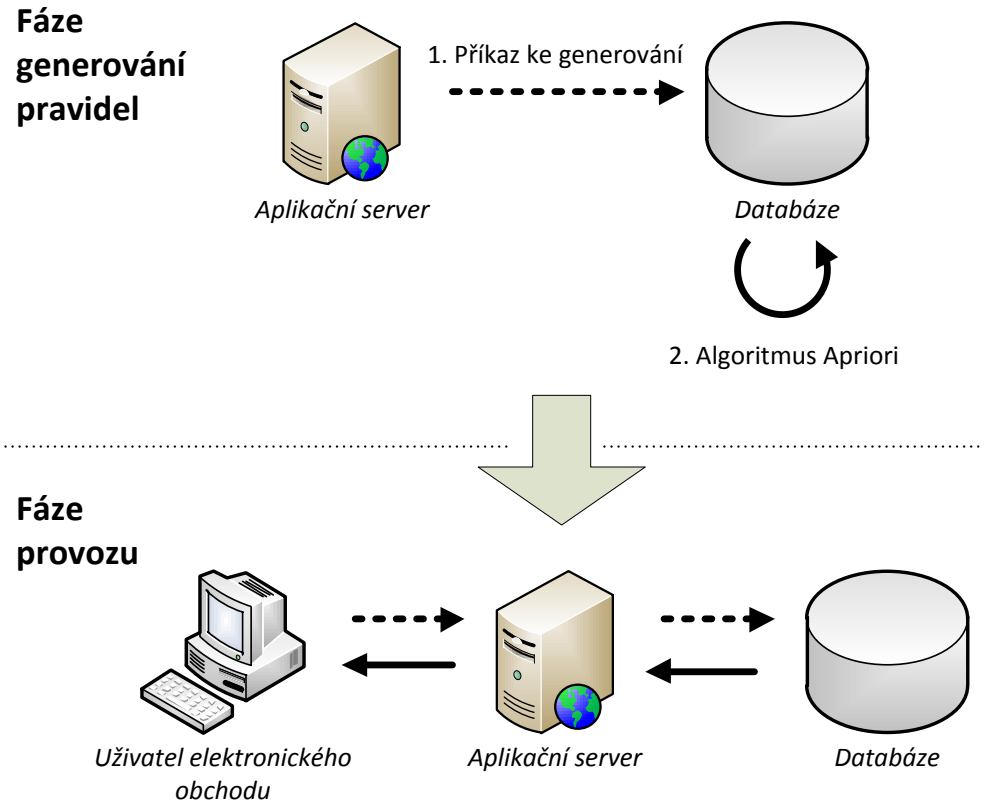
dat do své databáze. Diskuze k této replikaci je nad rámec této diplomové práce, nicméně z hlediska získávání znalostí z databáze je možné ji považovat za integraci dat.

3.4.3 Redukce dat

Pro výpočet je třeba vybírat pouze relevantní data. Redukce se týká dvou oblastí dat:

1. Databáze obsahuje velké množství produktů, ale pouze zlomek z nich jsou prodávané produkty. Zbytek tvoří ty, které byly staženy z prodeje, ty které ještě nebyly zveřejněny nebo ty, které vznikly omylem. Neprodávané produkty tak nemá smysl zahrnovat do výpočtu doporučení, protože:
   (a) jako cíl doporučení nemá neprodávaný produkt z obchodního hlediska smysl
   (b) jako zdroj doporučení je neprodávaný produkt nezajímavý, protože neprodávané produkty si zobrazuje minimum zákazníků
2. Data objednávek zahrnují také tzv. „drobný prodej“. Jedná se o zboží prodané přímo v prodejně bez předchozí objednávky. Toto zboží tak nezahrnuje informaci o uživateli, který si zboží koupil. Proto jsou tyto transakce vyloučeny z výpočtu.

Pro účely této diplomové práce jsem připravil datové sady, jejich použití je popsáno v kapitole 4. Tyto datové sady jsou anonymizované a předem redukované podle výše zmíněných kritérií.



Obrázek 3.4: Ilustrace databázové implementace

3.5 Vymezení pojmů pro účely této implementace

Pro účely návrhu a implementace této práce zde definuji některé nové pojmy a některé obecné pojmy z oblasti dolování asociačních pravidel rozšířím.

3.5.1 Transakce

Transakce se skládá z unikátního identifikátoru transakce a množiny položek, které transakce obsahuje. Transakce se obvykle v kontextu internetového obchodování rovná jedné účetní objednávce, tedy každá objednávka vytvořená uživatelem se považuje automaticky za samostatnou transakci. V rámci této práce jsem za vhodnější považoval transakci definovat jako shrnutí objednaného zboží jedním uživatelem, jak je vysvětleno v části 3.2.2. Každá transakce také obsahuje informaci o datu objednání každé své položky.

3.5.2 Položka

Položka je jeden produkt v elektronickém obchodu, jak byl popsán v kapitole 2.5. Položka má svůj unikátní identifikátor.

3.5.3 Transakční prostor

Transakční prostor je množina všech položek, které systém obsahuje a množina všech transakcí, které systém historicky vytvořil z daných položek.

3.5.4 Asociační pravidlo

Obecné asociační pravidlo bylo popsáno v části 2.5. Asociační pravidlo má několik kvalitativních atributů:

- *frekvence těla* - počet transakcí, ve kterých se objevuje tělo pravidla
- *frekvence hlavy* - počet transakcí, ve kterých se objevuje hlava pravidla
- *frekvence dvojice* - počet transakcí, ve kterých se objevuje tělo pravidla. Jinými slovy počet transakcí, které obsahují zároveň obě položky z tohoto pravidla
- atributy spolehlivost, lift a CI definované z předchozí kapitoly

3.5.5 Doporučení

Doporučení je abstraktní pohled na asociační pravidlo, jedná se o doporučení jednoho konkrétního produktu. Doporučovací systém jej vrací na základě dotazu, který obsahuje identifikátor jednoho nebo více produktů, pro které se má vyhledat doporučení. Doporučení se skládá z ID doporučeného produktu a vypočtené *váhy*. Váha doporučení se rovná váze asociačního pravidla, jak byla popsána v 3.2.3.

3.5.6 Parametry algoritmu

Algoritmus pro výpočet asociačních pravidel je možné ovlivnit následujícími parametry:

- *minimální frekvence těla* - v kolika transakcích se musí položka vyskytnout, aby byla vzata v úvahu při výpočtu
- *minimální frekvence dvojice* - v kolika transakcích se musí obě položky pravidla vyskytnout zároveň, aby bylo pravidlo akceptováno
- *minimální spolehlivost* - minimální spolehlivost pravidla, aby bylo pravidlo akceptováno
- *minimální lift* - minimální lift pravidla, aby bylo pravidlo akceptováno
- *důležitost časové složky* definovaná v 3.2.2 - jakou váhu má algoritmus přikládat koeficientu časového rozpětí. Tento parametr je reálné číslo od 0 do 1

