

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DATAMINING Z JABBERU

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

JAROSLAV SENDLER

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DATAMINING Z JABBERU

DATAMINING FROM JABBER

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

JAROSLAV SENDLER

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. JOZEF MLÍCH

BRNO 2011

Abstrakt

Předmětem této bakalářské práce bylo seznámení se s problematikou komunikace přes Jabber síť, která zde byla rozebrána. Konkrétním cílem bylo vytvoření jednoduchého Jabber klienta, který by byl schopen získávat statistická data. Nashromážděná data sloužila pro pozdější analýzu a grafickou reprezentaci informací z nich získaných.

Abstract

The objective of this thesis was acquaint oneself with problems of communication via Jabber network, which was also analyzed. The specific objective was to create a simple Jabber's client which would be able to obtain statistical data. The collected data was used for analysis and graphic representation of information.

Klíčová slova

Jabber, XMPP, robot, datamining, dolování dat, RapidMiner.

Keywords

Jabber, XMPP, robot, datamining, RapidMiner.

Citace

Jaroslav Sendler: Datamining z jabberu, bakalářská práce, Brno, FIT VUT v Brně, 2011

Datamining z jabberu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jozefa Mlícha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....

Jaroslav Sendler

16. května 2011

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu bakalářské práce Ing. Jozefovi Mlíchovi za ochotu a kladný přístup při konzultacích. Dále za poskytnutí hardware na němž běžel program a sbíral data.

© Jaroslav Sendler, 2011.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1	Úvod	3
2	XMPP	4
2.1	Architektura	4
2.2	XML	7
2.3	Stanza	8
2.4	Rozšíření	10
3	Data mining	13
3.1	Transformace dat	16
3.2	Metody dolování dat	19
3.3	Shlukování	20
3.4	Programy	23
4	Implementace	25
4.1	Databáze	25
4.2	Architektura	28
4.3	Robot	28
5	Vyhodnocení výsledků	30
5.1	Manuální rozbor dat	30
6	Závěr	31
A	Obsah CD	34
B	Manual	35
C	Konfigurační soubor	36
D	Slovník zkratk	37
E	Stanza - základní schéma	39
E.1	Iq	39
E.2	Message	39
E.3	Presence	40
E.4	Přehled průběhu rozšíření	41
F	Návrh databáze	44

¹ Kapitola 1

² Úvod

³ Dnes mezi velmi se rozšiřující technologie na poli sítě. Cílem této práce je získat neznámé
⁴ informace z real-time komunikační sítě Jabber. Předmětem této bakalářské práce bylo se-
⁵ známení se s problematikou komunikace přes Jabber síť, která zde byla rozebrána. Kon-
⁶ krétním cílem bylo vytvoření jednoduchého Jabber klienta, který by byl schopen získávat
⁷ statistická data. Nashromážděná data sloužila pro pozdější analýzu a grafickou reprezentaci
⁸ informací z nich získaných.

9 Kapitola 2

10 XMPP

11 V následující kapitole jsou, pro usnadnění a jednodušší pochopení, rozebrány základní sta-
12 vební kameny protokolu Extensible Messaging and Presence Protocol (XMPP). Konkrétně
13 jsou zde popsány stávající vlastnosti implementace, architektura protokolu XMPP obecně
14 [22, 23] a další detaily protokolu [1, 24, 14]. Vzhledem k požadavkům na dolování v da-
15 tech popsaných v následující kapitole je kladen důraz na vybraná rozšíření [21, 12]. Tato
16 rozšíření tvoří základ pro některé rozšířené statusy, jako je například User Tune [18], User
17 Mood [20], User Location [6] a další. Další informace použité pro popis a pochopení XML
18 jazyka byly čerpány z [10, 9].

19 Vznik samotného protokolu XMPP je datován do roku 2004 (březen), kdy na něj byl
20 přejmenován Jabber. Původní projekt Jabber byl vytvořen roku 1998 autorem Jeremie
21 Millerem, který ho založil za účelem vytvořit svobodnou otevřenou IM službu. Uvedený
22 projekt měl obsahovat tři základní vlastnosti, do kterých se zahrnují jednoduchost a sro-
23 zumitelnost pro implementaci, jednoduchost v oblasti šíření a otevřenost podobě veřejně
24 dostupného popisu samotného protokolu. Základní vlastnosti a výhody klientů a serverů
25 budou podrobněji popsány níže. Roku 1999, 4.ledna byl vytvořen první server se jménem
26 Jabber. Komunita vývojářů se chopila iniciativy a vytvořila klienty, kteří dokázali se ser-
27 verem komunikovat, pro různé platformy (Linux, Macintosh, Windows). Roku 2004 byl
28 protokol XMPP přidán mezi RFC¹ dokumenty. Základní norma popisující obecnou struk-
29 turu protokolu je RFC 3920 [22] a RFC 3921 [23], který se zaměřuje na samotný instant
30 messaging a zobrazení stavu. Další zdokumentovaná rozšíření jsou vydávána v podobě tzv.
31 XEP (XMPP Extension Protocol) dokumentů, které jsou známé také pod starším názvem
32 JEP (Jabber Enhancement Proposal). Dnešní počet těchto norem se blíží k číslu 300. Každý
33 XEP obsahuje stav vývoje (schválení), ve kterém se zrovna nachází.

34 Jako bezpečnostní prvky jsou zde podporovány SASL, TLS a GPG. XMPP protokol
35 je postaven na obecném značkovacím jazyce XML, proto vlastnosti popsané dále v této
36 kapitole platí i pro tento protokol.

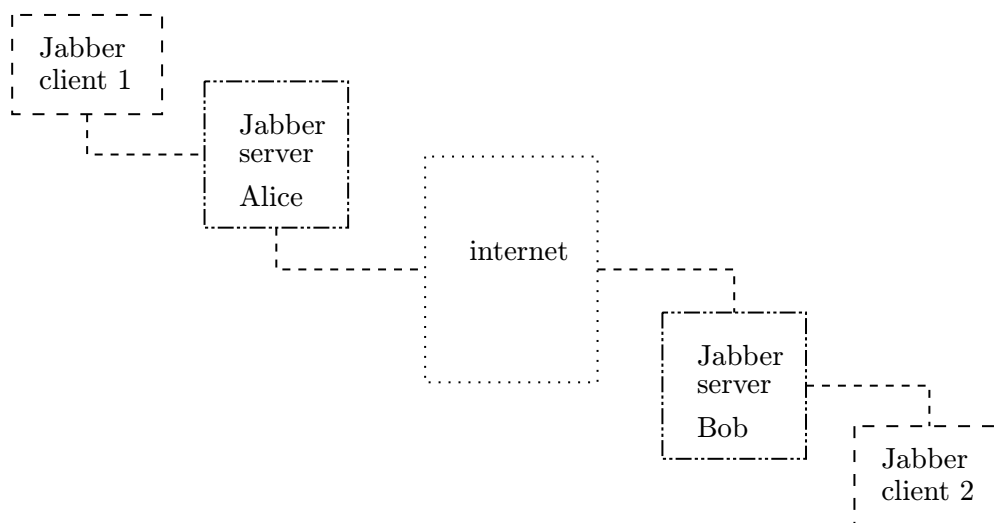
37 2.1 Architektura

38 Dobře navržená internetová technologie je tvořena správně fungujícími komponenty, které
39 mezi sebou dokáží vytvořit spojení a následně započít komunikaci. Pro popis Jabber ar-
40 chitektury v této práci bylo čerpáno z [1, 24]. Tato struktura se nejvíce podobá struktuře
41 posílání e-mailů. Hlavní předností Jabber sítě je, tak jako u elektronické pošty, její decentra-

¹RFC request of comments – žádost o komentáře

42 lizace. V případě Jabberu je decentralizace chápána jako možnost provozovat vlastní server,
 43 na rozdíl od jiných komunikačních systémů jako je například facebook, kde existuje pouze
 44 jediný poskytovatel služby. V případě serveru je kladen důraz na spolehlivost a rozšiřitelnost
 45 a u klienta na uživatele. Každý server pracuje samostatně, což znamená, že chod ani vý-
 46 padek jiné datové stanice žádným způsobem jeho běh neovlivní. V případě výpadku jiného
 47 serveru bude nedostupný pouze seznam kontaktů a služeb, které registrovaným uživatelům
 48 poskytoval.

49 Obrázek 2.1 znázorňující distribuovanou architekturu Jabberu byl převzat z [1] a dopl-
 50 něn o názvy jednotlivých komponent. Komunikace dvou Jabber klientů probíhá za účasti
 jejich serverů a sítě, která je spojuje. Spojení mezi nimi bývá často šifrováno.



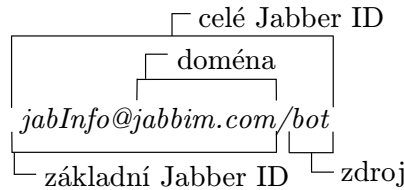
Obrázek 2.1: Distribuovaná architektura Jabber.

51 Architektura Jabber serverů využívá velké množství mezi-doménových připojení po-
 52 dobně jako internetový systém elektronické pošty. Komunikace klienta z jedné domény s
 53 klientem z jiné na rozdíl od e-mailového modelu nevyžaduje spolupráci třetích stran. Kli-
 54 ent se spojí s „domácím“ serverem, který přímo naváže spojení se serverem požadovaného
 55 klienta. Tyto vlastnosti jsou zárukou pro bezpečný přenos zpráv, znemožňující „krádeže“
 56 JID², který je popsán níže, a spamování.

58 Jabber ID

59 Jabber ID (JID) je jednoznačný virtuální identifikátor uživatele na síti. V případě založení
 60 účtu nejsou rozlišována velká a malá písmena, což znamená, že Jabber není case-sensitive.
 61 Jednoznačný Jabber identifikátor je složen ze dvou částí: *Jabber bare* nebo-li čisté ID a
 62 *resource* [22]. Základní část na první pohled připomíná e-mailovou adresu *user@server*.
 63 Druhá část slouží k přesné identifikaci jednotlivých spojení. Je použita ke směrování síťového
 64 provozu s uživateli v případě otevření většího množství spojení pod jedním uživatelem.
 65 Společně Jabber bare a resource tvoří tzv. *full JID* — *user@server/resource* například
 66 *jabInfo@jabim.cz/bot*. Jednotlivé části uživatelského jména popsané v tomto odstavci jsou
 67 ukázány v obrázku 2.2.

²uživatelské jméno



Obrázek 2.2: Rozebraná struktura Jabber ID.

68 Další vymoženost JabberID oproti e-mailové adrese je jeho možnost používat prakticky
 69 libovolné národní znaky u doménových jmen a uživatelských účtů [24]. Využíváním kó-
 70 dování UNICODE, se XMPP stává plně mezinárodní a není jako jiné protokoly omezen
 71 rozsahem ASCII tabulky. Přestože je tato vymoženost k dispozici, doposud není žádným
 72 výrazným způsobem využívána.

73 Klient

74 Klient je často jednoduchá aplikace pracující se vzdálenými službami, které jsou provo-
 75 vány serverem. V této práci je zastoupen robotem s konzolovým rozhraním. XMPP svou
 76 architekturou nutí, aby byl co nejjednodušší. Vlastnosti, které by měl mít, jsou shrnuty,
 77 podle [14] do tří bodů:

- 78 1. komunikace s jedním Jabber serverem pomocí TCP socketu, který garantuje spolehlivé
 79 doručení zpráv na rozdíl od UDP. Nad tímto transportním protokolem dále běží
 80 kryptografický protokol TLS, který zabezpečuje komunikaci klient–server a server–
 81 server.
- 82 2. rozparsování a následná interpretace příchozí XML zprávy „stanza“ (kapitola 2.3)
- 83 3. porozumění sadě zpráv (*message*, *iq*, *presence*) z Jabber jádra [22]

84 Server

85 Informace použité pro popis XMPP serveru byly čerpány z [14]. K hlavním charakteristi-
 86 kám serveru oproti klientovi, jehož základní vlastností byla jednoduchost, patří stabilita a
 87 bezpečnost. Je pro něj vyhrazen TCP port 5222. Komunikace mezi servery je realizována
 88 přes port 5269. Každý server uchovává seznam zaregistrovaných uživatelů, který nevykazuje
 89 žádný jiný server. Zaregistrovaní uživatelé v daném seznamu se mohou do sítě připojovat
 90 pouze přes něj. To zajišťuje nemožnost „krádeže“ účtu. Protože XMPP komunikace probíhá
 91 přes síť, musí mít každá entita adresu, v tomto případě nazvána JabberID. XMPP spoléhá
 92 na DNS což znamená, že používá jména na rozdíl od IP protokolu.

93 Server Jabber je systém spravující tok dat mezi jednotlivými komponentami, které spo-
 94 lečně tvoří Jabber služby. Například *Jabber Session Manager* (JSM) poskytne funkce pro
 95 IM komunikaci a práci se seznamem kontaktů. Komunikace mezi jednotlivými servery, jak
 96 je uvedeno na obrázku 2.1, je zprostředkována za pomoci komponenty *S2S* (server to ser-
 97 ver). Při připojení klienta k serveru je komunikace řízená pomocí *C2S* (client to server).
 98 Jak již bylo řečeno, Jabber síť využívá doménová jména místo špatně zapamatovatelných
 99 IP adres. Pro tento způsob identifikace je určena služba *dnsrv*, která se stará o překlad
 100 názvů. V podstatě je to komponenta, která zajišťuje směrování paketů na jiný server.

V tabulce 2.1 jsou shrnuty informace o serverech Jabberu. První sloupec tvoří jméno, následuje programovací jazyk, v němž je napsán. Většina aplikací pro servery je vydávána pod licencí GPL³. U všech aplikací byla zkoumána nejaktuálnější verze. Její číslo lze nalézt ve třetím sloupci. Všechny servery lze provozovat na operačním systému Linux a Windows. Na platformě Mac OS mohou být použity všechny zde jmenované vyjma jabberd2. Pět z šesti zde představených programů pro server Jabber jsou stále vyvíjeny, tedy kromě jabberd14. Hlavním účelem tabulky je prezentovat důležité vlastnosti serverů v oblasti podpory rozšířených statusů. Jedná se o standardy *pubsub*⁴ (XEP-0060) [12] a o jeho verzi zaměřenější více na uživatele *pep*⁵ (XEP-0163) [21]. Obě tato rozšíření tvoří nezbytnou základnu pro *rozšířené statusy* a proto je jejich podpora jak u serverů, tak klientů vyžadována. Podrobněji toto téma bude rozebráno v některé následující podkapitole.

