Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Sběr statistických dat v telefonní ústředně Asterisk

Plzeň, 2016 Václav Žák

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne Václav Žák

Poděkování

Rád bych poděkoval vedoucímu své diplomové práce Ing. Vladimíru Toncarovi, Ph.D a mému konzultantovi Ing. Martinu Zímovi, Ph.D za cenné rady, připomínky, informace a čas, který mi věnovali při řešení dané problematiky.

Abstract

This thesis deals with analyzing call detail records (CDR) produced by call queues on an Asterisk PBX, and with the steps leading to providing the data suitable for such analysis. In particular, it solves the parsing of non-trivial amounts of data from the PBX and storing it in suitable structures in a temporary processing database. After that, the CDR data is processed by an ETL (extraction, transformation, loading) application that feeds the data into a data warehouse. Both the processing database and data warehouse are realized using a relational database management system, namely the open-source PostgreSQL. A large set of CDR data needed for testing the whole solution was created by running a simulated PBX traffic. The text also describes several possibilities of analyzing the data and includes visualization examples.

Keywords

asterisk, voip, data warehouse, relation database

Abstrakt

Tato diplomová práce se zabývá analyzováním dat hovorů, která produkují hovorové fronty telefonní ústředny Asterisk, a postupy vedoucí k zajištění vhodnosti dat pro takovou analýzu. Konkrétně řeší analýzu netriviálního množství dat z telefonní ústředny a ukládá je ve vhodné struktuře v dočasné produkční databázi. Poté jsou tyto data zpracovány ETL (extraction, transformaction, loading) aplikací, která převádí data do datového skladu. Produkční databáze a datový sklad jsou realizovány v relačním systému řízení báze dat, konkrétně v open-source řešení PostgreSQL. Velký objem dat potřebný pro testování celého řešení byl vytvořen spuštěním simulovaného provozu telefonní ústředny. Text také popisuje několik možností analýzy dat a zahrnuje vizualizaci příkladů.

Klíčová slova

asterisk, voip, datový sklad, relační databáze

Obsah

1. Úvod

Běžně užívanou alternativou ke klasickým komunikačním technologiím je používání protokolu IP pro telekomunikační služby. Pro přenos hlasu přes IP je možné použít internet, intranet nebo libovolného datového spojení s dostatečnou šířkou pásma. Jelikož tyto IP telekomunikace produkují hodně dat, je zde motivace tyto data rozumně zpracovávat a vyhodnocovat.

Cílem práce bylo vytvořit sadu programů, které poskytnou data pro statistické a analytické vyhodnocení telefonního provozu. Sada programů má umožnit čtení dat o událostech, které se v reálném čase dějí na telefonní ústředně, a postupně data zpracovat až do podoby datového skladu, se kterým lze pracovat například prostředky OLAP. Důležitou podmínkou bylo strukturovat řešení tak, aby výkonově vyhovovalo i v případě, že daná telefonní ústředna je i dlouhodobě vystavena velkému provozu, který odpovídá například i velkému call centru.

V první části diplomové práce bude představena telefonní ústředna Asterisk, která je schopna nahradit klasické pobočkové ústředny. Dále pak budou rozebrány základy IP telefonie, komunikační protokoly a druhy monitorovacích rozhraní (Asterisk Management Interface) umožnujících monitorování provozu těchto ústředen, které využívají pro komunikaci bránu VoIP.

Ve druhé části práce bude analyzováno monitorování Asterisk ústředny a převážně provoz její hovorové fronty, který bude názorně předveden v podobě uskutečnění hovoru, a sledování AMI protokolu, ze kterého bude čerpat aplikace pro sběr dat do relační databáze.

Třetí část bude řešit požadavek mezistupně vytváření datového skladu, a to dočasné ukládání nasbíraných dat v relační databázi, kvůli výkonu a jejich množství. Dále budou představeny různé druhy databázových systémů a jeden z nich bude vybrán pro následnou implementaci. Ve vybraném databázovém systému bude vytvořen model relační databáze, který dovoluje nasbíraná data ukládat a také bude vytvořena aplikace, která sběr dat hovorové fronty a jejich ukládání do vytvořené relační databáze umožňuje.

V další, čtvrté, části bude představena teorie datových skladů a pro vyhodnocení faktů monitorování hovorové fronty z telefonní ústředny Asterisk bude navržena struktura takového datového skladu, která umožní analytické zpracování dat z průběžné relační databáze. Bude vybrán vhodný databázový systém a implementován program, který dovede z dočasné relační databáze tyto data dávkovým způsobem převádět.

V poslední části bude vytvořen simulovaný provoz, který bude ověřovat funkčnost všech vytvořených programů a na základě kterého proběhne vyhodnocení a vizualizace výsledků.

2. Telefonní ústředna Asterisk

2.1 Co je Asterisk

Projekt “The Asterisk” začal v roce 1999, když vývojář Mark Spencer vydal prvotní verzi kódu. Od té doby byla mnohonásobně rozšířena, o což se zasloužila hlavně celosvětová komunita, čítající tisíce vývojářů, a firma Digium, kteří Asterisk zároveň udržují a testují. Asterisk je šířen pod dvěma licencemi. První je open source licence GNU GPL a druhá je proprietární licence, která umožňuje využití uzavřených a patentovaných zdrojových kódů (např. kodek G.729). Původně byl navržen pro operační systém Linux, ale nyní je možno ho používat na dalších unixových systémech (NetBSD, OpenBSD, FreeBSD, Mac OS X a Solaris). Důležité je, že umožňuje komunikaci s VoIP telefonní sítí. Pro komunikační spojení, tedy pro spojení s okolní telefonní sítí VoIP, používá například následující protoky:

- SIP - Session Initiation Protocol
- IAX - Inter-Asterisk Exchange
- H.323

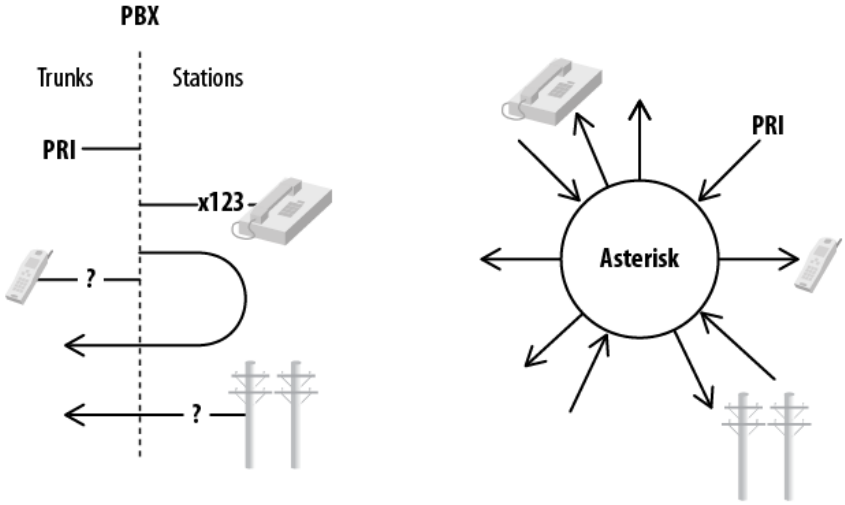
Obecně se dá říci, že Asterisk je open-source framework pro tvorbu komunikačních aplikací, který nepotřebuje dodatečný hardware, dokáže tedy proměnit na komunikační server i obyčejný počítač. Může být použit v aplikacích IP PBX, VoIP bránách, konferenčních serverech a dalších zakázkových řešeních a používá se v malých, středních a velkých podnicích, call centrech, dopravcích společnostech a vládních agenturách po celém světě.[1]
V současné době existuje více než jeden milion komunikační systémů založených na Asterisk a používá se ve více než 170ti zemích světa. [2] Asterisk se může stát základem kompletního systému firemní telefonní ústředny nebo se může použít ke zlepšení nebo rozšíření systému stávajícího. Dá se tedy říct, že Asterisk je pobočková ústředna zahrnující vylepšení a nové služby.

2.1.1 Pobočková ústředna PBX

Pobočková ústředna, čili PBX, je také nazývána jako “společný komunikační systém” nebo “obchodní telefonní systém”, který sjednocuje výstupní body všech telefonů do veřejné telefonní sítě. PBX systémy tedy zpracovávají interní firemní provoz mezi stanicemi a působí jako jedna výchozí stanice pro komunikaci s okolním světem. Výhody tohoto systému jsou centralizovaná struktura, jednoduchá obsluha, velmi vysoká spolehlivost (až 99,9 %) a má velký dopad na úspory, jelikož společnost platí pouze za jeden přístupový bod [3].

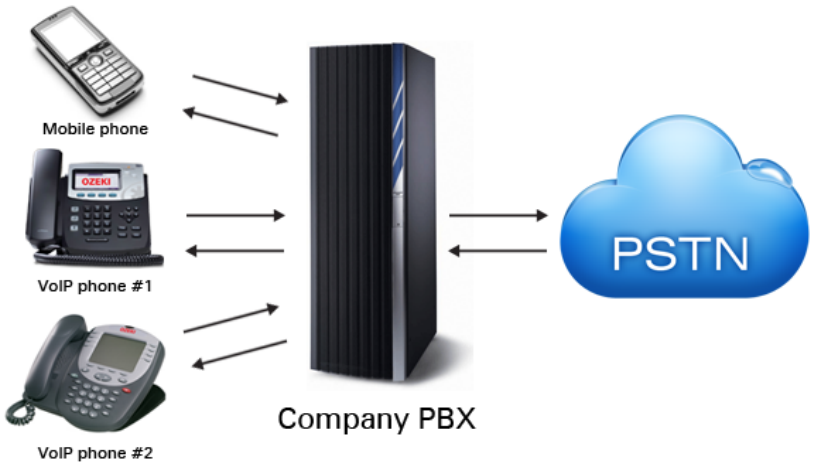
2.1.2 Asterisk a PBX

V tradičním pojetí PBX je logický rozdíl mezi stanicemi (tzv. *stations* - telefonní přístroje) a rozhraními, které se připojují k vnějšímu světu (tzv. *trunks*). To znamená, že například nelze nainstalovat externí bránu na stanici a přesměrovávat externí hovory bez nutnosti vytočit nejdříve číslo linky. [1] Jak je vidět na obrázku *Obrázek 1*, Asterisk tyto služby naopak spojuje.



Obrázek 1: Rozdíl mezi architekturou klasických PBX a Asterisk [2]

Na základě toho lze říci, že Asterisk byl původně vytvořen jako hnací motor pro PBX systémy. Vytváří pro něj nové služby, jako je hlasová schránka, automatické spojení hovorů, hovorové fronty, konferenční most, parkování, plány vytáčení a vnitřní volání. Asterisk zahrnuje technologie a protokoly, díky kterým jej můžete připojit k vnějšímu světu pomocí VoIP nebo tradičních telefonů (*Obrázek 2*).



Obrázek 2: PBX systém, VoIP telefony a PSTN (veřejná telefonní síť)

2.2 Základní části Asterisku

Telefonní ústředna Asterisk je postavena na modulech. Modul je komponenta, která poskytuje specifické funkce, jako je například ovládání kanálu, nebo prostředky, které umožňují například připojení k externí technologii. [1]

2.2.1 Moduly

Moduly jsou zaváděny do Asterisku na základě konfiguračního souboru *modules.conf*. Tento soubor uchovává, z jakých technologií se ústředna bude skládat. Je možné upravit konfigurační soubor podle vlastních představ nebo získat přímo předpřipravená řešení. Je také možné začít používat Asterisk bez načtených modulů a postupně přidávat moduly ručně z konzole, pokud by bylo třeba vyladit systém kvůli výkonnosti. [2]

Typy modulů Asterisku zahrnují následující:

- aplikace
- přemostění
- CDR moduly (nahrávání hovoru)
- CEL moduly (logování kanálu událostí)
- ovladače kanálu
- překládání kodeků
- funkce vytáčecího plánu

- moduly PBX
- prostředky
- testovací moduly

2.2.2 Konfigurační soubory

Tak jako moduly definují různé funkce, kanály, prostředky, kodeky a další moduly, konfigurační soubory definují jejich nastavení a parametry. Již v předchozí kapitole byl zmíněn konfigurační soubor, který uchovává informace, z jakých součástí se bude Asterisk skládat. Konfigurační soubory celkově zahrnují desítky těchto souborů a uchovávají se v linuxové adresářové struktuře v */etc/asterisk/*.

Nadřízeným konfiguračním souborem je *asterisk.conf*, který umožňuje ladit různá nastavení, které mohou ovlivnit chod Asterisku jako celku.

Dalšími konfigurační soubory, mimo zmíněné umožňující počáteční konfiguraci [1]:

indications.conf - definuje parametry pro různé zvuky vytáčení, které se nastavují pro zvyklosti v dané zemi, kde se volající nachází.

musiconhold.conf – definuje nastavení hudební smyčky, která hraje zákazníkovi při čekání na připojení agenta.

Tak jako sestavení modulů, jako součást telefonního systému, je možné všechny tyto konfigurační soubory nastavit dle požadavků a vyrobit tak telefonní ústřednu „šitou přesně na míru“.

2.2.3 Kanál

Kanál je spojení, které přináší volání do Asterisk PBX a vytváří se při určitém stavu, kterým může být vyzvánění nebo uskutečnění hovoru. Kanál může být napojen na tradiční telefon či telefonní linku nebo VoIP telefon.

Pojem kanálu znázorňuje následující příklad volání z VoIP telefonu do Asterisk PBX:

1. VoIP telefon je připojen k PBX Asterisk pomocí SIP protokolu. Telefon vytočí číslo v telefonní ústředně.
2. VoIP telefon dostane kanál vytvořený Asteriskem
3. Asterisk pomocí některé služby vyvolá funkci pro vytáčení na cílové telefonní číslo. Vytvoří se další kanál.
4. Asterisk tyto kanály propojí a jdou přes ně informace o stavu hovoru a případně i zvuková či obrazová data

2.2.4 Dialplan

Plán vytáčení (*dialplan*) je součást konfigurace Asterisku. Asterisk zpracovává všechny kanály podobným způsobem, což znamená, že například interní uživatel může existovat na vnějším zdroji (například mobilní telefon). Plán vytáčení se tak použije přesně stejným způsobem, jako kdyby uživatel existoval na interní lince.

Všechny kanály, které přijdou do systému, zpracovává plán vytáčení, který obsahuje skripty určující jakým způsobem příchozí hovory obsloužit. Z toho se dá usoudit, že *dialplan* je skriptovací programovací jazyk, ve kterém je naprogramován průběh telefonních hovorů.

2.3 VoIP brána

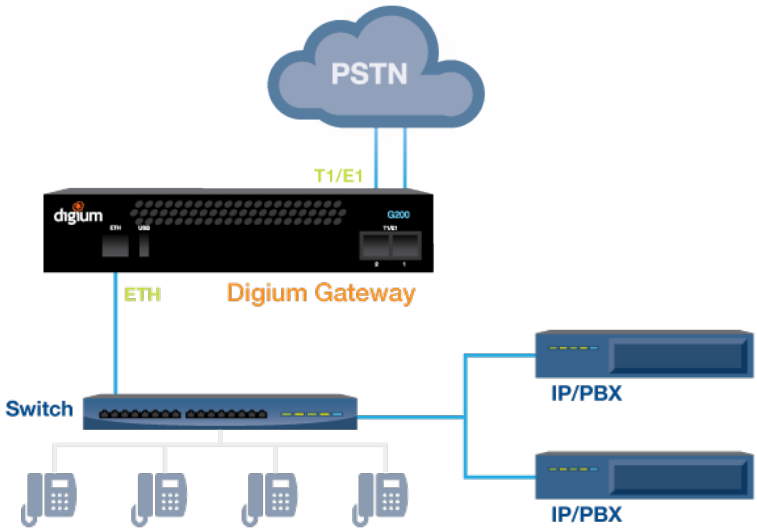
2.3.1 VoIP

Voice over Internet Protocol, zkratka VoIP, je technologie, umožňující přenos digitalizovaného hlasu v těle paketů protokolů UDP/TCP/IP prostřednictvím počítačové sítě nebo jiného média, prostupného pro protokol IP.

2.3.2 Co je VoIP brána

VoIP brána je síťové zařízení, které slouží jako spojovací prvek, kdy je telefonní hovor zčásti přenášen pomocí VoIP a zčásti jinou metodou. Dalo by se tedy říci, že je to propojení mezi tradičními telefony a VoIP telefony. Typicky se používá právě při připojení systému PBX, který zahrnuje tyto VoIP telefony. Brána se dá připojit ke stávajícímu systému prostřednictvím buď analogové nebo digitální linky. PBX ústředna vidí VoIP bránu buď jako telefonní společnost nebo další síťovou ústřednu. Volání z PBX ústředny do vnějšího světa jsou přeměněna na VoIP hovory a poslána přes internet k jinému poskytovateli VoIP služeb. Přijaté hovory ze zdrojů VoIP jsou pak převedeny do příslušného staršího protokolu a doručeny do PBX ústředny. [5]

Používání brány k přípojení VoIP telefonního systému do tradiční telefonní linky dává smysl, když není možné použít přímo VoIP volání nebo když aplikace požaduje spolehlivost veřejné telefonní ústředny. Příklad VoIP brány je uveden na obrázku *Obrázek 3*.



Obrázek 3: Příklad VoIP brány použitím Digium produktu[8]

2.3.3 Asterisk a VoIP brána

Asterisk podporuje klasické digitální telefony (TDM) i VoIP protokoly, což umožňuje udělat bránu mezi různými typy kanálů. Použitím Asterisku je možno udělat i mnohem sofistikovanější výchozí brány s redundantními linkami. Přidáváním VoIP telefonů dává systému Asterisk BPX rozšíření o nové služby a tím snižuje náklady. [1]

2.4 Další nové služby pro PBX

Mimo rozšíření o VoIP bránu, jak je nastíněno v kapitole *2.3.3 Asterisk a VoIP brána*, lze Asterisk PBX vylepšit o další služby.

2.4.1 Interaktivní hlasový systém

IVR (Interactive Voice Response) systém je samoobslužný hlasový systém, který umožňuje využít sofistikované hlasové technologie a zvýšit automatizaci v oblasti obsluhy zákazníků. Tento automatizovaný systém odpovídá na volání zaznamenanou nebo syntetizovanou řečí a vyzývá volajícího odpovědět na nabídku možností zadáváním informací prostřednictvím tónové volby nebo přímo mluvením do telefonu.

Složitější aplikace zahrnují resetování hesla, hlasové průzkumy, dotazy zůstatku na bankovním účtu, kontrolu stavu letu, sledování zásilky, atd.

Klíčovou myšlenkou je automatizovat rutinní a opakující se úlohy, které by jinak vyžadovaly čas a úsilí zaměstnanců. Potenciál v úsporách dává IVR velmi rychlou návratnost investice (ROI – Return On Investment), protože na serveru může potenciálně nahradit několik zaměstnanců (agentů). [2]

2.4.2 Call centrum

Call centrum je speciálních oddělení společnosti, jejichž účelem je zvládnout velké množství telefonních hovorů. Call centra obvykle obsluhují zákaznický servis, podporu či telemarketing. Zaměstnanci jako pracovníci call centra jsou označovány jako agenti nebo zástupci zákaznických služeb. Call centra nabývají od velmi malých oddělení po masivní, vysoce optimalizované se stovky nebo dokonce tisíce agentů. [2]

Automatická obsluha

Automatická obsluha (ACD - Automatic Call Distributor) je specializovaný telefonní systém, který distribuuje příchozí hovory do skupin agentů přiřazených k různým hovorovým frontám. Hovorové fronty jsou seřazený seznam hovorů, který má být odeslán agentům. ACD dohlíží na proces směrování příchozích hovorů do správných front a přiřazuje přednost hovorům na základě různých faktorů. Může se jednat o pořadí jejich příchodů, důležitost zákazníka nebo například naléhavost situace. Algoritmus, podle kterého jsou směrovány hovory na agenty, se nazývá strategie fronty.

Strategie fronty

Základní strategie v případě FIFO (first-in, first-out), kdy se obslouží vždy první zákazník v pořadí, probíhá tak, že jsou zavolání všichni dostupní agenti a pokud jeden zvedne hovor, ostatní přestanou vyzvánět. Další složitější strategie je například směrování hovoru na agenta, který nejdelší dobu nikoho neobsluhoval. Další běžné strategie zahrnují algoritmy např. round robin, linear hunt nebo náhodné směrování.

Zákazníci

Při čekání ve frontě, zákazníci obvykle poslouchají kombinace marketingových zpráv, stavových zpráv či hudbu. Marketingové zprávy jsou prosté zvukové nahrávky, které

jsou pouštěny ve frontě v pravidelných intervalech. Stavové zprávy poskytují volajícímu specifické informace o jejich stavu (počet volajících před nimi ve frontě, odhadovaná doba čekání).

Některé vyspělejší systémy v hovorových frontách podporují tzv. virtuální fronty. Systém virtuální fronty umožňuje volajícímu poskytnout číslo pro zpětné volání a poté ho odpojí. Postavení zákazníka ve frontě je pak při zpětném zavolání zachováno.

Vytáčení

Odchozí hovory call centra jsou často používány s vytáčejícími aplikacemi, které spojí agenta s požadovaným cílem bez použití ručního vytáčení. Této funkci se říká *click-to-*call (klikni a volej). Používání této funkce na počítačích je často spojována s CRM (Customer Relationship Management) softwarem, tedy systémem pro řízení vztahů se zákazníky.

2.4.3 Konferenční most

Konferenční most umožňuje skupině lidí účastnit se telefonního hovoru. Je to funkce, která se používá převážně pro obchodní konference. Nejběžnější forma mostu je hovor, který zakládá administrátor (správce) a ostatní účastníci se k němu připojují. Administrátor konference vybírá, zda účastníkům povolí poslouchat nebo jim umožní i mluvit. Při zavěšení hovoru administrátorem jsou odpojeni všichni účastníci.

Konferenční systémy typicky podporují více konferenční místností, z nichž každá může obsahovat více účastníků. Celkový počet konferenčních místností a účastníků se liší v závislosti na modelu, schopnostech hardwaru a licenčních podmínek.

Konferenční místnosti mohou být volitelně zabezpečeny kódem PIN. Některé systémy používají společný PIN pro všechny účastníky, zatímco jiné používají vlastní PIN pro každou z nich.

Pokročilé konferenční systémy obsahují grafické uživatelské rozhraní, které umožňuje všem účastníkům, aby zjistili, kdo je v současné době mluví a případně kdo se připojil ke konferenci. Administrátoři a moderátoři mají obecně více komplexní pohled, který obsahuje rozšířené ovládací prvky. Některé tyto systémy zahrnují dynamické zasedací místnosti, tj. místnosti, které jsou vytvořeny na základě určitého plánu.

2.4.4 Voicemail

Hlasové zprávy umožňují volajícímu zanechat zprávy pro předplatitele (uživatele) systému. Systémy hlasových zpráv jsou často používány ve spojení s PBX systémy, mobilní telefony a telefonními službami.

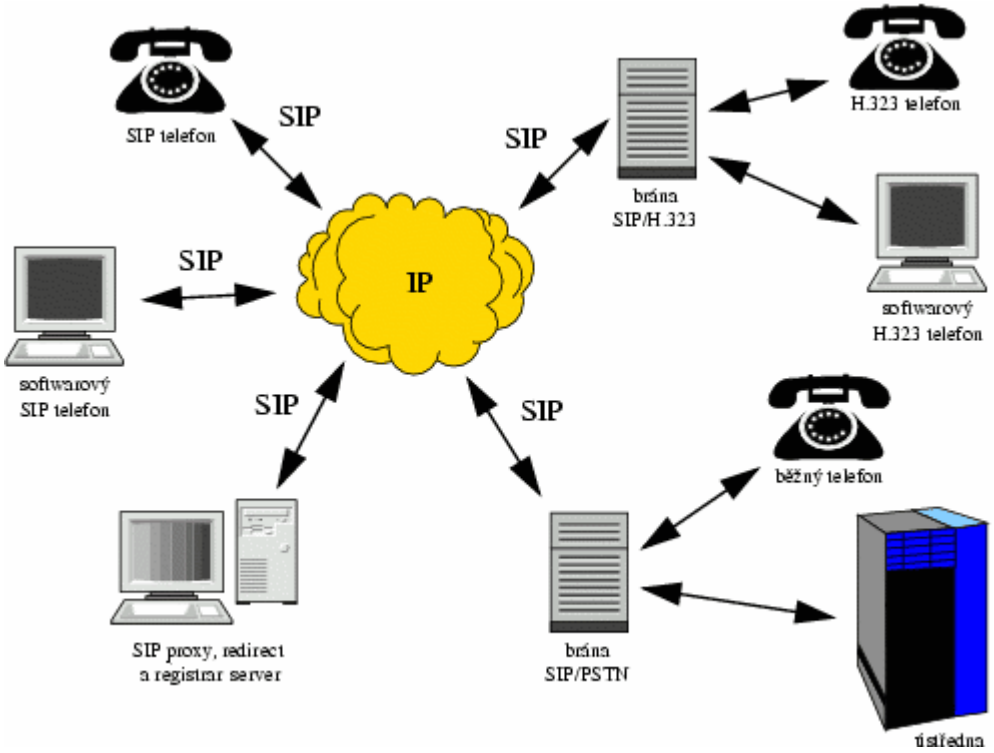
Systém hlasových zpráv obsahuje několik základních komponent. Proces sběru hlášení je aktivován, když volající není schopen dosáhnout uživatele systému. Sběr hlášení zpráv přijímá data z telefonního systému, který udává, jaký účastník se pokoušel volat. Aplikace sběru hlášení zahraje uvítání a poté zaznamená zprávu. Uvítání může být standardní, které defaultně nastavuje systém nebo vlastní záznam uživatele. Jakmile je zpráva zaznamenána, systém předá oznámení uživateli.

Toto je řešeno různými způsoby v závislosti na typu telefonního systému, ve kterém jsou hlasové zprávy integrovány. Ve většině případů pošle systém příkaz do nadřazeného systému (např. PBX), který zapne upozornění na čekající zprávu v telefonu uživatele. Oznámení se také může poslat emailem, který tento zvukový soubor se zprávou obsahuje jako přílohu.

Klasické systémy hlasových zpráv vyžadují přihlášení do aplikace a ověření čísla své linky a přehrávají zprávy postupně. Modernější systémy umožňují uživateli, aby prozkoumal zprávy na svém počítači nebo v mobilním telefonu přímo pomocí aplikace s GUI. Rozšířené hlasové zprávy jsou klíčovým prvkem k systémům sjednocených zpráv (Unified Messaging platforem), které kombinují více formátů zpráv do jediného přístupového bodu pro účastníka. V těchto systémech může schránka obsahovat hlasové zprávy, e-mail, fax a někdy i text v podobě chatu nebo SMS zpráv.

2.5 Protokol SIP

Session Initiation Protocol (zkráceně SIP) je VoIP protokol určený pro přenos signalizace v multimediální komunikaci, vytvořený skupinou IETF MMUSIC. Formálně je protokol určen pro vytváření, úpravy a ukončování relací s jedním nebo více účastníky. Těmito relacemi jsou převážně VoIP telefonní hovory nebo konference. [6] Prvky a možnosti komunikace protokolu jsou uvedeny na obrázku *Obrázek 4*.



Obrázek 4: Prvky a možnosti komunikace protokolu SIP[7]

2.5.1 Charakteristika SIP

SIP je textově orientovaný protokol. Formát SIP požadavků a odpovědí je založen na protokolu HTTP.

Koncové body, které při komunikaci využívají SIP, používají následující tři protokoly:

- SIP – k vytváření a ukončování relací
- SDP (Session Description Protokol) – k výměně informací o audio/video kanálech
- RTP (Real Time Protocol) – k odesílání audio/video streamů v reálném čase přes síť

SIP zprávy jsou vyměňovány mezi koncovými body v transakcích. Transakce se skládá z požadavku a souvisejícími odpověďmi. Zprávy náležící jedné transakci sdílejí stejný transakční identifikátor nazývaný *Cseq*. Každá transakce by měla mít Cseq číslo unikátní. [4]

Protokol SIP nespecifikuje jaký má být použit transportní protokol. SIP umí použít oba

základní transportní protokoly TCP i UDP), ale obvykle se používá protokol UDP, který lze snadněji implementovat. [6]

2.5.2 Komponenty SIP

Síť SIP se může skládat z několika komponent. Rozdíl mezi komponentami je logický, v praxi jsou často některé z komponent spojeny dohromady na jednom SIP serveru (obvyklý případ je spojení proxy/registrar na jednom serveru). [4]

User Agent

SIP telefon (hardwarový i softwarový) je skrz SIP viděn jako *User Agent* a může zastávat dvě role:

1. User Agent Client – zasílá požadavky
2. User Agent Server – přijímá požadavky a odesílá odpovědi

Registrar

Úkol tohoto serveru je přijímat registrované zprávy od SIP klientů. Tato registrovaná zpráva obsahuje informaci o umístění (IP adresa) SIP klienta (např. telefon). Server uchovává tyto informace v databázi a zná tedy IP adresu každého registrovaného uživatele.