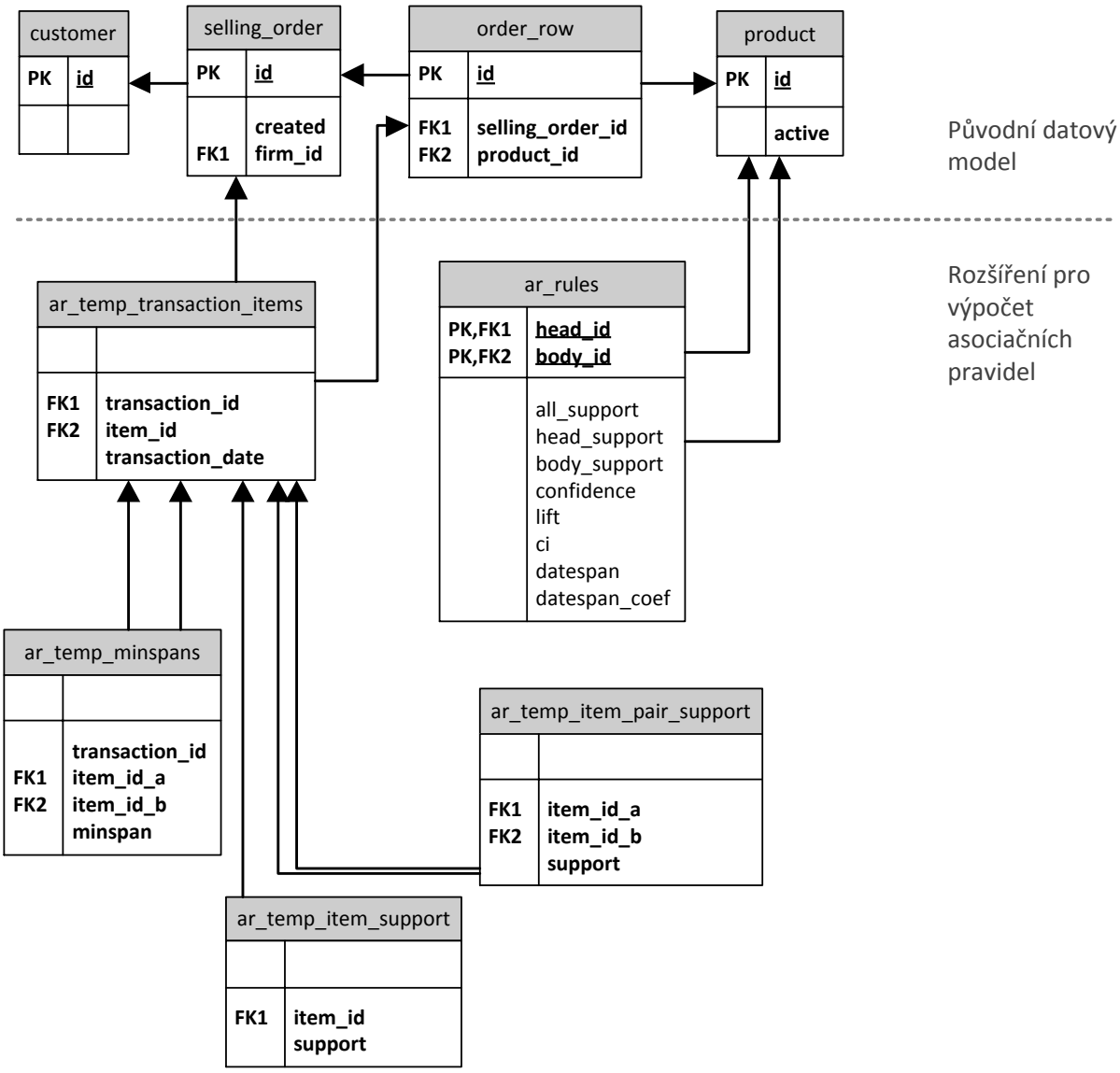
Cílem parametrů stanovujících minimální frekvenci je zabránit pravidlům, která vznikla náhodně v malém počtu objednávek a nemusí tak nutně odpovídat reálnému trendu u zákazníků. U zbývajících parametrů jde zejména o zajištění dostatečné kvality asociačních pravidel.

3.6 Návrh databázové implementace

Implementace v databázovém systému je v tomto případě realizována v čistém SQL. Veškeré výpočty asociačních pravidel probíhají přímo na databázovém serveru. Pravidla jsou zde na konci výpočtu uložena v tabulce. Pro účely výpočtu jsem vytvořil také několik dočasných tabulek, které obsahují data mezivýpočtů.

3.6.1 Databázové schéma

Diagram 3.5 znázorňuje zjednodušený databázový model současné aplikace Czechcomputer.cz a nově zavedené tabulky, sloužící k uložení dat doporučovacího systému.



Obrázek 3.5: Databázový diagram implementace (data, která se netýkají této práce, jsem pro jednoduchost vynechal)

Z původního databázového návrhu Czechcomputer byly vybrány čtyři tabulky, které se generování týkají:

`customer` obsahuje zákazníky, kteří si objednávají zboží

`selling_order` obsahuje objednávky těchto zákazníků

`order_row` obsahuje řádky objednávky, které odkazují na jednotlivé produkty objednané v dané objednávce

`product` obsahuje všechny produkty

Zavedl jsem 5 nových tabulek, z toho 4 dočasné a jednu pro trvalé uložení vygenerovaných asociačních pravidel.

`ar_temp_transaction_items` obsahuje všechny vazby mezi položkami a transakcemi s časovou složkou obsahující datum objednání

`ar_temp_minspans` má nejmenší časovou vzdálenost ve dnech pro každé dvě položky v každé z transakcí. Tato hodnota se používá jako parametr pro funkci TSC.

`ar_temp_pair_support` má hodnotu frekvence pro každou z dvojic položek

`ar_temp_item_support` obsahuje hodnotu frekvence pro každou položku

`ar_rules` obsahuje finální podobu asociačních pravidel, včetně všech parametrů každého asociačního pravidla (tedy frekvence, podpora, spolehlivost, CI).

3.6.2 Průběh výpočtu

Výpočet probíhá za použití SQL příkazů v těchto krocích:

1. Zpracování dat
   (a) Vytvoření a naplnění tabulky položek `ar_temp_transaction_items`. To je provedeno prostým výběrem z tabulek produktů (pro určení ID položky), firem (pro určení ID transakce) a objednávek (pro určení času objednání).
   (b) Vymazání dosavadního obsahu tabulky `ar_rules`.
2. Výpočet asociačních pravidel
   (a) Vytvoření a naplnění tabulky `ar_temp_minspans`. Je proveden kartézský součin položek v každé objednávce a výběr nejmenší absolutní hodnoty z rozdílu datumů.
   (b) Vytvoření a naplnění tabulky `ar_temp_item_support`. To je realizováno výběrem z tabulky. `ar_temp_transaction_items` obsahujícím počet unikátních ID transakcí každé položky
   (c) Vytvoření a naplnění tabulky `ar_temp_pair_support`. To je realizováno výběrem z tabulky. `ar_temp_minspans` obsahujícím počet unikátních ID transakcí každé dvojice
   (d) Naplnění tabulky `ar_rules` asociačními pravidly s frekvence a spolehlivosti. Toho je dosaženo výběrem z `ar_temp_pair_support`, `ar_temp_item_support` a `ar_temp_minspans`.
   (e) Doplnění tabulky `ar_rules` o další parametry, které je možné vypočítat dodatečně. Parametry `lift`, `CI`, `datespan` (časové rozpětí) a `datespan_coef` (koeficient časového rozpětí) jsou vypočteny podle odpovídajících vzorců.
3. Odstranění všech dočasných tabulek.