Server	Jazyk	Verze	XEP-0060	XEP-0163
ejabberd	Erlang/ Top	2.1.6	ANO	ANO
Openfire	java	3.6.4	ANO	ANO
jabberd2	c	2.2.11	NE	NE
jabberd14	c, c++	1.6.1.1	ANO	NE
Prosody	lua	0.7.0	NE	ANO
Tigase	java	5.0.0	ANO	ANO

Tabulka 2.1: Přehled Jabber serverů.

Z výše uvedené tabulky je zřejmé, že aplikace pro servery, které jsou stále ve vývoji, podporují tzv. *rozšířené statusy*. Tedy kromě programu jabberd2.

2.2 XML

Jazyk XML (eXtensible Markup Language) [9], meta-jazyk pro deklaraci strukturovaných dat, je jádrem protokolu XMPP. Samotný jazyk vznikl rozšířením meta-jazyka SGML, jež slouží pro deklaraci různých typů dokumentů. Základní vlastností je jednoduchá definice vlastních značek (tagů). Dokument XML se skládá z elementů, které můžeme navzájem zanořovat. Vyznačujeme je pomocí značek — počáteční a ukončovací. Pomocí tohoto jazyka je tvořena *stanza* popsána v následující kapitole.

Ukázka možné struktury dokumentu psaného jazykem XML je zobrazena na příkladu 2.1. Standardně je předpokládáno, že je psán v kódování UTF-8 [10], ale je-li jako v tomto případě použito jiné, musí být konkrétní kódování uvedeno na jeho počátku. V opačném případě nemusí být obsah správně zobrazen. Na začátku dokumentu se také uvádí verze XML, ve které je dokument psán (1. řádek příkladu). Následuje kořenový element, který je uzavřen na samotném konci dokumentu. 4. řádek prezentuje možnost použití prázdného elementu, který obsahuje jeden atribut s názvem zkratky fakulty. Velký význam zde mají úhlové závorky. Jsou jimi z obou stran obaleny všechny elementy.

³General Public License — všeobecná veřejná licence GNU

⁴Publish-Subscribe

⁵Personal Eventing Protocol

```

1      <?xml version="1.0" encoding="iso-8859-2"?>
2      <fakulta>
3          <název>Fakulta informačních technologií</název>
4          <zkratka fakulty="FIT"/>
5          <typy studia>
6              <bakalářské titul="Bc."></bakalářské>
7              <magisterské></magisterské>
8              <doktorské></doktorské>
9          </typy studia>
10     </fakulta>

```

Příklad 2.1: Ukázka základního XML dokumentu.

2.3 Stanza

Základní jednotkou pro komunikaci založenou na XML je stanza. Z jednoduššího pohledu je možné se na ni dívat jako na jeden dlouhý XML soubor. Při zahájení komunikace se tento soubor „otevře“. Jeho samotné uzavření probíhá až při odhlášení od sítě, nebo-li přepnutí klienta do stavu offline. Stanzu je tedy možné vnímat jako stream, který obsahuje všechna data probíhající komunikace. Mezi elementy používané pro komunikaci klienta se serverem patří tyto tři: *message*, *presence* a *iq*. Každý zde uvedený člen má svůj jednoznačný význam. V následujících odstavcích jsou jednotlivé části stanzy blíže definovány a na reprezentativních příkladech jsou ukázány jejich základní struktury a možnosti využití v praxi.

První prvek, který bude charakterizován je označen anglickým výrazem *message* (zpráva). Jak již název napovídá, slouží k posílání zpráv všeho druhu. Je to základní metoda pro rychlý přenos informací z místa na místo. Zprávy jsou typu „push“, což znamená, že jsou odeslány a není očekávána žádná aktivita od příjemce, která by přijetí potvrdila. Jedno z dosavadních využití se nachází v klasické komunikace po internetu, tzv. instant messaging (IM). K dalším možným použitím patří skupinový chat a oznamovací nebo upozorňující zprávy. Každá z těchto zpráv je tvořena z minimální povinné struktury. Tak jako u klasické poštovní korespondence nesmí chybět adresa odesílatele a adresa příjemce, kterému je zpráva adresována. Podle možnosti použití jsou zprávy děleny do kategorií. Jmenovitě toto rozdělení implementuje atribut *type*, který může nabývat jednu ze čtyř hodnot. Jsou rozlišovány zprávy pro komunikaci mezi dvěma entitami, skupinový chat, upozornění, chybová zpráva a v neposlední řadě zpráva bez kontextu vyžadující odpověď příjemce. Nakonec nesmí být opomenut blok zprávy, pro uživatele IM nejdůležitější, nesoucí vlastní obsah.

Základní použití struktury elementu *message* je prezentováno na příkladu 2.2. Na prvním řádku je uveden atribut, značící odesílatele. Druhý řádek obsahuje JID klienta, který zprávy přijímá. Následuje informace o typu zprávy a poté je uveden element *body* nesoucí samotný obsah.

```

1      <message from="user@jabbbim.com"
2              to="jabinfo@jabbbim.com/bot"
3              type="chat"
4      <body> Kolik je hodin? </body>
5      </message>

```

Příklad 2.2: Použití elementu *message*.

Další částí stanzy je poskytována struktura pro *request-response* (žádost–odpověď)

vazbu, podobnou metodám GET, POST a PUT z protokolu HTTP [24]. Zkráceně je označována pomocí dvou počátečních písmen *Info/Query* nebo-li IQ. Na rozdíl od elementu *message* tvoří *iq* spolehlivější přenos, optimalizovaný pro výměnu dat (binární data). K dalším rozdílům patří povinnost příjemce odpovědět na každou přijatou zprávu, nebo-li potvrdit její doručení. Skutečnost, že je na právě požadovanou zprávu odpovězeno, zajišťuje parametr *id*. Iq dotaz nebo odpověď musí obsahovat stejnou hodnotu tohoto atributu jako zpráva vytvořená žádajícím subjektem. Další povinný atribut rozděluje iq na čtyři typy. Jednotlivé žádosti na proces nebo akci jsou posílány samostatně [23]. V příloze E je uvedena rozsáhlejší struktura tohoto elementu. Použití nachází v případech, které nastavují, žádají nebo informace posílají. Tato struktura je využívána pro novou registraci, posílání seznamu kontaktů a další.

Příklad 2.3 znázorňuje základní použití elementu *iq*. Uživatel *user* posílá dotaz na získání seznamu kontaktu (řádek 5.).

```
1      <iq from="user@jabbim.com/doma"
2          to="user@jabbim.com"
3          id="uhhfw23648"
4          type="get"
5          <query xmlns="jabber:iq:roster"/>
6      </iq>
```

Příklad 2.3: Použití elementu *iq*.

Poslední a pro tuto práci nejdůležitější prvek stanzy je *presence*. V případě, že nemá určeného příjemce, tak funguje způsobem jako broadcast. Což znamená, že jsou informace směrovány všem klientům, kteří jsou zaregistrováni k jejímu odběru. Presence v českém překladu informace o stavu (přítomnost) rozesílá dostupnost ostatních entit v síti. Jedná se tedy o nastavení uživatelské dostupnosti tak jako na jiných real-time komunikačních a sociálních systémech.

Existuje několik základních stavů statusů, které reprezentují aktuální dosažitelnost uživatele. Tento jev je vyjádřen pomocí elementu *show*, který disponuje čtyřmi možnostmi. První oznamuje, že je uživatel k dispozici a schopen aktivní komunikace. Druhá často se vyskytující možnost naznačuje, že je subjekt krátkou dobu pryč od svého IM klienta. Tento a další dva stavy, popsané dále, jsou často změněny bez lidského zásahu (pomocí pc nebo jiného zařízení) prostřednictvím funkce známé jako „auto-away“. Poslední dva stavy charakterizují delší časové období nečinnosti. Tato oznámení o změně stavu uživatele jsou často zasílána pouze kontaktům, které se nacházejí v režimu online. Tato optimalizace přispívá ke snížení síťového provozu, jelikož presence v reálném čase při komunikaci využívá velké množství šířky pásma.

Základní použití *presence* je zobrazeno v příkladu 2.4. Kontakt *jabinfo@jabbim.com/bot* (1. řádek) posílá informace o svém stavu (řádek č. 2) a svůj status (č. 3).

```
1      <presence from="jabinfo@jabbim.com/bot"
2          <show> online </show>
3          <status> Jsme zde. </status>
4      </presence>
```

Příklad 2.4: Použití elementu *presence*.

Obsáhlejší struktura elementu *presence* je zobrazena v příloze E, kde je rovněž k nalezení

189 přehled všech možných stavů.

190 Jak již bylo zmíněno v části o Jabber ID, Jabber podporuje práci s více současně připoje-
191 nými klienty k jednomu Jabber účtu. Vysvětlení funkčnosti bude prezentováno na příkladu
192 uživatele přihlášeného na stolním počítači a z klienta v mobilním telefonu. U obou těchto
193 připojení je použit stejný Jabber bare, ale odlišného resource, například *domov* a *mobile*.
194 Právě tento rozdíl v tzv. „full“ adrese účtu zajišťuje jednu ze dvou možných podmínek
195 pro správnou adresaci zpráv. Druhá možnost, která bude uplatněna při použití adresy účty
196 pouze ve formě Jabber bare, je nastavení priority u jednotlivých programů. Priorita je číslo
197 v rozsahu hodnot od -128 do 127, kde klient s větší prioritou má přednost před klientem s
198 nižší. Nastane-li případ připojení více klientů se stejnou prioritou, každý server se při roze-
199 sílání zpráv zachová podle vlastní implementace. Některé rozešlou zprávy všem klientům,
200 jiné naopak jen poslednímu přihlášenému.

201 2.4 Rozšíření

202 Dále se tato práce zabývá rozšířeními protokolu XMPP o další vlastnosti k jejichž popisu
203 slouží XEP. Pro tuto práci jsou nepostradatelné „statusy“, pro které tvoří základ standardy
204 XEP-0060 [12] a XEP-0163 [21] zkráceně PEP⁶. Obě tato rozšíření umožňují strukturované
205 pracovat, používat a přenášet další XEP protokoly. Jako příklady relevantní k práci jsou zde
206 uvedeny protokoly *User Location* (kde se uživatel právě nachází) [6], *User Tune* (co uživatel
207 poslouchá za hudbu) [18], *User Mood* (aktuální nálada uživatele) [20] a *User Activity* (co
208 uživatel právě dělá) [11]. Jsou to tedy protokoly založené na PEP, které vyžadují podporu
209 nejen v klientech, ale i na straně serveru (zobrazuje tabulka 2.1). S touto informací úzce
210 souvisí další protokol XEP-0115 [7], který umožňuje zjistit podporované schopnosti klienta,
211 případně, které informace je ochoten přijímat. Tato vlastnost bude popsána níže v části
212 zabývající se podporovanými vlastnostmi.

213 Všechna tato rozšíření by mohla být přidána přímo do statusu viz příklad 2.4, avšak
214 ten je primárně určen k informování o přítomnosti na IM síti. Hlavní rozdíl mezi PEP a
215 obyčejným posílání stavu pomocí presence je v pravomoci klienta přijmout nebo odmítnout
216 informaci, na rozdíl od presence, jež je přijata vždy.

217 Základ přenosu informací začíná na straně klienta, který chce všechny ve svém roster
218 listu (seznam kontaktů) informovat o statusu. Zašle zprávu obalenou v elementu *iq* serveru.
219 Ukázka této zprávy je prezentována na příkladu 2.5, který znázorňuje zaslání informace o
220 druhu hudby, kterou v danou chvíli uživatel poslouchá. Využívá k tomu rozšíření *User*
221 *Tune*, definovaném na řádku číslo 5. Základ zprávy oznamující začátek vysílání informací
222 o rozšířených stavech je vždy stejný. Liší se pouze řádkem 3. a obsahem elementu *item* v
223 příkladu 2.5.

224 V případě úspěšného přijetí *iq* zprávy serverem, každý, kdo se zaregistroval k odebírání
225 rozšířených statusů, obdrží oznámení ve formě *message*. Oznámení bude také doručeno všem
226 resources. Celá zpráva i všechny další náležitosti jsou uvedeny v příloze E.4.

227 Podporované vlastnosti

228 Jednotlivá rozšíření protokolu XMPP jsou nepovinná, a proto nemusí být ve všech klient-
229 ských aplikacích podporována. Pro zjištění podporovaných rozšíření se používá XEP-0115
230 Entity Capabilities [7]. Toto rozšíření výrazně snižuje počet a velikost komunikací a přenosů

⁶Personal Eventing via Pubsub

```

1      <iq from='user@jabber.com' type='set' id='pub1'>
2          <pubsub xmlns='http://jabber.org/protocol/pubsub'>
3              <publish node='http://jabber.org/protocol/tune'>
4                  <item>
5                      <tune xmlns='http://jabber.org/protocol/tune'>
6                          <artist>Daniel Landa</artist>
7                          <length>255</length>
8                      ...

```

Příklad 2.5: Začátku vysílání rozšířeného statusu.

231 zpráv mezi uživateli. Dotazem zobrazeným na příkladu 2.6 je zjištěna schopnost jednotlivých klientů, kterou následně server využije pro správné směrování rozšířených statusů. 232 Všechny zde zmiňované rozšíření a protokoly z této kapitoly je možné u každého klienta 233 (seznam klientů obsahuje tabulka v příloze E.4) vyčíst z atributu *ver* (druhá část u atributu 234 *node*), který je vypočítán ze všech podporovaných protokolů klienta, viz [7]. 235

```

1<iq from="user@jabber.com" id="disco1"
2    to="jabinfo@jabber.com/bot" type="get">
3    <query xmlns="http://jabber.org/protocol/disco#info"
4        node="http://code.google.com/p/exodus/#QgayPKawpkPSDYmwT/WM94uAlu0=" />
5</iq>

```

Příklad 2.6: Dotaz na podporované protokoly.

236 Další rozšíření

237 V následujících několika odstavcích budou přiblíženy specifikace jednotlivých rozšíření XEP, 238 které slouží jako zdrojová data pro dolování a jsou relevantní k tématu práce.

