Univerza v Mariboru
Fakulteta za elektrotehniko,
računalništvo in informatiko

Tomaž Hrnčič

ANALIZA IN PRIMERJAVA ORODIJ ZA INTEGRACIJO PODATKOV

Diplomsko delo

Maribor, September, 2013

ANALIZA IN PRIMERJAVA ORODIJ ZA INTEGRACIJO PODATKOV

Diplomsko delo

Študent:	Tomaž Hrnčič
Študijski program:	Visokošolski študijski program Informatika in Tehnologije Komuniciranja
Smer:	Sistemska Podpora Informatiki in Tehnologijam Komuniciranja
Mentor:	doc. dr. Marko Hölbl, univ. dipl. inž. rač. in inf.
Somentor:	asist. Andrej Sevčnikar, mag. univ. dipl. inž. rač. in inf.
Lektor(ica):	Nina Skube, prof. slov.

Maribor, September, 2013

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

Številka: E1009026
Datum in kraj: 29. 08. 2012, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 01/2010)
izdajam

SKLEP O DIPLOMSKEM DELU

1. **Tomažu Hrnčiču,** študentu visokošolskega strokovnega študijskega programa INFORMATIKA IN TEHNOLOGIJE KOMUNICIRANJA, smer Sistemska podpora informatiki in tehnologijam komuniciranja, se dovoljuje izdelati diplomsko delo pri predmetu Skladiščenje podatkov in poročanje.

2. **MENTOR: doc. dr. Marko Hölbl**
   SOMENTOR: asist. Andrej Sevčnikar

3. **Naslov diplomskega dela:**
   ANALIZA IN PRIMERJAVA ORODIJ ZA INTEGRACIJO PODATKOV

4. **Naslov diplomskega dela v angleškem jeziku:**
   ANALYSIS AND COMPARISON OF ETL TOOLS

5. Diplomsko delo je potrebno izdelati skladno z "Navodili za izdelavo diplomskega dela". Skladno s 7. členom *Pravilnika o postopku priprave in zagovora diplomskega dela na dodiplomskem študiju*, je bilo odobreno podaljšanje roka za oddajo diplomskega dela do 30. 09. 2013. Diplomsko delo študent-ka odda v treh izvodih (dva vezana izvoda in en v spiralo vezan izvod) ter en izvod elektronske verzije v referatu za študentske zadeve.

Pravni pouk: Zoper ta sklep je možna pritožba na senat članice v roku 3 delovnih dni.

Dekan:

Obvestiti:
- kandidata,
- mentorja,
- somentorja,
- odložiti v arhiv.

ZAHVALA

Zahvaljujem se mentorju doc. dr. Marku Hölblu in somentorju asist. Andreju Sevčnikarju za vso pomoč in vodenje pri opravljanju diplomske naloge.

Posebna zahvala velja tudi staršem, ki so mi omogočili študij in izkazovali vso podporo.

ANALIZA IN PRIMERJAVA ORODIJ ZA INTEGRACIJO PODATKOV

Ključne besede: ETL, podatkovno skladišče, integracija, analiza, primerjava orodij

UDK: 004.6(043.2)

Povzetek

Diplomska naloga je namenjena predstavitvi, analizi in primerjavi treh izbranih orodij za integracijo podatkov: Pentaho Data Integration, Oracle Data Integrator in Microsoft SQL Server Integration Services. Prav tako je predstavljen splošen problem integracije, ter način, ki je potreben za njegovo razrešitev. Razložen je pojem ETL in naloge njegovih korakov v procesu integracije. Diploma na kratko opiše tudi vlogo ter arhitekturo podatkovnih skladišč, ki se uporabljajo ob izvajanju omenjenih postopkov.

Na praktičnem primeru je izvedena analiza virov podatkov, na osnovi katere je zgrajen model podatkovnega skladišča, ter podrobno opisan in izveden postopek integracije podatkov za vsa omenjena orodja. Za dosego zanesljive primerjave orodij, so vsi procesi ETL izvajani pod enakimi pogoji in tehnologijami podatkovnih baz. Na koncu so prikazani rezultati primerjav po izbranih merilih na podlagi analiz in izkušenj, ter predstavljene splošne ugotovitve.

ANALYSIS AND COMPARISON OF ETL TOOLS

Key words: ETL, data warehouse, integration, analysis, tool comparison

UDK: 004.6(043.2)

Abstract

The thesis aims to present, analyse and compare three distinct data integration tools: Pentaho Data Integration, Oracle Data Integrator and SQL Server Integration Services. It explains the general problem of integration and the manner in which it should be resolved. It also explains the concept of ETL, describes its steps in the process of integration and briefly explains the role and architecture of data warehouses, which are used in the implementation of these procedures.

A data source analysis and a data warehouse model are described and build based on a practical example, after which the data integration procedures are executed and explained in deatail for each mentioned product. In order to achieve a reliable comparison, all ETL processes are executed under the same conditions and database technologies. In the end, results are compared and presented based on chosen criteria, analysis and observations.

VSEBINA

UPORABLJENE KRATICE

ETL – Extract, Transform & Load

PDI – Pentaho Data Integration

ODI – Oracle Data Integrator

SSIS – SQL Server Integration Services

BIDS – Business Intelligence Development Studio

SQL – Structured Querry Language

BI – Business Intelligence

MySQL – Tehnologija podatkovnih baz

KM – Knowledge Module

LKM – Loading Knowledge Module

IKM – Integration Knowledge Module

CKM – Check Knowledge Module

DSN – Data Source Name

NoSQL – Tehnologija podatkovnih baz

ODBC – Open Database Connectivity

JVM – Java Virtual Machine

OS – Operating System

JDBC – Java Database Connectivity

OLE DB - Object Linking and Embedding Database

ADO – ActiveX Data Objects

RDBMS – Relational Database Management System

OLTP – Online Transaction Processing

GUI – Graphical User Interface

1 UVOD

Ob ustvarjanju oziroma polnjenju podatkovnega skladišča s podatki se lahko pojavi veliko praktičnih problemov, ki zahtevajo posebne postopke s katerimi določimo kateri podatki se bodo prenašali, kako se bodo oblikovali ter kako jih bomo povezali v primeru, da prejemamo podatke iz večih virov. Ta korak je bistvenega pomena in zato tukaj nastopi pomemben proces, ki ga imenujemo ETL oziroma Zajemi, Pretvori, in Naloži (ang. Extract, Transform & Load). Omenjen postopek nam omogoča pretvorbo podatkov na standardiziran in učinkovit način.

Na trgu je veliko orodij, ki se lotevajo tega problema, zato za določeno podjetje vsa orodja ponavadi niso primerna. ETL procesi lahko postanejo precej kompleksni in se lahko ob nepravilni zasnovi pojavijo večji operacijski problemi. Zato je izbira in implementacija samih ETL procesov pomemben del življenjskega kroga vsakega podatkovnega skladišča.

Namen raziskave je ugotoviti prednosti in slabosti posameznih orodij v obsegu izvedenega primera, saj je izbira pravega sistema za integracijo podatkov, ključnega pomena za vsako podjetje, ki želi izvajati pomembne poslovne odločitve na podlagi podatkov, ki so jim na voljo.

Diplomska naloga se osredotoča na področje integracije podatkov in tri orodja, s katerimi te integracije izvajamo. Ta orodja so Oracle Data Integrator, SQL Server Integration Services in Pentaho Data Integration. Omenjena orodja so opisana, kot je tudi izveden in opisan primer, ki služi kot iztočnica za primerjavo orodij.

Ker je orodij več in ker obsegajo širok spekter funkcionalnosti, diplomska naloga ne bo obsegala vseh podrobnosti posameznih orodij, temveč se bo omejila na primer, ki se bo izvedel za vsa orodja in osredotočila na področja, ki so zanjo bistvena. V nalogi se bodo uporabljale podatkovne baze tipa MySQL (izvirne kot tudi ciljne), zaradi česar bodo ugotovitve temeljile predvsem na omenjeni tehnologiji. Končne rezultate bo mogoče preveriti v testnih okoljih samih orodij.

V nadaljevanju je najprej razložen problem integracije in razpršenosti podatkov, ter pomen postopka ETL in podrobnejši opis njegovih korakov. Opisano je tudi delovanje in vloga podatkovnih skladišč v postopkih integracije, kot tudi njihove arhitekture.

Tretje poglavje se osredotoča na predstavitev izbranih orodij za integracijo in njihovih glavnih lastnosti, ter programov, ki so se uporabljali za ostale funkcije, katere orodja za integracijo ne obsegajo. Sledi analiza izvornih podatkov, na podlagi katere je nato oblikovan in opisan model ciljnega podatkovnega skladišča oz. njegovih dimenzij in tabele dejstev.

V četrtem poglavju je na kratko razložen način delovanja okolij orodij ETL ter podrobno opisan in prikazan postopek same integracije za vsako orodje posebej. Opisane so uporabljene komponente oz. načini, s katerimi smo zajeli potrebne funkcionalnosti za uspešno izvedbo integracije. V poglavju so prav tako razvidni različni principi delovanja omenjenih produktov.

Peto poglavje obsega analizo in primerjavo orodij po merilih, ki so se izbrala na podlagi obstoječih norm testiranja programov, ter obsega izvedenega praktičnega primera. Kriteriji zajemajo uporabniško izkušnjo, kot tudi samo funkcionalnost produktov. Na koncu poglavja še je predstavljen povzeteh splošnih ugotovitev.

.

2 PROBLEM INTEGRACIJE PODATKOV

Digitalni svet podatkov je predstavitev fizičnega sveta, zato je za podjetja zelo pomembno, da so s temi sredstvi sposobna učinkovitega upravljanja s pomočjo različnih informacijskih sistemov. To posledično ustvarja veliko tekmovalnost in iskanje različnih priložnosti. V tem okolju postaja integracija obstoječih informacijskih sistemov vedno bolj nepogrešljiva, z namenom ohranjanja konkurenčnosti in doseganja mnogih zahtev.

V osnovi integracija večih informacijskih sistemov cilja na združevanje izbranih sistemov tako, da lahko oblikuje novo celoto in ponudi uporabnikom iluzijo interakcije z enim informacijskim sistemom. Glavna razloga za integracijo sta dva [10]:

1. Za nek obstoječi informacijski sistem se lahko ustvari integriran pogled, da se olajša dostop do informacij in omogoči ponovna uporaba skozi eno točko dostopa.

2. Ob potrebi po informaciji, so podatki iz različnih komplementarnih informacijskih sistemov združeni, da dobijo celovitejšo osnovo za zadovoljitev potreb.

Obstaja ogromno aplikacij, ki podpirajo integracijo podatkov. Na področju poslovne inteligence (ang. Business intelligence - BI) se integrirana informacija uporablja za poizvedovanja in poročanja o poslovnih dejavnostih, statistične analize, spletne analitične procese in podatkovno rudarjenje, z namenom predvidevanja, odločanja, načrtovanja ter nasplošno doseganja trajne konkurenčne prednosti.

Ko se podjetja odločajo o različnih ukrepih, temeljijo svoje odločitve na podatkih, ki so jim na voljo. Ti podatki se nahajajo v t. i. podatkovnih skladiščih, nad katerimi se izvajajo poizvedbe in analize. Vendar je najprej treba te podatke pridobiti in v ta podatkovna skladišča naložiti. Problemi nastanejo, ko so obstoječi podatki neprimerni za določen informacijski sistem. Razlogov je lahko več: nekonsistentnost, zastarelost, nepopolnost, napačnost ipd. Takšne podatke imenujemo *umazani podatki* (ang. dirty data), nad katerimi je v večini primerov nemogoče izvajati podrobnejših analiz z namenom pridobivanja koristnih informacij. Da bi jih lahko uporabili, jih je treba najprej spremeniti in urediti na način, ki je za nas primeren. Ta korak je izjemno pomemben, saj določa, kateri podatki bodo dostopni v podatkovnem skladišču. Omenjen proces se imenuje Zajemi, Pretvori in Naloži (ang. ETL - Extract, Transform & Load) [8].

2.1 Postopek zajemi, pretvori in naloži

Čeprav je gradnja postopka zajemanja, pretvarjanja in nalaganja (ang. Extract, Transform & Load - ETL) *neviden* postopek končnega uporabnika, lahko trdimo, da je 70% vseh sredstev, potrebnih za implementacijo in vzdrževanje tipičnega podatkovnega skladišča. ETL doda podatkom vrednost in je veliko več, kot samo zbiranje podatkov iz izvirnih sistemov v podatkovna skladišča. V procesu se izvajajo funkcije, ki iz izvornih sistemov preoblikujejo podatke v nam uporabne informacije. Če izvorni podatki niso pravilno izvlečeni, prečiščeni in naloženi na pravi način, je tudi vzpostavitev podatkovnega skladišča praktično nemogoča. Specifično ETL [12]:

- Odstrani napake in popravi manjkajoče podatke
- Priskrbi dokumentirane ukrepe zaupanja v podatke
- Zajame pretok transakcijskih podatkov za varno hranjenje
- Prilagodi podatke iz večih virov za skupno uporabo
- Strukturira podatke za uporabo v orodjih oziroma aplikacijah končnih uporabnikov

ETL je preprost in obenem zapleten postopek. Osnovni namen tega procesa je pridobiti podatke iz obstoječih virov in jih prenesti v podatkovno skladišče. Ker je uporabnikov, ki vidijo potrebo po urejenih in preoblikovanih podatkih vedno več, je zato razumljivo, da se naslednji korak ETL sistema razdeli v stotine manjših, odvisno od sistema, ki ga gradimo in virov ter ciljnih baz iz katerih zajemamo oziroma zapisujemo podatke. Celotni postopek ETL lahko v grobem združimo v 3 korake [13]:

Zajem podatkov (ang. Extract)

Prvi del ETL procesa vključuje pridobitev podatkov iz določenih obstoječih sistemov. Najpogosteje je to najtežji korak tega procesa, saj določa podlago za vse nadaljnje faze. Večina podatkovnih skladišč vsebuje podatke iz različnih virov in vsak vir lahko hrani podatke v drugačni obliki in formatu. Velikokrat so to relacijske baze in datoteke, vendar lahko vsebujejo tudi drugačne podatkovne strukture. V osnovi je cilj tega koraka spremeniti obliko teh podatkov v format, ki je primeren za transformacijsko procesiranje. Določen del ekstrakcije vključuje razčlenjevanje teh podatkov, čemur sledi preverjanje, če so podatki primerni in dosegajo pričakovano strukturo. V nasprotnem primeru, se lahko podatki tudi delno ali popolnoma zavrnejo.

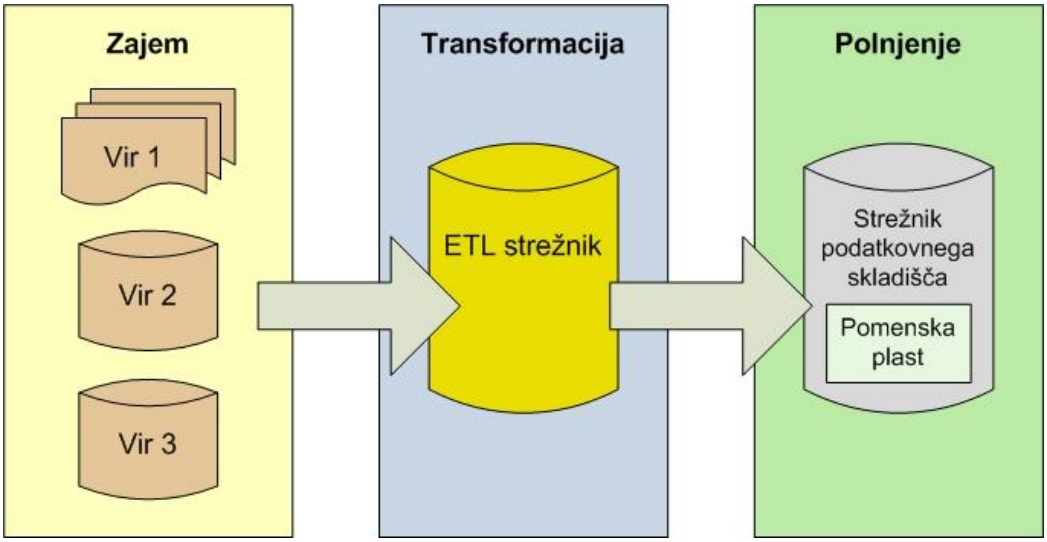
Pretvorba podatkov (ang. Transform)

Korak transformacije doda podatkom vrednost. Medtem ko prva dva koraka samo premikata oziroma restrukturirata podatke, čiščenje in usklajevanje dejansko spremeni podatke in nam pove, ali se podatki lahko uporabijo za njihov predvideni namen. V tej fazi se nanašamo na serijo pravil ali funkcij nad zajetimi podatki, da pripravimo podatke za vnos v podatkovno skladišče. Nekateri podatki ne potrebujejo nič ali pa zelo malo manipulacije. V drugih primerih pa se izvedejo nekatere od naslednjih transformacij, z namenom doseganja poslovnih in tehničnih potreb končne baze:

- Izbira samo nekaterih stolpcev za prenos (ali izbira "NULL" stolpcev za izločitev)
- Prevod kodiranih vrednosti (pri spolu bi črka M pomenila moški spol)
- Kodiranje vrednosti (npr. "moški" v "1")
- Sortiranje
- Združevanje podatkov iz večih virov
- Agregacija (povzemanje večih vrstic podatkov, npr. skupna prodaja za vsako trgovino)
- Generiranje vrednosti nadomestnih ključev
- Pivotiranje (spreminjanje stolpcev v vrstice ali obratno)
- Deljenje enega stolpca v več stolpcev
- Disagregacija ponavljajočih se stolpcev v ločeno tabelo
- Izvajanje kakršnekoli preproste ali kompleksne podatkovne validacije

Nalaganje podatkov (ang. Load)

Zadnja faza naloži spremenjene podatke v končno bazo, ponavadi v podatkovno skladišče. Od organizacije do organizacije se ta proces zelo razlikuje in je lahko zelo preprost ali zelo zapleten. V osnovi korak deluje kot most med delovnim območjem, ki ga izvaja izbrano ETL orodje, in platformo podatkovnega skladišča. Load naloži prilagojene podatke in izvede potrebne vnose, posodobitve in odstranitve v ciljni podatkovni bazi.