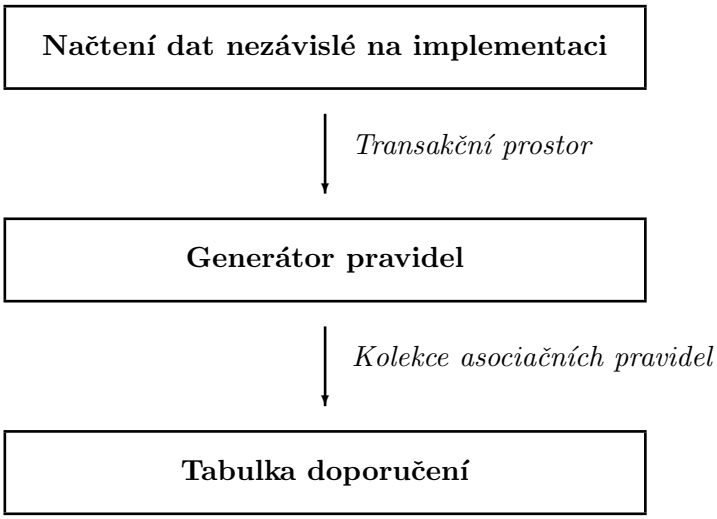
Na konci výpočtu obsahuje tabulka `ar_rules` vypočtená asociační pravidla a všechny jejich parametry.

3.7 Návrh paměťové implementace

V této variantě je systém pro doporučování navržen jako samostatný modul. Ten na pokyn periodického plánovače spouští generování asociačních pravidel, které pak převádí na jednotlivá doporučení. Vstupy a výstupy této knihovny je možné napojit prakticky na libovolný systém, který pracuje s položkami a transakcemi (objednávkami). Knihovna není přímo napojena na konkrétní implementaci internetového obchodu Czechcomputer. Poskytuje pouze obecné třídy a rozhraní.

3.7.1 Proces generování doporučení

Proces generování doporučení produktů se bude skládat ze tří částí: načtení vstupních dat, generování asociačních pravidel a vytvoření doporučení na základě vybraných položek. To znázorňuje obrázek 3.6.



Obrázek 3.6: Proces generování asociačních pravidel

Transakční prostor je vytvořen klientským kódem v závislosti na implementaci. Může jít o načtení dat z relační databáze, XML souboru nebo jakéhokoliv jiného zdroje. Poté klientský kód předá vytvořený kód generátoru pravidel. Ten v závislosti na implementaci a zvolených parametrech vygeneruje kolekci asociačních pravidel. Ta jsou následně předána v konstruktoru tabulce doporučení, která slouží jako paměťově úsporná paměťová struktura. Aplikace ji může opakovaně používat z libovolného počtu vláken.

3.7.2 Návrh jednotlivých tříd

Návrh počítá s maximální možnou rozšiřitelností. Vstupní data pro generování doporučení jsou knihovně předložena ve struktuře nezávislé na zdroji dat. Diagram tříd je znázorněn na obrázku 3.7.

Knihovna se skládá z následujících balíků:

cz.vutbr.fit.recommend – Obsahuje základní třídy a rozhraní pro využívání knihovny.

cz.vutbr.fit.recommend.data – Obsahuje třídy pro import transakčního prostoru z datových zdrojů.

cz.vutbr.fit.recommend.apriori – Obsahuje samotné paměťové implementace algoritmu Apriori.

Třída cz.vutbr.fit.recommend.AssociationRule

Představuje jedno vygenerované asociační pravidlo. Tato třída je neměnitelná a tedy bezpečná pro přístup z více vláken.

Třída cz.vutbr.fit.recommend.AssociationRuleGenerator

Obecné rozhraní pro každý generátor asociačních pravidel s metodou `generateRules()` která na základě daného transakčního prostoru vygeneruje kolekci asociačních pravidel.

Třída cz.vutbr.fit.recommend.AssociationRuleGeneratorListener

Rozhraní pro sledování postupu generování pravidel pomocí zpětného volání metody `onGenerationProgress()` na tomto rozhraní. Může být použito například pro vizualizaci postupu v uživatelském rozhraní pomocí indikátoru průběhu.

Třída cz.vutbr.fit.recommend.Recommendation

Doporučení konkrétního produktu obsahující vypočítanou váhu doporučení. Třída je neměnitelná a tedy bezpečná pro přístup z více vláken.

Třída cz.vutbr.fit.recommend.RecommendationTable

Datová struktura, která obsahuje jednotlivé doporučení pro produkty. Vzhledem ke zvýšeným nárokům na operační paměť umožňuje omezit počet doporučení na prvních N doporučení podle váhy. Třída je neměnitelná a tedy bezpečná pro přístup z více vláken, takže je možné její instanci přímo použít ve vícevláknovém prostředí webové aplikace.

Třída cz.vutbr.fit.recommend.data.TransactionItem

Položka v systému obsahující pouze identifikátor položky.

Třída cz.vutbr.fit.recommend.data.Transaction

Transakce v systému. Obsahuje libovolné množství položek a udržuje informaci o času objednání jednotlivých položek.

Třída cz.vutbr.fit.recommend.data.TransactionSpace

Transakční prostor, jak je popsán v 3.5.3

Třída cz.vutbr.fit.recommend.data.TransactionSpaceBuilder

pomocná třída, která umožňuje jednodušší vytváření transakčního prostoru. Klientský kód volá opakovaně metodu `addAssociation()` pro každou vazbu mezi transakcí a položkou. Nakonec voláním `build()` vygeneruje transakční prostor.

Třída cz.vutbr.fit.recommend.apriori.AprioriGenerator

Tato třída je základní generátor pravidel pomocí algoritmu Apriori. Umožňuje konfiguraci pomocí dvou parametrů algoritmu:

minFrequency určuje, jakou minimální frekvenci musí pravidlo mít, aby bylo zařazeno ve výsledcích

minConfidence určuje, jakou minimální spolehlivost musí pravidlo mít, aby bylo zařazen ve výsledcích

AprioriGenerator je první implementace, která počítá pouze se základní možností úprav konfigurace a nezohledňuje časové vzdálenosti mezi položkami.

Třída cz.vutbr.fit.recommend.apriori.TimeSpanAprioriGenerator

Pokročilejší generátor pravidel pomocí Apriori. Umožňuje nastavení řady parametrů algoritmu:

minBodyFrequency – celočíselný parametr, který určuje, s jakou nejmenší hodnotou frekvence těla bude ještě pravidlo zařazeno ve výsledcích.

minConfidence – reálný parametr, který určuje, s jakou nejmenší hodnotou confidence bude ještě pravidlo zařazeno ve výsledcích.

timeSpanWeight – reálný parametr, který určuje, důležitost koeficientu časového rozpětí, který byl definován v části 3.2.2.

minLift – celočíselný parametr. Určuje minimální lift pravidla, aby bylo zařazeno ve výsledcích.

minAllFrequency – celočíselný parametr určující minimální frekvenci, aby bylo pravidlo zařazeno ve výsledcích.

maxRulesPerProduct – při zpracování velkých objemů dat může být třeba omezit množství vygenerovaných pravidel kvůli paměťovým nárokům. Tento celočíselný parametr umožňuje zvolit, kolik maximálně pravidel na jeden produkt se má zachovat. Zachová se vždy N produktů s nejvyšší vahou.