239 Prvním rozšířením, nad rámec základních vlastností Jabberu, které zde bude podrobněji 240 rozebráno, je elektronická verze klasické vizitky nebo-li *VCard*. Jeho specifikací se zabý- 241 vají dva standardy. Jelikož novější verze XEP dokumentu [13] se v době psaní této práce 242 nacházela ve stavu „experimental“, což znamená, že ještě není schválena jako standard, je 243 pouze ve stavu návrhu. Proto bylo použito verze starší [16]. Jednoduše řečeno je *VCard* 244 struktura, která nese informace o uživateli jako je jméno, příjmení, e-mail, adresa bydliště 245 i zaměstnání a další údaje. Data jsou dále zveřejňována na síti, z čehož vyplývá, že jsou 246 dostupná ostatním uživatelům. Vyplnění těchto osobních údajů je dobrovolné a tak se u 247 některých uživatelů nachází pouze přezdívka a JID, které jsou často předdefinovány auto- 248 maticky. Nedílnou součástí všech sociálních a komunikačních systému jsou malé fotografie, 249 loga nebo ikony, kterými se uživatelé prezentují. V síti Jabber tomu není jinak, a proto je 250 samotný obrázek zahrnut přímo do *VCard* v položce *photo*. Podrobnější informace o jeho 251 nastavení a přijímání je možné nalézt v *vCard-Based Avatars* [19], který jej definuje.

252 Díky základní podmínce XMPP protokolu (otevřenost) existuje mnoho různých aplikací, 253 pomocí kterých lze v síti Jabber komunikovat. S programy, používanými uživateli, úzce 254 souvisí další zde implementované rozšíření. Jedná se o realizaci *Software Version* dokumentu 255 [17], který se právě zabývá získáváním informací o samotných aplikacích. Je-li toto rozšíření 256 podporováno je díky němu možné zjistit jméno a verzi používané aplikace. Informace o 257 operačním systému často nejsou kvůli bezpečnosti ani vyplněny. Podrobnější informace o 258 softwarové výbavě klienta je možné zjistit pomocí XEP [7], o kterém již bylo dříve psáno v

259 odstavci zabývajícím se podporovanými vlastnostmi klientských aplikací.

260 S rozšířením tzv. „chytrých“ mobilních zařízení mezi širší veřejnost vzniklo několik no-
261 vých disciplín spojených s určováním zeměpisné polohy, jako je například geocaching. Geo-
262 grafická poloha je přenášena ve formě souřadnic popisující přímo zeměpisnou šířku a délku.
263 Současně lze informaci o poloze přenášet i slovně ve formě adresy. Příkladem slovního po-
264 pisu je ulice, číslo popisné, město a další. Mnoho aplikací, které mají k dispozici GPS
265 přijímač, vysílají a aktualizují zeměpisné informace automaticky, například po určité době
266 nebo změně polohy o určitou vzdálenost. Toto a další níže popsané rozšíření jsou postaveny
267 na již zmiňovaném PEP. Některé části protokolů jsou zjednodušeny a připraveny tím pro
268 „mobilní instant messaging“.

269 Pro sdělení informací o stavu klienta není v základní verzi Jabberu mnoho. Pomocí
270 presence je možné „pouze“ prozradit, zda je uživatel připraven komunikovat nebo je mo-
271 mentálně nedostupný a to v několika verzích lišících se délkou nepřítomnosti. Pokročilejší
272 nastavení statusu nabízí *User Mood* [20] a to ve formě sdělení současné nálady, jako je
273 například radost. Další možné upřesnění činnosti uživatele jsou definovány v *User Activity*
274 [11], kde každá činnost je složena z povinné obecné kategorie a nepovinné, která informaci
275 upřesňuje. Příkladem může být *eating* a *having-a-snack* tj. uživatel jí, uživatel svačí.

276 K poslednímu rozšíření implementovanému v této práci patří *User Tune* [18], které
277 umožňuje uživateli šířit informace o aktuálně poslouchané hudbě. Některé dnešních hu-
278 dební přehrávače dokáží automaticky spolupracovat s IM klientem a předávat informace o
279 hudbě bez nutného lidského zásahu. Ve zprávě jsou tedy přenášeny informace o skladbě,
280 interpretovi, albu a další informace, které mohou být získávány z MP3 ID3v1 nebo novější
281 ID3v2 tag.

282 Podpora rozšíření v aplikacích je ukázána v tabulce v příloze G. Z této tabulky vyplývá,
283 že rozšířenost aplikací podporující výše popsaná rozšíření je poměrně malá. Například v
284 předcházející zmiňované části o poslouchané hudbě, při stavu, kdy program toto rozšíření
285 nepodporuje, je posíláno pomocí normální presence. Jméno skladatele, alba a další podrob-
286 nosti jsou shrnuty do statusu, tudíž jsou doručeny všem uživatelům ze seznamu kontaktů.

287 Kapitola 3

288 Data mining

289 Třetí kapitola se zabývá procesem dobývání znalostí z databází. Popisuje jej jako disciplínu,
290 která vznikla za účelem vytěžení informací z dat, která jsou v nepřehledném množství uklá-
291 dána v databázích. Díky velikosti dnešních disků, objem ukládaných dat neustále roste. S
292 tím také úzce souvisí zvětšující se poměr nepotřebných a zašumělých dat vůči užitečným
293 informacím. V této kapitole jsou mimo jiné popsány metody používané k dolování z dat,
294 které jsou relevantní k této práci.

295 Na začátku kapitoly je rozebrán pojem získávání znalostí databází, jehož jednu pod-
296 statnou část tvoří samotný data mining. Dále je vysvětlena základní terminologie, pro kterou
297 bylo čerpáno z [8]. Cílem první podkapitoly je přiblížení způsobu, jakým byla data uložena
298 v databázi, připravena k samotnému data miningu. Celá druhá podkapitola je věnována vy-
299 braným metodám pro dolování dat a vlastnostem, které je od sebe navzájem odlišují. Jsou
300 zde rozebrány *asociační pravidla*, pro jejichž popis bylo čerpáno z [2]. Pro ostatní metody,
301 které jsou popsány dále, byla jako zdroj informací použita kniha [5]. Poté následuje třetí
302 podkapitola, která se podrobněji zabývá jednou z metod pro dolování dat a to *shlukováním*.
303 Obsahem této části jsou již konkrétní algoritmy pro shlukování dat [27, 3] a také metoda
304 *k-Means* využívaná v praktické části této práce. Kapitulu uzavírá stručný přehled vybra-
305 ných programů pro data mining a podrobnější seznámení s nástrojem *RapidMiner*, který je
306 v této práci využíván pro samotné dolování.

307 Terminologie

308 Pojem data mining nebo-li česky dolování dat se začal ve vědeckých kruzích objevovat
309 počátkem 90. let 20. století. První zmínka pochází z konferencí věnovaných umělé in-
310 teligenci (IJCAI'89¹–mezinárodní konference konaná v Detroitu, AAAI'91² a AAAI'93–
311 americké konference v Californii a Washingtonu, D.C) [2].

312 Tradiční metoda získání informací z dat je realizována jejich manuální analýzou a inter-
313 pretací. V praxi ji například nalezneme v odvětví zdravotnictví, vědy, marketingu (efektivita
314 reklamních kampaní, segmentace zákazníků) a dalších. Pro tyto a mnoho dalších disciplín
315 je manuální zpracování příliš pomalé, drahé a vysoce subjektivní. Další důvod k přechodu
316 na jiné metody je objemnost dat, která dramaticky vzrostla a tudíž se manuální analýza
317 stává zcela nepraktická. Databáze rychle rostou ve dvou následujících kategoriích:

- 318 1. počet záznamů nebo-li objektů v databázi

¹International Joint Conference on Artificial Intelligence

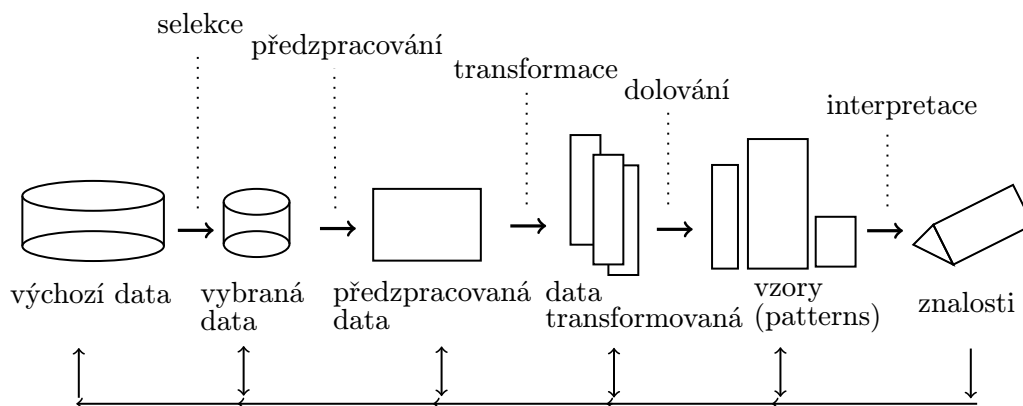
²Association for the Advancement of Artificial Intelligence

319 2. počet polí nebo-li atributů objektů v databázi

320 Proces data mining je pouze jedna část z odvětví nazývané dobývání znalostí z data-
 321 bází nebo-li KDD³ definované níže v definici 3.0.1. Vznik disciplíny KDD je důsledkem
 322 nepřehledného množství automaticky sbíraných dat, která je potřeba dále využívat. Pod-
 323 statným znakem celého procesu je správnost reprezentace výsledků formou, která má k
 324 uživateli nejbližší. Jako příklad bude uvedena implikace ve tvaru rozhodovacích pravidel,
 325 asociační pravidla, rozhodovací stromy, shluky podobných dat a další. Základem KDD je
 326 praktická použitelnost metod. Očekává se zjištění nových skutečností namísto prezentování
 327 již známých informací.

328 **Definice 3.0.1** KDD je chápáno jako interaktivní a iterativní proces tvořený kroky se-
 329 lekce, předzpracováním, transformace, vlastního „dolování“ (data-mining viz 3.0.2) a in-
 330 terpretace [2].

331 Grafické znázornění definice 3.0.1 je popsáno schématem na obrázku 3.1, který pre-
 332 zentuje časový harmonogram v KDD. Schéma znázorňuje následnost jednotlivých procesů,
 333 které tvoří KDD. KDD je iterativní proces, z čehož vyplývá, že skutečnosti nalezené v pře-
 334 dešlých částí zjednoduší a zpřesní vstupy pro následující fáze. Jakmile jsou znalosti získány,
 335 jsou prezentovány uživateli. Pro přesnost může být část procesu KDD ještě upravena. Tím
 336 budou získány „přesnější a vhodnější“ výsledky.



Obrázek 3.1: Proces dobývání znalostí z databází podle knihy autora Fayyad [4].

337 Vzhledem k obrázku 3.1, který prezentuje jednotlivé kroky získávání znalostí z databází
 338 budou dále tyto procesy popsány. Prvním část v KDD je tvořena výchozími daty, které slouží
 339 jako zdroj pro ostatní fáze. Samotný popis získávání těchto dat je popsán v předcházející
 340 kapitole. Procesu selekce dat, je kladen za cíl, vybrat co možná „nejúčinnější“ množinu
 341 dat a tím i zmenšit její celkový objem. V této části získávání znalostí se při vybírání dat
 342 bere ohled na to, jak se jednotlivá data vztahují ke konkrétnímu uživateli. Následující proces
 343 nazvaný transformace se zabývá převedením dat do vhodného formátu pro samotné dolování
 344 informací. Tato část je popsána v následující podkapitole, kde hlavní úlohu při transformaci
 345 je čas. Vybrané metody pro dolování dat jako je například shlukování a další, jsou taktéž
 346 v této práci popsány níže.

³Knowledge Discovery in Database

347 Získávání znalostí z databází je proces složen z několika kroků vedoucích od surových
348 dat k formě nových poznatků. Iterativní proces je složen, tak jak je prezentováno v [5], z
349 následujících kroků:

- 350 • **čištění dat** – fáze, ve které jsou nepodstatné údaje odstraněny z kolekce.
- 351 • **integrace dat** – kombinování heterogenních dat z několika zdrojů do společného
352 jediného zdroje.
- 353 • **výběr dat** – rozhodování o relevantních datech.
- 354 • **transformace dat** – také známý jako konsolidace dat. Fáze, ve které jsou vybraná
355 data transformována do formy vhodné pro dolování.
- 356 • **data mining** – zásadní krok, ve kterém jsou aplikovány vzory na data.
- 357 • **hodnocení modelů** – vzory dat zastupují získané znalosti.
- 358 • **prezentace znalostí** – konečná fáze, zjištěné poznatky jsou reprezentovány uživa-
359 teli. Tento základní krok využívá vizualizační techniky, které pomáhají uživa-
360 telům porozumět a správně interpretovat získané výsledky.

361 Jak je uvedeno v [5], běžně jsou některé z těchto kroků kombinovány dohromady. Kroky
362 čištění dat a integrace dat mohou být provedeny společně, tak jako to prezentuje schéma
363 na obrázku 3.1.

364 V této podsekci jsou ve stručnosti vysvětleny základní nejdůležitější pojmy dále v práci
365 využívané.

366 Definic výrazu data mining se v odborné literatuře nachází několik. Zde uvedená je
367 kombinací dvou „definic“ z [15].

368 **Definice 3.0.2** Data Mining je proces objevování znalostí, který používá různé analytické
369 nástroje sloužící k odhalení dříve neznámých vztahů a informací z velmi rozsáhlých databází.
370 Výsledkem je predikční model, který je podkladem pro rozhodování [15].

371 Mezi další čteně se vyskytující pojmy v tomto odvětví patří například data, znalosti a
372 informace. Tyto termíny jsou často mezi sebou zaměňovány, proto jsou níže jejich významy
373 striktně definovány tak jako v [8].

374 Jedna z několika existujících definic pojmu data je uvedena v definici 3.0.3, která je po-
375 pisuje z pohledu informačního. Data často nemají sémantiku (význam) a bývají zpracována
376 čistě formálně.

377 **Definice 3.0.3** Data jsou z hlediska počítačového pouze hodnoty různých datových typů.

