UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Mladen Babić

PODATKOVNO SKLADIŠČE ZA SPREMLJANJE POSLOVANJA S TRANSAKCIJSKIMI RAČUNI

DIPLOMSKA NALOGA
NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, 2008

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Mladen Babić

PODATKOVNO SKLADIŠČE ZA SPREMLJANJE POSLOVANJA S TRANSAKCIJSKIMI RAČUNI

DIPLOMSKA NALOGA
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Viljan Mahnič

Ljubljana, 2008

Zahvala

Zahvaljujem se vsem sodelavcem v podjetju SRC.SI, ki so me vpeljali v področje podatkovnega skladiščenja in mi z veseljem podajali svoje znanje. Hvaležen sem tudi mentorju prof. dr. Viljanu Mahniču za vodenje in koristne nasvete pri pisanju diplomskega dela. Prijateljem, Lini in Miroslavu, se zahvaljujem za lektoriranje diplome in vso podporo, ki jo lahko dajo le pravi prijatelji. Hvala Renatini in moji družini za spodbude in udobje, ki sem jih bil deležen v času celotnega študija. Nazadnje gre posebna zahvala moji Renati, ki mi je z veliko ljubezni, vere in potrpežljivosti pomagala premagati marsikatero oviro v času študija.

Kazalo

Kazalo slik

Seznam kratic

CIF (*angl. Corporate Information Factory)* – Inmon-ova arhitektura podatkovnega skladišča
ER (*angl. Entity – Relationship*) – entitetno – relacijski model
ETL (*angl. Extract, Transform, Load*) – zajemanje, transformacija in polnjenja podatkov v podatkovno skladišče
OLAP (*angl. Online Analytic Processing*) – sprotna analitična obdelava podakov
OLTP (*angl. Online Transaction Processing*) – sistem za sprotno obdelavo transakcij
PS (*angl. Data Warehouse*) – podatkovno skladišče
PPS (*angl. Data Mart*) – področno podatkovno skladišče
SQL (*angl. Structured Query Language*) – strukturiran povpraševalni jezik za delo s podatkovnimi bazami
TRR (*angl. Bank Transaction Account*) – transakcijski račun

Povzetek

V tem delu predstavimo postopek izdelave prototipa podatkovnega skladišča za spremljanje poslovanja s transakcijskimi računi občanov. Kot uvod v obsežno področje podatkovnega skladiščenja podamo teoretične osnove področja, ki vsebujejo osnovno terminologijo, opis osnovnih gradnikov podatkovnega skladišča, kot tudi opis uporabljene razvojne metodologije. Nadaljujemo z načrtovanjem dimenzijskih modelov podatkovnega skladišča. Izdelamo tri dimenzijske podatkovne modele, s pomočjo katerih predstavimo poslovanje s transakcijskimi računi in pokažemo, kako se le – to vklaplja v celovito rešitev za analizo bančnih storitev. Pri načrtovanju modelov se držimo priporočil iz literature, ki se nanašajo na izgradnjo zvezdnih shem v primeru različnih izdelkov in storitev ter spremljanje posameznih transakcij in periodičnih posnetkov stanja. Tako načrtovani dimenzijski modeli omogočajo analiziranje poslovanja s transakcijskimi računi na različnih nivojih podrobnosti. Pri realizaciji sistema ETL, ki skrbi za osveževanje podatkovnega skladišča, damo poseben poudarek na razvoj postopkov ETL, avtomatizacijo njihovega izvajanja, ter rokovanje z napakami, ki se pojavljajo ob izvajanju postopkov ETL. Delo zaključimo s podajanjem primerov uporabe podatkovnega skladišča za spremljanje poslovanja s transakcijskimi računi. Možne razširitve in smernice nadaljnjega razvoja podatkovnega skladišča podamo v sklepnih ugotovitvah.

Ključne besede: podatkovno skladišče, področno podatkovno skladišče, dimenzijski model, sistem ETL, transakcijski račun.

Abstract

In this work we present implementation of a data warehouse prototype for bank transaction accounts business analysis. As an introduction to extensive data warehousing area we provide some theoretical bases, which include basic terminology, a description of essential elements of data warehouse as well as a description of development methodology we use. We proceed with designing dimensional models of the data warehouse. We design three dimensional data models, which represent bank transaction accounts business activities. We also show integration of designed models into complete bank product analysis solution. When designing models, we use literature recommendations for designing heterogeneous products star schemas, transaction star schemas and periodic snapshot star schemas. By using these recommendations, we design dimensional models, which enable transaction accounts business analysis on different grain levels. The next step in the process of making the data mart is carrying out the ETL system, part of the system that is responsible for refreshing the data warehouse. We develop ETL processes, module for automatic execution of the ETL processes, and module for ETL process error handling. At the end of our work we provide some use cases of the data warehouse for bank transaction accounts. Possible upgrades and guidelines for the future developments of the data warehouse for bank transaction accounts are given in the conclusion.

Keywords: data warehouse, data mart, dimensional model, ETL system, bank transaction account.

1. Uvod

Banke se danes srečujejo s številnimi poslovnimi izzivi. Po eni strani so to izzivi, ki jih prinaša trg, ki se stalno spreminja in v zadnjem času vedno bolj globalizira. Tudi konkurenca na trgu je vedno večja. Po drugi strani smo tudi stranke vedno bolj zahtevne. Želimo biti obravnavane individualno ter želimo nam prilagojeno ponudbo. Da bi zadostile svojim strankam in pridobile nove stranke, morajo banke ponujati nove, sodobne bančne produkte, kot tudi produkte izven njihovega klasičnega asortimana. Banke se v želji po večjem tržnem deležu, manjših stroških, večji učinkovitosti in uspešnosti združujejo oziroma prevzemajo manjše banke. Nenazadnje, da bi poslovale uspešno, morajo banke stalno nadzirati tudi tveganja, s katerimi se srečujejo. Za uspešen odziv na naštete poslovne izzive, so ključnega pomena pravočasne in točne poslovne informacije, do katerih je potrebno zagotoviti dostop tako na strateškem, kot tudi na operativnem nivoju. Operativni sistemi, ki zajemajo podatke na operativni ravni, v večini primerov niso primerni za podporo procesom odločanja. Zaradi tega se banke odločajo za izgradnjo sistemov poslovnega obveščanja. Poslovno obveščanje je sistem, ki omogoča analizo podatkov o poslovanju organizacije in analizo posledic sprejetih odločitev. Osrednji del sistema za poslovno obveščanje je podatkovno skladišče, ki predstavlja enotno shrambo vseh pomembnih podatkov o poslovanju organizacije. Podatkovno skladišče se redno osvežuje s podatki iz izvornih sistemov z izvajanjem postopkov ETL, ki zagotovijo kakovost in integriranost podatkov. Podatke podatkovnega skladišča navsezadnje lahko uporabljamo za različna poročanja, poizvedovanja, OLAP kocke, analitične aplikacije, podatkovno rudarjenje in drugo.

V tem diplomskem delu izdelamo prototip podatkovnega skladišča, ki omogoča analizo poslovanja s transakcijskimi računi občanov. Kot uvod v področje podatkovnega skladiščenja, podamo osnovne definicije in cilje podatkovnega skladišča. Izdelava podatkovnega skladišča je izredno obsežna naloga, ki zahteva dobro poznavanje vsebine poslovnega področja in ustrezne razvojne metodologije. Pri razvoju prototipa podatkovnega skladišča se držimo Kimballove razvojne metodologije [4, 5, 6], ki temelji na iterativnem razvoju in integraciji manjših, vsebinsko omejenih delov podatkovnega skladišča, ki jih imenujemo področna podatkovna skladišča. Poleg Kimballovega pristopa opišemo tudi ostale pogosto uporabljene razvojne pristope ter poudarimo osnovne razlike med njimi. Kot bomo videli, je Kimballova arhitektura podatkovnega skladišča razdeljena na štiri dele in sicer izvorne sisteme, področje za pripravo podatkov, predstavitvene strežnike ter uporabnike.

Pri načrtovanju podatkovnih modelov za spremljanje poslovanja s transakcijskimi računi občanov uporabljamo dimenzijsko modeliranje. Dimenzijski podatkovni modeli so v denormalizirani obliki. Uporabnikom so, za razliko od normaliziranih modelov, razumljivi in intuitivni. Prvi problem, ki se pojavi ob načrtovanju dimenzijskih modelov na področju bančništva je, kako predstaviti poslovanje z določenim bančnim produktom, ki ima bogato transakcijsko zgodovino. Zaželjeno je, da podatkovno skladišče hrani podatke na najnižjem nivoju podrobnosti, to je na nivoju posameznih transakcij. Obstaja pa cela vrsta nujnih poslovnih vprašanj, katerih odgovore je težko povzeti neposredno iz transakcijske zgodovine. Drugi problem, ki se pojavi ob načrtovanju podatkovnega modela je, da banke ponujajo veliko število produktov, ki so po svoji naravi heterogeni. V takšnem okolju imamo dva pogleda na poslovanje s produkti: prilagojen pogled na poslovanje s posameznim bančnim produktom in globalen pogled na poslovanje z vsemi bančnimi produkti hkrati. Oba pogleda težko predstavimo v enem samem dimenzijskem modelu. Omenjene probleme rešimo tako, da izdelamo tri dimenzijske modele. Za potrebe spremljanja prometa transakcijskih računov občanov, izdelamo transakcijsko shemo, na osnovi katere lahko analiziramo poslovanje s transakcijskimi računi na najnižjem nivoju podrobnosti. Drugi dimenzijski model hrani

podatke na višjem nivoju, saj omogoča spremljanje dnevnih stanj na transakcijskih računih, ki jih izračunamo na osnovi prometa transakcijskih računov. Z drugim dimenzijskim modelom hkrati predstavimo tudi prilagojen pogled na posamezni bančni produkt, to je transakcijske račune. Zadnji dimenzijski model predstavlja globalni pogled na bančno poslovanje in omogoča spremljanje dnevnih stanj za poljubno vrsto bančnih produktov.

V četrtem poglavju izdelamo sistem ETL, ki skrbi za avtomatično osveževanje podatkovnega skladišča za spremljanje poslovanja s transakcijskimi računi. Najprej predstavimo strukturo področja za pripravo podatkov, komponento zalednega dela podatkovnega skladišča, ki je uporabnikom nedostopen. Področje za pripravo podatkov vsebuje podatkovne strukture, v katere začasno prenesemo izvorne podatke. Vsebuje tudi ročno kodirane postopke ETL, ki jih izdelamo na osnovi preslikav izvornih atributov v atribute ustreznih dimenzijskih tabel in tabel dejstev. Postopki ETL definirajo logiko zajemanja, preoblikovanja in čiščenja izvornih podatkov, ter njihovo polnjenje v podatkovno skladišče. Tipični problemi, ki se pojavljajo pri izdelavi teh postopkov, se nanašajo na zajemanje velike količine izvornih podatkov in uveljavljanje kakovosti podatkov. Za rešitev problema zajemanja velike količine izvornih podatkov ustvarimo v področju za pripravo podatkov začasne in arhivske tabele. Znotraj postopkov ETL uporabimo omenjene tabele za zajem le spremenjenih oziroma novih izvornih podatkov. Kakovost podatkov uveljavljamo s pomočjo izvajanja kontrol in transformacij izvornih podatkov. Kontrole so realizirane na osnovi vnaprej definirane množice pravil, ki jim izvorni podatki morajo zadoščati. S pomočjo transformacij izvornih podatkov zagotovimo, da podatki ustrezajo določenim standardom. Podatkovnega skladišča ne osvežujemo z nekakovostnimi izvornimi podatki, temveč takšne podatke zapisujemo skupaj z opisom napake v poseben del področja za pripravo podatkov. Na ta način skrbniki izvornih sistemov pridobijo povratno informacijo o nepravilnih podatkih, ki jih je potrebno popraviti. V praksi se velikokrat uporablja tudi povratno nalaganje kakovostnih podatkov podatkovnega skladišča nazaj v izvorne sisteme, ki ga v tem diplomskem delu ne obravnavamo. Avtomatično osveževanje podatkovnega skladišča zahteva opredelitev časovnega okvirja izvajanja postopkov ETL kot tudi definiranje vrstnega reda izvajanja posameznih korakov postopka ETL. Čas izvajanja posameznega postopka ETL določimo s pomočjo baznih opravil. Pravilni vrstni red izvajanja korakov postopka ETL zagotovimo z ustvarjanjem posebnih podatkovnih struktur, ki se nahajajo v posebnem delu področja za pripravo podatkov. Omenjene podatkovne strukture nam podajo tudi zgodovinski pregled polnjenja podatkovnega skladišča (dnevnik polnjenja podatkovnega skladišča), kot tudi informacijo o tem, iz katerega izvornega sistema podatke zajemamo pri izvajanju določenega postopka ETL.

Vsaka sprememba v poslovnem okolju predstavlja grožnjo za podjetja, ki se ne znajo hitro prilagoditi novim zahtevam trga in novim potrebam kupcev. Uporaba podatkovnega skladišča omogoča podjetjem, da hitro ugotovijo trende v poslovanju in se na njih hitro odzivajo. Hiter odziv na spremembe, lahko zagotovi podjetjem prednost pred ostalimi tekmeci in s tem tudi večji tržni delež. V bančnem sektorju se podatkovno skladišče tipično uporablja za upravljanje odnosov s strankami, upravljanje tveganj, uravnavanje likvidnosti, spremljanje dobičkonosnosti, interno poročanje in poročanje zunanjim institucijam. Prototip podatkovnega skladišča, ki ga zgradimo v tem diplomskem delu, je predvsem uporaben za analiziranje poslovanja s transakcijskimi računi občanov. V petem poglavju podamo nekaj možnih primerov uporabe prototipa podatkovnega skladišča.

2. Predstavitev podatkovnega skladiščenja

2.1 Osnovne definicije

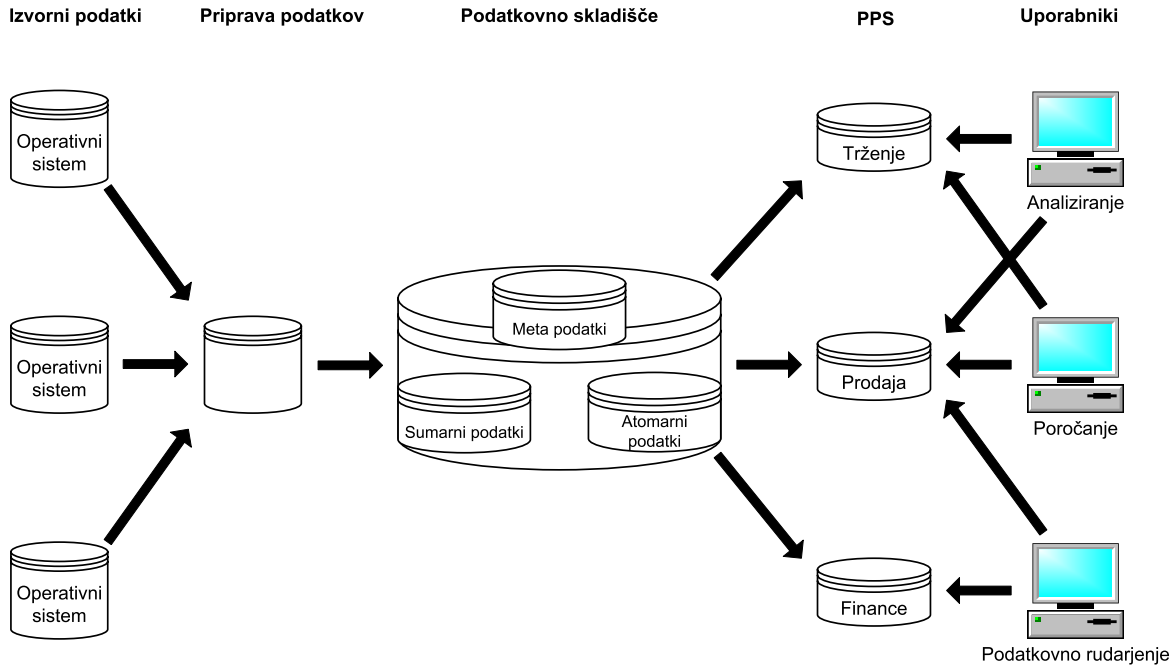
Izraz *podatkovno skladišče (PS)* je prvi uporabil "oče" podatkovnega skladiščenja William Inmon, ki ga je definiral kot [3]:

Podatkovno skladišče je subjektno usmerjena, integrirana, časovno odvisna in nespremenljiva zbirka podatkov ustvarjena zaradi podpiranja procesov odločanja.

Karakteristike PS podane v definiciji razložimo na sledeči način:

- *Subjektna usmerjenost*: podatki skladišča so organizirani okoli specifičnih subjektov, kot so stranka ali izdelek.
- *Integriranost*: podatki so zbrani iz različnih izvorov in so integrirani okoli subjektov v konsistentnem formatu.
- *Časovna odvisnost*: PS ohranja zgodovinske podatke, ki so potrebni za ugotavljanje trendov v poslovanju, deviacij in dolgoročnih odnosov.
- *Nespremenljivost*: uporabniki ne morejo spreminjati podatkov. PS se spreminja s kontroliranimi postopki polnjenja. Nespremenljivost zagotavlja, da vsi uporabniki delajo z istimi podatki.

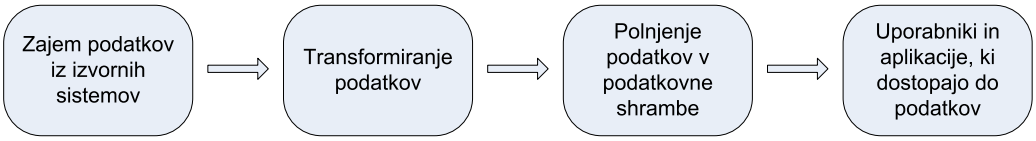
Primer splošnega PS je prikazan na sliki 1.



Slika 1. Primer podatkovnega skladišča.

Medtem ko PS predstavlja zbirko podatkov je *podatkovno skladiščenje* pojem, ki označuje celoten proces. Vključuje celo vrsto aktivnosti, od zajemanja podatkov iz izvornih sistemov, do uporabe podatkov pri procesih odločanja. Oziroma bolj natančno, kot to prikazuje slika 2,

vsebuje zajemanje, transformacijo in polnjenje podatkov, kot tudi dostop do podatkov s strani končnih uporabnikov in aplikacij.

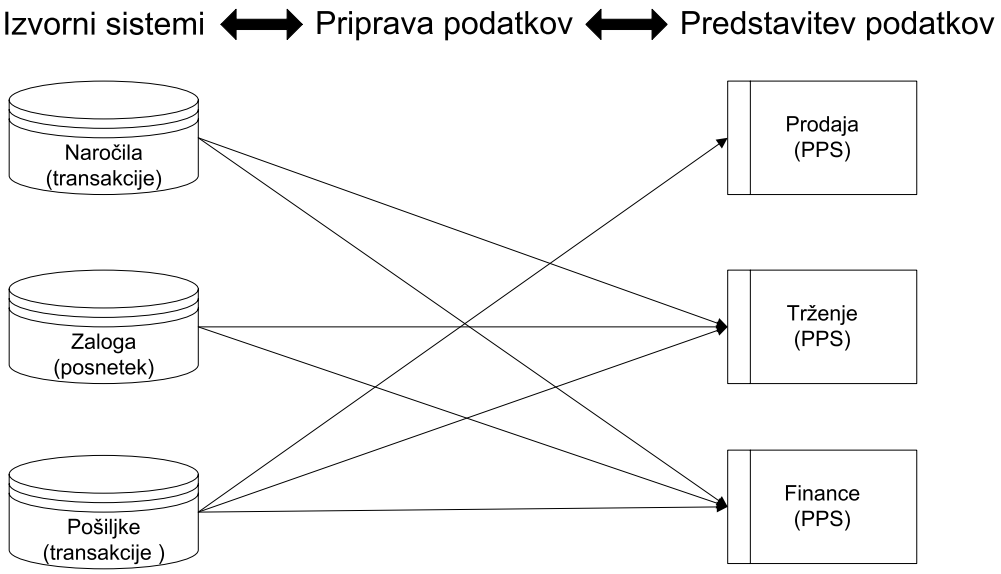


Slika 2. Proces podatkovnega skladiščenja.

Področno podatkovno skladišče (PPS) je podobno podatkovnemu skladišču, le da hrani podatke za določeno poslovno področje. Ker je manjšega obsega kot PS, tudi podpira manj aplikacij. Razlikujemo neodvisno in odvisno PPS:

- *Neodvisno PPS* je zgrajeno neposredno iz izvornih sistemov.
- *Odvisno PPS* je zgrajeno iz podatkov, izvlečenih iz PS.

Neodvisno PPS je ponavadi začasna rešitev in je zgrajeno na osnovi specifičnih zahtev, naprimer prodajnega oddelka. Čeprav se lahko izkaže za uspešno na nivoju oddelka, dolgoročno lahko predstavlja problem, ko se organizacija odloči zgraditi kompatibilno podatkovno infrastrukturo na nivoju celotnega podjetja [10]. Primer takega PPS je prikazan na sliki 3.



Slika 3. Primer neodvisnega PPS.

Odvisna PPS so zgrajena, da bi podala uporabnikom prilagojen pogled specifičnih podatkov, ki jih potrebujejo. Tako naprimer za potrebe finančnih analitikov zgradimo PPS, ki vsebuje samo finančne podatke. Ponavadi je povpraševanje bistveno hitrejše, kot v samem PS. Velika podjetja preferirajo odvisna PPS, ker podatki prihajajo iz izvora, to je PS, ki je bil zgrajen iz perspektive celotnega podjetja [10]. Na ta način je ohranjena "ena sama verzija resnice", ker vsi delajo z istimi podatki, čeprav se nahajajo na različnih mestih. Primer odvisnega PPS je prikazan na sliki 1.

2.2 Osnovni cilji podatkovnega skladišča

Osnovni cilji, ki jih mora izpolnjevati PS so [4]:

- *Informacija mora biti lahko dostopna***.** Vsebina PS mora biti razumljiva in čitljiva. Orodja za dostop do PS morajo biti enostavna za uporabo in morajo vračati rezultate povpraševanj v minimalnem času.
- *PS mora dosledno predstavljati informacije.* Podatki v PS morajo biti verodostojni in kakovostni preden jih predstavimo uporabnikom.
- *PS mora biti prilagodljivo in prožno.* PS mora biti načrtovano tako, da se lahko prilagaja spremenjenim uporabniškim potrebam, poslovnim okoliščinam, podatkom in tehnologijam.
- *PS mora ščititi informacijo.* PS mora učinkovito kontrolirati dostop do zaupnih informacij.
- *PS mora služiti kot osnova za izboljšan proces odločanja.*
- *Uporabniki morajo sprejeti PS.* Uspešnost PS merimo glede na sprejemanje s strani uporabnikov.

2.3. Alternativni razvojni pristopi

Čeprav so podatkovna skladišča danes široko razširjena, ne obstaja enotno mnenje katera razvojna metodologija je najboljša. Uporabljata se dva pristopa. Prvi je povezan z Billom Inmonom, pionirjem na področju podatkovnega skladiščenja, ki predlaga uporabo pristopa od zgoraj navzdol (angl. top – down). Drugi je povezan z Ralphom Kimballom, visoko cenjenim svetovalcem in avtorjem številnih knjig s področja podatkovnega skladiščenja. Kimball predlaga uporabo pristopa od spodaj navzgor (angl. bottom – up), ki temelji na iterativnem dodajanju posameznih PPS. Oba pristopa imata prednosti, ampak vsebujeta tudi omejitve. Preden podamo kratek pregled obeh pristopov in osnovnih razlik med njima, navedemo nekatera skupna izhodišča obeh avtorjev [8]:

- Skoraj vse organizacije imajo koristi od gradnje PS in analitičnega okolja kot podpore odločanju.
- Osnovni cilj vsakega PS je objaviti kakovostne podatke in jih narediti lahko dostopne uporabnikom in aplikacijam.
- Razumno je zajeti celotno organizacijo kot izhodiščno točko pri načrtovanju arhitekture PS iz razloga dolgoročne integracije in nadgradljivosti.
- Neodvisna PPS povzročajo težave, saj so zgrajena, da zadovoljijo specifične potrebe uporabnikov in ne upoštevajo ostale obstoječe ali planirane analitične podatke.

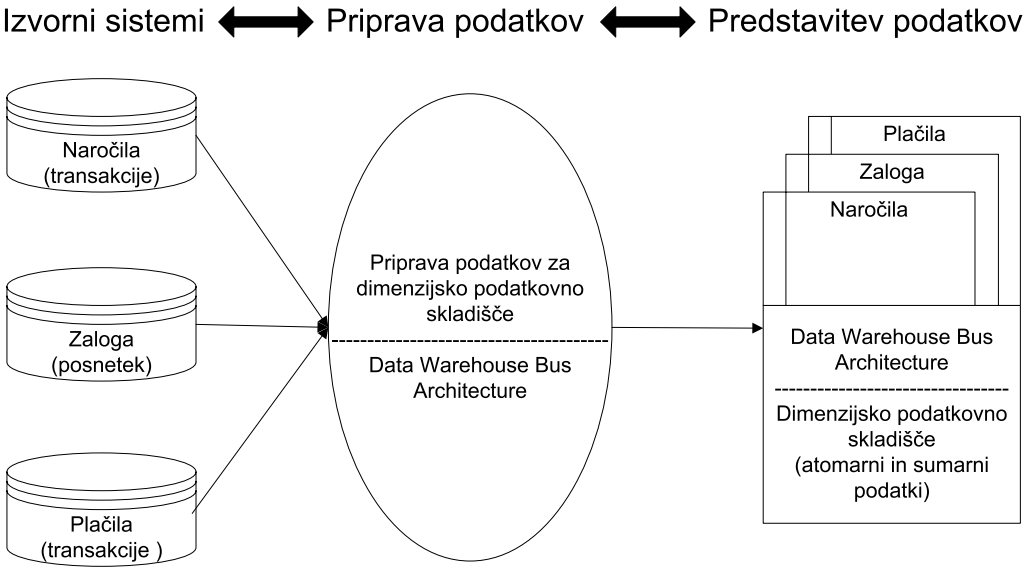
2.3.1 Kimballov pristop

Planiranje in razvoj celovitega PS (angl. enterprise data warehouse) je izredno obsežna in težka naloga. Kimball priporoča razdelitev naloge na obvladljive podnaloge, katerih rešitve potem združujemo v celoto. Vodilna misel tega pristopa je: “Start small, think big“. To pomeni, da imamo pri planiranju razvoja PS v mislih arhitekturo, ki omogoča razvoj in

implementacijo celovitega PS, hkrati se osredotočimo na izpolnjevanje manjših, lažje rešljivih nalog – na razvoj posameznih PPS.

Pri Kimballovem pristopu najprej zgradimo PPS, ki omogočajo poročanje in analiziranje za specifične poslovne procese. PPS vsebujejo atomarne podatke (podatke na najnižjem nivoju podrobnosti) in po potrebi še sumarne podatke. Navsezadnje so PPS združena v obširno PS. Združitev PPS realiziramo preko skupnega "vodila", implementacije, ki jo Kimball imenuje The Data Warehouse Bus Architecture. Začetno PPS črpa podatke iz majhnega števila izvornih sistemov. Zaradi svojega omejenega obsega, se lahko PPS hitro zgradi po razmeroma nizki ceni in preskrbi hitro vračilo vložka. Če se PPS izkaže za uspešno, iterativno dodajamo nova PPS, oziroma nadgradimo obstoječa. PPS so med seboj povezana preko medsebojno skladnih skupnih dimenzij in merljivih dejstev. Z uporabo le teh uporabniki lahko povprašujejo po vseh PPS hkrati.

Kot je razvidno iz Kimballove arhitekture PS prikazane na sliki 4, so izvorni podatki transformirani v predstavljivo kakovostno informacijo v področju za pripravo podatkov (angl. data staging area). Priprava podatkov se začne z usklajenimi zajemi podatkov iz izvornih sistemov, nakar jih prečistimo, medsebojno uskladimo, preoblikujemo in združimo. Tako pripravljene podatke napolnimo v predstavitveno področje. Predstavitveno področje je dimenzijsko strukturirano. Dimenzijski model vsebuje isto informacijo kot normalizirani model, ampak je bolj intuitiven in optimizira povpraševanje. Dimenzijsko modeliranje bolj podrobno obravnavamo v tretjem poglavju.

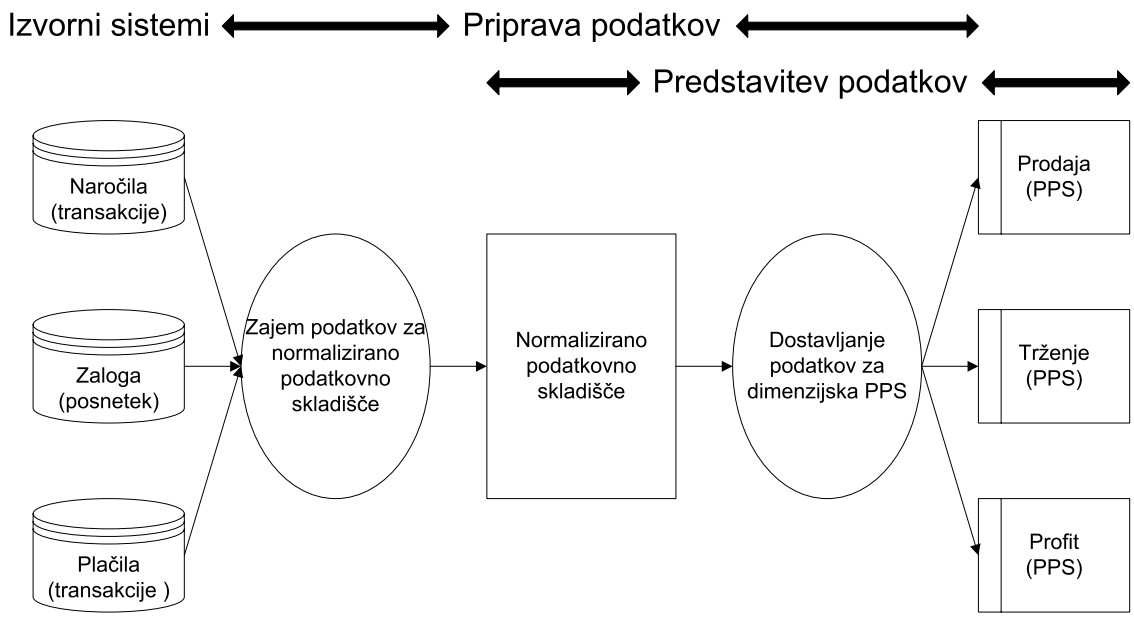


Slika 4. Kimballova arhitektura PS - The Data Warehouse Bus Architecture.

Dolgoročno tveganje pri uporabi tega pristopa so nekonsistence, ki se pojavljajo v PPS in s tem tudi morebitne "različne verzije resnice" videne s strani uporabnikov. Strogo vzdrževanje medsebojno skladnih skupnih dimenzij in merljivih dejstev pomaga pri ublaževanju takih tveganj.

2.3.2 Inmonov pristop

V večini velikih organizacij je razvoj celovitega PS željeni končni cilj. Slika 5 prikazuje Inmonovo Corporate Information Factory (CIF) arhitekturo PS.



Slika 5. Inmonova arhitektura PS – Corporate Information Factory (CIF).

Podobno kot pri Kimballovem pristopu, se tudi pri Inmonovem pristopu uporabljajo usklajeni zajemi podatkov iz izvornih sistemov. Podatki se polnijo v normalizirano relacijsko podatkovno bazo, ki vsebuje atomarne podatke. Normalizirano PS se potem uporablja kot izvor za dimenzijsko modelirana PPS, ki vsebujejo sumarne podatke potrebne za specifične poslovne procese. Očitno je, da so atomarni podatki predstavljeni bistveno drugače kot sumarni podatki.

Inmonova metodologija načrtovanja PS generira visoko konsistentne dimenzijske poglede na podatke v PPS, saj so vsa PPS napolnjena iz istega izvora – normaliziranega PS. Poleg tega je tako načrtovanje PS odporno na poslovne spremembe. Tudi gradnja novih PPS je relativno lahka naloga. Glavna pomankljivost pristopa je, da predstavlja izredno obsežen projekt na obsežnem področju. Zahteva tudi velike denarne in časovne stroške preden imajo uporabniki koristi od podatkovnega skladišča.

2.3.3 Osnovne razlike med Kimballovim in Inmonovim pristopom

Preden podamo osnovne razlike med pristopoma, naj omenimo, da obe strategiji pri pogoju, da sta uspešno izvršeni, kot rezultat dajeta integrirano celovito PS.

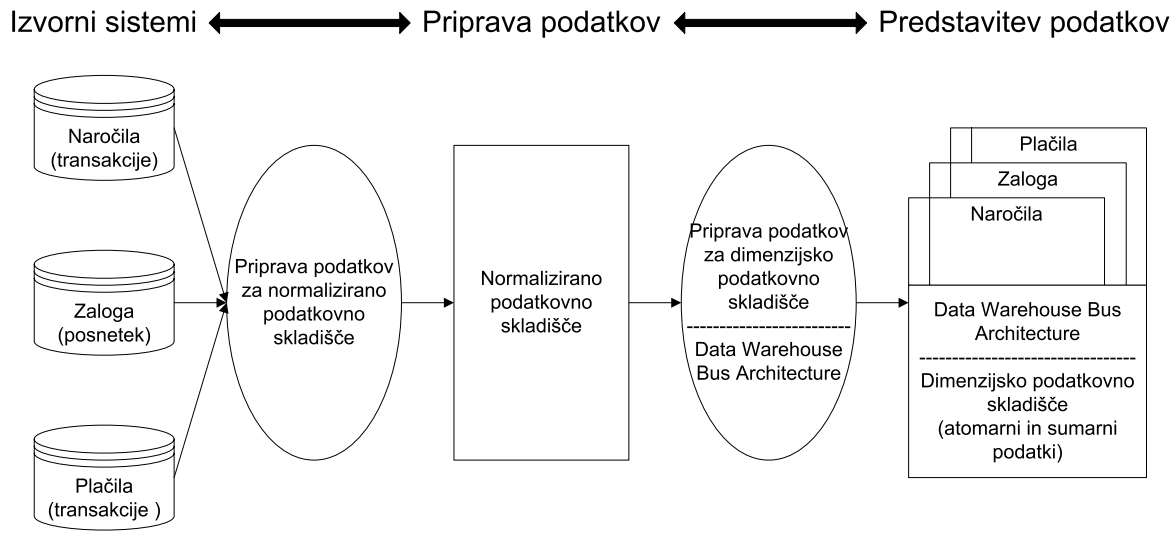
Takoj opazimo dve osnovni razliki med Kimballovim in Inmonovim pristopom. Prva zadeva potrebe po normalizirani podatkovni strukturi pred polnjenjem v dimenzijsko zasnovane PPS, ki se pojavlja pri Inmonovi CIF arhitekturi. Zagovorniki normaliziranih podatkovnih struktur trdijo, da se lahko hitreje napolnijo kot dimenzijski model. Kimball dvomi v korist tovrstne rešitve, ker so podatki podvrženi večkratnim postopkom ETL, preden so predstavljeni uporabnikom. Druga poglavitna razlika je v ravnanju z atomarnimi podatki. Inmon trdi, da morajo biti shranjeni v normaliziranem PS, medtem ko Kimball trdi da morajo biti dimenzijsko strukturirani. Poslovne uporabnike morebiti ne zanimajo podrobnosti posamezne atomarne transakcije, ampak ne moremo predvideti načine, na katere želijo povzeti transakcijske aktivnosti. Še več, takšni povzetki oziroma vprašanja, ki zanimajo poslovne uporabnike se nepredvidljivo in stalno spreminjajo. Kimball navaja, da je prednost hranjenja atomarnih podatkov v dimenzijskih strukturah ta, da lahko vedno zelo hitro in na

človeku intuitiven način povzamemo podatke na katerikoli način. Če so sumarni podatki dimenzijsko strukturirani in atomarni podatki v normaliziranih strukturah, kot je to pri CIF arhitekturi, potem vrtanje v globino (angl. drill-down) zna biti težavno, ker so spodaj ležeče strukture zelo različne.

Razlik, ki se pojavljajo med obema pristopoma je še precej in lahko rečemo, da so v svetu podatkovnega skladiščenja skozi čas izbruhnile prave "verske vojne" med privrženci obeh pristopov. A ker tema tega diplomskega dela ni podajanje razlik med pristopoma na tem mestu končamo z njihovo obravnavo.

2.3.4 Hibridni pristop

Poglavje, ki obravnava različne pristope končamo s kratko predstavitvijo hibridnega pristopa [8]. Nekatere organizacije privzamejo tak pristop in teoretično združijo najbolje od obeh pristopov opisanih v razdelikih 2.3.1 in 2.3.2. Kot to prikazuje slika 6, hibridno PS vsebuje normalizirano PS iz CIF arhitekture, kot tudi dimenzijsko PS atomarnih in sumarnih podatkov iz Kimballove arhitekture. Postavlja se seveda vprašanje ali je uporabljanje takega pristopa upravičeno glede na rastoče stroške in časovne zastoje povezane s redundančnimi postopki ETL in hranjenjem atomarnih podatkov.



Slika 6. Arhitektura hibridnega PS.

2.4 Razlike med sistemi OLTP in podatkovnimi skladišči

Eno izmed največjih bogastev, ki ga poseduje katerakoli organizacija je informacija. Le – ta se znotraj organizacije hrani v dveh oblikah: sistemih OLTP in podatkovnih skladiščih. Grobo povedano, skozi operativne sisteme OLTP pridobimo podatke in iz podatkovnih skladišč jih jemljemo. Tako naprimer uporabniki sistema OLTP jemljejo naročila, vpisujejo nove stranke, zapisujejo pritožbe. Naenkrat imajo opravka samo z enim zapisom. Konstantno opravljajo ene in iste naloge. Uporabniki podatkovnega skladišča štejejo nova naročila in jih primerjajo z naročili preteklega meseca. Zanima jih vzrok pridobitve

novih strank ali vzrok pritožb strank. Njihova vprašanja, ki se stalno spreminjajo, zahtevajo več sto ali tisoč zapisov, ki so združeni v smiselne odgovore.

OLTP je sistem za sprotno obdelavo transakcij. Le – ta zapisuje transakcije v podatkovno bazo v realnem času, ki so potem takoj vidne vsem uporabnikom. Sistemi OLTP in podatkovna skladišča so zgrajeni na osnovi zelo različnih zahtev. V nadaljevanju poudarimo nekaj bistvenih razlik med njima [9]:

1) Delovna obremenitev:
- PS je načrtovano za povpraševanje in iz tega razloga tudi optimizirano za veliko število različnih poizvedbenih operacij.
- Sistem OLTP podpira le predhodno določene operacije. Aplikacije so iz tega razloga načrtovane in optimizirane samo za te operacije.