3.7.3 Příklad použití

Program 3 ukazuje, jak se bude programátorské rozhraní využívat. Jedná se o typické použití nadstavby v pseudokódu:

3.7.4 Automatické testy korektnosti

Pro účely paměťové implementace výpočtu asociačních pravidel jsem se rozhodl vytvořit automatické testy, které kontrolují korektní fungování implementace. Tyto testy jsem používal během vývoje pro včasné odchycení zanesených chyb a regresí. Společnost MSPS s r.o. připravila vstupní data a referenční výstupy, které je možné použít pro kontrolu, zda algoritmus Apriori pracuje správně.

Automatické testy používají knihovnu pro jednotkové testování JUnit[4].

[4] http://junit.org/

Program 3 Ukázka načtení transakčního prostoru, vytvoření tabulky doporučení a její použití

```
class TestAssociationRules {
  RecomendationTable table;
  ...

  void fazeGenerovani() {
    TransactionSpaceBuilder tsb = new TransactionSpaceBuilder()
    while (hasMoreAssociations()) {
      tsb.addAssociation(...);
    }
    TransactionSpace space = tsb.build()

    AssociationRuleGenerator generator = new TimeSpanAprioriGenerator(...)
    Collection<AssociationRule> rules = generator.generateRules(space)
    this.table = new RecommendationTable(rules, ...)
  }
  ...

  void fazePouziti() {
    table.recommendForProduct(productId);
    ...
  }
}
```

Třídy testů

AssociationRuleTestUtil Tato třída obsahuje statickou metodu `test(String inputFile, String referenceOutputFile, AssociationRuleGenerator generator)`, kterou je možné použít z jednotlivých testů. Tato metoda:

1. načte vstupní data v daném souboru na disku
2. vygeneruje na jejich základě asociační pravidla pomocí daného generátoru
3. porovná výsledek s referenčními daty v jiném souboru na disku. Pokud výsledky nesouhlasí, vypíše nesouhlasící řádky na chybový výstup a vygeneruje selhání testu (`Assert.fail()`).

AprioriGeneratorTest JUnit test na fungování třídy `AprioriGenerator`.

TimeSpanAprioriGeneratorTest JUnit test na fungování třídy `TimeSpanAprioriGenerator`.

RecommendationTableTest Tato třída je JUnit test, který testuje správnost generování doporučení pro jednotlivé položky (`recommendForStockItem()`) i celý košík (`recommendForBasketTest()`)

Výsledky testů

Žádný z testů nehlásí při spuštění chybu. Výsledky testů je možné kdykoliv ověřit. Způsob spouštění těchto testů je popsán v příloze.

3.7.5 Poznámky k paměťové implementaci

Během realizace paměťové implementace jsem řešil problémy způsobené velkými objemy zpracovávaných dat, zejména týkajících se paměťové a časové náročnosti. Ty částečně nebo úplně znemožňovaly použití nadstavby s většími datovými sadami. Při jejich řešení jsem použil nástroj YourKit Java Profiler,[5] který je určen pro analýzu výkonu aplikací na platformě Java. Postupným vyhledáváním úzkých míst se podařilo omezit nebo eliminovat většinu problému a paměťovou implementaci úspěšné realizovat.

Výkon paměťové implementace

Paměťová implementace generování asociačních pravidel už nyní vykazuje přijatelné výsledky co do rychlosti. To ukazuje následující kapitola 4. Přesto jsem se rozhodl provést analýzu výkonu pro další optimalizace. Výkon při generování pravidel je pro provoz kritický, proto by další zrychlování implementace bylo žádané i přesto, že počáteční podmínky už současná implementace splňuje. Další optimalizace této implementace jsem už v rámci této práce neprováděl, nicméně výstup z této analýzy může být dobrým počátečním bodem, kde s optimalizací začít.

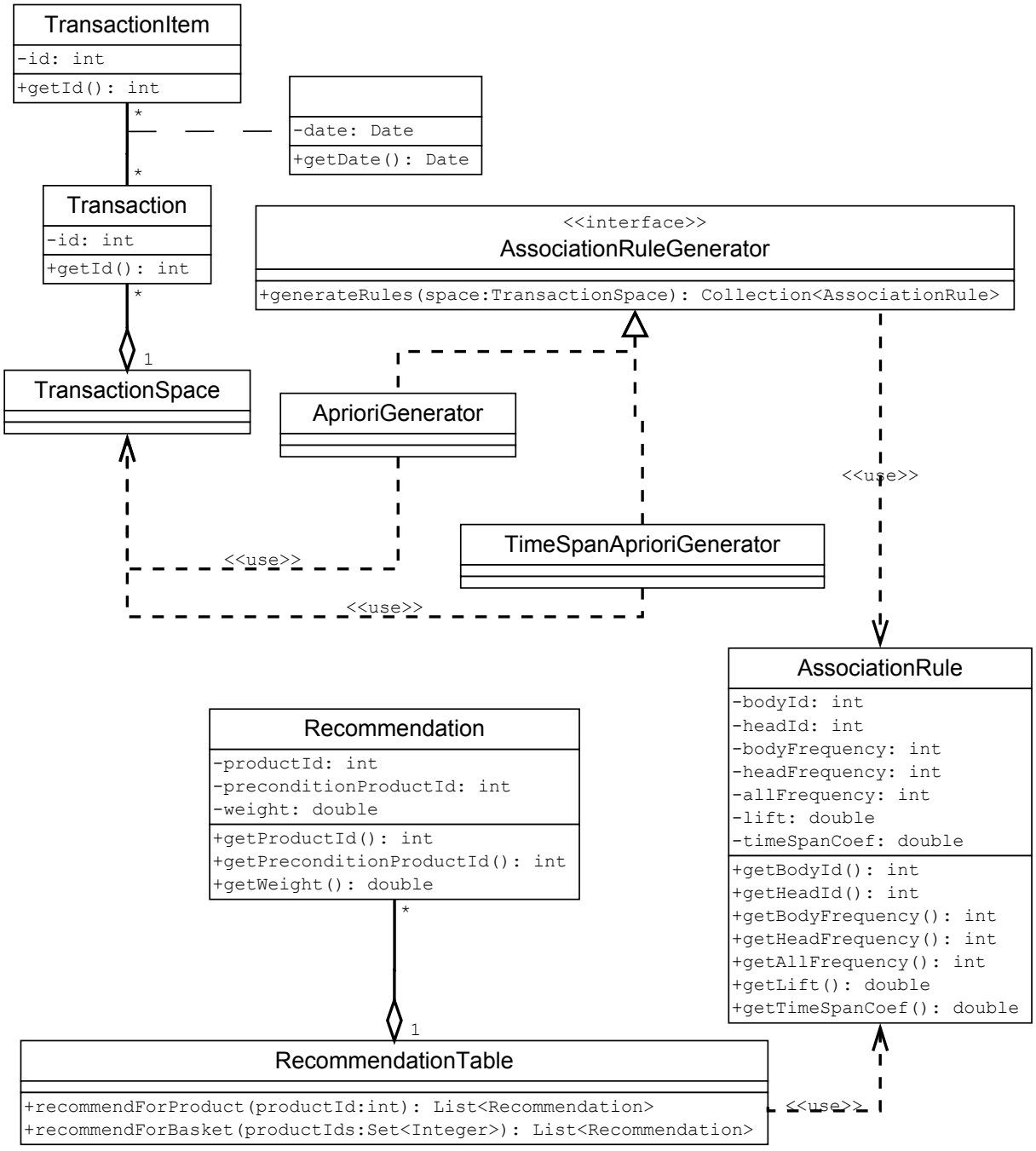
Testoval jsem generování pravidel při datové sadě obsahující objednávky za 90 dnů. Zaměřil jsem se na třídu `TimeSpanAprioriGenerator`, která obsahuje logiku pro generování asociačních pravidel.