Slika 1: Postopek ETL

2.2 Podatkovno skladišče

Podjetja sprejemajo pomembne poslovne odločitve na podlagi podatkov, ki so jim na voljo. V večini primerov so ti podatki razpršeni in nepovezljivi, zato potrebujejo sistem oziroma bazo, v katerem so združeni na način, ki omogoča izvajanje analiz iz različnih vidikov poslovanja. Ta baza se imenuje podatkovno skladišče.

V računalništvu je podatkovno skladišče vrsta podatkovne baze, namenjena poročanju in podatkovni analizi. Je osrednja shramba podatkov, ki se ustvari s pomočjo integracije podatkov iz enega ali več virov. Podatkovna skladišča hranijo trenutne kot tudi zgodovinske podatke, ki se uporabljajo za izdelavo različnih poročil, kot so npr. letne in četrtletne primerjave. Karakteristike, ki ločijo podatkovna skladišča od ostalih operacijskih okolij so: organizacija podatkov na način, ki združi pomembne podatke z namenom lažjega dostopa, več kopij nekega podatka iz različnih obdobij časa je skupaj združenih, in ko podatek vnesemo v bazo, se ne posodobi. Namesto tega se zgodovinski podatki shranjeni v podatkovnem skladišču periodično osvežijo s podatki iz operacijskih podatkovnih baz. Glavne lastnosti lahko razdelimo na [7]:

- integriranost (v podatkovnem skladišču se hranijo podatki iz različnih virov)
- zgodovina (hranimo podatke za več let nazaj)
- področna usmerjenost (podatki predstavljajo pomembnejša področja poslovanja)
- statičnost (sistem podatkovnega skladišča ni namenjen aktivnim transakcijam temveč poizvedovanju)

V osnovi poznamo 3 vrste arhitektur podatkovnih skladišč [22]:

- **Centralizirana** arhitektura je sestavljena iz osrednjega podatkovnega skladišča, ki hrani množico področnih skladišč. Pri tem se področna skladišča polnijo izključno iz centralnega. Uporabljena metodologija izgradnje sistema pa ne priporoča izpeljave intervjujev kot načina za pridobivanje zahtev uporabnikov o podatkovnem skladišču.

- **Distribuirana** arhitektura vsebuje množico samostojnih področnih skladišč, ki so med seboj povezane in oblikujejo podatkovno skladišče. Skupno podatkovno vodilo je koncept, ki pri omenjeni arhitekturi omogoča izgradnjo samega podatkovnega skladišča. Pri tem se pristop izgradnje imenuje *od-spodaj-navzgor* (ang. bottom-up).

- **Federativna** je novejša arhitektura, katere temenj je skupni informacijski model. Cilj je zagotovitev nizkih stroškov in hitrejšo vrnitev vloženih sredstev. Pri tem se uporabljajo neodvisna področna skladišča, integracija podatkov pa kasneje ni potrebna. Arhitektura še je v fazi uveljavitve.

Ob izbiri pravilne arhitekture, ki zagotavlja optimalno delovanje in učinkovito izrabo virov je ključno poznavanje pozitivnih in negativnih lastnosti posameznih arhitektur. V primeru napačne izbire tvegamo propad podatkovnega skladišča, kar je na področju informatike relativno pogost pojav.

3 OPIS ORODIJ IN PRIPRAVE NA INTEGRACIJO

Orodja za integracijo so nepogrešljiva programska oprema, ki nam omenjen proces izredno olajša in pohitri. Čeprav je ročno kodiranje integracij z uporabo SQL skript še vedno zelo razširjen pojav, ima uporaba ETL orodij svoje prednosti. V nadaljevanju so opisana tri orodja, na katera se diplomska naloga osredotoča in s katerimi smo izvedli postopke ETL.

3.1 Microsoft SQL Server Integration Services

SQL Server Integration Services (SSIS) je sidro v trilogiji produktov, ki gradijo Microsoft SQL Server Business Intelligence (BI) platformo, ki se uporablja za izvedbo različnih nalog za migracijo podatkov. Je platforma za integracijo podatkov in potek dela aplikacij. Vsebuje hitro in fleksibilno orodje, ki se uporablja za izvajanje ETL postopkov.

SSIS skupaj z Analysis Services in Reporting Services oblikujejo sistem, ki Microsoft postavlja na zemljevid področja poslovne inteligence. V najpreprostejši obliki lahko na SSIS gledamo kot na ETL orodje, vendar se lahko uporablja tudi za druge namene, kot so npr. avtomatizirano vzdrževanje podatkovnih baz SQL Server, posodabljanje multidimenzionalnih podatkovnih kock, ali pošiljanje elektronskih sporočil z informacijami o statusu operacij definiranih od uporabnika. Z uporabo *vleci-in-spusti* (ang. drag-and-drop) upravljanja, lahko ETL razvijalci sestavljajo zapletene procese in hitra podatkovna čiščenja, ki konkurirajo znanim orodjem, kot tudi orodjem po meri [2].

Tipična uporaba SSIS zajema:

- Združevanje podatkov iz heterogenih podatkovnih shramb
- Polnjenje podatkovnih skladišč in informacijskih tržnic
- Čiščenje in standardizacijo podatkov
- Grajenje poslovne inteligence v procese podatkovnih transformacij
- Avtomatiziranje administrativnih funkcij in nalaganja podatkov

3.2 Pentaho Data Integration

Z nenehnim večanjem vrst in količine podatkov potrebujejo organizacije hitre in enostavne načine zajemanja in izkoriščanja teh podatkov. Vendar največji izziv, ki danes bremeni IT organizacije, je zagotavljanje konsistentne in eno-verzijske rešitve čez vse vire informacij v analitično-pripravljenem formatu.

Pentaho Data Integration (PDI), imenovan tudi kot Kettle, vsebuje pogon za integracijo podatkov in grafične uporabniške vmesnike (Graphical User Interface - GUI), ki omogočajo uporabnikom definiranje postopkov teh integracij in transformacij. Podpira delovanje na posameznih računalnikih, kot tudi na oblaku in kopici. Z močnimi ETL lastnostmi, intuitivnim in bogatim grafičnim okoljem in odprto ter standardizirano arhitekturo, postaja PDI vsebolj priljubljena izbira na tem področju. Omogoča popolno ETL upravljanje, vključno z [24]:

- Preprosto grafično obliko.
- Široko povezovalnostjo za vse tipe podatkov, vključno z raznolikimi in velikimi količinami podatkov.
- Skalabilnostjo in dobro zmogljivostjo.
- Integracijami ogromnih količin podatkov, analizami in poročanjem, vključno s Hadoop, NoSQL, tradicionalnim spletnim procesiranjem transakcij (ang. Online transtaction Processing – OLTP) in analitičnimi podatkovnimi bazami.
- Sodobno, odprto in standardizirano arhitekturo.

3.3 Orace Data Integrator

Oracle Data Integrator (ODI) ponuja celovite rešitve za grajenje, izvajanje, in upravljanje realno-časovnih podatkovnih arhitektur za okolja SOA (ang. Service-Oriented Arhitecture), poslovne inteligence (Business Intelligence - BI) in podatkovna skladišča. Dodatno združuje vse elemente podatkovne integracije: realno-časovni premiki podatkov, transformacije, sinhronizacije, kvaliteta podatkov, upravljanje podatkov in podatkovne storitve z namenom zagotavljanja pravočasnosti, točnosti in konsistentnosti informacij na kompleksnih sistemih.

Je zelo razširjen produkt, ki ponuja nove pristope k definiranju podatkovnih transformacij in procesov integracije, zaradi česar posledično sledi hitrejši in enostavnejši razvoj ter vzdrževanje. Osnovan na edinstveni ETL arhitekturi, ODI ne zagotavlja samo visoke zmogljivosti izvajanja integracij podatkov in procesov validacije, ampak je tudi stroškovno zelo učinkovita rešitev.

Oracle Data Integrator izvaja močan deklarativni zasnovni pristop podatkovni integraciji, ki loči deklarativna pravila od podrobnosti implementacij. Temelji na edinstveni ETL arhitekturi, ki odpravi potrebo po samostojnem ETL strežniku in lastniškem pogonu ter namesto tega izkorišča povezane lastnosti pogonov upravljalnikov sistema za relacijske podatkovne baze (Relational Database Management System - RDBMS). Ta kombinacija zagotavlja dobro produktivnost za razvoj, kot tudi za vzdrževanje in visoko zmogljivost za izvajanje podatkovnih transformacij in procesov validacij [4].

ODI uporablja tudi arhitekturo *Zajemi, Naloži in Pretvori* (ang. Extract, Load & Transform - E-LT), ki odstrani potrebo po dodatnem ETL strežniku med izvorno in ciljno bazo. Sistem izkorišča izvorne in ciljne strežnike za izvajanje kompleksnih transformacij, večina katerih se izvaja v stanju, ko strežnik ni zaposlen s poizvedbami končnih uporabnikov. Rezultat je izboljšana zmogljivost in razširljivost v primerjavi s tradicionalnimi ETL produkti [20].

3.4 Ostala uporabljena orodja

Ob načrtovanju in izvajanju integracij smo si pomagali še z nekaterimi drugimi orodji, ki so nam služila pri opravilih, ki jih integracijska orodja praviloma ne obsegajo:

- **DBDesigner** 4 je sistem za oblikovanje virtualnih baz, ki integrira modeliranje, ustvarjanje, oblikovanje in vzdrževanje v eno okolje. Povezuje strokovne lastnosti, kot tudi enostaven in čist vmesnik za dosego učinkovitega načina za upravljanje nad podatkovnimi bazami.

- **XAMPP** je brezplačen in odprtokodni paket, ki zajema več prednastavljenih spletnih strežnikov. Sestavljen je iz Apache HTTP strežnika, MySQL podatkovne baze in prevajalcev za skripte, napisane v jezikih PHP in Pearl.

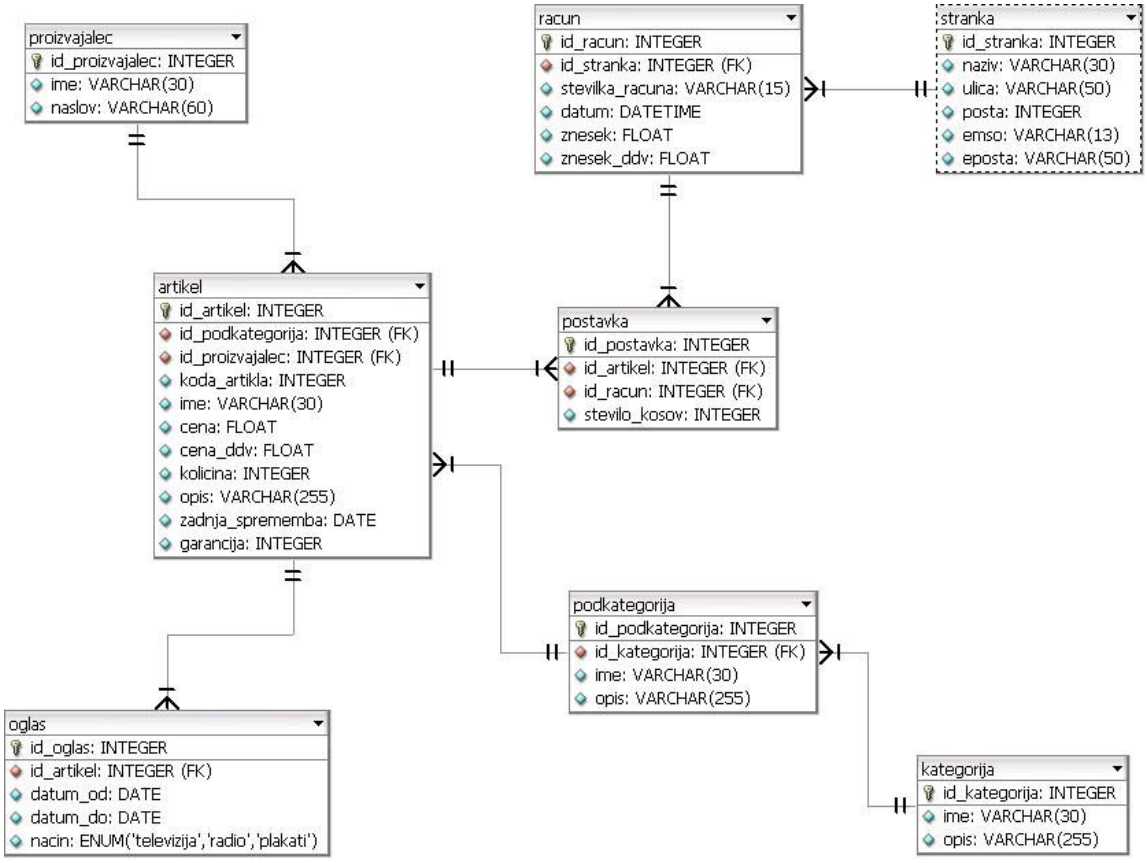
- **MySQL Querry Browser** je grafično orodje namenjeno ustvarjanju, izvajanju in optimizaciji poizvedb v grafičnem okolju. Čeprav lahko vse poizvedbe izvajane v tem orodju zaženemo tudi z ukazno vrstico v okolju »mysql«, nam MySQL Querry Browser omogoča iskanje in urejanje podatkov na hitrejši in enostavnejši način.