2) Spremembe podatkov:
- PS se redno spreminja s postopki polnjenja in ga končni uporabniki ne posodabljajo sami.
- V sistemu OLTP končni uporabniki redno spreminjajo podatkovno bazo z izdajo posameznih stavkov za spremembo podatkov (vstavljanje, brisanje, sprememba zapisa). Baza vedno odraža trenutno stanje vsake poslovne transakcije.

3) Načrtovanje sheme:
- PS pogosto uporablja denormalizirane ali delno denormalizirane sheme zaradi optimizacije povpraševanja.
- Sistem OLTP pogosto uporablja popolnoma normalizirane sheme iz razloga optimizacije vstavljanja, brisanja in spreminjanja zapisov ter zagotavljanja konsistentnosti podatkov.

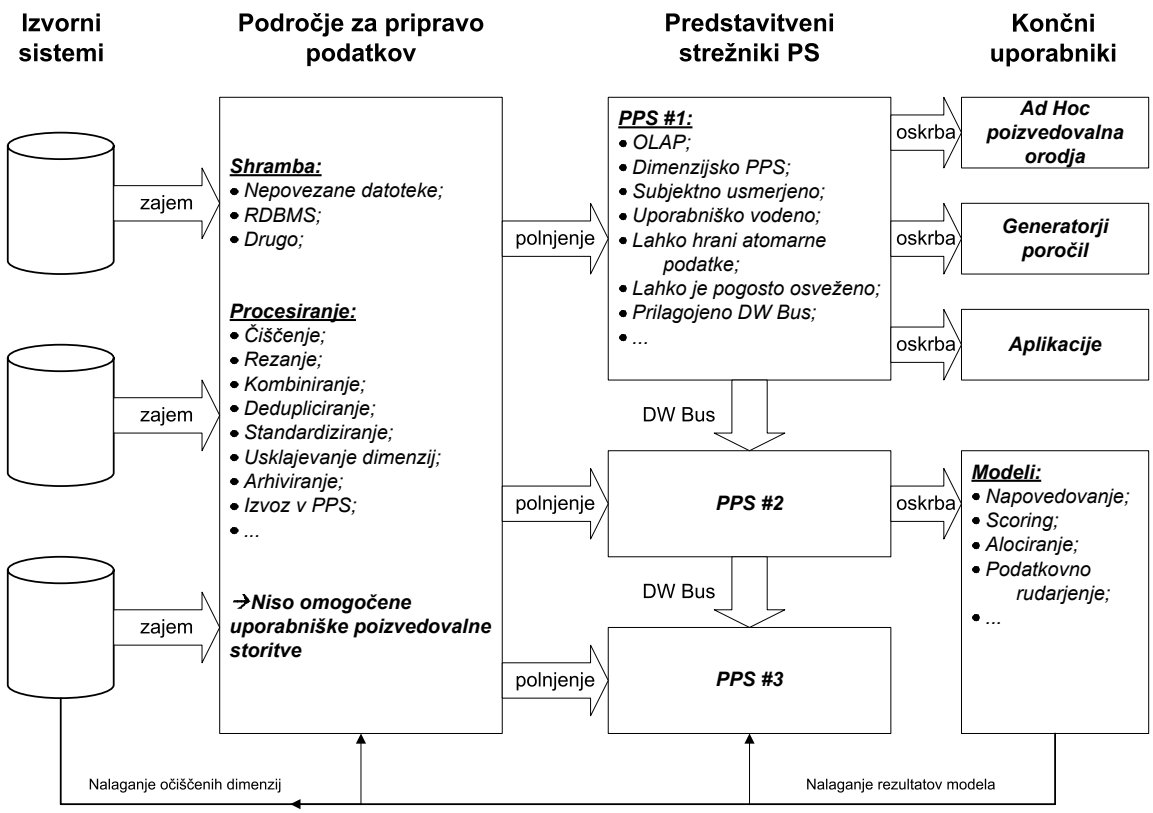
4) Tipične operacije:
- Tipična poizvedba v PS pregleduje več tisoč ali milijonov zapisov.
- Tipična OLTP operacija dostopa do peščice zapisov.

5) Zgodovinski podatki:
- PS hrani podatke za veliko mesecev ali let zaradi podpore zgodovinskih analiz.
- Sistem OLTP hrani podatke nekaj tednov ali mesecev, le zaradi potrebe obdelave trenutne transakcije.

Iz podanih razlik lahko ugotovimo da sistemi OLTP v večini primerov niso primerni za podporo procesom odločanja, ker so v osnovi načrtovani in optimizirani za drugačne namene.

2.5 Osnovni gradniki podatkovnega skladišča

V nadaljevanju pričujoče diplomske naloge se odločimo bolj podrobno preučiti Kimballov pristop načrtovanja PS. Podamo definicije osnovnih gradnikov PS in ostalih pojmov, ki se pogosto pojavljajo pri načrtovanju PS. Določene definicije, ki smo jih že podali v prejšnjih poglavjih ponovimo tako, da ustrezajo definicijam, ki jih Ralph Kimball podaja v [4]. Ker na tržišču v tem trenutku ne obstaja splošni dogovor glede definicij gradnikov PS, se v nadaljevanju diplomske naloge držimo poimenovanj in definicij podanih v tem poglavju. Osnovne gradnike PS prikažemo na sliki 7.



Slika 7. Osnovni gradniki podatkovnega skladišča.

2.5.1 Izvorni sistem

Izvorni sistem je operativni sistem, katerega naloga je zajemanje poslovnih transakcij. Glavni prioriteti izvornega sistema sta neprekinjeno delovanje in razpoložljivost. Običajno ohranjajo malo zgodovinskih podatkov. Analiziranje in poročanje predstavlja veliko breme za tovrstne sisteme. V večini primerov imajo organizacije več izvornih sistemov, ki nimajo med seboj prilagojenih oziroma na nivoju celotne organizacije definiranih osnovnih dimenzij, kot so produkt, stranka ali čas. Lahko mislimo, da se izvorni sistem nahaja izven samega PS, saj nimamo kontrole nad vsebino in obliko podatkov, ki jih iz njih pridobimo.

2.5.2 Področje za pripravo podatkov

Področje za pripravo podatkov sestavljajo shranjevalno področje in množica procesov, ki čistijo, transformirajo, kombinirajo, deduplicirajo, arhivirajo in pripravljajo izvorne podatke za uporabo v PS. Predstavljamo si, da zajema vse med izvornim sistemom in predstavitvenim strežnikom (glej sliko 7). Glavna omejitev, ki velja za področje za pripravo podatkov je, da ne ponuja poizvedovanja in predstavitvenih storitev.

2.5.3 Predstavitveni strežnik

Predstavitveni strežnik je ciljna strojna oprema na kateri so podatki PS shranjeni in organizirani z namenom direktnega poizvedovanja s strani končnih uporabnikov, razvijalcev poročil in aplikacij. Podatki so predstavljeni in hranjeni v dimenzijskem ogrodju. Če predstavitveni strežnik temelji na relacijski bazi, bodo tabele organizirane v obliki zvezdnih shem.

2.5.4 Področno podatkovno skladišče

PPS je logična podmnožica celotnega PS. PPS omeji PS na posamezen poslovni proces ali množico poslovnih procesov, ki so pomembni posebni poslovni skupini ali oddelku. Vsako PPS mora biti predstavljeno z dimenzijskim modelom in zgrajeno s pomočjo prilagojenih dimenzij in dejstev. Le – ta omogočajo kombiniranje in skupno uporabo PPS, saj imajo prilagojene dimenzije in dejstva v vseh PPS isti pomen. PPS vsebujejo atomarne podatke in po potrebi tudi sumarne podatke, ki so zgrajeni z namenom optimiziranja samega poizvedovanja.

2.5.5 Podatkovno skladišče

PS je unija vseh svojih sestavnih PPS. Predstavlja vir podatkov organizacije nad katerim se lahko izvajajo različne poizvedbe. PS se redno posodablja s kontroliranimi postopki polnjenja.

2.5.6 OLAP

OLAP je splošna aktivnost poizvedovanja in predstavljanja tekstualnih in numeričnih podatkov iz PS, kot tudi specifičen dimenzijski način poizvedovanja in predstavljanja podatkov s pomočjo kocke OLAP. OLAP je pristop, ki omogoča hitre odgovore na analitična vprašanja, ki so po svoji naravi večdimenzijska. Ponavadi ga uporabljamo v poslovnem poročanju za prodajo, trženje, finančno poročanje, poročanje vodstvu, napovedovanje in podobno. V jedru vsakega sistema OLAP je koncept kocke OLAP, oziroma bolj pravilno večdimenzijske ali hiper kocke. Sestavljena je iz numeričnih dejstev, ki so kategorizirana po dimenzijah. Dejstva so izpeljana iz zapisov tabele dejstev, dimenzije pa so izpeljane iz dimenzijskih tabel. Rezultat poizvedbe OLAP je tipično prikazan v matrični obliki. Dimenzije so predstavljene v vrsticah in stolpcih, dejstva pa kot vrednosti matrike.

Pri zahtevnih poizvedbah kocke OLAP lahko podajo odgovor tudi do 1000 krat hitreje kot sistemi OLTP. Razlog je v uporabi agregatov. Agregati so zgrajeni iz tabele dejstev s spreminjanjem zrna pri določenih dimenzijah in združevanju podatkov po teh dimenzijah. Število možnih agregatov je določeno z vsemi možnimi kombinacijami dimenzijskih zrnatosti. Vsi možni agregati vsebujejo odgovore na katerokoli vprašanje, na katero je možno odgovoriti iz podatkov. Ker je število agregatov, ki jih je potrebno izračunati ponavadi veliko, se izračuna le vnaprej določeno število agregatov.

2.5.7 Aplikacija končnega uporabnika

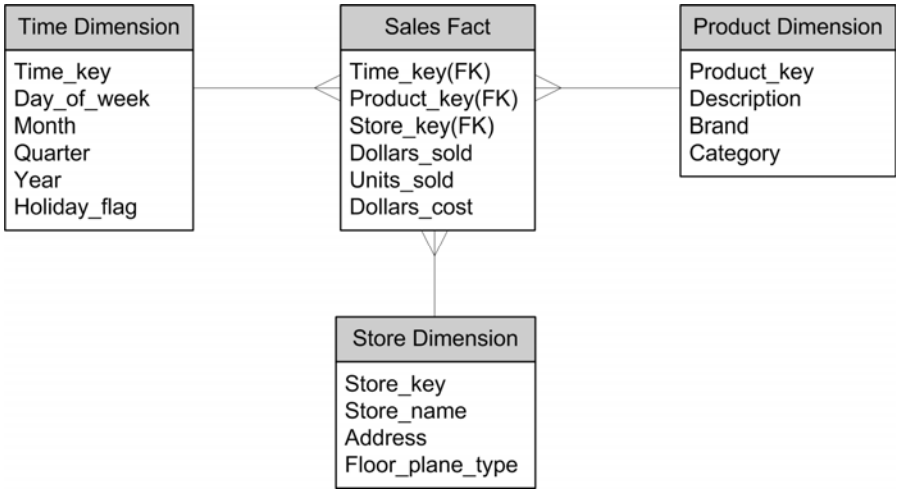
Aplikacija končnega uporabnika je zbirka orodij namenjena poizvedovanju, analiziranju in predstavljanju informacije s ciljem podpore neke poslovne potrebe. Minimalna množica takšnih orodij je sestavljena iz orodja za dostop do podatkov, preglednice, grafičnega paketa in uporabniškega vmesnika. Orodje, ki omogoča končnemu uporabniku dostop do podatkov, je odjemalec PS. V relacijskem PS, odjemalec vzdržuje sejo s predstavitvenim strežnikom, znotraj katere mu pošilja ločene zahtevke SQL. Rezultat zahtevka SQL potem prikaže uporabniku v razumljivi obliki (naprimer obliki poročila ali grafa). Orodja, namenjena končnemu uporabniku PS, varirajo od zelo enostavnih ad hoc poizvedovalnih orodij do zelo obsežnih orodij za modeliranje. Ad hoc poizvedovalna orodja omogočajo uporabniku oblikovanje lastnih vprašanj SQL in so zelo uporabno orodje za poslovne analitike, ki dobro poznajo vsebino PS. Orodja za modeliranje so prefinjena vrsta odjemalca PS, kot so napovedovalni modeli ali orodja za podatkovno rudarjenje.

3. Načrtovanje dimenzijskega modela

3.1 Teoretične osnove

3.1.1 Sestavni deli dimenzijskega podatkovnega modela

Dimenzijsko modeliranje je tehnika logičnega načrtovanja, ki skuša predstaviti podatke v modelu, ki je intuitiven in omogoča veliko poizvedovalno zmogljivost [4]. Kot rezultat dimenzijskega modeliranja pridobimo dimenzijski model, ki je zaradi svoje karakteristične strukture znan tudi pod imenom *zvezdna shema* [6]. To ime so načrtovalci podatkovnih baz dolgo uporabljali zaradi tega, ker je diagram podoben zvezdi. Vsebuje eno veliko centralno tabelo in množico manjših spremljajočih tabel, ki predstavljajo krake zvezde. Centralno tabelo imenujemo *tabela dejstev*, medtem ko spremljajoče tabele imenujemo *dimenzijske tabele*. Primer dimenzijskega modela je prikazan na sliki 8.



Slika 8. Dimenzijski model na logičnem nivoju, ki ponazarja proces maloprodaje.

Tabela dejstev je centralna tabela dimenzijskega modela, ki vsebuje izmerjene podatke o rezultatih poslovanja v obliki t.i. merljivih dejstev (angl. measurable facts) [6]. Dimenzije prisotne v tabeli dejstev definirajo *zrno* tabele dejstev, ki nam pove kakšen je pomen posamezne vrstice v tabeli dejstev. Vse meritve, ki so prisotne v tabeli dejstev, morajo biti enakega zrna. Najbolj uporabna dejstva so *numerična*, *zvezna* in *aditivna*. Obstajajo tudi dejstva, ki so semi-aditivna ali ne-aditivna. Semi-aditivna dejstva se lahko seštejejo samo po nekaterih dimenzijah, medtem ko seštevanje ne-aditivnih dejstev pripelje do nesmiselnih rezultatov. Aditivnost je zelo pomembna lastnost dejstva, ker uporabniki najbolj pogosto povprašujejo po velikem številu zapisov iz tabele dejstev, ki jih želijo grupirati po določenih dimenzijah in sumirati po dejstvih. Tabele dejstev so v praksi zelo velike in jih v večjih organizacijah merimo v gigabajtih (GB) oziroma terabajtih (TB). Ponavadi imajo veliko število vrstic in majhno število stolpcev. Ne skušamo jih polniti z dejstvi, ki bi pomenila, da se nič ni zgodilo. Tako naprimer v tabelo dejstev dnevne prodaje iz slike 8, ne bomo vstavili zapisa, ki bi pomenil, da se na določen dan v določeni trgovini, nek izdelek ni prodal. Čeprav so tabele dejstev zaradi tega redke, vseeno zasedejo 90% ali več prostora v dimenzijski podatkovni bazi. Vsaka tabela dejstev ima dva ali več tujih ključev preko katerih je povezana s primarnimi ključi dimenzijskih tabel. Kadar se vsi tuji ključi tabele dejstev ujemajo z ustreznimi primarnimi ključi dimenzijskih tabel, zagotavljajo referenčno integriteto. Tabela

dejstev ima v splošnem svoj primarni ključ, ki je sestavljen iz podmnožice tujih ključev. Tak ključ imenujemo sestavljen ključ.

Dimenzijska tabela je tabela v dimenzijskem modelu, ki vsebuje opise določene dimenzije posla [5]. Primarni ključ dimenzijske tabele služi kot osnova za referenčno integriteto s katerokoli tabelo dejstev s katero je povezana. Ponavadi imajo dimenzijske tabele majhno število vrstic (pogosto manj kot 1 mio. vrstic) in veliko število stolpcev. V dobro načrtovanem dimenzijskem modelu imajo določene dimenzijske tabele tudi do 100 stolpcev (*atributov*), saj si prizadevamo dodati čim več pomembnih tekstualnih opisov. Najboljši atributi so *tekstualni in diskretni*, zato ker so najbolj uporabni za opisovanje enega izmed elementov dimenzije. Atributi dimenzije služijo kot primarni vir omejitev v poizvedbah in kot čelne vrstice v poročilih, kar lahko ponazorimo z naslednjim primerom, ki se navezuje na dimenzijski model iz slike 8:

Primer 1:

Denimo, da nas zanima skupni znesek prodaje in skupno število prodanih izdelkov za vsako posamezno blagovno znamko v prvem četrtletju leta 2008. Rezultat takega povpraševanja lahko pridobimo z naslednjim vprašanjem SQL:

```
SELECT p.Brand as Znamka, SUM(f.Dollars_sold) as Sk. znesek prodaje($),
SUM(f.Units_sold) as Št. izdelkov
FROM Sales f, Product p, Time t
WHERE f.Product_key =  p.Product_key
      AND f.Time_key = t.Time_key
      AND t.Quarter = '1 Q 2008'
GROUP BY p.Brand
ORDER BY p.Brand;
```

Kot lahko vidimo smo zgornje vprašanje SQL omejili po atributu *quarter* (kvartal) iz časovne dimenzije *time* (čas) in hkrati uporabili grupiranje rezultata po atributu *brand* (blagovna znamka) iz produktne dimenzije *product* (izdelek). V SELECT delu vprašanja SQL smo uporabili funkcijo *sum*, ki sešteje dejstvi *dollars_sold (*znesek*)* in *units_sold (*število izdelkov*)*.

Rezultat, ki ga vrne povpraševanje vsebuje v svoji čelni vrstici natanko tiste dimenzijske atribute, ki smo jih navedli v SELECT delu vprašanja SQL in zgleda takole:

Znamka	*Sk. znesek prodaje($)*	*Št. Izdelkov*
Axon	*780*	*263*
Framis	*1044*	*509*
Widget	*213*	*444*
Zapper	*95*	*39*

Poleg zvezdne sheme poznamo še *snežinkasto shemo*. Pri snežinkasti shemi normaliziramo dimenzijske tabele tako, da jih razčlenimo v drevesne strukture z morebitnimi večimi vgnezdenimi nivoji [5]. Tabela dejstev bo ostala ista kot pri zvezdni shemi, medtem ko dimenzije predstavimo v tretji normalni obliki. Čeprav z uporabo snežinkaste sheme prihranimo prostor in lažje vzdržujemo dimenzijski model, Kimball v splošnem nasprotuje

uporabi snežinkaste sheme, ker se ogroža razumljivost in brskalna učinkovitost dimenzijskega modela.

3.1.2 Heterogene produktne sheme

Finančne ustanove, naprimer banke, ponujajo veliko število produktov, ki so po svoji naravi heterogeni. Pri bankah produkti predstavljajo različne vrste računov. Banke tako svojim strankam ponujajo transakcijske račune, varčevalne račune, gotovinske kredite, hipotekarne kredite, kreditne kartice, depozite, vzajemne sklade itn. V takšnemu okolju imamo ponavadi dva zorna kota na aktivnosti računov, ki jih težko predstavimo v eni sami tabeli dejstev [4, 5, 6].

Prvi je *globalni pogled*, ki omogoča pregledovanje različnih vrst računov hkrati. Omogoča, naprimer planiranje prodaje dodatnih ali nadgradnje obstoječih produktov strankam banke, kot tudi povpraševanje po celotnem portfelju računov posamezne stranke. Ker imamo v večjih bankah tudi čez 20 različnih produktov, bi bilo nepraktično pregledovati 20 ali več ločenih tabel dejstev, vsakič ko rabimo profilirati bazo strank. Zaradi tega predstavimo globalni pogled v eni sami tabeli dejstev, ki omogoča pregled poslovanja z vsemi vrstami računov. To tabelo imenujemo *osnovna tabela dejstev*. Osnovna tabela dejstev vsebuje le omejeno število dejstev, ki so smiselni za vse vrste računov hkrati. Prav tako potrebujemo osnovne dimenzijske tabele, ki vsebujejo atribute, ki so skupni vsem vrstam računov.

Drugi je *pogled na posamezno vrsto računov*, naprimer transakcijske račune. Tukaj srečamo celo vrsto posebnih dejstev, ki so smiselna samo za določeno vrsto računov. Ta dejstva ne morejo biti vključena v osnovno tabelo dejstev, ker bi (ob predpostavki, da to naredimo za vsako posamezno vrsto računov) le – ta imela veliko število dejstev katerih večina bi imela ničelne vrednosti za katerokoli posamezno vrstico. Rešitev je v uporabi *prilagojene tabele dejstev*, ki vsebuje dejstva specifična za določeno vrsto računov. Podobno potrebujemo še prilagojene dimenzijske tabele, ki vsebujejo atribute specifične za posamezno vrsto računov. Prilagojene tabele vsebujejo tudi dejstva in atribute prisotne v osnovnih tabelah. S tem se izognemo povezovanju osnovnih in prilagojenih tabel vsakič, ko hočemo pridobiti celotno množico dejstev in atributov. Vsaka prilagojena dimenzijska tabela je podmnožica osnovne dimenzijske tabele in vsaka prilagojena tabela dejstev je podmnožica osnovne tabele dejstev.

3.1.3 Transakcijske sheme in sheme periodičnih posnetkov stanj

Pri poslih, ki imajo bogato transakcijsko zgodovino, ponavadi potrebujemo dve tabeli dejstev s katerimi predstavimo celotno poslovanje. Ena tabela dejstev vsebuje posamezne transakcije, medtem ko druga vsebuje periodične posnetke stanja [4, 5].

Transakcijske sheme uporabljamo za pregled poslovnih aktivnosti na nivoju posameznih transakcij. Posamezen zapis tabele dejstev predstavlja dogodek, ki se je zgodil v določenem trenutku časa. Tako naprimer v bančnem okolju, predstavljajo transakcije dnevne prilive oziroma odlive določenega računa. Podatki na nivoju transakcij podpirajo podrobno analiziranje vedenja in jih zaradi tega pogosto uporabljamo pri podatkovnem rudarjenju. Obstaja pa tudi cela vrsta nujnih poslovnih vprašanj na katere je nepraktično odgovarjati direktno iz transakcijskih shem.

Periodične posnetke stanja potrebujemo zaradi kumulativnega pogleda poslovnih aktivnosti v rednih časovnih intervalih. Tako pridobimo sliko poslovnih aktivnosti ob koncu

dneva, tedna, meseca ali kakšnega drugega časovnega obdobja. V večini primerov periodični posnetki vsebujejo več dejstev kot transakcijske tabele. Vpeljemo lahko poljubno število dejstev pod pogojem, da se vsa dejstva nanašajo na isti časovni interval. Periodične posnetke stanja zgradimo tako, da inkrementalno dodajamo učinke posameznih transakcij posnetku v katerega časovni interval sodijo.

3.1.4 Koraki pri načrtovanju dimenzijskega modela

V [4, 5, 6] Kimball priporoča uporabo štirih zaporednih korakov pri načrtovanju posamezne dimenzijske sheme. Izberemo:

1. Poslovni proces, ki ga želimo modelirati,
2. zrno tabele dejstev,
3. dimenzije in
4. merljiva dejstva.

V prvem koraku načrtovanja se odločimo, kateri poslovni proces bomo modelirali, pri čemer upoštevamo uporabniške zahteve in dostopnost izvornih podatkov. Prvi dimenzijski model, ki ga mislimo zgraditi mora biti zmožen podati odgovore na najbolj nujna poslovna vprašanja. Zaradi hitrosti in enostavnosti izvedbe je zaželjeno da se najprej implementirajo PPS, ki se polnijo iz enega samega izvora.

Drugi korak definira zrno tabele dejstev, kar pomeni, da se odločimo kakšen nivo podrobnosti podatkov bo dostopen v dimenzijskem modelu. Zaželjeno je, da razvijemo model za najbolj atomarno informacijo zajeto s strani poslovnega procesa, ker nam le – ta omogoča največjo analitično fleksibilnost. Tipična zrna so posamezne transakcije ali dnevni oziroma mesečni posnetki stanj.

Tretji korak zahteva opredelitev dimenzij. Že samo zrno tabele dejstev pogosto določi minimalno množico potrebnih dimenzij. Ponavadi lahko dodamo še dodatne dimenzije. Tipične dimenzije so čas, produkt, stranka, promocija, status, tip transakcije itn.

V zadnjem koraku izberemo merljiva dejstva. Dejstva morajo ustrezati zrnu tabele dejstev. Tipična merljiva dejstva so numerične aditivne vrednosti, kot so naprimer prodane količine ali numerične semi-aditivne vrednosti, kot je naprimer dnevno stanje računa.

3.2 Uporabniške zahteve

Transakcijski račun (TRR) namenjen fizičnim osebam je bančni produkt, ki je danes najbolj zastopan med prebivalstvom. Banke ponujajo različne vrste TRR prilagojene potrebam različnih ciljnih skupin. Imetnikom omogoča urejanje vsakodnevnih denarnih zadev v domači ali tuji valuti, kot so prejemanje različnih nakazil (plače, honorarji, pokojnine), plačevanje rednih mesečnih obveznosti prek trajnih nalogov in direktnih obremenitev, opravljanje nakazil, brezgotovinsko plačevanje z uporabo plačilnih kartic, dvig in polog gotovine na bankomatih in drugo. Zaradi velike konkurence, skušajo banke vedno znova ponuditi nekaj več svojim komitentom, jih na ta način obdržati oziroma privabiti nove stranke. Nemalokrat ponujajo ob odprtju TRR še brezplačno nezgodno zavarovanje, sodelovanje v različnih shemah ugodnosti ali brezplačno uporabo elektronskega bančništva.

V tem diplomskem delu razvijemo dimenzijske modele na osnovi zahtev namišljenih uporabnikov. Čeprav so uporabniki namišljeni, so tako zahteve kot tudi dimenzijski modeli v nekoliko bolj bogati obliki prisotni v skoraj vsakem bančnem PS.

Uporabniške zahteve v grobem narekujejo razvoj dimenzijskih modelov s katerimi bo možno:

- spremljanje prometa TRR fizičnih oseb,
- spremljanje dnevnih stanj TRR fizičnih oseb,
- spremljanje dnevnih stanj za vse vrste računov fizičnih oseb hkrati.

Prvo točko uporabniških zahtev realiziramo v podpoglavju 3.3 z izgradnjo transakcijske sheme TRR fizičnih oseb, drugo točko pa v podpoglavju 3.4 z izgradnjo sheme dnevnega posnetka stanj za TRR fizičnih oseb. Ob realizaciji druge točke pridobimo pogled na posamezno vrsto računov in sicer večvalutne transakcijske račune fizičnih oseb. Realizacijo tretje točke uporabniških zahtev predstavimo v podpoglavju 3.5 v katerem zgradimo shemo dnevnega posnetka stanj za vse vrste računov fizičnih oseb. Zadnji dimenzijski model ponazarja globalni pogled na bančno poslovanje, kot smo ga obravnavali v razdelku 3.1.2.

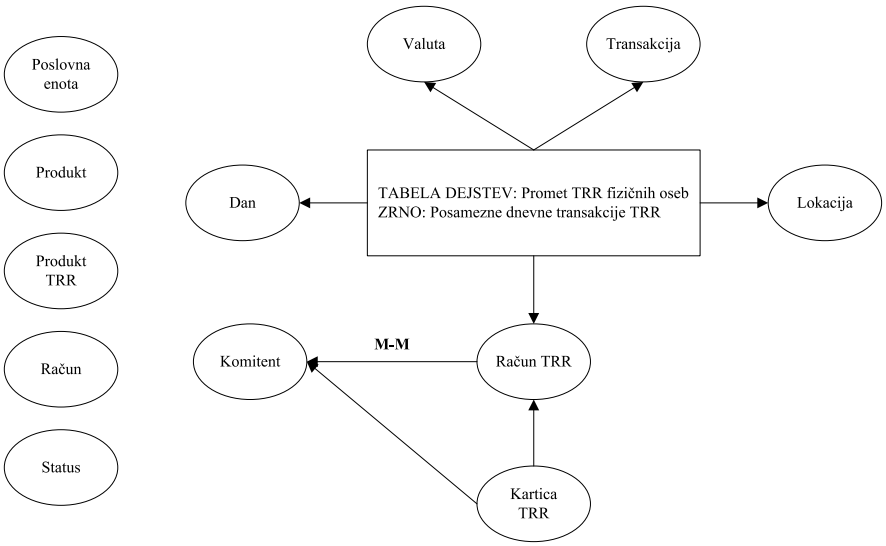
Na koncu še predpostavimo, da podatke s katerimi bomo polnili PPS, pridobimo iz enega samega izvora, ki smo ga predhodno analizirali. Z analizo izvornega sistema pridobimo razumevanje izvornih struktur in podatkov, kot tudi informacijo o tem kakšen model je sploh možno zgraditi glede na dostopnost podatkov.

3.3 Dimenzijski model za spremljanje prometa TRR fizičnih oseb

Načrtovanje transakcijske sheme pričnemo z izbiro poslovnega procesa. Le – ta je definiran z uporabniško zahtevo podano v prvi točki podpoglavja 3.2 – spremljanje prometa TRR fizičnih oseb. Nadaljujemo z izbiro zrna tabele dejstev. Odločimo se spremljati posamezne dnevne transakcije (promet) na transakcijskih računih fizičnih oseb. Enkrat, ko je zrno tabele dejstev določeno, je dokaj enostavno opredeliti dimenzije. Namreč običajno želimo odgovoriti na naslednja vprašanja: kdaj in kje se je transakcija izvršila, v kateri valuti, za kakšno vrsto transakcije gre in na katerem računu jo knjižimo. Odgovore na zastavljena vprašanja pridobimo iz dimenzij dan, lokacija, valuta, transakcija in račun TRR. Seveda je možno dodati še več dimenzij, kar je odvisno od konkretnih uporabniških zahtev in razpložljivih podatkov v izvornih sistemih. V našem primeru dodamo še dimenziji kartica TRR in komitent, ki niso direktno povezane s tabelo dejstev, temveč posredno preko dimenzije račun TRR. Dimenzija transakcijskih računov bo vsebovala tudi tuje ključe preko katerih bo povezana z dimenzijami produkt TRR, status in poslovna enota. Na koncu opredelimo še dejstva. V našem primeru bo v tabeli dejstev eno samo dejstvo in sicer znesek transakcije.

Razvoj dimenzijskega modela zahteva komunikacijo med stranko in izvajalcem, kot tudi komunikacijo med člani projektne skupine. Da bi olajšali to komunikacijo, pogosto uporabljamo grafična orodja, kot so *diagram tabele dejstev* (angl. fact tabl diagram) in *podrobnosti tabele dejstev* (angl. fact table detail) [4]. Ker ne vsebujejo preveč tehničnih podrobnosti sta ti dve orodji zelo primerni kot uvod v celoten model. Diagram tabele dejstev je logičen diagram s katerim poimenujemo tabelo dejstev, opredelimo zrno in prikažemo vse dimenzije, s katerimi je tabela dejstev povezana. Diagram tabele dejstev za spremljanje prometa TRR fizičnih oseb je prikazan na sliki 9. Lahko opazimo, da so prikazane tudi

dimenzije, ki niso povezane s tabelo dejstev za spremljanje prometa TRR. Te dimenzije uporabimo v ostalih dimenzijskih modelih PPS.



Slika 9. Diagram tabele dejstev za spremljanje prometa TRR fizičnih oseb.

Podrobnosti tabele dejstev nam prikažejo dimenzijske ključe ter celoten seznam dejstev skupaj z njihovimi lastnostmi. Slika 10 prikazuje podrobnosti tabele dejstev za spremljanje prometa TRR fizičnih oseb.

TABELA DEJSTEV: Promet TRR fizičnih oseb
ZRNO: Posamezne dnevne transakcije TRR
DIMENZIJE (FK): Dan, Valuta, Transakcija, Lokacija, Račun TRR
DEJSTVA: Znesek (aditiven)

Slika 10. Podrobnosti tabele dejstev za spremljanje prometa TRR fizičnih oseb.

Pri razvoju fizičnega podatkovnega modela začnemo z načrtovanjem dimenzijskih tabel. Za vsako posamezno dimenzijo, ki je dostopna iz tabele dejstev opredelimo ime tabele, ime atributov, podatkovne tipe, tekstualni opis atributov in vzorčne vrednosti. Definicije vseh dimenzijskih tabel uporabljenih v dimenzijskih modelih so podane v prilogi 7.1, medtem ko na tem mestu opišemo pomen posamezne dimenzije:

- *Datumska dimenzija **DIM_DAN***:

  Posamezna vrstica dimenzijske tabele opisuje določen dan v letu. Dan je opredeljen z datumom. Dimenzija vsebuje veliko uporabnih opisnih atributov med katerimi naštejemo samo nekatere: besedni opis datuma, naziv dneva, naziv meseca, opis kvartala, opis leta, zaporedne številke dneva v tednu, mesecu, letu kot tudi indikatorje vikendov, praznikov in delovnih dni. Dimenzija omogoča tudi vrtanje v globino, saj je hierarhično organizirana (hierarhija leto – kvartal – mesec – teden – dan).

- *Dimenzija komitentov (fizične osebe)* ***DIM_FO_KOMITENT****:*

Posamezna vrstica dimenzijske tabele opisuje določenega komitenta banke. Vsebuje atribut KOMITENT_ID_IZVOR v katerega shranjujemo izvorni ključ (enolični identifikator referenčnega zapisa na izvoru). Poleg osnovnih podatkov o komitentu (ime, priimek, titula, spol, emšo, davčna številka, kraj rojstva, datum rojstva, datum smrti, država rojstva, državljanstvo, rezidentnost) hranimo tudi podatke o poklicu, izobrazbi in delovnem mestu komitenta. Nenazadnje nas zanimajo tudi kontaktni podatki komitenta (naslov za pošiljanje izpiskov, naslov stalnega in začasnega prebivališče kot tudi različne telefonske številke in elektronski naslov). Dimenzija je povezana preko povezovalne tabele (REL_RACUN_KOMITENT) z dimenzijo transakcijskih računov (DIM_RACUN_TRR). Medsebojno povezanost omenjenih dimenzij obravnavamo v nadaljevanju poglavja.

- *Dimenzija kartic TRR* ***DIM_KARTICA_TRR****:*

Posamezna vrstica dimenzijske tabele opisuje kartico izdano določenemu komitentu za določen transakcijski račun. V stolpcu KARTICA_ID_IZVOR shranjujemo izvorni ključ. Preostali atributi hranijo informacijo o datumu izdaje in veljavnosti kartice, tipu kartice (npr. študentska, zlata itn.), statusu kartice (npr. aktivna, blokirana, neaktivna) kot tudi podatke o morebitni blokadi kartice (datum in vzrok blokade). Dimenzija je preko tujega ključa RACUN_ID povezana z dimenzijo DIM_RACUN_TRR, ter preko tujega ključa KOMITENT_ID z dimenzijo DIM_FO_KOMITENT.

- *Dimenzija lokacij* ***DIM_LOKACIJA****:*

Posamezna vrstica dimenzijske tabele opisuje določeno lokacijo, kjer se je izvršila neka transakcija. Vsebuje atribut LOKACIJA_ID_IZVOR, ki hrani izvorni ključ. Preostala atributa sta namenjena dolgemu opisu lokacije (npr. poslovna enota, bankomat, spletna banka itn.) in indikatorju lokacije (npr. PE za poslovno enoto, BAN za bankomat, SB za spletno banko).

- *Dimenzija poslovnih enot* ***DIM_POSLOVNA_ENOTA****:*

Posamezna vrstica dimenzijske tabele opisuje določeno poslovno enoto. Atribut PE_ID_IZVOR hrani izvorni ključ. Ostali atributi definirajo opis poslovne enote, vodjo poslovne enote, naslov poslovne enote, delovni čas poslovne enote, kontaktne podatke poslovne enote (telefon in telefax), status aktivnosti poslovne enote kot tudi datume pričetka in morebitnega konca veljavnosti poslovne enote. Dimenzija vsebuje hierarhijo in sicer poslovna enota – regija – država.

- *Dimenzija produktov TRR* ***DIM_PRODUKT_TRR****:*

Posamezna vrstica prilagojene dimenzijske tabele opisuje določen produkt, ki pripada skupini produktov TRR. Dimenzija je prilagojena tako, da poleg osnovnih atributov, ki so značilni za vse produkte, ki jih banka ponuja, vsebuje tudi prilagojene atribute, specifične za skupino produktov TRR. Med osnovne atribute sodi atribut PRODUKT_ID_IZVOR, ki hrani izvorni ključ. Atribute, ki podajajo dolgi opis produkta, oznako produkta, veljavnost produkta, status produkta, ročnost produkta, skupino produkta in področje produkta tudi štejemo med osnovne atribute. Med prilagojene atribute pa sodijo atributi, ki hranijo podatke o stroških vodenja, obrestnih merah pozitivnega, negativnega in nedovoljenega stanja, limitih (možnost pridobitve

avtomatičnega oziroma izrednega limita), nezgodnem zavarovanju in številu mesečnih izpiskov. V podpoglavju 3.5, znotraj katerega razvijemo dimenzijski model za spremljanje dnevnih stanj za vse vrste računov, opišemo dimenzijo vseh produktov (DIM_PRODUKT), ki jih ponuja banka. Kot bomo videli, vsebuje osnovna dimenzija DIM_PRODUKT natanko tiste atribute, ki jih pri prilagojeni dimenziji DIM_PRODUKT_TRR štejemo med osnovne. Velja, da je prilagojena dimenzija DIM_PRODUKT_TRR (če upoštevamo samo osnovne atribute) podmnožica osnovne dimenzije DIM_PRODUKT, saj vsebuje natanko iste zapise kot dimenzija DIM_PRODUKT. Podobno kot datumska dimenzija, vsebuje tudi dimenzija DIM_PRODUKT_TRR hierarhijo in sicer področje – skupina produktov – produkt. Tako naprimer osnovni transakcijski račun (OTRR), sodi v skupino produktov transakcijskih računov (TRR), ki sodi v področje poslovanja s fizičnimi osebami.