Výsledky znázorněné na obrázku 3.8 naznačují, že významnou část procesorového času (59%) zabírá metoda `getTimeSpanCoef()`, která zjišťuje koeficient časové složky (TSC) pro daný pár položek. V současné implementaci tato metoda prochází všechny umístění ve všech transakci, které jsou pro obě položky společná, a hledá dva takové časy, jejichž rozdíl je nejnižší. Potom tyto časy zprůměruje přes všechny transakce.

Tato část by se pravděpodobně dala zefektivnit předpočítáním některých hodnot, popřípadě ukládáním už jednou spočítané koeficientu do paměti (bylo by třeba mít se na pozoru před kvadratickou paměťovou složitostí).

Další podněty k optimalizaci by pak mohly být například urychlení metody `hashCode()` na některých třídách často vkládaných do hashovacích tabulek, jako jsou `TransactionItem` a `TransactionItemPair`). Dále pak také zrychlení matematického výpočtu hodnoty funkce TSC.

[5] http://www.yourkit.com/



Obrázek 3.7: Diagram tříd

Name	Time (ms)		Own Time (ms)
TimeSpanAprioriGenerator.generateRules()	25 381	100 %	951
TimeSpanAprioriGenerator.getTimeSpanCoeficient()	15 038	59 %	296
java.lang.**Math.exp**()	9 687	38 %	0
Transaction.getDaySpan()	2 418	10 %	780
java.util.**HashSet.iterator**()	670	3 %	218
java.util.**HashMap$KeyIterator.next**()	546	2 %	265
java.util.**Date.getTime**()	312	1 %	312
java.util.**HashMap.get**()	109	0 %	46
TransactionItem.hashCode()	62	0 %	62
java.util.**HashSet.contains**(Object)	2 028	8 %	15
TransactionItem.hashCode()	1 045	4 %	1 045
java.util.**HashMap$KeyIterator.next**()	592	2 %	280
java.util.**HashSet.iterator**()	15	0 %	15
TransactionItemPair.hashCode()	1 638	6 %	265
AssociationRuleComparatorByWeight.compare()	1 060	4 %	0
Main2$1.onGenerationProgress()	826	3 %	0
TransactionItem.getSupport()	390	2 %	390
TransactionItemPair.equals()	374	1 %	358
ch.qos.logback.classic.**Logger.debug**()	187	1 %	0

Obrázek 3.8: Výstup z YourKit Java Profileru

Kapitola 4

Testování a výkon

V této kapitole se zabývám testováním obou implementací podle různých kritérií na několika sadách reálných obchodních dat a zhodnocením těchto testů. Zaměřuji se zejména na testování výkonnosti a paměťové náročnosti. Jsou to zásadní kritéria pro možnost nasazení do produkčního prostředí.

4.1 Použité parametry pro algoritmus Apriori

Pro účely testování bylo nutné zvolit parametry pro algoritmus Apriori tak, aby se pokud možno blížily parametrům nastaveným při reálném provozu.

- *minimální frekvence těla* = 30 a *minimální frekvence dvojice* = 3. Tyto hodnota vychází z patentové dokumentace společnosti Amazon. [4]
- *minimální spolehlivost* = $0,05$ tato hodnota byla zvolena experimentálně na základě vygenerovaných pravidel a pokrytí produktů těmito pravidly.
- *důležitost časové složky* byla zvolena jako $0,5$

4.2 Vzorky dat

Pro účely testování systému doporučování na základě košíku jsou k dispozici reálná data z objednávek na serveru Czechcomputer.cz. Z dostupných dat uvedených v části 1.4 jsem vytvořil 6 sad vstupních dat, na kterých jsem prováděl testování. Tabulka 4.1 ukazuje metriky vstupních dat podle časového rozpětí každé datové sady. Metriky zahrnují počet položek, počet transakcí a celkový počet vazeb mezi transakcemi a položkami.

Každá datová sada obsahuje reálná data z objednávek na serveru Czechcomputer.cz. Konec časového rozpětí je ve všech případech den 31.12.2010 a začátek odpovídá uvedenému počtu dnů před tímto datem. Vzorky vstupních dat jsou součástí této práce v anonymizované podobě.

V připravených datových sadách neroste počet vazeb lineárně s časovým rozsahem, jak by bylo intuitivní, ale pomaleji. Důvodem je, že čím dále do historie výběr dat sahá, tím větší procento položek tvoří neprodávatelné produkty, které se pro výpočet asociačních pravidel nezohledňují, jak je vysvětleno v 3.4.3. Průměrná doba, po kterou je produkt prodávatelný je méně než rok, takže rok staré transakce obsahují pouze určité procento položek, které jsou platné.

Metrika	Datová sada					
	7 dnů	**30 dnů**	**60 dnů**	**90 dnů**	**180 dnů**	**360 dnů**
Položek	3 454	7 958	9 397	10 061	10 571	10 956
Transakcí	3 182	23 483	35 262	43 180	51 245	61 264
Vazeb	7 829	65 584	111 210	146 208	183 110	228 325

Tabulka 4.1: Dostupné datové sady

4.3 Statistiky vygenerovaných pravidel

Pro každou z datových sad jsem nageneroval odpovídající pravidla. Počet vygenerovaných pravidel ukazuje tabulka 4.2. Je zde zároveň vidět, pro kolik produktů existuje alespoň jedno pravidlo (v řádku nazvaném *pokrytí produktů*), a jaký podíl tvoří tyto produkty k počtu všech produktů (*poměrné pokrytí*).

Metrika	Datová sada					
	7 dnů	**30 dnů**	**60 dnů**	**90 dnů**	**180 dnů**	**360 dnů**
Pravidel celkem	96	149 747	449 489	736 624	963 719	913 639
Pokrytí produktů	5	358	772	1 091	1 359	1 703
Poměrné pokrytí (%)	1,4	4,4	8,2	10,8	12,9	15,5

Tabulka 4.2: Statistiky vygenerovaných pravidel

Jak je vidět, počty pravidel opět nerostou lineárně. Dokonce počet pravidel v případě největší datové sady 360 klesl oproti poloviční datové sadě 180. Důvodem je nastavení algoritmu na minimální spolehlivost, které při vzrůstajícím počtu transakcí vyřazuje větší množství pravidel. Pravidla generovaná na větším vzorku mají totiž statisticky nižší úroveň spolehlivosti, kvůli rozmělnění položek souvisejícím se zvyšujícím se procentem neprodávatelných produktů ve starších datech.

4.4 Testování vzájemné ekvivalence implementací

Oba způsoby implementace algoritmu Apriori (databázová a paměťová) by měly být zcela ekvivalentní. Proto jsem testoval, zda generují stejná asociační pravidla. Testy jsem provedl na stejných datových sadách, které jsou popsány v tabulce 4.1 a ve všech případech byla množina vygenerovaná asociačních pravidel shodná pro obě implementace.