3.5 Analiza izvornih podatkov

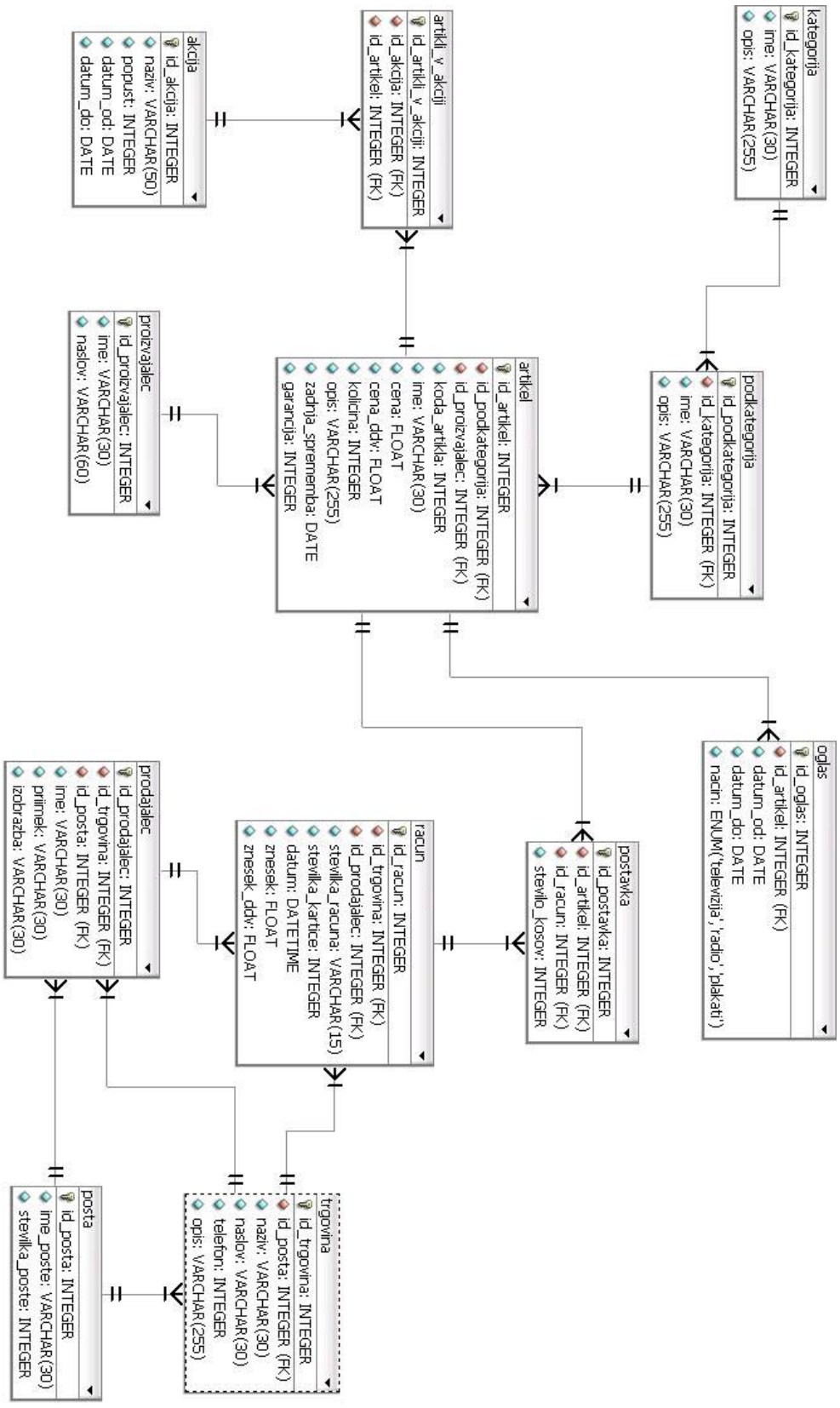
Pred začetkom načrtovanja integracij, je treba podrobno preučiti vire, ki so nam na razpolago in iz katerih bomo črpali vse potrebne podatke za našo ciljno podatkovno skladišče. Korak je ključnega pomena, saj lahko z dobro analizo izvemo, kakšne poizvedbe bo po integraciji mogoče izvajati in kako bo podatkovno skladišče sestavljeno oziroma oblikovano.

Za diplomsko nalogo in testiranje integracijskih orodij smo uporabili podatke iz treh ločenih virov. Prvi je podatkovna baza trgovine za športno opremo tipa MySQL. Drugi je podatkovna baza spletnega dela trgovine, katere podatki segajo od leta 2007 naprej, tretji vir pa je datoteka »KarticeZvestobe« tipa Excel, ki vsebuje podatke o vseh strankah, ki uporabljajo kartico zvestobe. Datoteka vsebuje naslednje podatke: številko kartice, ime, priimek, kraj, ulico, pošto, datum rojstva in davčno številko.

Cilj naloge je združiti podatke na način, ki nam bo omogočal poročanje in analiziranje podatkov iz različnih vidikov: koliko se je prodalo v posameznem mesecu prejšnjega leta, katera je najboljša stranka, kateri izdelek je najslabše prodajan ipd.



Slika 2: E-R diagram spletnega dela trgovine



Slika 3: E-R diagram trgovine

3.6 Podatkovno skladišče – dimenzijski model

Na področju informatike in transakcijskih podatkovnih baz je pogost problem razpršenost podatkov. Nad podatki v takšnem stanju je izvajanje zapletenih poizvedb precej oteženo oziroma nemogoče. Zato je treba podatke združiti in oblikovati v takšno obliko, ki nam bo omogočala hitro in učinkovito poizvedovanje in vračanje informacij, ki so za nas uporabne. Rešitev je izgradnja podatkovnega skladišča.

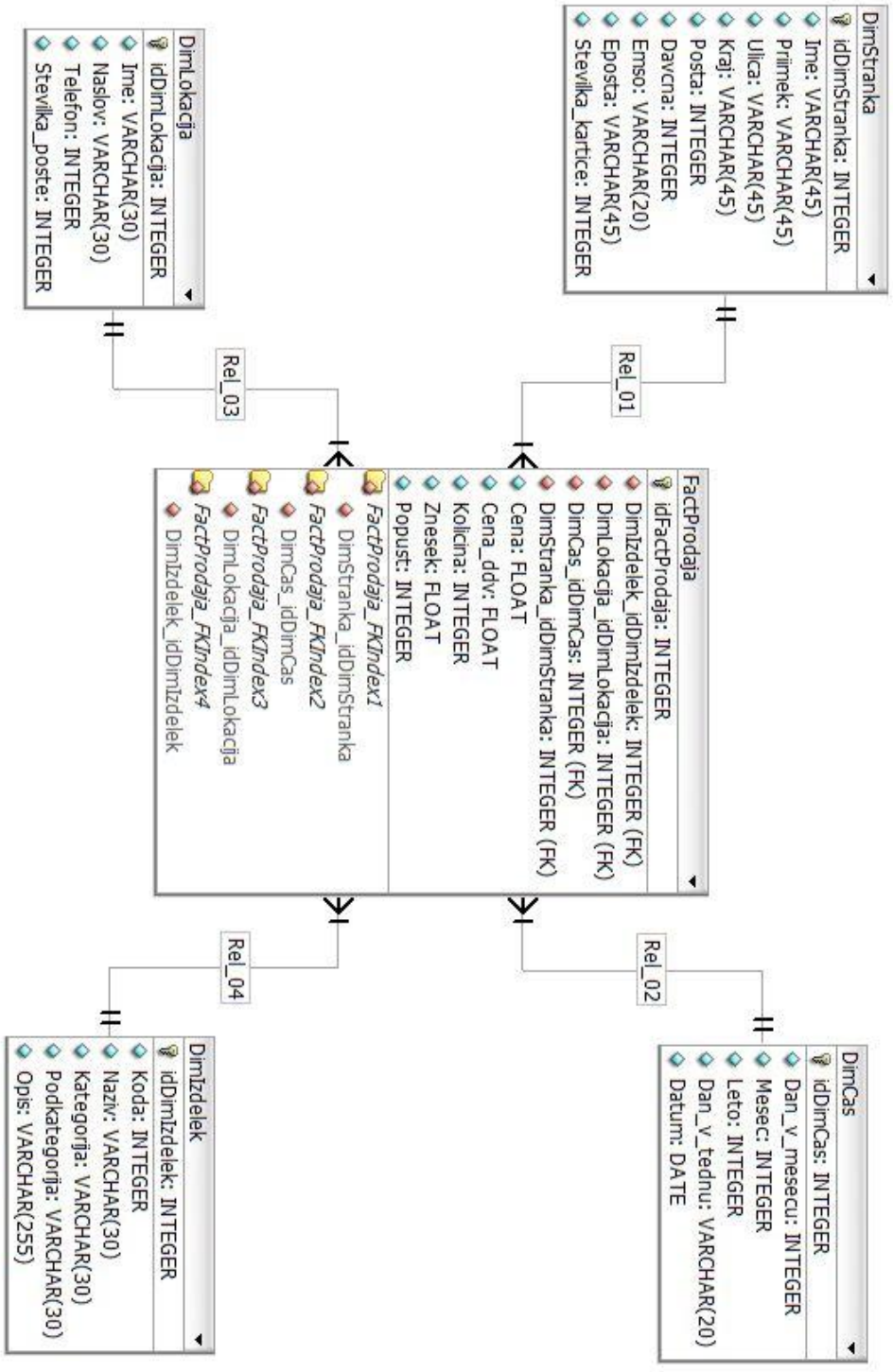
V našem primeru smo s programom DBDesigner 4 za trgovino oblikovali in zgradili podatkovno skladišče, ki bo omogočalo lažji pregled nad preteklimi transakcijami računov oziroma prodaje, ki so se izvajale v prodajalnah. Sestavljeno je v obliki zvezdnate sheme iz štirih dimenzij in ene tabele dejstev imenovane »Prodaja«, ki beleži naslednje podatke:

- »Cena« (tipa Float): beleži ceno posameznih artiklov
- »Cena_ddv« (tipa Float): beleži ddv posameznih artiklov
- »Kolicina« (tipa Integer): beleži koliko artiklov vsebuje transakcija
- »Znesek« (tipa Float): beleži skupni znesek transakcije
- »Popust« (tipa Integer): beleži popust na artikel

Dimenzije obsegajo 4 različne skupine:

- »DimStranka« hrani podatke strank (če obstajajo) za vsako izvedeno transakcijo
- »DimCas« hrani podatke časa oz. kdaj se je posamezna transakcija izvedla
- »DimLokacija« hrani podatke o trgovinah in lokacijah transakcij
- »DimIzdelek« pa hrani podatke o kupljenih izdelkih

Vsaka dimenzija ima vrsto atributov, ki podrobneje opisujejo določeno področje. Točen izgled diagrama in atributov dimenzij lahko vidimo na sliki 4.



Slika 4: Diagram skladišča podatkov Prodaja

4 IZGRADNJA IN IZVEDBA ETL PROCESOV

V nadaljevanju je opisan proces izvedbe postopkov ETL za vsako orodje posebej. V nekaterih primerih so določeni koraki podobni korakom, ki so že bili izvedeni na enak ali pa zelo podoben način, zato je njihov opis ponekod skrajšan. Pri vseh orodjih smo poskušali doseči enake rezultate z namenom lažje in točne primerjave glede na izbrana merila.

4.1 ETL s Pentaho Data Integration

Ob zagonu orodja Pentaho Data Integration (PDI) se postopek integracije začne z ustvaritvijo novega okna za transformacije, ki služi kot območje za načrt samega ETL postopka od vira podatkov do končnega podatkovnega skladišča. Pri tem si pomagamo z različnimi komponentami, ki opravljajo določene naloge. Za lažje iskanje so te razdeljene v skupine glede na tip dela, ki ga opravljajo. V nadaljevanju sledi nekaj primerov komponent za integracijo:

	Komponenta *Table exists* se uporablja za preverjanje obstoja določene tabele v podani podatkovni bazi. Tip rezultata tega koraka je boolean flag.
	*Access input* omogoča direktno branje iz datotek tipa Microsoft Access »MDB«. Komponenti določimo pot do datoteke, navedemo iskano tabelo ter atribute, ki jih želimo prenesti v tok podatkov.
	Komponenta *Clone row* ustvari kopije (klonira) vrstic in jih nato neposredno posreduje naprej takoj za izvirno vrstico do naslednjih korakov.
	Korak *Table output* nam omogoča vnos podatkov v ciljno podatkovno bazo. Ustvarimo povezavo na želeno podatkovno bazo, kot tudi tabelo.
	Če je vrednost nekega atributa enaka poljubni navedeni vrednosti, nam *Null if* vrednost tega atributa spremeni v »null« (prazna vrednost). Če želimo, lahko vse vhodne vrstice dodamo z ukazom *Get fields*.
	Select values je uporaben za izbiranje, preimenovanje, spreminjanje podatkovnih tipov in določanje dolžine in natančnosti atributov v podatkovnem toku.
	Komponenta *Add constants* je preprost in učinkovit način za dodajanje konstantnih vrednosti v tok podatkov.

Tabela 1: Primeri komponent orodja PDI

Točen potek integracije določamo s povezovanjem komponent. Pri tem pazimo na vrstni red, saj se podatki v toku nenehno spreminjajo, dodajajo in brišejo.

4.1.1 Polnjenje dimenzije DimCas

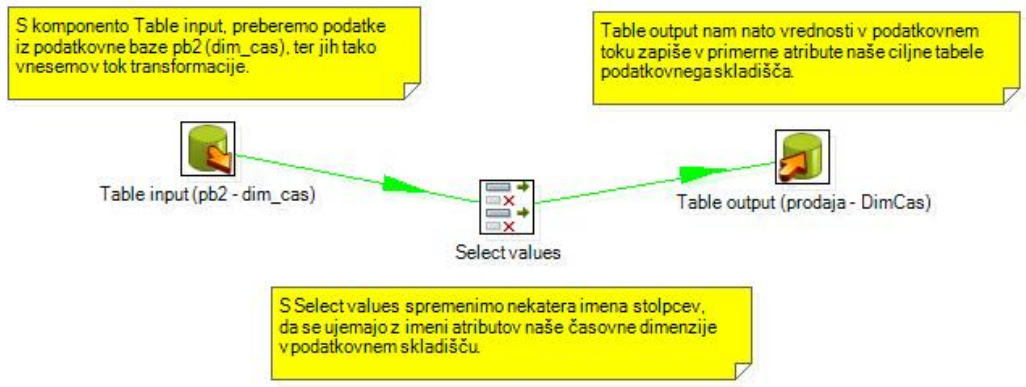
Prva dimenzija, ki jo napolnimo, se imenuje »DimCas«, ki se nahaja v naši podatkovni bazi »Prodaja«. Za to integracijo potrebujemo tri komponente – *Table input*, *Select values* in *Table output*. S *Table input* pridobimo atribute časa, ki jih dobimo iz tabele »dim_cas« v podatkovni bazi »pb2«. Komponenti določimo povezavo do strežnika in baze. Ker je baza tipa MySQL, moramo vnesti tudi uporabniško ime in geslo, s katerim dobimo pravice za dostop. Po uspešni povezavi lahko nato izberemo specifične podatke, ki jih želimo zajeti in vnesti v tok podatkov. To storimo z uporabo SQL stavka:

```
SELECT
  id_cas
, leto
, mesec
, dan_v_mesecu
, dan_v_tednu
, datum
FROM dim_cas;
```

Poizvedba 1: Zajemanje podatkov za dimenzijo DimCas

Tem atributom nato spremenimo imena, da se bodo ujemala imenom v naši dimenziji. Uporabimo komponento *Select values*. Od vseh imen je treba spremeniti samo ime primarnega ključa »id _cas« v »idDimCas«. Obenem spremenimo tudi tip atributa »datum«, saj je datum v naši dimenziji tipa *DATE*, v tabeli »dim_cas« pa tipa *DATETIME*. V nasprotnem primeru bi PDI ob zagonu procesa javil napako.

Sledi še tretja komponenta, ki vse urejene podatke zapiše v našo dimenzijo. V *Table output* določimo povezavo do ciljnega strežnika, podatkovne baze in dimenzije. Poljubno lahko komponenti spremenimo tudi ime. Ob uspešni povezavi je načrtovanje integracije za dimenzijo »DimCas« končan (Slika 5). Sledi še zagon samega postopka za integracijo.



Slika 5: Podatkovni tok za dimenzijo DimCas

V primeru neuspešne integracije, se pod rubriko *Execution Results -> Logging* izpiše vzrok in na katerem mestu je prišlo do napake. Ko napake odpravimo, lahko postopek ponovno zaženemo. V primeru uspešne integracije, pa lahko rezultat preverimo s poizvedbo na dimenziji »DimCas«.

V skladišče bilo zapisanih 2189 vrstic v času 1,4 sekunde.

4.1.2 Polnjenje dimenzije DimLokacija

Ustvarili smo novo transformacijo. Dimenzija »DimLokacija« vsebuje podatke o krajih, kjer so se izvajale transakcije, zato potrebujemo podatke iz tabel »Trgovina« in »Posta« podatkovne baze trgovine. V transformacijo smo prenesli komponento *Table Input* in ji določili povezavo do izvirnih tabel baze »pb2« ter vnesli poizvedbo, ki vrne primarni ključ, naziv, naslov, telefon in številko pošte, ter jih s tem posredovali v podatkovni tok:

```
SELECT
  id_trgovina, t.naziv, t.naslov, t.telefon, p.stevilka_poste
FROM trgovina t, posta p
WHERE t.id_posta = p.id_posta;
```

Poizvedba 2: Zajemanje podatkov iz tabel Trgovina in Posta

V naši ciljni tabeli sta dva stolpca drugače imenovana, zato smo uporabili *Select Values* ter pod rubriko *Select & Alter* preimenovali stolpca »id_trgovina« in »naziv« v »idDimLokacija« in »ime«. Za vnost podatkov v podatkovno skladišče smo nato uporabili korak *Table Output*, ji določili povezavo do podatkovne baze »Prodaja« in njene tabele »DimLokacija«, ter nato povezali vse tri komponente.

Zagon transformacije je vrnil 5 vrstic v skupnem času 0.2 sekunde.