- *Dimenzija transakcijskih računov **DIM_RACUN_TRR**:*

Posamezna vrstica prilagojene dimenzijske tabele opisuje določen transakcijski račun. Podobno kot prej obravnavana dimenzija produktov TRR, vsebuje dimenzija transakcijskih računov osnovne in prilagojene atribute. Osnovni atribut RACUN_ID_IZVOR hrani izvorni ključ. Ostali osnovni atributi definirajo produkt, datum odprtja, zaprtja in aktivacije računa, status računa, bonitetno oceno računa ter podatke o matični in zadnji poslovni enoti kateri račun pripada. Prilagojeni atributi se nanašajo na podatke o limitu računa, kot so odobreni znesek limita, datum veljavnosti in poteka limita, ter datum odobritve in preklica limita. V podpoglavju 3.5 obravnavamo osnovno dimenzijsko tabelo DIM_RACUN, ki hrani vse bančne račune, ne glede na vrsto računa. Dimenzija DIM_RACUN vsebuje natanko tiste atribute, ki jih pri prilagojeni dimenziji DIM_RACUN _TRR štejemo med osnovne. Velja, da je prilagojena dimenzija DIM_RACUN _TRR (če upoštevamo samo osnovne atribute) podmnožica osnovne dimenzije DIM_RACUN, saj vsebuje natanko iste zapise kot dimenzija DIM_RACUN. Prilagojena dimenzijska tabela DIM_RACUN_TRR je povezana preko tujega ključa PRODUKT_ID z dimenzijo produktov TRR, preko tujega ključa ZADNJI_STATUS_ID z dimenzijo statusov ter preko tujih ključev MAT_PE_ID in ZADNJA_PE_ID z dimenzijo poslovnih enot. Dimenzija je povezana preko povezovalne tabele (REL_RACUN_KOMITENT) z dimenzijo komitentov. Medsebojno povezanost omenjenih dimenzij obravnavamo v nadaljevanju poglavja.

- *Dimenzija statusov **DIM_STATUS**:*

Posamezna vrstica dimenzijske tabele opisuje določen status računa. Atribut STATUS_ID_IZVOR hrani izvorni ključ. Ostale atribute uporabljamo za podajanje opisa statusa in indikatorja statusa.

- *Dimenzija transakcij **DIM_TRANSAKCIJA**:*

Posamezna vrstica dimenzijske tabele opisuje določeno transakcijo. Atribut TRANS_ID_IZVOR hrani izvorni ključ. Preostali atributi določajo opis tipa transakcije (npr. odliv), indikator tipa transakcije (npr. O za odliv), ter opis same transakcije (npr. dvig gotovine ali provizija banke za opravljeno transakcijo).

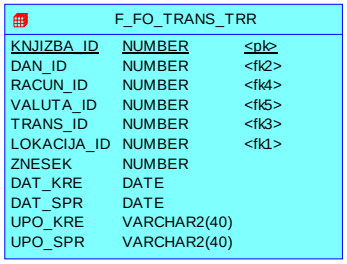
- *Dimenzija valut **DIM_VALUTA**:*

Posamezna vrstica dimenzijske tabele opisuje določeno valuto. Atribut VALUTA_ID_IZVOR hrani izvorni ključ. Preostali atributi so namenjeni za podajanje oznake valute, dolgega opisa valute v slovenskem in tujem jeziku, enote

valute, države valute, datuma začetka in konca veljavnosti valute, kot tudi indikator in opis veljavnosti valute.

V vseh dimenzijskih tabelah uporabljamo nadomestne ključe (angl. surrogate key). To pomeni, da za primarni ključ ne uporabimo izvorni produkcijski ključ (ali pametni ključ sestavljen iz ostalih atributov dimenzijske tabele), temveč ga sami generiramo z uporabo sekvence. Nadomestni ključi nimajo pomena, ker na osnovi ključa absolutno nič ne moremo povedati o zapisu, ki ga ta ključ določa. Uporaba izvornih produkcijskih ključev je odsvetovana, ker se lahko spremenijo ali ponovno uporabijo za nove zapise, kar povzroči velike težave [4]. Pri vseh tabelah uporabljamo še meta podatkovne atribute DAT_KRE, DAT_SPR, UPO_KRE in UPO_SPR v katerih hranimo čas oziroma uporabnika spremembe zapisa.

Nadaljujemo z definicijo tabele dejstev. Slika 11 prikazuje transakcijsko tabelo dejstev F_FO_TRANS_TRR.

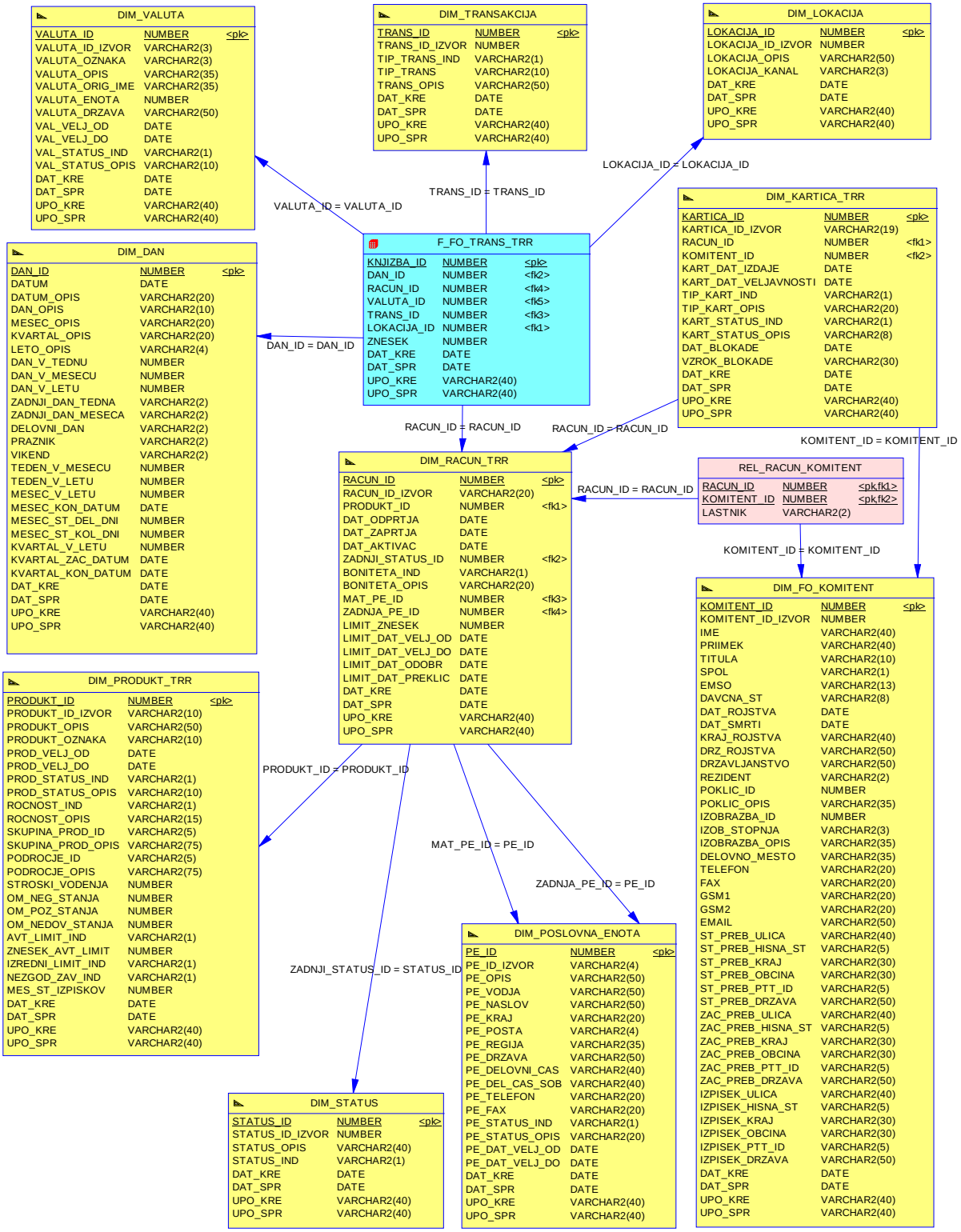


Slika 11. Tabela dejstev F_FO_TRANS_TRR.

Poleg ključev dimenzijskih tabel (DAN_ID, RACUN_ID, VALUTA_ID, TRANS_ID, LOKACIJA_ID), vsebuje še dejstvo ZNESEK , enolično povratno povezavo (referenco) na izvorni zapis KNJIZBA_ID, ter meta podatkovne atribute. Znesek je vrednost, ki je vedno pozitivna in šele ob povezavi z dimenzijo DIM_TRANSAKCIJA preko tujega ključa TRANS_ID lahko ugotovimo za kakšno vrsto transakcije gre (priliv/odliv). Provizija (npr. ob dvigu gotovine na tujem bankomatu) se v tabeli dejstev F_FO_TRANS_TRR prikaže kot samostojna transakcija.

Slika 12 prikazuje celoten fizični podatkovni model za potrebe spremljanja prometa TRR fizičnih oseb. Kot je razvidno iz modela, vsebuje tabela dejstev F_FO_TRANS_TRR pet tujih ključev, preko katerih je povezana z ustreznimi primarnimi ključi dimenzijskih tabel. Tabela dejstev ima primarni ključ na polju KNJIZBA_ID, ki služi kot enolični identifikator transakcije tako na izvoru kot tudi v PPS. Preko atributa KNJIZBA_ID vedno lahko referenciramo nazaj na izvorno transakcijo. Dimenzija transakcijskih računov je *večvrednostna dimenzija*. En račun je namreč lahko povezan z enim ali večimi komitenti, pri čemer je en komitent lastnik računa, ostali komitenti so pooblaščenci. Iz tega razloga ne moremo vključiti komitenta kot atribut v dimenziji transakcijskih računov, saj bi s tem prekršili zrno dimenzijske tabele. Podobno ne moremo vključiti dimenzijo komitentov v tabelo dejstev, saj bi prekršili zrno tabele dejstev. Rešitev je v uporabi povezovalne tabele REL_RACUN_KOMITENT. Povezovalna tabela vsebuje dva tuja ključa (RACUN_ID, KOMITENT_ID) preko katerih je povezana s primarnimi ključi dimenzijskih tabel DIM_RACUN_TRR in DIM_FO_KOMITENT. Primarni ključ povezovalne tabele je sestavljen iz obeh tujih ključev (RACUN_ID, KOMITENT_ID). S pomočjo povezovalne tabele enostavno določimo relacije med transakcijskimi računi in komitenti. Če se določen račun večkrat pojavi v povezovalni tabeli, pri čemer se ključ komitenta razlikuje pomeni, da

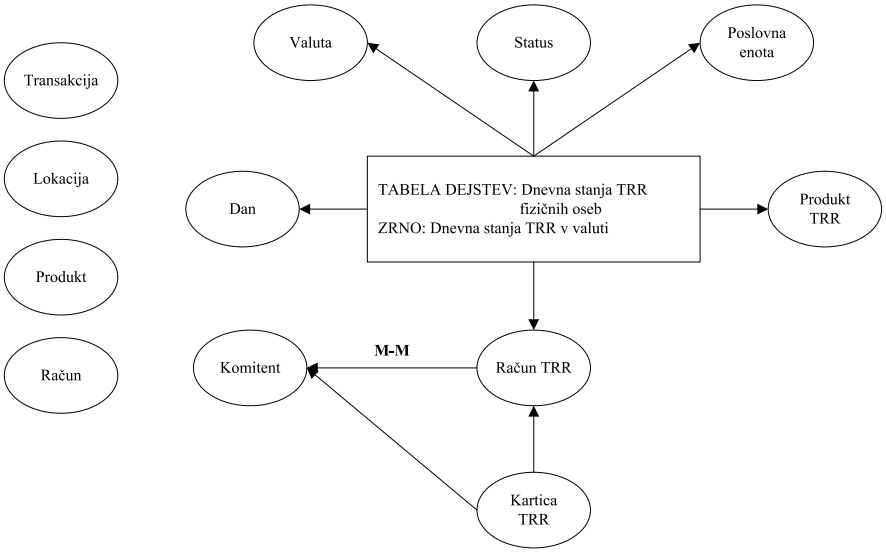
ima račun več različnih komitentov. Podobno, če se za istega komitenta v povezovalni tabeli pojavljajo različni ključi računov pomeni, da ima komitent več različnih transakcijskih računov. Ugotavljanje, ali je komitent lastnik, ali pooblaščenec transakcijskega računa, določamo z atributom LASTNIK, ki ima dve možni vrednosti in sicer DA za lastnika računa ter NE za pooblaščenca.



Slika 12. Fizični podatkovni model: spremljanje prometa TRR fizičnih oseb.

3.4 Dimenzijski model za spremljanje dnevnih stanj TRR fizičnih oseb

V tem poglavju modeliramo dimenzijski model za potrebe spremljanja dnevnih stanj transakcijskih računov fizičnih oseb. Izberemo zrno tabele dejstev – dnevna stanja računov TRR v posamezni valuti. Določen transakcijski račun v neki valuti ima na določen dan največ en zapis v tabeli dejstev. Uporabimo dimenzije, ki smo jih definirali v prejšnjem podpoglavju: dan, produkt TRR, račun TRR, kartica TRR, komitent, valuta, status in poslovna enota. Na koncu opredelimo še dejstva: dnevno stanje v domači valuti, dnevno stanje v valuti, kot tudi število transakcij, ki so vplivale na dnevno stanje računa. Na diagramu tabele dejstev na sliki 13 grafično predstavimo dosedanje razmišljanje.



Slika 13. Diagram tabele dejstev za spremljanje dnevnih stanj TRR fizičnih oseb.

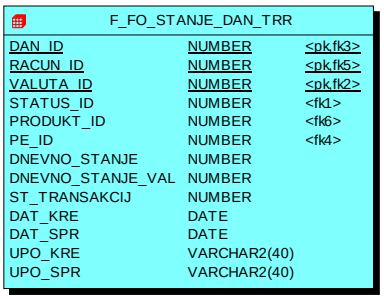
Slika 14 prikazuje podrobnosti tabele dejstev za spremljanje dnevnih stanj TRR fizičnih oseb. Opazimo, da sta dejstvi dnevno stanje v domači valuti in dnevno stanje v valuti ne-aditivna glede na dimenzijo dan. Če želimo izračunati povprečno dnevno stanje računa v nekem časovnem obdobju moramo deliti seštevek dnevnih stanj računa v opazovanem obdobju s številom dni v tem obdobju, sicer dobimo nesmiselne rezultate.

TABELA DEJSTEV: Dnevna stanja TRR fizičnih oseb
ZRNO: Dnevna stanja TRR v valuti
DIMENZIJE (FK): Dan, Valuta, Status, Poslovna enota, Produkt TRR, Račun TRR
DEJSTVA: Dnevno stanje v domači valuti (ne-aditivno glede na dimenzijo dan), Dnevno stanje v valuti (ne-aditivno glede na dimenzijo dan), Število dnevnih transakcij (aditivno)

Slika 14. Podrobnosti tabele dejstev dnevnih stanj TRR fizičnih oseb.

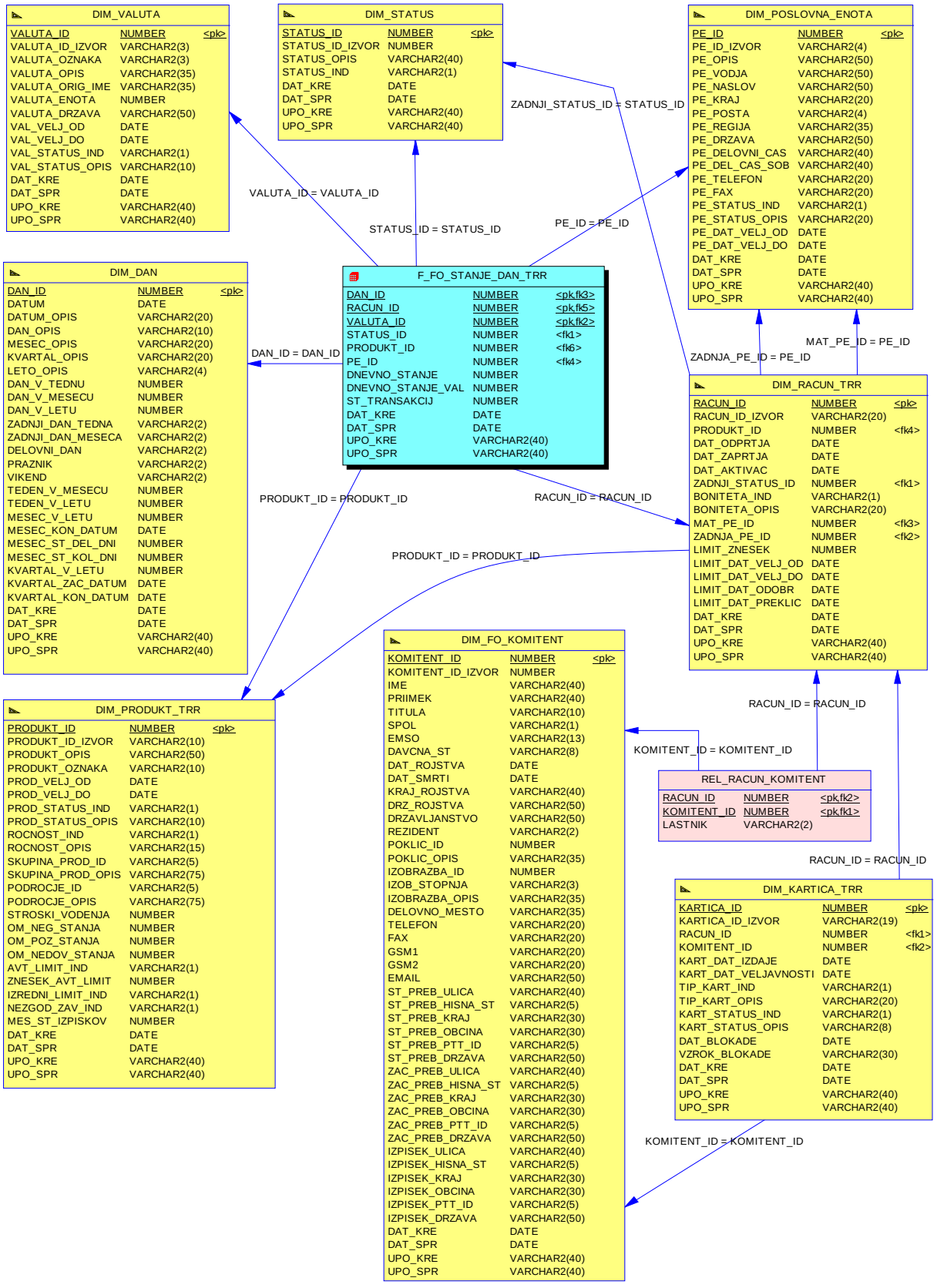
Ker so dimenzijske tabele med posameznimi dimenzijskimi modeli medsebojno usklajene, uporabimo že obstoječe definicije zahtevanih dimenzij (dan, produkt TRR, račun TRR, kartica TRR, komitent, valuta, status, poslovna enota), kot smo jih obravnavali v podpoglavju 3.3.

Slika 15 prikazuje tabelo dejstev F_FO_STANJE_DAN_TRR v kateri hranimo dnevna stanja večvalutnih transakcijskih računov. Poleg ključev dimenzijskih tabel (DAN_ID, RACUN_ID, VALUTA_ID, STATUS_ID, PRODUKT_ID, PE_ID), vsebuje še dejstva (DNEVNO_STANJE, DNEVNO_STANJE_VAL, ST_TRANSAKCIJ) ter že opisane meta podatkovne atribute.



Slika 15. Tabela dejstev F_FO_STANJE_DAN_TRR.

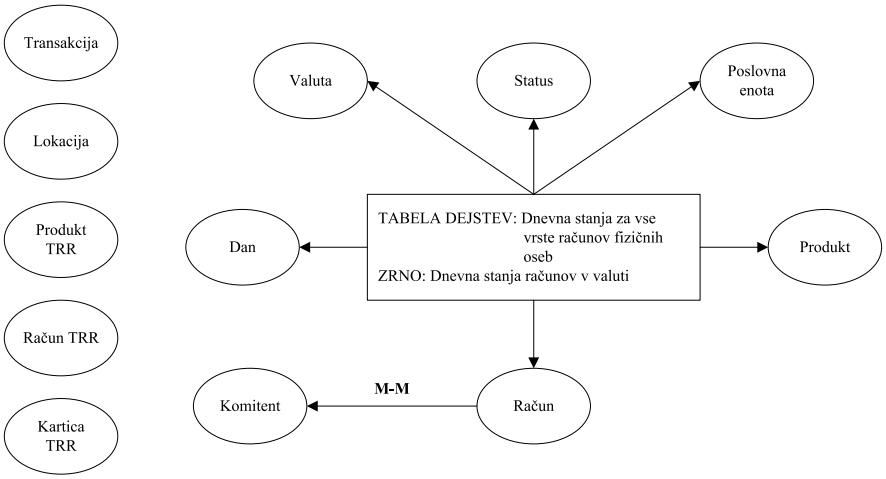
Slika 16 podaja celoten fizični podatkovni model za spremljanje dnevnih stanj večvalutnih transakcijskih računov fizičnih oseb. Prilagojena tabela dejstev F_FO_STANJE_DAN_TRR vsebuje šest tujih ključev preko katerih je povezana z ustreznimi primarnimi ključi dimenzijskih tabel. Lahko opazimo, da podmnožica tujih ključev (DAN_ID, RACUN_ID, VALUTA_ID) hkrati sestavlja tudi primarni ključ tabele dejstev. Ker model ponazarja pogled na posamezno vrsto računov, uporabljamo tudi prilagojeni dimenzijski tabeli DIM_RACUN_TRR in DIM_PRODUKT_TRR. Kot lahko opazimo je tudi dimenzija statusov računa (DIM_STATUS) povezana s tabelo dejstev. Uporabna je, ker lahko hitro ugotovimo status določenega računa (naprimer aktiven, neaktiven, blokiran itn.), ne da bi rabili pregledovati obsežno dimenzijo računov. Iz podobnega razloga vključimo tudi dimenzijo poslovnih enot. Pogosto nas zanima h kateri poslovni enoti sodi določen račun.



Slika 16. Fizični podatkovni model: dnevna stanja TRR fizičnih oseb.

3.5 Dimenzijski model za spremljanje dnevnih stanj za vse vrste računov

Dimenzijski model za spremljanje dnevnih stanj za vse vrste računov fizičnih oseb je precej podoben modelu, ki smo ga razvili v podpoglavju 3.4, vendar ne vsebuje toliko podrobnosti specifičnih za posamezno vrsto računa. Razlika je seveda v tem, da je sedaj nabor računov večji in glede na produkte heterogen. Izberemo zrno tabele dejstev – dnevna stanja vseh računov fizičnih oseb v posamezni valuti. Uporabimo že definirane dimenzije dan, valuta, status, poslovna enota in komitent. Ker želimo pridobiti globalen pogled na dnevna stanja vseh računov, dodamo dve novi dimenziji: produkt in račun. V dimenziji produkt hranimo informacijo o vseh produktih, ki jih banka ponuja, medtem ko dimenzija račun vsebuje vse bančne račune, ne glede na njihovo vrsto. Dejstvi, ki nas zanimata sta dnevno stanje računa v domači valuti in dnevno stanje računa v valuti. Slika 17 prikazuje diagram tabele dejstev za spremljanje dnevnih stanj vseh vrst računov fizičnih oseb.



Slika 17. Diagram tabele dejstev za spremljanje dnevnih stanj za vse vrste računov.

Podrobnosti tabele dejstev prikažemo na sliki 18. Podobno kot v prejšnjem podpoglavju sta dejstvi dnevno stanje v domači valuti in dnevno stanje v valuti ne-aditivni glede na dimenzijo dan.

TABELA DEJSTEV: Dnevna stanja za vse vrste računov fizičnih oseb
ZRNO: Dnevna stanja računov v valuti
DIMENZIJE (FK): Dan, Valuta, Status, Poslovna enota, Produkt, Račun
DEJSTVA: Dnevno stanje v domači valuti (ne-aditivno glede na dimenzijo dan),
Dnevno stanje v valuti (ne-aditivno glede na dimenzijo dan)

Slika 18. Podrobnosti tabele dejstev za dnevna stanja za vse vrste računov fizičnih oseb.

V nadaljevanju opišemo pomen dveh novih dimenzij (produkt in račun), medtem ko opisi ostalih uporabljenih medsebojno usklajenih dimenzij ostanejo takšni, kot smo jih obravnavali v podpoglavju 3.3.

- *Dimenzija produktov **DIM_PRODUKT**:*

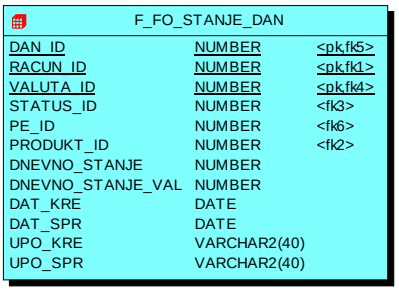
Posamezna vrstica osnovne dimenzijske tabele opisuje določen produkt. Dimenzija vsebuje po en zapis za vsak posamezen produkt, ki ga banka ponuja, kot so

naprimer študentski transakcijski račun, gotovinski kredit ali varčevalni račun namenjen mladim osebam. Dimenzija vsebuje osnovne atribute prisotne v dimenziji DIM_PRODUKT_TRR, kot smo jih obravnavali v podpoglavju 3.3. Glede na število zapisov je dimenzija vseh produktov večja od dimenzije DIM_PRODUKT_TRR, saj poleg vseh produktov, ki pripadajo skupini produktov TRR, vsebuje tudi vse ostale produkte, ki jih ponuja banka (krediti, depoziti, varčevanje itn.). Podobno kot dimenzija DIM_PRODUKT_TRR, vsebuje tudi dimenzija DIM_PRODUKT hierarhijo in sicer področje – skupina produktov – produkt.

- *Dimenzija računov **DIM_RACUN**:*

Posamezna vrstica osnovne dimenzijske tabele opisuje določen račun. Dimenzija vsebuje en zapis za vsak posamezen račun ne glede na vrsto računa. Atributi prisotni v dimenziji so osnovni atributi prilagojene dimenzijske tabele DIM_RACUN_TRR, ki smo jo obravnavali v podpoglavju 3.3. Velja, da je prilagojena dimenzija DIM_RACUN_TRR (če upoštevamo samo osnovne atribute) podmnožica osnovne dimenzije DIM_RACUN, saj vsebuje natanko iste zapise kot dimenzija DIM_RACUN. Osnovna dimenzijska tabela DIM_RACUN je povezana preko tujega ključa PRODUKT_ID z dimenzijo vseh produktov DIM_PRODUKT, preko tujega ključa ZADNJI_STATUS_ID z dimenzijo statusov ter preko tujih ključev MAT_PE_ID in ZADNJA_PE_ID z dimenzijo poslovnih enot. Dimenzija je povezana preko povezovalne tabele (REL_RACUN_KOMITENT) z dimenzijo komitentov.

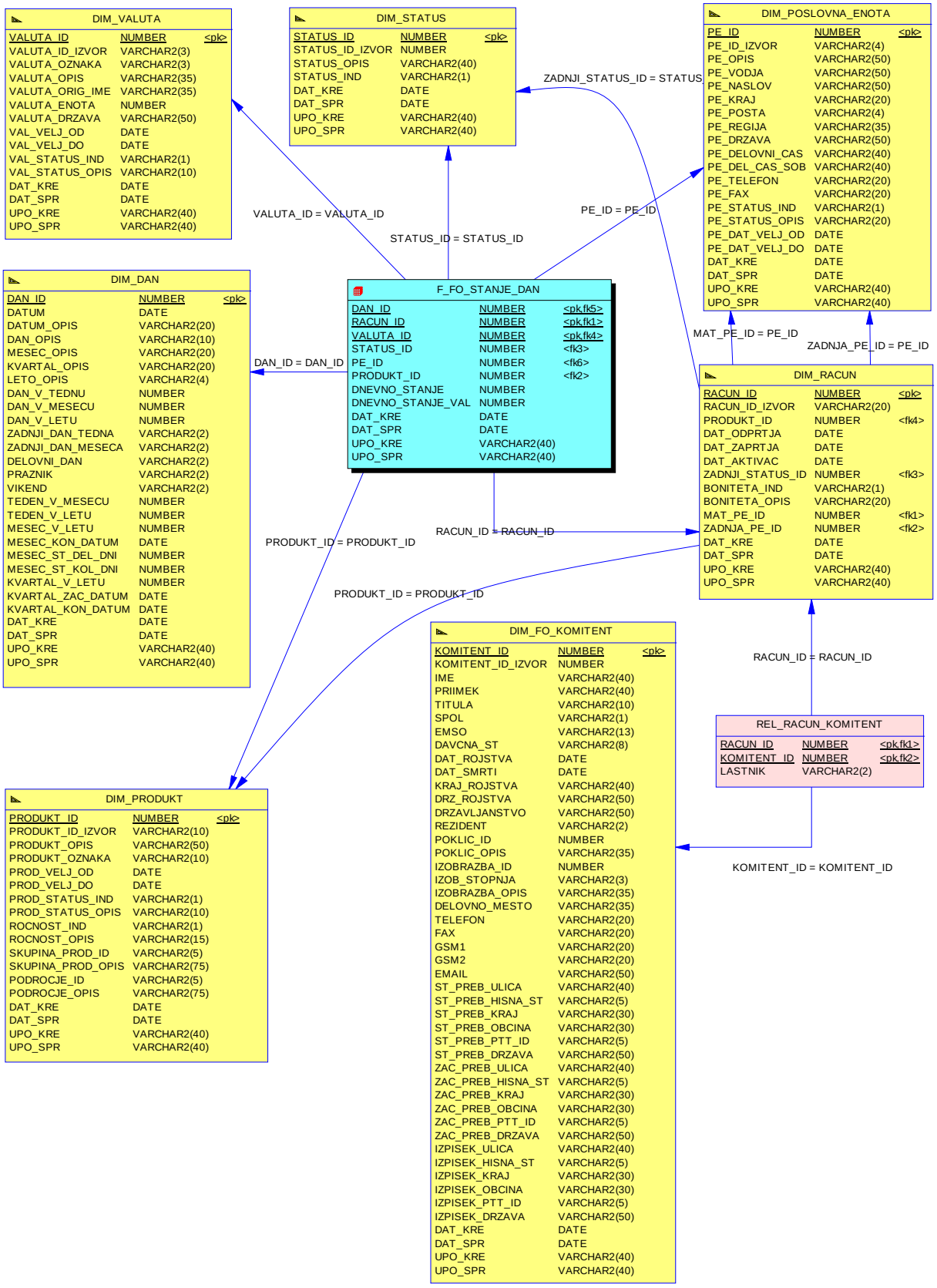
Slika 19 podaja definicijo tabele dejstev F_FO_STANJE_DAN. Podobno kot pri tabeli dejstev F_FO_STANJE_DAN_TRR (slika 11), lahko ugotovimo, da poleg ključev dimenzijskih tabel (DAN_ID, RACUN_ID, VALUTA_ID, STATUS_ID, PRODUKT_ID, PE_ID), vsebuje tudi dejstvi (DNEVNO_STANJE , DNEVNO_STANJE_VAL) ter meta podatkovne atribute. Po strukturi je torej ista kot tabela dejstev F_FO_STANJE_DAN_TRR z razliko, da ne vsebuje dejstva ST_TRANSAKCIJ.



Slika 19. Tabela dejstev F_FO_STANJE_DAN.

Slika 20 prikazuje fizični podatkovni model za spremljanje dnevnih stanj za vse vrste računov fizičnih oseb. Osnovna tabela dejstev F_FO_STANJE_DAN vsebuje šest tujih ključev (DAN_ID, RACUN_ID, VALUTA_ID, STATUS_ID, PE_ID, PRODUKT_ID) preko katerih je povezana z ustreznimi primarnimi ključi dimenzijskih tabel. Podmnožica tujih ključev (DAN_ID, RACUN_ID, VALUTA_ID) hkrati sestavlja tudi primarni ključ tabele dejstev. Model predstavlja globalni pogled poslovanja in iz tega razloga uporabljamo osnovni dimenzijski tabeli DIM_RACUN in DIM_PRODUKT. Dimenzija računov DIM_RACUN je večvrednostna dimenzija (podobno kot dimenzija DIM_RACUN_TRR pri dimenzijskem

modelu iz podpoglavja 3.3). Za določanje relacij med komitenti in računi uporabimo že obravnavano povezovalno tabelo REL_RACUN_KOMITENT.



Slika 20. Fizični podatkovni model: Dnevna stanja za vse vrste računov fizičnih oseb.

4. Sistem ETL

Sistem ETL predstavlja osnovo vsakega PS. Pravilno zasnovan sistem ETL omogoča zajem podatkov iz izvornih sistemov, uveljavlja kakovost podatkov in konsistenčne standarde, prilagaja podatke iz različnih izvorov in nazadnje predstavi podatke v obliki, ki omogoča končnim uporabnikom sprejemanje različnih odločitev. Čeprav je gradnja sistema ETL aktivnost, ki je večinoma nevidna končnim uporabnikom, z lahkoto porabi 70% virov potrebnih za implementacijo in vzdrževanje tipičnega PS [7]. Ena izmed najbolj pomembnih zgodnjih arhitekturnih odločitev je način gradnje sistema ETL. Lahko izberemo med komercialnimi orodji za gradnjo sistema ETL ali ročnim kodiranjem sistema ETL. Obe izbiri imata svoje prednosti in pomankljivosti [7]. V tem diplomskem delu se odločimo za ročno kodiranje sistema ETL. V nadaljevanju poglavja delno realiziramo sistem ETL za potrebe polnjenja dimenzijskih modelov razvitih v tretjem poglavju.

4.1 Struktura področja za pripravo podatkov

Arhitektura PS je sestavljena iz dveh delov: zalednega dela PS (angl. back room) in predstavitvenega dela PS (angl. front room). Zaledni del PS služi za zajem in pripravo podatkov in ne dovoljuje nobenih povpraševalnih servisov, saj je le to naloga predstavitvenega dela PS. Zaledni del PS je sestavljen iz izvornih sistemov in področja za pripravo podatkov, medtem ko je predstavitveni del PS sestavljen iz predstavitvenih strežnikov in uporabnikov (glej sliko 7).

Predpostavimo, da se področje za pripravo podatkov nahaja na Oracle 10.2 relacijski podatkovni bazi z imenom PROD_BANKA. Baza vsebuje tri sheme:

- ***Shema DW_STAGE:***

Shema DW_STAGE vsebuje začasne in arhivske tabele, kot tudi ročno kodirane postopke ETL. Začasne tabele služijo kot vmesne tabele pri polnjenju dimenzijskih tabel in tabel dejstev. V njih začasno prenesemo izvorne zapise, ob začetku izvajanja določenih postopkov ETL. Z uporabo začasnih tabel se izognemo ponovnemu zajemu izvornih podatkov in s tem tudi obremenjevanju izvornega sistema ob ponovitvi postopka ETL zaradi morebitnega izpada. Arhivske tabele služijo za arhiviranje uspešno napolnjenih izvornih zapisov, to je zapisov pri katerih ni prišlo do vnaprej definirane napake. Pod ročno kodiranimi postopki ETL razumemo programsko kodo znotraj katere realiziramo zajem, transformacijo in polnjenje izvornih podatkov v dimenzijske tabele in tabele dejstev. Logično je posamezen postopek ETL razdeljen na več posameznih korakov. Tako naprimer, razdelimo postopek polnjenja dimenzije komitentov na dva koraka.V prvem koraku prenesemo izvorne podatke v začasno tabelo. V drugem koraku pa zajete izvorne podatke ustrezno preoblikujemo in napolnemo v dimenzijo komitentov. Postopke ETL predstavimo v podpoglavju 4.4.

- ***Shema DW_PRESENT:***

Shema DW_PRESENT vsebuje vse dimenzijske tabele in tabele dejstev, ki smo jih obravnavali v tretjem poglavju.

- ***Shema DW_LOAD:***

Shema DW_LOAD vsebuje tabele in programsko kodo potrebno za realizacijo modula za ravnanje z napakami in avtomatizacijo postopkov ETL.

Poleg omenjenih baznih objektov (tabel in programske kode), vsebujejo sheme tudi ostale bazne objekte, ki so potrebni za uspešno delovanje sistema ETL (sekvence, pravice, sinonimi, samoprožilce itn.), katerih ne bomo posebej obravnavali.

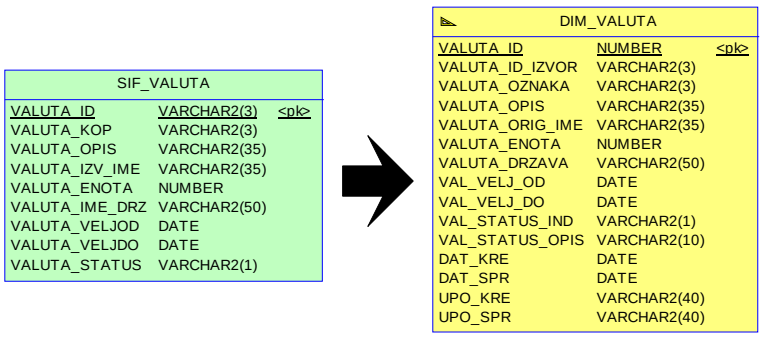
4.2 Preslikave izvornih atributov

Za potrebe polnjenja PPS, predpostavimo da podatke pridobimo iz enega samega izvornega sistema in sicer Oracle 10.2 relacijske podatkovne baze z imenom TRR. Baza podatkov vsebuje dve shemi, TRR in SIFRANTI, na katerih se nahajajo izvorne strukture, ki vsebujejo podatke o poslovanju s transakcijskimi računi.

Pred razvojem postopkov ETL je potrebno natančno določiti kako se atributi izvornih tabel preslikajo v atribute tabel dimenzijskega modela. Le to je naloga podatkovnega arhitekta, ki pred izdelavo same preslikave naredi podrobno analizo izvornih sistemov. Naloga razvijalca programskih rešitev je, da na osnovi podane preslikave izdela ustrezen postopek ETL. V nadaljevanju podamo primer preslikave na osnovi katere realiziramo polnjenje dimenzije DIM_VALUTA. Definicije ostalih preslikav podamo v prilogi 7.2.

- ***Preslikava za polnjenje dimenzije DIM_VALUTA:***

Dimenzijsko tabelo DW_PRESENT.DIM_VALUTA polnimo iz izvornega šifranta SIFRANTI.SIF_VALUTA, ki se nahaja na bazi TRR. Ker šifrant vsebuje relativno majhno število zapisov, osvežimo dimenzijo valut v celoti ob vsakem polnjenju. Iz istega razloga pri polnjenju dimenzije valut, ne uporabljamo začasne in arhivske tabele. Slika 21 prikazuje tok podatkov pri polnjenju dimenzije DIM_VALUTA. Iz slike sta razvidni strukturi izvorne in ciljne tabele.



Slika 21. Tok podatkov pri polnjenju dimenzije DIM_VALUTA.