378 Informace lze chápat jako data, která byla obohacena o sémantiku (význam), jsou tedy
379 již zpracovaná a interpretována uživatelem. Znalosti, jsou řazeny do stejné kategorie jako
380 informace, ale jejich interpretace bývá ještě složitější. Často bývají tvořeny shluky informací,
381 proto jsou reprezentovány jako odvozené informace. Podle studijní opory [8] jsou znalosti
382 informace, které jsou zařazeny do souvislostí.

3.1 Transformace dat

Transformace dat tvoří třetí část z celkového procesu dobývání znalostí z databází. Než se data dostala do tohoto stavu, bylo na nich provedeno několik kroků, ve kterých byla upravována. V první fázi byla sbírána Jabber komunikace, která je popsána v první kapitole. Druhá fáze byla zaměřena na zúžení výsledné množiny a proto byla vybrána jen relevantní data. I přes tyto kroky relační databáze obsahuje velké množství dat, která se nenachází ve stavu, aby mohla být použita jako zdroj pro data mining. Jak bude popsáno v následující podkapitole většina metod pro dolování dat pracuje pouze s daty, která obsahují kvantitativní proměnné. Za tímto účelem je potřeba všechny atributy tabulky z databáze, které mají být nadále používány, převést na měřitelné hodnoty.

V této práci se bude pracovat s atributy nesoucí informace o jak aktuálních tak minulých stavech uživatelů, kteří si přidali účet *jabInfo@jabbin.com* do svého seznamu kontaktů. A tak byla jejich každá změna statusu uložena do databáze. Z důvodu nečíselné hodnoty stavů, jako je například *available*, *away* a další, je třeba provést jejich transformaci na kvantitativní hodnoty. Proces byl proveden pomocí bijektivního zobrazení. Kde zobrazení je, podle [8], funkce s definičním oborem S a oborem hodnot T , která je nazývána binární relace $f \subseteq S \times T$. V této relaci se nevyskytují dvě různé dvojice (s, t_1) a (s, t_2) , kde $s \in S$ a $t_1, t_2 \in T$. Prvky t_1 a t_2 jsou různé. Z toho vyplývá, že každému prvku s z množiny S je přiřazen jednoznačně právě jeden prvek $t \in T$. Tuto definici je možné zapsat ve tvaru:

$$f : S \rightarrow T,$$

kde S a T jsou množiny (D_f, H_f) .

Konkrétní případ transformace z této práce tedy bude obsahovat množinu S , kde

$$S = \{Available, Chat, Away, DND, XA, Unavailable\}$$

a množina T , kde

$$T = \{120, 110, 90, 70, 50, 0\},$$

do které budou jednotlivé prvky z množiny S bijektivně zobrazeny. Kdy bijekce je zobrazení, které každému prvku z cílové množiny, konkrétně z množiny T , přiřazuje právě jeden prvek z množiny počáteční, tedy S .

Při výběru velikosti hodnoty, pro výslednou množinu T , bylo čerpáno z programu Gajim, který je multiplatformní klient s velkou podporou standardů a rozšiřujících protokolů. Hodnoty v množině T byly zvoleny tak, aby měly sestupné uspořádání, a aby bylo možné je dobře mezi sebou porovnávat. Jsou-li vybrány dvě hodnoty například *available* a *unavailable*, vzdálenost mezi nimi musí být větší než vzdálenost například u hodnot *chat* a *away*. Na druhou stranu prvky ze vstupní množiny S jsou striktně definovány podle Jabber standardu, který je popsán v RFC [22].

404 Temporální data

Druhá podstatná transformace, pro kterou bylo čerpáno z [25], se tak jako první nezabývá transformováním dat textových na data, jejichž obsah by byl tvořen kvantitativními proměnnými. V této části jsou řídká temporální data transformována na hustá. Pro následné vyhodnocení a data mining je potřeba řádkům z tabulky presence přidat konečné časové razítko, které by vymezilo interval doby platnosti těchto dat.

410 Jak již bylo uvedeno, hlavním rozdílem mezi temporálními databázemi a ostatními je
 411 schopnost uchovávat časové údaje. Své uplatnění nachází v odvětvích, kde je potřeba zpraco-
 412 vávat stará a zároveň nová data, například v oblastech medicíny, finančnictví, monitorování
 413 a dalších. Jednotlivé záznamy, které jsou závislé na čase, jsou v databázích ukládány jako
 414 samotné body, tedy diskrétně. Přestože v reálném světě je většina těchto údajů z pohledu
 415 času spojitých.

416 K dalšímu popisu temporálních databází nyní budou charakterizovány tři důležité po-
 417 jmy, pomocí nichž jsou databáze dále děleny. Prvním pojmem je *granualita*, která udává
 418 nejmenší časovou jednotku, kterou databáze rozlišují. Hodnoty, které může nabývat, jsou
 419 hodina, den, rok a další. Velikost granuality ovlivňuje velikost objemu dat, který je přímo
 420 úměrný s přesností záznamů. Dalším pojmem je *čas platnosti*, která reprezentuje období,
 421 kdy je daný fakt v modelovém světě pravdivý. Posledním termínem je *čas transakce*, která
 422 definuje přítomnost faktu v databázi a možnost jej získat. Čas transakce a čas platnosti jsou
 423 na sobě nezávislé a definují dvě rozdílné časové osy, kdy každá může disponovat s jinou gra-
 424 nualitou. Souhrnný název pro výše uvedené tři pojmy, který se používá je systém časových
 425 razítek. Při použití těchto systému lze následně tvořit dotazy zaměřené na různá časová
 426 období, jako je minulost, přítomnost a budoucnost. Tyto dotazy jsou velmi jednoduché a to
 427 díky rozšířenému jazyku TSQL, který vychází z klasické podoby dotazovacího jazyka SQL.

428 Temporální databáze jsou rozděleny, podle systému časových razítek, na tyto základní:
 429 *snímková*, *transakční*, *platného času*, *obojího času* (bitemporální). Entity z databáze, která
 430 je použita v této práci, je možné zařadit do kategorie *snímkových tabulek* (snapshot). K
 431 jejím hlavním rysům patří zaznamenávání stavu dat v jistém okamžiku. Čas transakce ani
 432 čas platnosti zde nejsou uplatněny.

433 Konkrétní příklad možné transformace je ukázán na části tabulky *presence* 3.1, která
 434 se ve stejném formátu nachází i v databázi, a modifikované tabulce *presence_modify* 3.2.
 Tabulka 3.2 se od původní liší přidáním sloupce *dateEnd* (podbarven šedě), který s atri-

id	date	toj	presence
87365	2011-4-4 13:53:59	JabInfo@jabbim.cz	Available
87369	2011-4-4 15:43:07	JabInfo@jabbim.cz	Away
...

Tabulka 3.1: Ukázka tabulky *presence*.

435
 436 butem *dateStart* vymezuje interval, kdy daná hodnota byla nebo je platná. Tato entita
 437 je pouze příkladem jak by mohla daná transformace vypadat. V této práci se žádná nová
 438 tabulka nevytvářela a ani se nemodifikovala již vytvořená. Celá transformace je popsána v
 další části této podkapitoly.

id	dateStart	dateEnd	toj	presence
87365	2011-4-4 13:53:59	2011-4-4 15:43:07	JabInfo@jabbim.cz	Available
87369	2011-4-4 15:43:07	INF	JabInfo@jabbim.cz	Away
...

Tabulka 3.2: Ukázka modifikované tabulky *presence_modify*.

439
 440 Postup jednotlivých kroků při transformaci časových údajů z tabulky *presence*, je zob-
 441 razen v následujícím výčtu. Tento zjednodušený popis algoritmu vyžaduje dva vstupní

parametry. První je JabberID uživatele, jehož položky v databázi mají být transformovány. Druhá nutná položka je datum, které bude sloužit jako upřesňující vstupní interval.

Data z tabulky presence jsou transformována na vektor ϑ o 288 dimenzích, kde z pohledu časového je jedna dimenze období vymezené 5 minutami. Den je rozdělen na úseky po 5 minutách ($\delta = 300s$), kterých je 288. Tedy

$$\vartheta = (\vartheta_1, \vartheta_0, \dots, \vartheta_N),$$

kde $N = 288$.

1. Vstupním parametrem je datum, které vymezuje data pro transformaci. Toto datum je převedeno na dvě data (počátek a konec dne), která tvoří hraniční body v intervalu ι .
2. Výběr dat z databáze, která splňují časové období definované intervalem ι a uživatel ID . Uložení těchto dat do množiny Γ .
3. Pokud zadanému dotazu neodpovídá žádný řádek z tabulky, jsou data označena jako prázdná. Výstupní transformovaný vektor ϑ je naplněn hodnotami reprezentujícími stav *Unavailable*. Algoritmus je **ukončen**.
4. Zjištění prvního statusu toho dne.
 - 4.1 Převod data o den dřívejšího na interval ι_1 , například $\langle 2011-08-22\ 00:00:00, 2011-08-22\ 23:59:59 \rangle$.
 - 4.2 Výběr dat vyhovujícím intervalu ι_1 a uživateli ID z databáze.
 - 4.3 Výběr posledního záznamu z množiny dat získaných z předešlého dotazu.
 - 4.4 Nalezení poslední presence a uložení její hodnoty do λ .
5. Výběr následujícího záznamu z množiny Γ .
6. Výpočet zda je časový interval mezi vybraným a následujícím záznamem větší jak δ .
 - 6.a Časový interval je větší než interval δ .
 - 6.1 Do výsledného vektoru ϑ je ukládána hodnota presence daného záznamu.
 - 6.2 Opakuj předešlý bod **6.1** kolikrát je interval δ menší než rozdíl mezi časy vybraného a následujícího záznamu.
 - 6.b Časový interval je menší než interval δ .
 - 6.1 Jsou vybírány další záznamy z množiny Γ dokud rozdíl mezi časy v sekundách není větší než interval δ .
 - 6.2 Jednotlivé presence záznamů jsou ukládány do pomocného pole, transformované do kvantitativních hodnot, jak je uvedeno v úvodu této podkapitoly.
 - 6.3 Každý prvek pole je vynásoben počtem sekund, zastoupených v daném intervalu.
 - 6.4 Výběr největšího prvku z pomocného pole a uložení jej do výsledného vektoru ϑ .
7. Celý proces je opakován od bodu **5**, dokud množina Γ není prázdná.

3.2 Metody dolování dat

Základ metod dolování dat je založen na statistice, posledních poznatcích z umělé inteligence či strojového učení. Hlavní cíl těchto netriviálních metod je společný — snaha zjištěné výsledky prezentovat srozumitelnou formou. Pro většinu používaných metod je společná vlastnost předpoklad, že objekty popsané pomocí podobných charakteristik patří do stejné skupiny (učení na základě podobnosti *similarity-based learning*). Objekty obsahující atributy, lze převést na body v n -rozměrném prostoru, kde n reprezentuje počet atributů. Vychází se z představy podobnosti bodů tvořící určité shluky v prostoru.

Další rozdíly mezi metodami, které byly prezentovány v [2], spočívají ve:

- schopnosti reprezentace shluků (např. otázka lineární separability)
- srozumitelnosti nalezených znalostí pro uživatele (symbolické vs. subsymbolické metody)
- efektivnosti znovupoužití nalezených znalostí
- vhodnosti typů dat
- a další ...

Problémy, které data mining řeší, se rozdělují do několika skupin. Do výčtu vybraných z nich, které budou následně rozebrány, patří *asociační pravidla*, *klasifikace*, *modely*, *predikce* a *shlukování*.

Při popisu asociačních pravidel, která jsou založena na syntaxi *IF-THEN*, bylo čerpáno z [2]. Jejich rozšíření se datuje do 90. let 20. století, kdy byly panem Agrawalem představeny v souvislosti s analýzou „nákupního košíku“. Použitelnost bude vysvětlena právě na příkladu analýzy nákupního košíku. Podstata příkladu je tvořena zákazníkem a jeho systémem nakupování. Jsou zjišťovány produkty, které jsou nakupovány současně. Hledají se nebo-li jsou vytvářeny společné vazby (asociační pravidla) mezi výrobky a určuje se jejich spolehlivost. Na základě těchto závislostí je upravováno umístění jednotlivých výrobků. Obecně jsou tedy asociační pravidla považována za konstrukci, která z hodnot jedné transakce odvozuje možnost výskytu závislostí v jiných transakcích. Jsou tedy hledány všechny vnitřní závislosti existující mezi daty.

K dalším metodám pro data mining patří klasifikace, která bude opět vysvětlena na příkladu, převzatého z [8]. Podle obsahu databáze nebo dotazníku bude každý klient banky zařazen do různých krizových skupin. Na základě těchto skupin pracuje „credit skóring“, jež klientovi poskytne nebo odepře například úvěr v bance. Další příklady využití jsou například ve zdravotnictví. Na základě zdravotního stavu pacienta a jeho příznaků, je pacient zařazen do tříd, které reprezentují jednotlivé nemoci. Klasifikací jsou, podle [5], jednotlivé zkoumané elementy rozděleny (podle hodnot atributů) do vhodných kategorií, které jsou předem vytvořeny z navzájem podobných objektů (tvorba profilů třídy). Při této metodě je upřednostňována přesnost před jednoduchostí a rychlostí. Zdroje klasifikovaných objektů jsou většinou tvořeny jednotlivými řádky v databázi. Vzory dat vytváří instance, Jejichž vlastnosti reprezentují atributy vyjádřené číselnou hodnotou.

Na modelech je založen třetí typ metod pro dolování dat. Základem modelů jsou trénovací data. Dále uvedené příklady vybraných klasifikačních modelů, byly čerpány z [5]. Především jsou to rozhodovací stromy, neuronové sítě, statistické metody, klasifikační pravidla a další.

Metodam která je postavena na myšlence, kde chronologicky seřazená data a vývoj jejich hodnot v minulosti tvoří základ pro určení hodnot budoucích, se nazývá predikce. Je řazena mezi velmi známé procesy, které na základě získaných znalostí předpovídají následující vývoj. Předpokládá se, že na základě informací získaných z dat v minulosti, bude možné postavit modely, které se budou chovat stejně nebo alespoň podobně i v budoucnu. Využití naleznu v předpovědi počasí (z naměřených meteorologických hodnot se určují budoucí předpokládané teploty), při vývoji cen na burze a dalších. Podklady pro popis predikce byly čerpány z [2, 5].