4.1.3 Polnjenje dimenzije DimIzdelek

Polnjenje dimenzije »DimIzdelek« poteka podobno kot pri dimenziji »DimLokacija«. Potrebovali smo *Table Input*, *Select values* ter *Table Output*. Tokrat smo brali podatke iz tabel »Artikel«, »Kategorija« in »Podkategorija«, kar smo dosegli z naslednjim SQL ukazom v komponenti *Table Input*:

```
SELECT a.id_artikel, a.ime, k.ime AS kateg, pk.ime AS podkateg,
a.opis
FROM artikel a, kategorija k, podkategorija pk
WHERE a.id_podkategorija = pk.id_podkategorija
AND pk.id_kategorija = k.id_kategorija;
```

Poizvedba 3: Branje podatkov artikla

V komponenti *Select Values* smo nato preimenovali večino dobljenih atributov:

Ime vhodnega atributa	Ime izhodnega atributa
id_artikel	koda
ime	naziv
kateg	kategorija
podkateg	podkategorija

Tabela 2: Prirejanje imen atributov artikla

Atribut »opis« je enak ciljnemu, zato ostane nespremenjen. *Table Output*-u smo nato določili bazo in tabelo »DimIzdelek« v katero se bodo zapisali podatki. Ob zagonu integracije se je prebralo in zapisalo 267 vrstic v 0,7 sekunde.

4.1.4 Polnjenje dimenzije DimStranka

To dimenzijo je bilo treba napolniti iz dveh različnih virov. Prvi je Excel datoteka »KarticeZvestobe.xls«. Ker *Table Input* ni namenjen branju datotek na lokalnem disku, smo za to nalogo potrebovali novo komponento imenovano *Excel Input*.

Korak *Excel Input* nam omogoča branje podatkov iz ene ali več Excel in OpenOffice datotek. Ponuja vrsto uporabnih funkcij, ki omogočajo lažji nadzor nad načinom branja in zajemanjem vrstic iz tabel oziroma delovnih listov. Te možnosti se nahajajo pod rubriko *Content*:

- Header: Označimo, če želimo ispustiti prvo vrstico tabele. V prvi vrstici so praviloma imena stolpcev.

- No empty rows: Označimo, če želimo izpustiti prazne vrstice.
- Stop on empty row: Konča branje datoteke, če naleti na prazno vrstico.
- Filename field: Ustvari imenovano polje z imenom datoteke ter ga posreduje na izhod koraka.
- Sheetname field: Ustvari imenovano polje z imenom delovnega lista ter ga posreduje na izhod koraka.
- Limit: Omeji število vrstic na vneseno število (0 pomeni vse vrstice).

Pred zajemom podatkov smo komponenti določili fizično pot do datoteke. Ker je datoteka imela več strani, smo pod opcijo sheets poskrbeli, da je vključena samo prva delovna stran »Sheet1«. Ob uspešnem preverjanju datoteke so se nato pod opcijo *Fields* izpisali vsi vključeni stolpci, ki se bodo prenesli v tok podatkov. Pred tem lahko preverimo njihove lastnosti kot so ime, tip, dolžina, natančnost ipd.

V dodani komponenti Select Values so se nato določila ciljna imena:

Ime vhodnega atributa	Ime izhodnega atributa
Številka kartice	Stevilka_kartice
Pošta	Posta
Davčna številka	Davcna

Tabela 3: Prirejanje atributov strank

Imena »Ime«, »Priimek«, »Kraj« in »Ulica« ostanejo enaka. Podobno kot prej, se še je dodal korak *Table Output*, ki se je povezal z ciljno dimenzijo »DimStranka«, nakar se je izvedla sama integracija. Prebralo in vneslo se je 466 vrstic v času 1,4 sekunde.

Sledil je drugi del vnosa podatkov v dimenzijo »DimStranka«, ki je zahteval branje iz podatkovne baze spletne trgovine. V *Table Input* se je tokrat vnesel naslov spletnega dela trgovine. Brali smo s tabele »stranka«, pri čemer smo uporabili SQL poizvedbo:

```
SELECT
  naziv
, ulica
, posta
, kraj
, emso
, eposta
FROM stranka;
```

Poizvedba 4: Zajemanje atributov iz tabele Stranka

S komponento *Select Values* je bilo treba spremeniti samo ime atributa »naziv« v »ime«, nato je sledil korak za vnos podatkov. Ker so v dimenziji »DimStranka« že bili dodani podatki, smo tokrat morali uporabiti nov način vnosa podatkov, saj *Table Output* ne omogoča posodabljanja že obstoječih vrstic. Za takšne primere se uporabi komponenta *Insert/Update*.

Omenjen korak najprej poišče vrstico v ciljni tabeli, ki se ujema z iskanim ključem. Če vrstice ne najde, vnese podatke v novo vrstico. V naprotnem primeru preveri vsa njena polja. Če se nove vrednosti razlikujejo od starih, vrstico posodobi, drugače pa se ne zgodi nič. Operacija se izvede za vse vhodne vrstice.

Ko smo komponento primerno uredili, se je zagnala integracija, ki je zapisala 104 vrstice v 0,2 sekunde.

4.1.5 Polnjenje tabele dejstev Prodaja

Polnjenje tabele dejstev se lahko začne pod pogojem, da so pred tem napolnjene vse dimenzijske tabele. Ko to naredimo, je treba zapisati SQL poizvedbo, ki bo v tok podatkov prenesla vse potrebne atribute, ki jih potrebujemo za uspešno izvedbo integracije tabele dejstev, ki bo zadovoljila potrebe uporabnika. Za našo tabelo dejstev smo uporabili naslednje podatke iz podatkovne baze trgovine:

- racun (znesek, znesek_ddv)
- postavka (stevilo_kosov)
- akcija (popust)

Atributi, ki se bodo potrebovali za primerjanje in iskanje primarnih ključev dimenzij, pa so:

- artikel (ime)
- racun (datum)
- trgovina (naziv)
- racun (stevilka kartice)

Zaradi drugih obremenitev strežnika, na katerem se nahaja podatkovna baza trgovine, kot tudi doseganja zanesljivejših časovnih rezultatov smo poizvedbo omejili na vrstice, katerih datum sega od 1.7.2006 do 5.7.2006. Ob upoštevanju omenjenih omejitev, smo v komponento *Table Input* vnesli naslednjo SQL poizvedbo:

```
SELECT a.ime AS ime_izdelka, LEFT(r.datum, 10) AS datum_nakupa,
SUM(p.stevilo_kosov) AS kolicina, SUM(r.znesek) AS znesek,
r.znesek AS cena, ak.popust AS popust, r.znesek_ddv AS cena_ddv,
t.naziv AS ime_trgovine, r.stevilka_kartice AS st_kartice
FROM artikel a, postavka p, racun r, akcija ak, artikli_v_akciji
ava, trgovina t
WHERE a.id_artikel = p.id_artikel
AND p.id_racun = r.id_racun
AND r.id_trgovina = t.id_trgovina
AND ava.id_artikel = a.id_artikel
AND ak.id_akcija = ava.id_akcija
AND r.datum BETWEEN '2006-07-01' AND '2006-07-05'
GROUP BY a.ime, LEFT(r.datum, 10);
```

Poizvedba 5: Zajemanje potrebnih atributov za polnjenje tabele dejstev

Če v tem trenutku preverimo tipe stolpcev, ki smo jih vnesli v podatkovni tok, ugotovimo, da so primerni vsi razen atributa »datum«, saj je njegov zapis drugačen od zapisa v naši dimenziji »DimCas«. V takšnem stanju ni primeren za našo integracijo, zato mu je treba tip pred nadaljevanjem spremeniti. Podobno kot pri polnjenju dimenzije »DimCas«, smo uporabili komponento *Select Values* in atributu »datum_nakupa« spremenili tip v *DATE* oziroma zapis »yyyy.mm.dd«.

Sledi korak, v katerem želimo v tok podatkov vnesti primerne primarne ključe dimenzij, da se lahko kopirajo v tabelo dejstev. To dosežemo s posebno komponento *Database Lookup*, ki primerja določen podan atribut s stolpcem druge tabele, nato pa v primeru najdene vrstice v tok podatkov vrne poljubno vrednost. Prvi iskani atribut je bil primarni ključ dimenzije »DimCas«. Vsakemu *Database Lookup-u* je najprej treba določiti povezavo do primerne dimenzije v našem podatkovnem skladišču. Z uspešno povezavo lahko pod poljema *Table field* in *Field1* določimo katera dva atributa se bosta primerjala. V našem primeru sta to »Datum« iz dimenzije »DimCas« in »datum_nakupa« iz tabele »Racun«. Treba je bilo še določiti polje, ki se bo vrnilo v podatkovni tok – v našem primeru primarni ključ »idDimCas«. S tem smo urejanje lookup koraka za iskanje prvega ključa zaključili. Sledijo trije enaki koraki za preostale ključe.

Prvi primerjalni atribut	Drigu primerjalni atribut	Vrnjen atribut
Ime	ime_trgovine	idDimLokacija
Stevilka_kartice	st_kartice	idDimStranka
Naziv	ime_izdelka	idDimIzdelek

Tabela 4: Primerjanje atributov in vračanje primarnih ključev

Ob iskanju in vnašanju primarnih in tujih ključev, je treba upoštevati izjeme, kjer lookup operacija ne vrne rezultata. V teh primerih se proces integracije neuspešno konča, saj je vnos tujih ključev v tabelo dejstev obvezen. Če smo v tem trenutku načrtovanja izvedli poizvedbo nad tabelo »Racun«, smo opazili, da so nekatere vrednosti atributa »stevilka_kartice« enake vrednosti 0. V naši dimenziji pa takšne vrednosti ni, zato je vanjo potrebno dodati eno vrstico, ki bo v takšnih primerih delovala kot privzeta vrednost za našo tabelo dejstev. Nad podatkovno bazo smo izvedli naslednji *SQL INSERT* ukaz:

```
INSERT INTO Prodaja.DimStranka
VALUES (1000, 'Neznano', 'Neznan', NULL, NULL, NULL, NULL,
NULL, NULL, 0);
```

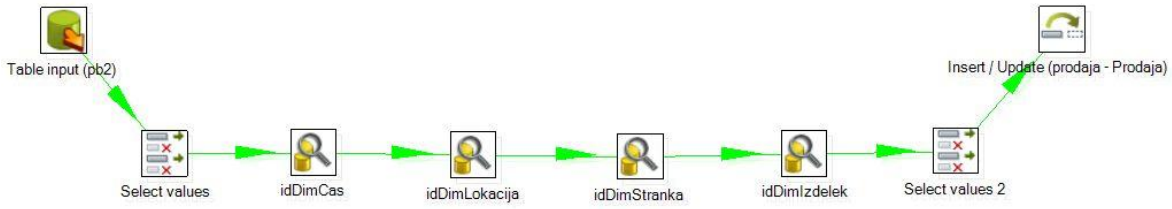
Poizvedba 6: Vnos privzetih vrednosti v dimenzijo DimStranka

Pred zagonom procesa integracije je bilo treba še spremeniti imena nekaterih atributov, da so se ujemali s stolpci tabele dejstev »Prodaja«. Transformacijo smo izvedli s *Select Values*:

Vhodno ime	Izhodno ime
idDimCas	DimCas_idDimCas
idDimLokacija	DimLokacija_idDimLokacija
idDimStranka	DimStranka_idDimStranka
idDimIzdelek	DimIzdelek_idDimIzdelek

Tabela 5: Prirejanje imen najdenih ključev

Imena stolpcev »kolicina«, »znesek«, »cena«, »popust« in »cena_ddv« so ostala nespremenjena. Za polnjenje tabele smo ponovno uporabili korak *Insert/Update* ter ga povezali s tabelo »Prodaja«.



Slika 6: Proces integracije tabele dejstev Prodaja

V tabelo dejstev se je ob zagonu integracije zapisalo 655 vrstic v času 20,3 sekunde.

4.2 ETL z Oracle Data Integrator

Pred začetkom načrtovanja integracije v orodju Oracle Data Integrator (ODI) se je treba prijaviti v delovni repozitorij. Pred tem poskrbimo, da je zagnan strežnik, na katerem se sam repozitorij nahaja. Ob uspešni prijavi se nam nato odpre okolje za razvoj integracij podatkov.

Najprej je treba ustvariti nov projekt v meniju Designer. Opazimo, da nam je ODI ustvaril projekt z različnimi komponentami:

- Mape (Folders)

Mape so komponente, s katerimi si pomagamo organizirati delo med načrtovanjem integracij. V mape se lahko dodajajo tudi podmape. Mape vsebujejo Pakete (Packages), Vmesnike (Interfaces) in Postopke (Procedures).

- Paketi (Packages)

Paket je največja enota izvajanje v orodju ODI. Paket predstavlja potek dela, sestavljenega iz sekvence korakov, organiziranih v diagram izvajanja. Paketi se sklicujejo na druge komponente projekta, kot so vmesniki, postopki ali spemenljivke.

- Vmesniki (Interfaces)

Vmesnik je pretok podatkov za večkratno uporabo. Je set deklarativnih pravil, ki opisujejo nalaganje podatkovnih baz ali drugih ciljnih struktur iz ene ali več izvirnih podatkovnih baz.

- Postopki (Procedures)

Je komponenta za večkratno uporabo, ki združuje zaporedje operacij, ki ne sodijo pod koncept vmesnika. Npr. sprejemanje e-pošte, praznjenje baz, razpakiranje datotek, ipd.

- Spremenljivke (Variables)

Vrednost spremenljivk se shranjuje v ODI. Ta vrednost se lahko spreminja med izvedbo.

- Zaporedja (Sequences)

Zaporedje predstavlja samodejno večanje vrednosti spremenljivk med uporabo. Med dvema uporabama je vrednost konstantna.

- Uporabniške Funkcije (User Functions)

Uporabniške funkcije omogočajo definiranje prilagojenih funkcij, za katere definiramo tehnologijsko-odvisne implementacije. Uporabljajo se v vmesnikih in postopkih.

- Moduli Znanja (Knowledge Modules)

Oracle Data Integrator uporablja Module Znanja (KM) na večih mestih načrtovanja projekta. KM je predloga kode povezana s podano tehnologijo, ki zagotavlja specifično funkcijo (nalaganje podatkov, povratno inženirstvo ipd.)

- Označbe (Markers)

Komponente projekta so lahko označene z namenom odražanja metodologije ali organizacije. Te označbe so organizirane v skupine in se lahko uporabljajo na večini objektov v projektu.

Za zajem podatkov (kot tudi za vnos v ciljne podatkovne baze) je najprej treba deklarirati strežnike in njihove tehnologije v ODI topologiji. To storimo v meniju *Topology -> Physical Arhitecture*. V našem primeru uporabljamo bazo MySQL, zato pod MySQL dodamo nov podatkovni strežnik. Ko strežnik poimenujemo, mu nato pod razdelkom JDBC dodelimo gonilnik tipa »com.mysql.jdbc.Driver« in pod URL vpišemo »jdbc:mysql://« ter naslov, port in ime podatkovne baze, iz katere bomo zajemali podatke. Pri tem se ne sme pozabiti tudi na vnos uporabniškega imena in gesla.

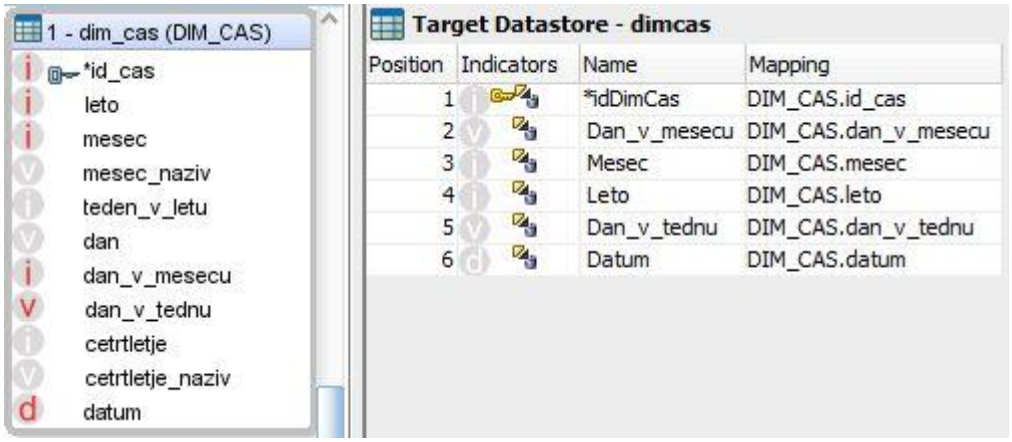
Sledi dodeljevanje fizične sheme za dodano podatkovno bazo. Ob uspešni povezavi nam ODI poda na izbiro podatkovne baze, ki so na voljo na strežniku. V našem primeru bomo jemali podatke iz baze »pb2«. Prav tako uporabimo bazo kot delovni katalog. Pod Context pa vpišemo poljubno ime logične sheme, ki se bo uporabljala globalno. Ko shemo shranimo se logična shema samodejno ustvari v meniju *Logical Arhitecture* pod tehnologijo MySQL.

Postopek ponovimo za vse ostale podatkovne baze (»pb2spletna« in ciljno bazo »Prodaja«).