Tento test je možné kdykoliv zopakovat za použití ukázkového programu, který je součást této diplomové práce. Způsob jeho použití je popsán v příloze.

4.5 Výkonnostní test generování pravidel

Elektronický obchod vyžaduje, aby data pro doporučování produktů byla generována dostatečně často. Nové produkty by měly být zahrnuty do systému doporučování co nejdříve, aby se zvýšil jejich prodej. Proto je třeba generovat asociační pravidel nejméně jednou denně. Výkonnost je tedy jedno z hlavních kritérií pro výběr implementace. V následující části popíšu testování obou implementací (paměťové i databázové) z hlediska výkonu.

4.5.1 Metodika

Všechny testy byly měřeny na jednom stroji, který nebyl významně vytěžován žádným dalším běžícím procesem. Každé časové měření bylo provedeno pětkrát a výsledky byly zprůměrovány. Testovaná databáze byla Oracle 11g, verze 11.1.0.7.0.

Všechna časová měření obsahují celkový čas strávený výpočtem pravidel, včetně podpůrných operací. V případě paměťové implementace čas obsahuje i dobu strávenou načítáním pravidel z databáze. Testování probíhalo měřením času vybraných sekcí programu pomocí volání metody `System.currentTimeMilis()`, která je standardní součást Javy. Měření je tedy pouze tak přesné, jak umožňuje tato metoda.

Testy je možné kdykoliv zopakovat za použití ukázkového programu, který je součást této diplomové práce. Způsob jeho použití je popsán v příloze.

4.5.2 Výsledky a zhodnocení

Výsledky testů ukazuje tabulka 4.3.

Metrika	**Datová sada**					
	7 dnů	**30 dnů**	**60 dnů**	**90 dnů**	**180 dnů**	**360 dnů**
Doba generování podle implementace (ms)						
paměťová	747	7 766	17 435	26 797	35 106	41 422
databázová	3 413	141 850	469 748	737 744	996 108	913 639

Tabulka 4.3: Rychlost generování pravidel

Testování výkonu ukázalo, že paměťová implementace je řádově rychlejší než implementace databázová, a to v průměru zhruba 21krát. Tyto výsledky jsou vizualizovány v grafu 4.1

Dále je vidět, že v obou případech roste časová náročnost výpočtu zhruba lineárně v závislosti na počtu vazeb mezi transakcemi a položkami. Graf 4.2 ukazuje dobu generování normalizovanou do měřítka $\langle 0, 1 \rangle$, kde 1 reprezentuje nejdelší dobu generování pro danou implementaci. Tyto výsledky naznačují dobrou škálovatelnost obou implementací.

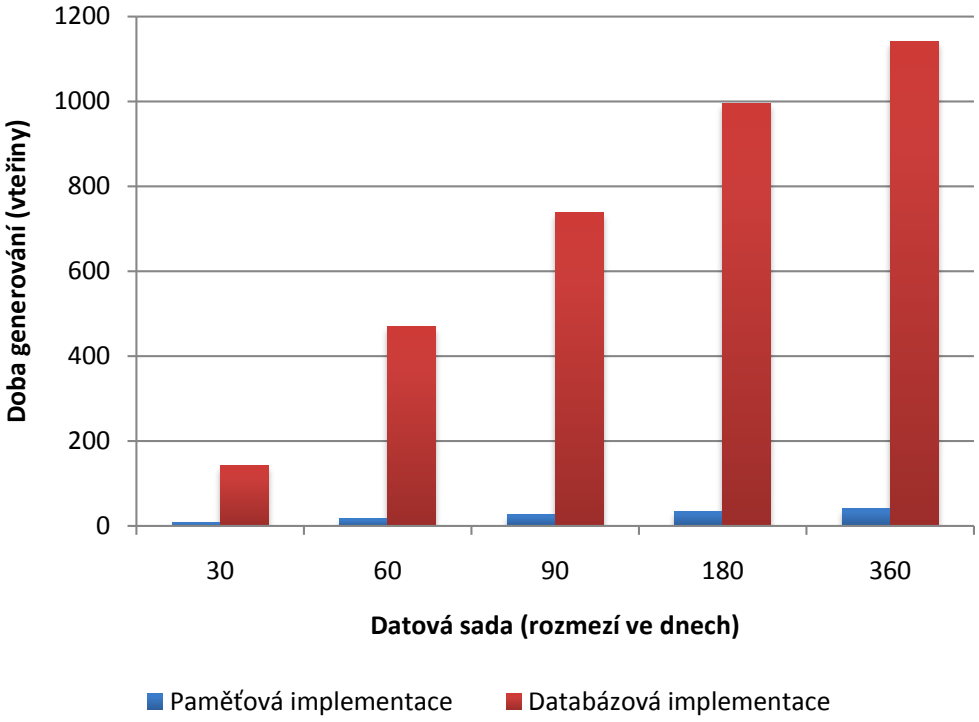
Obě implementace podle těchto výsledků splňují požadavky kladené na rychlost generování pravidel, i když paměťová implementace požadavky splňuje lépe.

4.6 Výkonnostní test vyhledávání pravidel

Kvůli požadavku na nízkou prodlevu při generování stránky je třeba, aby vyhledávání pravidel pro doporučení produktů trvalo minimální čas. V této části popíši testování výkonu při vyhledávání pravidel v obou implementacích.

4.6.1 Metodika

Bylo měřeno vyhledávání asociačních pravidel pro 10 000 náhodně zvolených produktů. Některé produkty mohly být při náhodné volbě zvoleny víckrát, některé vůbec. Opět bylo provedeno pět měření a jejich výsledky byly zprůměrovány. Tentokrát byla pro přesnější měření rychlosti zejména paměťové implementace použita metoda `System.nanoTime()`. Asociační pravidla, ve kterých bylo vyhledáváno, vznikla vygenerováním z datové sady obsahující 90 dnů v historii objednávek.



Obrázek 4.1: Doby generování obou implementací podle datové sady

4.6.2 Výsledky a zhodnocení

Výsledky měření rychlosti vyhledávání pravidel jsou v tabulce 4.4. Tyto výsledky ukazují,

	Doba vyhledávání	
Implementace	**10 000 produktů**	**Jeden produkt**
paměťová	$5,8ms$	$0,6\mu s$
databázová	$77\ 598,4ms$	$7\ 759,8\mu s$