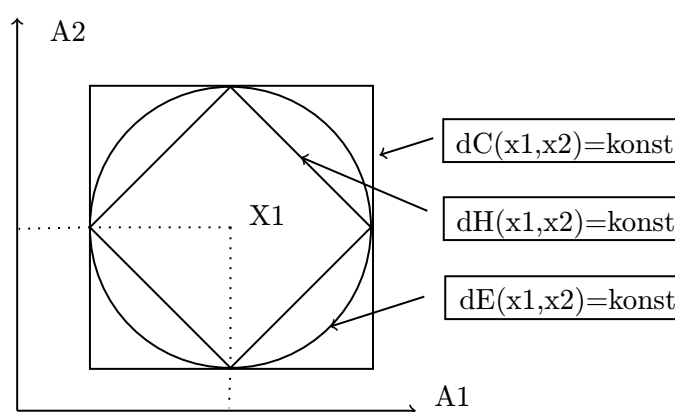
Poslední zde uvedená metoda je zaměřená na dělení objektů do předem neznámých skupin. Proces dělení probíhá na základě specifikace objektů a jejich odlišnosti od ostatních shluků. Tato část, pro kterou bylo čerpáno z [27, 3], bude podrobně rozebrána v následující podkapitole.

3.3 Shlukování

V této podkapitole je shlukování rozděleno na několik metod shlukové analýzy podle [5]. U každé z nich jsou popsány její základní vlastnosti a uvedeny algoritmy relevantní k práci. Poslední metoda *metoda rozkladu* je rozebrána podrobněji z důvodu jejího praktického využití v této práci.

Shlukování je zaměřeno na dělení objektů do předem neznámých skupin. Proces dělení probíhá na základě specifikace objektů a jejich odlišnosti od ostatních shluků.

Většina níže popsaných metod a algoritmů je založena na výpočtu vzdáleností mezi objekty. Tato vzdálenost lze vyjádřit různými mírami, podle knihy [2] například pomocí *Hammingovy vzdálenosti* (dH), *Euklidovské vzdálenosti* (dE) a *Čebyševovy vzdálenosti* (dC). Rozdíl mezi těmito typy určující vzdálenosti, graficky vyjadřuje obrázek 3.2. Kde $X1$ je střed, od něhož jsou jednotlivými obrazy znázorněny dané vzdálenosti. Konkrétně pomyslné body umístěné po obvodu kruhu jsou všechny stejně vzdáleny od středu $X1$. Tato vzdálenost je označena jako Euklidovská. Další 2D těleso čtverec, který je vodorovný s osami $A1$ a $A2$ prezentuje Čebyševovu vzdálenost. Po obvodu posledního obrazce, čtverce otočeného o 45° podle osy $A1$, jsou všechny pomyslné body stejně vzdáleny od bodu $X1$ Hammingovou vzdáleností.



Obrázek 3.2: Srovnání výpočtu vzdáleností od bodu x_1 [2].

Jako první jsou zde rozebrány *metody založené na modelu*, které se pokouší přiřadit

547 data k určitému matematickému modelu na základě společných optimalizovaných vlastností.
548 Většina procesů je založena na předpokladu, že jsou data generována pomocí standardních
549 statistik. Mezi zástupné metody této shlukovací analýzy se řadí Expectation–Maximization
550 (EM) a Self Organizing Oscillator Network, dále jen SOON. Algoritmus SOON je založen na
551 neuronové síti. Je to metoda vycházející z algoritmu SOM⁴ [27]. Metoda EM je rozšířením
552 algoritmu *k-means*, který bude podrobně rozebrán v následující části.

553 Hlavní princip *metody hierarchického shlukování* je založen na tvorbě stromové hierar-
554 chie shluků, která je známá pod názvem *dendrogram*. Hierarchické metody, podle [5], mohou
555 být rozděleny do dvou skupin a to na základě principu, kterým jsou dendrogramy vytvářeny.
556 První možnost je *aglomerativní přístup*, který shlukuje menší shluky, kdy výsledkem je jen
557 jeden. Druhý přístup, *divizní*, je založen na opačném předpokladu. Na počátku je tedy jeden
558 velký shluk, který je postupně rozdělován dokud není počet shluků roven počtu objektů
559 [27]. Mezi zástupce této metody například patří algoritmus AGNES⁵.

560 K dalším metodám patří *metody založené na mřížce*, které kvantují datový prostor do
561 konečného počtu pravoúhlých buněk. Tyto buňky jsou uspořádány do víceúrovňové mříž-
562 kové struktury. Zmíněná struktura tvoří základ pro shlukové operace. Hlavní výhoda tohoto
563 přístupu je rychlost zpracování, které většinou nebere ohled na počet datových objektů. Čas
564 zpracování závisí pouze na počtu buněk v každé dimenzi kvantovaného prostoru. Mezi zá-
565 stupce metod založených na mřížce patří metoda STING — STatistical INformation Grid,
566 který pracuje se statickými informacemi uloženými v buňkách mřížky. Algoritmus je roz-
567 dělen do dvou částí. První si klade za cíl rekurzivně rozdělit datový prostor na pravoúhlé
568 buňky. Druhá fáze testuje spojitost mezi sousedy relevantních buněk [27]. Mezi další metody
569 založené na mřížce patří WaveCluster⁶, využívající vlnkové transformace k rozdělení pro-
570 storu dat. Tato transformace zdůrazňuje shluky v prostoru a objekty jím vzdálené potlačuje
571 [5].

572 *Metody založené na hustotě* vychází z *m*-rozměrného prostoru, ve kterém jsou zobra-
573 zeny objekty ve formě bodů. Místa v prostoru s větší koncentrací objektů ve srovnání s
574 ostatními oblastmi jsou nazývány shluky. Výchozí předpoklad je existence okolí jednotli-
575 vých bodů (sousedství). Jedna z charakteristik metod založených na hustotě je schopnost
576 vypořádat se s vzdálenými hodnotami, označovanými jako šum [27]. Jako příklad je uvedena
577 metoda DBSCAN⁷, která je založena na hustotě objektů v prostoru. U jednotlivých objektů
578 je zkoumáno jejich okolí. Algoritmus je ovlivňován dvěma parametry, velikostí shluku ε a
579 minimálním počtem objektů v daném shluku *MinPts*, které spolu úzce souvisí (viz [5]).
580 Bod splňující obě podmínky je označen za jádro. Za pomoci jader je rozšiřována množina
581 objektů spojených na základě hustoty. Obsahuje-li jádro x_1 ve svém okolí další jádro x_2
582 znamená to, že jádro x_1 je přímo dosažitelné z jádra x_2 . Tímto způsobem jsou vytvářeny
583 výsledné *shluky*. V opačném případě, body, které nesplňují dvě zmíněné podmínky, jsou
584 označeny jako *šum*.

585 Všechny zde doposud zmiňované metody poskytují dobré výsledky pouze s malým poč-
586 tem dimenzí, tak jak je to popsáno v [8]. S narůstajícím počtem atributů roste počet
587 nerelevantních dimenzí určených pro shlukování. S tímto také přibývá zvětšená produkce
588 zašumění a znesnadnění nalezení relevantních shluků. Data jsou roztroušena do mnoha di-
589 menzí a tím odpadá možnost použití vzdálenostních funkcí. Zmíněné problémy shlukování
590 velkých dat řeší dvě techniky *metoda transformace rysů* a *metoda výběru atributů*. Pro

⁴Self-Organizing Map

⁵AGglomerative Nesting

⁶Clustering Using Wavelet Transformation

⁷Density-Based Spatial Clustering of Applications with Noise

591 efektivní shlukování je možné použít například algoritmus CLIQUE⁸.

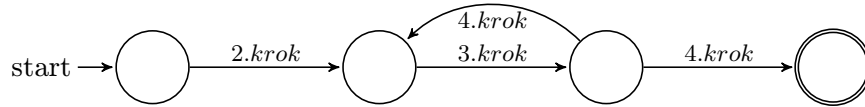
592 Metody rozkladu

593 Metody rozkladu rozdělují datové prvky do několika podmnožin, nazývané shluky. Počet
594 shluků musí být znám před zahájením samotného procesu. Přiřazení do konkrétních tříd
595 je, podle [27], jednoznačné nebo probíhá na základě míry příslušnosti objektů do shluků.
596 Pro velký počet objektů, se kterými se pracuje, jsou využívány různé iterační optimalizace.

597 Hlavním zástupcem u uvedených metod je algoritmus k -means, který je popsán níže.
598 Tvoří základ pro většinu metod shlukování nejen pro metody rozkladu. K dalším metodám
599 se řadí k -medoidů, k -modů, k -histogramů, fuzzy shluková analýza a další.

600 k -means

601 Shlukování pomocí algoritmu k -means je používáno pro data obsahující kvantitativní pro-
602 měnné a pro data, která nejsou příliš zašumělá. Základní proces je tvořen iterativním roz-
603 dělováním objektů do tříd na základě vzdáleností od jejich středů. Střed nebo-li centroid
604 shluku je vektor, jehož vzdálenost od součtu vzdáleností objektů v této třídě je minimální.
605 Celý tento proces je prezentován na obrázku 3.3 pomocí jednoduchého schématu konečného
606 automatu. Jednotlivé kroky konečného automatu odpovídají krokům k -means zobrazeného
pomocí algoritmu 3.1. Pro výpočet vzdáleností mezi objekty samotnými nebo mezi objekty



Obrázek 3.3: Algoritmus k -means zobrazený pomocí konečného automatu.

607 a středem je použita euklidovská vzdálenost⁹, která je vyobrazena na obrázku 3.2.

609 K hlavním výhodám algoritmu k -means patří jeho relativní efektivnost. Složitost algo-
610 ritmu je $O(TKN)$, kde N je počet objektů, K je počet shluků a T je počet iterací. Obvykle
611 platí, že počet objektů je mnohem větší než počet iterací i shluků. Na druhou stranu má i
612 řadu nevýhod, kvůli kterým je často různými způsoby modifikován (k -medoids, k -medians).
613 K hlavním „nedostatkům“ patří předem nutná znalost počtu shluků (tříd) K , do kterých
614 budou objekty zařazeny. Druhý často se vyskytující problém je samotné ukončení algo-
615 ritmu, které nastane u nalezení lokálního optima namísto optima globálního. Tato nepřes-
616 nost vzniká nevhodně zvoleným rozmístěním počátečních středů. Původní nemodifikovaná
617 verze algoritmu nedefinuje jak se má postupovat, jsou-li nalezeny prázdné shluky.

618 K -menas je algoritmus, kterým jsou přiřazovány objekty (vektory) x_n , kde $n = 1, \dots, N$,
619 do S_k , kde $k = 1, \dots, K$, shluků. V prvním kroku jsou určeny počáteční středy tříd, do kterých
620 se budou objekty shlukovat. Určení počátečních centroidů c_k probíhá například náhodným
621 výběrem K objektů nebo K prvních objektů souboru. Druhým krokem jsou zkoumány
622 jednotlivé vzdálenosti objektů x_n od počátečních středů c_j pomocí euklidovské vzdálenosti.
623 Na základě nejmenší zjištěné vzdálenosti mezi objektem a centroidem je objekt zařazen
624 do shluku, kterému náleží právě tento střed. Ve třetím kroku, tak jako u kroku prvního,
625 jsou hledány nové středy shluků. Nyní již však nejsou zvoleny náhodně, ale spočítány. Jsou

⁸CLustering In QUest

⁹mean = střed, centroid je vektor průměrů

vypočítány na základě průměrných jednotlivých hodnot objektů a uložen jako m -rozměrný vektor. Čtvrtým krokem se algoritmus dostává do konečné fáze, kdy mohou nastat dva možné případy. Nově nalezené středy nejsou příliš vzdáleny od předchozích centroidů a proto je algoritmus ukončen. Druhá častěji se vyskytující možnost iterativně provádí algoritmus od druhého kroku dokud neplatí první možnost nebo dokud se objekty nepřestanou přemísťovat úplně. Při popisu tohoto algoritmu bylo čerpáno z [27, 2]. Níže zobrazený algoritmus 3.1 prezentuje krok po kroku metodu k -means.

-
1. náhodně zvol rozklad do K shluků
 2. urči centroidy pro všechny shluky v aktuálním rozkladu
 3. pro každý příklad x
 - 3.1 urči vzdálenosti $d(x, c_k)$, $k = 1, \dots, K$, kde c_k je centorid k -tého shluku
 - 3.2 nechť $d(x, c_l) = \min_k d(x, c_k)$
 - 3.3 není-li x součástí shluku l (k jehož centoridu c_l má nejblíže), přesuň x do shluku l
 4. došlo-li k nějakému přesunu, potom jdi na 2, jinak konec
-

Algoritmus 3.1: Metoda k -means byla převzata z [2].

Díky jednoduchosti a relativní rychlosti je metoda k -means stále výrazně využívána. Uplatnění nachází v široké škále oblastí jako je například biologie nebo počítačová grafika. Vzhledem k enormnímu počtu možného uspořádání nejsou výsledky vždy přesné, ale často pouze přibližné.

3.4 Programy

V současné době na programovém trhu existuje mnoho systému, které jsou zaměřeny na data mining. Mezi nejrozšířenější a nejdostupnější nástroje patří Weka a RapidMiner. K těmto nástrojům je také možné zařadit program FIT-miner vyvíjený na fakultě informačních technologií v Brně. V této práci byl pro samotný data miningg využit program RapidMiner, který dostal přednost před ostatními. Z pohledu nástroje FIT-miner, který ve své základní části podporuje z databází pouze Oracle, se RapidMiner jevil jako vhodnější. Kompatibilitu pro databáze typu PostgreSQL již měl zabudovanou a tak nebylo potřeba vyvíjet žádné doplňující moduly, jak by to bylo u FIT-mineru. V případě nástroje Weka, RapidMiner působil propracovanějším dojmem a také nabízí lepší grafické zobrazení vyhodnocených výsledků.

Dalším velmi rozšířeným a často používaným nástrojem je jazyk R . R vychází z jazyka S , který ale není jako jazyk R volně šiřitelný. Statistický a grafický nástroj R je tedy volně dostupným jazykem a prostředím, které je ovládáno pouze z příkazové řádky. Pro jednodušší práci jej lze rozšířit o grafické rozhraní jako je RKWard nebo R Commander. Samotnou aplikaci lze rozšířit o mnoho statistických doplňků, které jsou taktéž zdarma.