Proxy

Proxy server přijímá SIP požadavky a přeposílá je do cílů koncových bodů nebo do další proxy, nacházející se nejblíže k cíli. Namísto proxy serveru lze použít i přímého spojení s cílem, to ale není ideální řešení, jelikož proxy server má přístup k databázi umístění registrar serveru a navíc by nemohlo být ověřeno, jestli má volající dovoleno hovor vykonat.

Redirect server

Redirect server obdrží žádost o navázání spojení, vyhledá v databázi volajícího a zjistí, kde se volaný účastník nachází a vrátí kód 3xx (přesměrování) s informacemi, na jakou URI má volající žádost zaslat. [6] Redirect server potřebuje mít přístup k registrar serveru, aby mohl čerpat z databáze umístění.

Gateway

Gateway server (brána) přijímá SIP hovory a překládá je do jiné telekomunikační sítě. Může se jednat od veřejnou telefonní síť, síť protokolu H.323 nebo Skype.

2.5.3 Adresace

Standardní SIP adresy (SIP URI) nabývají tvaru *sip:jméno@server.doména*, tedy prefixem *sip* a adresou podobnou emailové adrese.

Při vykonání hovoru na SIP telefon existují dvě možnosti, jak zjistit umístění uživatele a to dotazování DNS nebo spolehnutí se na službu poskytující informace o umístění.

Dotazování DNS

Dotazování DNS (Domain Name Server) – SIP telefon vezme název domény z SIP adresy, převede ji na IP adresu a kontaktuje tento server za účelem možnosti hovor provést.

Služba umístění

Použití služby poskytnutí umístění – SIP (RFC 3261) nespecifikuje, jak by měla být služba poskytnutí umístění implementována. Obvyklé řešení je přes společně umístěný proxy a registrar server.

Obvyklý sled událostí vykonání hovoru přes službu umístění:

1. SIP telefony jsou registrované na registrar serveru. Server uloží umístění telefonů (jejich čísla a IP adresy) do databáze.
2. Proxy server má přístup do této databáze.
3. Všechny telefony v této síti znají adresy proxy serveru a pokud chtějí provést hovor, zašlou žádost tomuto proxy serveru. Pokud účastník chce volat na číslo 01 a adresa proxy serveru je *192.168.0.1*, telefon zavolá SIP URI ve tvaru *sip:01@192.168.0.1*.
4. Proxy server přepošle požadavek hovoru do cílové destinace telefonu.

2.5.4 Řízení relací

Pro vytvoření a řízení multimediální relace musí SIP zajistit následujících pět činností: [6]

- Lokalizace účastníka – nalezení spojení s koncovou stanicí
- Zjištění stavu účastníka – zjištění, jestli je účastník schopen relaci navázat (může mít obsazeno, přesměrováno atd.)
- Zjištění možností účastníka – zjištění, jaké jsou možnosti účastníka (typ kodeku, max. přenosová rychlost, audio/video, atd.)
- Vlastní navázání spojení – probíhá prostřednictvím protokolu SDP, který popisuje navázané spojení a odkazuje na RTP datový tok
- Řízení probíhajícího spojení – případné změny vlastností v průběhu relace a činnosti spojené s jejím ukončováním

2.6 PBX systémy na bázi Asterisk

2.6.1 „Vannila“ Asterisk

Pod pojmem „Vanilla“ Asterisk se rozumí takový Asterisk systém, která neobsahuje grafické uživatelské rozhraní (GUI) pro jeho administraci. V rámci Asterisk se jedná o aplikaci s názvem FreePBX. Celé složení a nastavení systému se tak konfiguruje přes příkazovou řádku. Bez administrátorského GUI, pro například jednoduchou úlohu v podobě přidání rozšíření, se musí přidávat řádky kódu do příslušných konfiguračních souborů. Provádění těchto úkonů vyžaduje ale obeznámení s konfigurací Asterisku obecně, není to tak pohodlné, jako v případě konfigurace přes grafické rozhraní. [39]

2.6.2 AsteriskNOW

AsteriskNOW je kompletní Linuxová distribuce, která obsahuje Asterisk, DAHDI a FreePBX.

Pod tímto se dá představit na míru vytvořený operační systém na Linux bázi. Obsahuje DAHDI, jehož zkratka znamená interface, obsahující ovladače pro hardwarová zařízení od firmy Digium a FreePBX, již zmíněné administrátorské grafické rozhraní, které slouží jako nástroj pro správu telefonní ústředny.

AsteriskNOW byl vytvořen pro vývojáře aplikací, systémové integrátory, studenty a jiné, kteří si chtějí vytvořit vlastní řešení na této bázi. Je volně k dispozici pro použití doma, ve škole nebo v práci. [8]

2.6.3 Kerio Operator

Kerio operator je VoIP ústředna, tedy systém pobočkové ústředny pro IP telefonii, která vychází z open source řešení Asterisk a je dostupná i jako virtuální stroj. Je určený pro malé a středně velké organizace, uvedla jej na trh společnost Kerio Technologies, která se snaží o maximální zjednodušení jeho integrace do možné původní infrastruktury.

Obsahuje automatickou konfiguraci telefonů Cisco, Linksys či Snom, která umožňuje připojování nových telefonů bez nutnosti zásahu správce. Pro připojení k veřejné telefonní síti lze využít mimo jiné výše zmíněné linky SIP. Speciální podpora pro více linek do veřejné sítě umožňuje úsporu nákladů a případně snadnou výměnu stávající telefonní ústředny za Kerio Operator.

Webové uživatelské rozhraní MyPhone umožňuje koncovým uživatelům vzdálený přístup k jejich hlasovým schránkám a nastavení telefonů s jednoduché ovládání.

Operator je k dispozici ve formě softwarového zařízení založeného na speciálním optimalizovaném a zabezpečeném operačním systému, a také ve formě virtuálního zařízení pro VMware, které umožňuje okamžité nasazení do virtuální infrastruktury nebo testování produktu na běžném PC. [10]

Přehled některých vlastností Kerio Operator

Dle zmíněných funkcí telefonní ústředny Asterisk obsahuje distribuce Kerio Operator následující vlastnosti [9]:

- **Přesměrování hovorů** – např. předávání hovorů na všechny linky či nastavitelné audio zprávy
- **Předávání a zpracování hovorů** – automatické IVR, monitorování hovorů
- **Produktivita uživatelů** – integrace AMI s dalšími CRM systémy, nahrávání hovorů
- **Bezpečnost** – blokace při nestandardním chování a šifrování hovorů
- **Správa** – informace o stavu systému v reálném čase, vzdálené monitorování

2.7 Typy rozhraní Asterisk

2.7.1 Rozhraní AGI

AGI (Asterisk Gateway Interface) je rozhraní PBX Asterisk, které ústředně přidávájí rozšíření o externě naimplementované zdroje. S využitím AGI může být Asterisk rozšířen o externí aplikace, které mohou být napsány ve skoro kterémkoliv programovacím (skriptovacím) jazyce. Do těchto aplikací lze směrovat proud dat z jádra Asterisku a data tak mohou být zpracována nebo upravena mimo Asterisk. Výběr programovacího jazyka (skriptovacího) zavisí na tom, k čemu je daný modul navržen. [11]

2.7.2 Rozhraní AMI

AMI (Asterisk Manager Interface) je jednoduchý protokol, který umožňuje vzdálenou správu a komunikaci s Asterisk serverem prostřednictvím TCP/IP. Má podporu pro editaci nebo vytvoření konfiguračních souborů a také možnost spravovat hovory, klienty, agenty, plány vytáčení, atd. Při komunikaci lze využívat příkazy v rámci konzole. Při zadávání příkazů je nutné dodržet jejich formální zápis, tzn. dodržet specifickou posloupnost a parametry. Jednotlivé parametry příkazu se oddělují odřádkováním. [12]

3 Analýza Asterisk Manager Interface protokolu

Jak bylo řečeno v kapitole *2.7.2 Rozhraní AMI*, AMI je prostý textový protokol, který funguje na principu odesílání a přijímání paketů. K analýze pro možný sběr dat bude potřeba vybrat vhodnou PBX, obsahující Asterisk a následně se k němu připojit přes AMI.

3.1 Výběr telefonní ústředny

V závislosti na prozkoumání telefonního systému Asterisk a jeho možných distribucí byl vybrán takový systém, který co nejlépe vyhovuje zadání.

Varianty distribuce pro požadovanou činnost telefonní ústředny (kapitola *2.6 PBX systémy na bázi Asterisk*) lze shrnout do následujících možností. Buď konfigurovat telefonní systém z čisté podoby Asterisk, který by obsahoval všechny náležitosti telefonní ústředny, nebo využít standardní Asterisk distribuci, kvůli jeho AMI integraci, nebo komplexní distribuci, zahrnující automatické vygenerování skriptů a potřebných konfigurací.

Kapitola *2.6.3 Kerio Operator* přehledně uvádí funkce a výhody řešení produktu Kerio Operator. Automaticky generuje skripty potřebné pro celkový chod telefonní ústředny. Jednoduše přes webové grafické rozhranní umožňuje registrování agentů a zákazníků, vytváření a nastavování hovorových front, dynamické a statické přidělování agentů frontám, podporuje VoIP a SIP protokol a obsahuje AMI. Distribuce Kerio Operator byla proto vybrána jako telefonní ústředna pro následující potřebné činnosti sběru dat hovorové fronty.

3.2 Připojení k AMI

AMI používá TCP port nakonfigurovaný v souboru *manager.conf*, který je obsažen v telefonní ústředně (Kerio Operator).

Ukázka obsahu tohoto konfiguračního souboru pro možné připojení:

```
...
[general]
```

```
displaysystemname = yes
enabled = yes
port = 5038
bindaddr = 192.168.234.131
…
```

K demostraci připojení k AMI běžícím v prostředí Kerio Operator může být použit program Telnet, který realizuje komunikaci mezi dvěma počítači pomocí vlastního protokolu [13].

Podle nastavení konfiguračního souboru AMI a předpokladu běžícího prostředí ústředny se připojení vyvolá následujícím příkazem v konzoli (terminálu) operačního systému Linux, běžícím na počítači, kterým se chceme k Asterisk serveru připojit [14]:

```
$ telnet 192.168.234.131:5038
```

Navázání úspěšného připojení k Asterisku:

```
Trying 192.168.234.131...
Connected to 192.168.234.131.
Escape character is '^]'.
Asterisk Call Manager/1.1
```

Tato zpráva znamená, že komunikace může začít. Pakety mohou být vysílány v kterémkoliv směru kdykoliv po ověření.

Pro přihlášení a ověření je nutné zadat akci s autentizačními údaji:

```
Action: Login
ActionID: 1
Username: ami
Secret: ami
```

Odpověď AMI na úspěšnou autentizaci vypadá následovně:

```
Response: Success
ActionID: 1
Message: Authentication accepted
```

3.3 Typy paketů

AMI definuje 3 druhy možných paketů [13]:

Akce - Actions

Akce je druh paketu, který odesílá klient. Pouze klient může tyto akce generovat. Příkladem použití akce je zadání autentizačních údajů v kapitole *3.2 Připojení k AMI*.

Odpovědi - Responses

Každá vyvolaná akce má alespoň jednu odpověď, která uvádí výsledky z provedených (nebo požadovaných) akcí. Příkladem je odpověď na autentizaci připojení k AMI v kapitole *3.2 Připojení k AMI*.

Události - Events

K dispozici jsou dva typy událostí. Ty, které jsou připojené k určité reakci vyvolané na konkrétní akci, a ty, které Asterisk generuje kvůli informaci o připojený klientech a věcech, které se dějí na serveru (například volání události, změny proměnných hodnot, agentů a dalších klientů, kteří se připojují nebo odpojují do/ze serveru).

Pro monitorování provozu bude zapotřebí sledovat právě tyto události.

V následující kapitole bude názorně předvedeno připojení k AMI, provedení hovoru a analýza toku událostí.

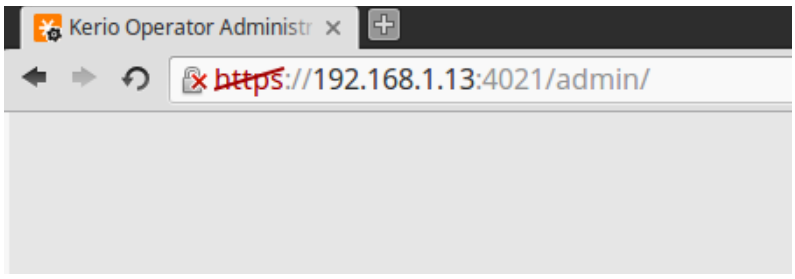
3.4 Sledování toku událostí

Aby se dalo analyzovat využití AMI pro monitorování provozu hovorových front, musí se nejdříve taková fronta (a jiné náležitosti) vytvořit ve správném softwaru.

Kvůli jednoduchosti bude jako komplexní řešení použita telefonní ústředna Kerio Operator v distribuci pro virtuální stroj Vmware. Lze tedy vytvořit provoz telefonní ústředny na jenom PC.

3.4.1 Konfigurace Kerio Operator

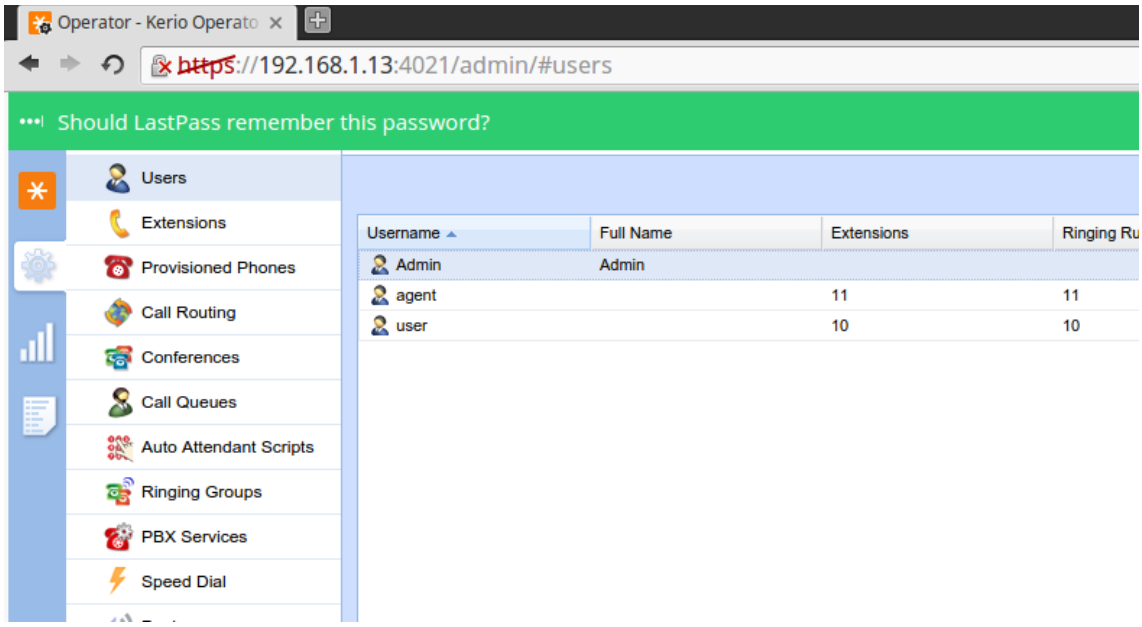
Kerio Operator poskytuje administrátorské GUI, dostupné ve webovém prohlížeči. Pro připojení stačí zadat adresu virtuálního serveru s příslušným portem jako adresu v prohlížeči (*Obrázek 5*).



Obrázek 5: Zadání adresy pro připojení do administrace Operatoru

V administraci vytvoříme uživatele agenta v záložce *Users* a přiřadíme jim linky (extensions) pro následnou registraci (*Obrázek 6*). Zároveň vytvoříme frontu (C*all Queues)* a přiřadíme vytvořeného agenta do této fronty. Předpokládáme, že je v Operator povoleno připojení aplikací třetích stran k AMI.

Agent bude mít telefonní číslo 11, zákazník 10 a hovorová fronta 01.



Obrázek 6: Administrátorské webové rozhraní Kerio Operator

3.4.2 Uskutečnění hovoru a sledování událostí

Založené linky (agenta, zákazníka) je potřeba registrovat v softwarovém telefonu pro uskutečnění hovoru.

Jeden z nejlepších open-source software pro VoIP telefonii je aplikace Linphone [15]. Má snadno konfigurovatelné grafické rozhraní (*Obrázek 7*), umožňuje registraci více telefonních linek a podporuje SIP. Pro demonstraci příkladu bude tento software v kombinaci s SIP protokolem použit.

Registrace linky přes SIP nabývá např. tento tvar:

```
sip:11@192.168.1.13
```

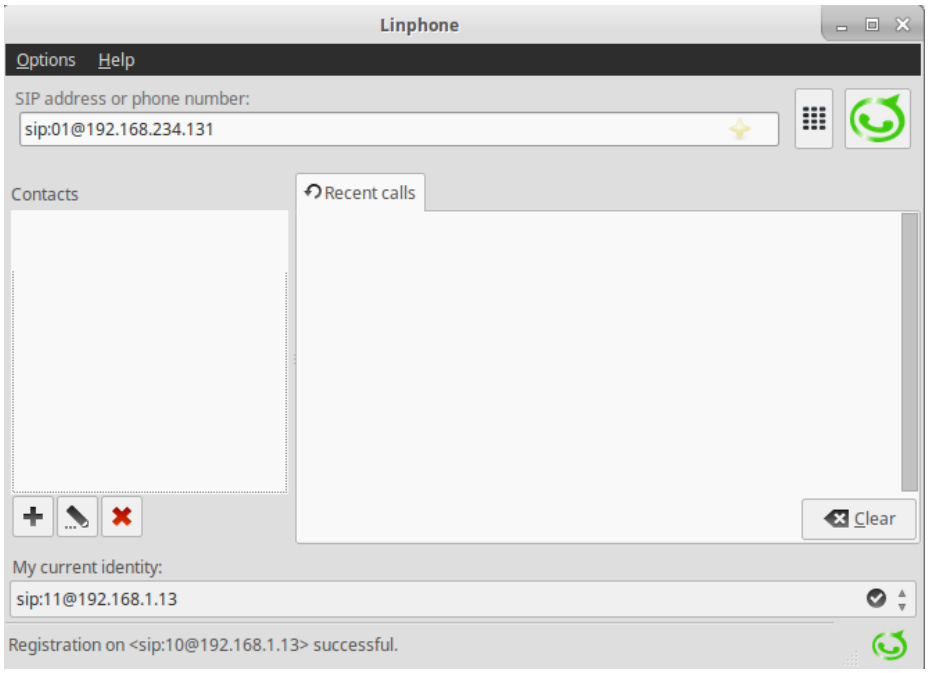
Vytočení hovoru (do hovorové fronty) má následující příkaz:

```
sip:01@192.168.1.13
```

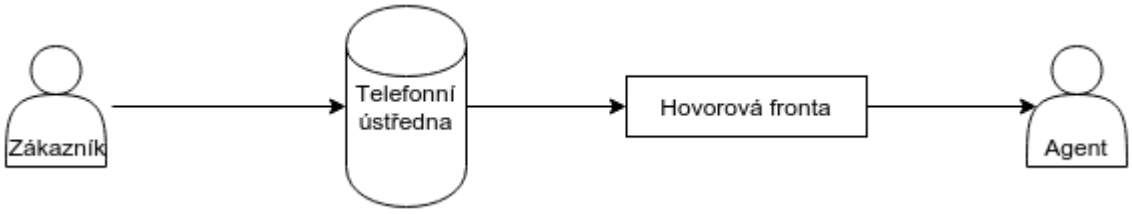
Scénář pro jednoduché uskutečnění hovoru

1. Zákazník vytočí číslo hovorové fronty, registrované v telefonním systému
2. Hovorová fronta přepojí hovor na dostupného agenta
3. Agent zvedne telefon
4. Zákazník zavěsí – konec hovoru

Scénář zobrazuje obrázek *Obrázek 8*.



Obrázek 7: Softwarový telefon Linphone.



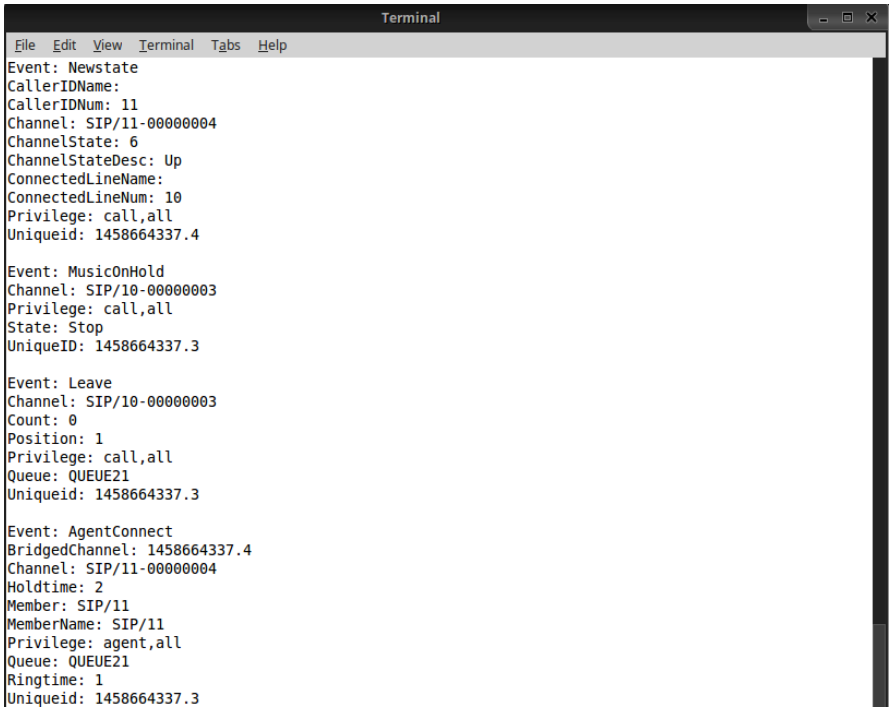
Obrázek 8: Diagram pro uskutečnění jednoduchého hovoru

Předpokládejme funkční připojení k AMI přes Telnet dle kapitoly *3.2 Připojení k AMI*. Po uskutečnění scénáře dle všech potřebných a výše zmíněných nastavení nám Telnet

začne vypisovat AMI události spjaté s hovorem (viz. *Obrázek 9*).

Ukázka struktury události ukončení hovoru z pohledu agenta, který byl k hovoru připojen:

```
Event: AgentComplete
Channel: SIP/11-00000004
HoldTime: 2
Member: SIP/11
MemberName: SIP/11
Privilege: agent,all
Queue: QUEUE21
Reason: agent
TalkTime: 26
Uniqueid: 1458664337.3
```



Obrázek 9: Výpis událostí službou Telnet.

3.4.3 Událostí AMI protokolu

Jak bylo zjištěno v předchozí kapitole, protokol AMI generuje různé události s různými atributy, které nabývají různých hodnot. Těchto událostí je v současné době přes 150 [17]. Události se dají rozdělit do několika kategorií a pro účely práce byly rozebrány ty nejdůležitější z nich.

Události stavu agenta

Přihlášení/odhlášení agenta

– Agentlogin, Agentlogoff

- Obsahují atributy s informacemi o názvu agenta, přidělenému kanálu a unikátní identifikátor události. Tento identifikátor je obsažen ve většině událostí a je unikátní [16].

```
Event: Agentlogin
Agent: <agent>
Channel: <channel>
Uniqueid: <uniqueid>
```

Stavy agenta ve frontě

- QueueMemberStatus, QueueMemberAdded, QueueMemberPaused

- Tyto události zaznamenávají změny agentů v hovorové frontě. Může to být jejich stav, přidání nového agenta do fronty či dočasná přestávka.
- Obsahují např. atributy se současným stavem agenta (jestli vyzvaní, hovoří, na příjmu nebo nedostupný), název, kolik uskutečnil hovorů, název fronty, jaký byl jeho poslední hovor, atd.

```
Event: QueueMemberStatus
Queue: <queue>
Location: <location>
MemberName: <membername>
Membership: <membership>
Penalty: <penalty>
CallsTaken: <callstaken>
LastCall: <lastcall>
Status: <status>
Paused: <paused>
```

- AgentCalled, AgentConnect, AgentComplete

- Události se vyvolají po zavolání agenta, respektive připojení hovoru a ukončení hovoru.
- Obsahuje např. atributy doba hovoru, doba čekání, důvod zavěšení (agent,zákazník,přenos) a unikátní identifikátor, který je stejný pro tyto události během jednoho hovoru.

```
Event: AgentComplete
Queue: <queue>
Uniqueid: <uniqueid>
```

```
Channel: <channel>
Member: <member>
MemberName: <membername>
HoldTime: <holdtime>
TalkTime: <talktime>
Reason: <reason>
```

Události stavu hovoru

- Dial

  - Zaznamenává vytočení v rámci PBX.

- Hangup

  - Zaznamená konec hovoru a jeho zavěšení.

- ExtensionStatus

  - Zaznamená změnu registrované linky a obsahuje stav této změny.

- MusicOnHold

  - Pokud se zákazníkovi přiřadí vnitřní kanál a jestě není spojen s agentem, slyší hudbu z ústředí a vyvolá se tato událost. Obsahuje mimo jiné atribut, který značí začátek/konec vyzváněcí hudby.

- Bridge

  - Agentovi i zákazníkovi je přiřazeno vždy číslo kanálu v ústředně. Tato událost spojuje tyto dva kanály, což značí začínající hovor mezi nimi.

- Join

  - Tato událost se vyvolá po připojení zákazníka v hovorové frontě, do které volá.
  - Obsahuje název fronty, kanál účastníka, unikátní ID, telefonní číslo.

- Newcallerid

  - Vyvolá se při přiřazení nového unikátního id zákazníkovi.
  - Obsahuje atributy telefonní číslo zákazníka, jméno (pokud je číslo v PBX registrováno), kanál a unikátní identifikátor.

Struktura události newcallerid:

```
Event: NewCallerid
Channel: <channel>
CallerIDNum: <callerid>
CallerIDName: <callerid>
Uniqueid: <uniqueid>
```

Události stavu záznamenávání

Tyto události se vyvolávají pro vnitřní zaznamenávání (logování) systému.

Stavy sytému

Do této kategorie patří vnitřní události Asterisku. Například vyvolání alarmu, stavu nerušit, vytvoření kanálu pro zaznamenávání, registrování SIP.

Stavy uživatele

Čistě jenom logování o vytvoření unikátního identifikátoru k číslu kanálu ve tvaru:

```
Event: UserEvent
Channel: <channel>
Uniqueid: <uniqueid>
```

Stavy s kontextovými údaji

Tyto události se vyvolávají velmi často. Obvykle značí nějaký obecný stav, který často systém kontroluje a tím se události často vyvolávají.

- Status

  - Obecný stav přiřazeného kanálu. Obsahuje např. atributy s názvem registrované linky, telefonní číslo, prioritu.

- QueueMember

  - Stavy členů fronty.

```
Event: QueueMember
Queue: name
Location: SIP/101
Membership: static
Penalty: 0
CallsTaken: 0
LastCall: 0
```

- QueueParams

  - Statistiky fronty – název, kolik bylo hovorů, kolik hovorů se dokončilo a kolik bylo odmítnuto.

4 Sběr dat do relační DB

V kapitole *3 Analýza Asterisk Manager Interface protokolu* byly rozebrány možnosti událostí AMI protokolu při uskutečnění hovoru v hovorové frontě Kerio Operátora. Tyto události, samozřejmě i s náležitými atributy, bude vhodné uložit do databázového systému pro další zpracování.

4.1 Vhodnost relační databáze

V případě vytváření relační databáze jsou známé výhody tohoto řešení. Data jsou snadno udržovatelná a dobře se udržuje i datová integrita. Jelikož data nejsou redundantní, budou zabírat také méně místa na úložišti a při změně výpisu z databáze se může měnit jen SQL dotaz a není potřeba znova vygenerovat celou databázi. Obecně ale například při vyskytnutí dalšího speciálního atributu znamená změnu celého schématu a nebude se jednat o jednoduchou úlohu. Navíc díky transakčnímu přístupu manipulace s daty nemusí vždy ukládání vykazovat velkou rychlost.

4.1.1 NoSQL

NoSQL (Not Only SQL) jsou databáze, které nepracují s relačním řízení dat. Nejsou primárně postavené na tabulkách a nevyužívají jazyk SQL pro práci s daty. Hlavní výhodou je vysoká optimalizace pro vyhledávání, avšak s malou funkcionalitou, často omezenou na jednoduchém ukládání dat.