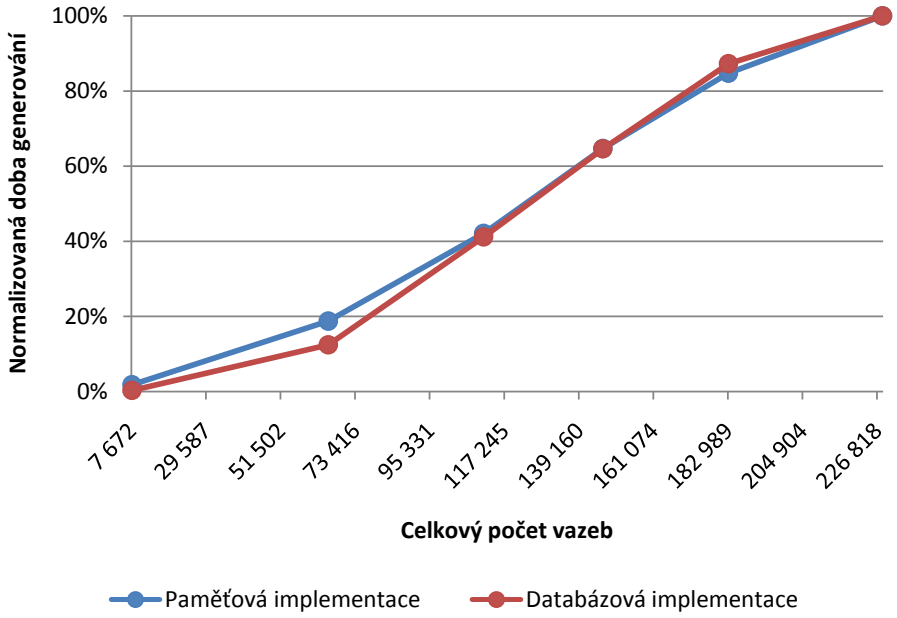
Tabulka 4.4: Rychlost vyhledávání pravidel

že paměťová implementace je v případě tohoto konkrétního testu o čtyři řády rychlejší (konkrétně 13 379x), než databázová implementace. Toto chování je očekávané, protože zatímco paměťová implementace pouze vyhledává výsledky v hash tabulce, databázová implementace musí provést dotaz do databázového systému. V případě databázové implementace trvalo získání doporučení na základě jednoho produktu v průměru 7,76 ms, což je pro naše účely také akceptovatelný výsledek.

I když obě implementace splňují požadavky na vyhledávání pravidel, i v tomto kritériu paměťová implementace přináší lepší výsledky.

4.7 Test paměťové náročnosti

V této části popisuji testováni paměťové implementace na její paměťovou náročnost. Z pochopitelných důvodů totiž existuje obava, že by paměťová náročnost mohla být pro pro-



Obrázek 4.2: Normalizovaná doba generování podle počtu položek

dukční nasazení příliš vysoká.

4.7.1 Metodika

Virtuální stroj Javy spoléhá při správě paměti na automatický garbage collector, který uvolňuje části paměti, které nejsou nadále využívány. Kvůli tomu není jednoduché přesně určit reálnou velikost využité paměti programem spouštěném ve virtuálním stroji. Nicméně je možné ji za běhu programu programu odhadnout. Explicitním vyvoláním garbage collectoru pomocí metody `System.gc()` a změřením volné paměti pomocí metody `Runtime.freeMemory()` ihned poté se dobereme množství volné paměti, kterou byl garbage collector v daném bodě programu schopen zajistit. Pokud změřené číslo odečteme od celkové velikosti paměti `Runtime.totalMemory()`, dostaneme číslo, které se bude zhruba blížit aktuálně alokované paměti v daném bodě programu.

Tuto metodiku jsem použil k naměření čtyř hodnot alokované paměti během procesu generování pravidel:

(a) alokovaná paměť před načtením vstupních dat

(b) alokovaná paměť po načtení vstupních dat

(c) paměť alokovaná během generování (bere se v potaz nejvyšší hodnota naměřená při generování)

(d) alokovaná paměť po vygenerování tabulky doporučení

Odečítáním mezi těmito čtyřmi hodnotami je možné se dobrat přibližných velikostí jednotlivých struktur. Další objekty v paměti jsou vzhledem k velikosti měřených struktur zanedbatelné.

4.7.2 Výsledky a zhodnocení

Výsledky měření jsou v tabulce 4.5.

	Datová sada					
Použití paměti	**7 dnů**	**30 dnů**	**60 dnů**	**90 dnů**	**180 dnů**	**360 dnů**
Vstupní data (b - a)	2 760	24 469	39 432	50 304	62 170	77 001
Generování pravidel (c - b)	0	8 092	24 480	39 245	52 290	50180
Tabulka doporučení (d - b)	1 421	13 954	37 645	60 822	79 268	75869

Tabulka 4.5: Měření využité paměti (v kilobytech)

Výsledky ukazují, že doporučení vygenerovaná z největší dostupné datové sady na paměťovém stroji zabírají méně než 80 MB paměti. Toto jsou data, která budou přítomna v operační paměti po celou dobu běhu aplikačního serveru. Je to z hlediska produkčního nasazení přijatelné řešení, protože velikost operační paměti přiřazené aplikaci na aplikačních serverech je v řádu gigabytů až desítek gigabytů.

Kapitola 5

Závěr

V této kapitole shrnu výstupy této práce a zhodnotím dosažené výsledky. Nastíním také možnosti dalšího rozvoje práce.

5.1 Shrnutí práce

V úvodu práce jsem seznámil čtenáře s východisky, která vedou ke snaze provádět analýzu nákupních košíku a konkrétně pak doporučování zboží v internetovém obchodě. Prozkoumal jsem, jak tuto problematiku řeší internetový obchod Amazon.com a některé obchody v České republice. Nakonec jsem uvedl důvody k vypracování této práce a popsal výchozí stav a dostupné prostředky pro její realizaci. Uvedl jsem také konkrétní požadavky, které navrhovaná nadstavba pro doporučování produktů musí splňovat.

V kapitole 2 práce jsem se zabýval technikami dolování dat a procesem získávání znalostí z databáze obecně, včetně předzpracování dat. Podrobněji jsem popsal teoretické základy pro dolování asociačních pravidel a použití asociačních pravidel při analýze nákupních košíků. Také jsem stručně seznámil čtenáře s prací společnosti Amazon.com zabývající se analýzou nákupních košíků.

V kapitole 3 jsem nejprve uvedl některé technické detaily architektury elektronického obchodu Czechcomputer.cz. Dále jsem popsal všechny teoretické základy pro výpočet doporučení produktů v reálném čase pomocí dolování asociačních pravidel. Navrhl jsem rozšíření základního pojetí dolování asociačních pravidel o časovou složku, která se snaží kompenzovat za časovou prodlevu mezi nákupy zboží jedním uživatelem. Popsal jsem dvě varianty implementace nadstavby a definoval požadavky na úspěšnou implementaci v produkčním nasazení. Pak jem provedl detailní návrh obou implementací .

Kapitola 4 se zabývá testováním obou implementací a jejich srovnáním v kritériích výkonnosti a paměťové náročnosti. Popisuje také použitou metodiku pro testování a vzorky dat, na kterých testování probíhalo.

5.2 Pokračování této práce

5.2.1 Úkony nad rámec této práce

Nad rámec této práce zbývá ještě řada úkonů, které je třeba provést pro úspěšné nasazení nadstavby v internetovém obchodě Czechcomputer.cz.

Použití nadstavby v elektronickém obchodě I když jsou implementace nadstavby plně funkční, je třeba ještě provést realizaci doporučování v rámci internetového obchodu. Ta zahrnuje návrh uživatelského rozhraní, implementaci a testování a další úkony související s nasazením do produkčního prostředí.

Analýza obchodních výsledků Po nasazení nadstavby bude vhodné provést analýzu, která ukáže, do jaké míry skutečně došlo ke zvýšení prodeje na základě doporučování zboží. Na základě této analýzy je pak možné provést další úpravu parametrů algoritmu pro dolování asociačních pravidel, případně provést úpravu algoritmu samotného.