Sedaj je treba zajeti strukturo tabel, nad katerimi se bo izvajala integracija podatkov. Projektu dodamo novo mapo v meniju *Models* v kateri se hranijo vse strukture tabel. Z vnosom novega modela mu dodelimo ime, tehnologijo (MySQL) in eno od prej dodanih logičnih shem. Sledi zagon postopka obratnega inženirstva (ang. Reverse Engineer*)*, ki nam ustvari vse tabele glede na podano shemo in jih shrani v ustvarjeno mapo. Postopek ponovno ponovimo za preostale baze. S tem so priprave končane, tako da se lahko začne načrtovanje samega postopka ETL.

4.2.1 Polnjenje dimenzije DimCas

Za začetek načrtovanja se ustvari in poimenuje nov vmesnik. Pod *Mapping* se nam nato odpre novo delovno okolje, ki je ključnega pomena za integracijo podatkov. V prvo okno prenesemo vse tabele iz katerih bomo zajemali podatke pri prvi integraciji. V tem primeru se bodo črpali podatki iz tabele »dim_cas« (pb2) v tabelo »DimCas« (Prodaja). V okno *Target Datastore* pa prenesemo tabelo »DimCas«. Vidimo, da se podatki samodejno posodobijo v obeh oknih, tako da lahko vidimo vsa imena atributov. Z načinom *povleci-in-spusti* priredimo (mapiramo) določene atribute iz tabele »dim_cas« v tabelo »DimCas« (Slika 7). Če ODI pri nekaterih atributih ni zaznal pravega tipa, lahko to opazimo z izpisom treh pik ob atributu. To popravimo z odprtjem problematične tabele in pod *Columns* atributu dodelimo pravilen tip.



Slika 7: Prirejeni atributi za dimenzijo DimCas

Ko je preslikovanje končano, preidemo na načrtovanje toka podatkov - *Flow*. V tem oknu lahko vidimo naš strežnik s podatkovno bazo »pb2« in tabelo »dim_cas«, ciljni strežnik s skladiščem »prodaja« in dimenzijo »DimCas« ter okvir območja priprav (ang. Staging Area), ki se (kot privzeto) prav tako nahaja na ciljnem strežniku.

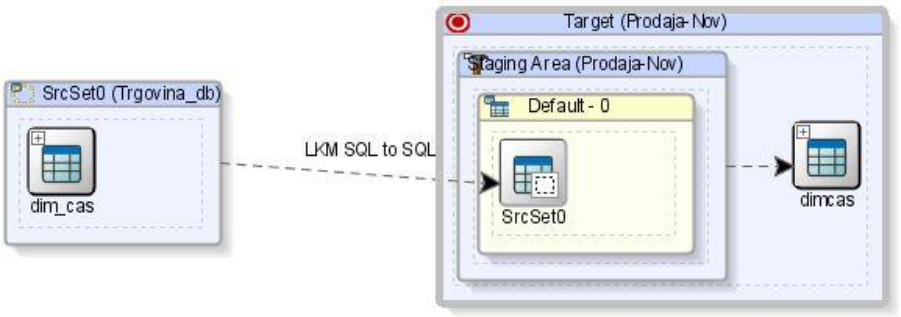
Območje priprav je prostor, v katerem ODI s pomočjo KM-jev shranjuje začasne tabele za optimizacijo same integracije. Te so lahko:

- Tabele in pogledi ustvarjeni med fazo zajemanja podatkov (C$ tabele)
- Tabele ustvarjene med fazo integracije ob uporabi določenih integracijskih modulov znanja (I$ tabele)
- Tabele napak ustvarjene med preverjanjem kakovosti podatkov (E$ tabele)
- Tabele, pogledi in stikala ustvarjena, ko je ciljna podatkovna baza uporabljena za vir podatkov z *Journalizing Knowledge* moduli (J$ objekti)

Za zajem podatkov potrebujemo poseben *Loading Knowledge Module* (LKM), ki podpira branje iz MySQL strežnikov in se imenuje »LKM SQL to SQL«. Tipični LKM uporablja naslednjo sekvenco korakov:

1. Odstrani začasno nalagalno tabelo (če obstaja).
2. Ustvari nalagalno tabelo v območju priprav.
3. Prenese podatke iz vira v ustvarjeno nalagalno tabelo z uporabo nalagalne metode. Prvi in drugi korak se ponovita za vse podatke, ki se prenašajo v območje priprav. Podatki se uporabijo v fazi integracije za polnjenje integracijske tabele.
4. Ko se faza integracije konča, se pred zaključkom vmesnika začasna nalagalna tabela odstrani.

Ta modul dodelimo našemu viru – »pb2«. Pri tem vidimo, da se ustvari puščica iz našega strežnika v območje priprav. Sledi še izbira modula, ki bo podatke zapisal iz območja priprav v naše skladišče. Uporabimo modul »IKM SQL Control Append«.



Slika 8: Diagram toka podatkov (Flow) za dimenzijo DimCas

Če želimo še lahko uporabimo modul za preverjanje napak (Check Knowledge Module). V nasprotnem primeru spremenimo možnost »FLOW_CONTROL« na »False«.

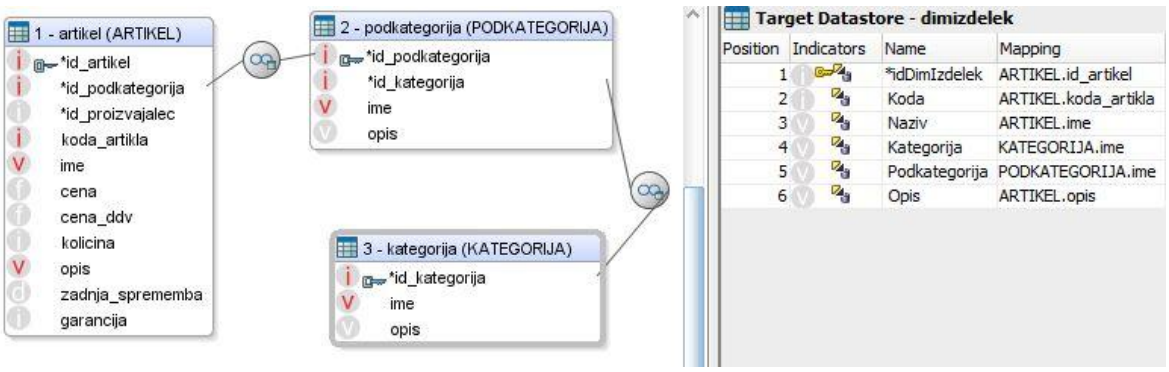
Ob zagonu integracije lahko v *Session List*, pod trenutnim datumom opazujemo potek le-te. V tem oknu se izpišejo tudi vse morebitne napake ob določenih korakih.

4.2.2 Polnjenje dimenzij DimLokacija in DimIzdelek

Proces polnjenja teh dveh dimenzij je podoben polnjenju dimenzije za čas, le da se v tem primeru uporablja več tabel hkrati. Za vsako dimenzijo posebej smo ustvarili nov *Vmesnik*, ju primerno poimenovali ter prešli na pogled *Mapping*. Za dimenzijo lokacij smo iz podatkovne baze trgovine potrebovali dve tabeli (»Trgovina« in »Posta«). Iz seznama modelov smo obe tabeli prenesli v okno načrtovanja. Pri prenašanju je treba poskrbeti, da ne prenašamo vseh tabel hkrati, saj se v tem primeru njihove povezave ne upoštevajo. Ob uspešnem prenosu smo opazili povezavo, ki poteka od atributa »id_posta« tabele »Trgovina« do stolpca »id_posta« tabele »Posta«. Ta povezava je identična SQL ukazu »TRGOVINA.id_posta=POSTA.id_posta«, pri čemer ugotovimo, da se lahko v orodju ODI delno izognemo pisanju zapletenih SQL ukazov, saj so poizvedbe predstavljene v grafični obliki, kar je za uporabnika prijaznejše in hitrejše.

Integraciji še je bilo treba dodati ciljno tabelo. Iz seznama modulov smo v okno *Target Datastore* tako prenesli tabelo »DimLokacija« in nato povezali iskane stolpce.

V primeru dimenzije »DimIzdelek« smo uporabili tabele »Artikel«, »Kategorija« in »Podkategorija« ter poskrbeli, da se prenesejo atributi »id_artikel«, »koda_artikla« ter imena artikla, kategorije, podkategorije in opisa. Kot ciljno tabelo pa smo izbrali »DimIzdelek«.



Slika 9: Določanje in prirejanje atributov za dimenzijo DimIzdelek

Pod rubriko *Flow* se sedaj v okviru virov vidi več tabel, ki so povezane s primarnimi oz. tujimi ključi. Za Nalagalni modul smo v obeh primerih izbrali »LKM SQL to SQL« ter povezali vir z območjem priprav, ki se je nahajalo na ciljnem strežniku. Podobno kot prej, uporabimo integracijski modul imenovan »IKM SQL Control Append«, ki nam integrira pripravljene podatke v našo dimenzijo.

4.2.3 Polnjenje dimenzije DimStranka

Podatki strank so tokrat v drugačni obliki, zato moramo pred načrtovanjem integracije deklarirat nov vir podatkov tipa *Microsoft Excel*. Vendar moramo pred tem ustvariti novo ime vira podatkov (Data Source Name - DSN), ki bo datoteko »KarticeZvestobe.xls« povezalo s fizično topologijo orodja ODI, ki pri tem zahteva uporabo gonilnika Jdbc-Odbc. Da to dosežemo, zaženemo orodje ODBC Data Source Administrator, ki je del operacijskega sistema Windows. Pod zavihkom *User DSN* dodamo nov vir podatkov ter mu določimo gonilnik, ki podpira tehnologijo Excel. Vir poimenujemo, poiščemo lokacijo na trdem disku in določimo različico datoteke.

Sedaj lahko v orodju ODI dodamo novo fizično arhitekturo s tehnologijo tipa Microsoft Excel in jo povežemo z ustvarjenim DSN-jem.

Slika 10: Določanje gonilnika in imena DSN za povezavo z datoteko Excel

Kot za vse vire podatkov, je pred uporabo treba zajeti in ustvariti strukturo modela tabele kartic zvestobe. Zato smo dodali nov *Model* in s postopkom *Reverse Engineering* ustvarili tabele za vse tri delovne liste datoteke »KarticeZvestobe.xls«. Pri tem je treba paziti, da označimo možnost *System table*, saj nam v nasprotnem primeru ne vrne želenih tabel oz. delovnih listov. S tem so tabele pripravljene za načrtovanje.

V okno zajemanja smo prenesli tabelo »Sheet1$«, v kateri so vsi podatki za zajem in prenos v dimenzijo »DimStranka«. Za pretok podatkov smo pa uporabili že znana modula »LKM SQL to SQL« in »IKM SQL Control Append« ter zagnali integracijo.

Drugi del polnjenja dimenzije zahteva branje iz tabele »Stranka« iz podatkovne baze spletne trgovine, ki smo jo dodali pred začetkom izvajanja integracij. Po prenosu smo ji določili stolpce za zajetje ter jih povezali z atributi tabele »DimStranka«:

KarticeZvestobe.xls	Pb2.Stranka	Prodaja.DimStranka
Ime	naziv	Ime
Priimek	/	Priimek
Kraj	kraj	Kraj
Ulica	ulica	Ulica
Pošta	posta	Posta
Davčna Številka	/	Davcna
/	emso	Emso
/	eposta	Eposta
Številka kartice		Stevilka_kartice

Tabela 6: Prirejanje imen stributov iz datoteke in izvirne baze

V oknu pretoka podatkov smo nato določili modul še za prenos podatkov »LKM SQL to SQL«. Za vnos se je tudi tokrat uporabil modul »IKM SQL Control Append«, saj omogoča tudi možnost posodabljanja podatkov v primeru obstoječih vrstic.

4.2.4 Polnjenje tabele dejstev Prodaja

Ob polnih dimenzijah podatkovnega skladišča je prišla na vrsto tabela dejstev »Prodaja«. Za polnjenje smo morali v prvo okno prenesti vseh šest tabel, ki jih potrebujemo za integracijo. Te so »Artikel«, »Artikli_v_akciji«, »Akcija«, »Postavka«, »Racun« in »Trgovina«. Ob tem smo pazili, da so se dodale vse povezave ključev, nato pa smo iz seznama modulov v okno ciljne tabele dodali tabelo dejstev »Prodaja«. Sledilo je prirejanje stolpcev, pri čemer smo morali nad dvema atributoma izvesti operacijo *SUM*, ki sešteva vrednosti nekega stolpca. To storimo v območju Lastnosti Preslikav (Mapping Properties) določenega ciljnega atributa. Če želimo, si lahko pri pisanju SQL ukazov pomagamo z Urejevalnikov Izrazov (Expression Editor) v katerem so shranjene vse operacije, ki jih potrebujemo za integracijo podatkov. V našem primeru generiramo dva ukaza:

- »SUM(POSTAVKA.stevilo_kosov)« za seštevek kosov prodanih artiklov
- »SUM(RACUN.znesek)« za seštevek zneskov

Operaciji se zapišeta v stolpec *Mapping* ob ciljna atributa »Kolicina« in »Znesek«.

Naslednji korak je določanje tujih ključev. Oracle Data Integrator ima za to operacijo čarovnika, ki nam za iskanje teh vrednosti ustvari komponento, ki se zažene ob integraciji. V orodni vrstici poiščemo možnost *Add a new Lookup*, v kateri pod *Driving Table* izberemo tabelo »Artikel«, pod *Lookup Table* pa dimenzijo »DimIzdelek«. V naslednjem koraku nato izberemo atributa iz obeh tabel (»Artikel.ime« in »DimIzdelek.naziv«), ki se bosta med sabo primerjala in vrnila vrednost iskanega ključa. Pred zaključkom, zaradi optimizacije same integracije, označimo možnost izvajanja lookup operacije na območju priprav. ODI nam je s tem ustvaril novo delovno tabelo »DIMIZDELEK(dimizdelek)«, ki pa trenutno še ne vrača rezultata. Zato ji je treba določiti atribut, ki se bo preslikal v tuj ključ »DimIzdelek_idDimIzdelek« tabele »Prodaja«. Označimo omenjen atribut in odpremo možnost urejevalnika izrazov, v katerem pod drevesnim seznamom lookup tabel poiščemo pravkar ustvarjeno tabelo. Nato v okno izrazov prenesemo atribut »idDimIzdelek«.

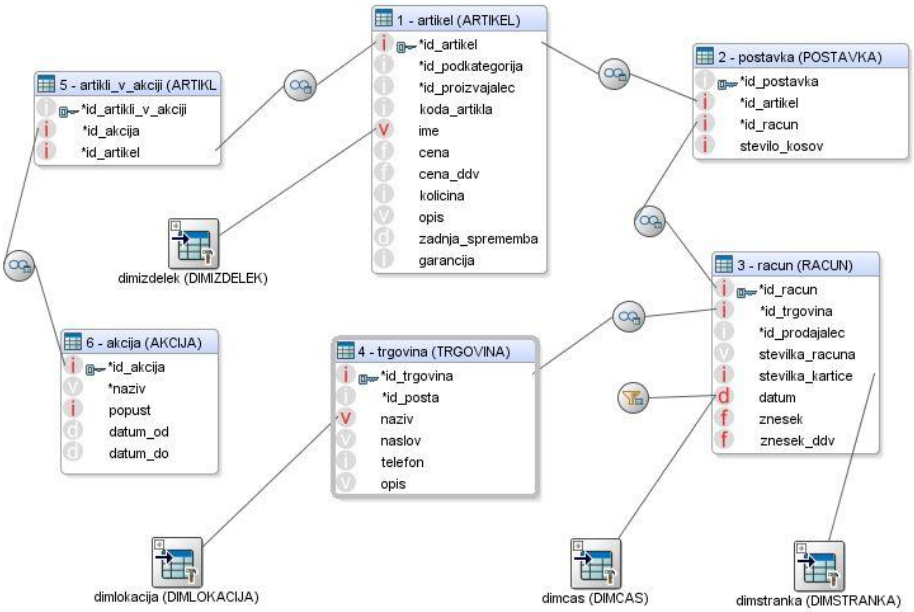
S tem smo ob uspešni primerjavi stolpcev »ime« in »naziv« preslikali vrednost ključa »idDimIzdelek« v tuj ključ »DimIzdelek_idDimIzdelek«. Lookup operacija je za ta atribut končana, zato postopek ponovimo za preostale tri tuje ključe.