NoSQL se hodí pro ukládání velkého množství dat (i strukturovaných). Pro taková data ale nelze uchovávat vzájemné vztahy a neumožňují transakce s podporou ACID, čímž na základě výsledné shody (Eventual Consistency) získáme větší dostupnost a škálovatelnost (navýšení kapacity DB při jejím nedostatku). Také distribuovaná architektura, která spočívá v rozložením umístění na více serverech, zprostředkovává větší odolnost vůči chybám. Díky potřebě ukládat velký objem dat by mohla být technologie NoSQL vhodná jako ekvivalentní přístup k řešení problému.

4.1.2 Porovnání RDB a NoSQL pro sběr dat

V předchozích kapitolách byly přiblíženy výhody a nedostatky těchto typů databází.

Pokud bychom nepředpokládali přechod z dočasné „produkční“ databáze do datového skladu, jistě by nějaká forma NoSQL databáze byla pro ukládání velkého množství nestrukturovaných dat výhodnější. Absence jazyka SQL, v rámci složitých dotazů při vytváření datového skladu, nám NoSQL databáze v tomto případě neposkytne ideální řešení. Zaměříme-li se také na možné proměnlivé atributy, které se vždy nemusí vyskytnout, relační model nám poskytne vhodnější řešení.

4.1.5 Relační DBMS

DBMS je databázový server, který spravuje databáze, komunikaci s klienty (lokálními nebo vzdálenými), vstupy a výstupy dat a jejich integritu. [19]

Tento databázový server musí být schopen efektivně pracovat s velkým množstvím dat a také musí být schopen řídit (vkládat, modifikovat, mazat) a definovat strukturu těchto perzistentních dat.

Charakteristické vlastnosti RDBMS [21]

- podpora pro definici relačních datových modelů
- správa klíčů: vlastní indexování, dodržování unikátních hodnot ve sloupcích, nad kterými je definován unikátní nebo primární klíč, implementace cizích klíčů
- využití některého jazyka vyšší úrovně pro manipulaci a definici dat (např. SQL) a vyřešení komunikačního kanálu mezi uživatelem či skriptem a DBMS
- autentizaci uživatelů a jejich autorizaci k operacím nad daty
- správu transakcí, atomičnost jednotlivých příkazů
- robustnost a zotavení po chybách bez ztráty dat
- uložené procedury
- triggery
- integritu dat
- kanály pro hlášení zpráv po úspěšně vykonaných dotazech, chybových hlášek, varování

4.2 Přehled vybraných RDBMS

4.2.1 PostgreSQL

PostgreSQL je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Jeho historie sahá do roku 1986. Za jeho počátkem stojí Michael Stonebraker z University of California v Berkeley. Milníkem ve vývoji PostgreSQL je rok 1996, kdy byl dotazovací jazyk POSTQUEL nahrazen jazykem SQL a jeho vývoj se odklonil od akademické obce směrem k open source. Od té doby prošel PostgreSQL dlouhým vývojem a díky práci stovek vývojářů z celého světa patří dnes ke špičce v oblasti databázových systémů. Běží nativně na všech rozšířených operačních systémech včetně Linux, UNIX a Windows. Splňuje podmínky ACID, plně podporuje cizí klíče, operace JOIN, pohledy, spouště a uložené procedury. Obsahuje většinu SQL92 a SQL99 datových typů a nechybí ani podpora moderních datových typů jako je JSON nebo XML. [20]

Podporuje také běh uložených procedur v několika programovacích jazycích. Mezi tyto jazyky patří Perl, Python, C, PL/pgSQL, což je speciální varianta jazyka vycházejícím z PL/SQL od firmy Oracle, nebo programovací jazyk Java (přes JDBC). [23]

PostgreSQL lze jednoduše nainstalovat ze všech veřejných depozitářů linuxových distribucí (případně repositáře vyhrazeného pro PostgreSQL). Je šířen pod vlastní licencí, která vychází z BSD a MIT licence. Umožňuje neomezené bezplatné používání, modifikaci a distribuci PostgreSQL a to ať pro komerční nebo nekomerční využití. PostgreSQL se může šířit se zdrojovými kódy nebo bez nich, zdarma nebo komerčně. Má vynikající pověst pro svou spolehlivost a bezpečnost [20].

4.2.2 MySQL

MySQL je databázový systém, který byl vytvořen švédskou firmou MySQL AB v roce 1995 a jeho autory jsou Michael Widenius a David Axmark. Je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. Nyní vlastněný společností Sun Microsystems, dceřinou společností Oracle Corporation.

Komunikace s MySQL probíhá pomocí jazyka SQL. Je snadně implementovatelný (např. OS Linux, Windows) a díky tomu, že je výkonný a jedná se o volně šiřitelný

software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace Linux, Apache, MySQL a PHP, jako základní software webového serveru (technologie LAMP). [22]

Mezi další podporované programovací jazyky patří např. C, C#, C++, Perl, Ruby, Python. [23]

MySQL bylo od počátku optimalizováno především na rychlost. Má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, vzhledem k možnostem konkurenčních produktů a výzvy stávajících uživatelů produktu.

4.2.3 MS SQL

Microsoft SQL Server označovaný také jako MS SQL je databázový systém od společnosti Microsoft, jehož první verze se objevila v roce 1989. Je využívaný ve většině případů s technologiemi firmy Microsoft. Může být volbou pro webové aplikace na platformě Windows (pomocí .NET nebo ASP), aby je bylo velmi snadné připojit s databází MS SQL. Microsoft tím nabízí komplexní služby pro vývojáře, které umožňují zkrácení času time to business, a tím snížení celkových nákladů na vývoj. [24]

MS SQL je komerční software, který se rozlišuje na několik edic. Je možné využít licenční sadu od verze Express, která je určena pro stolní počítače a jako jediná je dostupná pro nekomerční účely zdarma, až po verzi Enterprise, která je ideální pro náročnou databázi a požadavky řešení business intelligence a nabízí nejvyšší úrovně služeb a výkonnosti pro nejdůležitější firemní úlohy. [25]

Podporovaných programovacích jazyků je celá škála. Mimo výše zmíněných jsou to např. Visual Basic, Java, Python. [23]

Jak bylo také zmíněno, přednostně je tento produkt vázán s technologiemi firmy Microsoft, tudíž oficiální podpora je pouze pro OS Windows.

4.2.4 Oracle

Oracle databáze (někdy také Oracle RDBMS nebo jednoduše Oracle) je databázový systém společnosti Oracle Corporation. Její vývoj začal v roce 1977, kdy Larry Ellison, v současnosti výkonný předseda a technologický ředitel, a jeho dva spolupracovníci založili firmu Software Development Laboratories (později Oracle Corporation). Tato

firma o několik let později vydala první komerční databázový produkt s názvem Oracle. [26]

Společnost Oracle je lídrem na trhu v oblasti databázových řešení. Ve velké míře je jejich systém využíván ve státní správě a ve velkých firmách, kde je nutné zabezpečit co nejrobustnější řešení. Co se týče architektury databáze, Oracle má vždy k jedné instanci serveru pouze jednu databázi. U ostatních systémů je možné v jedné instanci souběžně obsluhovat několik databází od sebe oddělených. Oracle využívá rozšíření pomocí uložených procedur a triggerů napsaných ve svém proprietárním jazyku PL/SQL, nebo je možné psát jejich kód v jazyce Java. [23]

Oracle RDBMS je dostupná v několika různých verzích, které se od sebe liší. Cena se pak odvíjí podle volby Edice, počtu procesorů či počtu připojených klientů, takže se předpokládají vysoké náklady na verze systému pro náročnější aplikace. [26]

4.2.5 Výběr RDBMS

Pro průběžné ukládání dat o provozu hovorové fronty byl zvolen RDBMS v podobě řešení PostgreSQL, jelikož je spolehlivý a bezpečný, podporuje hlavní standardy jazyka SQL a umožňuje neomezené bezplatné používání. Pohybuje se také na předních příčkách popularity relačních DBMS [27].

4.3 Návrh datového modelu

V kapitole *3.4 Sledování toku událostí* byly rozebrány události AMI protokolu, které Asterisk generuje při provozu hovorové fronty. Bylo potřeba vytvořit takový datový model, aby byla zaručena integrita dat, omezila se duplicita a správně se zachovala struktura a časový sled událostí.

4.3.1 Schéma datového modelu

Evoluce řešení

Předpokladem pro vytvoření modelu relační databáze je vytvořit takový model, který by byl schopen ukládat značné množství událostí spjaté s provozem hovorové fronty. Jak již bylo řečeno, AMI protokol postupně generuje tyto události a s nimi je nutné i ukládat

atributy (jejich názvy) s nabytými hodnotami. Některé atributy nemusí vždy tyto hodnoty obsahovat. Důležité také je, aby se uchovávala časová značka proběhlých událostí při další identifikaci hovoru.

Jednoduchou podobu modelu by mohla ztvárnit i jedna tabulka, jejíž sloupce by obsahovaly hodnoty časové značky, názvu události a konečného počtu všech možných atributů všech událostí. Toto řešení však není dostatečné. Z hlediska redundance dat nesplňuje 3. normální formu a v tabulce by vznikala zbytečná duplicita dat, která by při velkém množství zbytečně zatěžovala databázový server.

Zjednodušení tohoto modelu nabízí rozklad událostí a jejich atributů. Tímto by vzniklo zbytečně moc tabulek, kdy každá z nich (kromě základní s atributy názvu události a její časové značky s vazbou 1:N) by naplňovala atributy každé události. Takto by se zamezilo velké duplicitě dat, ale vzniklo by velké množství tabulek (1 tabulka = 1 typ události). Negativním hlediskem tohoto řešení by byla stále zachovalá, byť nižší, duplicita dat a značná pomalost případných dotazů.

Pokud by události, které mají většinu podobných atributů, byly spojeny do společných tabulek, mohl by se počet tabulek snížit a zvýšit rychlost dotazů do databáze. Ani to není řešení, které by bylo ideální pro dočasnou databázi a splnění řádného vytvoření relačního databázového modelu.

Ideální podoba modelu musí odlišit události a jejich atributy, zároveň zachovat jejich strukturu a správně přiřadit časové značky a hodnoty atributů. Hodnoty atributů musí být provázány s názvem atributu a typem události podle předem dané struktury.

Finální podoba modelu

Na základě výše zmíněných možných podob modelu byl vytvořena finální podoba modelu dočasné databáze, která se dá dobře využít při tvorbě datového skladu. Podoba modelu je znázorněna na obrázku *Obrázek 10*.

Tabulka events

Tabulka *events* uchovává názvy událostí a jejich identifikátor, jež je současně primárním klíčem. Atribut názvu události (*name*) je datového typu řetězec.

Tabulka attributes

Tato tabulka uchovává veškeré možné atributy událostí, které se nacházejí v tabulce

events. Podobně jako tato tabulka obsahuje atribut názvu (*name*) a identifikátor (*id*) jako primární klíč.

Tabulka structure

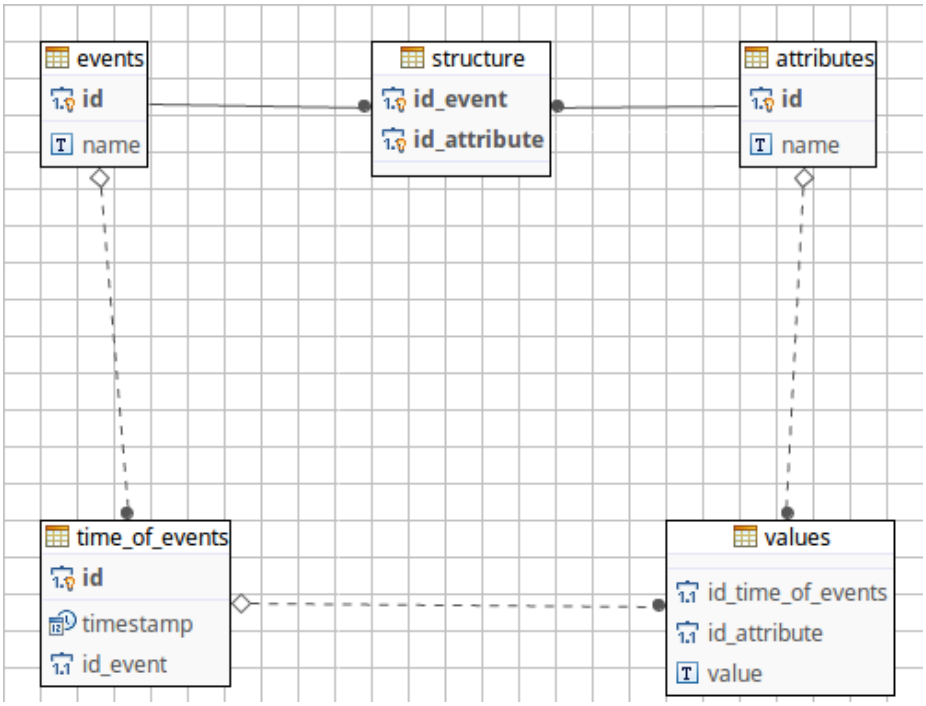
Tabulka *structure* uchovává strukturu událostí a jejich atributů. Hlavní atributy tabulky jsou cizí klíče identifikátorů tabulek *events* a *attributes*. S těmito tabulkami je spojena vazbami 1 ku N.

Tabulka time_of_events

Kromě identifikátoru jako primárního klíče obsahuje tabulka *time_of_events* cizí klíč tabulky událostí (*events*) a atribut *timestamp*, který přidává časovou značku každé proběhnuvší události. Je datového typu timestamp. S tabulkou *events* je ve vztahu 1 ku n, tedy že jedna událost může proběhnout vícekrát.

Tabulka values

Tabulka *values*, jak název vypovídá, uchovává hodnoty atributů událostí AMI protokolu. Každá tato hodnota atributu má jako cizí klíče identifikátory tabulek *attributes* a *time_of_events*.



Obrázek 10: Datový model relační databáze

4.3.2 Implementace modelu

Implementace modelu do relační DBMS PostgreSQL byla vytvořena s PostgreSQL.. PostgreSQL byl nainstalován v operačním systému Linux Mint. Jako vývojové grafické prostředí byla použita aplikace SQL Workbench/j, která pro komunikaci s PostgreSQL využívá Java rozhraní JDBC. Použité programy, jejich verze a data vydání jsou přehledně vypsány v tabulce *Tabulka 1*.

Název	Verze	Datum vydání
Linux Mint	17.3	07.12.2015
PostgreSQL	9.4.6	11.02.2016
SQL Workbench/j	Build 119	31.01.2015
JDBC	9.4.1208	23.12.2015

Tabulka 1: Verze software pro implementaci modelu relační databáze

Ukázka SQL kódu pro vytvoření tabulky *events:*

```
CREATE TABLE events
(
id smallint DEFAULT nextval('events_id_seq'::regclass) NOT NULL,
name  varchar(100) NOT NULL
);
ALTER TABLE public.events
   ADD CONSTRAINT events_pk
   PRIMARY KEY (id);
```

4.4 Vytvoření aplikace pro sběr dat

V kapitole 4.2 přehledu vybraných RDBMS bylo řečeno, že PostgreSQL podporuje širokou škálu programovacích jazyků. Jako jeden z doporučených je jazyk Java, který byl vybrán pro vytvoření aplikace pro ukládání dat z AMI do relační databáze.

4.4.1 Komunikace s AMI

Existují aplikace, které umožňují přípojení na AMI a jsou schopny provádět příkazy nebo číst události přes TCP/IP stream. Je to zvlášť užitečné, pokud jde o snahu sledování stavu telefonie uvnitř Asterisku.

Asterisk-Java

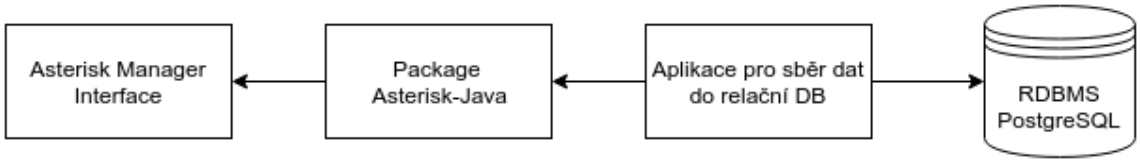
Asterisk-Java je framework v podobě balíku obsahující Java třídy, které umožňují Java

aplikacím komunikovat s Asterisk ústřednou (v tomto případě Kerio Operator). Podporuje obě rozhraní a to jak AGI, tak AMI. [28]

Je tedy ideálním řešením pro zprostředkovávání komunikace s telefonní ústřednou.

4.4.2 Návrh aplikace

Uvažujeme zahrnutí Asterisk-Java frameworku pro komunikaci s telefonní ústřednou. Model fungování aplikace znázorňuje *Obrázek 11*.



Obrázek 11: Schéma fungování aplikace

Aplikace se díky frameworku Asterisk-Java připojí k AMI. Po úspěšném přihlášení se toky událostí a jejich atributů budou ukládat průběžně do databáze. Před uložením se vytvoří časová značka události a událost a její atributy se vhodně rozparsují dle struktury vytvořené v relační databázi.

4.4.3 Implementace Java aplikace

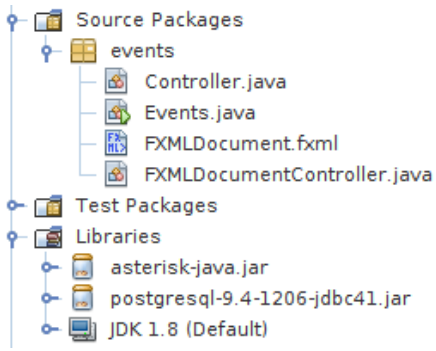
Jako vývojové prostředí (IDE) pro jazyk Java byla využita aplikace Netbeans a pro vytvoření grafického uživatelského rozhraní (GUI) technologie JavaFX a její Scene Builder, který lze přidat jako plugin do IDE Netbeans. Propojení s PostgreSQL zajistí JDBC a jako framework k připojení k AMI výše zmíněný Asterisk-Java. Použité programy, jejich verze a data vydání jsou přehledně vypsány v tabulce *Tabulka 2*.

Název	Verze	Datum vydání
Java Development Kit	Java 8 Update 66	16.11.2015
Netbeans	8.1	22.10.2015
JavaFX scene builder	2.0	21.03.2014
JDBC	9.4.1207	23.12.2015
Asterisk-Java	1.0.0.CI-SNAPSHOT	19.12.2014
VMware	12.1.0 build 3272444	08.12.2015
Kerio Operator pro VM	2.4.1 build 4587	30.06.2015

Tabulka 2: Verze software pro implementaci aplikace pro sběr dat

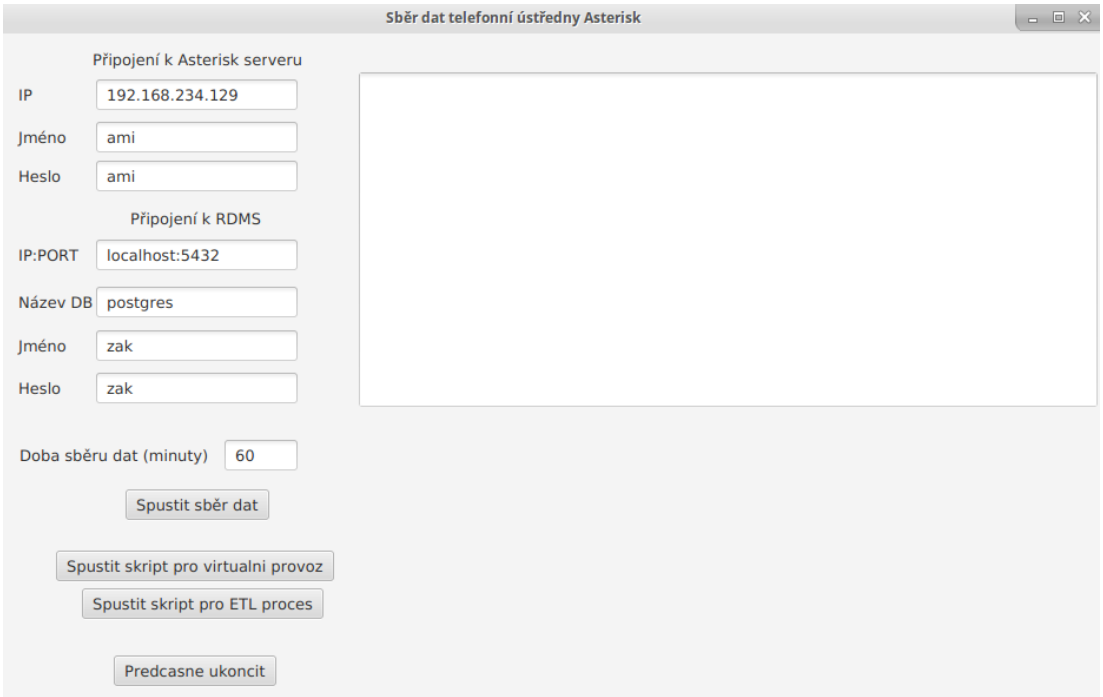
Struktura aplikace

Aplikace je rozdělena dvou částí. První je balík tříd, který obsahuje vlastní naprogramovanou funkčnost aplikace. Další částí jsou pak externí knihovny pro propojení s PostgreSQL databází a napojení na AMI a sledování událostí. Struktura je znázorněna na obrázku *Obrázek 12*.



Obrázek 12: Struktura aplikace

Třída *Events* obstarává připojení k AMI a předává události do třídy *Controller.* Tato třída obstarává rozparsování události a jeho atributů a její uložení do databáze. Obstarává také připojení k této databázi. Třída FXMLDocumentController pak obstarává funkčnost GUI aplikace a soubor fxml její vzhled. Vzhled aplikace pro sběr dat a jejich ukládání do databáze je zobrazen na obrázku *Obrázek 13*.



Obrázek 13: Vzhled aplikace pro sběr a ukládání dat do DB

Ukládání dat

Algoritmus ukládání dat probíhá následujícím způsobem:

1. Předpokládejme konečnou množinu událostí AMI protokolu. Tabulka *events* v databázi bude naplněna názvy těchto událostí (podle naplnění této tabulky můžeme definovat, jaké události se budou ukládat).
2. Naparsujeme řetězec události s atributy předaný z AMI zprostředkovaný připojením přes Asterisk-Java. Název události bude uložen do proměnné *nameEvent* jako řetězec. Taktéž uložíme do proměnné *timestamp* datového typu timestamp časovou značku zachycené události. Názvy atributů a jejich hodnoty budou ukládány do kontejneru. Tento kontejner bude dynamické pole (kolekce *arrayList – Tabulka 3*).

Řádek	Název atributu	Hodnota atributu
0	Atribut 1	Hodnota atributu 1
1	Atribut 2	Hodnota atributu 2
2	Atribut 3	Hodnota atributu 3
3	Atribut 4	Hodnota atributu 4
n	Atribut n	Hodnota atributu n

Tabulka 3: Konternej naparsované události

3. Tabulka *attributes* je naplněna názvy atributů událostí. Jako v případě událostí můžeme předem definovat, jaké atributy událostí se budou ukládat. V případě, že se atribut nebude nacházet v tabulce *attributes*, se záznam v kontejneru jednoduše přeskočí.
4. Identifikátory (cizí klíče) tabulek *events* a *attributes* se propojí v tabulce *structure* tak, aby struktura atributů v událostech zůstala zachována. Příkazem select vybereme identifikátor události podle uložené proměnné *nameEvent*, která obsahuje její název. Taktéž se přistupuje s vybráním identifikátoru atributu. Tyto identifikátory uložíme do proměnných pro další manipulaci.
5. Vytvořené proměnné času události její identifikátor z tabulky *events* se uloží do tabulky *time_of_events*.
6. Vybereme identifikátor času události (*id* v tabulce *time_of_events*), uložíme do

proměnné a spolu s hodnotou ukládáme tento identifikátor (jako cizí klíč) do tabulky *values.*

7. Opakujeme pro každou událost.

Ukázka Java kódu pro příkaz vložení získaných hodnot do tabulky *structure* a kontrola jejich existence:

```
String sql3 = "INSERT INTO structure (id_event,id_attribute) SELECT ?,
? WHERE NOT EXISTS (select * FROM structure WHERE id_event=? AND
id_attribute=?);";
preparedStmt = c.prepareStatement(sql3,
Statement.RETURN_GENERATED_KEYS);
preparedStmt.setInt(1, event_id);
preparedStmt.setInt(2, attribute_id);
preparedStmt.setInt(3, event_id);
preparedStmt.setInt(4, attribute_id);
preparedStmt.execute();
```

Složitost algoritmu

Pokud vezmeme v úvahu počet vstupů *N* jako počet událostí toku jdoucího postupně ke zpracování, bude asymptotická složitost řešení v nejhorším případě O(N), což nám zajistí, že výkonové problémy v případě hovorové fronty nemusí nastat. Pokud bude zapotřebí sledovat vícenásobný provoz, čtení dat může běžet paralelně na více vláknech či procesech.

5. Budování datového skladu

5.1 Business Intelligence

Bussiness Inteligence (zkráceně BI) je zastřešující termín, který se vztahuje ke znalostem, procesům, technologiím, aplikacím a postupům, které usnadňují podnikové rozhodování. Technologie BI pracuje s použitými (historickými) daty v požadovaném kontextu a pomáhá přijímat podniková rozhodnutí pro budoucnost.

BI se soustřeďuje hlavně na interní informace o provozních aspektech, které souvisejí s taktickým a strategickým plánováním. Informace jsou obvykle nějakým způsobem uspořádány a přímo nebo nepřímo se odvozují z aktuálních podnikových procesů. Výstup základních dat se používá hlavně k interní analýze, ale lze jej také zkombinovat s externí analýzou, jako je např. kompetitivní analýza *SWOT.* [32]

Příklady využití získaných informací:

- změny v podílu na trhu
- změny v chování a preferencí zákazníků
- objem prodeje jednotlivých produktů

Tyto informace pomáhají manažerům lépe pochopit podmínky a postavení firmy na trhu ve srovnání s konkurencí, snížit nejistotu v rozhodování a lepší přizpůsobivost měnícím se podmínkám okolí. [32]

Účelem Bussiness Intellingence je konverze velkých objemů dat z transakčních systémů na poznatky, které jsou potřebné pro koncové uživatele a lze je pak efektivně využít v rozhodovacím procesu.

5.1.1 Co zahrnuje BI

Do oblasti BI lze zahrnout nejrůznější formy reportování, online analýzy, statistické analýzy, predikce a dolování dat.

BI podporuje strategické rozhodování v organizaci v kladení otázek:

- jak využít potenciál stále se množících firemních dat o podnikání

- jak účinně a včas dodávat správným lidem správné informace, které jsou důležité pro rozhodování

Jedná se o analytické a vykazovací nástroje, které umožňují využít firemní data k analýze již proběhlých jevů.

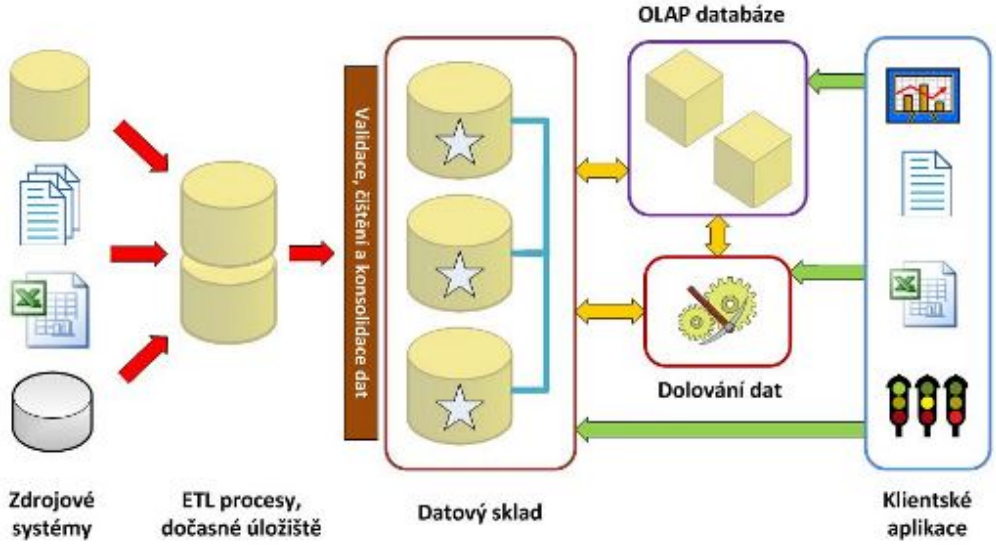
5.1.2 Důvody pro zavedení BI

Každá společnost si dokáže ze svých transakčních systému zjistit základní informace, např. jaký má zisk a jaké jsou náklady. Méně z nich dokáže zjistit, jaké faktory ovlivňují jejich hodnoty a jaká je struktura zákazníků, která ovlivňuje firemní zisk. Získání těchto informací ze standardních systémů provází většinou delší trvání a s tím související vyšší náklady. Hlavními důvody k použití BI nástrojů jsou v poskytování informací včas, přehledně (v požadované struktuře) a bez nutnosti dožadování se výstupů od IT oddělení společnosti. [30]

5.2 Komponenty BI

BI řešení se z pohledu architektury skládá z několika základních komponent. Jejich vazby přehledně zobrazuje schéma na obrázku *Obrázek 14*.