5.2.2 Další náměty pro rozšíření práce

Použití obecných pravidel $A \Rightarrow B$ $(k \geq 2)$ V této práci byla pro doporučování produktů uvažována pouze asociační pravidla $i \Rightarrow j$ a to mimo jiné na základě práce společnosti Amazon.com. Jako námět pro další rozšíření této práce by mohlo být zajímavé otestovat použití obecných pravidel $A \Rightarrow B$ (A, B jsou množiny položek). To by znamenalo použití algoritmu Apriori v jeho obecné podobě s parametrem $k \geq 2$.

Použití víceúrovňových pravidel Víceúrovňová asociační pravidla umožňují dolovat asociace nejen mezi položkami, ale i mezi kategoriemi položek v konceptuální hierarchii. Některé položky může být totiž vhodnější doporučovat na základě příslušnosti do kategorie, nikoliv na základě konkrétního prodeje položky. Intuitivním příkladem takové asociace je nákup počítačové myši a zároveň podložky pod myš. V takovém případě nezáleží na značce ani modelu zakoupené myši, protože podložky pod myši jsou univerzální.

Implementace marketingových zvýhodnění produktů Určité zboží je někdy z obchodních důvodu třeba různými způsoby zvýhodňovat, aby se prodalo rychleji. Další rozšíření by mohlo umožňovat administrátorovi označit určité produkty za zvýhodněné a tím přimět algoritmus, aby je uživateli zobrazoval častěji

5.3 Zhodnocení výsledků

V rámci této práce byly vypracovány dvě implementace nadstavby pro doporučování produktů v rámci elektronického obchodu Czechcomputer.cz. Ty jsou co do výstupu ekvivalentní. Obě implementace jsou z pohledu stanovených požadavků zcela funkční a použitelné. Jako lepší varianta pro použití se nyní zdá paměťová implementace z důvodů lepšího výkonu. Databázová implementace nicméně zůstává jako další možnost, pokud by se při další práci ukázalo, že paměťová implementace má závažné problémy, které nyní nejsou známé.

Celkově práci považuji za úspěšnou. Během práce jsem detailně nastudoval problematiku analýzy nákupních košíků s použitím asociačních pravidel a dolování asociačních pravidel z transakčních dat. V obecnější rovině jsem také studoval problematiku předzpracování a následného dolování dat. Přínosem této práce pro mně bylo pochopení některých souvislostí mezi teoretickou rovinou dolování dat a uvedením do praxe. Také jsem získal představu, jakým způsobem lze v Javě řešit problémy při práci s velkými datovými sadami načtenými do paměti.

Literatura

[1] Berry, M. J. et al.: *Data mining techniques: for marketing, sales, and customer*. Wiley Publishing, Inc., 2004, ISBN 0-471-47064-3.

[2] Coppock, D. S.: Data Modelling and Mining: Why Lift? 2002, [Online; navštíveno 3.4.2011].
URL http://www.information-management.com/news/5329-1.html

[3] Han J., Kamber M.: *Data Mining: Concepts and Techniques, Second Edition*. Morgan Kaufmann Publishers, 2006, ISBN 978-1-55860-901-3.

[4] Jacobi, J. A. et al.: U.S. Patent 6,317,722. 2001, [Online; navštíveno 13.4.2011].
URL http://www.google.com/patents/about?id=fb8IAAAAEBAJ&dq=6,317,722

[5] Jacobi, J. A. et al.: U.S. Patent 7,113,917. 2006, [Online; navštíveno 13.4.2011].
URL http://www.google.com/patents/about?id=dtp6AAAAEBAJ&dq=7,113,917

[6] Jacobi, J.A. et al.: U.S. Patent 6,266,649. 2001, [Online; navštíveno 13.4.2011].
URL http://www.google.com/patents/about?id=NIAIAAAAEBAJ&dq=6,266,649

[7] Linden, G. et al.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. 2003, [Online; navštíveno 3.1.2010].
URL http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

[8] Olson, D. et al.: *Advanced Data Mining Techniques*. Springer, 2008, ISBN 978-3-540-76916-5.

[9] Rud, O. P.: *Data Mining. Praktický průvodce dolováním dat pro efektivní prodej, cílený marketing a podporu zákazníků (CRM)*. Computer Press, 2001, ISBN 80-7226-577-6.

[10] Vercellis, C.: *Business Intelligence: Data Mining and Optimization for Decision Making*. Wiley, 2009, ISBN 9780470511381.

[11] Zendulka, J., Bartík, V., Lukáš, R. aj.: Získávání znalostí z databází - studijní opora. 2006.

Dodatek A

Návod k přeložení a spuštění demonstračního programu

A.1 Požadavky

Ke spuštění je třeba mít nainstalované prostředí Java SE 6. K přeložení je nutné prostředí JDK 6 a nástroj pro sestavování Maven [1].

A.2 Přeložení

Program přeložíte spuštěním příkazu Mavenu v kořenovém adresáři projektu:

```
mvn package -Dmaven.test.skip
```

Maven vytvoří spustitelný JAR soubor (`recommend-1.0.jar`). Knihovny potřebné pro spuštění získáte příkazem Mavenu

```
mvn dependency:copy-dependencies -DoutputDirectory=.
```

A.3 Spuštění

Program spustíte standardně pomocí příkazu `java`:

```
java -jar recommend-1.0.jar
```

Program je implicitně spuštěn v módu pro vestavěnou databázi H2. Pro spuštění v prostředí Oracle je nejprve nutné do aktuálního adresáře nakopírovat knihovnu s Oracle JDBC driver pod názvem `ojdbc.jar` a poté spustit program s parametrem `--driverClass` takto:

```
java -jar recommend-1.0.jar --driverClass=oracle.jdbc.OracleDriver ...
```

[1] http://maven.org/

A.4 Parametry příkazové řádky

A.4.1 Základní parametry

–**url=** specifikuje URL databáze (povinný parametr)

–**username=** specifikuje uživatelské jméno pro přihlášení k databázi (povinný parametr)

–**password=** specifikuje heslo pro přihlášení k databázi (implicitně prázdné)

–**driverClass=** plný název třídy databázového driveru (implicitně `org.h2.Driver`)

–**help** zobrazí nápovědu

A.4.2 Parametry spouštějící akce

–**import-data** přegeneruje databázové schéma a spustí import z vybraného CSV souboru. Pro import je třeba nastavit další parametr:

  –**datafile=** cesta k CSV souboru obsahující data

–**generate-in-java** spustí generování asociačních pravidel v paměti. Další parametry:

  –**memory** změří využití paměti (může prodloužit dobu generování)

  –**compare** po vygenerování spustí porovnání asociačních pravidel s pravidli, která jsou aktuálně v databázi

  –**ruleOutputFile** zapíše vygenerovaná pravidla do CSV souboru

–**generate-in-db** vygeneruje pravidla v aktuální databázi

A.4.3 Parametry algoritmu Apriori

–**minBodyFrequency=** specifikuje minimální frekvenci těla pravidla (implicitně 30)

–**minAllFrequency=** specifikuje minimální frekvenci obou položek pravidla (implicitně 3)

–**minConfidence=** specifikuje minimální spolehlivost (v rozsahu 0 až 1, implicitně 0.05)

–**timeSpanWeight=** specifikuje váhu časové složky (implicitně 0.5)

–**minLift=** specifikuje minimální lift (implicitně vypnuto)