Na koncu zajemanja podatkov, je treba omejiti še poizvedbo na poprej omenjen datum. To dosežemo s filtrom, ki mu lahko določamo različne omejitve. V našem primeru povlečemo atribut »datum« iz tabele »Racun«, s čemer se ustvari prazen filter. Pod njegovimi lastnostmi smo nato vpisali SQL ukaz, ki omeji poizvedbo na 5 dni transakcij:

```
RACUN.datum BETWEEN '2006-07-01' AND '2006-07-05';
```

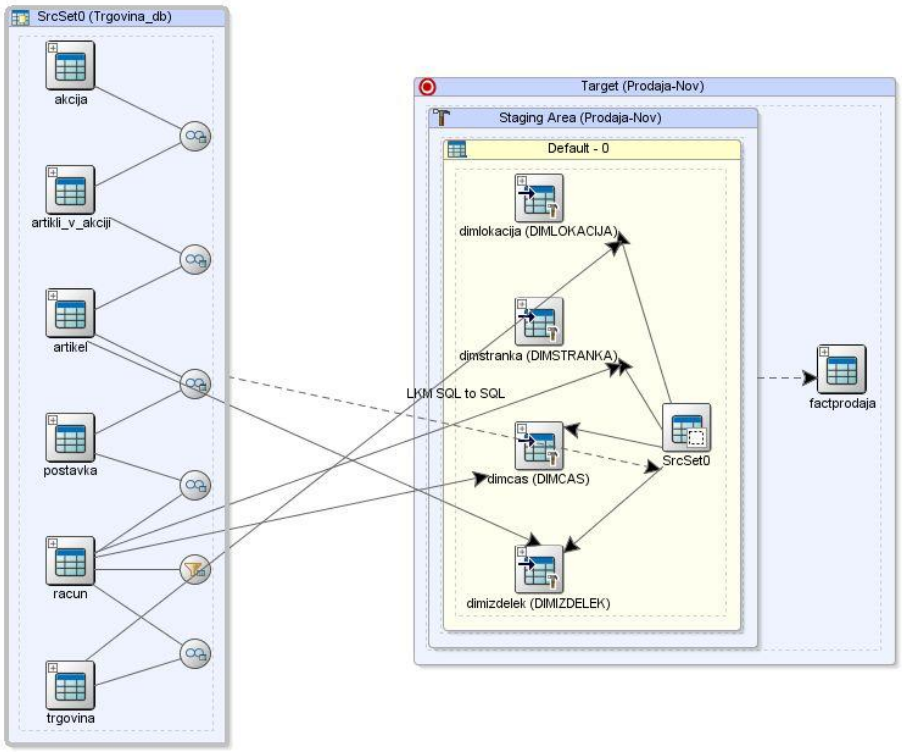
Poizvedba 7: Omejevanje poizvedbe

S tem imajo vsi ciljni atributi svojo preslikavo. Prazen je ostal samo primarni ključ tabele dejstev, ki pa se zaradi *Auto Increase* nastavitve vnaša in povečuje avtomatsko ob dodajanju novih vrstic.



Slika 11: Načrt zajemanja podatkov za tabelo dejstev Prodaja

V naslednjem koraku smo odprli okno pretoka podatkov ter integraciji določili že znane module in zagnali integracijo.



Slika 12: Načrt pretoka podatkov za tabelo dejstev Prodaja

4.3 ETL z Microsoft SQL Server Integration Services

Ob zagonu orodja SQL Server Business Intelligence Development Studio (BIDS) se nam odpre pozdravno okno, v katerem odpremo prejšnje projekte ali pa ustvarimo nove. V našem primeru smo ustvarili nov projekt, katerega predloga se imenuje Integration Services Project, ki nam odpre potrebna okna za integracijo podatkov:

- Nadzor Pretoka (Control Flow). V oknu nadzora pretoka urejamo in nastavljamo opravila, ki omogočajo funkcionalnost v obliki povezanih paketov. Pri tem si pomagamo z različnimi elementi nadzora pretoka podatkov, ki se pojavijo ob odprtju omenjenega okna.

- Pretok Podatkov (Data Flow). Okno pretoka podatkov v paketu storitve za integracijo se uporablja z uporabo specifičnih gradnikov, ki so namenjeni za pretok podatkov: viri, ki izvlečejo podatke, transformacije, destinacije za vnašanje podatkov in poti, ki povezujejo vhode in izhode komponent za nadzor podatkov v podatkovni pretok.

- Obravnava Dogodkov (Event Handler). Ob zagonu integracije vsebniki in opravila ustvarjajo dogodke. Po potrebi lahko zato določimo naloge oz. obravnave, ki se bodo izvedle kot odgovor na pojave določenih dogodkov. Npr. ustvarimo lahko obravnavo, ki pošlje elektronsko pošto v primeru neuspešnega zaključka nekega opravila.

- Upravitelji Povezav (Connection Managers). BIDS vključuje raznolike upravitelje povezav za zadovoljitev potreb po povezovanju različnih tipov strežnikov in virov podatkov. Upravitelje povezav uporabljajo komponente pretoka podatkov, ki berejo in vnašajo podatke v različne tipe shramb in dnevniki, ki zapisujejo informacije na strežnik, SQL server tabelo ali datoteko. Npr. paket z opravilom za pošiljanje elektronske pošte uporablja upravitelja povezav tipa SMTP za povezavo do Simple Mail Transfer Protocol strežnika.

- Orodjarna (Toolbox). V tem oknu so zbrani vsi elementi oz. gradniki, ki se uporabljajo ob različnih korakih integracije. To okno se osvežuje in spreminja glede na naše trenutno načrtovanje. V primeru načrtovanja pretoka podatkov se pojavijo vsi elementi, ki obsegajo omenjen korak.

4.3.1 Polnjenje dimenzije DimCas

Za polnjenje nove dimenzije našega podatkovnega skladišča smo najprej ustvarili SSIS paket, ki smo ga preimenovali v »DimCas.dtsx«. V okno *Control Flow* smo nato vnesli element *Data Flow Task*. Ta element enkapsulira gonilnik podatkovnega pretoka, ki prenaša podatke med viri in destinacijami ter omogoča uporabnikom izvajati transformacije, čistiti podatke in jih spreminjati med prenašanjem. V našem primeru je dovolj en *Data Flow Task*, vendar jih je lahko med sabo povezanih tudi več, kot bo to prikazano v kasnejših primerih.

Omenjena komponenta se ureja v oknu *Data Flow*. Za branje podatkov iz podatkovne baze naše trgovine smo iz orodnega okna vnesli element *ADO NET Source*, ki prebere podatke od .NET ponudnika in jih ponudi na voljo našemu podatkovnemu pretoku. Ker je potrebno tej komponenti definirati vir podatkov, smo ustvarili novega upravitelja povezav in uporabili ponudnika imenovanega ODBC Data Provider (MySQL Data Provider v našem primeru ni deloval pravilno), pri čemer smo nato ustvarili nov DSN preko ODBC Data Source Administratorja operacijskega sistema Windows. Po uspešni povezavi smo *ADO NET Source*-u dodelili ustvarjenega upravitelja, ki smo ga imenovali »MySQL-DSN_pb2.pb2« in vnesli SQL ukaz, ki bo iz podatkovne baze prebral vse podatke tabele »dim_cas«:

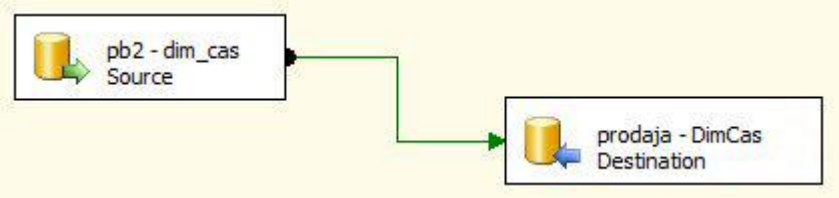
```
SELECT * FROM pb2.dim_cas;
```

Poizvedba 8: Branje atributov tabele Dim_cas

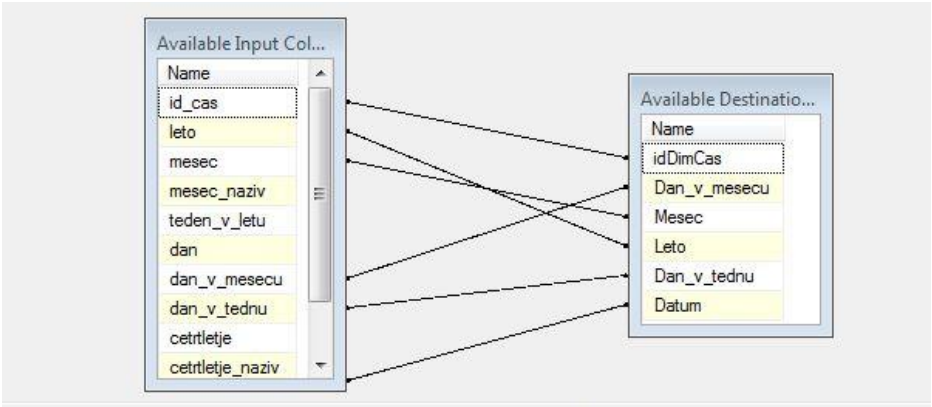
Pravilen ukaz nam pod rubriko *Columns* izriše tabelo dobljenih stolpcev, kjer lahko določamo in spreminjamo njihova imena.

Dobljene podatke smo nato morali zapisati v naše skladišče. To nalogo opravlja komponenta *ADO NET Destination*, ki naloži podatke iz podatkovnega toka v vrsto različnih ADO.NET-kompatibilnih podatkovnih baz, ki uporabljajo podatkovne tabele ali poglede. Ponuja možnost vnašanja podatkov že ustvarjenim tabelam, kot tudi možnost ustvarjanja novih tabel. Komponento smo s puščico povezali s prejšnjim korakom, da se podatki, pridobljeni iz našega vira prenesejo naprej. Podobno kot prej smo ustvarili novega upravitelja povezav »MySQL-DSN_prodaja2.root« za naše podatkovno skladišče in ga dodelili ciljni komponenti.

Pred zagonom integracije je bilo treba še spremeniti imena stolpcev in jih pravilno dodeliti stolpcem v naši tabeli »DimCas«. Ko *ADO NET Destination* komponenti določimo ciljno tabelo, lahko pod možnostjo *Mapping* s povezovanjem določamo, kateri stolpci se bodo preimenovali in prepisali v podatkovno skladišče.



Slika 13: Povezani komponenti ADO NET Source in ADO NET Destination



Input Column	Destination Column
id_cas	idDimCas
dan_v_mesecu	Dan_v_mesecu
mesec	Mesec
leto	Leto
dan_v_tednu	Dan_v_tednu
datum	Datum

Slika 14: Prirejena imena stolpcev tabel časa

Ob zagonu integracije se napake in opozorila izpisujejo v oknu *Progress*. V našem primeru se je izpisalo nekaj opozoril v zvezi z neizkoriščenimi podatki, ki smo jih izpustili med načrtovanjem. Za našo integracijo niso bistvenega pomena, vendar bi se opozorilom lahko izognili z bolj nazornim SQL ukazom. Integracija se je zaključila z zeleno obarvanima komponentama in zaključenim progress dnevnikom, kar nakazuje na uspešno zapisane podatke v podatkovno skladišče.

4.3.2 Polnjenje dimenzij DimIzdelek in DimLokacija

Polnjenje dimenzij za izdelke in lokacije poteka podobno kot pri dimenziji »DimCas«. Za obe smo uporabili komponento *Data Flow Task* ter komponenti *ADO NET Source* in *ADO NET Destination*. Za branje in zapisovanje smo uporabili že ustvarjene upravitelje povezav »MySQL-DSN_pb2.pb2« in »MySQL-DSN_prodaja2.root« ter ju povezali z dodanima korakoma. Sledilo je vnašanje SQL poizvedbe za zajem podatkov:

- Za dimenzijo »DimIzdelek«:

```
SELECT a.id_artikel, a.ime, k.ime AS kategorija, pk.ime AS
podkategorija, a.opis
FROM artikel a, kategorija k, podkategorija pk
WHERE a.id_podkategorija = pk.id_podkategorija
AND pk.id_kategorija = k.id_kategorija;
```

Poizvedba 9: Branje podatkov izdelka

- Za dimenzijo »DimLokacija«:

```
SELECT t.id_trgovina, t.naziv, t.naslov, t.telefon,
p.stevilka_poste
FROM trgovina t, posta p
WHERE t.id_posta = p.id_posta;
```

Poizvedba 10: Branje atributov za dimenzijo DimLokacija

Imena stolpcev pa so se priredila na enak način kot prej. V dimenzijo lokacij so se vpisali vsi zajeti atributi, pri dimenziji izdelkov pa vsi, razen primarnega ključa, saj se lahko zaradi nastavitve *Auto Increase* na ciljni bazi zapolnjuje in veča sam. Sledil je zagon integracije.

4.3.3 Polnjenje dimenzije DimStranka

Ta integracija zahteva polnjenje tabele iz dveh virov. Prvi je datoteka kartic zvestobe tipa .xls, drugi pa je tabela strank iz podatkovne baze spletne trgovine. Za branje iz Excel datoteke potrebujemo novo komponento imenovano *Excel Source*.

Excel Source se uporablja za branje podatkov iz delovnih listov datotek tipa Excel. Omogoča štiri različne načine dostopa do podatkov:

- Tabela ali pogled.
- Ime tabele ali pogleda specificiranega v spremenljivki.
- Z SQL ukazom. Poizvedba je lahko parametrizirana.
- Z SQL poizvedbo shranjeno v spremenljivki.

Komponenta uporablja *Excel Connection Manager* za povezavo z datotekami in njihovimi podatki. Upravitelj se nastavi z določitvijo fizične poti do datoteke ter njene verzije, pri čemer se ob uspešni povezavi v komponenti izbere primeren delovni list, s katerega želimo brati podatke. Prvo vrstico smo nato nastavili kot vir poimenovanja stolpcev, zato se ta vrstica ni prenesla v tok podatkov integracije.

Sledila je določitev ciljne tabele »DimStranka« v komponenti *ADO NET Destination* ter prirejanje imen stolpcev. Za popoln seznam kupcev smo še morali v dimenzijo »DimStranka« dodati podatke iz podatkovne baze spletne trgovine. Dodali smo obe znani ADO NET komponenti ter ustvarili novega upravitelja povezav še za spletno trgovino imenovanega »MySQL-DSN_pb2spletna.pb2spletna«, ga povezali z vhodno komponento ter vnesli SQL poizvedbo za pridobitev vseh podatkov tabele »Stranka«. Ker Destination komponenta podpira tudi posodabljanje podatkov v tabeli za že obstoječe podatke, jo lahko ponovno uporabimo tudi v primerih kadar te želimo ohraniti.

4.3.4 Polnjenje tabele dejstev Prodaja

Ob vseh napolnjenih dimenzijah našega podatkovnega skladišča je nato nastopil še čas za tabelo dejstev. Ustvarili smo nov *Data Flow Task* ter dodali *Souce* komponento, ki bo prebrala vse potrebne podatke iz baze trgovine. Ko jo povežemo s primernim *Connection Managerjem*, dobimo možnost vnosa SQL poizvedbe:

```
SELECT a.ime AS ime_izdelka, LEFT(r.datum, 10) AS datum_nakupa,
SUM(p.stevilo_kosov) AS kolicina, SUM(r.znesek) AS znesek,
r.znesek AS cena, ak.popust AS popust, r.znesek_ddv AS cena_ddv,
t.naziv AS ime_trgovine, r.stevilka_kartice AS st_kartice
FROM artikel a, postavka p, racun r, akcija ak, artikli_v_akciji
ava, trgovina t
WHERE a.id_artikel = p.id_artikel
AND p.id_racun = r.id_racun
AND r.id_trgovina = t.id_trgovina
AND ava.id_artikel = a.id_artikel
AND ak.id_akcija = ava.id_akcija
AND r.datum BETWEEN '2006-07-01' AND '2006-07-05'
GROUP BY ime_izdelka, datum_nakupa;
```

Poizvedba 11: Zajemanje atributov, ki so potrebni za polnjenje tabele dejstev

Datum v izvirni tabeli je drugačnega formata, kot je datum v naši dimenziji. Zato je treba s pomočjo komponente *Data Conversion* spremeniti format iz tipa »yyyy.mm.dd hh:mm:ss« v tip »yyyy.mm.dd«.

Data Conversion omogoča pretvorbo tipa podatkov vhodnih stolpcev v drugačen tip in jih nato kopira in posreduje na nov izhodni stolpec. Npr. določen paket lahko zajema podatke iz večih virov, nakar s pomočjo te komponente pretvori stolpce v tip, ki ga zahteva ciljna baza. Na enem vhodnem stolpcu lahko izvedemo več pretvorb hkrati. S to transformacijo lahko paket izvaja naslednje pretvorbe:

- Spreminjanje tipa podatkov.
- Določanje dolžine podatkovnih nizov in natančnost ter obseg numeričnih podatkov.
- Navajanje kodne strani.

V našem primeru smo tip *DT_DBTIMESTAMP* pretvorili v *DT_DATE* in izhodni stolpec imenovali »datum_nakupa_nov«. Prav tako moramo pretvoriti dolžino številke kartice iz 4-bitnega integerja (DT_I4) v 8-bitnega (DT_I8), saj nam drugače med integracijo vrne napako, ker je ciljni atribut večje velikosti kot vhodni podatki.