Data ze zdrojových systémů jsou pomocí procesů ETL plněna do datového skladu, nad kterým je pro zrychlení analýzy postavena databáze OLAP. Interakci BI řešení s koncovým uživatelem pak zprostředkovávají klientské aplikace.



Obrázek 14: Vazby komponent Bussiness Inteligence

5.2.1 Zdrojové systémy

Jak již bylo řečeno, data z informačních systémů jsou typicky ukládána v transakčních databázích. OLTP (Online Transaction Processing) je technologie právě takového uložení dat v databázi, která umožňuje jejich co nejsnadnější a nejbezpečnější modifikaci v víceuživatelském prostředí. [31]

Tyto systémy jsou zdrojem našich informací pro kontrolní činnosti, nejsou však vhodné pro zpracování a analýzu těchto dat. Jsou primárně optimalizována pro zpracování velkého množství operací (transakcí). Data jsou v databázích OLTP z důvodu odstranění redundancí typicky ukládána ve značném počtu vzájemně provázaných tabulek.

Získat z takové databáze například odpověď na otázku:

Jaký byl objem prodeje v daném regionu v daném časovém období ve srovnání s předešlým rokem?

znamená propojit množství tabulek a agregovat velmi mnoho záznamů. Dnešní relační databázové stroje si sice s takovým úkolem poradí, ale na odpověď však se musí dlouho čekat a navíc výsledek bude databázový server počítat na úkor ostatních úloh, tedy transakcí zajišťujících skutečný chod podnikových procesů. To může vést k jejich zpomalení nebo dokonce dočasnému zastavení, což je z pochopitelných důvodů v lepším případě nežádoucí, v tom horším nepřípustné.

Mimo to jsou data ve většině organizací uložena v různých systémech, které využívají odlišných platforem a mezi nimi nemusí být žádná přímá vazba. Proto data, která jsou získaná z těchto systémů, nejsou bez dalšího zpracování konzistentní a nelze je snadno použít pro získání korektních a smysluplných informací. [31]

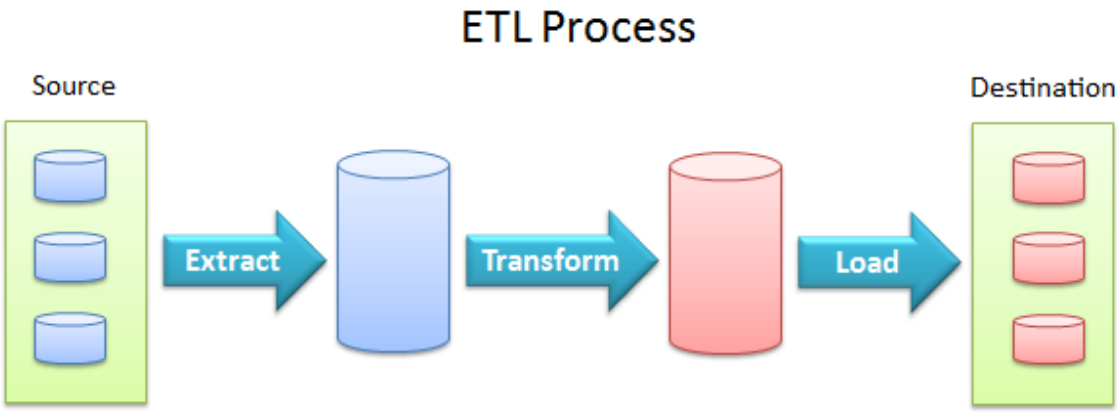
5.2.2 Komponenty datové transformace

Zkratka ETL (Extract, Transform, Load) označuje nástroje pro automatizované plnění databázové komponenty (datového skladu) daty z různorodých datových zdrojů.

Zdrojové systémy velice často dostatečně nekontrolují, zda zadané údaje splňují nutné podmínky pro platnost, jestli jsou vzájemně konzistentní nebo zda nejsou duplicitní. Úlohou procesů ETL je tak nejen načtení dat ze zdrojových systémů a jejich uložení do centrálního datového skladu, ale zejména jejich integrace, validace, čištění a transformace tak, aby datový sklad obsahoval pouze správná a důvěryhodná data.

Ta jsou klíčem k získávání přesných výstupů z BI řešení a jedině na jejich základě lze pak přijímat správná rozhodnutí. Toto úložiště mimo jiné umožňuje jednou načtená data opětovně v transformacích využít bez nutnosti zatěžovat opakovaně zdrojové systémy OLTP. [33]

Schéma ETL procesu znázorňuje *Obrázek 15*.



Obrázek 15: ETL proces [33]

Extrakce

Extrakce je první krok v procesu získávání dat do prostředí datového skladu. Data jsou obvykle uložena v různých zdrojových systémech (např. databázové systémy, SAP, ERP, textové soubory) a je potřeba je převést do stanoveného jednotného formátu budoucího datového skladu. Poté jsou data připravena pro další manipulace, tedy proces transformace.

Transformace

Úloha přeměny dat může produkovat následující úlohy [30]:

- aplikování obchodních „bussiness“ pravidel
  - takzvaných derivací, například nová opatření měřítek a dimenzí
- čistění
  - např. přemapování *NULL* na *0*
  - zkracování výrazů → *Muž* na *M*, *Žena* na *Ž*

- filtrování
  - např. výběr jen některých sloupců tabulky
- spojování a rozdělování sloupců tabulek
- spojování dat z různých zdrojů
  - např. SQL funkce *merge* či *lookup*
- transpozice řádků a sloupců tabulek
- validace dat
  - aplikování na jednoduchých nebo komplexních datech
  - např. odstranění záznamů z procesu transformace, pokud nějaký atribut záznamu je prázdný

Načtení

V poslední fázi ETL procesu je potřeba získaná, upravená a očištěná data uložit do formy databázové komponenty jako cílového systému. Nejčastěji jsou používány datové sklady, popřípadě datová tržiště (Data Marts) či marketingové databáze (Marketing Database). [34]

Jelikož záleží na povaze zdrojových dat a požadavcích na cílový systém, náročnost ukládání může být různá. Záleží na formě datového skladu, protože některé systémy mohou pouze přepisovat staré informace novými, jiné zase požadují aktualizovat inkrementálně. Existují i komplexnější systémy, které udržují historii a audit všech změn v datech.

5.2.3 Databázové komponenty - Datový sklad

Řešením odstraňujícím problémy s různorodými zdroji je spolehlivé centrální úložiště konzistentních a důvěryhodných dat, známé jako datový sklad (označovaný zkratkou DW z anglického Data Warehouse). Ten je základem architektury BI, protože poskytuje jeden pravdivý pohled pro celou organizaci.

Definice datového skladu

Pro datový sklad lze využít definici Billa Inmona, jednoho ze zakladatelů datových skladů [30]:

„Datový sklad je integrovaný, subjektivně orientovaný, stálý a časově rozlišený souhrn dat, uspořádaný pro podporu potřeb managementu“.

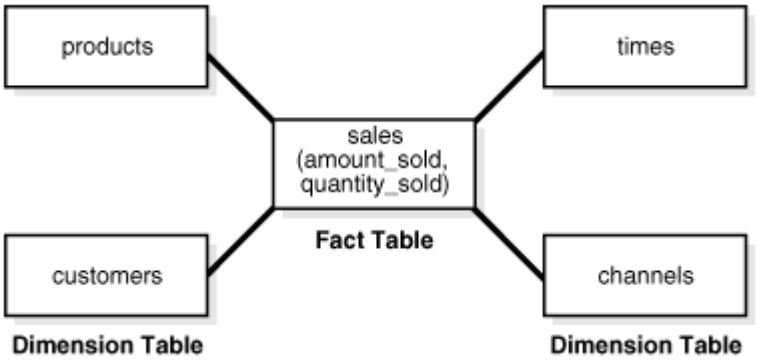
Datový sklad typicky obsahuje veškerá data dostupná v podniku, a to jak aktuální, tak i historická. Jedná se sice obvykle o relační databázi, její struktura je však optimalizovaná pro datové analýzy.

Schéma datového skladu

Schéma je sbírka databázových objektů, včetně tabulek, pohledů, indexů a synonym. Lze uspořádat objekty schématu v modelech schématu určených pro datové sklady mnoha různými způsoby. Většina datových skladů používá dimenzionální model. [35]

Hvězda

Aby bylo možné informace z datového skladu získávat rychle, jsou tabulky organizované ve strukturách připomínajících hvězdu. Schéma hvězda je nejjednodušší datové schéma datového skladu. Tato struktura umožňuje získat potřebné informace bez nutnosti spojení velkého množství tabulek.



Obrázek 16: Schéma datového skladu typu "hvězda" [35]

Střed hvězdy se skládá z tabulky faktů obsahující sledované číselné hodnoty určené k analýze, které se nazývají měřítka a body hvězdy, v podobně tabulek jako návazností na tabulku faktů, jsou tabulky dimenzí. Jak je ukázáno na obrázku *Obrázek 16*, měřítkem tabulky faktů může být prodejní částka a počet prodaných produktů v případě, pokud sledujeme prodeje. Dimenzionální tabulky zase v tomto příkladu popisují nějakou vyskytující se entitu, jako je čas, produkty a zákazníci.

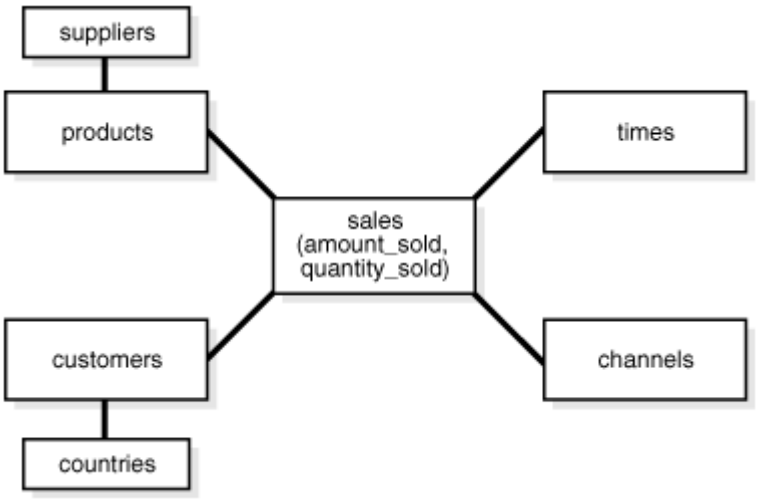
Tyto dimenze poskytují význam jednotlivým číselným hodnotám měřítek uložených v tabulce faktů. Ke konkrétnímu počtu prodaných kusů se pak dá z tabulky dimenze

vyčíst o jaký produkt se jedná, kdy byl produkt prodaný, odkud pochází zákazník a kolik kusů se prodalo.

Sněhová vločka

Sněhová vločka je složitější schéma datového skladu než hvězda. Stejně jako hvězda je název sněhové vločky odvozen od podoby jejího diagramu.

Schéma sněhové vločky se snaží normalizovat dimenze kvůli eliminaci nadbytečností. To znamená, že data dimenze jsou rozdělena do více tabulek namísto jedné velké. I když toto řešení šetří kapacitu, zvyšuje se počet tabulek dimenzí, které vyžadují k připojení cizí klíč. Výsledek jsou pak složitější dotazy a snížený výkon dotazu.



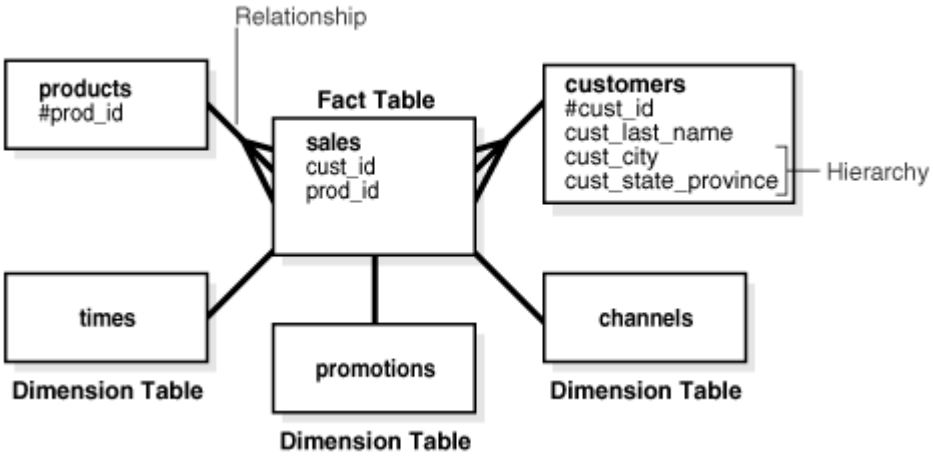
Obrázek 17: Schéma datového skladu typu "sněhová vločka" [35]

Na obrázku *Obrázek 17* je grafické znázornění příkladu tohoto datového schématu. Dimenzionální tabulka produktu je rozšířena o dimenzi dodavatele a tabulka zákazníků o dimenzi země, z jaké pochází.

Hierarchií se pak nazývá tato logické struktura, která používá pořadí úrovní jako prostředku k uspořádání dat. Úroveň tedy poskytuje pozici v dané hierarchii.

Vztahy mezi úrovněmi specifikují uspořádání úrovní „shora-dolů“, tedy od širšího uspořádání od kořene na užší, již konkrétní informace. Definují vztahy rodič-dítě mezi úrovněmi v hierarchii. Hierarchie jsou také základními komponenty umožňující složitější přepisování. Například databáze může agregovat existující tržby ve čtvrtletní

bázi na roční agregaci, když jsou známy dimenzionální závislosti mezi těmito obdobími. Objekty datového skladu a jejich vztahy jsou zobrazeny na obrázku *Obrázek 18*.



Obrázek 18: Příklad objektů datového skladu a jejich vztahů [35]

Data v datových skladech se na rozdíl od databází OLTP nemění a používají se jen ke čtení. Výjimkou z tohoto pravidla je plnění nových dat, které se provádí v pravidelných intervalech tak často, aby byly splněny požadavky na aktuálnost informací pro účely analýzy. Zdaleka nejčastěji se používá plnění na denní bázi, kdy je možné analyzovat data z předešlého dne. Samozřejmě ale záleží na předem stanoveném intervalu.

5.2.4 Analytické komponenty

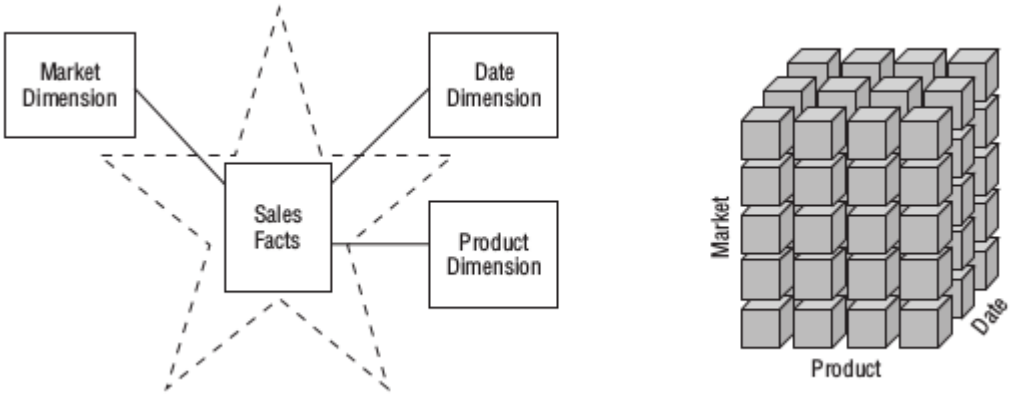
Analytické komponenty pokrývají činnosti spojené s vlastním zpřístupněním dat a jejich analýzou.

Analytické komponenty se dají rozdělit na:

- Systémy On-line Technologie Analytického Zpracování (OLAP) - vrstva zaměřená na pokročilé a dynamické analytické úkoly.
- Dolování dat (Data Mining) - systémy zaměřené na sofistikovanou analýzu velkého množství dat.
- Reporting - analytická vrstva zaměřená na prostředky pro prezentaci dat, které je možné analyzovat.

OLAP

Dimenzionální modely realizované v relačních databázových systémech jsou označované jako hvězdy, protože jejich schéma je podobné s hvězdicovitou strukturou. Dimenzionální modely realizované v multidimenzionální databázovém prostředí jsou označovány jako OLAP kostky, jak je znázorněno na obrázku *Obrázek 19*. Systémy OLAP jsou tedy systémy, jejichž technologie je založená na multidimenzionální databázi. Data ze schématu hvězda lze také zobrazit jako kostku (ROLAP).



Obrázek 19: Rozdíl mezi schématem "hvězda" a OLAP kostkou

Tento hlavní princip multidimenzionální tabulky umožňuje pružně měnit jednotlivé dimenze a umožnit tak uživateli sledovat data týkající se ekonomické reality podniku z různých pohledů (neboli z pohledu různých zaměnitelných dimenzí). [30]

OLAP se liší od klasických transakčních systémů (OLTP) hlavně účelem vlastního použití. V kapitole *5.2.1 Zdrojové systémy* bylo řečeno, že OLTP systémy jsou zdrojem informací pro kontrolní činnost, zpracovávají velké množství transakcí, napomáhají automatizaci a optimalizaci. OLAP pracují s informacemi pro analytickou činnost, které jsou odvozeny z dat transakčních systémů a jsou určeny především pro podporu rozhodovacích činností organizace.

Data pro OLTP jsou nejčastěji ukládána v relačních databázích v normalizované podobě (3. Normální Forma). Analytická data není vhodné ukládat tímto způsobem, neboť pokud je potřeba poskytnout uživateli možnost rychle nahlížet na data z pohledu různých dimenzí, lze to v případě normalizovaně uložených dat zajistit jen velmi obtížně. Data pro OLAP jsou proto ukládána v multidimenzionální struktuře, která je optimalizována pro uložení a analýzy multidimenzionálních dat.

Hierarchická struktura vytvořených dimenzí definuje různé úrovně agregace dat a zahrnuje také faktor času a díky tomu můžeme sledovat historický vývoj definovaných ukazatelů. Multidimenzionální analýzy, probíhající nad OLAP kostkou, získávají hodnotu vytvořeného ukazatele, který přísluší ke zvoleným dimenzím. Na obrázku *Obrázek 19* při zvoleném ukazateli Market (Trh) určíme dimenze Product (produkt) a Date (datum-čas). Takto bychom mohli kombinovat další dimenze se zvolenými ukazateli. Volbou určité kombinace dimenzí je určen prvek multidimenzionální databáze, obsahující hodnotu nebo algoritmus pro výpočet této hodnoty. Standardním ukazatelem je obvykle ekonomická proměnná, která je sledována přes dimenzi času a současně přes několik dalších dimenzí. [31]

Pokud by totiž bylo nutné provádět součty mnoha hodnot až při zobrazování dat odpovídajících zvoleným pohledům, odezva systému by mohla být příliš velká. Předem vypočítané a v OLAP kostkách uložené hodnoty agregovaných dat, odpovídající jednotlivým hierarchiím dimenzí, umožňují snadno měnit detail zobrazovaných dat a pružně zaměňovat dimenze, přes něž jsou data nahlížena.

Technologie pro uložení OLAP kostek je možno rozčlenit na:

- ROLAP - data zůstávají v původních relačních databázích. Oddělená sada relačních tabulek je použita k uložení agregací. ROLAP je vhodný pro rozsáhlé databáze nebo na stará data, která nejsou často analyzována.
- MOLAP -poskytuje nejlepší výkon ve fázi analýzy, neboť je právě pro mnoha dimensionální dotazy speciálně optimalizován. Je vhodný pro malé až středně velké objemy dat, kdy kopírování všech dat.
- HOLAP - slučuje prvky MOLAP a DOLAP. Ponechává původní data v relačních tabulkách, ale ukládá agregace v multidimensionálním formátu. Poskytuje propojení mezi rozsáhlými objemy dat v relačních tabulkách a zároveň nabízí výhodu rychlejšího výkonu multidimensionálně uložených agregací.
- DOLAP – umožuje se připojit k centrálnímu úložisti dat OLAP a stáhnout si potřebnou podmnožinu kostky na lokální úložiště. Analytické operace jsou pak prováděny na touto kostkou, takže není potřeba být připojený k serveru s OLAP. Toto je výhodné zejména pro mobilní aplikace.

Základní operace v OLAP systémech [36]

- Drill-down – umožňuje uživateli ve zvolených instancích jisté agregační úrovně nastavit jemnější agregační úroveň
- Roll-up – jde o opak předešlé operace. Ve zvolených instancích jisté agregační úrovně nastavuje hrubší agregační úroveň.
- Pivoting – umožňuje „otáčet“ datovou krychlí, tj. měnit úhel pohledu na data na úrovni prezentace obsahu datového skladu.
- Slicing – dovoluje provádět řezy datovou kostkou, tedy nalézt pohled, v němž je jedna dimenze pevně zafixována v jistých instancích jisté agregační úrovně. Jinými slovy tato dimenze aplikuje filtr na instance příslušné agregační úrovně dané dimenze.
- Dicing – je obdobou „slicingu“, jenž umožňuje nastavit takový filtr pro více dimenzí.

Dolování dat

Nástroje dolování dat slouží pro netriviální dobývání skrytých, předem neznámých a potenciálně užitečných informací z dat a využívají přitom četných matematických a statistických technik. Hlavním rozdílem mezi dolováním dat a OLAP analýzou je, že zatímco OLAP zkoumá vztahy známé a dobře strukturované a pracuje nad agregovanými daty (dimenze a ukazatele jsou pevně svázány), dolování dat pracuje zpravidla nad neagregovanými daty datového skladu a jeho cílem je nacházet nové skutečnosti a vztahy ve zkoumaných datech. [37]

Dolování dat může být využito např. pro detekci podvodů (daňové úřady, pojišťovny, banky), analýzy nákupních košíků, segmentace zákazníků, udržení zákazníků, stanovení diagnózy (lékařství), analýzu chování návštěvníků webových stránek a mnohé další.

Data mining zpravidla pracuje se strukturovanými daty, ale existují specifické druhy dolování převážně nestrukturovaných dat.

Typy převážně nestrukturovaných dat:

• Text mining – jde o dolování informací z textových nestrukturovaných dat.

• Web mining – jde o dolování informací z dat pocházejících z webových stránek

Reporting

Reporting je činnost spojená se získáváním dat z datových úložišť a jejich zobrazováním uživatelům. Reporting lze podle jeho charakteru rozdělit na:

- standardní – jde o zpravidla periodické generování výkazů, které mají stále stejnou strukturou.
- ad hoc – specifický jednorázový výkaz vytvářený na základě aktuálních potřeb uživatele

Speciálním případem reportingu patřícím do oblasti BI, je reporting nad OLAP kostkou. Nejčastější výstupy OLAP reportingu mají podobu:

- kontingenční tabulka
- kontingenční graf
- dashboard

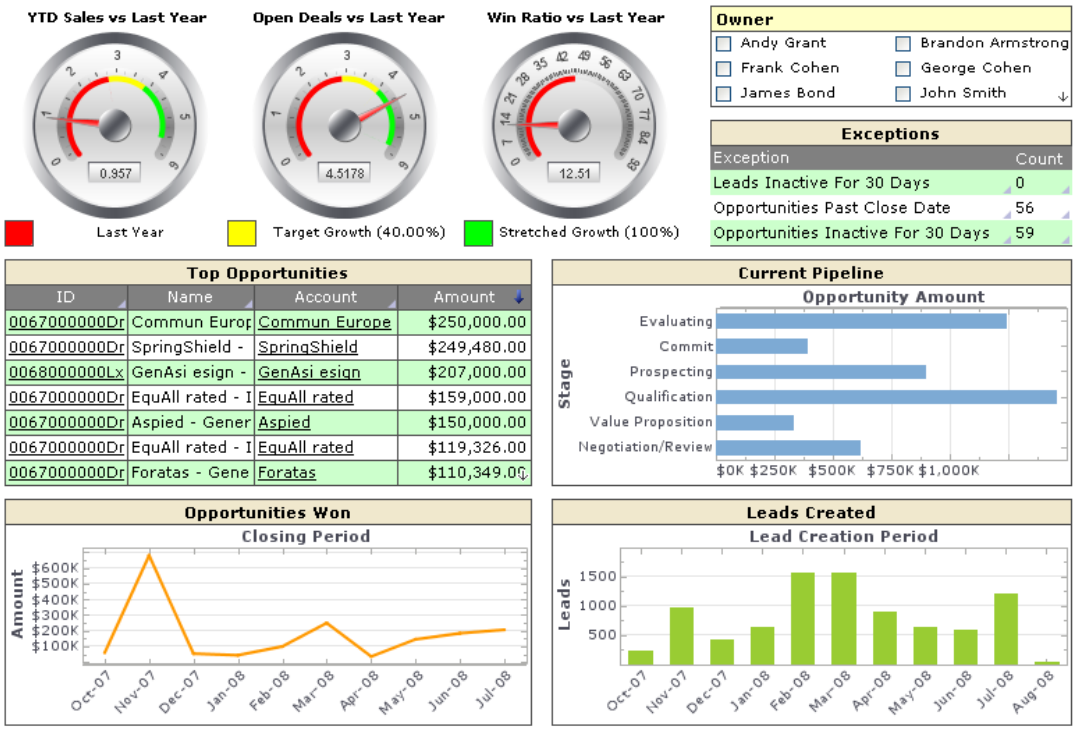
5.2.5 Nástroje pro koncové uživatele

Stále častěji se do BI a reportingových nástrojů integrují prezentační funkce, které tvoří další vrstvu nad kontingenčními tabulkami a grafy. Jde o nejrůznější přehledy, dashboardy (*Obrázek 20*), manažerské kokpity, tabulkové procesory (*Obrázek 21*) a další.

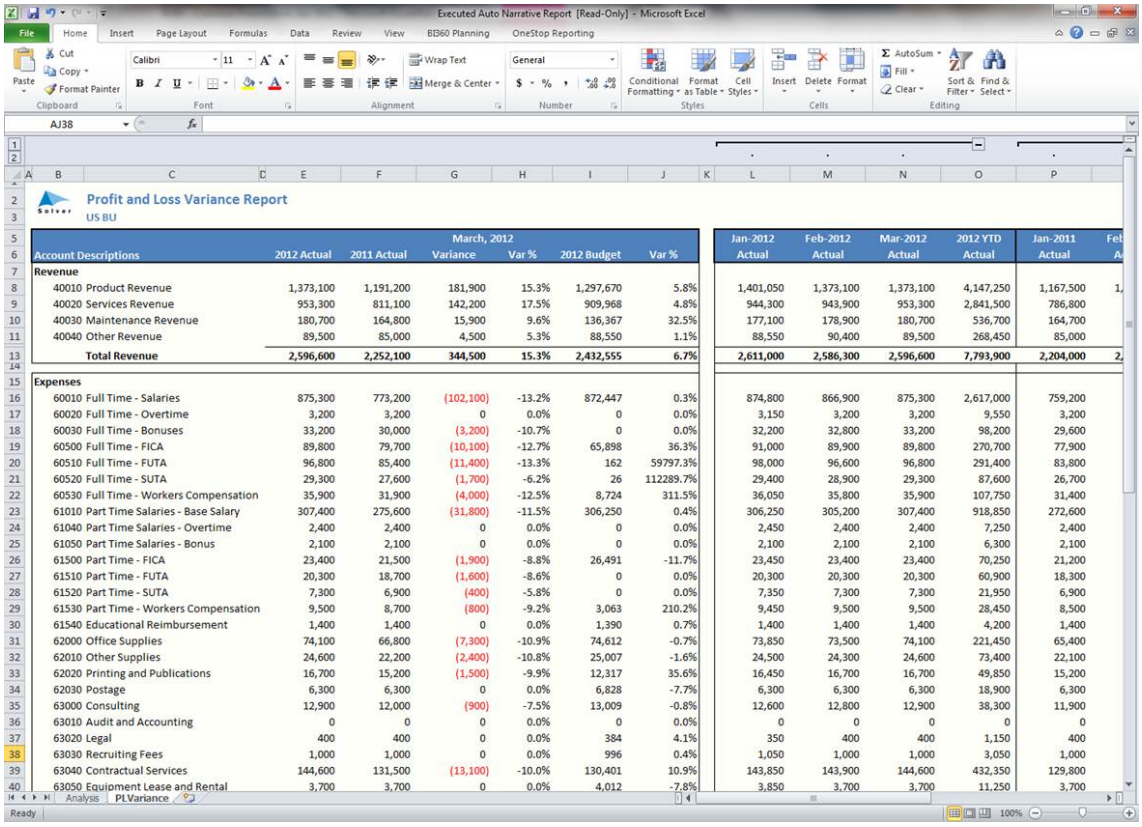
Významnými funkcemi report-nástrojů jsou:

- Pravidelné vytváření a zasílání specifických reportů zaměstnancům, kteří je potřebují pro své rozhodování.
- Zasílání výstrahy, např. na email nebo mobilní telefon, v případě, že se určitý ukazatel nevyvíjí podle plánu, resp. je pod hranicí tolerance.
- Zobrazení analýz a ukazatelů prostřednictvím manažerského kokpitu, pomocí několika obrazovek s grafy, tabulkami a barevnými indikacemi podle toho, zda je ve sledované oblasti dodržen plán.