Mnoho programů pro dolování dat je založena na přístupu vizuálního programování. Jedná se o proces, při kterém je uživatelem, za pomoci grafických prostředků, navržen algoritmus a další postup práce. Jako příklad lze uvést poloprofesionální aplikaci *Orange*, u které jsou nejdůležitější části psány pomocí C++ a rozšíření lze implementovat v jazyce Python.

Na vybraných technicky zaměřených vysokých školách existují skupiny, které se zabývá výzkumem a vývojem nástrojů pro data mining. Jako příklad lze uvést již dříve zmiňovaný FIT-miner z fakulty informačních technologií v Brně nebo také projekt LISp-Miner z

661 Vysoké školy ekonomické v Praze. LISp-Miner je otevřený akademický systém určený pro
662 výuku a výzkum metod pro dobývání znalostí z databází.

663 **RapidMiner**

664 RapidMiner je, tak jak je popsán na oficiálních stránkách produktu [26], celosvětově nej-
665 používanější open-source systém pro dolování dat. Je možné jej používat jako samotnou
666 aplikaci nebo jej začlenit jako komponentu do vlastních výrobků v podobě knihovny pro ja-
667 zyk Java. Pro zájemce je nabízen také ve verzích pro firmy, které jsou rozdílné v poplatcích,
668 podpoře pro zákazníka, záruce a dalších balíčcích služeb zajišťující celkovou komplexnost a
669 spolehlivost produktu.

670 Jak již většina podobných aplikací, je v současné době implementován v jazyce Java,
671 díky které nabízí flexibilní nejen grafické prostředí. K vybraným základním rysům toho
672 nástroje, tak jak jsou prezentovány firmou *Rapid-i*, patří: výkonné, přesto intuitivní grafické
673 uživatelské rozhraní pro návrh procesů, jednoduché řešení pro transformaci dat, kontrola
674 výsledků již při samotném návrhu a další. Nástroj RapidMiner podporuje širokou škálu
675 metod a algoritmů pro data minig. Mnoho algoritmů je implementováno přímo v aplikaci,
676 ale také je použito metod z konkurenčního softwaru Weka. V základní verzi určené pro
677 veřejnost je k nalezení přes 100 procesů k modelování. Jsou zde zastoupeny jak metody
678 klasifikační a asociační, tak i metody shlukovací z nichž lze jmenovat například DBSCAN,
679 k -medoids a hlavně k -means.

680 K dalším schopnostem RapidMineru je možnost spuštění jeho samotného pomocí gra-
681 fického rozhraní nebo z příkazové řádky. Jak již bylo uvedeno dříve, je také možné jej
682 použít jako knihovnu v jazyce Java. V této práci jsou použity první dvě možnosti. Pomocí
683 grafického prostředí byl vytvořen experiment, otestována jeho funkčnost a následně pro
684 jednotlivá shlukování použita šablona procesu, která byla volána z příkazové řádky. Tato
685 možnost je k dispozici díky tomu, že jsou projekty v programu RapidMiner ukládány do
686 čitelné a strukturované formy za pomoci značkovacího jazyka xml.

687 Kapitola 4

688 Implementace

689 Obsahem čtvrté kapitoly je popis praktické části této práce. Jsou zde charakterizovány
690 jednotlivé prvky, které byly použity jak pro získání dat, tak pro jejich následné uložení.

691 Cílem této práce je dolování dat z Jabberu. Jak již bylo dříve napsáno, Jabber je real-
692 time komunikační služba díky níž mohou její uživatelé komunikovat, informovat nebo sdílet
693 svůj status s jinými uživateli. Celá tato vzájemná komunikace skrývá nepřeborné množství
694 informací o klientech dané sítě. Všechna tato navzájem vyměňovaná nebo poskytnutá data
695 následně poslouží jako zdroj samotnému dolování. Pro jejich uskladnění je využita databáze,
696 jejichž strukturální návrh prezentuje obrázek 4.1.

697 Druhá část této kapitoly je zaměřena na Jabber klienta nebo-li robota, který je im-
698 plementován v jazyce C++ pouze s konzolovým rozhraním. Robot v této práci hraje roli
699 pasivního uživatele, který informace pouze přijímá. Ve vybraných případech dokáže uživa-
700 tele ze svého seznamu kontaktů vyzvat k zaslání odpovědi s informacemi na odpověď, o
701 kterou žádal. Struktura samotného robota je popsána níže a reprezentována obrázkem ??.
702 Na konci části o Jabber robotovi je zmíněno o knihovně *gloox*, kterou je zprostředkovány
703 všechny náležitosti Jabber komunikace.

704 4.1 Databáze

705 Data, která jsou sbírána robotem, jsou ukládána do objektově-relační databáze Postgre-
706 SQL¹. PostgreSQL nebo-li také Postgres byl použit ve verzi 8.4.7 a je provozován na ope-
707 račním systému Ubuntu Maverick verze 10.10.

708 Návrh databáze

709 Struktura databáze, do které je ukládána veškerá komunikace Jabber robota, využívá rela-
710 ční model. Obrázkem 4.1 jsou prezentovány nejdůležitější části databáze. Celou strukturu
711 návrhu je možné nalézt v příloze F. V jednotlivých částech návrhu databáze je počítáno s
712 jejím druhotným využitím obsahu, které bude popsáno v dalších oddílech této práce.

713 V době návrhu databáze zcela nebylo zřejmé, která data budou následně analyzována. Z
714 toho důvodu se struktura databáze snaží zachytit všechna „důležitá“ data. Předem nebylo
715 určen typ znalostí, kterých by se při dolování mělo dosáhnout. Za účelem uchování dat, je
716 v návrhu databáze obsažena tabulka *xml*, která je nositelem obsahu jak všech přijatých,
717 tak i odeslaných zpráv. Tabulky *debug*, *level* a *logarea*, které v zúženém návrhu databáze,

¹vyvinul se z projektu Ingres

718 uvedeném na obrázku 4.1, nejsou zobrazeny, jsou určeny pouze jako doplňkové informace
719 k typu zprávy. Příkladem může být řádek v tabulce *xml* a sloupcem *area* s hodnotou 32,
720 kterým jsou označeny zprávy spojené s překladem IP adres a doménových jmen. Hodnota
721 *level* je pro většinu zpráv nastavena na hodnotu nula, což znamená že nejde o zprávu
722 obsahující chybu ani varování. Z pohledu návrhu vytváří tabulka *xml* redundantní data,
723 dalo by se tedy říct, že je nadbytečná. Je možné ji nahradit jednotlivými dalšími tabulkami,
724 které jsou zaměřeny na konkrétní data. Z druhého pohledu je ale patrné, že tabulka *xml*
725 v atributu *message* obsahuje vždy ucelenou jednu zprávu i s elementy, kterými je obalen
726 samotný obsah. Příkladem entity, která již obsahuje konkrétní data bez *xml* prvků, je
727 tabulka *message*, která je taktéž vyobrazena v příloze E. Jejím obsahem jsou zprávy vzniklé
728 při Jabber komunikaci mezi uživatelem a robotem, jehož schopnosti budou popsány níže.

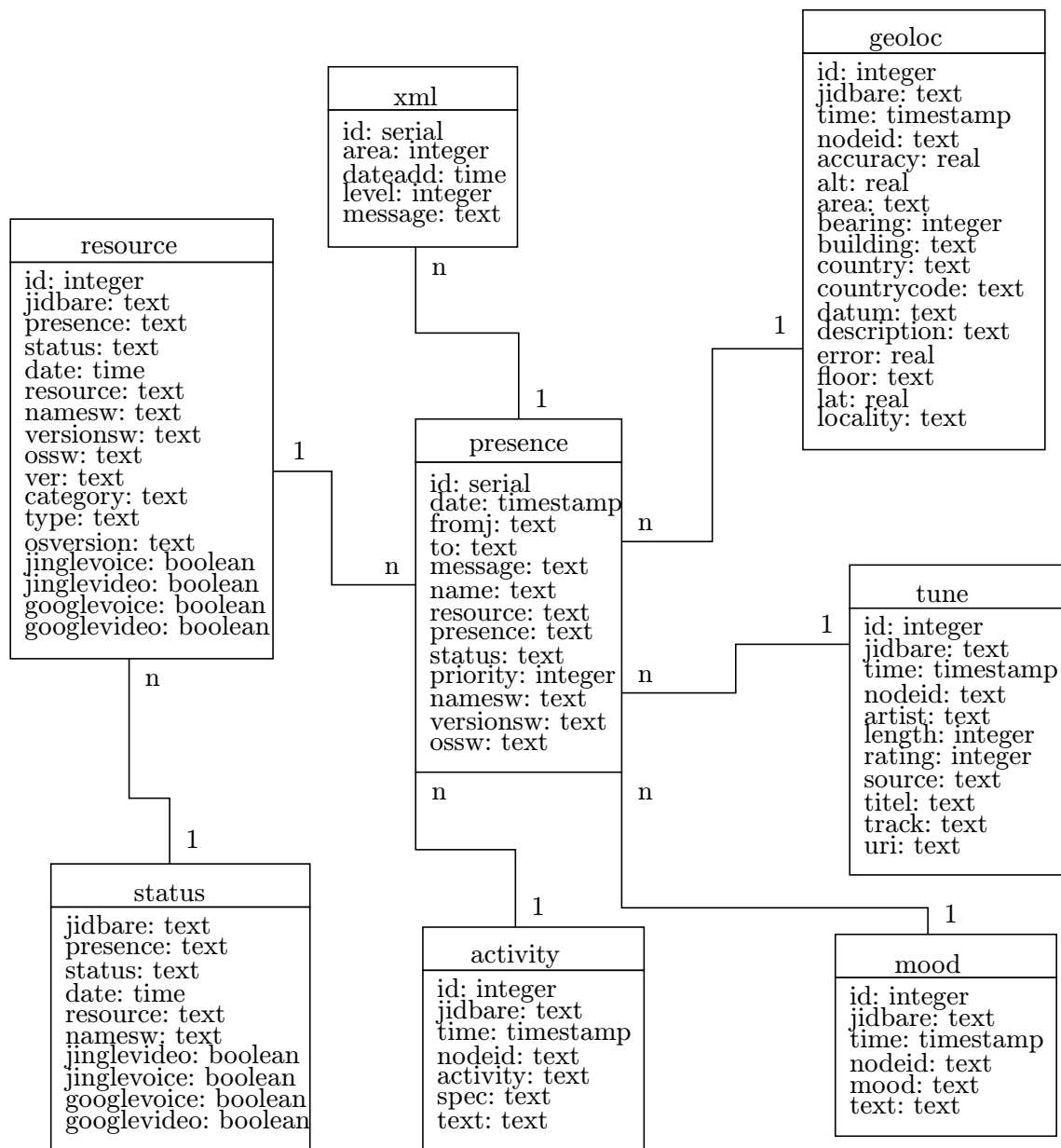
729 Tabulka *presence* z pohledu XMPP standardu prezentuje jeden ze tří základních částí
730 stanzy, element *presence*. Základním cílem této tabulky je shromažďování uživatelských
731 statusů. Tento požadavek je splněn atributem *presence*. Jak již bylo zmíněno ve třetí ka-
732 pitole, v části která se zabývá transformací, atribut *presence* obsahuje hodnoty typu text.
733 Příklad všech možných hodnot, kterých tento atribut může nabývat je prezentován v pří-
734 loze E v části zabývající se elementem *presence*. K dalším atributům této tabulky patří
735 *message*, který je reprezentován textovým řetězcem a rozšiřuje informace o stavu uživatele.
736 V této části je často obsažen text, který je do statusu přidán automaticky IM klientem.
737 Jako příklad lze uvést větu „I’m not here right now“ generována aplikací Miranda po delším
738 časovém úseku nečinnosti. Transformovaný obsah této entity tvoří důležitou část pro další
739 krok v KDD a to v podobě dat, ze kterých budou „dolovány“ informace.

740 Většina rozšíření standardu XMPP, která jsou popsána ve druhé kapitole, jsou pre-
741 zentována pomocí pěti tabulek. Konkrétně to jsou tabulky *geoloc*, *tune*, *mood*, *activity* a
742 *vcard*. Poslední rozšíření informující o verzi a jménu IM klienta a systému bude rozebráno
743 v následujícím odstavci v souvislosti s další tabulkou *resource*. Pro obsah a názvy atributů
744 všech pěti tabulek s rozšířeními se staly vzory dokumenty jednotlivých XEP, které je defi-
745 nují. Podrobnější popis všech atributů lze tedy nalézt právě v těchto standardech. Tabulky
746 kromě těchto atributů obsahují také položku *jidbare*, která, jak již název naznačuje, obsa-
747 huje pouze čisté JabberID bez *resources*. Tato skutečnost vyplývá již ze samotného návrhu
748 XEP protokolů. Adresa uživatele, který posílá serveru zprávu s jedním z možných rozšíření,
749 je také pouze v základním stavu. V tabulce *vcard* jsou také položky, jejichž obsah je tvořen
750 fotografiemi. Tyto atributy ale nejsou binárního typu, tak jako obrázky samotné, nýbrž
751 textového. Obrázky jsou zde uloženy ve stavu, tak jak jsou přenášeny po síti, tedy pomocí
752 datového formátu Base64. Base64 je algoritmus, který převádí binární data na řetězec, který
753 je tvořen pouze znaky ASCII tabulky.

754 S posledním rozšířením nazvaném *Software version* je spjata tabulka *resource*. Její něko-
755 lik sloupců tvoří právě informace o IM klientovi a operačním systému, na kterém běží. Tyto
756 základní údaje jsou definovány v protokolu [17], kde je také možné čerpat bližší informace.
757 Pokročilejší atributy, jako je *type* a *osversion* nacházejí svůj podklad v XEP dokumentu [7]
758 taktéž jako atribut *ver*, který je podrobněji popsán ve druhé kapitole této práce. V nepo-
759 slední řadě se zde nachází zmínka o podpoře audia a videa ať už v podobě *jingle* rozšířené
760 nebo *google*. tyto sloupce jsou typu boolean, tudíž hodnota *true* znamená podporu daného
761 sloupce. Přesto tyto výše uvedené údaje nejsou hlavním důvodem existence této tabulky.
762 Její primární cíl je uchovávat informace o připojených uživateli a všech jejich *resources*.