Sledilo je primerjanje podatkov ter iskanje primarnih ključev dimenzijskih tabel. V orodju BIDS je v povezavi s podatkovnimi bazami MySQL ta postopek nekoliko drugačen, ker je pred uporabo komponente *Lookup* potrebno vse primerjalne atribute in iskane ključe MySQL tabele vnesti v datoteke Predpomnilnik (Cache). Iz teh datotek se s pomočjo Lookup komponente nato izvede operacija iskanja in vračanja atributov.

Zato smo v našem v oknu *Control Flow* dodali štiri *Data Flow* komponente (ena za vsako cache transformacijo) in v območje načrtovanja pretoka podatkov prvega opravila dodali korak *ADO NET Source*, ki je iz prve dimenzije »DimCas« zajel vse atribute. Te podatke

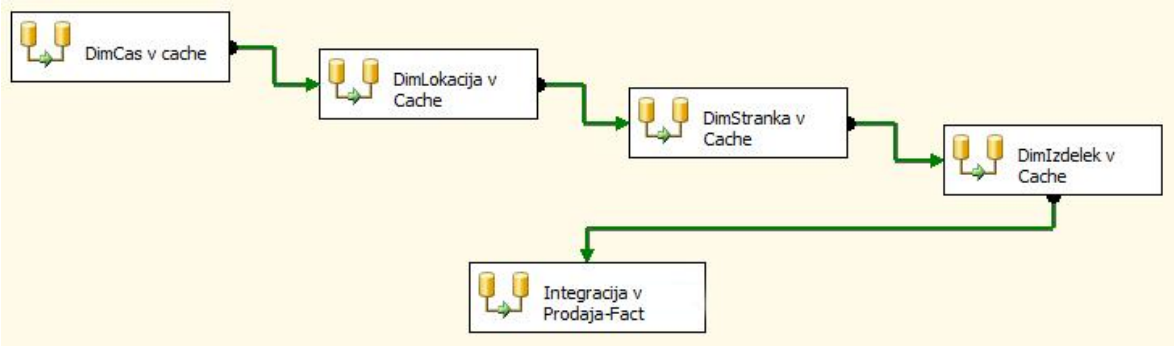
smo nato morali preslikati v cache datoteko, kar storimo s pomočjo *Cache Transform*. Da bi komponenta vedela kam shraniti podatke, je treba določiti upravitelja povezav za datoteke Cache. Vanj vnesemo ime datoteke in fizično pot na lokalnem disku, čemur sledi nastavljanje atributov tabele. Za iskanje ključev smo v našem primeru potrebovali dva atributa dimenzije »DimCas« (»idDimCas« in »Datum«), zato smo datoteki Cache določili dva podobna atributa in ju poimenovali »CasID« in »Datum«. Pomembno je paziti na stolpce *Index Position* in *Type*, saj se morajo ujemati s podatki, ki jih pridobimo iz naših dimenzij:

Ime	Pozicija Indeksa	Tip
CasID	0	[DT_I8]
Datum	1	[DT_DATE]

Tabela 7: Primer nastavljanja lastnosti tabele v datoteki Cache

Stolpce *Length*, *Precision* in *Scale* smo v našem primeru lahko pustili prazne. S tem se je implementacija nove povezave zaključila.

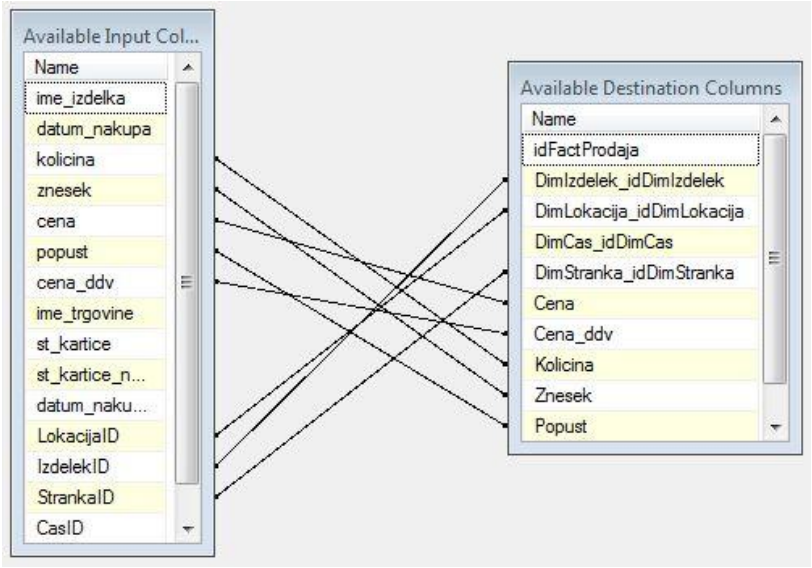
Komponenti *Cache Transform* smo nato določili ustvarjeno povezavo ter preslikali vhodna atributa »idDimCas« in »Datum« v »CasID« in »Datum«. Postopek ustvarjanja Cache datotek je bil za dimenzijo »DimCas« končan, vendar ga je bilo treba ponoviti še za preostale tri dimenzije. *Data Flow Task*-e smo nato povezali ter poskrbeli, da se ob zagonu integracije vse zažene v pravem vrstnem redu, pri čemer se najprej izvedejo transformacije Cache in nato sama integracija tabele dejstev.



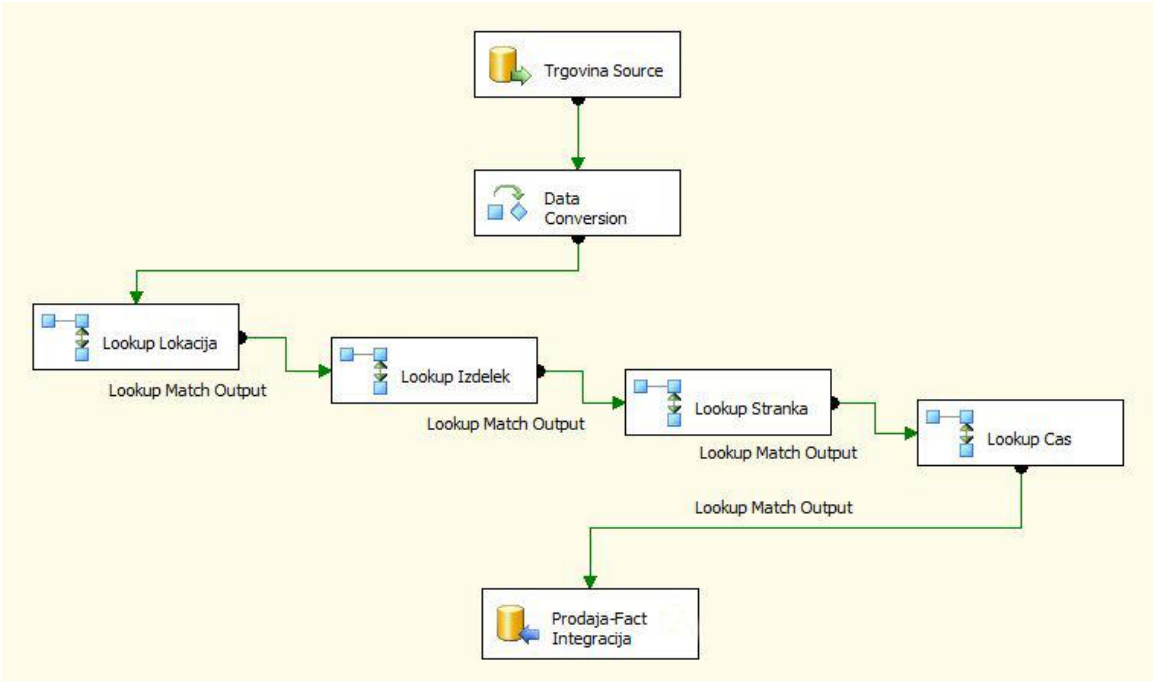
Slika 15: Pretok opravil integracije za Prodajo

Sedaj smo lahko nadaljevali z načrtovanjem lookup opravil, s katerimi poiščemo primarne ključe dimenzij. Za komponento *Data Conversion* so bile dodane 4 komponente *Lookup*, katerim so se dodale povezave Cache datotek. Ob odprtju komponent, se pod rubriko *Cache mode* nahajajo 3 nastavitve. Za naš primer je bila najbolj primerna »Full cache«, kar nam pred zagonom iskanja ključev, celotno referenčno tabelo cache zajame v predpomnilnik. Ta nastavitev je najhitrejša, vendar moramo paziti, da so te tabele relativno

majhne, saj nam lahko v nasprotnem primeru zmanjka prostora, zaradi česar se postopek prekine. Sledila je določitev tipa (Cache) in imena povezave (LookupCacheCas). Sedaj je bilo treba določiti še atribut, ki se bo primerjal z imenom lokacije. Pod oknom *Columns* smo tako povezali atributa »datum_nakupa« in »datum« ter na lookup tabeli označili »CasID«, kot atribut, ki se vrne v tok podatkov. Postopek se je ponovil za druge tri dimenzije in njihove *Lookup* komponente. Na koncu se je dodal korak *ADO NET Destination* za vnos vseh potrebnih stolpcev, ter zagnala integracija za tabelo dejstev.



Slika 16: Preslikovanje atributov za tabelo dejstev



Slika 17: Postopek zadnjega opravila integracije

5 ANALIZA IN PRIMERJAVA ORODIJ

V diplomski nalogi smo uporabili merila, ki so pogosto uporabljena v procesih testiranj [15] in ki smo jih lahko uporabili v obsegu izvedenega praktičnega primera. Kriteriji zajemajo zahtevnost nastavitev orodij in intuitivnost uporabe, ter tehnične lastnosti kot so podpora, funkcionalnost in zmogljivost. Na koncu so predstavljene tudi splošne ugotovitve primerjav.

5.1 Zahtevnost namestitve in nastavitve orodij

Obsežna orodja velikokrat zahtevajo obsežne namestitve. Med takšna orodja sodi Oracle Data Integrator, pri katerem je pred uporabo treba namestiti številne elemente paketa. Ob zagonu namestitve samega orodja smo čarovniku določili vse potrebne komponente, ki smo jih potrebovali v nalogi. Za razliko od SSIS in PDI, je namestitev delovnih repozitorijev obvezna. Repozitorij je centralni prostor, kamor se hranijo različni podatki, kot so informacije o projektih, njihovih paketih, nastavitvah ipd. Oracle vsebuje dva tipa repozitorijev – Glavni repozitorij in Delovni repozitorij. Izdelava le teh je lahko relativno zahtevna, zato smo za naš projekt namestili demonstracijsko okolje, ki nam je omogočalo pogoje za izvedbo integracij. Ob tem še je potrebno nastaviti spremenljivko `JAVA_HOME`, ki vsebuje pot do JVM (Java Virtual Machine), primerne za nameščen ODI.

SSIS je del paketa SQL Server 2008, ki se lahko namesti skupaj s Server okoljem. Med postopkom se namestijo potrebne komponente oz. aplikacije, ki so za nas pomembne (Database Engine Services, Business Intelligence Development Studio, Integration Services, ipd.). Sledi določanje računov za različne nameščene storitve.

Namestitev PDI je v tem primeru najhitrejša, saj ne zahteva nobenih posebnih korakov, le razpakiranje paketa določene različice. Po potrebi nastavimo administratorski račun in geslo.

5.2 Intuitivnost in praktičnost orodij

Vsako programsko orodje, ki se želi približati čim širšemu krogu uporabnikov, mora delovati na sistemu, ki je dostopen in enostaven za uporabo.

PDI in SSIS sta si pri tem precej podobna. Oba delujeta na standardiziranem principu dodajanja in povezovanja različnih komponent oziroma korakov, kar je za povprečnega uporabnika dokaj razumljivo. SSIS uporablja veliko čarovnikov, ki nam olajšajo in skrajšajo čas načrtovanja, ki bi ga drugače porabili za preučevanje in nastavljanje

določenih komponent. Praktičnost teh dveh orodij je zato predvsem odvisna od naše organizacije.

ODI pa je na začetku nekoliko neintuitiven. Moduli znanja, vmesniki, uporaba različnih topologij, postopki *reverse-engineering* ipd., vse to je na prvi pogled drugačno, zato jih je pred uporabo orodja treba preučiti. Prav tako je postavitev oken drugačna, kar lahko začetne uporabnike zmede.

Kljub slabšemu začetku se je kmalu izkazalo, da je načrtovanje postopkov integracij s tem orodjem enostavnejše in hitrejše zaradi grafičnega vmesnika, ki nam pomaga pri oblikovanju poizvedb. ODI prav tako predvideva in sestavlja nekatere dele pretoka podatkov, kar nam olajša tudi ta korak načrtovanja. Vgrajeni moduli znanja, ki se uporabljajo za branje in integracijo podatkov, so bili uporabni za MySQL kot tudi za veliko število drugih tipov podatkovnih baz. Po potrebi pa lahko te module napišemo tudi sami, če želimo izvajati kakšno specifično opravilo.

5.3 Podpora operacijskim sistemom

Ob izbiranju primernega ETL orodja, je zelo pomembno, na katerem operacijskem sistemu deluje določeno podjetje, saj predstavlja zamenjava takšnega sistema velik časovni kot tudi finančni strošek, ki si ga v večini primerov večja podjetja ne morejo privoščiti. Zato je podpora različnim platformam precejšnja prednost, ki integracijskim orodjem odpira dodatna tržišča.

	PDI	ODI	SSIS
Linux	Da	Da	Ne
Windows	Da	Da	Da
Solaris	Da	Da	Ne
Mac OS	Da	Da	Ne

Tabela 8: Primerjava podpore različnim operacijskim sistemom

V tabeli vidimo, da orodji proizvajalcev Pentaho in Oracle podpirata vse večje operacijske sisteme, medtem ko je SQL Server Integration Services omejen le na Windows. Ob tem lahko omenimo, da je okolje Windows sicer najbolj razširjen OS, zato SSIS dobro uspeva kljub omejeni podpori drugim platformam.

5.3 Podpora in kompatibilnost z mysql

Na področju informatike obstaja ogromno tehnologij, na katerih temeljijo različne podatkovne baze. Za integracijska orodja je zato zelo pomembno, da podpirajo čim širši spekter tipov baz, da lahko zadovoljijo vse potrebe ciljnih uporabnikov. Ena od najbolj razširjenih platform je MySQL, katero smo uporabljali v naši nalogi.

Pentaho Data Integration je z MySQL imel najmanj težav, saj je ob povezovanju z bazami, brez komplikacij bral in pisal v podatkovne tabele. Pred tem tudi ni bilo potrebe po dodajanju oziroma nastavljanju posebnih dodatkov, ki bi zagotovili komunikacijo med tehnologijama. Prav tako smo med načrtovanjem integracij v orodju Oracle Data Integrator dokaj hitro uspešno povezali MySQL s topologijo orodja.

Ob uporabi SSIS pa smo naleteli na nekaj težav, ki so nam podaljšale čas načrtovanja postopkov ETL. Na prvo smo naleteli ob povezovanju s podatkovno bazo trgovine, kjer privzet upravitelj MySQL povezav ni deloval pravilno v povezavi z vhodno komponento. Tako smo morali uporabiti in ustvariti posebne Odbc povezave za MySQL, preko katerih so vhodne komponente lahko komunicirale z bazo. Kasneje smo naleteli na podoben problem pri zapisovanju v podatkovno skladišče. Orodje nam je vračalo napako sintakse, saj je bil generiran SQL orodja BIDS nezdružljiv s privzetimi nastavitvami gonilnika baze MySQL. Rešitev je bila ukaz, ki spremeni nastavitve MySQL konektorjev, da delujejo v ANSI-kompatibilnem načinu za posamezno sejo ob vsakem zagonu integracije. Ta ukaz smo vnesli v Odbc konektor:

```
SET SESSION sql_mode= 'ansi';
```

Poizvedba 12: Spreminjanje nastavitev za primerno vnašanje podatkov v bazo MySQL

Podrobnejša raziskava problema je razkrila, da gre za pogost in znan problem, ki pa do tega trenutka še ni bil odpravljen [25].

Druga ovira je nastopila ob načrtovanju lookup transformacij, kjer je bilo treba pred tem nekatere atribute dimenzij prenesti v Cache datoteke. Brez tega se lookup ni mogel izvesti, ker komponenta ne podpira neposrednega branja podatkovnih baz tipa MySQL.

Tem problemom se lahko izognemo z določenimi dodatki, ki jih vključimo v paket SSIS, vendar so ti plačljivi.

5.4 Čas integracije

Zelo pomemben dejavnik vsakega ETL postopka je čas izvajanja integracije. Pri tem je cilj doseči razmere, pri katerih bo ta čas čim krajši. V našem primeru smo zaradi testiranja večih orodij in omejevanja poizvedb na strežniku, opravljali z relativno manjšo količino podatkov.