Na základě informací o BI, které tato kapitola obsahuje, bude v následující kapitole popsán postup, jakým byl vytvořen datový sklad pro analýzu telefonních hovorů hovorové fronty.



Obrázek 20: Příklad reportu v podobě Dashboard



Profit and Loss Variance Report
US BU

Account Descriptions	March, 2012 2012 Actual	2011 Actual	Variance	Var %	2012 Budget	Var %	Jan-2012 Actual	Feb-2012 Actual	Mar-2012 Actual	2012 YTD Actual	Jan-2011 Actual
Revenue											
40010 Product Revenue	1,373,100	1,191,200	181,900	15.3%	1,297,670	5.8%	1,401,050	1,373,100	1,373,100	4,147,250	1,167,500
40020 Services Revenue	953,300	811,100	142,200	17.5%	909,968	4.8%	944,300	943,900	953,300	2,841,500	786,800
40030 Maintenance Revenue	180,700	164,800	15,900	9.6%	136,367	32.5%	177,100	178,900	180,700	536,700	164,700
40040 Other Revenue	89,500	85,000	4,500	5.3%	88,550	1.1%	88,550	90,400	89,500	268,450	85,000
Total Revenue	**2,596,600**	**2,252,100**	**344,500**	**15.3%**	**2,432,555**	**6.7%**	**2,611,000**	**2,586,300**	**2,596,600**	**7,793,900**	**2,204,000**
Expenses											
60010 Full Time - Salaries	875,300	773,200	(102,100)	-13.2%	872,447	0.3%	874,800	866,900	875,300	2,617,000	759,200
60020 Full Time - Overtime	3,200	3,200	0	0.0%	0	0.0%	3,150	3,200	3,200	9,550	3,200
60030 Full Time - Bonuses	33,200	30,000	(3,200)	-10.7%	0	0.0%	32,200	32,800	33,200	98,200	29,600
60500 Full Time - FICA	89,800	79,700	(10,100)	-12.7%	65,898	36.3%	91,000	89,900	89,800	270,700	77,900
60510 Full Time - FUTA	96,800	85,400	(11,400)	-13.3%	162	59797.3%	98,000	96,600	96,800	291,400	83,800
60520 Full Time - SUTA	29,300	27,600	(1,700)	-6.2%	26	112289.7%	29,400	28,900	29,300	87,600	26,700
60530 Full Time - Workers Compensation	35,900	31,900	(4,000)	-12.5%	8,724	311.5%	36,050	35,800	35,900	107,750	31,400
61010 Part Time Salaries - Base Salary	307,400	275,600	(31,800)	-11.5%	306,250	0.4%	306,250	305,200	307,400	918,850	272,600
61040 Part Time Salaries - Overtime	2,400	2,400	0	0.0%	0	0.0%	2,450	2,400	2,400	7,250	2,400
61050 Part Time Salaries - Bonus	2,100	2,100	0	0.0%	0	0.0%	2,100	2,100	2,100	6,300	2,100
61500 Part Time - FICA	23,400	21,500	(1,900)	-8.8%	26,491	-11.7%	23,450	23,400	23,400	70,250	21,200
61510 Part Time - FUTA	20,300	18,700	(1,600)	-8.6%	0	0.0%	20,300	20,300	20,300	60,900	18,300
61520 Part Time - SUTA	7,300	6,900	(400)	-5.8%	0	0.0%	7,350	7,300	7,300	21,950	6,900
61530 Part Time - Workers Compensation	9,500	8,700	(800)	-9.2%	3,063	210.2%	9,450	9,500	9,500	28,450	8,500
61540 Educational Reimbursement	1,400	1,400	0	0.0%	1,390	0.7%	1,400	1,400	1,400	4,200	1,400
62000 Office Supplies	74,100	66,800	(7,300)	-10.9%	74,612	-0.7%	73,850	73,500	74,100	221,450	65,400
62010 Other Supplies	24,600	22,200	(2,400)	-10.8%	25,007	-1.6%	24,500	24,300	24,600	73,400	22,100
62020 Printing and Publications	16,700	15,200	(1,500)	-9.9%	12,317	35.6%	16,450	16,700	16,700	49,850	15,200
62030 Postage	6,300	6,300	0	0.0%	6,828	-7.7%	6,300	6,300	6,300	18,900	6,300
63000 Consulting	12,900	12,000	(900)	-7.5%	13,009	-0.8%	12,600	12,800	12,900	38,300	11,900
63010 Audit and Accounting	0	0	0	0.0%	0	0.0%	0	0	0	0	0
63020 Legal	400	400	0	0.0%	384	4.1%	350	400	400	1,150	400
63030 Recruiting Fees	1,000	1,000	0	0.0%	996	0.4%	1,050	1,000	1,000	3,050	1,000
63040 Contractual Services	144,600	131,500	(13,100)	-10.0%	130,401	10.9%	143,850	143,900	144,600	432,350	129,800
63050 Equipment Lease and Rental	3,700	3,700	0	0.0%	4,012	-7.8%	3,850	3,700	3,700	11,250	3,700

Obrázek 21: Report v podobě tabulky Excel

5.3 Vytváření datového skladu

Teorie datových skladů, popsaná v předchozí kapitole, rozebírá, jakým způsobem a jakými náležitostmi by měl být požadovaný datový sklad vytvořen.

Hlavní požadavek datových skladu je, aby dokázal z nějakého pohledu analyzovat hovory hovorové fronty. Vytvořená průběžná relační databáze je schopna uchovávat události s AMI protokolu, které souvisí s průběhem těchto hovorů. Bylo potřeba vytvořit takový model, aby dokázal uchovávat potřebná data pro analýzu a následné vyhodnocení (reporting). S tím zajisté souvisí i implementace programu, který provede ETL proces převodu dat z relační databáze do datového skladu, samozřejmě, po výběru vhodného RDBMS databázového systému.

5.3.1 Vytváření modelu datového skladu

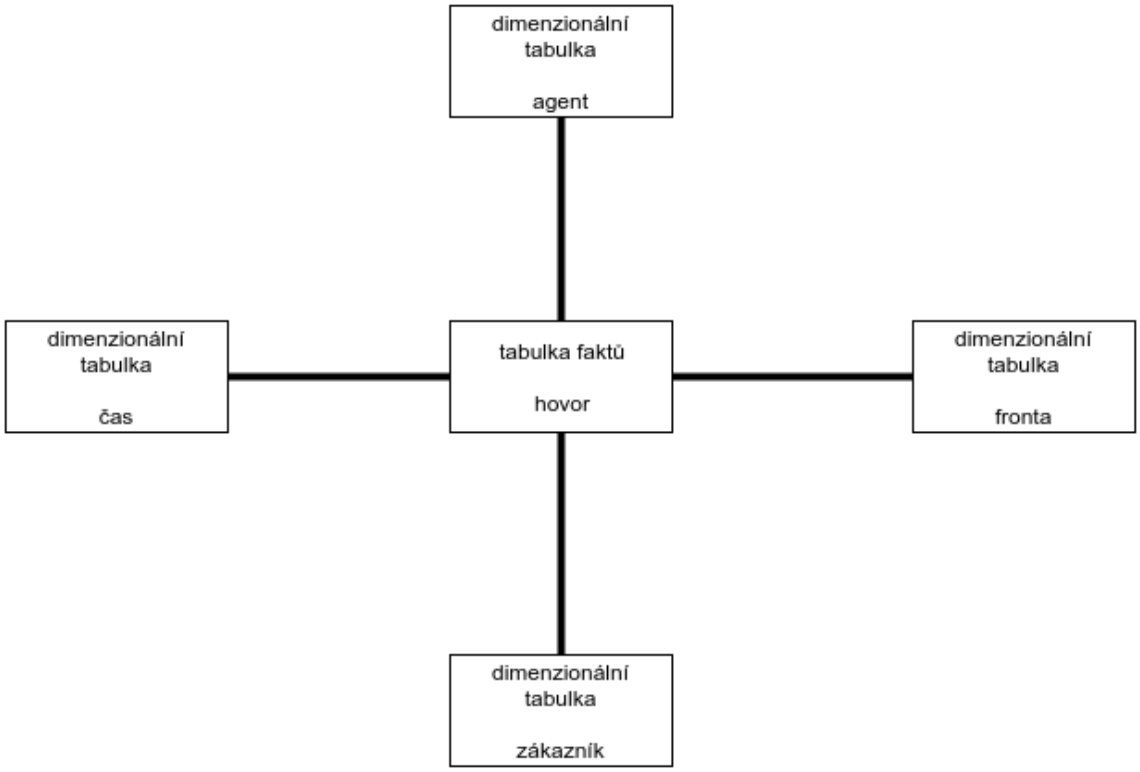
Pro sledování hovorů hovorové fronty bylo vhodné vytvořit schéma, které je standardem pro datové sklady. Toto schéma by mohlo nabývat typu „hvězda“ nebo z hlediska hierarchie dimenzí typu „sněhová vločka“.

Tabulka faktů zprostředkovává informace o těchto hovorech. Dimenze schématu, na základě informací o IP PBX, budou obsahovat čas, volajícího, agenta a hovorovou frontu. Na základě těchto informací bude schéma typu „hvězda“ (*Obrázek 22*).

Časové okamžiky hovoru

Každý hovor naplňují tří okamžiky (ve formě událostí)

1. Připojení volajícího – Okamžik, kdy volající vytočí číslo ústředny a je přiřazen do fronty, kde bude čekat na přepojení na agenta.
2. Začátek hovoru – Okamžik, kdy agent zvedne telefon a začne komunikovat s volajícím.
3. Konec hovoru – Okamžik, kdy hovor končí zavěšením telefonu jednou ze stran účastníků hovoru, popřípadě chybou přenosu.



Obrázek 22: Schéma datového skladu

Tabulka faktů

Na základě informací z kapitoly *3 Analýza Asterisk Manager Interface protokolu*, lze do tabulky faktů zahrnout následující atributy:

- délka hovoru
- délka vyzvánění
- délka čekání zákazníka ve frontě na přepojení agenta
- informace, jestli byl hovor připojen
- kým byl hovor ukončen (agent, zákazník, chyba přenosu)
- cizí klíče tabulky čas
  - pro připojení volajícího
  - pro začátek hovoru
  - pro konec hovoru
- cizí klíč tabulky agent

- cizí klíč tabulky fronta
- cizí klíč tabulky zákazník

Dimenze čas

Časová značka každé události je zachytávána a ukládána do relační databáze. Pro výpočet trvání délky hovoru či délky vyzvánění nebo vytěžení tohoto atributu z události je potřeba získat čas události připojení zákazníka, události začátku hovoru a události konce hovoru. Do této dimenze je proto potřeba přidat další tabulku jako užší úroveň hierarchie. Tato tabulka bude uchovávat typ časové značky a její identifikátor, jako primární klíč, bude v tabulce časové značky provázán skrz cizí klíč tohoto atributu.

Dimenze agent

Dimenze agent bude obsahovat názvy agentů dané hovorové fronty. Primární klíč tabulky bude číselný identifikátor na bázi autoinkrementální funkce a bude cizím klíčem v tabulce faktů.

Dimenze fronta

Tato dimenze bude naplňovat název existující hovorové fronty. Jako u ostatních dimenzí, identifikátor jako primární klíč bude cizím klíčem tabulky faktů.

Dimenze zákazníka

Dimenze bude naplňovat telefonní čísla připojených zákazníků, kteří nejen uskutečnili ale i neuskutečnili (například v případě ukončení hovoru zákazníkem z důvodu dlouhého čekání na připojení agenta) telefonní hovory. Tabulka bude také obsahovat atribut, jehož hodnoty budou určovat region volajícího zákazníka. Aby se dalo takto určit region zákazníka, předpoklad musí být telefonní předvolba pevné linky určitého regionu v ČR. V ostatních případech (mobilní a jiné telefony, u kterých nelze jednoduše určit polohu zákazníka) se bude jevit hodnota atributu jako „region neznámý“. Ideální řešení polohy či jiných informací o zákaznících by mohlo určit napojení na systém pro řízení vztahu se zákazníky (CRM), který by informace o například registrovaných zákaznících v nějaké formě uchovával.

5.3.2 Implementace modelu datového skladu

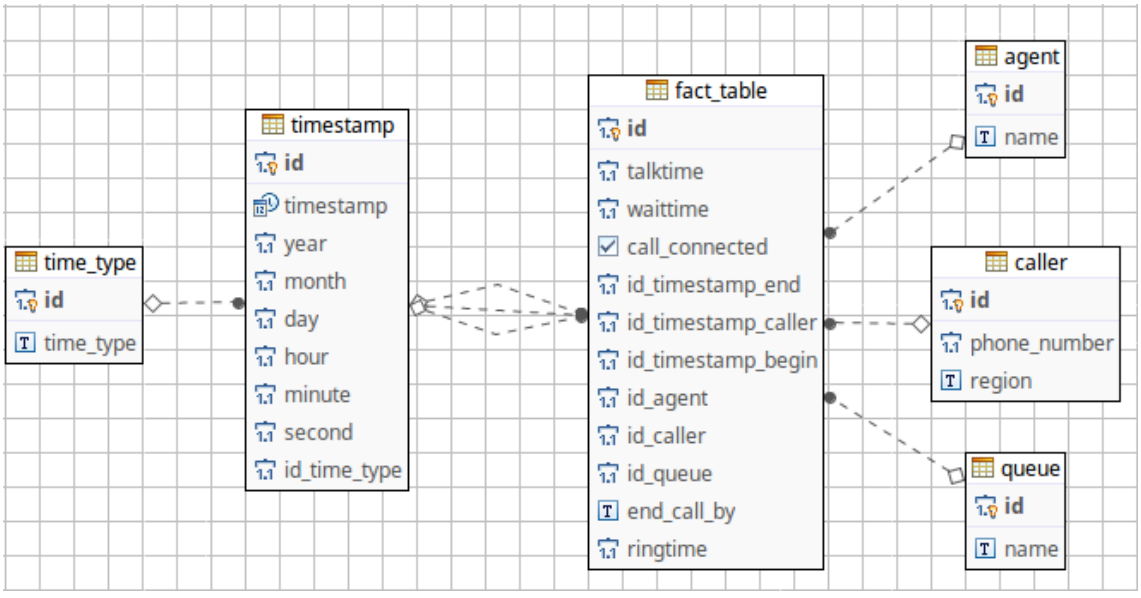
Z důvodu implementace relační databáze v RDBMS PostreSQL byl pro implementaci datového skladu použit tento systém s PostgreSQL ve stejné verzi 9.4.6. PostgreSQL

byl nainstalován na operačním systému Linux Mint ve verzi 17.3. Jako vývojové grafické prostředí jsem použil aplikaci SQL Workbench/j (ve verzi 119), která pro komunikaci s PostgreSQL využívá Java rozhraní JDBC (verze 9.4.1208). Použité programy, jejich verze a data vydání jsou přehledně vypsány v tabulce *Tabulka 4*.

Název	Verze	Datum vydání
Linux Mint	17.3	07.12.2015
PostgreSQL	9.4.6	11.02.2016
SQL Workbench/j	Build 119	31.01.2015
JDBC	9.4.1208	23.12.2015

Tabulka 4: Verze software pro implementaci modelu datového skladu

Na základě kapitoly *5.3.1 Vytváření modelu datového skladu* byl implementován model v podobě, kterou značí obrázek *Obrázek 23*. Tento model je typu „hvězda“, ale z hlediska eliminace nadbytečností v podobě typu časové značky, byla tato dimenze rozložena na relaci 1:N tabulkou, která uchovává názvy časových značek, které se používají při identifikaci hovoru.



Obrázek 23: Model datového skladu pro analýzu dat

Ukázka SQL kódu pro vytvoření dimenzionální tabulky agentů:

```
CREATE TABLE datawarehouse.caller
(
   id            bigserial      NOT NULL,
   phone_number  integer        NOT NULL,
   region        varchar(100)
);

ALTER TABLE datawarehouse.caller
   ADD CONSTRAINT caller_pk
   PRIMARY KEY (id);
```

5.3.3 Implementace ETL procesu

Kapitola *5.2.2 Komponenty datové transformace* definuje proces ETL z hlediska teorie datových skladů.

Dle analýzy a implementace modelu v předchozích kapitolách se dá shrnout ETL proces do následujících bodů:

- **Extrakce** - v tomto případě půjde o příkaz *SELECT* z jiného schématu
- **Transformace** - převedení produkčních dat do formátu vhodného pro data datového skladu
- **Nahrání** – nahrání dat do datového skladu

Tento postup bude aplikován pro naplnění dimenzí a ve finále faktové tabulky.

Plnění časové dimenze

Dimenze se v průběhu času mění přidáváním dat z produkční databáze (na inkrementální úrovni).

Nejprve je potřeba stanovit sled událostí AMI protokolu, ze kterých budou čerpány časové složky proběhlého hovoru.

Časová značka připojení zákazníka

Analýzou událostí AMI protokolu při uskutečnění hovoru v hovorové frontě byla zjištěna událost, která značí připojení účastníka (zákazníka) k hovorové frontě. Jedná se o událost *join.*

Pro uložení tohoto atributu stačí vybrat časové značky (atribut *timestamp*) události (podle identifikátoru atributu) z produkční databáze tabulky *time_of_events* a uložit je jako nové záznamy do této dimenze datového skladu.

Časová značka začátku hovoru

Začátek hovoru v hovorové frontě znamená připojení zákazníka k volnému agentovi. Pokud agent zvedne telefon, vyvolá se událost *agentconnect*. Stejným postupem jako v případě připojení zákazníka bude aplikováno uložení těchto časových značek pro začátek hovoru (s jiným atributem pro danou událost).

Časová značka konce hovoru

Událost *agentcomplete* označuje ukončení hovoru. Tato událost se vyvolá, pokud ukončí událost agent, zákazník či dojde k neočekávanému přerušení hovoru. Toto neočekávané ukončení hovoru zajišťuje chybějící informace o datech, které by v případě nuceného vyžádání ukončení od jedné strany hovoru mohlo nastat. Uložení těchto dat v tabulce dimenzí je opět stejné jako v případě předchozích dvou časových značek.

Naplnění užší hierarchie této dimenze, která značí typ časové značky obstará příkaz ke vložení předem známých údajů:

```
INSERT INTO datawarehouse.time_type (time_type) values
('caller_connect'),('call_begin'), ('call_end');
```

Naplnění číselného identifikátoru, označujícího číslo časové značky proběhne sekvencí, která obstará automatické číslování:

```
CREATE SEQUENCE date_time_id_seq
       INCREMENT BY 1
       MINVALUE 1
       CACHE 1
       NO CYCLE;
```

Naplnění atributu *timestamp* obstará následují příkaz pro výběr dat z produkční databáze:

```
INSERT INTO datawarehouse.timestamp (timestamp,id_time_type)
SELECT t.timestamp, d.id
FROM time_of_events t
JOIN events e ON e.id=t.id_event, datawarehouse.time_type d
WHERE ((e.name='join' AND d.time_type='caller_connect') OR
(e.name='agentconnect' AND d.time_type='call_begin') OR
(e.name='agentcomplete' AND d.time_type='call_end'))
AND timestamp NOT IN (SELECT dt2.timestamp FROM
datawarehouse.timestamp dt2);
```

Rozložení časové značky

Naplnění dat pro rozložení časové značky na atributy rok, měsíc, den, hodina, minuta a sekunda obstará činnost, která se provede v případě definované události nad

databázovou tabulkou. Tato činnost, známá v RDBMS systémech jako *trigger*, proběhne po vložení záznamů časové značky.

Vytvoření *triggeru*:

```
CREATE TRIGGER tr_separatetime AFTER INSERT ON datawarehouse.timestamp
 FOR EACH ROW EXECUTE PROCEDURE separate_time();
```

Vytvoření procedury *triggeru* v jazyce *plpg/sql*:

```
CREATE OR REPLACE FUNCTION separate_time() RETURNS trigger AS $return$
DECLARE
    cas timestamp;
    rok int2;
    mesic int2;
    den int2;
    hodina int2;
    minuta int2;
    sekunda int2;
BEGIN
SELECT INTO cas new.timestamp;
SELECT INTO rok EXTRACT(YEAR FROM cas);
SELECT INTO mesic EXTRACT(month FROM cas);
SELECT INTO den EXTRACT(day FROM cas);
SELECT INTO hodina EXTRACT(hour FROM cas);
SELECT INTO minuta EXTRACT(minute FROM cas);
SELECT INTO sekunda FLOOR(EXTRACT(second FROM cas));
UPDATE datawarehouse.timestamp SET year = rok, month = mesic, day =
den, hour = hodina, minute = minuta, second = sekunda WHERE timestamp
= cas;
RETURN new;
END;
$return$ LANGUAGE plpgsql;
```

Pro kontrolu duplicity údajů při inkrementálním dávkovém vkládání údajů byla použita alternativa k funkci *merge*, která je implementovaná v PostgreSQL až ve verzi 9.5 a nebyla při vytváření datového skladu dostupná. Byla tedy použita kontrola, zda nově vkládané záznamy již v tabulce existují.

Plnění dimenze agentů

Název agentů hovorové fronty, kteří se účastní hovorů, poskytuje událost *queuememberstatus*, tedy její atribut *membername*.

Pro uložení vybraných agentů z produkční databáze sloučí následující příkaz:

```
INSERT INTO datawarehouse.agent (name)
SELECT DISTINCT v.value
FROM (values v
JOIN time_of_events t ON t.id = v.id_time_of_events
JOIN attributes a ON a.id = v.id_attribute)
JOIN events e ON e.id=t.id_event
WHERE a.name='membername' AND e.name='queuememberstatus'
AND v.value NOT IN (SELECT a.name FROM datawarehouse.agent a);
```

Formát těchto dat z produkční databáze bude shodný s formátem v dimenzi datového skladu, není třeba jej transformovat.

Automatický přírůstek číslování zajistí, jako v případě plnění časové dimenze, založení sekvence.

Inkrementální přírůstek a zamezení duplicit zajistí kontrola vkládaného záznamu, tedy kontrola, jestli se stejný agent už v tabulce dimenze nenachází.

Plnění dimenze zákazníků

Telefonní čísla zákazníků zprostředkovává atribut *calleridnum* události *join* uložený v produkční databázi. V případě uložení vybraných zákazníků poslouží podobný příkaz jako v případě ukládání agentů, pouze s jiným výběrem atributu události a události samotné.

Jelikož jsou v produkční databázi ukládány hodnoty atributů událostí jako řetězce, je potřeba při vkládání tuto hodnotu transformovat na celé číslo.

Následující SQL příkaz zajistí toto přetypování:

```
CAST(v.value as integer)…
```

Sekvence nám opět zajistí automatické číslování.

Zjišťování regionu zákazníka

Jak již bylo uvedeno, tabulka dimenze uchovává region volajícího zákazníka. Název regionu byl zvolen dle krajů ČR a je identifikován pomocí předvolby telefonního čísla.

Naplnění tohoto názvu regionu proběhne triggerem, jež se spustí po vložení telefonních čísel zákazníků.

Procedura po spuštění triggeru porovná počáteční čísla hodnoty telefonního záznamu s oficiální předvolbou pro ČR a uloží záznamy názvů regionu.

Ukázka části kódu spuštěné procedury triggerem po vložení záznamů:

```
SET region =  CASE
WHEN CAST(ph AS TEXT) LIKE '37%' THEN 'Plzensky'
```

Plnění dimenze fronty

Událost AMI protokolu *queuememberstatus* zobrazuje v pravidelném opakování či změně stav agenta ve frontě. Jeden z jejího atributu také značí v jaké frontě se agent nachází. Proto výběr názvu hovorové fronty poslouží událost *queuememberstatus* a její atribut *queue*.

Automatické číslování bylo zajištěno opět sekvencí a transformace nebyla z důvodu stejného datového typu nutná.

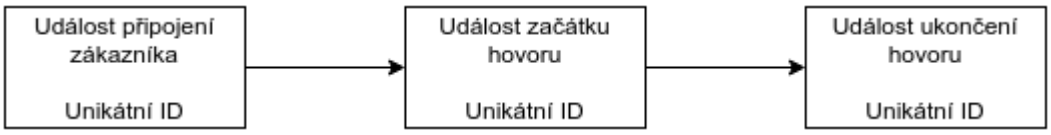
Plnění tabulky faktů

Tabulka faktů čerpá informace současně z jednotlivých dimenzí a produkční databáze. Produkční databáze uchovává všechny události a jejich atributy, ale žádným způsobem je nejde jednoduše propojit tak, aby se dal identifikovat jednotlivý hovor při jasné identifikaci účastníků.

Hlavním problémem tedy při plnění tabulky faktů bylo vyřešit, jakým způsobem propojit časové značky hovoru.

Při zkoumání sledu událostí bylo zjištěno, že obsahují stejný atribut *uniqueid*.

Tento atribut se přiřadí při připojení zákazníka do fronty a vyvolává se i v událostech začátku a konce hovoru. Tímto atributem se tedy dají propojit tři časové značky hovoru a v tomto propojení značí právě jeden takový proběhlý hovor. Výhodou je, že hodnota tohoto atributu je unikátní a již se v jiných hovorech, i v případě dalšího volání stejným zákazníkem, neopakuje. Unikátní identifikátor dokáže propojit tyto události a z toho samozřejmě vyplývá i možnost výběru potřebných atributů (např. kým byl hovor ukončen), které události obsahují, a jejich uložení do tabulky faktů. Digram sledu událostí hovoru znázorněn na obrázku *Obrázek 24*.



Obrázek 24: Diagram sledu událostí hovoru

Při plnění tabulky musíme nahradit klíče z produkční databáze odkazy do dimenzí datového skladu. Pro naplnění cizích klíčů časových značek (*id_timestamp_caller*) hovorů bylo potřeba nejprve provést výběr těchto dat podle typu časové značky (cizí klíč *id_time_type* dimenzionální tabulky *timestamp*), tedy její hodnoty *'caller_connect'* v hierarchii stejné dimenze.

Naplnění proběhne příkazem:

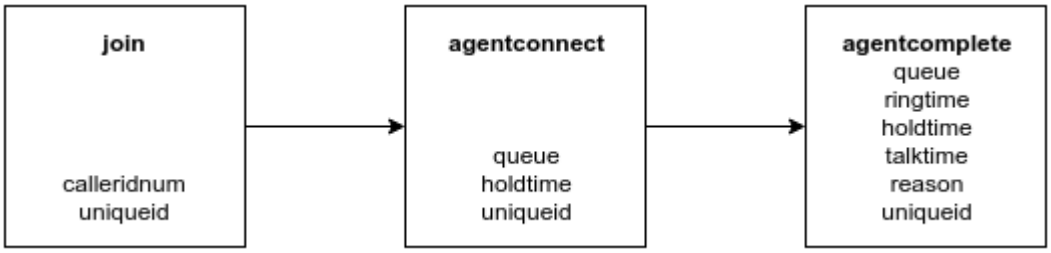
```
INSERT INTO datawarehouse.fact_table(id_timestamp_caller)
SELECT dt.id
FROM datawarehouse.timestamp dt
JOIN datawarehouse.time_type dtt ON dtt.id=dt.id_time_type
WHERE dtt.time_type='caller_connect' AND
dt.id NOT IN (SELECT df2.id_timestamp_caller FROM
datawarehouse.fact_table df2);
```

Po naplnění tohoto identifikátoru se vyvolá trigger, jehož spuštěná procedura vyplní zbytek údajů pro každý řádek v tabulce takto:

1. Podle uloženého klíče, identifikující časovou značku uloží do dočasné proměnné unikátní identifikátor.
2. Podle hodnoty identifikátoru najde zbylé dvě události (pro identifikaci hovoru) a jejich cizí klíče značící časové značky dočasně uloží do proměnných. Již je tedy identifikovaný jeden hovor.
3. Podle události o připojení zákazníka (*join*) porovná hodnotu jejího atributu pro telefonní číslo zákazníka s hodnotou atributu v tabulce dimenze zákazníka a tuto hodnotu uloží do dočasné proměnné.
4. Dle předchozího bodu udělá stejnou operaci pro identifikaci agenta hovoru a hovorové fronty. Nyní jsou již všechny cizí klíče dimenzí připraveny k naplnění.
5. Z události konce hovoru se vyberou hodnoty atributů pro délku hovoru (*talktime*), délku čekání zákazníka ve frontě (*holdtime*), délku vyzvánění agenta (*ringtime*) a hodnota atributu pro označení, kým byl hovor ukončen (*reason*). Také se uloží do dočasných proměnných a navíc se délka hovoru a délka vyzvánění transformují na vhodný datový typ. Kvůli kontrole hodnot pro délku hovoru a vyzvánění mohou být tyto hodnoty případně přepočítány z časových značek událostí hovoru.
6. Poslední atribut tabulky faktů, který je potřeba vyplnit, je označení, jestli byl

hovor uskutečněn. Při neexistenci události o začátku hovoru je nastavena hodnota datového typu atributu na *FALSE*, v opačném případě na hodnotu *TRUE*.