763 Entita *resources* tvoří základ pro tabulku *status*. Její obsah je automaticky generován z
764 obsahu tabulky popsané výše. Počet řádků odpovídá počtu uživatelů, kteří jsou v seznamu
765 kontaktů tohoto robota. Primárním cílem je informování o aktuálním stavu uživatele. Je

766 tedy poskytován stav, ve kterém se uživatel nachází s přihlédnutím na prioritu a resources.
 767 Řádek entity obsahuje status a dostupnost uživatele a jeho resources s největší přihlášenou
 768 prioritou. Nastane-li situace, že má uživatel více připojení pod jedním účtem je vzato v
 potaz přihlášení s nejnovější změnou presence.



Obrázek 4.1: Vybraná část struktury databáze.

770 4.2 Architektura

771 4.3 Robot

772 V této části bude popsána aplikace, která Jabber komunikaci s ostatními entitami real-time
773 sítě zajišťuje. Tento konzolový robot vyvíjený pro operační systém Linux také vhodným způ-
774 sobem pracuje s databází, kam jsou jednotlivá získaná data ukládána. Jabber, jádro síťové
775 komunikace [22], je implementováno za pomoci volně dostupné knihovny *gloox*. V porovná-
776 ním s jinými knihovnami disponuje lepší podporou a dokumentací. Gloox plně podporuje
777 standart XMPP Core [22] a z větší části i standard XMPP IM [23]. Dodatečně je plně
778 podporováno kolem třiceti XEP standardů například vcard-temp [16] a další.

779 Při implementaci robota byly využity základní prvky objektově-orientovaného progra-
780 mování, jako jsou například třídy a dědičnost. Každá třída zde zastupuje jednoznačně oddě-
781 litelné elementy robota. Ať už se jedná o blok komunikující s databází, různá rozšíření nebo
782 o jádro robota. Všechny zde jmenované objekty hrají důležitou roli jak v části dekompozice
783 na menší ucelené celky tak i v celkové komplexnosti programu. Jako příklad lze uvést třídy,
784 které reprezentují jednotlivá pro tuto práci vhodná rozšíření, jejichž základ je v podobě
785 XEP dokumentů.

786 Návrh bota

787 Struktura návrhu tříd robota, která je v základní části prezentována obrázkem ??, je velmi
788 podobná struktuře návrhu databáze, obrázek 4.1. Mnoho atributů z robota a z databáze jsou
789 ve vztahu jedna ku jedné. Samotná komunikace a řízení mezi těmito entitami je zprostřed-
790 kováno pomocí knihovny *libpq*. Libpq je programátorské rozhraní pro databázi PostgreSQL.
791 Je to sada knihovnických funkcí v jazyce C, které umožňují klientským programům zadávat
792 dotazy na server a následně na ně obdržet odpovědi. Pomocí této knihovny je řízen přenos
793 dat mezi robotem, který sbírá data, a databází, kam jsou ukládána. Navázání spojení a
794 následné jednotlivé dotazy probíhají v blokujícím režimu. Údaje potřebné k navázání spo-
795 jení mezi robotem a databází jsou uloženy v souboru *connect.h* v podobě předdefinovaných
796 konstant například *DB_USER*. V tomto dokumentu jsou uvedeny všechny údaje nutné k
797 navázání spojení ať už s databází nebo i s komunikačním Jabber serverem. Tudíž se stává
798 hlavní částí určenou k editaci připojení a Jabber účtu na němž robot běží. K dalšímu pouze
799 textovému dokumentu patří *const.h*, který je zaměřen k definování jednotlivých příkazů
800 a dotazů pro databázi. Jsou to příkazy pro tvorbu a aktualizaci tabulek, které jsou tak-
801 též definované jako konstanty. Identifikační názvy, pro snadné odlišení, jsou psány velkými
802 písmeny a začínají textem *DB_*.

803 Třída, která využívá zde doposud popsané konstanty a rozhraní libpq, je nazvaná *Data-*
804 *base*. Obsahuje mnoho funkcí, které jsou přetěžovány, a které zajišťují samotnou provázanost
805 mezi daty a databází. Pomocí těchto funkcí jsou data získaná z komunikace přenášena a
806 ukládána do jednotlivých tabulek databáze. Pro zahájení samotné práce s databází slouží
807 funkce *start()*, která zajistí připojení k databázi a provede nutné inicializační kroky. Jako
808 je tvorba nových tabulek, při prvním zapnutí aplikace, nebo kontrola existence těchto entit.

809 Pro rozšíření podle dokumentů XEP popsaných v kapitole druhé, v části zabývající
810 se relevantními rozšířeními, je základ nadtřída *Extension*. Tato třída obsahuje základní
811 funkce a parametry pro všechna rozšíření. Jako příklad lze uvést uživatelské jméno klienta,
812 který dané rozšíření „šíří“ po síti. Obsahuje také dvě čistě virtuální metody, sloužící k
813 rozparsování části xml zprávy a k vyčištění proměnných. Třídy, které z *Extension* dědí a jsou

814 prezentovány na návrhu robota uvedeném na obrázku ??, jsou následující: *Geoloc*, *Tune*,
815 *Mood*, *Activity* a *SwVersion*. Samotnou strukturou těchto rozšíření, jak již bylo uvedeno
816 výše, jsou pomocí parametrů kopírovány vlastnosti jednotlivých XEP dokumentů. Tedy
817 ke každému atributu xml zprávy existuje jak proměnná uchovávající její obsah tak i tzv.
818 „get“ a „set“ metoda. Pomocí virtuální metody jsou jednotlivé části rozšíření rozparsovány,
819 uloženy do tříd a následně vloženy do databází. Poslední doposud nezmiňované rozšíření v
820 podobě *vcard*, je využito z knihovny gloox, která jej implementuje.

821 Základní část, kterou je možné považovat za jádro aplikace, je třída *Bot*. Tato třída je
822 založena na již zmiňované C++ knihovně gloox. Je implementována za pomoci vybraných
823 tříd, ze kterých dědí. Jsou to především „handler“ třídy, kterými je zajišťován přístup k
824 jednotlivým zprávám. Nebo-li jejich prostřednictvím je získán přístup k samotným elemen-
825 tům stanzy, jako je message, iq, a presence, jejichž vlastnosti a využití jsou popsány v druhé
826 kapitole. Konkrétní seznam tříd, ze kterých je děděno, zobrazuje návrh robota na obrázku
827 ??. Jako příklad lze uvést třídu, z prostoru jmen gloox, *VcardHandler*, díky níž je možné
828 získat a zpracovávat vcard uživatelů, kteří jsou v seznamu kontaktů robota. Samotné přidá-
829 vání uživatelů do seznamu kontaktů je prováděno automaticky. Pouze na straně klienta je
830 nutné požádat robota o autorizaci, která však proběhne automaticky. Jednotlivé kontakty
831 nejsou žádným způsobem tříděny do skupin, tudíž existuje pouze jedna.

832 Rozšíření

833 ==zpravy, odpovida reaguje na to a na to

834 ==pouziti castecneho komentovani podle stylu doxygenu, makefile

835 Kapitola 5

836 Vyhodnocení výsledků

837 5.1 Manuální rozbor dat

838 –charakter dat, jaka data jsem nasbiral –popst jak jsem data prochael rucne, tedy manualni
839 dataming, pocet zazanmu, delka behu programu, velikost datatabze, pocet uzivatelu...

840 Kapitola 6

841 Závěr

842 POKACOVANI =tridit lidi do podskupin v kotakt listu, naprikład podle zpravy zaslene s
843 dotazem k autorizaci=: kazda skupina by mohla mít jin prava, jiné dotazy an služby...mozne
844 rozšířit o různé služby, které poskytují server, podle dotazu....

845

Literatura

- [1] Adams, D.: *Programming jabber*. Sebastopol: O'Reilly, první vydání, 2002, 455 s., ISBN 05-960-0202-5.
- [2] Berka, P.: *Dobývání znalostí z databází*. Praha: Academia, první vydání, 2003, 366 s., ISBN 80-200-1062-9.
- [3] Bramer, M.: *Principles of Data mining*. London: Springer, první vydání, 2007, 343 s., ISBN 18-462-8765-0.
- [4] Fayyad, U. M.; Smyth, P.: *Advances in knowledge discovery and data mining*. California: MIT Press, první vydání, 1996, 611 s., ISBN 02-625-6097-6.
- [5] Han, J.; Kamber, M.: *Data mining : concepts and techniques*. San Francisco: Morgan Kaufmann Publisher, druhé vydání, 2006, 770 s., ISBN 15-586-0901-6.
- [6] Hildebrand, J.; Saint-Andre, P.: XEP-0080: User Location. [online], 15-09-2009, [cit. 16. května 2011].
URL [<{http://xmpp.org/extensions/xep-0080.html}>](http://xmpp.org/extensions/xep-0080.html)
- [7] Hildebrand, J.; Saint-Andre, P.; Tronçon, R.; aj.: XEP-0115: Entity Capabilities. [online], 26-02-2008, [cit. 16. května 2011].
URL [<{http://xmpp.org/extensions/xep-0115.html}>](http://xmpp.org/extensions/xep-0115.html)
- [8] Hruška, T.: *Informační systémy : IIS/PIS*. Brno: Fakulta informačních technologií, 2008, 14733 s.
- [9] Kolektiv autorů: Extensible Markup Language (XML) 1.0. [online], 26-11-2008, [cit. 16. května 2011].
URL [<{http://www.w3.org/TR/2008/REC-xml-20081126/}>](http://www.w3.org/TR/2008/REC-xml-20081126/)
- [10] Kosek, J.: *XML pro každého : podrobný průvodce*. Praha: Grada, první vydání, 2000, 163 s., ISBN 80-716-9860-1.
- [11] Meijer, R.; Saint-Andre, P.: XEP-0108: User Activity. [online], 29-10-2008, [cit. 16. května 2011].
URL [<{http://xmpp.org/extensions/xep-0108.html}>](http://xmpp.org/extensions/xep-0108.html)
- [12] Millard, P.; Saint-Andre, P.; Meijer, R.: XEP-0060: Publish-Subscribe. [online], 12-07-2010, [cit. 16. května 2011].
URL [<{http://xmpp.org/extensions/xep-0060.html}>](http://xmpp.org/extensions/xep-0060.html)

- 876 [13] Mizzi, S.; Saint-Andre, P.: XEP-0292: vCard4 Over XMPP. [online], 02-26-2008, [cit.
877 16. května 2011].
878 URL <{http://xmpp.org/extensions/xep-0292.html}>
- 879 [14] Moore, D.; Wright, W.: *Jabber developer's handbook*. Indianapolis: Sams Publishing,
880 první vydání, 2004, 487 s., iISBN 06-723-2536-5.
- 881 [15] Nemrava, M.; Pospíšil, J.: Dolování dat a jeho aplikace. [online], 2006, [cit. 16. května
882 2011].
883 URL <{http://www.spatial.cs.umn.edu/paper_ps/dmchap.pdf}>
- 884 [16] Saint-Andre, P.: XEP-0054: vcard-temp. [online], 07-16-2008, [cit. 16. května 2011].
885 URL <{http://xmpp.org/extensions/xep-0054.html}>
- 886 [17] Saint-Andre, P.: XEP-0092: Software Version. [online], 02-15-2007, [cit. 16. května
887 2011].
888 URL <{http://xmpp.org/extensions/xep-0092.html}>
- 889 [18] Saint-Andre, P.: XEP-0118: User Tune. [online], 30-01-2008, [cit. 16. května 2011].
890 URL <{http://xmpp.org/extensions/xep-0118.html}>
- 891 [19] Saint-Andre, P.: XEP-0153: vCard-Based Avatars. [online], 16-08-2006, [cit.
892 16. května 2011].
893 URL <{http://xmpp.org/extensions/xep-0153.html}>
- 894 [20] Saint-Andre, P.; Meijer, R.: XEP-0107: User Mood. [online], 29-10-2008, [cit.
895 16. května 2011].
896 URL <{http://xmpp.org/extensions/xep-0107.html}>
- 897 [21] Saint-Andre, P.; Smith, K.: XEP-0163: Personal Eventing Protocol. [online],
898 12-07-2010, [cit. 16. května 2011].
899 URL <{http://xmpp.org/extensions/xep-0163.html}>
- 900 [22] Saint-André, P.: Extensible Messaging and Presence Protocol (XMPP): Core.
901 [online], 10-2004, [cit. 16. května 2011].
902 URL <{http://tools.ietf.org/html/rfc3920}>
- 903 [23] Saint-André, P.: Extensible Messaging and Presence Protocol (XMPP): Instant
904 Messaging and Presence. [online], 10-2004, [cit. 16. května 2011].
905 URL <{http://tools.ietf.org/html/rfc3921}>
- 906 [24] Saint-André, P.; Smith, K.; Troncon, R.: *XMPP : the definitive guide : building
907 real-time applications with jabber technologies*. Sebastopol: O'Reilly, první vydání,
908 2009, 287 s., iISBN 978-059-6521-264.
- 909 [25] WWW Stránky: Database Systems. [online], 2007, [cit. 16. května 2011].
910 URL
911 <{http://www.cs.uct.ac.za/mit_notes_devel/Database/Latest/index.html}>
- 912 [26] WWW Stránky: RapidMiner. [online], 2011, [cit. 16. května 2011].
913 URL <{http://rapid-i.com/content/view/181/190/}>
- 914 [27] Řezánková, H.; Húsek, D.; Snášel, V.: *Shluková analýza dat*. Praha: Professional
915 Publishing, druhé vydání, 2009, 218 s., iISBN 978-808-6946-818.