V nadaljevanju podamo preslikavo izvornih atributov tabele SIF_VALUTA v atribute dimenzije DIM_VALUTA:

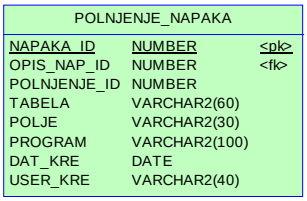
Izvor *(SIF_VALUTA SV)*	**Cilj** *(DIM_VALUTA DV)*	**Opombe**
-	DV.VALUTA_ID	Sekvenca

		SEQ_DIM_VALUTA.NEXTVAL
SV.VALUTA_ID	DV.VALUTA_ID_IZVOR	Where SV.VALUTA_ID = DV.VALUTA_ID_IZVOR
SV.VALUTA_KOP	DV.VALUTA_OZNAKA	
SV.VALUTA_OPIS	DV.VALUTA_OPIS	
SV.VALUTA_IZV_IME	DV.VALUTA_ORIG_IME	
SV.VALUTA_ENOTA	DV.VALUTA_ENOTA	
SV.VALUTA_IME_DRZ	DV.VALUTA_DRZAVA	
SV.VALUTA_VELJOD	DV.VAL_VELJ_OD	
SV.VALUTA_VELJDO	DV.VAL_VELJ_DO	
-	DV.VAL_STATUS_IND	IF SV.VALUTA_VELJDO IS NULL THEN ‘A’ ELSE ‘N’;
-	DV.VAL_STATUS_OPIS	IF SV.VALUTA_VELJDO IS NULL THEN ‘Aktivna’ ELSE ‘Neaktivna’;
-	DV.DAT_KRE	SYSDATE ob kreiranju zapisa
-	DV.DAT_SPR	SYSDATE ob spremembi zapisa
-	DV.UPO_KRE	USER ob kreiranju zapisa
-	DV.UPO_SPR	USER ob spremembi zapisa

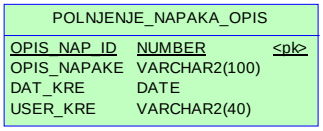
4.3 Modul za rokovanje z napakami polnjenja

Napake predstavljajo odstopanja izvornih podatkov od množice pravil definiranih s strani uporabnikov. Da bi lahko zagotovili kakovost podatkov, mora biti sistem ETL zasnovan tako, da zazna napake v izvornih podatkih, prepreči njihovo polnjenje v PPS in obvesti uporabnike o obstoju napake. Na ta način se lahko izvorni podatki popravijo in ob naslednjem polnjenju prenesejo v PS.

Za potrebe spremljanja napak polnjenja ustvarimo na shemi DW_LOAD dve tabeli in sicer POLNJENJE_NAPAKA in POLNJENJE_NAPAKA_OPIS. Slika 22 prikazuje definicijo tabele POLNJENJE_NAPAKA, medtem ko slika 23 prikazuje definicijo tabele POLNJENJE_NAPAKA_OPIS.



Slika 22. Definicija tabele POLNJENJE_NAPAKA.



Slika 23. Definicija tabele POLNJENJE_NAPAKA_OPIS.

Kot lahko opazimo na sliki 22, vsebuje tabela napak zaporedno številko polnjenja PS (POLNJENJE_ID), ki se enolično določi ob izvajanju postopka ETL. Preko tujega ključa POLNJENJE_ID je tabela povezana s tabelo POLNJENJE_INFO, iz katere pridobimo informacijo o posameznih polnjenjih PS. Tabelo POLNJENJE_INFO podrobno obravnavamo v podpoglavju 4.5. Atribut NAPAKA_ID služi kot enolični identifikator napake polnjenja. Preko tujega ključa OPIS_NAP_ID je tabela povezana s tabelo, ki vsebuje opise napak POLNJENJE_NAPAKA_OPIS. Atribut TABELA hrani informacijo o tem v kateri začasni tabeli se nahaja izvorni zapis, ki vsebuje napako. Atribut POLJE pove ob kontroli katerega atributa tabele TABELA, smo zaznali napako. Atribut PROGRAM nam pove katera procedura (korak postopka ETL) se je izvajala, ko je bila ugotovljena napaka.

Na isti shemi ustvarimo tudi programski paket PKG_ERROR_UTIL. Definicija paketa je podana v prilogi 7.3. Uporaba paketa je precej enostavna. Kadar zaznamo napako pri izvornem zapisu, jo z uporabo funkcije *PKG_ERROR_UTIL.zapisi_napako* zapišemo v tabelo POLNJENJE_NAPAKA. Podani opis napake (parameter funkcije) pred tem zapišemo v POLNJENJE_NAPAKA_OPIS. Funkcija nazadnje vrne številko napake, ki jo priredimo atributu NAPAKA_ID, zapisa začasne tabele, pri katerem smo napako zaznali. Uporabo programskega paketa PKG_ERROR_UTIL ponazorimo v podpoglavju 4.4.

4.4 Realizacija postopkov ETL

PPS predstavljeno v tretjem poglavju osvežujemo z izvajanjem šestih postopkov ETL, ki jih razvijemo v nadaljevanju podpoglavja.

4.4.1 Postopek za polnjenje dimenzije datumov

Postopek uporabljamo za polnjenje dimenzije DIM_DAN. Avtomatično se izvede enkrat na leto in sicer prvi dan v letu ob 00:00 h zjutraj. Sestavljen je iz enega samega koraka, ki ga implementiramo s proceduro *PKG_DIM_DAN.load_dim_dan* (glej prilogo 7.4.1). Procedura najprej poišče največji DATUM v dimenziji DIM_DAN. Od tega datuma naprej napolni dimenzijo DIM_DAN s podatki za eno leto vnaprej. Tako naprimer, če je največji DIM_DAN.DATUM = 31.12.2008, bo procedura napolnila dimenzijo DIM_DAN s podatki za obdobje DATUM = 1.1.2009 do DATUM = 31.12.2009. Ob prvem polnjenju je dimenzija prazna in jo napolnimo s podatki za leto 2008.

4.4.2 Postopek za polnjenje dimenzij iz izvornih šifrantov

Postopek uporabljamo za polnjenje sedmih dimenzij iz izvornih šifrantov, ki se nahajajo na shemi SIFRANTI izvorne baze TRR. Avtomatično se izvede enkrat na dan in sicer ob 00:00 h zjutraj. Postopek je sestavljen iz naslednjih korakov:

- ***Korak 1***: Polnjenje dimenzije DIM_VALUTA
- ***Korak 2***: Polnjenje dimenzije DIM_TRANSAKCIJA
- ***Korak 3***: Polnjenje dimenzije DIM_LOKACIJA
- ***Korak 4***: Polnjenje dimenzije DIM_POSLOVNA_ENOTA
- ***Korak 5***: Polnjenje dimenzije DIM_STATUS
- ***Korak 6***: Polnjenje dimenziji DIM_PRODUKT_TRR in DIM_PRODUKT

Kot lahko opazimo, vsak posamezen korak polni eno izmed dimenzij, navedeno v korakih zgoraj. Prvi korak postopka implementiramo s proceduro *PKG_DIM_VALUTA.load_dim_valuta* (glej prilogo 7.4.2), ki jo razvijemo na osnovi preslikave izvornih atributov podane v podpoglavju 4.2. Procedura osveži dimenzijo DIM_VALUTA tako, da iz izvorne tabele SIFRANTI.SIF_VALUTA prenese vse zapise, ki še ne obstajajo v dimenziji. Zapise, ki v dimenziji že obstajajo osvežimo z izvornimi podatki, ne glede na to ali so se izvorni podatki spremenili. Pri realizaciji si pomagamo z Oraclovim stavkom *MERGE*, ki glede na podani pogoj ujemanja izvorne in dimenzijske tabele zagotovi ažuriranje oziroma vstavljanje zapisov v dimenzijo. V primeru, da se zapisa ujameta glede na atributa SIF_VALUTA.VALUTA_ID in DIM_VALUTA.VALUTA_ID_IZVOR je pogoj stavka *MERGE* izpolnjen, kar pomeni da se izvede ažuriranje zapisa dimenzije določenega z atributom VALUTA_ID_IZVOR. V primeru, da se zapisa ne ujameta glede na atributa SIF_VALUTA.VALUTA_ID in DIM_VALUTA.VALUTA_ID_IZVOR, pa izvorni zapis določen s primarnim ključem SIF_VALUTA.VALUTA_ID, vstavimo v dimenzijo DIM_VALUTA. Na podoben način realiziramo tudi ostale korake (2 – 6) postopka ETL in jih iz tega razloga ne obravnavamo. Preslikave izvornih atributov na osnovi katerih realiziramo ostale korake (2 – 6) podamo v prilogi 7.2.

4.4.3 Postopek za polnjenje dimenzije komitentov

Postopek uporabljamo za polnjenje dimenzije DIM_FO_KOMITENT. Avtomatično se izvede enkrat na dan in sicer ob 00:30 h zjutraj. Postopek je sestavljen iz dveh korakov:

- ***Korak 1***: Zajem izvornih podatkov v začasno tabelo TEMP_FO_KOMITENT
- ***Korak 2***: Preoblikovanje in polnjenje izvornih podatkov v DIM_FO_KOMITENT

Preslikavo izvornih atributov, na osnovi katere realiziramo postopek, podamo v prilogi 7.2. V prilogi je podana tudi definicija izvornih struktur.

V prvem koraku postopka, ki ga implementiramo s proceduro *PKG_DIM_FO_KOMITENT.load_temp_fo_komitent* (glej prilogo 7.4.3), zajamemo izvorne podatke in jih prenesemo v začasno tabelo TEMP_FO_KOMITENT. Začasna tabela je po strukturi enaka izvornemu pogledu VW_REG_FIZOS z razliko, da vsebuje še atribut NAPAKA_ID. Arhivska tabela ARH_FO_KOMITENT, v kateri hranimo že obdelane izvorne zapise, je po strukturi tudi enaka izvornem pogledu, le da vsebuje še atribut DAT_ARH. Pomen atributov NAPAKA_ID in DAT_ARH obravnavamo pri drugem koraku postopka. Izvorni pogled VW_REG_FIZOS vsebuje atribut SPREMEMBA. Ob vsakem ažuriranju ali dodajanju novega komitenta na izvoru, se atribut VW_REG_FIZOS.SPREMEMBA znova enolično določi. Pred samim zajemom izvornih podatkov, procedura zbriše vse zapise začasne tabele. Zaradi velike količine izvornih podatkov, prenesemo v začasno tabelo samo tiste izvorne zapise, ki ne obstajajo v arhivski tabeli ARH_FO_KOMITENT, kar ugotovimo s primerjanjem atributov VW_REG_FIZOS.SPREMEMBA in ARH_FO_KOMITENT.SPREMEMBA. Ob samemu prenosu izvornih podatkov naredimo tudi preoblikovanje črkovnih nizov, tako da odstranimo morebitne presledke, ki se nahajajo pred ali za nizom.

Drugi korak postopka implementiramo s proceduro *PKG_DIM_FO_KOMITENT.load_dim_fo_komitent* (glej prilogo 7.4.3). Procedura zaporedno obdeluje zapise začasne tabele TEMP_FO_KOMITENT. V nadaljevanju besedila označujemo začasno tabelo s TEMP in dimenzijo komitentov z DK. Za trenutno obdelovani zapis začasne tabele najprej ugotovimo

ali komitent že obstaja v dimenziji komitentov, kar naredimo s primerjanjem atributov DK.KOMITENT_ID_IZVOR in TEMP.KOMITENT_ID. V primeru, da komitent že obstaja v dimenziji, si zapomnimo njegov enolični identifikator DK.KOMITENT_ID. V primeru, da komitent ne obstaja v dimenziji komitentov, s pomočjo sekvence generiramo novo šifro komitenta. Procedura nadaljuje z izvajanjem kontrol atributov začasne tabele v okviru katerih preverjamo ali podatki zadoščajo definirani množici pravil. V primeru uspešno izvedenih kontrol, opravimo morebitne transformacije podatkov. V nadaljevanju podamo nekaj tipičnih kontrol in transformacij ki jih izvajamo v proceduri:

- *Kontrola atributa TEMP.EMSO*: preverimo ali je vrednost atributa EMSO sestavljena iz samih numeričnih znakov ter ali je dolžine 13. Kontrolo opravimo s funkcijo *PKG_DIM_FO_KOMITENT.check_emso* (glej prilogo 7.4.3).
- *Kontrola atributa TEMP.IME*: ker je ime komitenta obvezen podatek, preverimo ali je vrednost atributa IME neničelna.
- *Transformacija atributa TEMP.IME*: vrednost atributa zapišemo z velikimi črkami. To transformacijo uporabljamo pri vseh črkovnih atributih, saj jih želimo v bazi imeti zapisane z velikimi črkami.
- *Transformacija atributa TEMP.TITULA*: okrajšave titul preoblikujemo v besede. Naprimer DR ali DR. preoblikujemo v DOKTOR. Transformacijo opravimo s funkcijo *PKG_DIM_FO_KOMITENT.edit_titula* (glej prilogo 7.4.3).

Če določen atribut ne zadošča kontroli, ki jo nad njim izvajamo, pokličemo funkcijo *PKG_ERROR_UTIL.zapisi_napako* (glej podpoglavje 4.3) z naslednjimi parametri: ime začasne tabele, zaporedna številka polnjenja PS, ime procedure ki se izvaja, opis napake in ime atributa, ki ne zadošča kontroli. Za klicem funkcije sprožimo izjemo *SKIP_ROW,* s katero priredimo atributu TEMP.NAPAKA_ID enolično številko napake, ki jo vrne funkcija *PKG_ERROR_UTIL.zapisi_napako*. Po uspešno opravljenih kontrolah in transformacijah posodobimo dimenzijo komitentov. Če gre za novega komitenta, dodamo dimenziji komitentov nov zapis, če gre za obstoječega komitenta pa ustrezen zapis dimenzije komitentov ažuriramo. Uspešno obdelan zapis začasne tabele nato arhiviramo. Arhiviranje realiziramo s proceduro *PKG_DIM_FO_KOMITENT.arh_temp_fo_komitent* (glej prilogo 7.4.3), ki prenese zapis iz začasne tabele v arhiv ARH_FO_KOMITENT. Atributu ARH_FO_KOMITENT.DAT_ARH priredimo datum in čas arhiviranja zapisa. Potem ko smo zapis arhivirali, ga zbrišemo iz začasne tabele. Klic izjeme *SKIP_ROW* (v primeru da smo zaznali napako) zagotovi neizvajanje dela kode, ki skrbi za polnjenje dimenzije komitentov ter arhiviranje in brisanje zapisa iz začasne tabele. Ko se drugi korak postopka konča, ostanejo v začasni tabeli samo zapisi, pri katerih smo zaznali napake. Ob povezovanju začasne tabele s tabelo POLNJENJE_NAPAKA po atributu NAPAKA_ID pridobimo informacijo o vrsti napake.

4.4.4 Postopek za polnjenje dimenzije transakcijskih računov

Postopek uporabljamo za polnjenje dimenziji DIM_RACUN_TRR in DIM_RACUN. Avtomatično se izvede enkrat na dan in sicer ob 01:00 h zjutraj. Postopek je sestavljen iz dveh korakov:

- ***Korak 1***: Zajem izvornih podatkov v začasno tabelo TEMP_RACUN_TRR

- ***Korak 2***: Preoblikovanje in polnjenje izvornih podatkov v DIM_RACUN_TRR in DIM_RACUN

Preslikavo izvornih atributov na osnovi katere realiziramo postopek podamo v prilogi 7.2. V prilogi je podana tudi definicija izvornih struktur.

Prvi korak postopka implementiramo s proceduro *PKG_DIM_RACUN_TRR.load_temp_racun_trr* (glej prilogo 7.4.4). Logika zajemanja izvornih podatkov je praktično ista kot pri prvem koraku postopka ETL, obravnavanega v razdelku 4.4.3. Razlika je samo ta, da sedaj uporabljamo izvor podatkov VW_PARTIJA_TRR_DW, začasno tabelo TEMP_RACUN_TRR in arhivsko tabelo ARH_RACUN_TRR.

Drugi korak postopka implementiramo s proceduro *PKG_DIM_RACUN_TRR.load_dim_racun_trr* (glej prilogo 7.4.4). Logika zaporednega obdelovanja zapisov začasne tabele TEMP_RACUN_TRR je praktično ista kot pri drugem koraku postopka ETL obravnavanega v razdelku 4.4.3. Razlika je v tem, da izvajamo drugačne kontrole in transformacije izvornih atributov ter polnimo obdelane zapise v dimenziji DIM_RACUN in DIM_RACUN_TRR. Posebnost je tudi ta, da ob odprtju novega transakcijskega računa dodamo zapis v povezovalno tabelo REL_RACUN_KOMITENT, pri čemer lastništvo računa (atribut LASTNIK) ugotavljamo glede na izvorni atribut IND_LASTNIK. Ob odprtju novega računa dodamo tudi zapisa v tabeli dejstev F_FO_STANJE_DAN in F_FO_STANJE_DAN_TRR, pri čemer dodelimo atributoma DNEVNO_STANJE in DNEVNO_STANJE_VAL vrednost 0.

4.4.5 Postopek za polnjenje dimenzije kartic TRR

Postopek uporabljamo za polnjenje dimenzije DIM_KARTICA_TRR. Avtomatično se izvede enkrat na dan in sicer ob 01:30h h zjutraj. Postopek je sestavljen iz dveh korakov:

- ***Korak 1***: Zajem izvornih podatkov v začasno tabelo TEMP_KARTICA_TRR
- ***Korak 2***: Preoblikovanje in polnjenje izvornih podatkov v DIM_KARTICA_TRR

Preslikavo izvornih atributov na osnovi katere realiziramo postopek podamo v prilogi 7.2. V prilogi je podana tudi definicija izvornih struktur.

Prvi korak postopka implementiramo s proceduro *PKG_DIM_KARTICA_TRR.load_temp_kartica_trr* (glej prilogo 7.4.5). Logika zajemanja izvornih podatkov je praktično ista kot pri prvem koraku postopka ETL, obravnavanega v razdelku 4.4.3. Razlika je samo ta, da sedaj uporabljamo izvor podatkov VW_KARTICE_TRR_DW, začasno tabelo TEMP_KARTICA_TRR in arhivsko tabelo ARH_KARTICA_TRR.

Drugi korak postopka implementiramo s proceduro *PKG_DIM_KARTICA_TRR.load_dim_kartica_trr* (glej prilogo 7.4.5). Logika zaporednega obdelovanja zapisov začasne tabele TEMP_KARTICA_TRR je praktično ista kot pri drugem koraku postopka ETL obravnavanega v razdelku 4.4.3. Razlika je v tem, da tokrat izvajamo drugačne kontrole in transformacije atributov ter polnimo obdelane zapise v dimenzijo DIM_KARTICA_TRR.

4.4.6 Postopek za polnjenje prometa in stanj transakcijskih računov

Postopek uporabljamo za polnjenje tabel dejstev F_FO_TRANS_TRR, F_FO_STANJE_DAN in F_FO_STANJE_DAN_TRR. Avtomatično se izvede enkrat na dan in sicer ob 02:00h h zjutraj. Postopek je sestavljen iz dveh korakov:

- ***Korak 1***: Zajem izvornih podatkov v začasno tabelo TEMP_TRR_KNJIZBE
- ***Korak 2***: Preoblikovanje in polnjenje izvornih podatkov v tabele dejstev

Preslikavo izvornih atributov na osnovi katere realiziramo postopek podamo v prilogi 7.2. V prilogi je podana tudi definicija izvornih struktur.

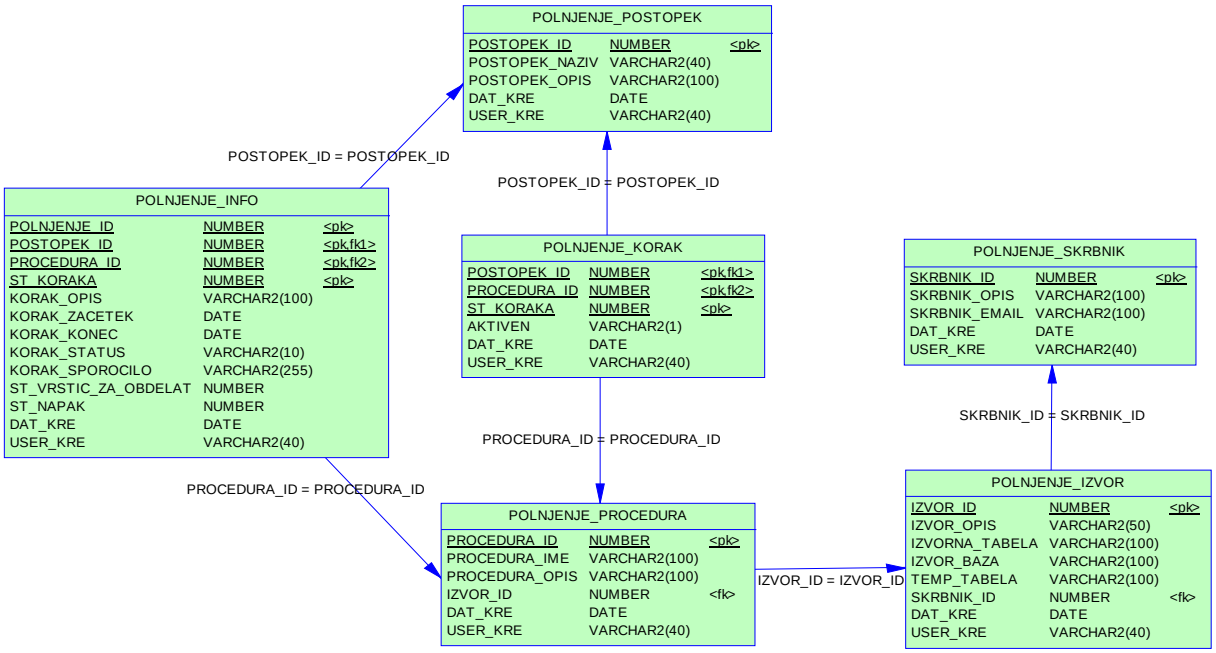
Prvi korak postopka implementiramo s proceduro *PKG_F_FO_TRANS_TRR.load_temp_trr_knjizbe* (glej prilogo 7.4.6). Logika zajemanja izvornih podatkov je praktično ista kot pri prvem koraku postopka ETL, obravnavanega v razdelku 4.4.3. Razlika je samo ta, da sedaj uporabljamo izvor podatkov VW_TRR_KNJIZBE_DW, začasno tabelo TEMP_TRR_KNJIZBE in arhivsko tabelo ARH_TRR_KNJIZBE.

Drugi korak postopka implementiramo s proceduro *PKG_F_FO_TRANS_TRR.load_f_fo_trans_trr* (glej prilogo 7.4.6). Procedura zaporedno obdeluje zapise začasne tabele TEMP_TRR_KNJIZBE. Ob izvajanju tega koraka so vse dimenzije PPS že osvežene z novimi podatki. Zaradi morebitnih napak pri izvajanju postopkov ETL, s katerimi polnimo dimenzije, procedura kontrolira obstoj računa, dneva, valute, transakcije in lokacije prometa v dimenzijah DIM_RACUN_TRR, DIM_DAN, DIM_VALUTA, DIM_TRANSAKCIJA in DIM_LOKACIJA. Tako naprimer za ugotavljanje obstoja transakcijskega računa, poiščemo zapis v dimenziji DIM_RACUN_TRR, katerega vrednost atributa RACUN_ID_IZVOR je enaka vrednosti atributa PARTIJA_ID trenutno obdelovanega zapisa. Če ne najdemo zapisa v ustrezni dimenziji, zapišemo napako in sprožimo izjemo *SKIP_ROW*. Z izjemo preskočimo del kode, ki skrbi za polnjenje tabele dejstev F_FO_TRANS_TRR, ažuriranje stanj v tabelah dejstev F_FO_STANJE_DAN in F_FO_STANJE_DAN_TRR, ter arhiviranje in brisanje zapisa začasne tabele. Izjema tudi priredi enolično številko napake trenutno obdelovanem zapisu začasne tabele (TEMP.NAPAKA_ID = *PKG_ERROR_UTIL.zapisi_napako*) in zagotovi, da zapisi z napakami obležijo v začasni tabeli ob koncu postopka ETL. V primeru da napake ne zaznamo, dodamo transakcijo v transakcijsko tabelo F_FO_TRANS_TRR. Sledi klic procedure *PKG_F_FO_STANJE_DAN_TRR.azuriraj_stanje* (glej prilogo 7.4.7), ki ob podanih parametrih DAN_ID, RACUN_ID, VALUTA_ID, TRANS_ID in ZNESEK, ažurira stanja transakcijskega računa v tabelah dejstev F_FO_STANJE_DAN_TRR in F_FO_STANJE_DAN. Ažuriranje stanja se začne z ugotavljanjem tipa transakcije, ki ga poiščemo v dimenziji DIM_TRANSAKCIJA glede na podani ključ TRANS_ID. V primeru priliva ostane vrednost zneska pozitivna, v primeru odliva pa znesku zamenjamo predznak. Sledi določanje vrednosti aributov STATUS_ID, PRODUKT_ID in PE_ID, ki jih preberemo iz dimenzije DIM_RACUN_TRR glede na podani ključ RACUN_ID. V nadaljevanju ažurira procedura vsa dnevna stanja podanega računa v F_FO_STANJE_DAN_TRR in F_FO_STANJE_DAN, pri katerih je atribut DAN_ID večji ali enak dnevu (podanemu parametru DAN_ID) izvršene transakcije, tako da stanju računa prišteje znesek transakcije. Če za podani parameter DAN_ID v F_FO_STANJE_DAN_TRR in F_FO_STANJE_DAN zapis še ne obstaja, ga dodamo. Prikazani način ažuriranja stanj upošteva tudi vse transakcije, ki pridejo v PPS s zakasnitvijo. Ko končamo z ažuriranjem stanj, uspešno obdelan zapis začasne tabele arhiviramo. Arhiviranje izvedemo s proceduro *PKG_F_FO_TRANS_TRR.arh_temp_trr_knjizbe* (glej prilogo 7.4.6), ki prenese zapis iz začasne tabele v arhiv ARH_TRR_KNJIZBE. Potem, ko smo zapis arhivirali, ga zbrišemo iz začasne tabele.

4.5 Avtomatizacija izvajanja postopkov ETL

V tem podpoglavju predstavimo del sistema ETL, ki skrbi za avtomatično izvajanje postopkov ETL.

Na shemi DW_LOAD ustvarimo šest tabel katerih strukturo in medsebojno povezanost prikažemo na sliki 24. Vse tabele, razen tabele POLNJENJE_INFO vzdržujemo ročno.



Slika 24. Struktura in medsebojna povezanost tabel za avtomatizacijo postopkov ETL.

V nadaljevanju na kratko opišemo pomen posamezne tabele:

- Tabela **POLNJENJE_SKRBNIK**:

  Tabela vsebuje opise skrbnikov izvornih podatkov, kot tudi njihove elektronske naslove. V primeru izpada postopka ETL obvestimo skrbnike, da PPS ni bilo osveženo z novimi podatki.

- Tabela **POLNJENJE_IZVOR**:

  Tabela vsebuje opise in imena izvornih tabel, njihovo lokacijo na izvorni bazi ter ime začasne tabele v katero prenesemo izvorne zapise. Preko tujega ključa SKRBNIK_ID je tabela povezana s tabelo POLNJENJE_SKRBNIK.

- Tabela **POLNJENJE_PROCEDURA**:

  Tabela vsebuje opise in imena procedur s katerimi realiziramo posamezne korake postopka ETL. Preko tujega ključa IZVOR_ID je tabela povezana s tabelo POLNJENJE_IZVOR. Preko tujega ključa lahko hitro ugotovimo iz katerega izvora polnimo podatke v PPS in katero začasno tabelo pri tem uporabljamo.

- Tabela **POLNJENJE_POSTOPEK**:

  Tabela vsebuje nazive in opise postopkov ETL.

- Tabela **POLNJENJE_KORAK**:

  Tabela definira korake določenega postopka ETL opredeljenega z atributom POSTOPEK_ID. Posamezen korak postopka je opredeljen z atributom PROCEDURA_ID. Vrstni red izvajanja korakov določamo z atributom ST_KORAKA. S pomočjo atributa AKTIVEN določimo ali naj se določen korak postopka ETL izvede (vrednost D) ali ne (vrednost N). Preko tujega ključa POSTOPEK_ID je tabela povezana s tabelo POLNJENJE_POSTOPEK in preko tujega ključa PROCEDURA_ID s tabelo POLNJENJE_PROCEDURA.

- Tabela **POLNJENJE_INFO**:

  Tabela predstavlja dnevnik polnjenja PS. Atribut POLNJENJE_ID določa zaporedno številko polnjenja PS. Za vsak izvedeni korak postopka ETL opredeljenega z atributom POSTOPEK_ID hranimo opis koraka ter začetek in konec izvajanja koraka. Atribut KORAK_STATUS lahko zavzame eno izmed treh možnih vrednosti in sicer USPEŠNO (brez napak polnjenja), NAPAKE (z napakami polnjenja) in IZPAD (postopek ETL se je ob izvrševanju koraka nenormalno končal). Atribut KORAK_SPOROCILO uporabljamo za bolj podrobno opisovanje statusa koraka. Atribut ST_VRSTIC_ZA_OBDELAT vsebuje število zapisov, ki so se v koraku obdelale, medtem ko atribut ST_NAPAK pove koliko obdelanih zapisov je vsebovalo napake.

Na shemi DW_LOAD ustvarimo tudi programski paket PKG_LOAD_UTIL. Definicija paketa je podana v prilogi 7.5. Proceduro *PKG_LOAD_UTIL.izvedi_postopek(p_postopek_id NUMBER)* uporabljamo za izvajanje postopka ETL, ki je določen s parametrom procedure p_postopek_id. Procedura na osnovi podatkov prisotnih v tabelah POLNJENJE_PROCEDURA, POLNJENJE_KORAK, POLNJENJE_POSTOPEK izvede posamezne korake postopka ETL v pravilnem vrstnem redu, kot ga določa atribut POLNJENJE_KORAK.ST_KORAKA. Ob izvajanju posameznega koraka procedura zapisuje podatke o samem izvajanju v tabelo POLNJENJE_INFO. Ob morebitnemu izpadu postopka ETL, pošlje procedura elektronsko pošto skrbniku izvora podatkov pri polnjenju katerega je prišlo do izpada.

Za vsak postopek ETL, katerega izvajanje želimo avtomatizirati, ustvarimo na bazi bazno opravilo (angl. database job). Bazno opravilo skrbi za redno izvajanje podane kode v definiranem časovnem intervalu. Če bi denimo želeli avtomatizirati postopek ETL, definiran v v tabeli POLNJENJE_POSTOPEK z atributom POSTOPEK_ID = 5, bi na bazi ustvarili naslednje bazno opravilo:

```
DECLARE
  X NUMBER;
BEGIN
  SYS.DBMS_JOB.SUBMIT
    ( job       => X,
      what      => 'dw_load.pkg_load_util.izvedi_postopek(5);',
      next_date => to_date('10.10.2008 01:00:00','mm/dd/yyyy hh24:mi:ss'),
      interval  => 'TRUNC(SYSDATE+1)+1/24',
      no_parse  => FALSE);
END;
```

Bazno opravilo definiramo s pomočjo sistemske procedure *SYS.DBMS_JOB.SUBMIT*. Parameter procedure *what* definira kodo, medtem ko parameter procedure *interval* definira časovni interval izvajanja kode. V našem primeru bo bazno opravilo poskrbelo, da se vsak dan ob enih zjutraj požene procedura *pkg_load_util.izvedi_postopek(5)*, ki izvede željeni postopek ETL.

5. Primeri uporabe podatkovnega skladišča

V tem poglavju predstavimo pet primerov uporabe podatkovnega skladišča. Znotraj posameznega primera podamo odgovor na določeno poslovno vprašanje. Kot odgovor razumemo podatke, ki jih s pomočjo vprašanja SQL pridobimo iz dimenzijskih modelov razvitih v tretjem poglavju. V realnem svetu bi uporabniki do podatkov PS dostopali preko raznih aplikacij implementiranih v predstavitvenem delu PS.

Primer 1:

Denimo, da nas zanima število transakcij na TRR glede na lokacijo izvršitve transakcije. Podatke želimo imeti razdeljene po posameznih kvartalih leta, ki ga podamo kot parameter. Odgovor pridobimo z naslednjim vprašanjem SQL:

```
SELECT   dan.kvartal_opis AS "Kvartal", dan.leto_opis AS "Leto",
       (SELECT lokacija_opis
          FROM dim_lokacija
         WHERE lokacija_ind = lok.lokacija_ind)
                                  AS "Lokacija transakcije",
       COUNT (trans.knjizba_id) AS "Število transakcij"
  FROM dim_dan dan, dim_lokacija lok, f_fo_trans_trr trans
 WHERE trans.dan_id = dan.dan_id
   AND trans.lokacija_id = lok.lokacija_id
   AND dan.leto_opis = :P_LETO_OPIS
GROUP BY dan.kvartal_opis,
       dan.kvartal_v_letu,
      dan.leto_opis
       lok.lokacija_ind,
       lok.lokacija_opis
ORDER BY lok.lokacija_opis, dan.kvartal_v_letu;
```

Lahko opazimo, da smo zgornje vprašanje SQL omejili po atributu *DIM_DAN.leto_opis* s podanim parametrom P_LETO_OPIS. Hkrati uporabljamo grupiranje rezultata po atributih *DIM_DAN.kvartal_opis*, *DIM_DAN.kvartal_v_letu*, *DIM_DAN.leto_opis* , *DIM_LOKACIJA-.lokacija_ind* in *DIM_LOKACIJA.lokacija_opis*.V SELECT delu vprašanja SQL uporabimo funkcijo *COUNT*, ki sešteje vse transakcije glede na atribute grupiranja rezultata. Primer rezultata, ki ga vrne zgornje povpraševanje je:

Kvartal	*Leto*	*Lokacija transakcije*	*Število transakcij*
Januar – Marec	*2007*	*BANKOMAT*	*9123453*
April – Junij	*2007*	*BANKOMAT*	*8987112*
Julij – September	*2007*	*BANKOMAT*	*6102009*
Oktober – December	*2007*	*BANKOMAT*	*10089891*
Januar – Marec	*2007*	*POSLOVNA ENOTA*	*134980*
April – Junij	*2007*	*POSLOVNA ENOTA*	*144221*
Julij – September	*2007*	*POSLOVNA ENOTA*	*100210*
Oktober – December	*2007*	*POSLOVNA ENOTA*	*150233*
Januar – Marec	*2007*	*SPLETNA BANKA*	*387001*
April – Junij	*2007*	*SPLETNA BANKA*	*429871*

Julij – September	*2007*	*SPLETNA BANKA*	*298123*
Oktober – December	*2007*	*SPLETNA BANKA*	*489998*
...	...	...	...

Poslovnih vprašanj vezanih na lokacijo izvršitve transakcije TRR je precej. Tako naprimer za potrebe trženja spletnega bančništva poiščemo tiste komitente banke, ki večino transakcij izvedejo v poslovnih enotah in katerih starost je med 20 in 40 let.

Primer 2:

Denimo, da nas zanima sumarni promet TRR po poslovnih enotah glede na tip transakcije in valuto transakcije. Podatke želimo imeti razdeljene po posameznih mesecih leta, ki ga podamo kot parameter. Odgovor pridobimo z naslednjim vprašanjem SQL:

```
SELECT   dan.mesec_opis AS "Mesec", dan.leto_opis AS "Leto",
         pe.pe_opis AS "Poslovna enota",
         topis.tip_trans AS "Tip transakcije",
         SUM (ftrans.znesek) AS "Skupni promet",
         dv.valuta_oznaka AS "Valuta"
    FROM dim_dan dan,
         dim_transakcija topis,
         dim_racun_trr rac,
         f_fo_trans_trr ftrans,
         dim_poslovna_enota pe,
         dim_valuta dv
   WHERE ftrans.trans_id = topis.trans_id
     AND ftrans.racun_id = rac.racun_id
     AND ftrans.dan_id = dan.dan_id
     AND rac.zadnja_pe_id = pe.pe_id
     AND dv.valuta_id = ftrans.valuta_id
     AND dan.leto_opis = :P_LETO_OPIS
GROUP BY dan.mesec_opis,
         dan.mesec_v_letu,
         dan.leto_opis,
         topis.tip_trans,
         pe.pe_opis,
         dv.valuta_oznaka
ORDER BY  pe.pe_opis, dan.mesec_v_letu, topis.tip_trans, dv.valuta_oznaka;
```

Lahko opazimo, da smo zgornje vprašanje SQL omejili po atributu *DIM_DAN.leto_opis* s podanim parametrom P_LETO_OPIS. Hkrati uporabljamo grupiranje rezultata po atributih *DIM_DAN.mesec_opis*, *DIM_DAN.mesec_v_letu*, *DIM_DAN.leto_opis* , *DIM_TRANS-AKCIJA.tip_trans*, *DIM_POSLOVNA_ENOTA.pe_opis* in *DIM_VALUTA.valuta_oznaka*.V SELECT delu vprašanja SQL uporabimo funkcijo *SUM*, ki sešteje vse zneske transakcij glede na atribute grupiranja rezultata. Primer rezultata, ki ga vrne zgornje povpraševanje je:

Mesec	***Leto***	***Poslovna enota***	***Tip transakcije***	***Skupni promet***	***Valuta***
Januar	*2007*	*PE Ljubljana 1*	*Priliv*	*3245980*	*EUR*
Februar	*2007*	*PE Ljubljana 1*	*Priliv*	*2113908*	*EUR*

Marec	*2007*	*PE Ljubljana 1*	*Priliv*	*4908123*	*EUR*
...	...	...	...	...	...
Januar	*2007*	*PE Koper 1*	*Priliv*	*924908*	*USD*
Februar	*2007*	*PE Koper 1*	*Priliv*	*822987*	*USD*
Marec	*2007*	*PE Koper 1*	*Priliv*	*762112*	*USD*
...	...	...	...	...	...
Oktober	*2007*	*PE Maribor 3*	*Odliv*	*123234*	*CHF*
November	*2007*	*PE Maribor 3*	*Odliv*	*102332*	*CHF*
December	*2007*	*PE Maribor 3*	*Odliv*	*89221*	*CHF*

Pridobljene podatke lahko uporabimo za ugotavljanje trenda rasti oziroma opada prometa TRR po poslovnih enotah in valutah oziroma za uravnavanje likvidnosti glede na zgodovinske podatke o prilivih in odlivih posameznih poslovnih enot.