7. Proběhne aktualizace řádku tabulky, tedy vložení nabytých hodnot z dočasných proměnných.



Obrázek 25: Diagram propojení událostí a jejich atributů v případě jednoho hovoru

Obrázek 25 znázorňuje propojení událostí a jejich atributů, které byly použity při vytváření procedury naplňování datového skladu.

Podobně jako v případě dimenzí potřebujeme sekvenci pro plnění identifikátoru.

Ukázka části SQL kódu procedury, která proběhne při spuštění triggeru:

```
SELECT INTO id_tcaller new.id_timestamp_caller ;
SELECT value INTO uniqueid
FROM ((time_of_events toe
JOIN datawarehouse.timestamp dt ON toe.timestamp=dt.timestamp
JOIN values v ON toe.id=v.id_time_of_events)
JOIN attributes a ON a.id=v.id_attribute)
JOIN datawarehouse.time_type dtt ON dtt.id=dt.id_time_type
WHERE dt.id=id_tcaller AND dtt.time_type='caller_connect' AND
a.name='uniqueid';
```

6. Ověření výsledků a jejich vizualizace

6.1 Vytváření simulovaného provozu

K ověření správnosti implementovaných programů bylo potřeba vytvořit virtuální provoz, který bude simulovat denní rutinu telefonní ústředny. Pro tyto účely je nutné simulovat provoz v hovorové frontě a tím vygenerovat nějaké netriviální množství dat.

Existují aplikace, které dokáží spojit telefonní hovor v rámci jednoho zařízení a virtuálního stroje jako serveru telefonní ústředny. V kapitole *3.4.2 Uskutečnění hovoru a sledování událostí* je uveden příklad uskutečnění takového hovoru přes registraci SIP linek zákazníka a agenta aplikací *Linphone*, ale pro naplnění produkční databáze netriviálním množství dat to není ideální řešení.

Ideálním řešení se předpokládá několikahodinový provoz s určitým počtem agentů v hovorové frontě a opakované uskutečňování hovorů (pseudonáhodně) zákazníky.

6.1.1 PJSIP

PJSIP je open source multimediální komunikační knihovna napsaná v jazyce C, která implementuje standardy protokolů jako např. SIP. Spojuje v sobě tento signalizační protokol s bohatým multimediálním frameworkem a funkčností NAT do úrovně API, které je vhodné pro téměř všechny typy systémů od stolních počítačů, vestavěných systémů, po mobilní telefony. [38]

PJSUA

PJSUA je SIP uživatelský agent (softwarový telefon), který se používá jako referenční implementace pro knihovnu PJSIP. [38] Používá se příkazy v příkazovém řádku a umožňuje, mimo jiné, registrovat telefonní linky přes SIP a uskutečňovat hovory s nastavitelnými parametry.

Provedení testovacího hovoru

Aby bylo možné vyzkoušet funkčnost PJSUA, bylo potřeba provést testovací hovor, podobný jako v ukázce v kapitole *3.4.2 Uskutečnění hovoru a sledování událostí*.

Předpokládejme tedy funkční virtuální server telefonní ústředny Kerio Operator, který je přístupný přes adresu *192.168.1.22.*

Registrace agenta: *sip:11@192.168.1.22.*

Registrace zákazníka: *sip:10@192.168.1.22.*

Hovorová fronta: *sip:01@192.168.1.22.*

Na operačním systému Linux Mint spustíme terminál, do kterého budeme zadávat příkazy, a vyhledáme složky se spouštěcím souborem PJSUA (ve správně zkompilovaném programu).

Následujícím příkazem registrujeme agenta telefonní ústředny:

```
$ ./pjsua-x86_64-unknown-linux-gnu --id="sip:11@192.168.1.22"
--registrar="sip:192.168.1.22" --realm="*" --username="11"
--password="11" --play-file=wav1.wav --auto-play --null-audio --local-
port=5050 --duration=30 --auto-answer=200
```

Parametry příkazu:

--id – cislo s adresou pro registraci agenta

--registrar – adresa serveru

--username, --password – jméno a heslo agenta registrovaného v Kerio Operátoru

--play-file, --auto-play – cesta zvukového souboru pro vyzvánění a cyklické opakování

--local-port – lokální port pro připojení k serveru

--duration – značí dobu hovoru v sekundách, kdy agent zavěsí

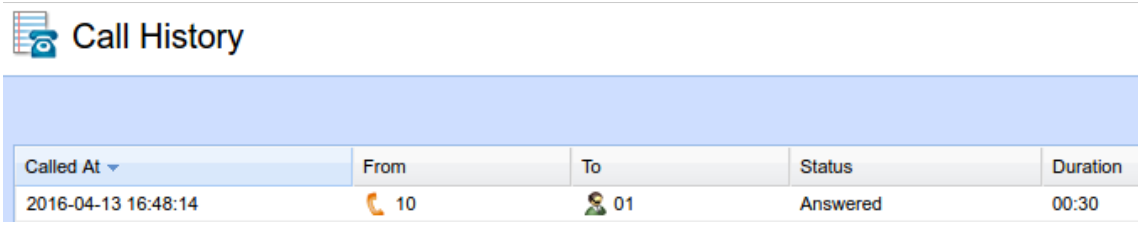
--auto-answer – značí dobu v milisekundách, kdy agent zvedne hovor

Spustíme jiný terminál s cestou k binárnímu souboru PJSUA a pro uskutečnění hovoru provedeme následující příkaz:

```
$ ./pjsua-x86_64-unknown-linux-gnu --id="sip:10@192.168.1.22"
--registrar="sip:192.168.1.22" --realm="*" --username="10"
--password="10" --local-port=5051 --auto-loop sip:01@192.168.1.22
--color --no-vad
```

Parametry příkazu se liší v přidání čísla a adresy hovorové fronty, kde chce zákazník hovor uskutečnit.

Záznam o hovoru nám zprostředkuje statistika v uživatelském rozhraní Kerio Operator, přístuptná přes webový prohlížec. Znázorněno na obrázku *Obrázek 26*.



Obrázek 26: Kontrola uskutečnění hovoru v administrátorské rozhranní Kerio Operator

Možnosti v simulovaném provozu

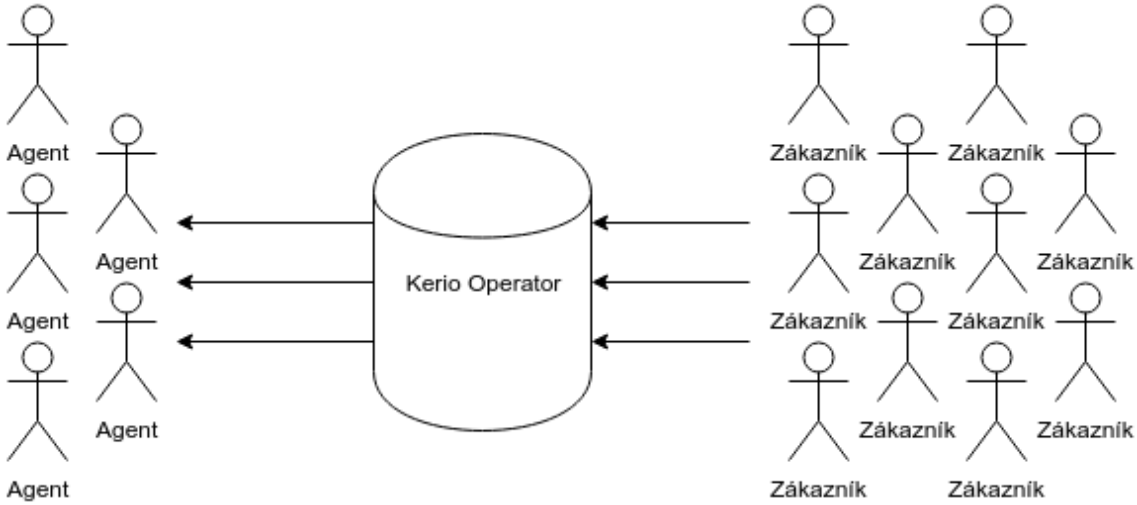
Jelikož PJSUA umožňuje zadávat jednoduše příkazy v příkazové řádce, je ideálním řešení pro vytvoření skriptu, který by tyto příkazy automatizoval a registroval tak jak účastníky hovoru, tak i prováděl telefonní hovory do hovorové fronty v pseudonáhodném běhu.

6.1.2 Vytváření skriptu pro simulovaný provoz

Na linuxové platformě je umožněno zadávat příkazy v příkazové řádce, interpretované tzv. Bash. Bash umožňuje také skriptování. Skriptování je zápis jednotlivých příkazů do souboru, které bychom jinak zadávali jednotlivě v příkazové řádce. Tento skript se pak spustí jedním příkazem.

Scénář provozu telefonní ústředny

Předpokládejme jednu telefonní ústřednu. Ústředna se bude skládat z až sedmi agentů, kteří budou v jedné hovorové frontě. Vytvářet hovory bude deset zákazníků, kteří budou opakovaně v pseudonáhodných intervalech po pseudonáhodně zvolených skupinách provádět pseudonáhodně dlouhé hovory do hovorové fronty. Pro analýzu sběru dat v hovorové frontě by měla být doba spuštění simulovaného provozu několik desítek hodin s tím, že budou vygenerovány stovky hovorů. Scénář simulace provozu je znázorněn na obrázku *Obrázek 27*.



Obrázek 27: Diagram simulace telefonního provozu

Implementace Bash skriptu

V Kerio Operator byly nejdříve vytvořeny telefonní linky pro 5-7 agentů (čísla 11-19) a 10 zákazníků (čísla 20-29).

Všechny linky agentů jsou na začátku skriptu registrovány. Samotné registrace agentů proběhli sériově v nových oknech terminálů.

Následující příkaz demonstruje registraci agenta ve skriptu:

```
xfce4-terminal -x ./pjsua-x86_64-unknown-linux-gnu --id="sip:11@$ip"
--registrar="sip:$ip" --realm="*" --username="11" --password="11"
--play-file=wav1.wav --auto-play --null-audio --auto-play --auto-
answer=200 --local-port=5070
```

Při případném použití jiné linuxové distribuce, která neobsahuje aplikaci terminálu *xfce4-terminal*, je nutné pro simulovaný provoz upravit název jiné terminálové aplikace.

Ve skriptu se cyklicky opakuje následující:

Vygeneruje se 10 pseudonáhodných čísel podle linek zákazníků (čísla v rozmezí 20-29), díle čísla, které značí dobu trvání hovorů (20-200 sekund) a číslo v rozpětí od 1 do 10, ze kterého se bude čerpat počet proběhlých hovorů.

Založí se 10 metod, které zprostředkují možný hovor do telefonní ústředny, který čerpá s náhodně vygenerovaných čísel doby trvání hovoru a po této době se hovor ukončí.

Příklad jedné této metody:

```
function call20 {
xfce4-terminal -x echo "quit" ;  sleep $randomduration1 | ./pjsua-
x86_64-unknown-linux-gnu --id="sip:20@$ip" --registrar="sip:$ip"
--realm="*" --username="20" --password="20" --play-file="wav1.wav"
 --auto-play --local-port=5081 --null-audio --auto-loop sip:01@$ip
--color --no-vad
}
```

Pak proběhne jeden hovor (jedna metoda) a podle pseudonáhodně vygenerovaného čísla (1-10) dále různý sled počtu a délky hovorů (2, 3, 5 nebo 9 hovorů). Jak bylo řečeno, hovory mají různou délku, proto agenti při vygenerovaní většího čísla mohou být při dalším vyvolání hovoru nedostupní a hovor neproběhne, což může poskytovat další užitečné informace při simulaci provozu telefonní ústředny.

Ukázka části kódu spouštění hovorů v podobě metod:

```
sleep $random4
if [ $(($randomcalling % 7)) -eq 0 ];
then call$random2 |& call$random3 |& call$random4
fi
```

Spuštění simulovaného provozu a ukládání dat

Před začátkem ukládání dat bylo zapotřebí spustit dříve vytvořenou Java aplikaci, která umožňuje ukládání dat do relační databáze. Poté proběhlo několikanásobné spuštění skriptu, které trvalo cca 24 hodin a bylo vygenerováno celkem přes 1200 hovorů. Nevýhoda vytvořené skriptu s použitím PJSUA je ta, že nelze nastavit náhodou dobu vyzvánění agenta, agent zvedá telefon ihned po jeho přiřazení (zavolání). Jako demonstrační ukázka funčnosti bylo provedeno ručně 10 hovorů s pomoci aplikace Linphone, kde různí zákazníci volají různé agenty, kteří čekají na zvednutí telefonu.

6.2 Vizualizace výsledků – výstupy z datového skladu

Vizualizace výsledků, tedy reporting v oblasti Bussiness Inteligence, je činnost spojená se získáváním dat z datových úložišť a jejich zobrazováním uživatelům. V případě této diplomové práce se jedná o standardní charakter reportingu, protože periodické generování výkazů má stejnou strukturu.

V kapitole *6.1 Vytváření simulovaného provozu* bylo popsáno, jakým způsobem byl vytvořen simulovaný provoz a jak proběhlo naplnění dat do datového skladu. Tyto data bylo potřeba nějakým způsobem zanalyzovat. Byly vytvořeny různé pohledy, které byly vizualizovány tabulkovým procesorem jako kontingenční tabulky a grafy.

6.2.1 Vytváření databázových pohledů

Pro čerpání z datového skladu byly vytvořeny databázové pohledy, které umožňují podle předem připravených prezentačních požadavků poskytnout uživateli snadný přehled dat na rozdíl od zdroje, kde jsou přímo uložena. Výhodou databázových pohledů, jakožto součást databázového systému, je také možnost čerpání z databáze v případě vytváření kontingenčních tabulek v tabulkovém procesoru. Pohledy jsou prosté výběry dat z datového skladu.

Obecný pohled

Tento pohled je základní prostý výběr dat z datového skladu, který udržuje strukturu faktové tabulky. Je tou souhrn informací o hovorech z tabulky faktů, která čerpá podle cizích klíčů z informací tabulek všech dimenzí.

Výběr dat, tedy příkaz SELECT jazyka SQL, byl implementován jako výběr všech atributů z faktové tabulky, které byly podle požadavku vhodně přejmenovány a spojením faktové tabulky s dimenzemi, kvůli identifikaci cizích klíčů.

Část příkazu pro výběr atributu a jeho vhodné přejmenování:

```
SELECT DISTINCT t.timestamp AS "cas hovoru",
```

Příklad spojení faktové tabulky s tabulkou dimenzí:

```
...
FROM
datawarehouse.fact_table ft
JOIN datawarehouse.timestamp t ON ft.id_timestamp_caller=t.id...
```

Obrázek 28 značí prvních pár výsledků spuštění SQL příkazu:

cas hovoru ▲	rok	mesic	den	hodina	minuta	sekunda	telefonni cislo	agent	fronta	doba hovoru	doba vyzvaneni	byl pripojen?	kym byl
2016-02-24 22:12:57	2016	2	24	22	12	57	29	sip/11	queue24	38	2	true	caller
2016-02-24 22:13:56	2016	2	24	22	13	56	23	sip/13	queue24	80	5	true	caller
2016-02-24 22:14:00	2016	2	24	22	14	0	25	sip/15	queue24	19	3	true	caller
2016-02-24 22:17:07	2016	2	24	22	17	7	26	sip/17	queue24	27	2	true	caller
2016-02-24 22:20:35	2016	2	24	22	20	35	27	sip/19	queue24	28	2	true	caller

Obrázek 28: Výsledek pohledu jako prostého výběru dat

Pohled výkonnosti agentů

Tento pohled značí, jak byli agenti výkonní během provozu hovorové fronty.

Pohled zprostředkovává:

- počet hovorů každého agenta

- poměr celkové doby hovorů na každého agenta fronty
- počet ukončených hovorů agentem
- počet ukončených hovorů zákazníkem
- průměrnou dobu hovoru každého agenta
- průměrnou dobu vyzvánění každého agenta
- poměry ukončených hovorů jednotlivých agentů

Může tedy analyzovat, který agent se drží předem nastavených pravidel (např. maximální délka hovoru), za jak dlouho agent zvedá telefon či jak často pokládá hovor při komunikaci se zákazníkem.

Z hlediska implementace bylo nutné sumarizovat výčet dat pro každého agenta.

Například pro celkovou dobu hovorů byl vytvořen atribut, ve kterém se sečetli všechny provedené hovory agenta a celkový počet se vydělil číslem *86400*, což je počet sekund za den, aby se v tabulkovém procesoru mohl vytvořit formát buněk tak, aby značil celkovou dobu hovoru ve tvaru *hh:mm:ss* . Následně po připojení dimenzí přes cizí klíče (jako v předchozím pohledu) musel být výsledek agregován podle názvu agentů a front, aby se dal vytvořit výsledek pro více sloupců. Výsledek příkazu *SELECT* tohoto pohledu je uveden na obrázku *Obrázek 29.*

Část kódu SQL pro vytvoření pohledu:

```
CREATE OR REPLACE VIEW swview4 AS
SELECT a.name as "agent", q.name "fronta",
(cast(sum(ft.talktime) as float8)/86400) AS "celkova doba hovoru",
...
FROM
...
GROUP BY a.name,q.name;
```

pocethovoru	nazev agenta	fronta	celkova doba	pocet	pocet	průměrná	průměrná doba	poměr	poměr
222	sip/11	queue24	0.15	4	217	1.69	59.15	0.02	0.98
83	sip/12	queue24	0.1	0	83	3.75	106.11	0	1
261	sip/13	queue24	0.18	0	260	3.52	61.04	0	1
84	sip/14	queue24	0.11	0	84	4.93	108.05	0	1
236	sip/15	queue24	0.17	0	234	2.31	62.72	0	1
131	sip/17	queue24	0.06	0	131	1.33	40.37	0	1
168	sip/19	queue24	0.09	0	167	1.25	47.87	0	1
21	neprobehle hovory	queue24		0	0			0	0

Obrázek 29: Výsledek pohledu výkonosti agenta

Pohled na délky hovorů

Různí agenti produkují různě dlouhé hovory. Tento pohled analyzuje celkový pohled na délky hovorů ve smyslu:

- počet hovorů v různých rozmezích délky hovoru
  - menší než třiceti sekundách
  - mezi třiceti sekundy a jednou minutou
  - delší než jednu minutu
- celková průměrná doba jednoho hovoru
- celková průměrná doba jednoho vyzvánění
- celková průměrná doba čekání zákazníka ve frontě

Vytváření těchto atributů ve výběru dat z datového skladu při zakládání pohledu proběhlo například v případě počtu hovorů menších než třicet sekund následovně:

```
CREATE OR REPLACE VIEW swview5 AS
SELECT SUM(CASE WHEN ft.talktime<30 THEN 1 ELSE 0 END) AS "pocet
hovoru kratsi nez 30 sekund",
FROM datawarehouse.fact_table ft;
```

Výsledek příkazu *SELECT* tohoto pohledu je uveden na obrázku *Obrázek 30*.

pocet hovoru kratsi nez 30 sekund	pocet mezi 30 - 60	pocet delsich nez 60	průměrná doba vyzvánění	průměrná doba hovoru
346	344	466	2.49	63.38

Obrázek 30: Výsledek pohledu pro délky hovorů

Pohled na počet hovorů v jednotlivých dnech

Tento pohled zprostředkovává počet hovorů v jednotlivých dnech, které byly připojeny na agenta a počet hovorů a které na agenta připojeny nebyly. Analyzuje tak denní souhrn, který značí vytíženost telefonní ústředny.

Pro snazší zobrazení v tabulkovém procesoru bylo vhodné změnit čísla měsíců na názvy při vytváření atributů výběru z datového skladu.

Počet těchto hovorů proběhl opět agregováním dat (jejich sumarizací) podle měsíců (dnu, let) následně:

```
CREATE OR REPLACE VIEW swview6 AS
SELECT t.day AS "den", t.year AS "rok",
SUM(CASE ft.call_connected WHEN TRUE THEN 1 ELSE 0 END) AS "pocet
pripojenych hovoru",
SUM(CASE ft.call_connected WHEN FALSE THEN 1 ELSE 0 END) AS "pocet
nepripojenych hovoru"
FROM datawarehouse.fact_table ft
JOIN datawarehouse.timestamp t ON ft.id_timestamp_caller=t.id
GROUP BY t.day, t.month,t.year;
```

Výsledek příkazu *SELECT* tohoto pohledu je uveden na obrázku *Obrázek 31*.

rok	mesic	den	pocet pripojenych hovoru	pocet nepripojenych
2016	unor	24	62	0
2016	unor	25	201	0
2016	brezen	2	71	0
2016	brezen	14	0	1
2016	brezen	18	124	3
2016	brezen	19	333	4
2016	brezen	27	1	1
2016	brezen	28	3	1
2016	duben	15	152	7
2016	duben	16	237	4
2016	duben	17	1	0

Obrázek 31: Pohled na počet hovorů v jednotlivých dnech

Pohled na postupný sled hovorů

Tento pohled byl vytvořen proto, aby bylo možné vytvořit v reportovacím nástroji histogram počtu hovorů v jednotlivých hodinách v jednotlivých dnech.

Byl vytvořen tak, že dokáže sumarizovat hovory při použité agregaci z hlediska jednotlivých let, měsíců a dnů.

Výsledek příkazu *SELECT* tohoto pohledu je uveden na obrázku *Obrázek 32*.

rok	mesic	den	hodina	minute	pocet hovoru
2016	4	15	23	34	1
2016	2	25	3	15	1
2016	2	25	4	1	1
2016	3	19	4	12	1
2016	4	16	2	45	1
2016	2	25	1	16	1
2016	3	19	8	9	1
2016	2	25	4	23	1
2016	2	24	23	24	1

Obrázek 32: Pohled na postupný sled hovorů

6.2.2 Použití reportovacího nástroje

Jako reportovací nástroj byl použit tabulkový procesor *LibreOffice Calc*, který je standardní součást programového vybavení linuxové distribuce Linux Mint. Tato alternativa aplikace *Excel*, jako součást kancelářského balíku *Microsoft Office*, umožňuje vytvářet kontingenční tabulky a grafy napojením přímo z relační databáze. Výběr lze do této aplikace zakomponovat a zobrazit v kontingenční tabulce. Výhodou napojení na relační databázi také je, že při inkrementálním přírůstku nových dat lze jednoduše tabulky aktualizovat a zobrazí se aktuální výsledky.

Reporty z jednotlivých pohledů

Všechny vytvořené pohledy byly importovány do tabulkového procesoru jako kontingenční tabulky. Při vytváření takových kontingenčních tabulek se lze napojit k DBMS a vybrat vhodný pohled. Při importu do kontingenční tabulky lze manipulovat se zobrazením atributů a vhodně je agregovat.

Report obecného pohledu

V obecném pohledu byly pro časové agregace vybrány atributy rok, měsíc, den, hodina, minuta, sekunda a ostatní atributy zobrazeny přehledně do sloupců. Takto lze pak vybrat a poté jednoduše podívat na hovory, které proběhly za určité období (například určitý den v měsíci). Na obrázku *Obrázek 33* je vyobrazena právě taková selekce.

fronta	- all -
rok	- all -
mesic	4
den	15
hodina	- all -
minuta	- all -
sekunda	- all -

cas hovoru	telefonni cislo	doba vyzv	byl pripoje	agent	doba hovo	kym byl ho
15.04.2016 18:09:00	29	1	1	sip/15	112	caller
15.04.2016 18:15:00	26	1	1	sip/12	173	caller
15.04.2016 18:20:00	23	2	1	sip/14	51	caller
15.04.2016 18:22:00	27	1	1	sip/11	149	caller
15.04.2016 18:22:00	26	1	1	sip/13	170	caller
15.04.2016 18:22:00	22	1	1	sip/15	83	caller
15.04.2016 18:22:00	25	2	1	sip/12	35	caller
15.04.2016 18:22:00	24	1	1	sip/14	166	caller

Obrázek 33: Kontingenční tabulka obecného pohledu

Report výkonnosti agentů

V případě pohledu na výkonnost agentů lze agregací zvolit hovorová fronta a zjistit následné všechny informace poskytnuté vytvořeným pohledem. Upravením formátu

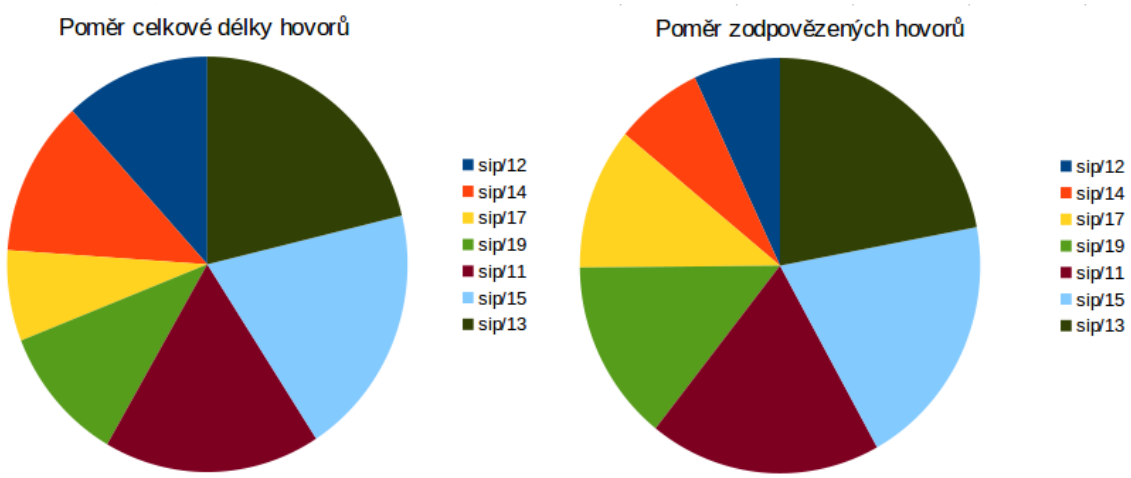
buněk lze pak také přehledněji zobrazit výsledky. Např. u vytvořeného atributu *celková doba hovoru* se nám po nastavení správného formátu (namísto desetinného čísla) zobrazí správný datový formát. Ukázka reportu na obrázku *Obrázek 34.*

fronta	- all -							
pocethovoru	nazev agenta	celkova doba hovoru	pocet	pocet u	průměrná do	průměr	poměr uk	poměr u
21	neprobehle hovory	00:00:00	0	0	0,00	0,00	0,00	0,00
83	sip/12	02:26:47	0	83	3,75	106,11	0,00	1,00
84	sip/14	02:31:16	0	84	4,93	108,05	0,00	1,00
131	sip/17	01:28:08	0	131	1,33	40,37	0,00	1,00
168	sip/19	02:13:14	0	167	1,25	47,87	0,00	1,00
222	sip/11	03:37:52	4	217	1,69	59,15	0,02	0,98
236	sip/15	04:04:36	0	234	2,31	62,72	0,00	1,00
261	sip/13	04:24:31	0	260	3,52	61,04	0,00	1,00

Obrázek 34: Kontingenční tabulka výkonnosti agentů

Z kontingenční tabulky lze také udělat grafické znázornění, které nám pomůže přehledněji zobrazit požadované informace.