⁹¹⁶ **Příloha A**

⁹¹⁷ **Obsah CD**

⁹¹⁸ **Příloha B**

⁹¹⁹ **Manual**

⁹²⁰ **Příloha C**

⁹²¹ **Konfigurační soubor**

922 Příloha D

923 Slovník zkratek

924 **DNS** — Domain Name System

925 **GPG** — dkshckdsjvlsdjvodsvjdfokj

926 **IM služby** — dkshckdsjvlsdjvodsvjdfokj

927 **IP** — Internet Protocol

928 **JEP** — dkshckdsjvlsdjvodsvjdfokj

929 **JID** — dkshckdsjvlsdjvodsvjdfokj

930 **SASL** — dkshckdsjvlsdjvodsvjdfokj

931 **TCP** — dkshckdsjvlsdjvodsvjdfokj

932 **TLS** — dkshckdsjvlsdjvodsvjdfokj

933 **WWW** — dkshckdsjvlsdjvodsvjdfokj

934 **XEP** — dkshckdsjvlsdjvodsvjdfokj

935 **XML** — dkshckdsjvlsdjvodsvjdfokj

936 **XMPP** — dkshckdsjvlsdjvodsvjdfokj

937 **e-mail** — dkshckdsjvlsdjvodsvjdfokj

938 **jabber** — dkshckdsjvlsdjvodsvjdfokj

939 **klient** — dkshckdsjvlsdjvodsvjdfokj

940 **presence** — dkshckdsjvlsdjvodsvjdfokj

941 **server** — dkshckdsjvlsdjvodsvjdfokj

942 **stanza** — dkshckdsjvlsdjvodsvjdfokj

943 **vCard** — dkshckdsjvlsdjvodsvjdfokj

944 **ID3v1** — dkshckdsjvlsdjvodsvjdfokj

945 **ID3v2** — dkshckdsjvlsdjvodsvjdfokj

946 **ASCII** — dkshckdsjvlsdjvodsvjdfokj

947 **utf-8** — dkshckdsjvlsdjvodsvjdfokj

948 **Base64** — dkshckdsjvlsdjvodsvjdfokj

949 Příloha E

950 Stanza - základní schéma

951 Přehled základních elementů, které jsou využívány při Jabber komunikaci. Struktura jed-
952 notlivých částí stanzy ukazuje pouze prvky relativní k této práci. Pomocí hranatých závorek
953 je znázorněna množina, ze které musí být vybrán právě jeden prvek. Na místě uvozovek se
954 očekává jakákoliv povolená hodnota.

955 E.1 Iq

```
1      <iq from=""  
2          to=""  
3          type="[ get , set , result , error ]"  
4          id=""  
5          Namespace  
6      </iq>
```

Algoritmus E.1: Popis elementu *iq*.

956 E.2 Message

```
1      <message from=""  
2          to=""  
3          type="[ normal , chat , groupchat , headline , error ]"  
4          id=""  
5          <body> </body>  
6          <x xmlns="jabber:x:event">  
7              [ Offline , Delivered , Displayed , Composing ]  
8          <subject> </subject>  
9          <thread> </thread>  
10         <error> </error>  
11         <x> </x>  
12     </message>
```

Algoritmus E.2: Popis elementu *message*.

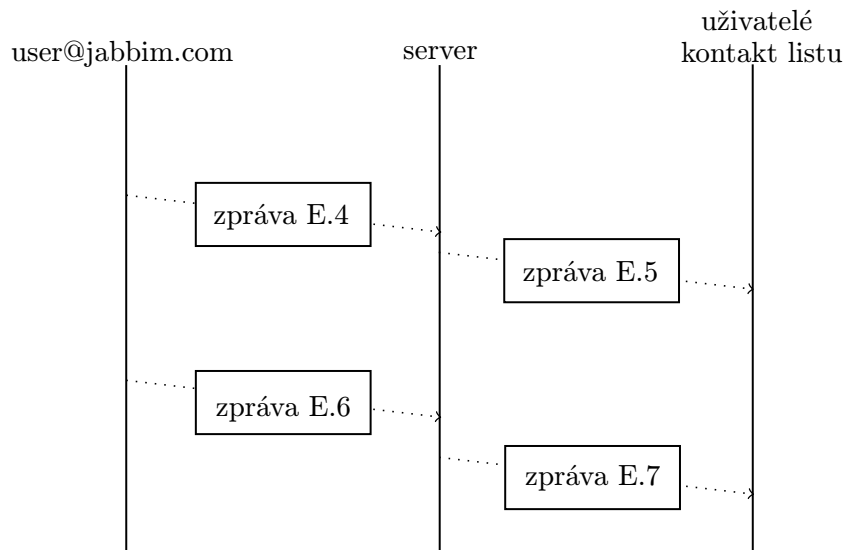
E.3 Presence

```
1  <presence from=""
2      to=""
3      type="[available, unavailable, probe, subscribe,
4            unsubscribe, subscribed, unsubscribed, error]"
5      id=""
6  <show>
7      [away, chat, dnd, normal, xa]
8  </show>
9  <status>    </status>
10 <priority>   </priority>
11 <error>      </error>
12 </presence>
```

Algoritmus E.3: Popis elementu *presence*.

958 E.4 Přehled průběhu rozšíření

959 Ukázka celého příkladu šíření statusu pomocí rozšíření *User Tune*. Uživatel *user* poslouchá hudbu a informuje server zasláním zprávy zobrazené v příkladu E.4.



Obrázek E.1: Ukázka „šíření“ *User Tune*.

960

```

1  <iq from="user@jabbim.com" type="set" id="pub1">
2    <pubsub xmlns="http://jabber.org/protocol/pubsub">
3      <publish node="http://jabber.org/protocol/tune">
4        <item>
5          <tune xmlns="http://jabber.org/protocol/tune">
6            <artist>Daniel Landa</artist>
7            <length>255</length>
8            <source>Nigredo</source>
9            <title>1968</title>
10           <track>5</track>
11          </tune>
12        </item>
13      </publish>
14    </pubsub>
15  </iq>

```

Algoritmus E.4: Informování serveru o právě přehrávající hudbě.

961 Server obdrží informace od klienta *user* zprávu o přehrávací hudbě. Pomocí elementu
962 *message* ji přepoše všem uživatelům z kontakt listu uživatele *user*, kteří jsou pro odběr
těchto typů zpráv zaregistrováni. Tato struktura zprávy je prezentována na příkladu E.5.

```
1  <message from="user@jabbim.com" type="set"
2      to="jabinfo@jabbim.com/bot" id="pub1">
3      <event xmlns="http://jabber.org/protocol/pubsub#event">
4          <items node="http://jabber.org/protocol/tune">
5              <item>
6                  <tune xmlns="http://jabber.org/protocol/tune">
7                      <artist>Daniel Landa</artist>
8                      <length>255</length>
9                      <source>Nigredo</source>
10                     <title>1968</title>
11                     <track>5</track>
12                 </tune>
13             </item>
14         </items>
15     </event>
16 </message>
```

Algoritmus E.5: Server informuje uživatele podporující rozšíření o stavu *user@jabbim.com*.

963 Zpráva o přehrávané hudbě je také přeposlána všem otevřeným spojením uživatele *user*,
964 ukázáno na příkladě E.6.
965

```
1  <message from="user@jabbim.com" type="set"
2      to="user@jabbim.com/doma" id="pub2">
3      <event xmlns="http://jabber.org/protocol/pubsub#event">
4          <items node="http://jabber.org/protocol/tune">
5              <item>
6                  <tune xmlns="http://jabber.org/protocol/tune">
7                      <artist>Daniel Landa</artist>
8                      <length>255</length>
9                      <source>Nigredo</source>
10                     <title>1968</title>
11                     <track>5</track>
12                 </tune>
13             </item>
14         </items>
15     </event>
16 </message>
```

Algoritmus E.6: Server přepoše informace o přehrávané hudbě všem otevřeným spojením
uživatele *user@jabbim.com*.

966 Přestane-li uživatel *user* poslouchat/vysílat informace o přehrávané hudbě, provede to
967 pomocí zprávy ukázané na příkladu E.7. Zpráva typu *iq*, ve které je položka *tune* nesoucí
informace o skladbě prázdná.

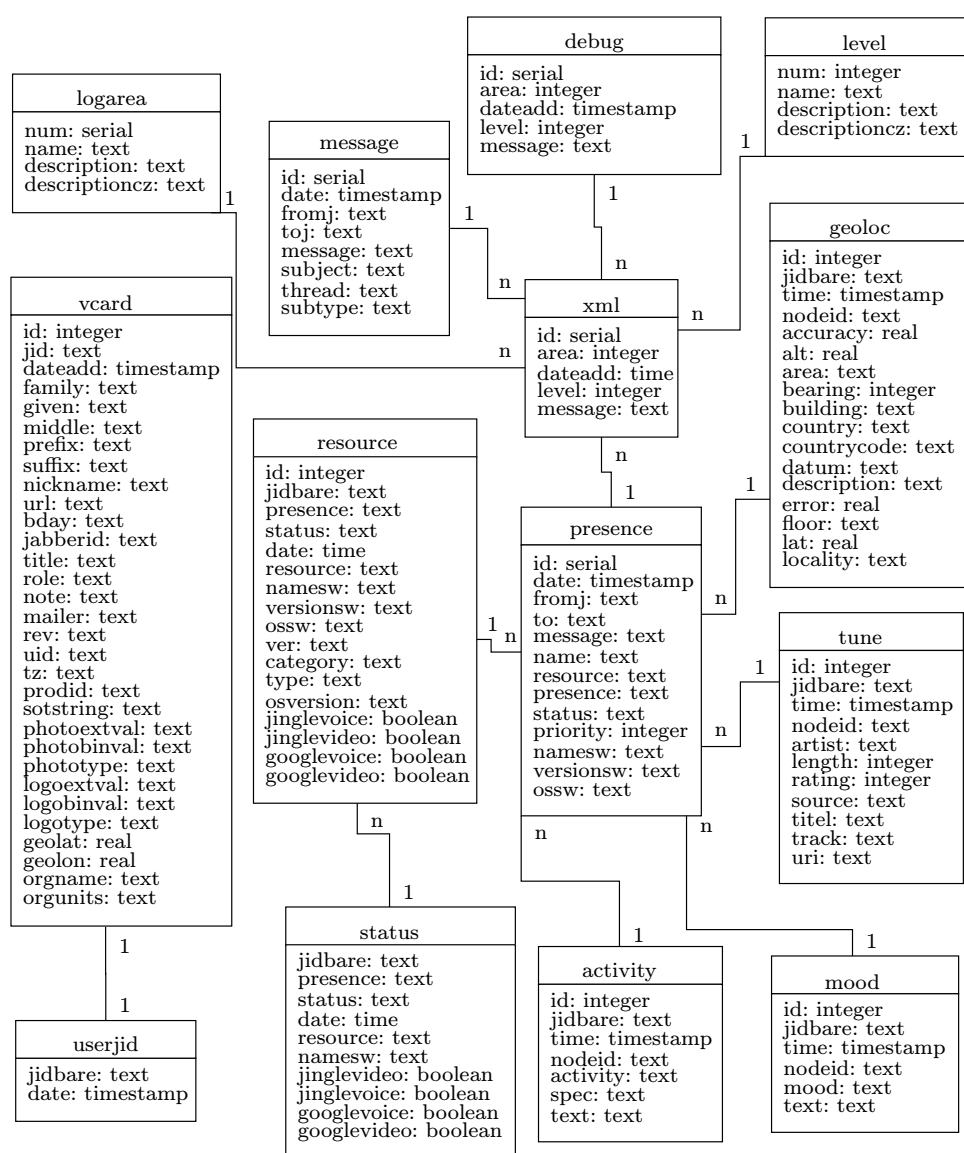
```
1 <iq from="user@jabbim.com/prace" type="set" id="pub1">
2   <pubsub xmlns="http://jabber.org/protocol/pubsub">
3     <publish node="http://jabber.org/protocol/tune">
4       <item>
5         <tune xmlns="http://jabber.org/protocol/tune"/>
6       </item>
7     </publish>
8   </pubsub>
9 </iq>
```

Algoritmus E.7: Uživatel ukončil „vysílání“ rozšířených zpráv o svém stavu.

968 Server informuje všechny účastníky odběru zprávou, která má položku *tune* prázdnou.
969 Tak jak to prezentuje příklad E.8.
970

```
1 <message from="user@jabbim.com"
2   to="jabinfo@jabbim.com/bot">
3   <event xmlns="http://jabber.org/protocol/pubsub#event">
4     <items node="http://jabber.org/protocol/tune">
5       <item>
6         <tune xmlns="http://jabber.org/protocol/tune"/>
7       </item>
8     </items>
9   </event>
10 </message>
```

Algoritmus E.8: Server informuje klienty o ukončení šíření rozšířeného statusu uživatele *user@jabbim.com*.



Obrázek F.1: Struktura databáze

973 Příloha G

974 Přehled klientů a jejich rozšíření

Klient	OS	XEP--60	XEP--163	XEP--80	XEP--92	XEP--107	XEP--108	XEP--118	XEP--	XEP--
Adium										
Agile Messenger										
AQQ										
Aytm										
beejive										
Beem										
BitlBee										
Bombus										
BuddyMob										
Chatopus										
Citron										
Claros Chat										
climm										
Coccinella										
Crosstalk										
Digsby										
eM Client										
emite										
Empathy										
Exodus										
Finch										
Gajim										
Galaxium										
glu										
GNU Freetalk										
Gossip										
iChat										
iJab										
IM+										
imov Messenger										
irssi-xmpp										
Jabbear										
Jabber Mix Client										
jabber.el										
Jabbim										
Jabbim for Android										
Jabiru										
JAJC										
Jappix										

Klient	OS	XEP--60	XEP--163	XEP--80	XEP--92	XEP--107	XEP--108	XEP--118	XEP--	XEP--
JBuddy Messenger										
Jeti										
Jitsi (SIP Communicator)										
JWChat										
Kadu										
Kopete										
Lampiro										
m-im										
mcabber										
mChat										
Miranda IM										
Monal IM										
OctroTalk										
OneTeam										
OneTeam for iPhone										
Oyo										
Pandion										
Poezio										
Pidgin										
Prodromus										
Psi										
Psi+										
Quiet Internet Pager (QIP)										
qutIM										
saje										
SamePlace										
Sim-IM										
Slimster										
SoapBox Communicator										
Spark										
SparkWeb										
Synapse										
Talkonaut										
Tigase Messenger										
Tigase Minichat										
Tkabber										
Tlen										
Trillian										
TrophyIM										
V&V Messenger										
Vacuum-IM										
Vayusphere										
WTW										
Xabber										
xmppchat										
Yambi										
Yaxim										

Tabulka G.1: Přehled podporovaných rozšíření u jednotlivých klientů.