Primer 3:

Denimo, da nas zanima sumarno nedovoljeno stanje na evrskih TRR, ter število TRR, ki so v nedovoljenem stanju. Podatke želimo imeti razdeljene po posameznih mesecih leta, ki ga podamo kot parameter. Odgovor pridobimo z naslednjim vprašanjem SQL:

```
SELECT   dan.mesec_opis "Mesec", dan.leto_opis "Leto",
         COUNT (rac.racun_id) "Število TRR",
         SUM (ds.dnevno_stanje + rac.limit_znesek) "Skupni nedovoljeno stanje",
         'EUR' AS "Valuta"
    FROM dim_racun_trr rac,
         f_fo_stanje_dan_trr ds,
         dim_dan dan,
         dim_valuta val
   WHERE rac.racun_id = ds.racun_id
     AND ds.dan_id = dan.dan_id
     AND ds.valuta_id = val.valuta_id
     AND val.valuta_oznaka = 'EUR'
     AND dan.leto_opis = :p_leto_opis
     AND ds.dan_id =
           (SELECT MAX (dan_id)
              FROM f_fo_stanje_dan_trr
             WHERE racun_id = ds.racun_id
               AND dan_id IN (SELECT dan_id
                                FROM dim_dan
                               WHERE mesec_opis = dan.mesec_opis)
               AND valuta_id = ds.valuta_id)
     AND ds.dnevno_stanje < (-rac.limit_znesek)
GROUP BY dan.mesec_opis, dan.mesec_v_letu, dan.leto_opis
ORDER BY dan.mesec_v_letu;
```

Lahko opazimo, da smo zgornje vprašanje SQL omejili po atributu *DIM_DAN.leto_opis* s podanim parametrom P_LETO_OPIS in po atributu *DIM_VALUTA.valuta_oznaka* = ‘EUR’. Hkrati uporabljamo grupiranje rezultata po atributih *DIM_DAN.mesec_opis*, *DIM_DAN-.mesec_v_letu* in *DIM_DAN.leto_opis*. V SELECT delu vprašanja SQL uporabimo funkcijo

SUM, ki sešteje vsa nedovoljena stanja računov, glede na mesec v katerega sodijo. Nedovoljeno stanje na določenem računu je presežek razpoložljivega stanja, to je zneska limita, ki je definiran v dimenziji *DIM_RACUN_TRR.limit_znesek*. V SELECT delu vprašanja uporabimo tudi funkcijo *COUNT*, s pomočjo katere pridobimo število TRR, ki imajo v določenem mesecu nedovoljeno stanje. Primer rezultata, ki ga vrne zgornje povpraševanje je:

Mesec	*Leto*	*Število TRR*	*Skupno nedovoljeno stanje*	*Valuta*
Januar	*2007*	*12511*	*37533*	*EUR*
Februar	*2007*	*11370*	*35114*	*EUR*
Marec	*2007*	*11041*	*34885*	*EUR*
April	*2007*	*9487*	*30474*	*EUR*
Maj	*2007*	*10844*	*31447*	*EUR*
Junij	*2007*	*11447*	*32551*	*EUR*
Julij	*2007*	*13554*	*39558*	*EUR*
Avgust	*2007*	*14114*	*44114*	*EUR*
September	*2007*	*15556*	*48114*	*EUR*
Oktober	*2007*	*11741*	*30114*	*EUR*
November	*2007*	*12998*	*36998*	*EUR*
December	*2007*	*14558*	*49551*	*EUR*

Primer 4:

Denimo, da nas zanimajo vsi komitenti, ki v določenem obdobju niso izvršili nobene transakcije. Za vsakega komitenta želimo prikazati tudi datum zadnje izvedene transakcije, številko računa in zadnje dnevno stanje računa v valuti. Obdobje oziroma število dni od zadnje izvršene transakcije, podamo kot parameter (P_ST_DNI). Odgovor pridobimo z naslednjim vprašanjem SQL:

```
SELECT   dan.datum AS "Datum zadnje aktivnosti",
         rac.racun_id_izvor "Račun",
         kom.ime || ' ' || kom.priimek "Komitent",
         ds.dnevno_stanje_val "Stanje", val.valuta_oznaka "Valuta"
    FROM dim_racun_trr rac,
         f_fo_stanje_dan_trr ds,
         dim_dan dan,
         dim_valuta val,
         dim_fo_komitent kom,
         rel_racun_komitent rel
   WHERE ds.racun_id = rac.racun_id
     AND ds.dan_id = dan.dan_id
     AND ds.valuta_id = val.valuta_id
     AND rac.racun_id = rel.racun_id
     AND rel.komitent_id = kom.komitent_id
     AND rel.lastnik = 'DA'
     AND rac.dat_zaprtja IS NOT NULL
     AND ds.dan_id =
            (SELECT MAX (dan_id)
               FROM f_fo_stanje_dan_trr
```

```
            WHERE racun_id = ds.racun_id AND valuta_id = ds.valuta_id)
    AND ds.dan_id < (SELECT dan_id
                     FROM dim_dan
                     WHERE datum = TRUNC (SYSDATE) - :P_ST_DNI)
ORDER BY dan.datum;
```

Lahko opazimo, da smo zgornje vprašanje SQL omejili po atributu *F_FO_STANJE_DAN-_TRR.dan_id*, tako da poiščemo dan zadnje spremembe stanja računa, pri čemer velja, da se stanje računa ni spremenilo v zadnjih P_ST_DNI dni. Zgornje vprašanje SQL smo omejili še po atributu *DIM_RACUN_TRR.dat_zaprtja*, tako da upoštevamo samo tiste račune, ki niso zaprti. Primer rezultata, ki ga vrne zgornje povpraševanje izvedeno 15.9.2008 za P_ST_DNI = 180 je:

Datum zadnje aktivnosti	***Račun***	***Komitent***	***Stanje***	***Valuta***
15.1.2007	*052001000046678*	*Marko Virant*	*-12,2*	*EUR*
15.1.2007	*052001000246514*	*Julija Lah*	*0,44*	*USD*
16.1.2007	*052001000040013*	*Tone Šorli*	*1,22*	*EUR*
17.1.2007	*052001001124114*	*Marija Čeh*	*-22,2*	*HRK*
17.1.2007	*052001000223442*	*Nika Flajs*	*32,3*	*EUR*
17.1.2007	*052001000111247*	*Diego Lakotnik*	*0*	*EUR*
18.1.2007	*052001000000880*	*Zdenka Par*	*2,1*	*EUR*
18.1.2007	*052001009748039*	*Mišo Modic*	*-332,2*	*CHF*
18.1.2007	*052001007873215*	*Miro Čas*	*-112,2*	*EUR*
19.1.2007	*052001000033111*	*Barbara Čuk*	*332,3*	*EUR*
...	...	...	...	...
19.3.2008	*052001000040083*	*Nenad Cvahte*	*123,1*	*EUR*

Primer 5:

Denimo, da nas zanima skupno število novih TRR po posameznih poslovnih enotah in produktih. Podatke želimo imeti razdeljene po posameznih mesecih leta, ki ga podamo kot parameter. Odgovor pridobimo z naslednjim vprašanjem SQL:

```
SELECT   dan.mesec_opis "Mesec", dan.leto_opis "Leto",
         pe.pe_opis "Poslovna enota", prod.produkt_opis "Produkt",
         COUNT (rac.racun_id) "Št. novih računov"
    FROM dim_dan dan,
         dim_racun rac,
         dim_produkt prod,
         dim_poslovna_enota pe
   WHERE rac.produkt_id = prod.produkt_id
     AND rac.zadnja_pe_id = pe.pe_id
     AND dan.datum = rac.dat_odprtja
     AND dan.leto_opis = :P_LETO_OPIS
GROUP BY dan.mesec_opis,
         dan.mesec_v_letu,
         dan.leto_opis,
         pe.pe_opis,
         prod.produkt_opis
```

ORDER BY pe.pe_opis, dan.mesec_v_letu;

Lahko opazimo, da smo zgornje vprašanje SQL omejili po atributu *DIM_DAN.leto_opis* s podanim parametrom P_LETO_OPIS in po atributu *DIM_RACUN.dat_odprtja*, ki zahteva da je bil račun odprt v letu P_LETO_OPIS. Hkrati uporabljamo grupiranje rezultata po atributih *DIM_DAN.mesec_opis*, *DIM_DAN.mesec_v_letu* in *DIM_DAN.leto_opis*, *DIM_POSLOVNA-_ENOTA.pe_opis* in *DIM_PRODUKT.produkt_opis*. V SELECT delu vprašanja SQL uporabimo funkcijo *COUNT*, ki sešteje vse račune glede na atribute grupiranja. Primer rezultata, ki ga vrne zgornje povpraševanje je:

Mesec	*Leto*	*Poslovna enota*	*Produkt*	*Št. novih računov*
Januar	*2007*	*PE Ljubljana 1*	*Zlati TRR*	*5*
Februar	*2007*	*PE Ljubljana 1*	*Zlati TRR*	*3*
Marec	*2007*	*PE Ljubljana 1*	*Zlati TRR*	*4*
...	*...*	*...*	*...*	*...*
Januar	*2007*	*PE Koper 1*	*Študentski TRR*	*7*
Februar	*2007*	*PE Koper 1*	*Študentski TRR*	*15*
Marec	*2007*	*PE Koper 1*	*Študentski TRR*	*9*
...	*...*	*...*	*...*	*...*
Oktober	*2007*	*PE Maribor 3*	*Gotovinski kredit*	*12*
November	*2007*	*PE Maribor 3*	*Gotovinski kredit*	*11*
December	*2007*	*PE Maribor 3*	*Gotovinski kredit*	*15*

Ker vse podatke pridobivamo iz dimenzijskega modela, ki predstavlja globalen pogled na poslovanje, bomo pridobili podatke za vse produkte, ki jih banka ponuja. Pridobljene podatke lahko uporabimo za ugotavljanje trenda rasti oziroma opada števila novih računov glede na produkte in poslovne enote. Podobno bi sestavili vprašanje SQL, ki poda število saldiranih računov glede na poslovne enote in produkte v izbranem obdobju.

6. Sklepne ugotovitve

V tem delu smo predstavili postopek izdelave prototipa podatkovnega skladišča za spremljanje poslovanja s transakcijskimi računi občanov. Pri izdelavi smo uporabili Kimballovo metodologijo razvoja podatkovnih skladišč. Metodologija temelji na iterativnem razvoju posameznih področnih podatkovnih skladišč, ki so povezana preko medsebojno skladnih skupnih dimenzij in merljivih dejstev. Gradnja podatkovnega skladišča je izredno obsežna naloga. Zato smo se pri delu omejili na izdelavo dimenzijskih podatkovnih modelov, s katerimi predstavimo poslovanje s transakcijskimi računi in sistema ETL, ki skrbi za avtomatično osveževanje podatkovnega skladišča. Implementacija podatkovnega skladišča v realnem poslovnem okolju zahteva veliko bolj natančno definicijo uporabniških zahtev, podrobno analizo izvornih sistemov, zagotavljanje ustrezne strojne opreme, kot tudi razvoj uporabniških aplikacij. S temi področji implementacije podatkovnega skladišča se v tem diplomskem delu večinoma nismo ukvarjali. Pri načrtovanju dimenzijskih podatkovnih modelov smo se srečali s problemom predstavitve bančnega poslovanja glede na dejstvi, da banke ponujajo veliko število heterogenih produktov in da imajo bogato transakcijsko zgodovino. Problem rešimo z izdelavo treh dimenzijskih modelov, ki temeljijo na konceptih heterogenih produktnih shem, transakcijskih shem in periodičnih posnetkov stanj. Tako načrtovani dimenzijski modeli omogočajo analiziranje poslovanja s transakcijskimi računi na različnih nivojih podrobnosti. Pri gradnji ročno kodiranega sistema ETL smo se v večji meri ukvarjali z izdelavo postopkov ETL, avtomatizacijo njihovega izvajanja ter rokovanjem z napakami, ki se pojavljajo ob izvajanju postopkov ETL. Težave, s katerimi se srečamo pri izdelavi postopkov ETL, se predvsem nanašajo na obremenitev izvornih sistemov ob zajemu izvornih podatkov, uveljavljanju kakovosti podatkov, ter hitrosti polnjenja podatkov v dimenzijsko ogrodje. Obremenitvi izvornih sistemov se izognemo tako, da vedno zajamemo samo spremenjene oziroma nove izvorne podatke, medtem ko kakovost podatkov zagotavljamo z izvajanjem kontrol in transformacij izvornih podatkov. Pri polnjenju podatkov v podatkovno skladišče se odločimo za tehniko polnjenja posameznega zapisa (angl. row at a time loading), ki je počasnejša od tehnike enkratnega polnjenja vseh zapisov (angl. bulk loading), vendar omogoča večjo fleksibilnost samega polnjenja.

Prototip podatkovnega skladišča, ki smo ga realizirali, je možno uporabiti na različne načine in v različne namene. Nekaj možnih primerov uporabe podatkovnega skladišča smo podali tudi v samem diplomskem delu. Med razvojem podatkovnega skladišča smo pridobili ideje za nadaljnji razvoj in nadgradnjo podatkovnega skladišča. Smiselno je, da obstoječe dimenzije oplemenitimo z dodatnimi atributi, kot tudi to, da v tabele dejstev dodamo več merljivih dejstev. S tem dodamo vrednost obstoječemu modelu, saj pridobimo boljši pogled na samo poslovanje. Nadaljnji razvoj vidimo predvsem v tem, da omogočimo analiziranje poslovanja z ostalimi bančnimi produkti. Razvoj zahteva izdelavo novih dimenzijskih modelov, ki podpirajo analizo transakcijskih aktivnosti posameznih produktov, kot tudi analizo poslovanja z bančnimi produkti na višjem nivoju. Za nove dimenzijske modele je potrebno izdelati tudi ustrezne postopke ETL, z izvajanjem katerih zagotovimo osveževanje podatkov. Podatkovno skladišče lahko nadgradimo tudi tako, da omogočimo povratno nalaganje podatkov. S tem zagotovimo, da se kakovostni podatki podatkovnega skladišča zapišejo nazaj v izvorne sisteme. Nenazadnje, možne razširitve vidimo tudi v predstavitvenem delu podatkovnega skladišča, ki smo ga v tem delu nekoliko zanemarili. Namreč, lahko razvijemo celo vrsto aplikacij, poročil ali modelov, ki uporabljajo podatke podatkovnega skladišča.

7. Priloge

7.1 Definicije dimenzijskih tabel

- Datumska dimenzija **DIM_DAN.**

Atribut	Pod. tip	Opis	Vzorčna vrednost
DAN_ID	NUMBER	Šifra dana v PS(sekvenca)	96
DATUM	DATE	Datum (dd.mm.yyyy)	05.04.2008
DATUM_OPIS	VARCHAR2(20)	Datum opisan s besedami	5. April, 2008
DAN_OPIS	VARCHAR2(10)	Naziv dneva	Sobota
MESEC_OPIS	VARCHAR2(20)	Naziv meseca	April
KVARTAL_OPIS	VARCHAR2(20)	Opis kvartala	April - Junij
LETO_OPIS	VARCHAR2(4)	Leto v formatu yyyy	2008
DAN_V_TEDNU	NUMBER	Zap. Št. Dneva v tednu	6
DAN_V_MESECU	NUMBER	Zap. Št. Dneva v mesecu	5
DAN_V_LETU	NUMBER	Zap. Št. Dneva v letu	96
ZADNJI_DAN_TEDNA	VARCHAR2(2)	Zadnji dan v tednu?	NE
ZADNJI_DAN_MESECA	VARCHAR2(2)	Zadnji dan v mesecu?	NE
DELOVNI_DAN	VARCHAR2(2)	Delovni dan?	NE
PRAZNIK	VARCHAR2(2)	Praznik?	NE
VIKEND	VARCHAR2(2)	Vikend?	DA
TEDEN_V_MESECU	NUMBER	Zap. Št. Tedna v mesecu	1
TEDEN_V_LETU	NUMBER	Zap. Št. Tedna v letu	14
MESEC_V_LETU	NUMBER	Zap. Št. meseca v letu	4
MESEC_KON_DATUM	DATE	Končni datum meseca(mm.dd.yyyy)	30.04.2008
MESEC_ST_DEL_DNI	NUMBER	Št. Delovnih dni v mesecu	22
MESEC_ST_KOL_DNI	NUMBER	Št. Koledarskih dni v mesecu	30
KVARTAL_V_LETU	NUMBER	Zap. Št. Kvartala v letu	2
KVARTAL_ZAC_DATUM	DATE	Zač. Datum kvartala(dd.mm.yyyy)	01.04.2008
KVARTAL_KON_DATUM	DATE	Končni datum kvartala(dd.mm.yyyy)	30.06.2008
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007
DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija komitentov(fizične osebe) **DIM_FO_KOMITENT**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
KOMITENT_ID	NUMBER	Šifra komitenta v PS	10252
KOMITENT_ID_IZVOR	NUMBER	Izvorna šifra komitenta	8001
IME	VARCHAR2(40)	Ime komitenta	Jošt
PRIIMEK	VARCHAR2(40)	Priimek komitenta	Novak
TITULA	VARCHAR2(10)	Titula komitenta	Gospod
SPOL	VARCHAR2(1)	Spol komitenta	M
EMSO	VARCHAR2(13)	Enotna matična številka osebe	1705952253158

DAVCNA_ST	VARCHAR2(8)	Davčna številka	18765589
DAT_ROJSTVA	DATE	Datum rojstva	17.05.1952
DAT_SMRTI	DATE	Datum smrti	Null
KRAJ_ROJSTVA	VARCHAR2(40)	Kraj rojstva	Zgornji Duplek
DRZ_ROJSTVA	VARCHAR2(50)	Država rojstva	Slovenija
DRZAVLJANSTVO	VARCHAR2(50)	Državljanstvo	Slovensko
REZIDENT	VARCHAR2(2)	Rezident?	DA
POKLIC_ID	NUMBER	Šifra poklica(izvorna)	12
POKLIC_OPIS	VARCHAR2(35)	Dolgi opis poklica	Diplomirani inženir računalništva
IZOBRAZBA_ID	NUMBER	Šifra izobrazbe(izvorna)	7
IZOB_STOPNJA	VARCHAR2(3)	Stopnja izobrazbe	7/1
IZOBRAZBA_OPIS	VARCHAR2(35)	Dolgi opis izobrazbe	Univerzitetna izobrazba
DELOVNO_MESTO	VARCHAR2(35)	Delovno mesto	Razvojni inženir
TELEFON	VARCHAR2(20)	Telefon	00386 1 458 44 22
FAX	VARCHAR2(20)	Fax	Null
GSM1	VARCHAR2(20)	Primarna GSM številka	00386 31 552 269
GSM2	VARCHAR2(20)	Dodatna GSM številka	Null
EMAIL	VARCHAR2(50)	Elektronski naslov	jost.novak@gmail.com
ST_PREB_ULICA	VARCHAR2(40)	Stalno prebivališče ulica	Dunajska ulica
ST_PREB_HISNA_ST	VARCHAR2(5)	Stalno prebivališče hišna številka	30
ST_PREB_KRAJ	VARCHAR2(30)	Stalno prebivališče kraj	Ljubljana
ST_PREB_OBCINA	VARCHAR2(30)	Stalno prebivališče občina	Ljubljana
ST_PREB_PTT_ID	VARCHAR2(5)	Stalno prebivališče pošta	1000
ST_PREB_DRZAVA	VARCHAR2(50)	Stalno prebivališče država	Slovenija
ZAC_PREB_ULICA	VARCHAR2(40)	Začasno prebivališče ulica	Dunajska ulica
ZAC_PREB_HISNA_ST	VARCHAR2(5)	Začasno prebivališče hišna številka	30
ZAC_PREB_KRAJ	VARCHAR2(30)	Začasno prebivališče kraj	Ljubljana
ZAC_PREB_OBCINA	VARCHAR2(30)	Začasno prebivališče občina	Ljubljana
ZAC_PREB_PTT_ID	VARCHAR2(5)	Začasno prebivališče pošta	1000
ZAC_PREB_DRZAVA	VARCHAR2(50)	Začasno prebivališče država	Slovenija
IZPISEK_ULICA	VARCHAR2(40)	Izpisek ulica	Dunajska ulica
IZPISEK_HISNA_ST	VARCHAR2(5)	Izpisek hišna številka	30
IZPISEK_KRAJ	VARCHAR2(30)	Izpisek kraj	Ljubljana
IZPISEK_OBCINA	VARCHAR2(30)	Izpisek občina	Ljubljana
IZPISEK_PTT_ID	VARCHAR2(5)	Izpisek pošta	1000
IZPISEK_DRZAVA	VARCHAR2(50)	Izpisek država	Slovenija
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007
DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija kartic TRR **DIM_KARTICA_TRR**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
KARTICA_ID	NUMBER	Šifra kartice v PS	121233
KARTICA_ID_IZVOR	VARCHAR2(19)	Izvorna šifra kartice	6762365057063227667
RACUN_ID	NUMBER	Šifra računa v PS	13100

KOMITENT_ID	NUMBER	Šifra komitenta v PS	358
KART_DAT_IZDAJE	DATE	Datum izdaje kartice(dd.mm.yyyy)	22.08.2006
KART_DAT_VELJAVNOSTI	DATE	Datum konca veljavnosti kartice(dd.mm.yyyy)	22.08.2010
TIP_KART_IND	VARCHAR2(1)	Indikator tipa kartice	S
TIP_KART_OPIS	VARCHAR2(20)	Besedni opis tipa kartice	Študentska ŠTR kartica
KART_STATUS_IND	VARCHAR2(1)	Indikator statusa kartice	B
KART_STATUS_OPIS	VARCHAR2(8)	Besedni opis statusa kartice	Blokirana
DAT_BLOKADE	DATE	Datum blokade kartice	13.06.2008
VZROK_BLOKADE	VARCHAR2(30)	Besedni opis vzroka blokade kartice	Trikrat napačno vnešena pin koda na bankomatu.
DAT_KRE	VARCHAR2(40)	Datum kreiranja zapisa	31.12.2007
DAT_SPR	VARCHAR2(40)	Datum spremembe zapisa	05.01.2008
UPO_KRE	DATE	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	DATE	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija lokacij **DIM_LOKACIJA**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
LOKACIJA_ID	NUMBER	Šifra lokacije v PS	114
LOKACIJA_ID_IZVOR	NUMBER	Izvorna šifra lokacije	11
LOKACIJA_OPIS	VARCHAR2(50)	Besedni opis lokacije	Spletna banka
LOKACIJA_IND	VARCHAR2(3)	Indikator lokacije	SB
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007
DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija poslovnih enot **DIM_POSLOVNA_ENOTA**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
PE_ID	NUMBER	Šifra poslovne enote v PS	7
PE_ID_IZVOR	VARCHAR2(4)	Izvorna šifra poslovne enote	7
PE_OPIS	VARCHAR2(50)	Ime poslovne enote	PE Ljubljana – Center2
PE_VODJA	VARCHAR2(50)	Vodja poslovne enote	Adam Krajcar
PE_NASLOV	VARCHAR2(50)	Naslov poslovne enote	Slovenska 32
PE_KRAJ	VARCHAR2(20)	Kraj poslovne enote	Ljubljana
PE_POSTA	VARCHAR2(4)	Pošta poslovne enote	1000
PE_REGIJA	VARCHAR2(35)	Regija poslovne enote	Notranjska
PE_DRZAVA	VARCHAR2(50)	Država poslovne enote	Slovenija
PE_DELOVNI_CAS	VARCHAR2(40)	Delovni čas	8:00 – 13:00; 15:00 – 17:00
PE_DEL_CAS_SOB	VARCHAR2(40)	Delovni čas sobota	8:00 – 12:00
PE_TELEFON	VARCHAR2(20)	Telefon	00386 1 527 44 20
PE_FAX	VARCHAR2(20)	Fax	00386 1 527 44 23
PE_STATUS_IND	VARCHAR2(1)	Indikator statusa aktivnosti PE	A
PE_STATUS_OPIS	VARCHAR2(20)	Besedni opis statusa	Aktivna
PE_DAT_VELJ_OD	DATE	Datum veljavnosti OD	05.01.2007
PE_DAT_VELJ_DO	DATE	Datum veljavnosti DO	Null

DAT_KRE	VARCHAR2(40)	Datum kreiranja zapisa	31.12.2007
DAT_SPR	VARCHAR2(40)	Datum spremembe zapisa	05.01.2008
UPO_KRE	DATE	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	DATE	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija produktov **DIM_PRODUKT**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
PRODUKT_ID	NUMBER	Šifra produkta v PS	3
PRODUKT_ID_IZVOR	VARCHAR2(10)	Izvorna šifra produkta	STRR
PRODUKT_OPIS	VARCHAR2(50)	Opis produkta	Študentski transakcijski račun
PRODUKT_OZNAKA	VARCHAR2(10)	Oznaka produkta	STRR
PROD_VELJ_OD	DATE	Datum začetka veljavnosti produkta(dd.mm.yyyy)	01.02.2000
PROD_VELJ_DO	DATE	Datum konca veljavnosti produkta(dd.mm.yyyy)	Null
PROD_STATUS_IND	VARCHAR2(1)	Indikator statusa aktivnosti produkta	A
PROD_STATUS_OPIS	VARCHAR2(10)	Besedni opis statusa produkta	Aktiven
ROCNOST_IND	VARCHAR2(1)	Indikator ročnosti	D
ROCNOST_OPIS	VARCHAR2(15)	Opis ročnosti	Dolgoročen
SKUPINA_PROD_ID	VARCHAR2(5)	Šifra skupine produktov(izvor)	TRR
SKUPINA_PROD_OPIS	VARCHAR2(75)	Besednio opis skupine produktov	Transakcijski računi
PODROCJE_ID	VARCHAR2(5)	Šifra področja(izvor)	FIZOS
PODROCJE_OPIS	VARCHAR2(75)	Besedni opis področja	Poslovanje s fizičnimi osebami
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007
DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija TRR produktov **DIM_PRODUKT_TRR**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
PRODUKT_ID	NUMBER	Šifra produkta v PS	3
PRODUKT_ID_IZVOR	VARCHAR2(10)	Izvorna šifra produkta	STRR
PRODUKT_OPIS	VARCHAR2(50)	Opis produkta	Študentski transakcijski račun
PRODUKT_OZNAKA	VARCHAR2(10)	Oznaka produkta	STRR
PROD_VELJ_OD	DATE	Datum začetka veljavnosti produkta(dd.mm.yyyy)	01.02.2000
PROD_VELJ_DO	DATE	Datum konca veljavnosti produkta(dd.mm.yyyy)	Null
PROD_STATUS_IND	VARCHAR2(1)	Indikator statusa aktivnosti produkta	A
PROD_STATUS_OPIS	VARCHAR2(10)	Besedni opis statusa produkta	Aktiven
ROCNOST_IND	VARCHAR2(1)	Indikator ročnosti	D
ROCNOST_OPIS	VARCHAR2(15)	Opis ročnosti	Dolgoročen
SKUPINA_PROD_ID	VARCHAR2(5)	Šifra skupine produktov(izvor)	TRR
SKUPINA_PROD_OPIS	VARCHAR2(75)	Besedni opis skupine produktov	Transakcijski računi
PODROCJE_ID	VARCHAR2(5)	Šifra področja(izvor)	FIZOS
PODROCJE_OPIS	VARCHAR2(75)	Besedni opis področja	Poslovanje s

			fizičnimi osebami
STROSKI_VODENJA	NUMBER	Stroški vodenja TRR (v EUR)	0
OM_NEG_STANJA	NUMBER	Obresta mera negativnega stanja	7,95%
OM_POZ_STANJA	NUMBER	Obrestna mera pozitivnega stanja	0,2%
OM_NEDOV_STANJA	NUMBER	Obrestna mera nedovoljenega stanja	10,2%
AVT_LIMIT_IND	VARCHAR2(1)	Indikator pridobitve avtomatskega limita	D
ZNESEK_AVT_LIMIT	NUMBER	Znesek avtomatsko odobrenega limita(v EUR)	200
IZREDNI_LIMIT_IND	VARCHAR2(1)	Indikator pridobitve izrednega limita	D
NEZGOD_ZAV_IND	VARCHAR2(1)	Indikator nezgodnega zavarovanja	D
MES_ST_IZPISKOV	NUMBER	Število mesečnih izpiskov	1
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007
DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija računov **DIM_RACUN**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
RACUN_ID	NUMBER	Šifra računa v PS	122501
RACUN_ID_IZVOR	VARCHAR2(20)	Izvorna šifra računa	020102770265489
PRODUKT_ID	NUMBER	Šifra produkta računa	STRR
DAT_ODPRTJA	DATE	Datum odprtja računa	15.08.2006
DAT_ZAPRTJA	DATE	Datum zaprtja računa	Null
DAT_AKTIVAC	DATE	Datum aktivacije računa	16.08.2006
ZADNJI_STATUS_ID	NUMBER	Šifra zadnjega statusa računa	1
BONITETA_IND	VARCHAR2(1)	Indikator bonitete računa	A
BONITETA_OPIS	VARCHAR2(20)	Besedni opis bonitete računa	Pozitivno 3M stanje
MAT_PE_ID	NUMBER	Šifra matične PE kjer je račun bil odprt	7
ZADNJA_PE_ID	NUMBER	Šifra zadnje PE kjer se račun trenutno vodi	7
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007
DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija računov TRR **DIM_RACUN_TRR**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
RACUN_ID	NUMBER	Šifra računa v PS	122501
RACUN_ID_IZVOR	VARCHAR2(20)	Izvorna šifra računa	020102770265489
PRODUKT_ID	NUMBER	Šifra produkta računa	STRR
DAT_ODPRTJA	DATE	Datum odprtja računa	15.08.2006
DAT_ZAPRTJA	DATE	Datum zaprtja računa	Null
DAT_AKTIVAC	DATE	Datum aktivacije računa	16.08.2006
ZADNJI_STATUS_ID	NUMBER	Šifra zadnjega statusa računa	1
BONITETA_IND	VARCHAR2(1)	Indikator bonitete računa	A
BONITETA_OPIS	VARCHAR2(20)	Besedni opis bonitete računa	Pozitivno 3M stanje
MAT_PE_ID	NUMBER	Šifra matične PE kjer je račun bil odprt	7

ZADNJA_PE_ID	NUMBER	Šifra zadnje PE kjer se račun trenutno vodi	7
LIMIT_ZNESEK	NUMBER	Znesek odobrenega limita	300
LIMIT_DAT_VELJ_OD	DATE	Datum začetka veljavnosti limita	15.01.2008
LIMIT_DAT_VELJ_DO	DATE	Datum konca veljavnosti limita	15.01.2009
LIMIT_DAT_ODOBR	DATE	Datum odobritve limita	14.01.2008
LIMIT_DAT_PREKLIC	DATE	Datum preklica limita	Null
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007
DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija statusov **DIM_STATUS**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
STATUS_ID	NUMBER	Šifra statusa v PS	1
STATUS_ID_IZVOR	NUMBER	Izvorna šifra statusa	1
STATUS_OPIS	VARCHAR2(40)	Besedni opis statusa	Aktiven
STATUS_IND	VARCHAR2(1)	Indikator statusa	A
DAT_KRE	VARCHAR2(40)	Datum kreiranja zapisa	31.12.2007
DAT_SPR	VARCHAR2(40)	Datum spremembe zapisa	05.01.2008
UPO_KRE	DATE	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	DATE	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija transakcij **DIM_TRANSAKCIJA**.

Atribut	Pod. tip	Opis	Vzorčna vrednost
TRANS_ID	NUMBER	Šifra transakcije v PS	18
TRANS_ID_IZVOR	NUMBER	Izvorna šifra transakcije	18
TIP_TRANS_IND	VARCHAR2(1)	Indikator tipa transakcije: P(riliv) ali O(dliv)	P
TIP_TRANS	VARCHAR2(10)	Tip transakcije	Priliv
TRANS_OPIS	VARCHAR2(50)	Besedni opis transakcije	Polog gotovine
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007
DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

- Dimenzija valut **DIM_VALUTA.**

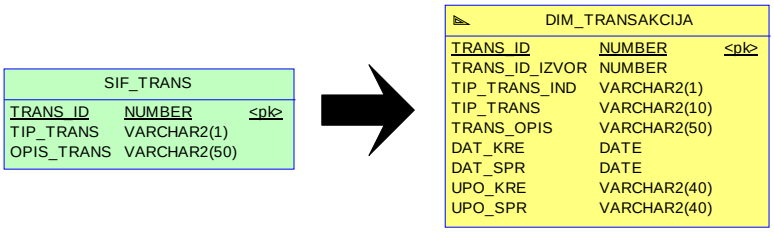
Atribut	Pod. tip	Opis	Vzorčna vrednost
VALUTA_ID	NUMBER	Šifra valute v PS	5
VALUTA_ID_IZVOR	VARCHAR2(3)	Izvorna šifra valute	HRK
VALUTA_OZNAKA	VARCHAR2(3)	Mednarodna oznaka valute	HRK
VALUTA_OPIS	VARCHAR2(35)	Ime valute v slovenskem jeziku	Hrvaška kuna
VALUTA_ORIG_IME	VARCHAR2(35)	Ime valute v tujem jeziku	Hrvatska kuna
VALUTA_ENOTA	NUMBER	Enota valute	1
VALUTA_DRZAVA	VARCHAR2(50)	Država valute	Hrvaška
VAL_VELJ_OD	DATE	Datum začetka veljavnosti valute	15.1.1995
VAL_VELJ_DO	DATE	Datum konca veljavnosti valute	Null
VAL_STATUS_IND	VARCHAR2(1)	Indikator statusa veljavnosti valute	A
VAL_STATUS_OPIS	VARCHAR2(10)	Opis statusa veljavnosti valute	Aktivna valuta
DAT_KRE	DATE	Datum kreiranja zapisa	31.12.2007

DAT_SPR	DATE	Datum spremembe zapisa	05.01.2008
UPO_KRE	VARCHAR2(40)	Uporabnik kreiranja zapisa	LOAD_DIM
UPO_SPR	VARCHAR2(40)	Uporabnik spremembe zapisa	LOAD_DIM

7.2 Preslikave izvornih atributov

- ***Preslikava za polnjenje dimenzije DIM_TRANSAKCIJA:***

Tok podatkov ter strukturi izvorne in ciljne tabele:

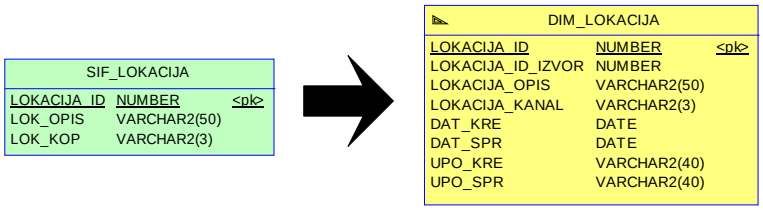


Preslikava izvornih atributov v atribute dimenzije DIM_TRANSAKCIJA:

Izvor *(SIF_TRANS ST)*	Cilj *(DIM_TRANSAKCIJA DT)*	Opombe
-	DT.TRANS_ID	Sekvenca SEQ_DIM_TRANS.NEXTVAL
ST.TRANS_ID	DT.TRANS_ID_IZVOR	Where ST_TRANS_ID = DT.TRANS_ID_IZVOR
ST.TIP_TRANS	DT.TIP_TRANS_IND	
-	DT.TIP_TRANS	Če ST.TIP_TRANS='P' potem PRILIV, sicer ODLIV
ST.OPIS_TRANS	DT.TRANS_OPIS	
-	DT.DAT_KRE	SYSDATE ob kreiranju zapisa
-	DT.DAT_SPR	SYSDATE ob spremembi zapisa
-	DT.UPO_KRE	USER ob kreiranju zapisa
-	DT.UPO_SPR	USER ob spremembi zapisa

- ***Preslikava za polnjenje dimenzije DIM_LOKACIJA:***

Tok podatkov ter strukturi izvorne in ciljne tabele:



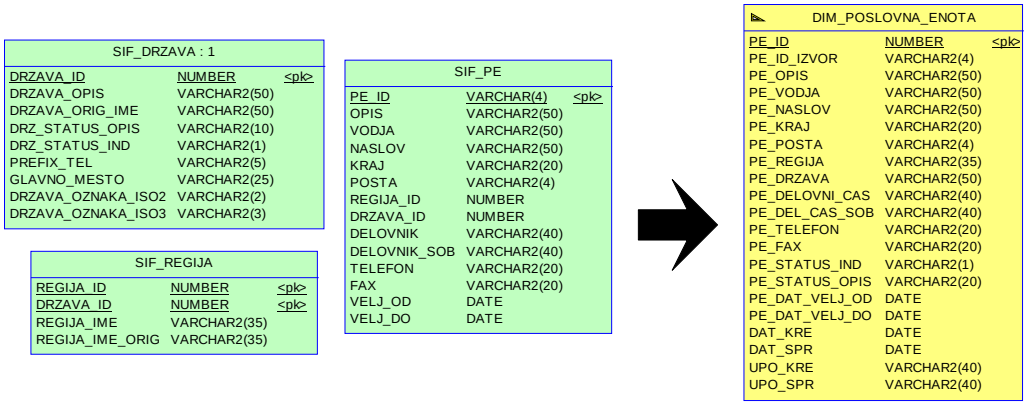
Preslikava izvornih atributov v atribute dimenzije DIM_LOKACIJA:

Izvor *(SIF_LOKACIJA SL)*	Cilj *(DIM_LOKACIJA DL)*	Opombe
-	DL.LOKACIJA_ID	Sekvenca SEQ_DIM_LOKACIJA.NEXTVAL
SL.LOKACIJA_ID	DL.LOKACIJA_ID_IZVOR	Where SL.LOKACIJA_ID = DL.LOKACIJA_ID_IZVOR
SL.LOK_OPIS	DL.LOKACIJA_OPIS	

SL.LOK_KOP	DL.LOKACIJA_IND	
-	DL.DAT_KRE	SYSDATE ob kreiranju zapisa
-	DL.DAT_SPR	SYSDATE ob spremembi zapisa
-	DL.UPO_KRE	USER ob kreiranju zapisa
-	DL.UPO_SPR	USER ob spremembi zapisa

- ***Preslikava za polnjenje dimenzije DIM_POSLOVNA_ENOTA:***

Tok podatkov ter strukture izvornih tabel in ciljne tabele:



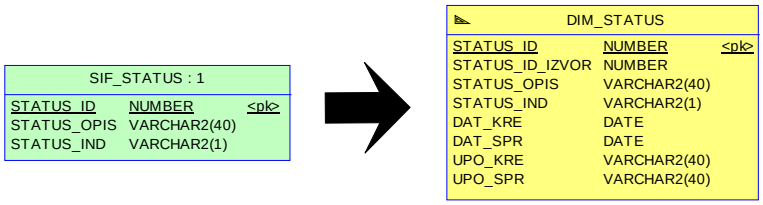
Preslikava izvornih atributov v atribute dimenzije DIM_POSLOVNA_ENOTA:

Izvor *(SIF_PE SP, SIF_REGIJA SR, SIF_DRZAVA SD)*	**Cilj *(DIM_POSLOVNA_ENOTA DP)***	**Opombe**
-	DP.PE_ID	Sekvenca SEQ_DIM_PE.NEXTVAL
SP.PE_ID	DP.PE_ID_IZVOR	
SP.OPIS	DP.PE_OPIS	
SP.VODJA	DP.PE_VODJA	
SP. NASLOV	DP.PE_NASLOV	
SP. KRAJ	DP.PE_KRAJ	
SP. POSTA	DP.PE_POSTA	
SR.REGIJA_IME	DP.PE_REGIJA	Select SR.REGIJA_IME from SIF_REGIJA SR where SR.regija_id = REC.regija_id and SR.drzava_id = REC.drzava_id;
SD.DRZAVA_OPIS	DP.PE_DRZAVA	Select SD.drzava_opis from SIF_DRZAVA SD where SD.drzava_id = REC.drzava_id;
SP.DELOVNIK	DP.PE_DELOVNI_CAS	
SP.DELOVNIK_SOB	DP.PE_DEL_CAS_SOB	
SP.TELEFON	DP.PE_TELEFON	
SP.FAX	DP.PE_FAX	
-	DP.PE_STATUS_IND	IF SP.VELJ_DO IS NULL THEN ‘A’ ELSE ‘N’;
-	DP.PE_STATUS_OPIS	IF SP.VELJ_DO IS NULL THEN ‘Aktivna’ ELSE ‘Neaktivna’;
SP.VELJ_OD	DP.PE_DAT_VELJ_OD	
SP.VELJ_DO	DP.PE_DAT_VELJ_DO	
-	DP.UPO_SPR	SYSDATE ob kreiranju zapisa
-	DP.UPO_KRE	SYSDATE ob spremembi zapisa
-	DP.DAT_SPR	USER ob kreiranju zapisa

-	DP.DAT_KRE	USER ob spremembi zapisa

- ***Preslikava za polnjenje dimenzije DIM_STATUS:***

Tok podatkov ter strukturi izvorne in ciljne tabele:

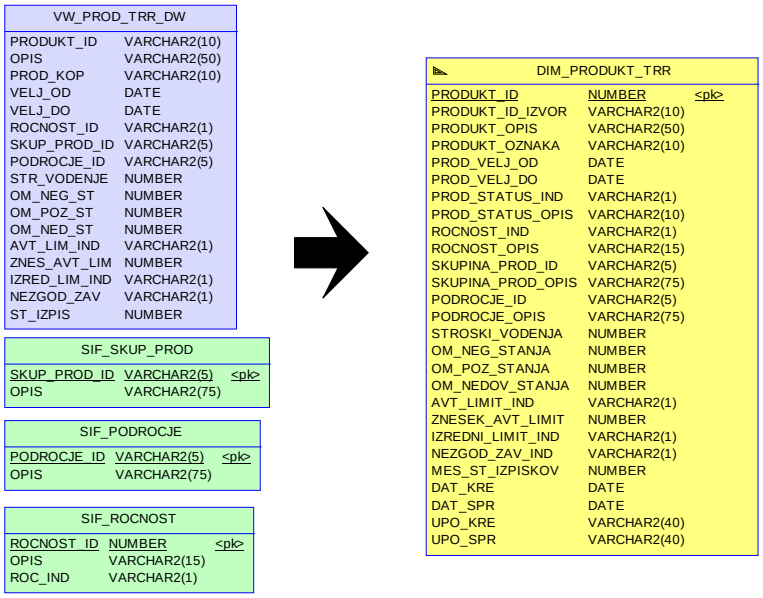


Preslikava izvornih atributov v atribute dimenzije DIM_STATUS:

Izvor ***(SIF_STATUS SS)***	**Cilj** ***(DIM_STATUS DS)***	**Opombe**
-	DS.STATUS_ID	Sekvenca SEQ_DIM_STATUS.NEXTVAL
SL.STATUS_ID	DS.STATUS_ID_IZVOR	Where SS.STATUS_ID = DS.STATUS_ID_IZVOR
SL.STATUS_OPIS	DS.STATUS_OPIS	
SL.STATUS_IND	DS.STATUS_IND	
-	DS.DAT_KRE	SYSDATE ob kreiranju zapisa
-	DS.DAT_SPR	SYSDATE ob spremembi zapisa
-	DS.UPO_KRE	USER ob kreiranju zapisa
-	DS.UPO_SPR	USER ob spremembi zapisa

- ***Preslikava za polnjenje dimenzije DIM_PRODUKT_TRR in DIM_PRODUKT:***

Tok podatkov ter strukture izvornih tabel in ciljne tabele:



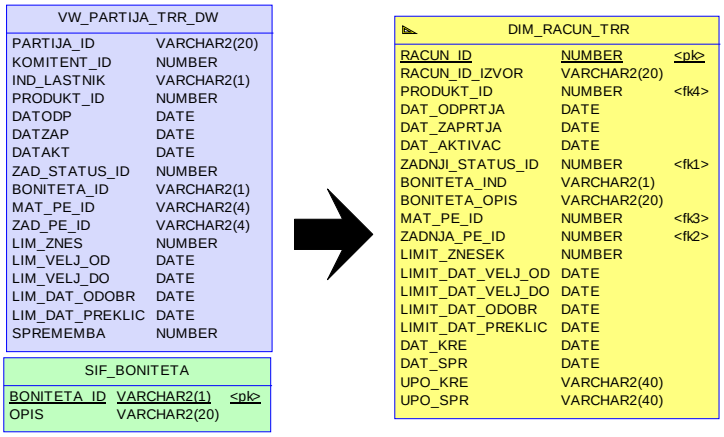
Preslikava izvornih atributov v atribute dimenzije DIM_PRODUKT_TRR:

Izvor *(VW_PROD_TRR_DW SPR, SIF_ROCNOST SR, SIF_PODROCJE SPO, SIF_SKUP_PROD SS)*	**Cilj** *(DIM_PRODUKT_TRR DP)*	**Opombe**
-	DP.PRODUKT_ID	Sekvenca SEQ_DIM_PRODUKT.NEXTVAL
SPR.PRODUKT_ID	DP.PRODUKT_ID_IZVOR	
SPR.OPIS	DP.PRODUKT_OPIS	
SPR.PROD_KOP	DP.PRODUKT_OZNAKA	
SPR.VELJ_OD	DP.PROD_VELJ_OD	
SPR.VELJ_DO	DP.PROD_VELJ_DO	
-	DP.PROD_STATUS_IND	IF SPR.VELJ_DO IS NULL THEN 'A' ELSE 'N';
-	DP.PROD_STATUS_OPIS	IF SPR.VELJ_DO IS NULL THEN 'Aktiven' ELSE 'Neaktiven';
SR.ROC_IND	DP.ROCNOST_IND	Select SR.roc_ind from SIF_ROCNOST SR where SR.rocnost_id = REC.rocnost_id;
SR.OPIS	DP.ROCNOST_OPIS	Select SR.opis from SIF_ROCNOST SR where SR.rocnost_id = REC.rocnost_id;
SPR.SKUP_PROD_ID	DP.SKUPINA_PROD_ID	
SS.OPIS	DP.SKUPINA_PROD_OPIS	Select SS.opis from SIF_SKUP_PROD SS where SS.skup_prod_id = REC.skup_prod_id;
SPR_PODROCJE_ID	DP.PODROCJE_ID	
SPO.OPIS	DP.PODROCJE_OPIS	Select SPO.opis from SIF_PODROCJE SPO where SS.podrocje_id = REC. podrocje_id;
SPR.STR_VODENJE	DP.STROSKI_VODENJA	
SPR.OM_NEG_ST	DP.OM_NEG_STANJA	
SPR.OM_POZ_ST	DP.OM_POZ_STANJA	
SPR.OM_NED_ST	DP.OM_NEDOV_STANJA	
SPR.AVT_LIM_IND	DP.AVT_LIMIT_IND	
SPR.ZNES_AVT_LIM	DP.ZNESEK_AVT_LIMIT	
SPR.IZRED_LIM_IND	DP.IZREDNI_LIMIT_IND	
SPR.NEZGOD_ZAV	DP.NEZGOD_ZAV_IND	
SPR.ST_IZPIS	DP.MES_ST_IZPISKOV	
-	DP.DAT_KRE	SYSDATE ob kreiranju zapisa
-	DP.DAT_SPR	SYSDATE ob spremembi zapisa
-	DP.UPO_KRE	USER ob kreiranju zapisa
-	DP.UPO_SPR	USER ob spremembi zapisa

OPOMBA: Dimenzijo **DIM_PRODUKT** polnimo na osnovi iste preslikave.

- ***Preslikava za polnjenje dimenziji DIM_RACUN_TRR in DIM_RACUN:***

Tok podatkov ter strukture izvornih tabel in ciljne tabele:



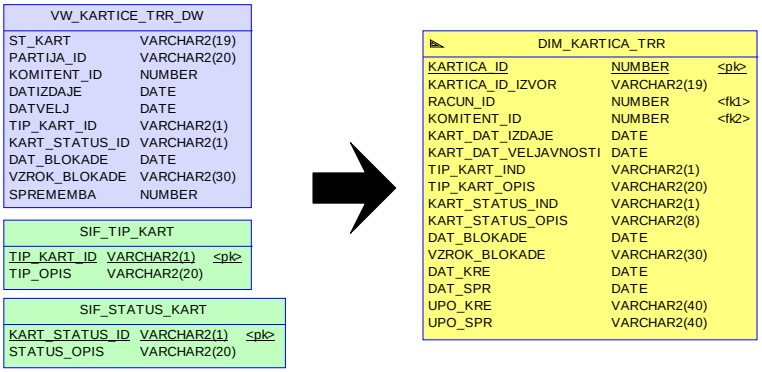
Preslikava izvornih atributov v atribute dimenzije DIM_RACUN_TRR, pri čemer REC označuje trenutno obdelovani zapis iz VW_PARTIJA_TRR_DW:

Izvor ***(VW_PARTIJA_TRR_DW VP, SIF_BONITETA SB)***	**Cilj** ***(DIM_RACUN_TRR DR)***	**Opombe**
-	DR.RACUN_ID	Sekvenca SEQ_DIM_RACUN.NEXTVAL
VP.PARTIJA_ID	DR.RACUN_ID_IZVOR	
-	DR.PRODUKT_ID	Select DP.produkt_id from DIM_PRODUKT_TRR DP where DP.produkt_id_izvor = REC.produkt_id;
VP.DATODP	DR.DAT_ODPRTJA	
VP.DATZAP	DR.DAT_ZAPRTJA	
VP.DATAKT	DR.DAT_AKTIVAC	
-	DR.ZADNJI_STATUS_ID	Select DS.status_id from DIM_STATUS DS where DS.status_id_izvor = REC.zad_status_id;
SB.BONITETA_ID	DR.BONITETA_IND	Select SB.boniteta_id from SIF_BONITETA SB where SB.boniteta_id = REC.boniteta_id;
SB.OPIS	DR.BONITETA_OPIS	Select SB.opis from SIF_BONITETA SB where SB.boniteta_id = REC.boniteta_id;
-	DR.MAT_PE_ID	Select DPE.pe_id from DIM_POSLOVNA_ENOTA DPE where DPE.pe_id_izvor = REC.mat_pe_id;
-	DR.ZADNJA_PE_ID	Select DPE.pe_id from DIM_POSLOVNA_ENOTA DPE where DPE.pe_id_izvor = REC.zad_pe_id;
VP.LIM_ZNES	LIMIT_ZNESEK	
VP.LIM_VELJ_OD	LIMIT_DAT_VELJ_OD	
VP.LIM_VELJ_DO	LIMIT_DAT_VELJ_DO	
VP.LIM_DAT_ODOBR	LIMIT_DAT_ODOBR	
VP.LIM_DAT_PREKLIC	LIMIT_DAT_PREKLIC	
-	DR.DAT_KRE	SYSDATE ob kreiranju zapisa
-	DR.DAT_SPR	SYSDATE ob spremembi zapisa
-	DR.UPO_KRE	USER ob kreiranju zapisa
-	DR.UPO_SPR	USER ob spremembi zapisa

OPOMBA: Dimenzijo **DIM_RACUN** polnimo na osnovi iste preslikave, pri čemer seveda ne polnimo podatkov o limitu računa.

- ***Preslikava za polnjenje dimenzije DIM_KARTICA_TRR:***

Tok podatkov ter strukture izvornih tabel in ciljne tabele:

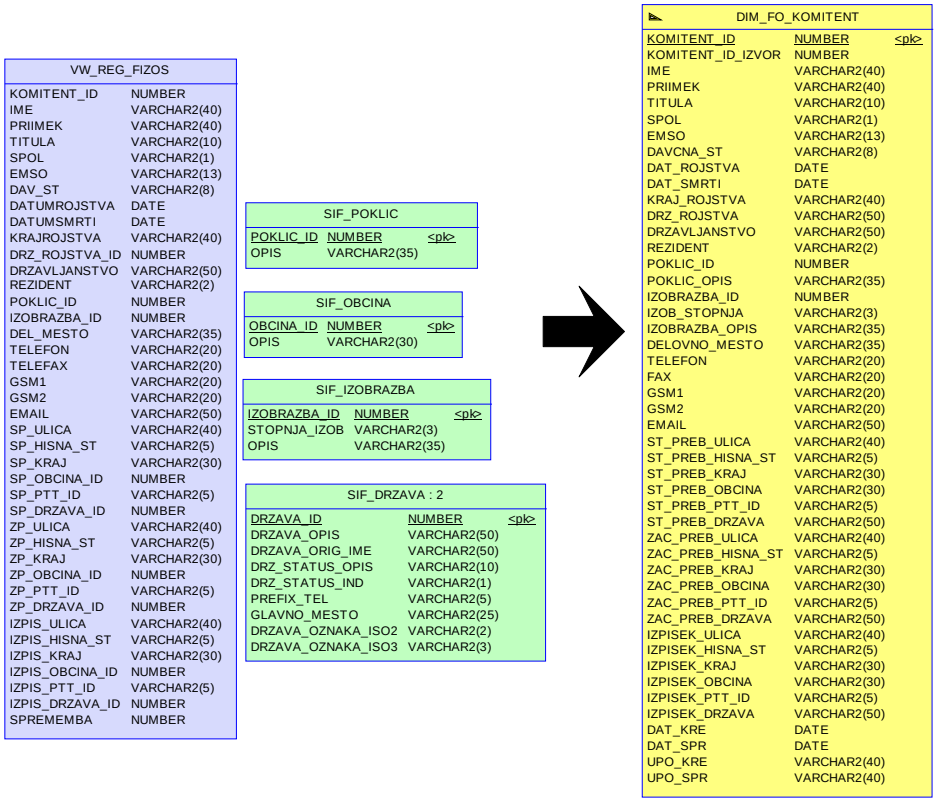


Preslikava izvornih atributov v atribute dimenzije DIM_KARTICA_TRR, pri čemer REC označuje trenutno obdelovani zapis iz VW_KARTICE_TRR_DW:

Izvor ***(VW_KARTICE_TRR_DW VK, SIF_TIP_KART ST, SIF_STATUS_KART SK)***	**Cilj** ***(DIM_KARTICA_TRR DK)***	**Opombe**
-	DK.KARTICA_ID	Sekvenca SEQ_DIM_KART_TRR.NEXTVAL
VK.ST_KART	DK.KARTICA_ID_IZVOR	
-	DK.RACUN_ID	Select DR.racun_id from DIM_RACUN_TRR DR where DR.racun_id_izvor = REC.partija_id;
-	DK.KOMITENT_ID	Select DK.komitent_id from DIM_FO_KOMITENT DK where DK.komitent_id_izvor = REC.komitent_id;
VK.DATIZDAJE	DK.KART_DAT_IZDAJE	
VK.DATVELJ	DK.KART_DAT_VELJAVNOSTI	
ST.TIP_KART_ID	DK.TIP_KART_IND	Select ST.tip_kart_id from SIF_TIP_KART ST where ST.tip_kart_id = REC.tip_kart_id;
ST.TIP_OPIS	DK.TIP_KART_OPIS	Select ST.tip_opis from SIF_TIP_KART ST where ST.tip_kart_id = REC.tip_kart_id;
SK.KART_STATUS_ID	DK.KART_STATUS_IND	Select SK.kart_status_id from SIF_STATUS_KART SK where SK.kart_status_id = REC.kart_status_id;
SK.STATUS_OPIS	DK.KART_STATUS_OPIS	Select SK.status_opis from SIF_STATUS_KART SK where SK.kart_status_id = REC.kart_status_id;
VK.DAT_BLOKADE	DK.DAT_BLOKADE	
VK.VZROK_BLOKADE	DK.VZROK_BLOKADE	
-	DK.DAT_KRE	SYSDATE ob kreiranju zapisa
-	DK.DAT_SPR	SYSDATE ob spremembi zapisa
-	DK.UPO_KRE	USER ob kreiranju zapisa
-	DK.UPO_SPR	USER ob spremembi zapisa

- ***Preslikava za polnjenje dimenzije DIM_FO_KOMITENT:***

Tok podatkov ter strukture izvornih tabel in ciljne tabele:



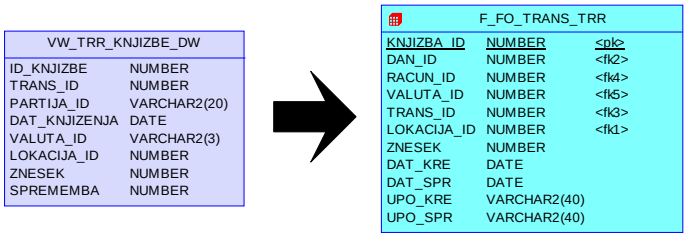
Preslikava izvornih atributov v atribute dimenzije DIM_FO_KOMITENT, pri čemer REC označuje trenutno obdelovani zapis iz VW_REG_FIZOS:

Izvor ***(VW_REG_FIZOS RF, SIF_POKLIC SP, SIF_OBCINA SO, SIF_IZOBRAZBA SI, SIF_DRZAVA SD)***	**Cilj** ***(DIM_FO_KOMITENT DK)***	**Opombe**
-	KOMITENT_ID	Sekvenca SEQ_DIM_KOMITENT.NEXTVAL
RF.KOMITENT_ID	KOMITENT_ID_IZVOR	
RF.IME	IME	
RF.PRIIMEK	PRIIMEK	
RF.TITULA	TITULA	
RF.SPOL	SPOL	
RF.EMSO	EMSO	
RF.DAV_ST	DAVCNA_ST	
RF.DATUMROJSTVA	DAT_ROJSTVA	
RF.DATUMSMRTI	DAT_SMRTI	
RF.KRAJROJSTVA	KRAJ_ROJSTVA	
SD.DRZAVA_OPIS	DRZ_ROJSTVA	Select SD.drzava_opis from SIF_DRZAVA SD where SD.drzava_id = REC.drz_rojstva_id;
RF.DRZAVLJANSTVO	DRZAVLJANSTVO	
RF.REZIDENT	REZIDENT	
SP.POKLIC_ID	POKLIC_ID	Select SP.poklic_i from SIF_POKLIC SP where

		SP.poklic_id = REC.poklic_id;
SP.OPIS	POKLIC_OPIS	Select SP.opis from SIF_POKLIC SP where SP.poklic_id = REC.poklic_id;
SI.IZOBRAZBA_ID	IZOBRAZBA_ID	Select SI.izobrazba_id from SIF_IZOBRAZBA SI where SI.izobrazba_id=REC.izobrazba_id;
SI.STOPNJA_IZOB	IZOB_STOPNJA	Select SI.stopnja_izob from SIF_IZOBRAZBA SI where SI.izobrazba_id=REC.izobrazba_id;
SI.OPIS	IZOBRAZBA_OPIS	Select SI.opis from SIF_IZOBRAZBA SI where SI.izobrazba_id=REC.izobrazba_id;
RF.DEL_MESTO	DELOVNO_MESTO	
RF.TELEFON	TELEFON	
RF.TELEFAX	FAX	
RF.GSM1	GSM1	
RF.GSM2	GSM2	
RF.EMAIL	EMAIL	
RF.SP_ULICA	ST_PREB_ULICA	
RF.SP_HISNA_ST	ST_PREB_HISNA_ST	
RF.SP_KRAJ	ST_PREB_KRAJ	
SO.OPIS	ST_PREB_OBCINA	Select SO.opis from SIF_OBCINA SO where SO.obcina_id = REC.sp_obcina_id;
RF.SP_PTT_ID	ST_PREB_PTT_ID	
SD.DRZAVA_OPIS	ST_PREB_DRZAVA	Select SD.drzava_opis from SIF_DRZAVA SD where SD.drzava_id = REC.sp_drzava_id;
RF.ZP_ULICA	ZAC_PREB_ULICA	
RF.ZP_HISNA_ST	ZAC_PREB_HISNA_ST	
RF.ZP_KRAJ	ZAC_PREB_KRAJ	
SO.OPIS	ZAC_PREB_OBCINA	Select SO.opis from SIF_OBCINA SO where SO.obcina_id = REC.zp_obcina_id;
RF.ZP_PTT_ID	ZAC_PREB_PTT_ID	
SD.DRZAVA_OPIS	ZAC_PREB_DRZAVA	Select SD.drzava_opis from SIF_DRZAVA SD where SD.drzava_id = REC.zp_drzava_id;
RF.IZPIS_ULICA	IZPISEK_ULICA	
RF.IZPIS_HISNA_ST	IZPISEK_HISNA_ST	
RF.IZPIS_KRAJ	IZPISEK_KRAJ	
SO.OPIS	IZPISEK_OBCINA	Select SO.opis from SIF_OBCINA SO where SO.obcina_id = REC.izpis_obcina_id;
RF.IZPIS_PTT_ID	IZPISEK_PTT_ID	
SD.DRZAVA_OPIS	IZPISEK_DRZAVA	Select SD.drzava_opis from SIF_DRZAVA SD where SD.drzava_id = REC.izpis_drzava_id;
-	DAT_KRE	SYSDATE ob kreiranju zapisa
-	DAT_SPR	SYSDATE ob spremembi zapisa
-	UPO_KRE	USER ob kreiranju zapisa
-	UPO_SPR	USER ob spremembi zapisa

- ***Preslikava za polnjenje tabele dejstev F_FO_TRANS_TRR:***

Tok podatkov ter strukturi izvorne in ciljne tabele:

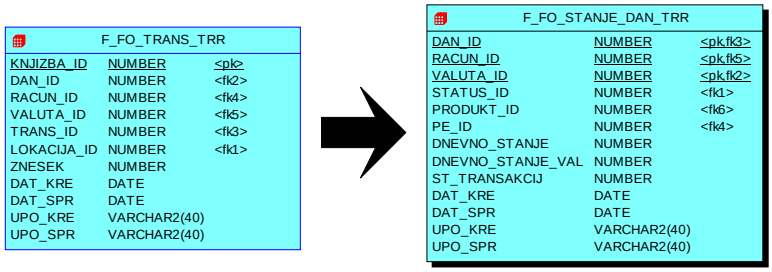


Preslikava izvornih atributov v atribute tabele dejstev F_FO_TRANS_TRR, pri čemer REC označuje trenutno obdelovani zapis iz VW_TRR_KNJIZBE_DW:

Izvor *(VW_TRR_KNJIZBE_DW TK)*	**Cilj *(F_FO_TRANS_TRR FT)***	**Opombe**
TK.ID_KNJIZBE	FT.KNJIZBA_ID	
TK.DAT_KNJIZENJA	FT.DAN_ID	Select DD.dan_id from DIM_DAN DD where DD.datum = REC.dat_knjizenja;
-	FT.RACUN_ID	Select DR.racun_id from DIM_RACUN_TRR DR where DR.racun_id_izvor = REC.partija_id;
-	FT.VALUTA_ID	Select DV.valuta_id from DIM_VALUTA DV where DV.valuta_id_izvor = REC.valuta_id;
-	FT.TRANS_ID	Select DT.trans_id from DIM_TRANSAKCIJA DT where DT.trans_id_izvor = REC.trans_id;
-	FT.LOKACIJA_ID	Select DL.lokacija_id from DIM_LOKACIJA DL where DL.lokacija_id_izvor = REC.lokacija_id;
TK.ZNESEK	FT.ZNESEK	
-	FT.DAT_KRE	SYSDATE ob kreiranju zapisa
-	FT.DAT_SPR	SYSDATE ob spremembi zapisa
-	FT.UPO_KRE	USER ob kreiranju zapisa
-	FT.UPO_SPR	USER ob spremembi zapisa

- ***Preslikava za polnjenje tabele dejstev F_FO_STANJE_DAN_TRR:***

Tok podatkov ter strukturi izvorne in ciljne tabele:



REC označuje trenutno obdelovani zapis iz F_FO_TRANS_TRR.

Izvor *(F_FO_TRANS_TRR TT)*	**Cilj *(F_FO_STANJE_DAN_TRR FT)***	**Opombe**
TT.DAN_ID	FT.DAN_ID	
TT.RACUN_ID	FT.RACUN_ID	
TT.VALUTA_ID	FT.VALUTA_ID	
-	FT.STATUS_ID	Select DR.zadnji_status_id from

		DIM_RACUN_TRR DR where DR.racun_id = REC.racun_id;
-	FT.PRODUKT_ID	Select DR.produkt_id from DIM_RACUN_TRR DR where DR.racun_id = REC.racun_id;
-	FT.PE_ID	Select DP.zadnja_pe_id from DIM_RACUN_TRR DP where DP.racun_id = REC.racun_id;
	FT.DNEVNO_STANJE	Zadnjemu stanju FT.dnevno_stanje prištejemo REC.ZNESEK, glede na TIP_TRANS(priliv je +, odliv je -). Če je VALUTA_OZNAKA != 'EUR', preračunamo znesek v EUR po tečaju veljavnem na DAN_ID. TIP_TRANS pridobimo iz DIM_TRANSAKCIJA glede na REC.TRANS_ID, VALUTA_OZNAKA dobimo iz DIM_VALUTA glede na REC.VALUTA_ID;
	FT.DNEVNO_STANJE_VAL	Zadnjemu stanju FT.dnevno_stanje_val prištejemo REC.ZNESEK, glede na TIP_TRANS(priliv je +, odliv je -). Prištevanje se izvede za valuto REC.VALUTA_ID in preračun valut ni potreben.
-	FT.ST_TRANSAKCIJ	Št. transakcij povečamo za 1 ob insertu
-	FT.DAT_KRE	SYSDATE ob kreiranju zapisa
-	FT.DAT_SPR	SYSDATE ob spremembi zapisa
-	FT.UPO_KRE	USER ob kreiranju zapisa
-	FT.UPO_SPR	USER ob spremembi zapisa

OPOMBA: Tabelo dejstev **F_FO_STANJE_DAN** polnimo na osnovi iste preslikave, pri čemer ne polnimo podatkov o številu transakcij (atribut ST_TRANSAKCIJ).

7.3 Programski paket PKG_EROR_UTIL

Koda je pisana v programskem jeziku Oracle PL/SQL.

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE DW_LOAD.pkg_error_util
AS
/******************************************************************************
   NAME:       PKG_ERROR_UTIL
   PURPOSE:    Uporablja se za belezenje napak polnjenja.

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        27.8.2008   Mladen Babic      1. Created this package.
******************************************************************************/
   g_status_izpad    VARCHAR2 (10) := 'IZPAD';
   g_status_uspeh    VARCHAR2 (10) := 'USPEŠNO';
   g_status_napake   VARCHAR2 (10) := 'NAPAKE';

   PROCEDURE get_oracle_napaka (p_status IN OUT VARCHAR2, p_err_msg IN OUT VARCHAR2);

   FUNCTION get_opis_napake_id (p_opis_napake polnjenje_napaka_opis.opis_napake%TYPE)
      RETURN NUMBER;

   FUNCTION zapisi_napako (
      p_tabela         VARCHAR2,
```

```
      p_polnjenje_id   NUMBER,
      p_program        VARCHAR2,
      p_opis_napake    VARCHAR2,
      p_atribut        VARCHAR2
   )
      RETURN NUMBER;
END pkg_error_util;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_load.pkg_error_util
AS
/******************************************************************************
   NAME:       PKG_ERROR_UTIL
   PURPOSE:    Uporablja se za belezenje napak polnjenja.

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        27.8.2008   Mladen Babic      1. Created this package.
******************************************************************************/

   /**
     Vrača status in Oracle opis napake.
   */
   PROCEDURE get_oracle_napaka (p_status IN OUT VARCHAR2, p_err_msg IN OUT VARCHAR2)
   IS
   BEGIN
      p_status     := g_status_izpad;
      p_err_msg    := p_err_msg || SQLERRM;
   END;

   /**
   Funkcija vrača polnjenje_napaka_opis.opis_nap_id za podani opis napake.
   Če opis napake ne obstaja, funkcija doda zapis v tabelo opisov napak(polnjenje_napaka_opis)
   in vrne opis_nap_id novega zapisa.
   */
   FUNCTION get_opis_napake_id (p_opis_napake polnjenje_napaka_opis.opis_napake%TYPE)
      RETURN NUMBER
   IS
      l_opis_nap_id   NUMBER;
   BEGIN
      BEGIN
         SELECT opis_nap_id
           INTO l_opis_nap_id
           FROM polnjenje_napaka_opis
          WHERE opis_napake = UPPER (p_opis_napake);
      EXCEPTION
         WHEN NO_DATA_FOUND
         THEN
            SELECT seq_opis_nap_id.NEXTVAL
              INTO l_opis_nap_id
              FROM DUAL;

            INSERT INTO polnjenje_napaka_opis
                        (opis_nap_id, opis_napake
                        )
                 VALUES (l_opis_nap_id, UPPER (p_opis_napake)
                        );
      END;

      RETURN l_opis_nap_id;
   END;

   /**
   Funkcija zapise napako v tabelo napak polnjenja polnjenje_napaka in vrne NAPAKA_ID.
   */
   FUNCTION zapisi_napako (
      p_tabela         VARCHAR2,
      p_polnjenje_id   NUMBER,
      p_program        VARCHAR2,
      p_opis_napake    VARCHAR2,
      p_atribut        VARCHAR2
   )
      RETURN NUMBER
```

```
   IS
      l_napaka_id     NUMBER;
      l_opis_nap_id   NUMBER;
   BEGIN
      SELECT seq_napaka_id.NEXTVAL
        INTO l_napaka_id
        FROM DUAL;

      l_opis_nap_id    := get_opis_napake_id (p_opis_napake);

      INSERT INTO polnjenje_napaka
                  (napaka_id, opis_nap_id, polnjenje_id, tabela, polje,
                   program
                  )
           VALUES (l_napaka_id, l_opis_nap_id, p_polnjenje_id, p_tabela, UPPER (p_atribut),
                   p_program
                  );

      RETURN (l_napaka_id);
   END;
END pkg_error_util;
/
```

7.4 Postopki ETL

Vsa koda je pisana v programskem jeziku Oracle PL/SQL.

7.4.1 Programski paket PKG_DIM_DAN

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE dw_stage.pkg_dim_dan
AS
/******************************************************************************
   NAME:       PKG_DIM_DAN
   PURPOSE:    Polnjenje dimenzije DIM_DAN.

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        14.08.2008  mladen babic     1. Created this package.
******************************************************************************/
   PROCEDURE load_dim_dan (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );
END pkg_dim_dan;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_stage.pkg_dim_dan
AS
/******************************************************************************
   NAME:       PKG_DIM_DAN
   PURPOSE:

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        14.08.2008             1. Created this package.
******************************************************************************/
   PROCEDURE load_dim_dan (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
```

```
)
IS
   l_start_date   DATE;
   l_end_date     DATE;
   l_st_vrstic    NUMBER := 0;
BEGIN
   -- DOLOČANJE začetnega datuma polnjenja
   BEGIN
      SELECT datum + 1
        INTO l start_date
        FROM dim_dan
       WHERE dan_id = (SELECT MAX (dan_id)
                         FROM dim_dan);
   EXCEPTION
      WHEN NO_DATA_FOUND
      THEN
         l_start_date    := TO_DATE ('01.01.2008', 'dd.mm.yyyy');
   END;

   -- DOLOČANJE KONČNEGA DATUMA POLNJENJA
   l_end_date    := ADD_MONTHS (l_start_date, 12) - 1;

   WHILE l_start_date <= l_end_date
   LOOP
      INSERT INTO dim_dan
                  (dan_id, datum, datum_opis, dan_opis, mesec_opis,
                   kvartal_opis, leto_opis, dan_v_tednu, dan_v_mesecu,
                   dan_v_letu, zadnji_dan_tedna, zadnji_dan_meseca,
                   delovni_dan, praznik, vikend, teden_v_mesecu,
                   teden_v_letu, mesec_v_letu, mesec_kon_datum,
                   mesec_st_del_dni, mesec_st_kol_dni, kvartal_v_letu,
                   kvartal_zac_datum, kvartal_kon_datum, dat_kre,
                   dat_spr, upo_kre, upo_spr)
         SELECT seq_dim_dan.NEXTVAL dan_id, l_start_date,
                   TO_CHAR (l_start_date, 'dd. ')
                || TRIM (TO_CHAR (l_start_date, 'Month'))
                || TO_CHAR (l_start_date, ', YYYY') datum_opis,
                TRIM (TO_CHAR (l_start_date, 'Day')) dan_opis,
                TRIM (TO_CHAR (l_start_date, 'Month')) mesec_opis,
                CASE
                   WHEN TRIM (TO_CHAR (l_start_date, 'Q')) =
                                                          '1'
                      THEN 'Januar - Marec'
                   WHEN TRIM (TO_CHAR (l_start_date, 'Q')) =
                                                          '2'
                      THEN 'April - Junij'
                   WHEN TRIM (TO_CHAR (l_start_date, 'Q')) =
                                                          '3'
                      THEN 'Julij - September'
                   WHEN TRIM (TO_CHAR (l_start_date, 'Q')) =
                                                          '4'
                      THEN 'Oktober - December'
                END kvartal_opis,
                TO_CHAR (l_start_date, 'YYYY') leto_opis,
                TO_NUMBER (TO_CHAR (l_start_date, 'D')) dan_v_tednu,
                TO_NUMBER (TO_CHAR (l_start_date, 'DD')) dan_v_mesecu,
                TO_NUMBER (TO_CHAR (l_start_date, 'DDD')) dan_v_letu,
                CASE
                   WHEN TO_NUMBER
                               (TO_CHAR (l_start_date, 'D')
                               ) IN (7)
                      THEN 'DA'
                   ELSE 'NE'
                END zadnji_dan_tedna,
                CASE
                   WHEN TRUNC (l_start_date) =
                          LAST_DAY
                                 (TRUNC (l_start_date)
                                 )
                      THEN 'DA'
                   ELSE 'NE'
                END zadnji_dan_meseca,
                CASE
                   WHEN TO_NUMBER (TO_CHAR (l_start_date, 'D')
                                  ) IN (6, 7)
                      THEN 'NE'
```

```
                  ELSE 'DA'
               END delovni_dan,
               CASE
                  WHEN TO_CHAR (l_start_date, 'dd.mm') IN
                         ('01.01', '02.01', '08.02', '27.04',
                          '01.05', '02.05', '25.06', '15.08',
                          '31.10', '01.11', '25.12', '26.12')
                     THEN 'DA'
                  ELSE 'NE'
               END praznik,
               CASE
                  WHEN TO_NUMBER (TO_CHAR (l_start_date, 'D')) IN
                                                       (6, 7)
                     THEN 'DA'
                  ELSE 'NE'
               END vikend,
               TO_NUMBER (TO_CHAR (l_start_date, 'W')) teden_v_mesecu,
               TO_NUMBER (TO_CHAR (l_start_date, 'WW')) teden_v_letu,
               TO_NUMBER (TO_CHAR (l_start_date, 'mm')) mesec_v_letu,
               LAST_DAY (l_start_date) mesec_kon_datum,
               NULL mesec_st_del_dni,
               TO_NUMBER
                    (TO_CHAR (LAST_DAY (l_start_date), 'dd')
                    ) mesec_st_kol_dni,
               TO_NUMBER (TO_CHAR (l_start_date, 'Q')) kvartal_v_letu,
               CASE
                  WHEN TRIM (TO_CHAR (l_start_date, 'Q')
                             ) = '1'
                     THEN TO_DATE ('1.1.2008', 'dd.mm.yyyy')
                  WHEN TRIM (TO_CHAR (l_start_date, 'Q')) = '2'
                     THEN TO_DATE ('1.4.2008', 'dd.mm.yyyy')
                  WHEN TRIM (TO_CHAR (l_start_date, 'Q')) = '3'
                     THEN TO_DATE ('1.7.2008', 'dd.mm.yyyy')
                  WHEN TRIM (TO_CHAR (l_start_date, 'Q')) = '4'
                     THEN TO_DATE ('1.10.2008', 'dd.mm.yyyy')
               END kvartal_zac_datum,
               CASE
                  WHEN TRIM (TO_CHAR (l_start_date, 'Q')
                             ) = '1'
                     THEN TO_DATE ('31.3.2008', 'dd.mm.yyyy')
                  WHEN TRIM (TO_CHAR (l_start_date, 'Q')) = '2'
                     THEN TO_DATE ('30.6.2008', 'dd.mm.yyyy')
                  WHEN TRIM (TO_CHAR (l_start_date, 'Q')) = '3'
                     THEN TO_DATE ('30.9.2008', 'dd.mm.yyyy')
                  WHEN TRIM (TO_CHAR (l_start_date, 'Q')) = '4'
                     THEN TO_DATE ('31.12.2008', 'dd.mm.yyyy')
               END kvartal_kon_datum,
               SYSDATE dat_kre, NULL dat_spr, USER upo_kre,
               NULL upo_spr
          FROM DUAL;

      l_start_date    := l_start_date + 1;
      l_st_vrstic     := l_st_vrstic + 1;
   END LOOP;

   UPDATE dim_dan a
      SET mesec_st_del_dni =
             (SELECT   COUNT (*)
                  FROM dim_dan
                 WHERE vikend = 'NE'
                   AND praznik = 'NE'
                   AND TO_CHAR (datum, 'mm.yyyy') =
                                          TO_CHAR (a.datum, 'mm.yyyy')
              GROUP BY TO_CHAR (a.datum, 'mm.yyyy'));

   p_status      := pkg_error_util.g_status_uspeh;
   p_message     := pkg_load_util.st_obdelanih_vrstic_opis (l_st_vrstic);
EXCEPTION
   WHEN OTHERS
   THEN
      pkg_error_util.get_oracle_napaka (p_status, p_message);
END;
END pkg_dim_dan;
/
```

7.4.2 Programski paket PKG_DIM_VALUTA

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE dw_stage.pkg_dim_valuta
AS
   /******************************************************************************
      name:     pkg_dim_valuta
      purpose:  Insert\Update podatkov tabele dim_valuta

      revisions:
      ver        date        author           description
      ---------  ----------  ---------------  ------------------------------------
      1.0        15.8.2008    mladen babic    1. created this package.
   ******************************************************************************/