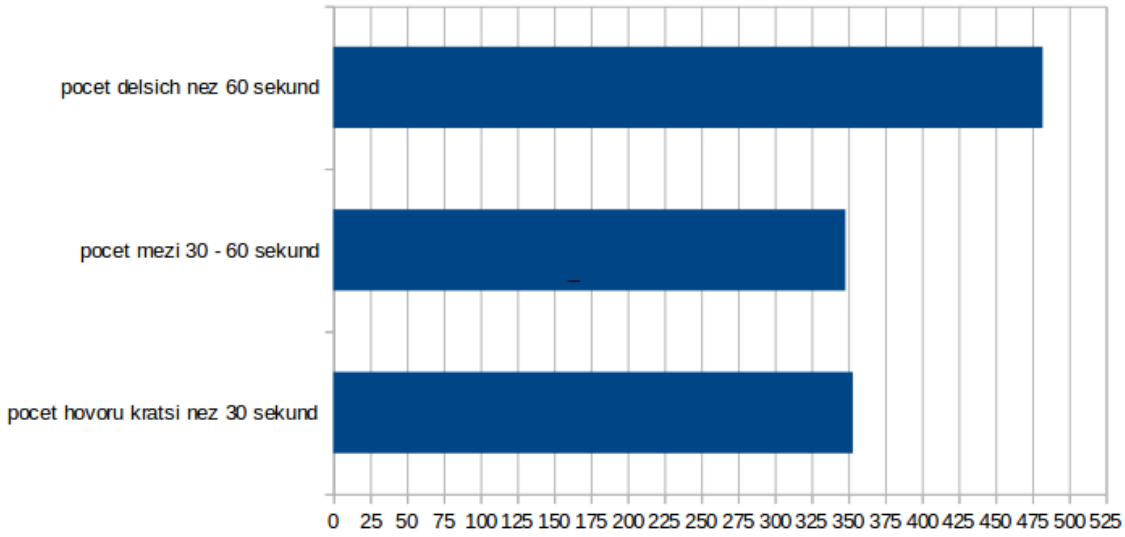
Například vytvořený koláčový graf (na obrázku *Obrázek 35*) , nám poskytne přehlednější zobrazení toho, jak se daný agent podílel na celkové délce všech hovorů či na počtu zodpovězených hovorů.



Obrázek 35: Koláčové grafy z analýzy agentů

Report informací o hovorech

Pohled na počet hovorů mezi intervaly s průměrnou dobou jednoho hovoru a průměrnou dobou vyzvánění může poskytnou například informaci ve formě grafu, kolik takových hovorů proběhlo a na základě toho stanovit maximální dobu hovoru, kterou agent může se zákazníkem absolvovat (*Obrázek 36*).



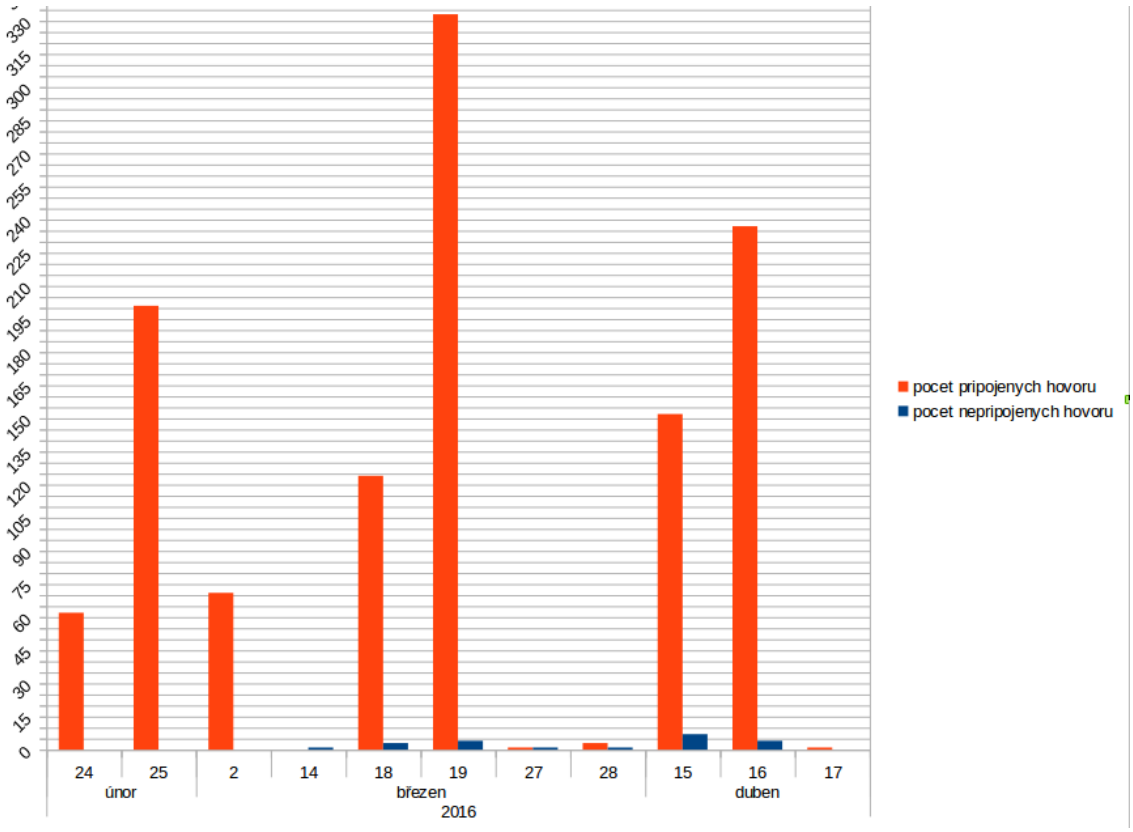
Obrázek 36: Graf počtu různě dlouhých hovorů

V případě posledního pohledu, to jest počet přijatých a nepřijatých hovorů v jednotlivých dnech, vytvořená kontingenční tabulka byla obohacena o celkový součet těchto hovorů a byl přidán filtr každého atributu. Filtr nám pak může poskytnout informace o těchto hovorech souhrnně za jednotlivé roky, měsíce či dny. Vytvořenou kontingenční tabulku znázorňuje obrázek Obrázek 37.

rok	mesic	den	pocet pripojeny	pocet nepripojeny	Sum	Sum
2016	únor	24	62	0	62	0
		25	201	0	201	0
	březen	2	71	0	71	0
		14	0	1	0	1
		18	124	3	124	3
		19	333	4	333	4
		27	1	1	1	1
		28	3	1	3	1
	duben	15	152	7	152	7
		16	237	4	237	4
		17	1	0	1	0
Total Result					**1185**	**21**

Obrázek 37: Kontingenční tabulka pohledu pro počet hovorů

Graf výsledků z kontingenční tabulky může znázorňovat například vytvořený graf na obrázku *Obrázek 38*. Jsou zde zobrazeny přijaté a nepřijaté hovory ve sloupcovém grafu v jednotlivých měřených dnech, jejich měsíců a let.

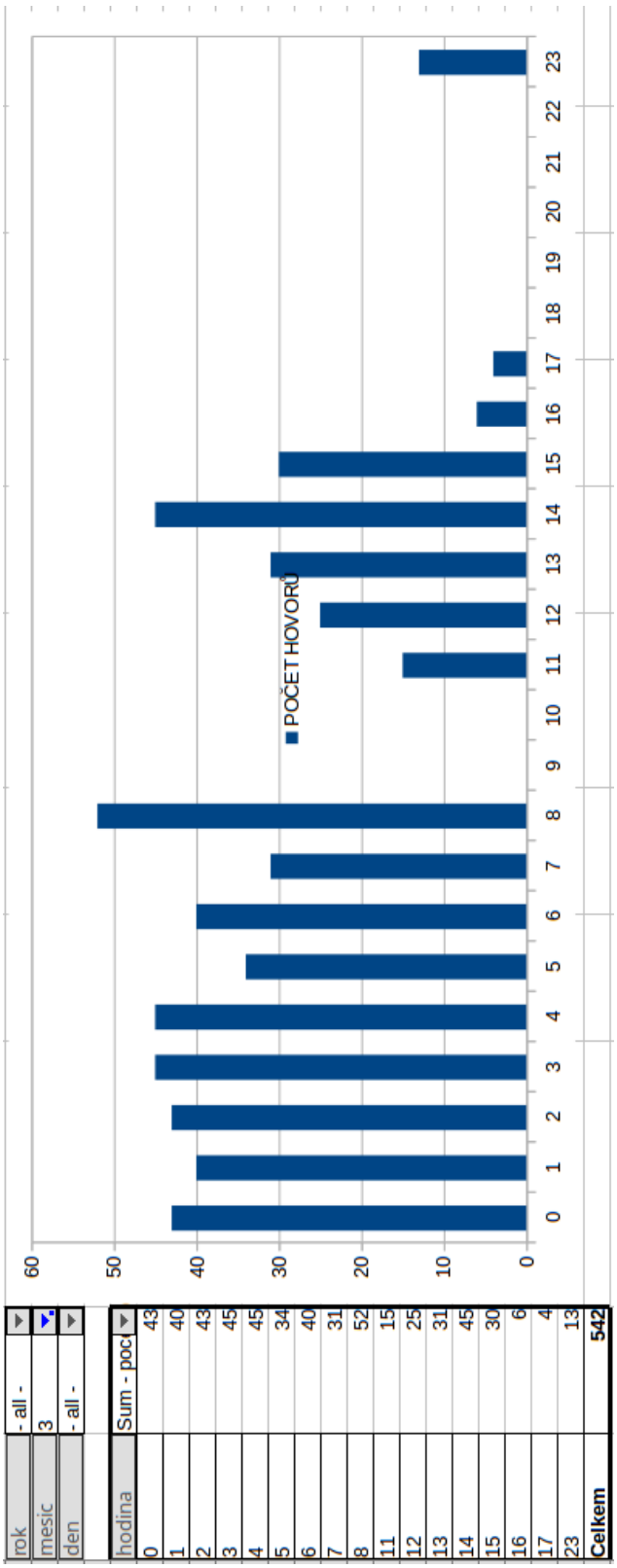


Obrázek 38: Graf počty připojených a nepřipojených hovorů za jednotlivé dny

Histogram hovorů

Histogram hovorů jako graf vizualizuje, jaká byla vytíženost hovorů fronty v jednotlivých hodinách. Dokáže nám tedy poskytnout informaci, kdy je potřeba zvýšit počet zaměstnanců a kdy jsou naopak agenti nevyužití. V případě vyhodnocování také hraje roli, jestli je tento den všední nebo se jedná o víkend či státní svátek. V tabulkovém procesoru toto lze identifikovat podle vybraného datumu.

Toto znázornění je vidět na obrázku *Obrázek 39: Histogram hovorů*, kde byl vybrán celý jeden měsíc a pivot tabulka sumarizuje počty hovorů v jednotlivých hodinách a pak i jejich celek.



Obrázek 39: Histogram hovorů

7. Závěr

V předkládané práci jsem řešil problematiku datového skladu, která spočívala ve vytvoření sady programů, které poskytly data ke statistickému a analytickému vyhodnocení.

V první části, tedy druhé kapitole jsem provedl seznámení se základy VoIP telefonie, komunikačními protokoly a telefonní ústřednou Asterisk a jeho monitorovacím rozhraním (AMI).

Ve třetí kapitole pak na základě zjištěných informací bylo provedeno testovací volání do hovorové fronty, na kterém bylo analyzováno rozhraní AMI. Při testování volání do hovorové fronty bylo zjištěno, že AMI generuje mnoho událostí s různými atributy, které bylo zapotřebí uložit do databáze. Tyto události byly detailně rozebrány a bylo navrhnuto řešení, jak tyto události ukládat.

Čtvrtá kapitola zahrnuje zamyšlení nad relačními databázemi a srovnává řešení vytváření databáze s nerelačními databázovými systémy. Bylo zjištěno, že relační databáze poskytuje pro danou úlohu vhodnější řešení a i jako mezistupeň vytváření datového skladu je dostačujícím nástrojem. Dále byl navrhnut model relační databáze a implementován ve vybraném RDBMS. Také byla vytvořena aplikace v jazyce Java, která umožňuje ukládání událostí z AMI rozhraní do vytvořené relační databáze.

V předposlední kapitole byl představen pojem Bussiness Inteligence, teorie budování datových skladů a popsány všechny části takového systému. Na základě nabytých informací a požadavků na sledování a analyzování hovorů v hovorové frontě byl navrhnut model datového skladu, představující schéma typu hvězda. Dále byl v RDBMS implementován proces ETL ve formě SQL procedury, který z dočasné relační databáze převádí potřebná data do datového skladu.

Poslední část práce byla věnována vytvoření virtuálního provozu telefonní ústředny, který naplnil relační databázi netriviálním množství dat, nad kterými byl spuštěn ETL proces a tedy byl naplněn potřebnými daty ke zpracování i datový sklad. Poté byl vytvořen report v tabulkovém procesoru na základě kontingenčních tabulek, který analyzoval data z datového skladu a vhodně je zobrazil do podoby různých typů grafů.

Seznam zkratek

.NET - Soubor technologií v softwarových produktech, které tvoří celou platformu, dostupnou pro Web (dot Net)

ACD - Automatic Call Distributor

ACID - Sada vlastností, zaručující spolehlivost databázových transakcí (Atomicity, Consistency, Isolation, Durability)

AGI - Asterisk Gateway Interface

AMI - Asterisk Manager Interface

API - Sada rutin, protokolů a nástrojů pro tvorbu softwaru (Application Programming Interface)

ASP - Technologie pro tvorbu webových aplikací a služeb (Active Server Pages)

BI - Bussiness Inteligence

BSD - Unixový operační systém (Berkeley Software Distribution)

CDR - Modul nahrávání hovoru v Asterisk (Storing Call Detail Records)

CEL - Modul logování kanálu událostí v Asterisk (Channel Event Logging)

CRM - Řízení vztahů se zákazníky (Customer Relationship Management)

DAHDI - Interface obsahující ovladače pro hardwarová zařízení (Digium/Asterisk Hardware Device Interface)

DNS - Hierarchický systém doménových jmen (Domain Name System)

DOLAP - Desktop OLAP

ERP - Plánování podnikových zdrojů (Enterprise Resource Planning)

ETL - Proces naplňování dat do datových skladů - Extrakce, Přeměna, Načtění (Extraction Transform Load)

FIFO - Strategie "první dovnitř, první ven" (First In First Out)

G.729 - Hlasový kodek pro přenos hlasu

GNU GPL - Všeobecná veřejná licence GNU (GNU General Public License)

GUI - Grafické uživatelské rozhraní (Graphical User Interface)

H.323 - Definice protokolů pro audio-vizuální relace komunikace v jakékoli paketové síti

HOLAP - Hybridní OLAP

HTTP - Internetový protokol určený pro výměnu hypertextových dokumentů (Hypertext Transfer Protocol)

IAX - Komunikační protokol pro Asterisk (Inter-Asterisk eXchange)

IDE - Integrované vývojové prostředí (Integrated Development Environment)

IETF MMUSIC - Pracovní skupina, která vytvořila SIP protokol (Multiparty Multimedia Session Control Working Group)

IP - Základní protokol pracující na síťové vrstvě (Internet Protocol)

IVR - Samoobslužný hlasový systém (Interactive Voice Response)

JAVA - Programovací jazyk

JDBC - **API** pro připojení k databázi pro jazyk Java (Java Database Connectivity)

LAMP - Sada svobodného softwaru používaného jako platforma pro implementaci dynamických webových stránek (Linux Apache MySQL(MariaDB) PHP(Perl,Python))

MIT - Svobodná licence vzniknuvší na Massachusettském technologickém institutu

MOLAP - Multidimenzionální OLAP

MS - Microsoft

NoSQL - Not Only SQL

OLAP - Technologie uložení dat (Online Analytical Processing)

OLTP - Technologie uložení dat (Online Transaction Processing)

PBX - Pobočková telefonní ústředna Private Branch Exchange)

PC - Osobní počítač (Personal Computer)

PHP - Skriptovací programovací jazyk

PIN - Osobní identifikační číslo (Personal Identification Number)

PL/pgSQL - PL/SQL pro PostgreSQL

PL/SQL - Procedurální nadstavba jazyka SQL (Procedural Language/Structured Query Language)

PSTN - Veřejná telefonní síť (Public Switched Telephone Network)

RDBMS - Relační systém řízení báze dat (Relational Database Management System)

RFC 3261 - Dokument pro popsání SIP protokolu

ROI – Návratnost investice (Return Of Investment)

ROLAP - Relační OLAP

RTP - Protokol standardizující paketové doručování zvukových a obrazových dat po internetu (Real-time Transport Protocol)

SAP - Softwarovým produktem sloužící pro řízení podniku (Systems - Applications - Products)

SDP - Formát popisující parametry streamování (Session Description Protokol)

SIP - Protokol pro inicializaci relací (Session Initiation Protocol)

SQL - Strukturovaný dotazovácí jazyk (Structured Query Language)

SWOT - Metoda pro identifikování silných a slabých stránek, příležitosti a hrozeb (Strengths Weaknesses Opportunities Threats)

TCP - Protokol transportní vrstvy v sadě protokolů TCP/IP (Transmission Control Protocol)

TCP/IP - Sada protokolů pro komunikaci v počítačové síti (Transmission Control Protocol/Internet Protocol)

TDM - Metoda přenosu signálů klasické digitální telefonie (Time Division Multiplex)

UDP - Protokol transportní vrstvy TCP/IP (User Datagram Protocol)

URI - Jednotný identifikátor zdroje (Uniform Resource Identifier)

VoIP – Technologie umožňující přenos digitalizovaného hlasu v těle paketů TCP/IP prostřednictvím počítačové sítě (Voice over Internet Protocol)

XML - Rozšiřitelný značkovací jazyk) (Extensible Markup Language)

Literatura

[1] MADSEN, Leif, Jim VAN MEGGELEN a Russell BRYANT. *Asterisk™:The Definitive Guide* [online]. 2011 [cit. 2016-03-25]. Dostupné z: http://www.asteriskdocs.org/en/3rd_Edition/asterisk-book-html/asterisk-book.html

[2] *Getting Started With Asterisk* [online]. 2016 [cit. 2016-03-25]. Dostupné z: http://www.asterisk.org/get-started

[3] WALLACE, Kevin. VoIP bez předchozích znalostí. Vyd. 1. Brno: Computer Press, 2007. Cisco systems. ISBN 978-80-251-1458-2.

[4] TONCAR, Vladimír. *VoIP Protocols: Introducing SIP* [online]. 2014 [cit. 2016-04-17]. Dostupné z: http://toncar.cz/Tutorials

[5] *What is a VoIP Gateway?* [online]. 2016 [cit. 2016-03-25]. Dostupné z: https://www.digium.com/products/voip-gateways

[6] ROSENBERG, Jonathan, Henning SCHULZRINNE, Gonzalo CAMARILLO, Alan JOHNSTON, Jon PETERSON, Robert SPARKS, Mark HANDLEY a Eve SCHOOLER. *SIP: Session Initiation Protocol* [online]. 2002 [cit. 2016-03-29]. Dostupné z: http://www.hjp.at/doc/rfc/rfc3261.html

[7] *Telefonujeme se SIP* [online]. 2006 [cit. 2016-03-28]. Dostupné z: http://www.root.cz/clanky/telefonujeme-se-sip/

[8] HINKLE, Channig. *Installing AsteriskNOW* [online]. 2015 [cit. 2016-04-01]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Installing+AsteriskNOW

[9] *Přehled vlastností Kerio Operator* [online]. 2016 [cit. 2016-04-01]. Dostupné z: http://www.kerio.cz/products/kerio-operator/features

[10] *Kerio Operator - Často kladené dotazy* [online]. 2011 [cit. 2016-04-01]. Dostupné z: http://www.pcobchod.sk/downloads/Operator.PDF

[11] *Asterisk AGI* [online]. 2016 [cit. 2016-04-02]. Dostupné z: http://www.voip-info.org/wiki/view/Asterisk+AGI

[12] *The Asterisk Manager Interface (AMI) Protocol* [online]. [cit. 2016-04-03]. Dostupné z: http://marcelog.github.io/articles/php_asterisk_manager_interface_protocol_tutorial_int roduction.html

[13] *Telnet* [online]. 2015 [cit. 2016-04-03]. Dostupné z: https://cs.wikipedia.org/wiki/Telnet

[14] GONZALEZ, Juan Carlos. *Log in, Log out from Asterisk AMI with Telnet* [online]. 2015 [cit. 2016-04-03]. Dostupné z: http://www.jcgonzalez.com/test-asterisk-ami-telnet

[15] WALLEN, Jack. *Best VOIP Clients for Linux* [online]. 2015 [cit. 2016-04-03]. Dostupné z: https://www.linux.com/news/software/applications/812894-best-voip-clients-for-linux-that-arent-skype/

[16] *Asterisk Manager: Events* [online]. 2014 [cit. 2016-04-03]. Dostupné z: http://www.voip-info.org/wiki/view/asterisk+manager+events

[17] JORDAN, Matt. *Asterisk 13 AMI Events* [online]. 2014 [cit. 2016-04-03]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+13+AMI+Events

[19] *Výkladový slovník* [online]. 2005 [cit. 2016-04-05]. Dostupné z: http://www.abclinuxu.cz/slovnik/rdbms

[20] *PostgreSQL - About* [online]. 2016 [cit. 2016-04-05]. Dostupné z: http://www.postgresql.org/about/

[21] *Relational database management system* [online]. 2016 [cit. 2016-04-05]. Dostupné z: https://en.wikipedia.org/wiki/Relational_database_management_system

[22] *What is MySQL?* [online]. 2016 [cit. 2016-04-05]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

[23] *System Properties Comparison Microsoft SQL Server vs. MySQL vs. Oracle NoSQL vs. PostgreSQL* [online]. 2016 [cit. 2016-04-05]. Dostupné z: http://db-engines.com/en/system/Microsoft+SQL+Server%3BMySQL%3BOracle+ NoSQL%3BPostgreSQL

[24] MISTRY, Ross a Stacia MISNER. *Introducing Microsoft SQL Server 2014* [online]. Redmond, Washington: Microsoft Press, 2014 [cit. 2016-04-29]. ISBN 978-073-5684-751. Dostupné z: https://blogs.msdn.microsoft.com/microsoft_press/2014/04/02/free-ebook-introducing-microsoft-sql-server-2014/

[25] *Licencování systému SQL Server* [online]. 2016 [cit. 2016-04-06]. Dostupné z: https://www.microsoft.com/cs-cz/server-cloud/products/sql-server/purchasing.aspx

[26] *Oracle's History: Innovation, Leadership, Results* [online]. 2007 [cit. 2016-04-06]. Dostupné z: http://www.oracle.com/us/corporate/profit/p27anniv-timeline-151918.pdf

[27] *Ranking of Relational DBMS* [online]. 2016 [cit. 2016-04-06]. Dostupné z: http://db-engines.com/en/ranking/relational+dbms

[28] REUTER, Stefan. *Asterisk-Java Documentation* [online]. 2014 [cit. 2016-04-06]. Dostupné z: https://maven.reucon.com/projects/public/asterisk-java/1.0.0.CI-SNAPSHOT/asterisk-java/#apidocsindex.html

[29] NOVOTNÝ, Ota, Jan POUR a David SLÁNSKÝ. *Business intelligence: jak využít bohatství ve vašich datech*. 1. vyd. Praha: Grada, 2005. Management v informační společnosti. ISBN 80-247-1094-3.

[30] KIMBALL, Ralph. a Margy ROSS. *The data warehouse toolkit: the complete guide to dimensional modeling* [online]. 2nd ed. New York: Wiley, c2002 [cit. 2016-04-07]. ISBN 978-0471200246. Dostupné z: http://www.essai.rnu.tn/Ebook/Informatique/The%20Data%20Warehouse%20Toolkit,%203rd%20Edition.pdf

[31] PETERKA, Miloslav. *BI Experts - Seznamte se s BI* [online]. 2010 [cit. 2016-04-09]. Dostupné z: http://www.daquas.cz/articles/379-seznamte-se-s-bi

[32] LABERGE, Robert. *Datové sklady: agilní metody a business intelligence*. 1. vyd. Brno: Computer Press, 2012. ISBN 978-80-251-3729-1.

[33] *10 Open Source ETL Tools* [online]. 2015 [cit. 2016-04-10]. Dostupné z: http://www.datasciencecentral.com/profiles/blogs/10-open-source-etl-tools

[34] *Data Warehouses/Data Marts Vs. Marketing Databases* [online]. 2002 [cit. 2016-04-10]. Dostupné z: http://www.dmnews.com/dataanalytics/data-warehousesdata-marts-vs-marketing-databases/article/76889/

[35] *Oracle - Database Data Warehousing Guide* [online]. 2015 [cit. 2016-04-10]. Dostupné z: https://docs.oracle.com/cd/B28359_01/server.111/b28313/logical.htm

[36] *Uložení dat v OLAP systémech* [online]. 2002 [cit. 2016-04-10]. Dostupné z: http://datamining.xf.cz/view.php?cisloclanku=2002102810

[37] ZÁVODNÝ, Martin. *Bussiness Inteligence Systems* [online]. 2012 [cit. 2016-04-10]. Dostupné z: http://tt.pef.czu.cz/Files/3_printVersion_219.pdf

[38] *PJSIP - wiki* [online]. 2014 [cit. 2016-04-10]. Dostupné z: http://trac.pjsip.org/repos

[39] CARPENTER, Colman. *Asterisk 1.4: the professional's guide : implementing, administering, and consulting on commercial IP telephony solutions* [online]. Birmingham, U.K.: Packt Publishing Ltd., c2009 [cit. 2016-04-19]. ISBN 978-1-847194-38-1. Dostupné z: http://asterisk.ru/store/files/Asterisk-1.4.pdf

Seznam tabulek

Seznam obrázků

Přílohy

A. Uživatelský manuál

A.1 Nastavení prostředí

A.1.1 Hardwarové požadavky

Dle informací dostupných z *http://blog.linuxmint.com/?p=2977* pro operační systém Linux Mint ve verzi 17.3 s nainstalovaným prostředím XFCE splňuje doporučené požadavky na hardware následující konfigurace:

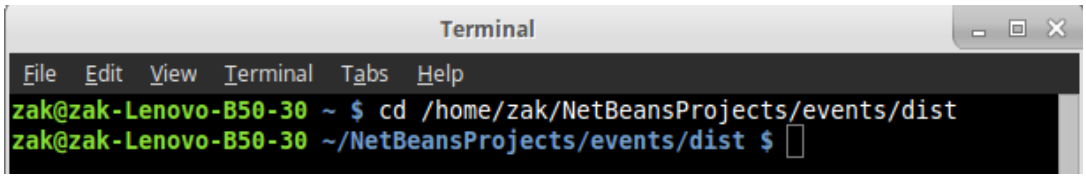
- CPU: 1Ghz+ (architektura x86 nebo x64)
- RAM: 1GB operační paměti
- HDD: 20GB
- Grafická karta podporující rozlišení alespoň 1024x768

A.1.2 Softwarové požadavky

- Operační systém Linux Mint 17.3 XFCE
- PostreSQL 9.4.7
- Java RE nebo Java DK verze 1.8

A.2 Spouštění aplikací

A.2.1 Spuštění aplikace pro sběr dat



Obrázek 40: Příklad otevření cesty do složky

1. Spustíme terminál a příkazem *cd* otevřeme cestu do složky obsahující Java spustitelný soubor *events.jar* (*Obrázek 40*).
2. Příkazem *java -jar events.jar* spustíme aplikaci pro sběr dat.

A.2.2 Spuštění skriptu pro proces ETL

1. Nalezneme cestu v terminálu ke skriptu pro spuštění ETL procesu podobně jako v případě spuštění Java aplikace.
2. Příkazem *chmod u+x etl_script.sh* nastavíme práva spuštění skriptu.
3. Příkazem s parametry v podobě *./etl_script.sh „název databáze“ „uživatelské jméno“* spustíme proces ETL. Podoba příkazu může být tedy následující: *./etl_script.sh postgres zak*

Na následujícím obrázku (*Obrázek 41*) vidíme úspěšné spuštění skriptu pro ETL.

```
zak@zak-Lenovo-B50-30 ~/mega/1_STRUKTURA_DVD/APLIKACE $ ./etl_script.sh postgres zak
 etl
-----

(1 row)

WARNING:  there is no transaction in progress
COMMIT
Skript procesu ETL dokoncen
```

Obrázek 41: Spuštění skriptu pro ETL

Poznámka: Skript pro ETL je spustitelný i z Java aplikace bez nutnosti použít příkazovou řádku, názorně ukázáno v kapitole *A.3. Ovládání*.

Poznámka 2: Spuštění procedury pro ETL proces lze samozřejmě uskutečnit i SQL příkazem v RDMS, tedy příkazem *SELECT etl();* .

A.2.3 Spuštění skriptu pro simulaci provozu telefonní ústředny

1. Stejným způsobem jako v předchozích kapitolách se dostaneme do složky

obsahující skript *sim_provoz.sh*.

2. Opět nastavíme práva příkazem:

   ```
   chmod u+x sim_provoz.sh
   ```

3. Aplikaci s parametrem IP adresy serveru telefonní ústředny a cestou k PJSIP(aplikace dostupna ve složce PODPURNE_PROGRAMY) spustíme příkazem:

   ```
   ./sim_provoz „ip adresa serveru“ „cesta k PJSUA“
   ```

   Podoba příkazu může být následující:

   ```
   ./sim_provoz 192.168.234.131 /home/zak/pjproject-2.4.5/pjsip-apps/bin/
   ```

Poznámka: Spuštění simulovaného provozu může být uskutečněno taktéž přes Java aplikaci a bude názorně ukázáno v kapitole *A.3. Ovládání*.