Čas posameznih integracij se je meril v samih orodjih, kjer so se najbolj razlikovali časovi zajemanja podatkov iz podatkovnih baz trgovine. Pri tem je treba upoštevati promet na strežniku, saj je bil v času izvajanja integracij različno obremenjen. Zato smo integracije zagnali večkrat in poskušali izmeriti povprečni čas, s katerim bi se čimbolj približali pravim rezultatom.

	PDI	**ODI**	**BIDS**
DimCas	2 s	12 s	12 s
DimLokacija	0,2 s	1 s	3,1 s
DimIzdelek	0,7 s	5 s	8 s
DimStranka	3 s	22 s	37 s
Prodaja	20 s	66 s	24 s

Tabela 9: Čas integracij posameznih orodij

Ob časovnih rezultatih opazimo zanimiv pojav kjer so integracije z orodjem ODI hitrejše od orodja BIDS pri polnjenju dimenzij, medtem ko pri polnjenju tabele dejstev, ODI naleti na težave. PDI je v našem primeru deloval najhitreje na vseh področjih in od obeh drugih orodij zajemal in zapisoval podatke dimenzij kar 6 do 7-krat hitreje. Razlika se zmanjša ob integraciji tabele dejstev, kjer se lahko primerja z okoljem BIDS.

Sklepamo lahko, da je Pentaho Data Integration najhitrejši v integraciji relativno manjše količine podatkov. Za takšne rezultate lahko morebiti vpliva tudi njegova dobra kompatibilnost z MySQL bazami. Ob vsem tem bi bilo zanimivo primerjati orodja ob večji količini vrstic, saj je ob polnjenju tabel dejstev razvidno, da se razkorak med hitrostmi ob večjih poizvedbah zmanjšuje.

5.5 ETL funkcionalnost

Za uspešno pretvorbo podatkov potrebujejo orodja čimbolj široko paleto operacij s katerimi zadovoljijo različne zahteve končnih uporabnikov. Te operacije oziroma funkcije najdemo v več fazah postopkov ETL, kjer opravljajo različne naloge, ki obsegajo povezovanje podatkov, njihovo zajemanje, pretvarjanje, preslikovanje, odstranjevanje, ipd.

V naši nalogi smo pri orodjih uporabljali naslednje transformacije oz. postopke:

- Branje/pisanje v MySQL podatkovne baze
- Branje iz datoteke Excel (Excel Input)
- Prirejanje imen atributov (Mapping)
- Prirejanje tipov atributov (Data Conversion)
- Iskanje ključev (Lookup Transformation)
- Štetje vrstic (Row Count)
- Agregacija (Aggregation)
- Dodajanje vrstic (Add Rows)
- Posodabljanje vrednosti (Update)

Dodatno:

- Zapisovanje v datoteke Cache (Cache Transform)

Med načrtovanjem smo pri vseh orodjih uspešno upravljali nad omenjenimi koraki pri čemer nismo naleteli na večje ovire pri iskanju rešitev za določeno funkcijo. Od morebitnih problemov (zraven lookup transformacij v SSIS, kjer je povečini kriva nezdružljivost z MySQL), bi lahko omenili pomanjkanje *Group By* funkcije v orodju ODI. Oblikovanje SQL poizvedb temelji na grafičnem vmesniku, ki samodejno generira Group By stavke in nad katerim nimamo direktnega nadzora, zato nas ta problem nekoliko preseneča. Posledica je bila večje število vrnjenih vrstic, ki pa so se nato sortirale in uredile v območju priprav.

5.6 Ugotovitve

Ob izvajanju integracij smo z vsemi tremi orodji dosegli želene rezultate. Ob tem je na čas načrtovanja integracij vplivalo predznanje oziroma neznanje orodij. Najenostavneje so potekale integracije v Pentaho orodju, saj je okolje zelo intuitivno in preprosto, v nekaterih primerih morda celo preveč, saj ne vsebuje veliko možnosti za avtomatizacijo postopkov, kar lahko pri kompleksnejših integracijah podaljša čas načrtovanja. Kompatibilnost z MySQL bazami je odlična, kot tudi hitrost izvajanja integracij. Pri tem je vprašljiva sposobnost integracij pri večji količini podatkov (in bazah drugačnega tipa), kjer sta se mu orodji ODI in SSIS v časovih izvedbe nekoliko približala.

V načinu oblikovanja poizvedb in splošnega načrtovanja ETL, se je ODI-jeva rešitev izkazala za zelo uporabno in hitro. To seveda zahteva določene priprave, zato je orodje bolj primerno za večje projekte in količine podatkov. Povezuje ogromno število tehnologij, kar je razvidno v njegovem seznamu topologij in temu primerno je tudi povezovanje z bazami MySQL potekalo brez zapletov. V našem primeru smo naleteli na manjše težave pri zajemanju podatkov in uporabi ukaza za grupiranje, kar je vplivalo tudi na daljše časove integracije.

SSIS pri tem zajema dobre strani intuitivnega in standardiziranega načrtovanja, kot tudi avtomatizacijo nekaterih opravil. Vsebuje preprostost okolja PDI in zadovoljive časove pri načrtovanju ter izvajanju kompleksnejših integracij, vendar je problem v splošni združljivosti, saj smo med delom naleteli na vrsto izjem in napak ob povezovanju z MySQL bazami, ki so zahtevale posebne rešitve. Ob tem orodje odlikuje obsežna dokumentacija (uradna kot tudi neuradna), ki nam je olajšala delo in pomagala pri omenjenih težavah. SSIS ni vsestranski produkt, saj deluje izključno na operacijskih sistemih Windows, vendar je podpora na tem področju toliko bolj močnejša.

Ob tem moramo upoštevati pomemben vidik poslovnega načrta proizvajalcev, ki razkriva, da sta ODI in SSIS plačljivi storitvi, medtem ko je PDI odprto-koden (ang. open-source), kar predstavlja pomemben dejavnik pri izbiranju pravega orodja za uporabo v poslovnem svetu.

6 SKLEP

V današnjem času je ustreznost podatkov in pridobivanje potrebnih informacij vedno bolj pomembno za uspešno konkuriranje na področju informatike, kot tudi drugje. Zato je treba za dosego teh zahtev podatke pretvoriti v takšno obliko, ki analitikom kar najbolje ustreza in nad katero je mogoče izvesti različne analize v najkrajšem možnem času. ETL orodij je veliko, zato je izbira pravega orodja za določeno podjetje ključnega pomena. Pri tem je treba upoštevati, kaj želimo, kakšno je naše predznanje, na katerih obstoječih sistemih se bodo integracije izvajale, katere so naše omejitve, kakšen je naš finančni položaj ipd. Ob nepravilni odločitvi je lahko proces integracij neprimerno otežen, kar pa lahko posledično vpliva na poslovne odločitve, ki temeljijo na neustreznih informacijah.

V diplomski nalogi smo opredelili težave, ki se pojavijo ob uporabi neintegriranih rešitev, kot tudi razloge za združevanje informacijskih sistemov. Ugotovili smo kako nam integracija omogoča celovitejši pogled na podatke, in v kakšno obliko je potrebno le-te preoblikovati, če želimo izvajati podrobne analize. Opisali smo delovanje in korake procesa zajema, pretvorbe in nalaganja podatkov, kot tudi pomen in vlogo podatkovnih skladišč v procesu integracije.

Spoznali smo tri razširjena programska orodja proizvajalcev Pentaho, Oracle in Microsoft, ki se uporabljajo za integracijo podatkov, njihove lastnosti in različne pristope k postopkom ETL. S prikazom na praktičnem primeru smo izvedli analizo oddaljenih, kot tudi lokalnih virov podatkov, ter oblikovali podatkovno skladišče primerno za vnos pretvorjenih podatkov. Na dokumentiranem postopku smo uspešno izvedli integracije z omenjenimi produkti ter ugotovili, da vsa tri orodja ponujajo vrsto različnih možnosti za dosego zastavljenih ciljev, vendar se načini načrtovanja in izvajanja integracij med seboj nekoliko razlikujejo. Nekatera temeljijo na preprostosti vmesnika in odprto-kodnem pristopu, druga na avtomatiziranih postopkih ter širokem naboru operacij in podpore. V našem primeru smo se osredotočili na delo z MySQL bazami in opazovali ter primerjali obnašanje orodij v povezavi s to tehnologijo. Ob tem smo ponekod naleteli na težave ob zajemu in vstavljanju podatkov v podatkovna skladišča, pri čemer je bil glavni vzrok problemov delna nekompatibilnost med tehnologijami uporabljenih baz in testiranih produktov, vendar smo jih v vseh primerih uspešno zaobili.

Pentaho Data Integration, SQL Service Integration Services in Oracle Data Integrator so močna programska orodja, ki uspešno konkurirajo na področju integracije. Vsa tri so sposobna izpolniti večino zahtev, vendar je izvedba odvisna predvsem od naših obstoječih sistemov, zmožnosti in znanja.

7 VIRI, LITERATURA

[1] P. Vassiliadis, A. Simitsis, S. Skiadopoulos, *Conceptual Modeling for ETL Processes*, http://cs.uoi.gr/~pvassil/publications/2002_DOLAP/dolap02_CR.pdf, zadnjič obiskano: 29.8.2013

[2] B. Knight, E. Veerman, G. Dickinson, D. Hinson, D. Herbold, *Professional Microsoft SQL Server 2008 Integration Services***,** Wiley, 2008

[3] Oracle, *Getting started Guide*, http://www.oracle.com/technetwork/middleware/data-integrator/overview/odigs-11g-168072.pdf, zadnjič obiskano: 12.6.2013

[4] Laura Hofman Miquel, *Getting Started with Oracle Data Integrator*, Oracle, 2010

[5] Adrian Sergio Pulvirenti, María Carina Roldán, *Pentaho Data Integration 4 Cookbook*, Packt, 2011

[6] Wikipedia, *MySQL*, http://en.wikipedia.org/wiki/MySQL, zadnjič obiskano: 12.8.2013

[7] Bilab, *Integracija podatkov - ključni proces BI projekta*, http://www.bilab.si/?show=content&id=12&men=16&oce=13, zadnjič obiskano: 29.8.2013

[8] Fon Silvers, *Building and Maintaining a Data Warehouse*, Auerbach, 2008

[9] Erhard Rahm, Hong Hai Do, *Data cleaning: Problems and Current Approaches*, http://dc-pubs.dbs.uni-leipzig.de/files/Rahm2000DataCleaningProblemsand.pdf, zadnjič obiskano: 5.7.2013

[10] P. Ziegler, Klaus R. Dittrich, *Data Integration - Problems, Approaches and Perspectives*, University of Zurich

[11] Alkis Simitsis, *Modeling and managing ETL processes*, http://www.dblab.ece.ntua.gr/pubs/uploads/TR-2003-11.pdf, zadnjič obiskano: 25.8.2013

[12] Ralph Kimball, Joe Caserta, *The Data Warehouse ETL Toolkit*, Wiley, 2004

[13] Wikipedia, *Extract, transform, load*, http://en.wikipedia.org/wiki/Extract,_transform,_load, zadnjič obiskano: 10.8.2013

[14] Darko Jagarinec, *OLAP in podatkovna skladišča*, http://www.bilab.si/uploads/clanki/arhivirana_datoteka_5.pdf, zadnjič obiskano: 14.8.2013

[15] ETL Tools, *Selection criteria for ETL tools*, http://www.etltool.com/categories-and-criteria-examined/, zadnjič obiskano: 24.7.2013

[16] Marko Hölbl, *Skladiščenje podatkov in poročanje*, FERI, Maribor, 2010

[17] Bilab, *Podatkovno skladišče*, http://www.bilab.si/?show=content&id=11&men=15&oce=13, zadnjič obiskano: 1.9.2013

[18] Shaker H. Ali El-Sappagh, A. M. Ahmed Hendawi, A. Hamed El Bastawissy, *A proposed model for data warehouse ETL processes*, http://www-users.cs.umn.edu/~hendawi/EMD%20at%20king%20Abdallah.pdf, zadnjič obiskano: 27.8.2013

[19] Oracle, *Knowledge Modules Developer's Guide*, http://www.oracle.com/technetwork/testcontent/oracledi-km-development-129055.pdf, 2006, zadnjič obiskano: 5.7.2013

[20] Trianzblog, *Data Integration: ETL vs. ELT*, http://trianzblog.com/wordpress/?p=203, zadnjič obiskano: 23.8.2013

[21] ETL-Tools.Info, *Pentaho Data Integration – Kettle ETL Tool*, http://etl-tools.info/en/pentaho/kettle-etl.htm, zadnjič obiskano: 18.7.2013

[22] Izidor Golob, Tatjana Welzer, *Arhitekture podatkovnih skladišč*, FERI, Maribor

[23] Craig Stedman, *Decision time: Automated data integration tools versus manual coding*, http://searchdatamanagement.techtarget.com/feature/Decision-time-Automated-data-integration-tools-versus-manual-coding, zadnjič obiskano: 30.8.2013

[24] Pentaho, *Pentaho Data Integration*, http://www.pentaho.com/explore/pentaho-data-integration/, zadnjič obiskano: 3.6.2013

[25] Matt Mason, *Writing to a MySQL database from SSIS*, http://blogs.msdn.com/b/mattm/archive/2009/01/07/writing-to-a-mysql-database-from-ssis.aspx, 2009, zadnjič obiskano: 26.8.2013

8 PRILOGE

8.1 Kazalo slik

8.2 Kazalo preglednic

8.3 Kazalo poizvedb

8.4 Vsebina zgoščenke:

- Diplomska naloga v elektronski obliki
- Podrobnejše slike procesov integracij in E-R diagramov
- Projekti testiranih ETL orodij

8.5 Podatki študenta:

Ime in priimek:	Tomaž Hrnčič
Naslov:	Rimska ploščad 3, Ptuj
Pošta:	2250, Ptuj
E-pošta:	tomaz.hrncic@gmail.com

8.6 Kratek življenjepis:

Rojen:	29.8.1987 na Ptuju
Šolanje:	Osnovna šola - Mladika na Ptuju (1994 - 2002)
	Srednja šola – SŠC Ptuj, Poklicna in Tehniška Elektro Šola, Smer – Računalniški tehnik (2002 - 2006)
	Visoka Šola – FERI Maribor, Smer – Informatika in Tehnologije Komuniciranja

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Tomaž Hrnčič,

z vpisno številko E1009026,

sem avtor/-ica diplomskega dela z naslovom:

Analiza in primerjava orodij za integracijo podatkov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)
  doc. dr. Marko Hölbl univ. dipl. inž. rač. in inf.

  in somentorstvom (naziv, ime in priimek)
  mag. Andrej Sevčnikar univ. dipl. inž. rač. in inf.
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v DKUM.

V Mariboru, dne 9. 9. 2013

Podpis avtorja/-ice:
Tomaž Hrnčič

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O USTREZNOSTI DIPLOMSKEGA DELA

Podpisani mentor dr. Marko Hölbl izjavljam, da je
(ime in priimek mentorja)
študent Tomaž Hrnčič izdelal diplomsko
(ime in priimek študenta-tke)

delo z naslovom: Analiza in primerjava orodij za integracijo podatkov

(naslov diplomskega dela)

v skladu z odobreno temo diplomskega dela, Navodili o pripravi diplomskega dela in mojimi navodili.

Datum in kraj: 9.9.2013

Podpis mentorja:

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE DIPLOMSKEGA DELA IN OBJAVI OSEBNIH PODATKOV DIPLOMANTOV

Ime in priimek diplomanta-tke: Tomaž Hrnčič

Vpisna številka: E1009026

Študijski program: VS Informatika in tehnologije komuniciranja

Naslov diplomskega dela: Analiza in primerjava orodij za integracijo podatkov

Mentor: doc. dr. Marko Hölbl univ. dipl. inž. rač. in inf.

Somentor: mag. Andrej Sevčnikar univ. dipl. inž. rač. in inf.

Podpisani-a Tomaž Hrnčič izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zakljucnega dela v Digitalno knjižnico Univerze v Mariboru.
Diplomsko delo sem izdelal-a sam-a ob pomoci mentorja. V skladu s 1. odstavkom 21. clena Zakona o avtorskih in sorodnih pravicah (Ur. l. RS, št. 16/2007) dovoljujem, da se zgoraj navedeno zakljucno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.

Tiskana verzija diplomskega dela je istovetna elektronski verziji, ki sem jo oddal za objavo v Digitalno knjižnico Univerze v Mariboru.

Podpisani izjavljam, da dovoljujem objavo osebnih podatkov vezanih na zakljucek študija (ime, priimek, leto in kraj rojstva, datum diplomiranja, naslov diplomskega dela) na spletnih straneh in v publikacijah UM.

Datum in kraj: 9.9.2013, MARIBOR

Podpis diplomanta-tke: Tomaž Hrnčič