   /*
     Insert\Update podatkov tabele dim_valuta
   */
   PROCEDURE load_dim_valuta (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );
END pkg_dim_valuta;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_stage.pkg_dim_valuta
AS
   PROCEDURE load_dim_valuta (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   )
   IS
   BEGIN
      MERGE INTO dim_valuta a
         USING (SELECT *
                  FROM sif_valuta) b
         ON (a.valuta_id_izvor = b.valuta_id)
         WHEN MATCHED THEN
            UPDATE
               SET a.valuta_oznaka = b.valuta_kop,
                   a.valuta_opis = b.valuta_opis,
                   a.valuta_orig_ime = b.valuta_izv_ime,
                   a.valuta_enota = b.valuta_enota,
                   a.valuta_drzava = b.valuta_ime_drz,
                   a.val_velj_od = b.valuta_veljod,
                   a.val_velj_do = b.valuta_veljdo,
                   a.val_status_ind =
                                 DECODE (b.valuta veljdo,
                                         NULL, 'A',
                                         'N'
                                        ),
                   a.val_status_opis =
                      DECODE (b.valuta_veljdo,
                              NULL, 'AKTIVNA',
                              'NEAKTIVNA'
                             )
         WHEN NOT MATCHED THEN
            INSERT (a.valuta_id, a.valuta_id_izvor, a.valuta_oznaka,
                    a.valuta_opis, a.valuta_orig_ime, a.valuta_enota,
                    a.valuta_drzava, a.val_velj_od, a.val_velj_do,
                    a.val_status_ind, a.val_status_opis)
            VALUES (seq_dim_valuta.NEXTVAL, b.valuta_id, b.valuta_kop,
                    b.valuta_opis, b.valuta_izv_ime, b.valuta_enota,
                    b.valuta_ime_drz, b.valuta_veljod, b.valuta_veljdo,
                    DECODE (b.valuta_veljdo, NULL, 'A', 'N'),
                    DECODE (b.valuta_veljdo, NULL, 'AKTIVNA', 'NEAKTIVNA'));
      p_message    :=
```

```
                    pkg_load_util.st_obdelanih_vrstic_opis (SQL%ROWCOUNT);
      p_status     := pkg_error_util.g_status_uspeh;
   EXCEPTION
      WHEN OTHERS
      THEN
         pkg_error_util.get_oracle_napaka (p_status, p_message);
   END load_dim_valuta;
END pkg_dim_valuta;
/
```

7.4.3 Programski paket PKG_DIM_FO_KOMITENT

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE dw_stage.pkg_dim_fo_komitent
AS
/******************************************************************************
   NAME:       PKG_DIM_FO_KOMITENT
   PURPOSE:    Polnjenje dimenzije komitentov(fizične osebe) iz izvora TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        26.8.2008   Mladen Babic     1. Created this package.
******************************************************************************/
   PROCEDURE load_temp_fo_komitent (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );

   PROCEDURE load_dim_fo_komitent (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );
END pkg_dim_fo_komitent;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_stage.pkg_dim_fo_komitent
AS
/******************************************************************************
   NAME:       PKG_DIM_FO_KOMITENT
   PURPOSE:    Polnjenje dimenzije komitentov(fizične osebe) iz izvora TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        26.8.2008   Mladen Babic     1. Created this package.
******************************************************************************/

   /**
   Urejanje spola komitenta.
   */
   FUNCTION edit_spol (p_spol dim_fo_komitent.spol%TYPE)
      RETURN dim_fo_komitent.spol%TYPE
   IS
      l_spol   dim_fo_komitent.spol%TYPE   := UPPER (p_spol);
      retval   dim_fo_komitent.spol%TYPE;
   BEGIN
      IF l_spol = 'M'
      THEN
         retval    := 'M';
      ELSIF l_spol IN ('Ž', 'Z')
      THEN
         retval    := 'Ž';
      ELSE
         retval    := 'N';
      END IF;
```

```
   RETURN retval;
END;

/**
Urejanje titule komitenta. Podano je samo vzorčno urejanje titule.
*/
FUNCTION edit_titula (p_titula dim_fo_komitent.titula%TYPE)
   RETURN dim_fo_komitent.titula%TYPE
IS
   l_titula   dim_fo_komitent.titula%TYPE   := UPPER (p_titula);
   retval     dim_fo_komitent.titula%TYPE;
BEGIN
   IF l_titula IN ('G', 'G.', 'GOSP.')
   THEN
      retval    := 'GOSPOD';
   ELSIF l_titula = 'GA.'
   THEN
      retval    := 'GOSPA';
   ELSIF l_titula IN ('DR', 'DR.')
   THEN
      retval    := 'DOKTOR';
   ELSIF l_titula IN ('PROF.', 'PROF')
   THEN
      retval    := 'PROFESOR';
   ELSE
      retval    := l_titula;
   END IF;

   RETURN retval;
END;

/**
Preverjanje emso komitenta. Mora biti dolzine 13 in sestavljen iz samih stevilk.
*/
FUNCTION check_emso (p_emso dim_fo_komitent.emso%TYPE)
   RETURN BOOLEAN
IS
   retval   BOOLEAN;
BEGIN
   IF     LENGTH (p_emso) = 13
      AND NVL (LENGTH (TRIM (TRANSLATE (p_emso, '0123456789', ' '))), 0) =
                                                                          0
   THEN
      retval    := TRUE;
   ELSE
      retval    := FALSE;
   END IF;

   RETURN retval;
END;

 /**
Preverjanje DAVCNE STEVILKE komitenta. Mora biti dolzine 8 in sestavljena iz samih stevilk.
*/
FUNCTION check_davst (p_davst dim_fo_komitent.davcna_st%TYPE)
   RETURN BOOLEAN
IS
   retval   BOOLEAN;
BEGIN
   IF     LENGTH (p_davst) = 8
      AND NVL (LENGTH (TRIM (TRANSLATE (p_davst, '0123456789', ' '))),
               0) = 0
   THEN
      retval    := TRUE;
   ELSE
      retval    := FALSE;
   END IF;

   RETURN retval;
END;

 /**
Preverjanje ali je nek string sestavljen iz samih številk.
*/
FUNCTION check_if_numeric (p_in_string VARCHAR2)
   RETURN BOOLEAN
```

```
   IS
      retval   BOOLEAN;
   BEGIN
      IF     NVL (LENGTH (TRIM (TRANSLATE (p_in_string, '0123456789', ' '))),
                  0
                 ) = 0
         AND LENGTH (p_in_string) > 0
      THEN
         retval    := TRUE;
      ELSE
         retval    := FALSE;
      END IF;

      RETURN retval;
   END;

   FUNCTION edit_ulica (p_ulica dim_fo_komitent.st_preb_ulica%TYPE)
      RETURN dim_fo_komitent.st_preb_ulica%TYPE
   IS
      l_ulica   dim_fo_komitent.st_preb_ulica%TYPE   := UPPER (p_ulica);
   BEGIN
      l_ulica    := REPLACE (l_ulica, '.', '. ');
      l_ulica    := REPLACE (l_ulica, '  ', ' ');
      l_ulica    := REPLACE (l_ulica, ',', ', ');
      l_ulica    := REPLACE (l_ulica, 'C.', 'CESTA');
      l_ulica    := REPLACE (l_ulica, 'NAS.', 'NASELJE');
      l_ulica    := REPLACE (l_ulica, 'U.', 'UL.');
      l_ulica    := REPLACE (l_ulica, 'UL.', 'ULICA');
      RETURN SUBSTR (RTRIM (l_ulica, ' '), 1, 40);
   END;

   FUNCTION edit_hisna_st (
      p_hisna_st   dim_fo_komitent.st_preb_hisna_st%TYPE
   )
      RETURN dim_fo_komitent.st_preb_hisna_st%TYPE
   IS
      l_hisna_st   dim_fo_komitent.st_preb_hisna_st%TYPE   := p_hisna_st;
   BEGIN
      l_hisna_st    := UPPER (LTRIM (l_hisna_st, '0'));
   END;

   /**
   Procedura za arhiviranje zapisa o komitentu.
   */
   PROCEDURE arh_temp_fo_komitent (p_rowid ROWID)
   IS
   BEGIN
      INSERT INTO arh_fo_komitent
                  (komitent_id, ime, priimek, titula, spol, emso, dav_st,
                   datumrojstva, datumsmrti, krajrojstva, drz_rojstva_id,
                   drzavljanstvo, rezident, poklic_id, izobrazba_id,
                   del_mesto, telefon, telefax, gsm1, gsm2, email,
                   sp_ulica, sp_hisna_st, sp_kraj, sp_obcina_id,
                   sp_ptt_id, sp_drzava_id, zp_ulica, zp_hisna_st,
                   zp_kraj, zp_obcina_id, zp_ptt_id, zp_drzava_id,
                   izpis_ulica, izpis_hisna_st, izpis_kraj,
                   izpis_obcina_id, izpis_ptt_id, izpis_drzava_id,
                   sprememba, dat_arh)
         SELECT komitent_id, ime, priimek, titula, spol, emso, dav_st,
                datumrojstva, datumsmrti, krajrojstva, drz_rojstva_id,
                drzavljanstvo, rezident, poklic_id, izobrazba_id,
                del_mesto, telefon, telefax, gsm1, gsm2, email, sp_ulica,
                sp_hisna_st, sp_kraj, sp_obcina_id, sp_ptt_id,
                sp_drzava_id, zp_ulica, zp_hisna_st, zp_kraj,
                zp_obcina_id, zp_ptt_id, zp_drzava_id, izpis_ulica,
                izpis_hisna_st, izpis_kraj, izpis_obcina_id, izpis_ptt_id,
                izpis_drzava_id, sprememba, SYSDATE
           FROM temp_fo_komitent
          WHERE ROWID = p_rowid;
   END;

   /**
   Zajem spremenjenih ali novih podatkov o komitentih iz izvora. Pri samem zajemu že naredimo
nekaj
   čiščenja podatkov s pomočjo funkcije TRIM, ki odreže presledke pri črkovnih vrednostih(pred
in za besedo).
```

```
   Pred polnjenjem izbrišemo začasno tabelo temp_fo_komitent.
   Parametri:
   p_load_id    - zaporedna številka polnjenja
   p_status     - indikator uspešnosti izvedbe koraka(USPEŠNO ali NAPAKA).
   p_message    - informacija o uspešnosti polnjenja.
   */
   PROCEDURE load_temp_fo_komitent (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   )
   IS
   BEGIN
      EXECUTE IMMEDIATE 'truncate table temp_fo_komitent';

      INSERT INTO temp_fo_komitent
                  (komitent_id, ime, priimek, titula, spol, emso, dav_st,
                   datumrojstva, datumsmrti, krajrojstva, drz_rojstva_id,
                   drzavljanstvo, rezident, poklic_id, izobrazba_id,
                   del_mesto, telefon, telefax, gsm1, gsm2, email,
                   sp_ulica, sp_hisna_st, sp_kraj, sp_obcina_id,
                   sp_ptt_id, sp_drzava_id, zp_ulica, zp_hisna_st,
                   zp_kraj, zp_obcina_id, zp_ptt_id, zp_drzava_id,
                   izpis_ulica, izpis_hisna_st, izpis_kraj,
                   izpis_obcina_id, izpis_ptt_id, izpis_drzava_id,
                   sprememba)
         SELECT komitent_id, TRIM (ime), TRIM (priimek), TRIM (titula),
                TRIM (spol), TRIM (emso), TRIM (dav_st), datumrojstva,
                datumsmrti, TRIM (krajrojstva), drz_rojstva_id,
                TRIM (drzavljanstvo), TRIM (rezident), poklic_id,
                izobrazba_id, TRIM (del_mesto), TRIM (telefon),
                TRIM (telefax), TRIM (gsm1), TRIM (gsm2), TRIM (email),
                TRIM (sp_ulica), TRIM (sp_hisna_st), TRIM (sp_kraj),
                sp_obcina_id, TRIM (sp_ptt_id), sp_drzava_id,
                TRIM (zp_ulica), TRIM (zp_hisna_st), TRIM (zp_kraj),
                zp_obcina_id, TRIM (zp_ptt_id), zp_drzava_id,
                TRIM (izpis_ulica), TRIM (izpis_hisna_st),
                TRIM (izpis_kraj), izpis_obcina_id, TRIM (izpis_ptt_id),
                izpis_drzava_id, sprememba
           FROM vw_reg_fizos izv
          WHERE NOT EXISTS (SELECT NULL
                              FROM arh_fo_komitent arh
                             WHERE izv.sprememba = arh.sprememba);

      p_message    :=
                   pkg_load_util.st_obdelanih_vrstic_opis (SQL%ROWCOUNT);
      p_status     := 'USPEŠNO';
   EXCEPTION
      WHEN OTHERS
      THEN
         pkg_error_util.get_oracle_napaka (p_status, p_message);
   END;

   /**
   Transformacija izvornih podatkov in  njihovo polnjenje v dimenzijsko tabelo
DIM_FO_KOMITENT.
   Parametri:
   p_load_id    - zaporedna številka polnjenja
   p_status     - indikator uspešnosti izvedbe koraka(USPEŠNO ali NAPAKA).
   p_message    - informacija o uspešnosti polnjenja.
   */
   PROCEDURE load_dim_fo_komitent (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   )
   IS
      skip_row                EXCEPTION;
      l_st_obdelanih_vrstic   NUMBER                          := 0;
      l_komitent_id           dim_fo_komitent.komitent_id%TYPE;
      r_komitent              dim_fo_komitent%ROWTYPE;
      b_nov_komitent          BOOLEAN;
      l_tabela                VARCHAR2 (100)         := 'TEMP_FO_KOMITENT';
      l_program               VARCHAR2 (100)     := 'LOAD_DIM_FO_KOMITENT';
      l_napaka_id             NUMBER;
   BEGIN
```

```
-- zaradi ponovitve koraka polnjenja, zbrišemo morebitne napake
UPDATE temp_fo_komitent
   SET napaka_id = NULL
 WHERE napaka_id IS NOT NULL;

-- OBDELAVA IZVORNIH PODATKOV
FOR r_rec IN (SELECT a.ROWID rid, a.*
                FROM temp_fo_komitent a)
LOOP
   BEGIN
      l_napaka_id                := NULL;

      -- UGOTAVLJANJE OBSTOJA KOMITENTA V PPS
      BEGIN
         SELECT komitent_id
           INTO l_komitent_id
           FROM dim_fo_komitent
          WHERE komitent_id_izvor = r_rec.komitent_id;

         b_nov_komitent    := FALSE;
      EXCEPTION
         WHEN NO_DATA_FOUND
         THEN
            b_nov_komitent    := TRUE;

            -- DOLOCANJE SIFRE V PPS ZA NOVEGA KOMITENTA
            SELECT seq_dim_komitent.NEXTVAL
              INTO r_komitent.komitent_id
              FROM DUAL;
      END;

      -- IME je obvezen podatek
      IF r_rec.ime IS NOT NULL
      THEN
         r_komitent.ime    := UPPER (r_rec.ime);
      ELSE
         l_napaka_id    :=
            pkg_error_util.zapisi_napako
                           (l_tabela,
                            p_polnjenje_id,
                            l_program,
                            'IME KOMITENTA JE OBVEZEN PODATEK!',
                            'IME'
                           );
         RAISE skip_row;
      END IF;

      -- PRIIMEK je obvezen podatek
      IF r_rec.priimek IS NOT NULL
      THEN
         r_komitent.priimek    := UPPER (r_rec.priimek);
      ELSE
         l_napaka_id    :=
            pkg_error_util.zapisi_napako
                          (l_tabela,
                           p_polnjenje_id,
                           l_program,
                           'PRIIMEK KOMITENTA JE OBVEZEN PODATEK!',
                           'PRIIMEK'
                          );
         RAISE skip_row;
      END IF;

      r_komitent.titula           := edit_titula (r_rec.titula);
      r_komitent.spol             := edit_spol (r_rec.spol);

      -- EMSO je obvezen podatek
      IF check_emso (r_rec.emso)
      THEN
         r_komitent.emso    := r_rec.emso;
      ELSE
         l_napaka_id    :=
            pkg_error_util.zapisi_napako
                               (l_tabela,
                                p_polnjenje_id,
                                l_program,
```

```
                              'EMSO KOMITENTA JE NEPRAVILEN!',
                              'EMSO'
                             );
   RAISE skip_row;
END IF;

-- DAVCNA_ST je obvezen podatek
IF check_davst (r_rec.dav_st)
THEN
   r_komitent.davcna_st    := r_rec.dav_st;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                       (l_tabela,
                        p_polnjenje_id,
                        l_program,
                        'DAVČNA ŠT. KOMITENTA JE NEPRAVILNA!',
                        'DAVCNA_ST'
                       );
   RAISE skip_row;
END IF;

r_komitent.dat_rojstva      := r_rec.datumrojstva;
r_komitent.dat_smrti        := r_rec.datumsmrti;
r_komitent.kraj_rojstva     := UPPER (r_rec.krajrojstva);

BEGIN
   SELECT drzava_opis
     INTO r_komitent.drz_rojstva
     FROM sif_drzava
    WHERE drzava_id = r_rec.drz_rojstva_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      r_komitent.drz_rojstva    := 'NEZNANO';
END;

r_komitent.drzavljanstvo    := r_rec.drzavljanstvo;
r_komitent.rezident         := r_rec.rezident;

BEGIN
   SELECT poklic_id, opis
     INTO r_komitent.poklic_id, r_komitent.poklic_opis
     FROM sif_poklic
    WHERE poklic_id = r_rec.poklic_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      r_komitent.poklic_id      := -1;
      r_komitent.poklic_opis    := 'NEZNANO';
END;

BEGIN
   SELECT izobrazba_id, stopnja_izob,
          opis
     INTO r_komitent.izobrazba_id, r_komitent.izob_stopnja,
          r_komitent.izobrazba_opis
     FROM sif_izobrazba
    WHERE izobrazba_id = r_rec.izobrazba_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      r_komitent.izobrazba_id      := -1;
      r_komitent.izob_stopnja      := 'NEZNANO';
      r_komitent.izobrazba_opis    := 'NEZNANO';
END;

r_komitent.delovno_mesto    := UPPER (r_rec.del_mesto);

-- TELEFON je obvezen podatek
IF check_if_numeric (r_rec.telefon)
THEN
   r_komitent.telefon    := r_rec.telefon;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
```

```
                  (l_tabela,
                   p_polnjenje_id,
                   l_program,
                   'TELEFONSKA ŠT. KOMITENTA JE NEPRAVILNA!',
                   'TELEFON'
                  );
   RAISE skip_row;
END IF;

IF check_if_numeric (r_rec.telefax)
THEN
   r_komitent.fax    := r_rec.telefax;
ELSE
   r_komitent.fax    := NULL;
END IF;

-- GSM1 je obvezen podatek
IF check_if_numeric (r_rec.gsm1)
THEN
   r_komitent.gsm1    := r_rec.gsm1;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                        (l_tabela,
                         p_polnjenje_id,
                         l_program,
                         'GSM1 ŠT. KOMITENTA JE NEPRAVILNA!',
                         'GSM1'
                        );
   RAISE skip_row;
END IF;

IF check_if_numeric (r_rec.gsm2)
THEN
   r_komitent.gsm2    := r_rec.gsm2;
END IF;

r_komitent.email            := UPPER (r_rec.email);

-- Ulica stalnega prebivališča je obvezen podatek
IF r_rec.sp_ulica IS NOT NULL
THEN
   r_komitent.st_preb_ulica    := edit_ulica (r_rec.sp_ulica);
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
         (l_tabela,
          p_polnjenje_id,
          l_program,
          'ULICA STALNEGA PREBIVALIŠČA KOMITENTA JE OBVEZEN PODATEK!',
          'ST_PREB_ULICA'
         );
   RAISE skip_row;
END IF;

-- Hišna številka stalnega prebivališča je obvezen podatek
IF check_if_numeric (r_rec.sp_hisna_st)
THEN
   r_komitent.st_preb_hisna_st    :=
                             edit_hisna_st (r_rec.sp_hisna_st);
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
         (l_tabela,
          p_polnjenje_id,
          l_program,
          'HIŠNA ŠT. STALNEGA PREBIVALIŠČA KOMITENTA JE NEPRAVILNA!',
          'ST_PREB_HISNA_ST'
         );
   RAISE skip_row;
END IF;

-- Kraj stalnega prebivališča je obvezen podatek
IF r_rec.sp_kraj IS NOT NULL
THEN
   r_komitent.st_preb_kraj    := UPPER (r_rec.sp_kraj);
```

```
            ELSE
               l_napaka_id    :=
                  pkg_error_util.zapisi_napako
                       (l_tabela,
                        p_polnjenje_id,
                        l_program,
                        'KRAJ STALNEGA PREBIVALIŠČA KOMITENTA JE PRAZEN!',
                        'ST_PREB_KRAJ'
                       );
               RAISE skip_row;
            END IF;

            BEGIN
               SELECT opis
                 INTO r_komitent.st_preb_obcina
                 FROM sif_obcina
                WHERE obcina_id = r_rec.sp_obcina_id;
            EXCEPTION
               WHEN NO_DATA_FOUND
               THEN
                  r_komitent.st_preb_obcina    := 'NEZNANO';
            END;

            -- PTT stalnega prebivališča je obvezen podatek
            IF check_if_numeric (r_rec.sp_ptt_id)
            THEN
               r_komitent.st_preb_ptt_id    := r_rec.sp_ptt_id;
            ELSE
               l_napaka_id    :=
                  pkg_error_util.zapisi_napako
                     (l_tabela,
                      p_polnjenje_id,
                      l_program,
                      'PTT STALNEGA PREBIVALIŠČA KOMITENTA JE NEPRAVILEN!',
                      'ST_PREB_PTT_ID'
                     );
               RAISE skip_row;
            END IF;

            -- Država stalnega prebivališča je obvezen podatek
            BEGIN
               SELECT drzava_opis
                 INTO r_komitent.st_preb_drzava
                 FROM sif_drzava
                WHERE drzava_id = r_rec.sp_drzava_id;
            EXCEPTION
               WHEN NO_DATA_FOUND
               THEN
                  l_napaka_id    :=
                     pkg_error_util.zapisi_napako
                        (l_tabela,
                         p_polnjenje_id,
                         l_program,
                         'DRŽAVA STALNEGA PREBIVALIŠČA KOMITENTA NE OBSTAJA V ŠIFRANTU
SIF_DRZAVA!',
                         'ST_PREB_DRZAVA'
                        );
                  RAISE skip_row;
            END;

            -- Če obstaja ulica zašasnega prebivališča, potem so vsi podatki
            -- začasnega prebivališča obvezni, sicer jih ne polnimo.
            IF r_rec.zp_ulica IS NOT NULL
            THEN
               -- Ulica začasnega prebivališča je obvezen podatek
               r_komitent.zac_preb_ulica    := edit_ulica (r_rec.zp_ulica);

               -- Hišna številka začasnega prebivališča je obvezen podatek
               IF check_if_numeric (r_rec.zp_hisna_st)
               THEN
                  r_komitent.zac_preb_hisna_st    :=
                                          edit_hisna_st (r_rec.zp_hisna_st);
               ELSE
                  l_napaka_id    :=
                     pkg_error_util.zapisi_napako
                        (l_tabela,
```

```
                        p_polnjenje_id,
                        l_program,
                        'HIŠNA ŠT. ZAČASNEGA PREBIVALIŠČA KOMITENTA JE NEPRAVILNA!',
                        'ZAC_PREB_HISNA_ST'
                       );
                  RAISE skip_row;
               END IF;

               -- Kraj začasnega prebivališča je obvezen podatek
               IF r_rec.zp_kraj IS NOT NULL
               THEN
                  r_komitent.zac_preb_kraj    := UPPER (r_rec.zp_kraj);
               ELSE
                  l_napaka_id    :=
                     pkg_error_util.zapisi_napako
                        (l_tabela,
                         p_polnjenje_id,
                         l_program,
                         'KRAJ ZAČASNEGA PREBIVALIŠČA KOMITENTA JE PRAZEN!',
                         'ZAC_PREB_KRAJ'
                        );
                  RAISE skip_row;
               END IF;

               BEGIN
                  SELECT opis
                    INTO r_komitent.zac_preb_obcina
                    FROM sif_obcina
                   WHERE obcina_id = r_rec.zp_obcina_id;
               EXCEPTION
                  WHEN NO_DATA_FOUND
                  THEN
                     r_komitent.zac_preb_obcina    := 'NEZNANO';
               END;

               -- PTT začasnega prebivališča je obvezen podatek
               IF check_if_numeric (r_rec.zp_ptt_id)
               THEN
                  r_komitent.zac_preb_ptt_id    := r_rec.zp_ptt_id;
               ELSE
                  l_napaka_id    :=
                     pkg_error_util.zapisi_napako
                        (l_tabela,
                         p_polnjenje_id,
                         l_program,
                         'PTT ZAČASNEGA PREBIVALIŠČA KOMITENTA JE NEPRAVILEN!',
                         'ZAC_PREB_PTT_ID'
                        );
                  RAISE skip_row;
               END IF;

               -- Država začasnega prebivališča je obvezen podatek
               BEGIN
                  SELECT drzava_opis
                    INTO r_komitent.zac_preb_drzava
                    FROM sif_drzava
                   WHERE drzava_id = r_rec.zp_drzava_id;
               EXCEPTION
                  WHEN NO_DATA_FOUND
                  THEN
                     l_napaka_id    :=
                        pkg_error_util.zapisi_napako
                           (l_tabela,
                            p_polnjenje_id,
                            l_program,
                            'DRŽAVA ZAČASNEGA PREBIVALIŠČA KOMITENTA NE OBSTAJA V ŠIFRANTU
SIF_DRZAVA!',
                            'ZAC_PREB_DRZAVA'
                           );
                     RAISE skip_row;
               END;
            END IF;

            -- Ulica izpiska je obvezen podatek
            IF r_rec.izpis_ulica IS NOT NULL
            THEN
```

```
   r_komitent.izpisek_ulica    :=
                                edit_ulica (r_rec.izpis_ulica);
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
               (l_tabela,
                p_polnjenje_id,
                l_program,
                'ULICA IZPISKA KOMITENTA JE OBVEZEN PODATEK!',
                'IZPISEK_ULICA'
               );
   RAISE skip_row;
END IF;

-- Hišna številka izpiska je obvezen podatek
IF check_if_numeric (r_rec.izpis_hisna_st)
THEN
   r_komitent.izpisek_hisna_st    :=
                           edit_hisna_st (r_rec.izpis_hisna_st);
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                (l_tabela,
                 p_polnjenje_id,
                 l_program,
                 'HIŠNA ŠT. IZPISKA KOMITENTA JE NEPRAVILNA!',
                 'IZPISEK_HISNA_ST'
                );
   RAISE skip_row;
END IF;

-- Kraj izpiska je obvezen podatek
IF r_rec.izpis_kraj IS NOT NULL
THEN
   r_komitent.izpisek_kraj    := UPPER (r_rec.izpis_kraj);
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                         (l_tabela,
                          p_polnjenje_id,
                          l_program,
                          'KRAJ IZPISKA KOMITENTA JE PRAZEN!',
                          'IZPISEK_KRAJ'
                         );
   RAISE skip_row;
END IF;

BEGIN
   SELECT opis
     INTO r_komitent.izpisek_obcina
     FROM sif_obcina
    WHERE obcina_id = r_rec.izpis_obcina_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      r_komitent.izpisek_obcina    := 'NEZNANO';
END;

-- PTT izpiska je obvezen podatek
IF check_if_numeric (r_rec.izpis_ptt_id)
THEN
   r_komitent.izpisek_ptt_id    := r_rec.izpis_ptt_id;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                       (l_tabela,
                        p_polnjenje_id,
                        l_program,
                        'PTT IZPISKA KOMITENTA JE NEPRAVILEN!',
                        'IZPISEK_PTT_ID'
                       );
   RAISE skip_row;
END IF;

-- Država izpiska je obvezen podatek
BEGIN
```

```
   SELECT drzava_opis
     INTO r_komitent.izpisek_drzava
     FROM sif_drzava
    WHERE drzava_id = r_rec.izpis_drzava_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
            (l tabela,
             p_polnjenje_id,
             l_program,
             'DRŽAVA IZPISKA KOMITENTA NE OBSTAJA V ŠIFRANTU SIF_DRZAVA!',
             'IZPISEK_DRZAVA'
            );
      RAISE skip_row;
END;

-- POLNJENJE DIMENZIJE DIM_FO_KOMITENT
IF b_nov_komitent
THEN
   INSERT INTO dim_fo_komitent
        VALUES r_komitent;
ELSE
   UPDATE dim_fo_komitent
      SET ime = r_komitent.ime,
          priimek = r_komitent.priimek,
          titula = NVL (r_komitent.titula, titula),
          spol = r_komitent.spol,
          emso = r_komitent.emso,
          davcna_st = r_komitent.davcna_st,
          dat_rojstva =
                     NVL (r_komitent.dat_rojstva, dat_rojstva),
          dat_smrti = NVL (r_komitent.dat_smrti, dat_smrti),
          kraj_rojstva =
                    NVL (r_komitent.kraj_rojstva, kraj_rojstva),
          drz_rojstva = r_komitent.drz_rojstva,
          drzavljanstvo =
                  NVL (r_komitent.drzavljanstvo, drzavljanstvo),
          rezident = NVL (r_komitent.rezident, rezident),
          poklic_id = r_komitent.poklic_id,
          poklic_opis = r_komitent.poklic_opis,
          izobrazba_id = r_komitent.izobrazba_id,
          izob_stopnja = r_komitent.izob_stopnja,
          izobrazba_opis = r_komitent.izobrazba_opis,
          delovno_mesto =
                  NVL (r_komitent.delovno_mesto, delovno_mesto),
          telefon = r_komitent.telefon,
          fax = r_komitent.fax,
          gsm1 = r_komitent.gsm1,
          gsm2 = NVL (r_komitent.gsm2, gsm2),
          email = NVL (r_komitent.email, email),
          st_preb_ulica = r_komitent.st_preb_ulica,
          st_preb_hisna_st = r_komitent.st_preb_hisna_st,
          st_preb_kraj = r_komitent.st_preb_kraj,
          st_preb_obcina = r_komitent.st_preb_obcina,
          st_preb_ptt_id = r_komitent.st_preb_ptt_id,
          st_preb_drzava = r_komitent.st_preb_drzava,
          zac_preb_ulica = r_komitent.zac_preb_ulica,
          zac_preb_hisna_st = r_komitent.zac_preb_hisna_st,
          zac_preb_kraj = r_komitent.zac_preb_kraj,
          zac_preb_obcina = r_komitent.zac_preb_obcina,
          zac_preb_ptt_id = r_komitent.zac_preb_ptt_id,
          zac_preb_drzava = r_komitent.zac_preb_drzava,
          izpisek_ulica = r_komitent.izpisek_ulica,
          izpisek_hisna_st = r_komitent.izpisek_hisna_st,
          izpisek_kraj = r_komitent.izpisek_kraj,
          izpisek_obcina = r_komitent.izpisek_obcina,
          izpisek_ptt_id = r_komitent.izpisek_ptt_id,
          izpisek_drzava = r_komitent.izpisek_drzava
    WHERE komitent_id = l_komitent_id;
END IF;

-- ARHIVIRANJE USPEŠNO OBDELANEGA ZAPISA
arh_temp_fo_komitent (r_rec.rid);
```

```
            -- BRISANJE USPEŠNO OBDELANEGA ZAPISA
            DELETE      temp_fo_komitent
                  WHERE ROWID = r_rec.rid;

            l_st_obdelanih_vrstic       := l_st_obdelanih_vrstic + 1;
         EXCEPTION
            -- OB NAPAKI OSTANE ZAPIS v TEMP TABELI in določimo mu polje NAPAKA_ID
            WHEN skip_row
            THEN
               UPDATE temp fo komitent
                  SET napaka_id = l_napaka_id
                WHERE ROWID = r_rec.rid;
         END;
      END LOOP;

      p_message    :=
            pkg_load_util.st_obdelanih_vrstic_opis (l_st_obdelanih_vrstic);
      p_status     := 'USPEŠNO';
   EXCEPTION
      WHEN OTHERS
      THEN
         pkg_error_util.get_oracle_napaka (p_status, p_message);
   END;
END pkg_dim_fo_komitent;
/
```

7.4.4 Programski paket PKG_DIM_RACUN_TRR

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE dw_stage.pkg_dim_racun_trr
AS
/******************************************************************************
   NAME:       PKG_DIM_RACUN_TRR
   PURPOSE:    Polnjenje matičnih podatkov za transakcijske račune iz izvora TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        28.8.2008   mladen babic     1. Created this package.
******************************************************************************/
   PROCEDURE arh_temp_racun_trr (p_rowid ROWID);

   PROCEDURE load_temp_racun_trr (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );

   PROCEDURE load_dim_racun_trr (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );
END pkg_dim_racun_trr;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_stage.pkg_dim_racun_trr
AS
/******************************************************************************
   NAME:       PKG_DIM_RACUN_TRR
   PURPOSE:    Polnjenje matičnih podatkov za transakcijske račune iz izvora TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        28.8.2008   mladen babic     1. Created this package.
******************************************************************************/
   PROCEDURE arh_temp_racun_trr (p_rowid ROWID)
   IS
```

```
BEGIN
   INSERT INTO arh_racun_trr
               (partija_id, produkt_id, datodp, datzap, datakt,
                zad_status_id, boniteta_id, mat_pe_id, zad_pe_id,
                lim_znes, lim_velj_od, lim_velj_do, lim_dat_odobr,
                lim_dat_preklic, sprememba, dat_arh, komitent_id)
      SELECT partija_id, produkt_id, datodp, datzap, datakt,
             zad_status_id, boniteta_id, mat_pe_id, zad_pe_id,
             lim_znes, lim_velj_od, lim_velj_do, lim_dat_odobr,
             lim dat preklic, sprememba, SYSDATE, komitent_id
        FROM temp_racun_trr
       WHERE ROWID = p_rowid;
END;

PROCEDURE load_temp_racun_trr (
   p_polnjenje_id   IN       NUMBER,
   p_status         IN OUT   VARCHAR2,
   p_message        IN OUT   VARCHAR2
)
IS
BEGIN
   EXECUTE IMMEDIATE 'truncate table temp_racun_trr';

   INSERT INTO temp_racun_trr
               (partija_id, produkt_id, datodp, datzap, datakt,
                zad_status_id, boniteta_id, mat_pe_id, zad_pe_id,
                lim_znes, lim_velj_od, lim_velj_do, lim_dat_odobr,
                lim_dat_preklic, sprememba, komitent_id, ind_lastnik)
      SELECT partija_id, TRIM (produkt_id), datodp, datzap, datakt,
             zad_status_id, boniteta_id, mat_pe_id, zad_pe_id,
             lim_znes, lim_velj_od, lim_velj_do, lim_dat_odobr,
             lim_dat_preklic, sprememba, komitent_id, ind_lastnik
        FROM vw_partija_trr_dw izv
       WHERE NOT EXISTS (SELECT NULL
                           FROM arh_racun_trr arh
                          WHERE izv.sprememba = arh.sprememba);

   p_message    :=
                pkg_load_util.st_obdelanih_vrstic_opis (SQL%ROWCOUNT);
   p_status     := 'USPEŠNO';
EXCEPTION
   WHEN OTHERS
   THEN
      pkg_error_util.get_oracle_napaka (p_status, p_message);
END;

PROCEDURE load_dim_racun_trr (
   p_polnjenje_id   IN       NUMBER,
   p_status         IN OUT   VARCHAR2,
   p_message        IN OUT   VARCHAR2
)
IS
   skip_row                EXCEPTION;
   l_st_obdelanih_vrstic   NUMBER                              := 0;
   l_komitent_id           dim_fo_komitent.komitent_id%TYPE;
   r_racun_trr             dim_racun_trr%ROWTYPE;
   r_racun                 dim_racun%ROWTYPE;
   b_nov_trr               BOOLEAN;
   l_lastnik               rel_racun_komitent.lastnik%TYPE;
   l_tabela                VARCHAR2 (100)           := 'TEMP_RACUN_TRR';
   l_program               VARCHAR2 (100)       := 'LOAD_DIM_RACUN_TRR';
   l_napaka_id             NUMBER;
   l_dan_id                NUMBER;
BEGIN
   -- zaradi ponovitve koraka polnjenja, zbrišemo morebitne napake
   UPDATE temp_racun_trr
      SET napaka_id = NULL
    WHERE napaka_id IS NOT NULL;

   -- OBDELAVA IZVORNIH PODATKOV
   FOR r_rec IN (SELECT a.ROWID rid, a.*
                   FROM temp_racun_trr a)
   LOOP
      BEGIN
         l_napaka_id                     := NULL;
```

```
-- UGOTAVLJANJE OBSTOJA RACUNA V PPS
BEGIN
   SELECT racun_id
     INTO r_racun_trr.racun_id
     FROM dim_racun_trr
    WHERE racun_id_izvor = r_rec.partija_id;

   r_racun.racun_id    := r_racun_trr.racun_id;
   b_nov_trr           := FALSE;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      b_nov_trr           := TRUE;

      -- DOLOCANJE SIFRE V PPS ZA NOV RACUN
      SELECT seq_dim_racun.NEXTVAL
        INTO r_racun_trr.racun_id
        FROM DUAL;

      r_racun.racun_id    := r_racun_trr.racun_id;
END;

-- UGOTAVLJANJE OBSTOJA KOMITENTA V PPS; Če ne obstaja javimo napako!
BEGIN
   SELECT komitent_id
     INTO l_komitent_id
     FROM dim_fo_komitent
    WHERE komitent_id_izvor = r_rec.komitent_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
             (l_tabela,
              p_polnjenje_id,
              l_program,
              'KOMITENT RAČUNA NE OBSTAJA V DIM_FO_KOMITENT!',
              'KOMITENT_ID'
             );
      RAISE skip_row;
END;

r_racun_trr.racun_id_izvor       := r_rec.partija_id;
r_racun.racun_id_izvor           := r_racun_trr.racun_id_izvor;

-- PRODUKT MORA OBSTAJATI V DIMENZIJI PRODUKTOV, SICER JAVIMO NAPAKO
BEGIN
   SELECT produkt_id
     INTO r_racun_trr.produkt_id
     FROM dim_produkt_trr
    WHERE produkt_id_izvor = r_rec.produkt_id;

   r_racun.produkt_id    := r_racun_trr.produkt_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
              (l_tabela,
               p_polnjenje_id,
               l_program,
               'PRODUKT RAČUNA NE OBSTAJA V DIM_PRODUKT_TRR!',
               'PRODUKT_ID'
              );
      RAISE skip_row;
END;

IF r_rec.datodp IS NOT NULL
THEN
   r_racun_trr.dat_odprtja    := r_rec.datodp;
   r_racun.dat_odprtja        := r_racun_trr.dat_odprtja;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                         (l_tabela,
                          p_polnjenje_id,
```

```
                        l_program,
                        'DATUM ODPRTJA NE SME BITI PRAZEN!',
                        'DAT_ODPRTJA'
                       );
   RAISE skip_row;
END IF;

-- DATUM ZAPRTJA RAČUNA MORA BITI VEČJI OD DATUMA ODPRTJA RAČUNA
IF NVL (r_rec.datzap, TO_DATE ('1.1.2999', 'dd.mm.yyyy')) >
                                                    r_rec.datodp
THEN
   r_racun_trr.dat_zaprtja    := r_rec.datzap;
   r_racun.dat_zaprtja        := r_racun_trr.dat_zaprtja;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
         (l_tabela,
          p_polnjenje_id,
          l_program,
          'DATUM ZAPRTJA RAČUNA JE MANJŠI OD DATUMA ODPRTJA RAČUNA!',
          'DAT_ZAPRTJA'
         );
   RAISE skip_row;
END IF;