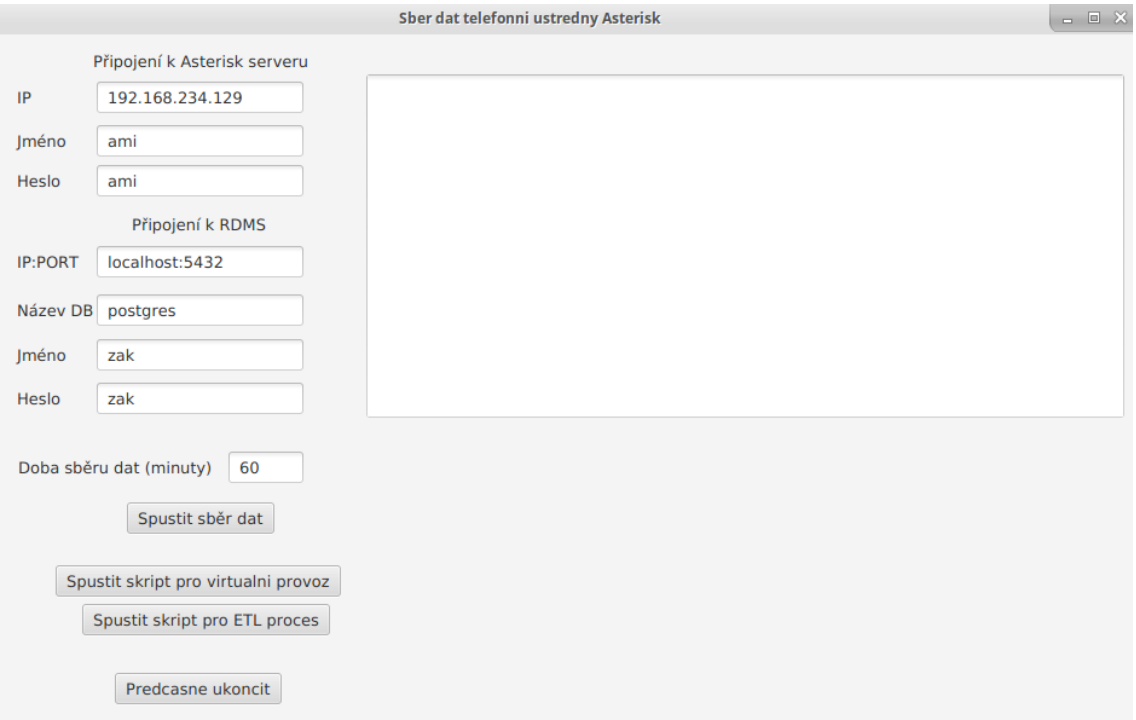
A.3. Ovládání

A.3.1 Spuštění sběru dat

Po spuštění aplikace podle návodu v kapitole *2.1* se nám zobrazí následující okno s GUI aplikace. V levém panelu jsou pole k vyplnění pro připojení k AMI Asterisk serveru a k přihlášení k PostgreSQL a pole s vyplněním doby sběru dat (minuty). Pod panelem se nachází tlačítka, jejichž funkce je snadno identifikovatelná podle názvu. V pravé části se pak nachází konzole, která vypisuje různé události (či chyby) při spuštění. GUI aplikace znázorněno na obrázku *Obrázek 42*.

Změníme předvyplněná pole k připojení k AMI Asterisk serveru a k přihlášení k PostgreSQL podle našich požadavků a vyplníme pole s počtem minut, po které bude aplikace sbírat data.

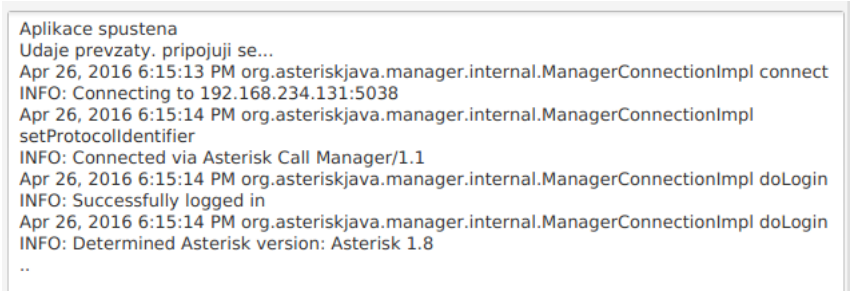
Připojení k Asterisk serveru, ať už na virtuálním stroji nebo jiném, zjistíme rovnou ze serveru, kde je adresa defaultně zobrazena (*Obrázek 43*).



Obrázek 42: GUI Java aplikace

Obrázek 43: IP adresa serveru Kerio Operator

Po stisknutí tlačítka Spustit nám konzole vypíše informace o úspěšném spuštění sběru dat (*Obrázek 44*).



Obrázek 44: Výpis terminálu při spuštění aplikace.

Stisknutím tlačítka *Předčasně ukončit* se ukončí aplikace před vypršení předem zadané lhůty.

A.3.2 Spuštění skriptů

Tlačítka *Spustit skript pro virtuální provoz* a *Spustit skript pro ETL* spouští předem nastavené skripty s těmito funkcemi. Jak bylo popsáno v kapitole *2*, oba skripty potřebují při spuštění zadat parametry. Tyto parametry s v případě spuštění z Java aplikace berou z příslušných vyplněných polí.

V případě spuštění skriptu pro virtuální provoz využívá parametr z pole *IP* a v případě spuštění skriptu pro ETL parametry z polí *název DB* a *uživatel*.

B. Dokument nasazení programu

B.1 Instalace a nastavení PostgreSQL na Linux Mint

B.1.1 Přidání repositáře a instalace PostgreSQL

1. Následující příkaz v terminálu vytvoří soubor, do kterého se přidá odkaz na repositář pro instalaci PostgreSQL:

   ```
   $ sudo touch /etc/apt/sources.list.d/pgdg.list
   ```

2. Vytvořený soubor otevřeme v nějakém textovém editoru:

   ```
   $ sudo gedit /etc/apt/sources.list.d/pgdg.list
   ```

3. Přidáme následující řádku:

   ```
   deb http://apt.postgresql.org/pub/repos/apt/ trusty-pgdg main
   ```

4. Přidáme klíč k repozitáři:

   ```
   $ sudo apt-get install wget ca-certificates
   $ wget --quiet -O -
   https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-
   key add -
   ```

5. Provedeme aktualizaci zdrojů balíčků

   ```
   $ sudo apt-get update
   ```

6. Nainstalujeme PostgreSQL ve verzi 9.4 a případně grafické rozhraní *pgadmin*

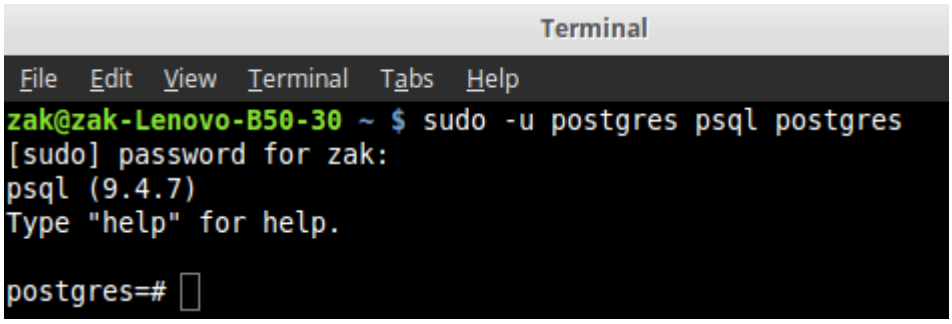
   ```
   $ sudo apt-get install postgresql-9.4 pgadmin3
   ```

Poznámka: Je možné použít i na jinou linuxovou distribuci, která má prostředí *XFCE*, přidání verze repositáře se odlišuje podle verze Linuxu.

B.1.2 Nastavení PostgreSQL k importu SQL Dump

1. Příkazem v terminálu se přepneme do konzole Postgres jak je vidět na obrázku *Obrázek 45:*

```
$ sudo -u postgres psql postgres
```

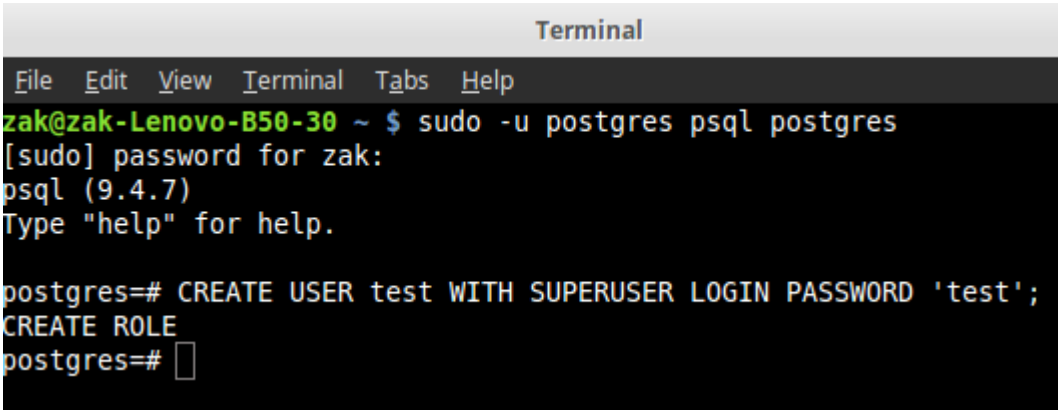


Obrázek 45: Připojení do konzole postgres

2. Následujícím příkazem vytvoříme uživatele, který bude mít oprávnění pro manipulaci s databázemi v PostgreSQL:

```
CREATE USER jméno_uživatele WITH SUPERUSER LOGIN PASSWORD
'heslo';
```

3. Vytvoření uživatele ověříme výpisem *CREATE ROLE* v konzoli (*Obrázek 46*):



Obrázek 46: Vytvoření uživatele v postgres konzoli

B.1.3 Import SQL Dump

Ve složce *SQL* v adresářové struktuře DVD přílohy nalezneme soubor *sqldump*.

V terminálu otevřeme cestu do této složky a následujícím příkazem importujeme celou databázi i s daty.

```
$ psql název_db < sqldump
```

Poznámka: Standardní název databáze, která se vytvoří při instalaci PostgreSQL je *postgres*. Lze si vytvořit i jinou databázi a provést import tam.

Poznámka 2: Je možné importovat i samotnou databázi bez vygenerovaných dat. V tomto případě použijeme k importu soubor *sqldump_nodata*, který je obsažen ve stejném adresáři.

B.2 Instalace Java Runtime Enviroment 1.8

Ve složce *PODPURNE_PROGRAMY* se nachází *JRE*, která je zapotřebí pro chod aplikací.

1. V terminálu se přepneme do této složky a rozbalíme archiv s *JRE:*

   ```
   $ tar -zxvf jre-8u91-linux-x64.tar.gz
   ```

2. Vytvoříme složku v „Option“ systémové složce pro instalaci aplikací třetích stran:

   ```
   $ sudo mkdir -p -v /opt/java/64
   ```

3. Rozbalený archiv přesuneme do této vytvořené složky:

   ```
   $ sudo mv -v jre1.8.0_91 /opt/java/64
   ```

4. Následujícími příkazy přidáme JRE do prostředí pro spouštění aplikací:

   ```
   $ sudo update-alternatives --install "/usr/bin/java" "java"
   "/opt/java/64/jre1.8.0_91/bin/java" 1
   $ sudo update-alternatives --set java
   /opt/java/64/jre1.8.0_91/bin/java
   ```

B.3 Instalace VMware pro spuštění virtuálního serveru Kerio Operator

Software *Vmware* pro spouštění virtuálních počítačů se nachází na DVD ve složce *PODPURNE_PROGRAMY*.

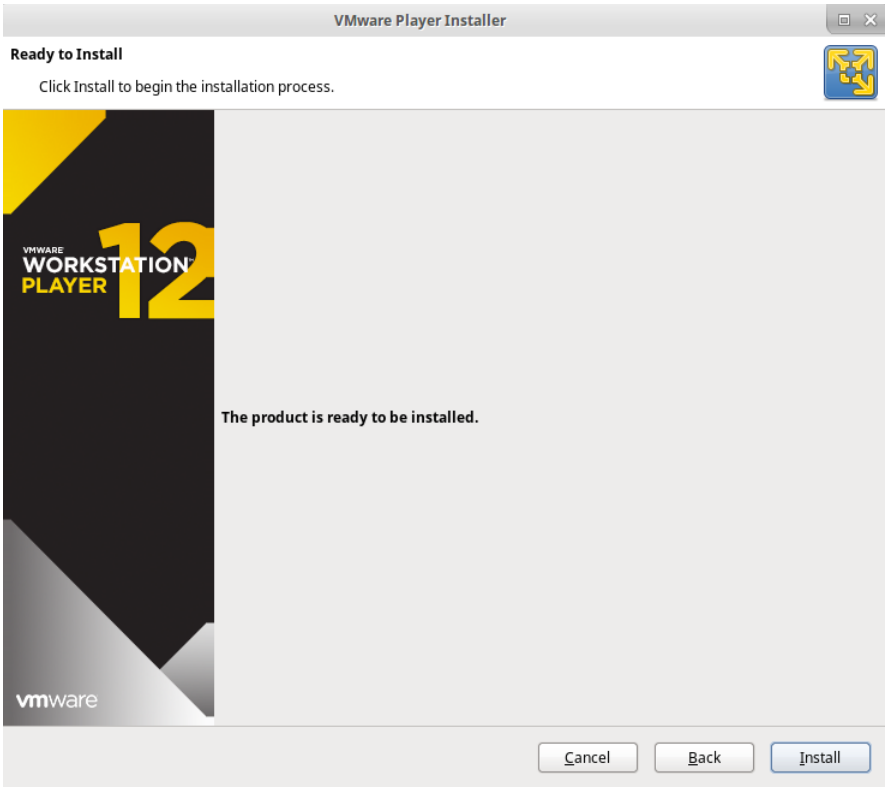
1. Pro instalaci je nutné nastavit oprávnění příkazem:

   ```
   $ chmod u+x Vmware-Player-12.bundle
   ```

2. Spuštění instalace proběhne následujícím příkazem:

   ```
   $ sudo ./Vmware-Player-12.bundle
   ```

3. Poté stačí následovat pokyny na obrazovce (*Obrázek 47*).



Obrázek 47: Instalace VMware

Poznámka: Virtuální server s nainstalovaným a nastaveným OS Kerio Operator (obsažen ve složce PODPURNE_PROGRAMY na DVD) stačí otevřít ve VMware.

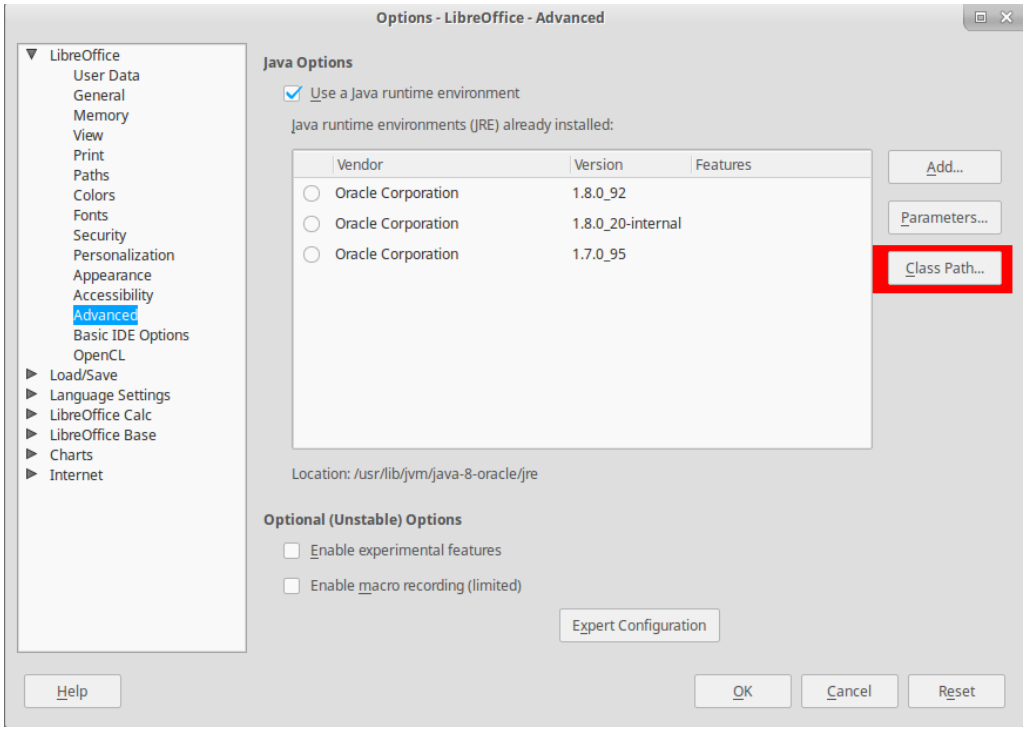
B.4 Nastavení LibreOffice Calc pro spojení s PostgreSQL databází

Soubor *report.ods* DVD přílohy obsahuje report dat z datového skladu. Pro aktualizace jeho obsahu je potřeba *Calc* napojit na vytvořenou PostreSQL databázi.

Pro vytvoření připojení v tabulkovém procesu je nutné založit databázové připojení. Ve složce *PODPURNE_PROGRAMY* se nachází *JDBC* ovladač, který nám připojení s databází zprostředkuje.

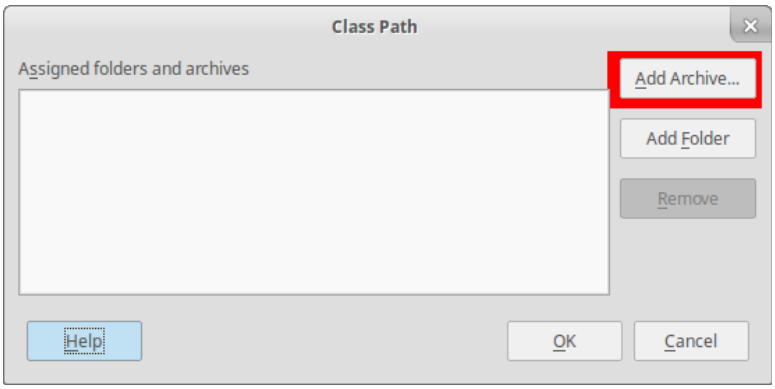
Nastavení v tabulkové procesoru proběhne následujícím způsobem.

1. Nejdříve je nutné přidat JDBC ovladač. Z nástrojového panelu aplikace *Calc* vybereme *Tools* → *Options* → *Advanced* a stiskneme tlačítko *Class Path* (*Obrázek 48*).



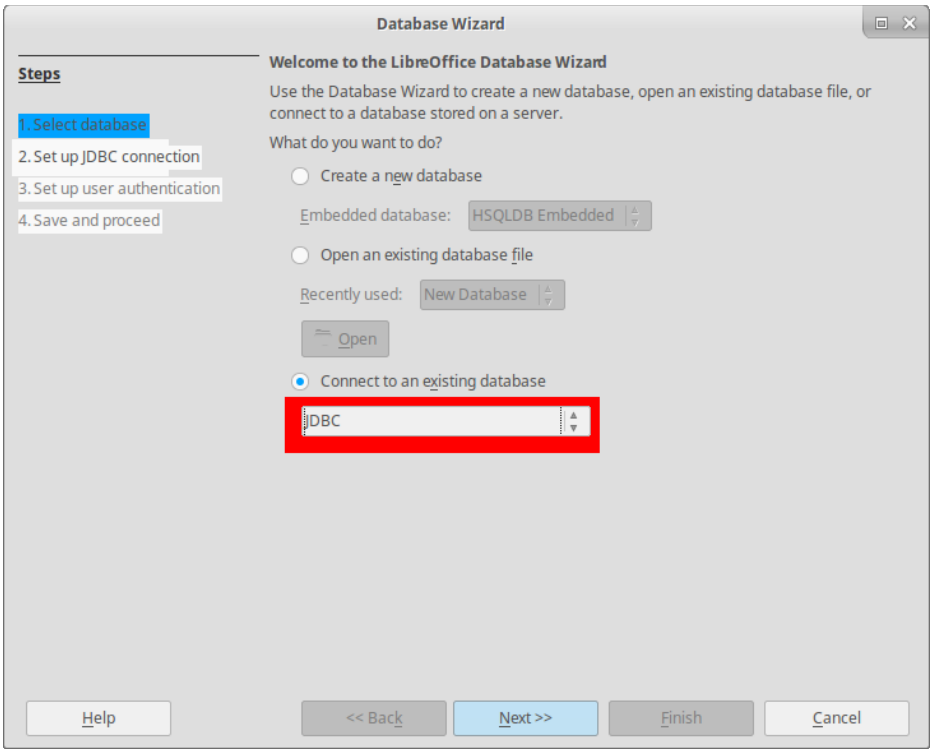
Obrázek 48: Rozšířená nastavní Java

2. Stisknutím tlačítka vybereme JDBC ovladač (*Obrázek 50*).

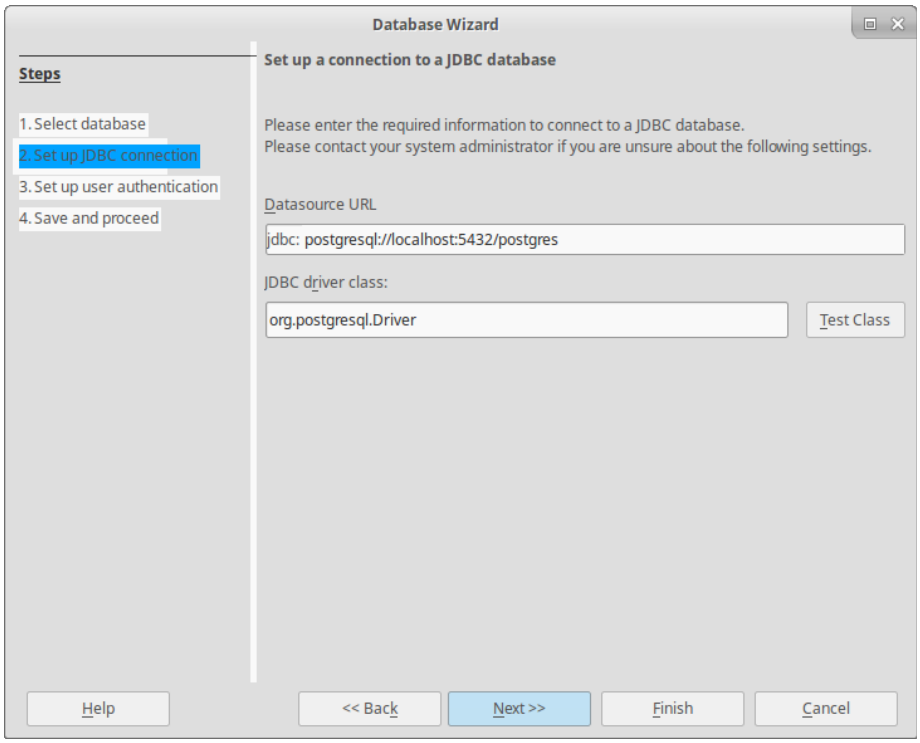


Obrázek 49: Přídaní Class Path

3. Poté je nutné *Calc* restartovat.
4. V nástrojové liště vybereme *File* → *New* → *Database* k založení nového databázového připojení.
5. Vybereme *JDBC* → *Next* a vyplnímě údaje k připojení (*Obrázek 50*).

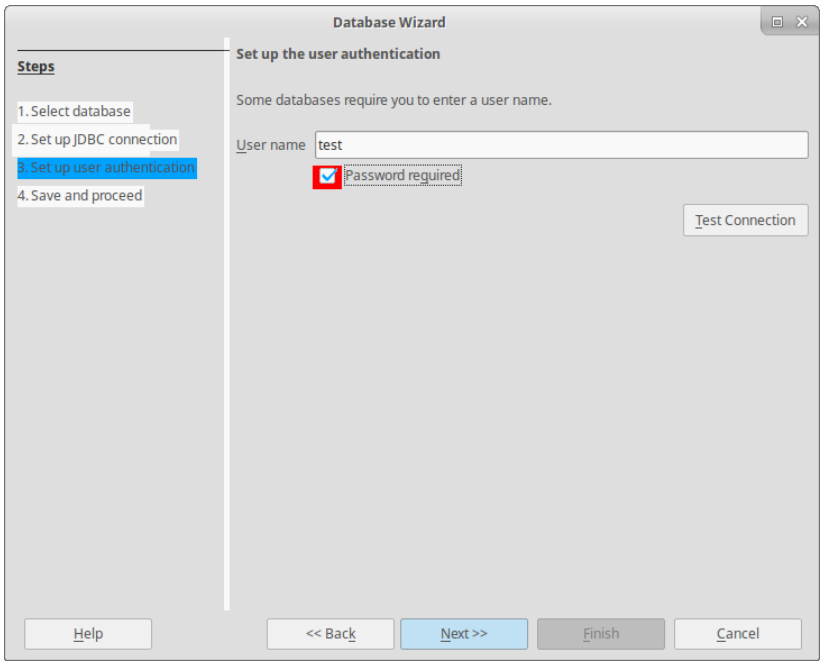


Obrázek 50: Připojení k databázi



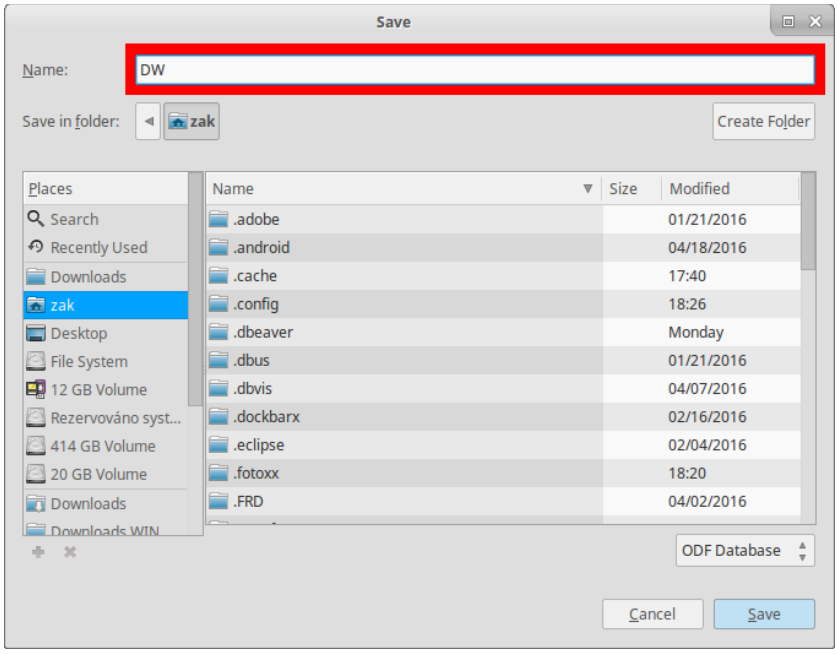
Obrázek 51: Vyplnění URL a třídy ovladače

6. V dalším okně vyplníme název uživatele PostgreSQL a zaškrtneme nutnost zadání hesla (*Obrázek 52*).



Obrázek 52: Autentizace připojení

7. Stiskneme *Next* a v dalším okně *Finish*.
8. Vyskočí nám další okno k uložení souboru s připojením. Je nutné jej pojmenovat *DW*, aby vzniklé kontingenční tabulky a grafy čerpaly ze správného připojení (*Obrázek 53*).



Obrázek 53: Uložení souboru s DB připojením s názvem DW

C. Obsah DVD

Zde je rozepsán obsah DVD nosiče přiloženého k této diplomové práci.

BASH_SKRIPTY – Obsahuje skripty ke spuštění procesu ETL a k simulaci provozu telefonní ústředny.

JAVA_APLIKACE – Obsahuje vytvořenou Java aplikaci pro sběr dat.

- SPUSTITELNA_APLIKACE - obsahuje spustitelnou verzi, včetně JavadDoc
- ZDROJOVE_KODY – obsahuje zdrojové kódy

PODPURNE_PROGRAMY – Obsahuje programové vybavení nutné ke spuštění aplikací.

POSTER – Vytvořený poster ve formátech PDF a PUB.

REPORT – Obsahuje soubor s reportem ve formátu ODS (Libre Office).

SQL – Obsahuje soubory sqldumb k integraci vytvořených programů do RDMS PostgreSQL. První varianta je včetně nasbíraných dat, druhá varianta nikoli (soubor *sqldump_nodata*).

TEXT – Obsahuje tento dokument ve formátech PDF a ODT (Libre Office).

READ_ME.txt – Obsahuje tento rozpis obsahu DVD nosiče.