-- DATUM AKTIVACIJE RAČUNA MORA BITI VEČJI ALI ENAK OD DATUMA ODPRTJA RAČUNA
IF r_rec.datakt >= r_rec.datodp
THEN
   r_racun_trr.dat_aktivac    := r_rec.datakt;
   r_racun.dat_aktivac        := r_racun_trr.dat_aktivac;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
         (l_tabela,
          p_polnjenje_id,
          l_program,
          'DATUM AKTIVACIJE RAČUNA JE MANJŠI OD DATUMA ODPRTJA RAČUNA!',
          'DAT_AKTIVAC'
         );
   RAISE skip_row;
END IF;

-- STATUS RAČUNA MORA OBSTAJATI V DIMENZIJI STATUSOV, SICER JAVIMO NAPAKO
BEGIN
   SELECT status_id
     INTO r_racun_trr.zadnji_status_id
     FROM dim_status
    WHERE status_id_izvor = r_rec.zad_status_id;

   r_racun.zadnji_status_id    := r_racun_trr.zadnji_status_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
                    (l_tabela,
                     p_polnjenje_id,
                     l_program,
                     'STATUS RAČUNA NE OBSTAJA V DIM_STATUS!',
                     'ZADNJI_STATUS_ID'
                    );
      RAISE skip_row;
END;

-- BONITETA RAČUNA MORA OBSTAJATI V SIFRANTU, SICER JAVIMO NAPAKO
BEGIN
   SELECT boniteta_id, opis
     INTO r_racun_trr.boniteta_ind, r_racun_trr.boniteta_opis
     FROM sif_boniteta
    WHERE boniteta_id = r_rec.boniteta_id;

   r_racun.boniteta_ind     := r_racun_trr.boniteta_ind;
   r_racun.boniteta_opis    := r_racun_trr.boniteta_opis;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
```

```
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
                (l_tabela,
                 p_polnjenje_id,
                 l_program,
                 'BONITETA RAČUNA NE OBSTAJA V SIF_BONITETA!',
                 'BONITETA_IND'
                );
      RAISE skip_row;
END;

-- MATIČNA PE MORA OBSTAJATI V DIM_POSLOVNA_ENOTA, SICER JAVIMO NAPAKO
BEGIN
   SELECT pe_id
     INTO r_racun_trr.mat_pe_id
     FROM dim_poslovna_enota
    WHERE pe_id_izvor = r_rec.mat_pe_id;

   r_racun.mat_pe_id    := r_racun_trr.mat_pe_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
                (l_tabela,
                 p_polnjenje_id,
                 l_program,
                 'MAT_PE_ID NE OBSTAJA V DIM_POSLOVNA_ENOTA!',
                 'MAT_PE_ID'
                );
      RAISE skip_row;
END;

-- ZADNJA PE MORA OBSTAJATI V DIM_POSLOVNA_ENOTA, SICER JAVIMO NAPAKO
BEGIN
   SELECT pe_id
     INTO r_racun_trr.zadnja_pe_id
     FROM dim_poslovna_enota
    WHERE pe_id_izvor = r_rec.zad_pe_id;

   r_racun.zadnja_pe_id    := r_racun_trr.zadnja_pe_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
             (l_tabela,
              p_polnjenje_id,
              l_program,
              'ZADNJA_PE_ID NE OBSTAJA V DIM_POSLOVNA_ENOTA!',
              'ZADNJA_PE_ID'
             );
      RAISE skip_row;
END;

r_racun_trr.limit_znesek         := NVL (r_rec.lim_znes, 0);
r_racun_trr.limit_dat_velj_od    := r_rec.lim_velj_od;

IF r_rec.lim_velj_do > r_rec.lim_velj_od
THEN
   r_racun_trr.limit_dat_velj_do    := r_rec.lim_velj_do;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                    (l_tabela,
                     p_polnjenje_id,
                     l_program,
                     'DATUM VELJAVNOSTI LIMITA NI PRAVILEN!',
                     'LIM_VELJ_DO'
                    );
   RAISE skip_row;
END IF;

r_racun_trr.limit_dat_odobr      := r_rec.lim_dat_odobr;
r_racun_trr.limit_dat_preklic    := r_rec.lim_dat_preklic;
```

```
IF r_rec.ind_lastnik = 'L'
THEN
   l_lastnik    := 'DA';
ELSIF r_rec.ind_lastnik = 'P'
THEN
   l_lastnik    := 'NE';
ELSIF r_rec.ind_lastnik = 'O'
THEN
   l_lastnik    := 'BP';
-- BIVSI POOBLASCENEC
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                           (l_tabela,
                            p_polnjenje_id,
                            l_program,
                            'INDIKATOR LASTNIKA NI PRAVILEN!',
                            'IND_LASTNIK'
                           );
   RAISE skip_row;
END IF;

-- POLNJENJE DIMENZIJE DIM_RACUN IN DIM_RACUN_TRR
IF b_nov_trr
THEN
   INSERT INTO dim_racun_trr
        VALUES r_racun_trr;

   INSERT INTO dim_racun
        VALUES r_racun;

   INSERT INTO rel_racun_komitent
               (racun_id, komitent_id, lastnik
               )
        VALUES (r_racun.racun_id, l_komitent_id, l_lastnik
               );

   -- se zapis inicialnega stanja:
   SELECT dan_id
     INTO l_dan_id
     FROM dim_dan
    WHERE datum = r_racun_trr.dat_odprtja;

   INSERT INTO f_fo_stanje_dan_trr
               (dan_id, racun_id, valuta_id,
                status_id,
                produkt_id,
                pe_id, dnevno_stanje, dnevno_stanje_val,
                st_transakcij
               )
        VALUES (l_dan_id, r_racun_trr.racun_id, 1,
                r_racun_trr.zadnji_status_id,
                r_racun_trr.produkt_id,
                r_racun_trr.zadnja_pe_id, 0, 0,
                0
               );

   INSERT INTO f_fo_stanje_dan
               (dan_id, racun_id, valuta_id,
                status_id,
                produkt_id,
                pe_id, dnevno_stanje, dnevno_stanje_val
               )
        VALUES (l_dan_id, r_racun_trr.racun_id, 1,
                r_racun_trr.zadnji_status_id,
                r_racun_trr.produkt_id,
                r_racun_trr.zadnja_pe_id, 0, 0
               );
ELSE
   UPDATE dim_racun_trr
      SET produkt_id = r_racun_trr.produkt_id,
          dat_odprtja = r_racun_trr.dat_odprtja,
          dat_zaprtja = r_racun_trr.dat_zaprtja,
          dat_aktivac = r_racun_trr.dat_aktivac,
          zadnji_status_id = r_racun_trr.zadnji_status_id,
          boniteta_ind = r_racun_trr.boniteta_ind,
```

```
                    boniteta_opis = r_racun_trr.boniteta_opis,
                    mat_pe_id = r_racun_trr.mat_pe_id,
                    zadnja_pe_id = r_racun_trr.zadnja_pe_id,
                    limit_znesek = r_racun_trr.limit_znesek,
                    limit_dat_velj_od = r_racun_trr.limit_dat_velj_od,
                    limit_dat_velj_do = r_racun_trr.limit_dat_velj_do,
                    limit_dat_odobr = r_racun_trr.limit_dat_odobr,
                    limit_dat_preklic = r_racun_trr.limit_dat_preklic
                WHERE racun_id = r_racun_trr.racun_id;

              UPDATE dim_racun
                 SET produkt_id = r_racun.produkt_id,
                     dat_odprtja = r_racun.dat_odprtja,
                     dat_zaprtja = r_racun.dat_zaprtja,
                     dat_aktivac = r_racun.dat_aktivac,
                     zadnji_status_id = r_racun.zadnji_status_id,
                     boniteta_ind = r_racun.boniteta_ind,
                     boniteta_opis = r_racun.boniteta_opis,
                     mat_pe_id = r_racun.mat_pe_id,
                     zadnja_pe_id = r_racun.zadnja_pe_id
                WHERE racun_id = r_racun.racun_id;

              UPDATE rel_racun_komitent
                 SET lastnik = l_lastnik
                WHERE racun_id = r_racun_trr.racun_id
                  AND komitent_id = l_komitent_id;
            END IF;

            -- ARHIVIRANJE USPEŠNO OBDELANEGA ZAPISA
            arh_temp_racun_trr (r_rec.rid);

            -- BRISANJE USPEŠNO OBDELANEGA ZAPISA
            DELETE      temp_racun_trr
                  WHERE ROWID = r_rec.rid;

            l_st_obdelanih_vrstic           := l_st_obdelanih_vrstic + 1;
         EXCEPTION
            -- OB NAPAKI OSTANE ZAPIS v TEMP TABELI in določimo mu polje NAPAKA_ID
            WHEN skip_row
            THEN
               UPDATE temp_racun_trr
                  SET napaka_id = l_napaka_id
                WHERE ROWID = r_rec.rid;
         END;
      END LOOP;

      p_message    :=
            pkg_load_util.st_obdelanih_vrstic_opis (l_st_obdelanih_vrstic);
      p_status     := 'USPEŠNO';
   EXCEPTION
      WHEN OTHERS
      THEN
         pkg_error_util.get_oracle_napaka (p_status, p_message);
   END;
END pkg_dim_racun_trr;
/
```

7.4.5 Programski paket PKG_DIM_KARTICA_TRR

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE dw_stage.pkg_dim_kartica_trr
AS
/******************************************************************************
   NAME:       PKG_DIM_KARTICA_TRR
   PURPOSE:    Polnjenje dimenzije TRR kartic DIM_KARTICA_TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        28.8.2008   mladen babic     1. Created this package.
******************************************************************************/
   PROCEDURE arh_temp_kartica_trr (p_rowid ROWID);
```

```
   PROCEDURE load_temp_kartica_trr (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );

   PROCEDURE load_dim_kartica_trr (
      p_polnjenje_id   IN       NUMBER,
      p status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );
END pkg_dim_kartica_trr;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_stage.pkg_dim_kartica_trr
AS
/******************************************************************************
   NAME:       PKG_DIM_KARTICA_TRR
   PURPOSE:    Polnjenje dimenzije TRR kartic DIM_KARTICA_TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        28.8.2008   mladen babic     1. Created this package.
******************************************************************************/
   PROCEDURE arh_temp_kartica_trr (p_rowid ROWID)
   IS
   BEGIN
      INSERT INTO arh_kartica_trr
                  (st_kart, partija_id, komitent_id, datizdaje, datvelj,
                   tip_kart_id, kart_status_id, dat_blokade,
                   vzrok_blokade, sprememba, dat_arh)
         SELECT st_kart, partija_id, komitent_id, datizdaje, datvelj,
                tip_kart_id, kart_status_id, dat_blokade, vzrok_blokade,
                sprememba, SYSDATE
           FROM temp_kartica_trr
          WHERE ROWID = p_rowid;
   END;

   PROCEDURE load_temp_kartica_trr (
      p_polnjenje_id   IN       NUMBER,
      p status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   )
   IS
   BEGIN
      EXECUTE IMMEDIATE 'truncate table temp_kartica_trr';

      INSERT INTO temp_kartica_trr
                  (st_kart, partija_id, komitent_id, datizdaje, datvelj,
                   tip_kart_id, kart_status_id, dat_blokade,
                   vzrok_blokade, sprememba)
         SELECT st_kart, partija_id, komitent_id, datizdaje, datvelj,
                tip_kart_id, kart_status_id, dat_blokade,
                TRIM (vzrok_blokade), sprememba
           FROM vw_kartice_trr_dw izv
          WHERE NOT EXISTS (SELECT NULL
                              FROM arh_kartica_trr arh
                             WHERE izv.sprememba = arh.sprememba);

      p_message    :=
                   pkg_load_util.st_obdelanih_vrstic_opis (SQL%ROWCOUNT);
      p_status     := 'USPEŠNO';
   EXCEPTION
      WHEN OTHERS
      THEN
         pkg_error_util.get_oracle_napaka (p_status, p_message);
   END;

   PROCEDURE load_dim_kartica_trr (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
```

```
   p_message        IN OUT   VARCHAR2
)
IS
   skip_row                EXCEPTION;
   l_st_obdelanih_vrstic   NUMBER                          := 0;
   l_kartica_id            dim_kartica_trr.kartica_id%TYPE;
   r_kartica               dim_kartica_trr%ROWTYPE;
   b_nova_kartica          BOOLEAN;
   l_tabela                VARCHAR2 (100)         := 'TEMP_KARTICA_TRR';
   l_program               VARCHAR2 (100)     := 'LOAD_DIM_KARTICA_TRR';
   l_napaka_id             NUMBER;
BEGIN
   -- zaradi ponovitve koraka polnjenja, zbrišemo morebitne napake
   UPDATE temp_kartica_trr
      SET napaka_id = NULL
    WHERE napaka_id IS NOT NULL;

   -- OBDELAVA IZVORNIH PODATKOV
   FOR r_rec IN (SELECT a.ROWID rid, a.*
                   FROM temp_kartica_trr a)
   LOOP
      BEGIN
         l_napaka_id               := NULL;

         -- UGOTAVLJANJE OBSTOJA KARTICE V PPS
         BEGIN
            SELECT kartica_id
              INTO l_kartica_id
              FROM dim_kartica_trr
             WHERE kartica_id_izvor = r_rec.st_kart;

            b_nova_kartica    := FALSE;
         EXCEPTION
            WHEN NO_DATA_FOUND
            THEN
               b_nova_kartica    := TRUE;

               -- DOLOCANJE SIFRE V PPS ZA NOVO KARTICO
               SELECT seq_dim_kart_trr.NEXTVAL
                 INTO r_kartica.kartica_id
                 FROM DUAL;
         END;

         -- UGOTAVLJANJE OBSTOJA RACUNA, KATEREMU KARTICA PRIPADA
         BEGIN
            SELECT racun_id
              INTO r_kartica.racun_id
              FROM dim_racun_trr
             WHERE racun_id_izvor = r_rec.partija_id;
         EXCEPTION
            WHEN NO_DATA_FOUND
            THEN
               l_napaka_id    :=
                  pkg_error_util.zapisi_napako
                           (l_tabela,
                            p_polnjenje_id,
                            l_program,
                            'RACUN KARTICE NE OBSTAJA V DIM_RACUN_TRR!',
                            'RACUN_ID'
                           );
               RAISE skip_row;
         END;

         -- UGOTAVLJANJE OBSTOJA KOMITENTA, KATEREMU KARTICA PRIPADA
         BEGIN
            SELECT komitent_id
              INTO r_kartica.komitent_id
              FROM dim_fo_komitent
             WHERE komitent_id_izvor = r_rec.komitent_id;
         EXCEPTION
            WHEN NO_DATA_FOUND
            THEN
               l_napaka_id    :=
                  pkg_error_util.zapisi_napako
                     (l_tabela,
                      p_polnjenje_id,
```

```
            l_program,
            'KOMITENT KARTICE NE OBSTAJA V DIM_FO_KOMITENT!',
            'KOMITENT_ID'
           );
      RAISE skip_row;
END;

-- DATUM IZDAJE KARTICE JE OBVEZEN PODATEK
IF r_rec.datizdaje IS NOT NULL
THEN
   r_kartica.kart_dat_izdaje    := r_rec.datizdaje;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
                  (l_tabela,
                   p_polnjenje_id,
                   l_program,
                   'DATUM IZDAJE KARTICE JE OBVEZEN PODATEK!',
                   'KART_DAT_IZDAJE'
                  );
   RAISE skip_row;
END IF;

-- DATUM VELJAVNOSTI KARTICE JE OBVEZEN PODATEK
IF r_rec.datvelj IS NOT NULL
THEN
   r_kartica.kart_dat_veljavnosti    := r_rec.datvelj;
ELSE
   l_napaka_id    :=
      pkg_error_util.zapisi_napako
             (l_tabela,
              p_polnjenje_id,
              l_program,
              'DATUM VELJAVNOSTI KARTICE JE OBVEZEN PODATEK!',
              'KART_DAT_VELJAVNOSTI'
             );
   RAISE skip_row;
END IF;

-- INDIKATOR TIPA KARTICE JE OBVEZEN PODATEK
BEGIN
   SELECT tip_kart_id
     INTO r_kartica.tip_kart_ind
     FROM sif_tip_kart
    WHERE tip_kart_id = r_rec.tip_kart_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
                (l_tabela,
                 p_polnjenje_id,
                 l_program,
                 'INDIKATOR TIPA KARTICE JE OBVEZEN PODATEK!',
                 'TIP_KART_IND'
                );
      RAISE skip_row;
END;

-- OPIS TIPA KARTICE JE OBVEZEN PODATEK
BEGIN
   SELECT tip_opis
     INTO r_kartica.tip_kart_opis
     FROM sif_tip_kart
    WHERE tip_kart_id = r_rec.tip_kart_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
                (l_tabela,
                 p_polnjenje_id,
                 l_program,
                 'INDIKATOR TIPA KARTICE JE OBVEZEN PODATEK!',
                 'TIP_KART_OPIS'
                );
```

```
      RAISE skip_row;
   END;

   -- INDIKATOR STATUSA KARTICE JE OBVEZEN PODATEK
   BEGIN
      SELECT kart_status_id
        INTO r_kartica.kart_status_ind
        FROM sif_status_kart
       WHERE kart_status_id = r_rec.kart_status_id;
   EXCEPTION
      WHEN NO_DATA_FOUND
      THEN
         l_napaka_id    :=
            pkg_error_util.zapisi_napako
                (l_tabela,
                 p_polnjenje_id,
                 l_program,
                 'INDIKATOR STATUSA KARTICE JE OBVEZEN PODATEK!',
                 'KART_STATUS_IND'
                );
      RAISE skip_row;
   END;

   -- OPIS STATUSA KARTICE JE OBVEZEN PODATEK
   BEGIN
      SELECT status_opis
        INTO r_kartica.kart_status_opis
        FROM sif_status_kart
       WHERE kart_status_id = r_rec.kart_status_id;
   EXCEPTION
      WHEN NO_DATA_FOUND
      THEN
         l_napaka_id    :=
            pkg_error_util.zapisi_napako
                     (l_tabela,
                      p_polnjenje_id,
                      l_program,
                      'OPIS STATUSA KARTICE JE OBVEZEN PODATEK!',
                      'KART_STATUS_OPIS'
                     );
      RAISE skip_row;
   END;

   r_kartica.dat_blokade      := r_rec.dat_blokade;
   r_kartica.vzrok_blokade    := UPPER (r_rec.vzrok_blokade);

   -- POLNJENJE DIMENZIJE DIM_KARTICA_TRR
   IF b_nova_kartica
   THEN
      INSERT INTO dim_kartica_trr
           VALUES r_kartica;
   ELSE
      UPDATE dim_kartica_trr
         SET kart_dat_izdaje = r_kartica.kart_dat_izdaje,
             kart_dat_veljavnosti = r_kartica.kart_dat_veljavnosti,
             tip_kart_ind = r_kartica.tip_kart_ind,
             tip_kart_opis = r_kartica.tip_kart_opis,
             kart_status_ind = r_kartica.kart_status_ind,
             kart_status_opis = r_kartica.kart_status_opis,
             dat_blokade = r_kartica.dat_blokade,
             vzrok_blokade = r_kartica.vzrok_blokade
       WHERE kartica_id = l_kartica_id;
   END IF;

   -- ARHIVIRANJE USPEŠNO OBDELANEGA ZAPISA
   arh_temp_kartica_trr (r_rec.rid);

   -- BRISANJE USPEŠNO OBDELANEGA ZAPISA
   DELETE      temp_kartica_trr
         WHERE ROWID = r_rec.rid;

   l_st_obdelanih_vrstic      := l_st_obdelanih_vrstic + 1;
EXCEPTION
   -- OB NAPAKI OSTANE ZAPIS v TEMP TABELI in določimo mu polje NAPAKA_ID
   WHEN skip_row
   THEN
```

```
               UPDATE temp_kartica_trr
                  SET napaka_id = l_napaka_id
                WHERE ROWID = r_rec.rid;
         END;
      END LOOP;

      p_message    :=
            pkg_load_util.st_obdelanih_vrstic_opis (l_st_obdelanih_vrstic);
      p_status     := 'USPEŠNO';
   EXCEPTION
      WHEN OTHERS
      THEN
         pkg_error_util.get_oracle_napaka (p_status, p_message);
   END;
END pkg_dim_kartica_trr;
/
```

7.4.6 Programski paket PKG_F_FO_TRANS_TRR

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE dw_stage.pkg_f_fo_trans_trr
AS
/******************************************************************************
   NAME:       PKG_F_FO_TRANS_TRR
   PURPOSE:    POLNJENJE PROMETA TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        29.8.2008   Mladen Babic     1. Created this package.
******************************************************************************/
   PROCEDURE arh_temp_trr_knjizbe (p_rowid ROWID);

   PROCEDURE load_temp_trr_knjizbe (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );

   PROCEDURE load_f_fo_trans_trr (
      p_polnjenje_id   IN       NUMBER,
      p_status         IN OUT   VARCHAR2,
      p_message        IN OUT   VARCHAR2
   );
END pkg_f_fo_trans_trr;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_stage.pkg_f_fo_trans_trr
AS
/******************************************************************************
   NAME:       PKG_F_FO_TRANS_TRR
   PURPOSE:    POLNJENJE PROMETA TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        29.8.2008   Mladen Babic     1. Created this package.
******************************************************************************/
   PROCEDURE arh_temp_trr_knjizbe (p_rowid ROWID)
   IS
   BEGIN
      INSERT INTO arh_trr_knjizbe
                  (id_knjizbe, trans_id, partija_id, dat_knjizenja,
                   valuta_id, lokacija_id, znesek, sprememba, dat_arh)
         SELECT id_knjizbe, trans_id, partija_id, dat_knjizenja,
                valuta_id, lokacija_id, znesek, sprememba, SYSDATE
           FROM temp_trr_knjizbe
          WHERE ROWID = p_rowid;
   END;
```

```
PROCEDURE load_temp_trr_knjizbe (
   p_polnjenje_id   IN       NUMBER,
   p_status         IN OUT   VARCHAR2,
   p_message        IN OUT   VARCHAR2
)
IS
BEGIN
   EXECUTE IMMEDIATE 'truncate table temp_trr_knjizbe';

   INSERT INTO temp_trr_knjizbe
               (id_knjizbe, trans_id, partija_id, dat_knjizenja,
                valuta_id, lokacija_id, znesek, sprememba)
      SELECT id_knjizbe, trans_id, partija_id, dat_knjizenja,
             valuta_id, lokacija_id, znesek, sprememba
        FROM vw_trr_knjizbe_dw izv
       WHERE NOT EXISTS (SELECT NULL
                           FROM arh_trr_knjizbe arh
                          WHERE izv.sprememba = arh.sprememba);

   p_message    :=
                pkg_load_util.st_obdelanih_vrstic_opis (SQL%ROWCOUNT);
   p_status     := 'USPEŠNO';
EXCEPTION
   WHEN OTHERS
   THEN
      pkg_error_util.get_oracle_napaka (p_status, p_message);
END;

PROCEDURE load_f_fo_trans_trr (
   p_polnjenje_id   IN       NUMBER,
   p_status         IN OUT   VARCHAR2,
   p_message        IN OUT   VARCHAR2
)
IS
   skip_row                EXCEPTION;
   l_st_obdelanih_vrstic   NUMBER                    := 0;
   r_trans                 f_fo_trans_trr%ROWTYPE;
   l_dummy                 VARCHAR2 (1);
   l_tabela                VARCHAR2 (100)         := 'TEMP_TRR_KNJIZBE';
   l_program               VARCHAR2 (100)      := 'LOAD_F_FO_TRANS_TRR';
   l_napaka_id             NUMBER;
BEGIN
   -- zaradi ponovitve koraka polnjenja, zbrišemo morebitne napake
   UPDATE temp_trr_knjizbe
      SET napaka_id = NULL
    WHERE napaka_id IS NOT NULL;

   -- OBDELAVA IZVORNIH PODATKOV
   FOR r_rec IN (SELECT a.ROWID rid, a.*
                   FROM temp_trr_knjizbe a)
   LOOP
      BEGIN
         l_napaka_id              := NULL;

         -- UGOTAVLJANJE OBSTOJA RACUNA V PPS
         BEGIN
            SELECT racun_id
              INTO r_trans.racun_id
              FROM dim_racun_trr
             WHERE racun_id_izvor = r_rec.partija_id;
         EXCEPTION
            WHEN NO_DATA_FOUND
            THEN
               l_napaka_id    :=
                  pkg_error_util.zapisi_napako
                                 (l_tabela,
                                  p_polnjenje_id,
                                  l_program,
                                  'RACUN NE OBSTAJA V DIM_RACUN_TRR!',
                                  'RACUN_ID'
                                 );
               RAISE skip_row;
         END;

         -- UGOTAVLJANJE OBSTOJA DAN-a V PPS
```

```
BEGIN
   SELECT dan_id
     INTO r_trans.dan_id
     FROM dim_dan
    WHERE datum = r_rec.dat_knjizenja;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg error util.zapisi napako
                                (l_tabela,
                                 p_polnjenje_id,
                                 l_program,
                                 'DAN NE OBSTAJA V DIM_DAN!',
                                 'DAN_ID'
                                );
      RAISE skip_row;
END;

-- UGOTAVLJANJE OBSTOJA VALUTE V PPS
BEGIN
   SELECT valuta_id
     INTO r_trans.valuta_id
     FROM dim_valuta
    WHERE valuta_id_izvor = r_rec.valuta_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
                           (l_tabela,
                            p_polnjenje_id,
                            l_program,
                            'VALUTA NE OBSTAJA V DIM_VALUTA!',
                            'VALUTA_ID'
                           );
      RAISE skip_row;
END;

-- UGOTAVLJANJE OBSTOJA TRANSAKCIJE V PPS
BEGIN
   SELECT trans_id
     INTO r_trans.trans_id
     FROM dim_transakcija
    WHERE trans_id_izvor = r_rec.trans_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
                  (l_tabela,
                   p_polnjenje_id,
                   l_program,
                   'TRANSAKCIJA NE OBSTAJA V DIM_TRANSAKCIJA!',
                   'TRANS_ID'
                  );
      RAISE skip_row;
END;

-- UGOTAVLJANJE OBSTOJA LOKACIJE V PPS
BEGIN
   SELECT lokacija_id
     INTO r_trans.lokacija_id
     FROM dim_lokacija
    WHERE lokacija_id_izvor = r_rec.lokacija_id;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      l_napaka_id    :=
         pkg_error_util.zapisi_napako
                        (l_tabela,
                         p_polnjenje_id,
                         l_program,
                         'LOKACIJA NE OBSTAJA V DIM_LOKACIJA!',
                         'LOKACIJA_ID'
                        );
```

```
            RAISE skip_row;
         END;

         r_trans.znesek           := NVL (r_rec.znesek, 0);
         r_trans.knjizba_id       := r_rec.id_knjizbe;

         -- ČE TRANS ŽE OBSTAJA JO NAJPREJ ZBRISEMO, POTEM PA TAKOJ INSERTAMO
         BEGIN
            SELECT 'X'
              INTO l dummy
              FROM f_fo_trans_trr
             WHERE knjizba_id = r_trans.knjizba_id;

            DELETE      f_fo_trans_trr
                  WHERE knjizba_id = r_trans.knjizba_id;

            pkg_f_fo_stanje_dan_trr.azuriraj_stanje (r_trans.dan_id,
                                                     r_trans.racun_id,
                                                     r_trans.valuta_id,
                                                     r_trans.trans_id,
                                                     r_trans.znesek,
                                                     'D'
                                                    );
         EXCEPTION
            WHEN NO_DATA_FOUND
            THEN
               NULL;
         END;

         INSERT INTO f_fo_trans_trr
              VALUES r_trans;

         pkg_f_fo_stanje_dan_trr.azuriraj_stanje (r_trans.dan_id,
                                                  r_trans.racun_id,
                                                  r_trans.valuta_id,
                                                  r_trans.trans_id,
                                                  r_trans.znesek,
                                                  'I'
                                                 );
         -- ARHIVIRANJE USPEŠNO OBDELANEGA ZAPISA
         arh_temp_trr_knjizbe (r_rec.rid);

         -- BRISANJE USPEŠNO OBDELANEGA ZAPISA
         DELETE      temp_trr_knjizbe
               WHERE ROWID = r_rec.rid;

         l_st_obdelanih_vrstic    := l_st_obdelanih_vrstic + 1;
      EXCEPTION
         -- OB NAPAKI OSTANE ZAPIS v TEMP TABELI in določimo mu polje NAPAKA_ID
         WHEN skip_row
         THEN
            UPDATE temp_trr_knjizbe
               SET napaka_id = l_napaka_id
             WHERE ROWID = r_rec.rid;
      END;
   END LOOP;

   p_message    :=
         pkg_load_util.st_obdelanih_vrstic_opis (l_st_obdelanih_vrstic);
   p_status     := 'USPEŠNO';
EXCEPTION
   WHEN OTHERS
   THEN
      pkg_error_util.get_oracle_napaka (p_status, p_message);
END;
END pkg_f_fo_trans_trr;
/
```

7.4.7 Programski paket PKG_F_FO_STANJE_DAN_TRR

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE dw_stage.pkg_f_fo_stanje_dan_trr
```

```
AS
/******************************************************************************
   NAME:       PKG_F_FO_STANJE_DAN_TRR
   PURPOSE:    Polnjenje tabele dejstev F_FO_STANJE_DAN_TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        1.9.2008    mladen babic     1. Created this package.
******************************************************************************/
   PROCEDURE azuriraj_stanje (
      p_dan_id      f_fo_trans_trr.dan_id%TYPE,
      p_racun_id    f_fo_trans_trr.racun_id%TYPE,
      p_valuta_id   f_fo_trans_trr.valuta_id%TYPE,
      p_trans_id    f_fo_trans_trr.trans_id%TYPE,
      p_znesek      f_fo_trans_trr.znesek%TYPE,
      p_trigger     VARCHAR2
   );
END pkg_f_fo_stanje_dan_trr;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_stage.pkg_f_fo_stanje_dan_trr
AS
/******************************************************************************
   NAME:       PKG_F_FO_STANJE_DAN_TRR
   PURPOSE:    Polnjenje tabele dejstev F_FO_STANJE_DAN_TRR

   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   1.0        1.9.2008    mladen babic     1. Created this package.
******************************************************************************/
   FUNCTION val2eur (p_dan_id NUMBER, p_valuta_id NUMBER, p_znesek NUMBER)
      RETURN NUMBER
   IS
   BEGIN
      NULL;
      -- funkcija ki pretvori znesek v valuti v znesek v eurih po veljavnem tecaju na dan_id;
      RETURN p_znesek;
   END;

   PROCEDURE azuriraj_stanje (
      p dan id      f fo trans trr.dan id%TYPE,
      p_racun_id    f_fo_trans_trr.racun_id%TYPE,
      p_valuta_id   f_fo_trans_trr.valuta_id%TYPE,
      p_trans_id    f_fo_trans_trr.trans_id%TYPE,
      p_znesek      f_fo_trans_trr.znesek%TYPE,
      p_trigger     VARCHAR2
   )
   IS
      r_dan_trr             f_fo_stanje_dan_trr%ROWTYPE;
      r_dan                 f_fo_stanje_dan%ROWTYPE;
      l_trans_ind           dim_transakcija.tip_trans_ind%TYPE;
      l_znesek              NUMBER;
      l_znesek_val          NUMBER;
      l_dan_id              NUMBER;
      l_dnevno_stanje       NUMBER;
      l_dnevno_stanje_val   NUMBER;
      l_st_transakcij       NUMBER;
   BEGIN
      SELECT tip_trans_ind
        INTO l_trans_ind
        FROM dim_transakcija
       WHERE trans_id = p_trans_id;

      -- CE GRE ZA INSERT V TRANSAKCIJSKO TABELO
      IF p_trigger = 'I'
      THEN
         l_st_transakcij    := 1;

         IF l_trans_ind = 'P'
         THEN
            l_znesek_val    := p_znesek;
```

```
         ELSIF l_trans_ind = 'O'
         THEN
            l_znesek_val    := -p_znesek;
         END IF;
      -- CE GRE ZA DELETE IZ TRANSAKCIJSKE TABELE
      ELSIF p_trigger = 'D'
      THEN
         l_st_transakcij    := -1;

         IF l_trans_ind = 'P'
         THEN
            l_znesek_val    := -p_znesek;
         ELSIF l_trans_ind = 'O'
         THEN
            l_znesek_val    := p_znesek;
         END IF;
      END IF;

      r_dan_trr.dan_id               := p_dan_id;
      r_dan.dan_id                   := p_dan_id;
      r_dan_trr.racun_id             := p_racun_id;
      r_dan.racun_id                 := p_racun_id;
      r_dan_trr.valuta_id            := p_valuta_id;
      r_dan.valuta_id                := p_valuta_id;

      SELECT zadnji_status_id
        INTO r_dan_trr.status_id
        FROM dim_racun_trr
       WHERE racun_id = p_racun_id;

      r_dan.status_id                := r_dan_trr.status_id;

      SELECT produkt_id
        INTO r_dan_trr.produkt_id
        FROM dim_racun_trr
       WHERE racun_id = p_racun_id;

      r_dan.produkt_id               := r_dan_trr.produkt_id;

      SELECT zadnja_pe_id
        INTO r_dan_trr.pe_id
        FROM dim_racun_trr
       WHERE racun_id = p_racun_id;

      r_dan.pe_id                    := r_dan_trr.pe_id;

      -- poiscem zadnje stanje za ta račun, za dan ki je še manjsi ali enak od dana
transakcije
      SELECT dnevno_stanje, dnevno_stanje_val, dan_id
        INTO l_dnevno_stanje, l_dnevno_stanje_val, l_dan_id
        FROM f_fo_stanje_dan_trr
       WHERE racun_id = p_racun_id
         AND valuta_id = p_valuta_id
         AND dan_id =
                (SELECT MAX (dan_id)
                   FROM f_fo_stanje_dan_trr
                  WHERE racun_id = p_racun_id
                    AND valuta_id = p_valuta_id
                    AND dan_id <= p_dan_id);

      l_znesek                       :=
                           val2eur (p_dan_id, p_valuta_id, l_znesek_val);
      r_dan_trr.dnevno_stanje_val    := l_dnevno_stanje_val + l_znesek_val;
      r_dan.dnevno_stanje_val        := r_dan_trr.dnevno_stanje_val;
      r_dan_trr.dnevno_stanje        := l_dnevno_stanje + l_znesek;
      r_dan.dnevno_stanje            := r_dan_trr.dnevno_stanje;
      r_dan_trr.st_transakcij        := 1;
      r_dan_trr.upo_kre              := USER;
      r_dan_trr.dat_kre              := SYSDATE;
      r_dan.upo_kre                  := USER;
      r_dan.dat_kre                  := SYSDATE;

      BEGIN
         INSERT INTO f_fo_stanje_dan_trr
              VALUES r_dan_trr;
```

```
         INSERT INTO f_fo_stanje_dan
              VALUES r_dan;

         -- update se sveh ostalih zapisov, katerih datum je vecji od datuma prispele
transakcije
         FOR r_rec IN (SELECT *
                         FROM f_fo_stanje_dan_trr
                        WHERE racun_id = p_racun_id
                          AND valuta_id = p_valuta_id
                          AND dan_id > p_dan_id)
         LOOP
            UPDATE f_fo_stanje_dan_trr
               SET dnevno_stanje_val = dnevno_stanje_val + l_znesek_val,
                   dnevno_stanje = dnevno_stanje + l_znesek
             WHERE racun_id = r_rec.racun_id
               AND valuta_id = r_rec.valuta_id
               AND dan_id = r_rec.dan_id;

            UPDATE f_fo_stanje_dan
               SET dnevno_stanje_val = dnevno_stanje_val + l_znesek_val,
                   dnevno_stanje = dnevno_stanje + l_znesek
             WHERE racun_id = p_racun_id
               AND valuta_id = p_valuta_id
               AND dan_id > p_dan_id;
         END LOOP;
      EXCEPTION
         WHEN DUP_VAL_ON_INDEX
         THEN
            -- samo za dan_id(ker imamo trasnakcije za posodobit pri stanje_dan_trr tabeli)
            UPDATE f_fo_stanje_dan_trr
               SET dnevno_stanje_val = dnevno_stanje_val + l_znesek_val,
                   dnevno_stanje = dnevno_stanje + l_znesek,
                   st_transakcij = st_transakcij + l_st_transakcij
             WHERE racun_id = p_racun_id
               AND valuta_id = p_valuta_id
               AND dan_id = p_dan_id;

            UPDATE f_fo_stanje_dan
               SET dnevno_stanje_val = dnevno_stanje_val + l_znesek_val,
                   dnevno_stanje = dnevno_stanje + l_znesek
             WHERE racun_id = p_racun_id
               AND valuta_id = p_valuta_id
               AND dan_id = p_dan_id;

            -- update se sveh ostalih zapisov, katerih datum je vecji od datuma prispele
transakcije
            FOR r_rec IN (SELECT *
                            FROM f_fo_stanje_dan_trr
                           WHERE racun_id = p_racun_id
                             AND valuta_id = p_valuta_id
                             AND dan_id > p_dan_id)
            LOOP
               UPDATE f_fo_stanje_dan_trr
                  SET dnevno_stanje_val = dnevno_stanje_val + l_znesek_val,
                      dnevno_stanje = dnevno_stanje + l_znesek
                WHERE racun_id = r_rec.racun_id
                  AND valuta_id = r_rec.valuta_id
                  AND dan_id = r_rec.dan_id;

               UPDATE f_fo_stanje_dan
                  SET dnevno_stanje_val = dnevno_stanje_val + l_znesek_val,
                      dnevno_stanje = dnevno_stanje + l_znesek
                WHERE racun_id = p_racun_id
                  AND valuta_id = p_valuta_id
                  AND dan_id = p_dan_id;
            END LOOP;
      END;
   END;
END pkg_f_fo_stanje_dan_trr;
/
```

7.5 Programski paket PKG_LOAD_UTIL

Koda je pisana v programskem jeziku Oracle PL/SQL.

Specifikacija paketa:

```
CREATE OR REPLACE PACKAGE dw_load.pkg_load_util
AS
/******************************************************************************
     name:     pkg_load_util
     purpose:  Paket za avtomatično izvajanje polnjenja PPS.

     revisions:
     ver        date        author           description
     ---------  ----------  ---------------  ------------------------------------
     1.0        25.8.2008    mladen babic    1. created this package.
  ******************************************************************************/

   /**
     Vrača število obdelanih vrstic v določenem koraku postopka.
   */
   FUNCTION st_obdelanih_vrstic_opis (p_st_obdelanih_vrstic IN NUMBER)
      RETURN VARCHAR2;

   FUNCTION st_vrstic_temp_tab (p_procedura_id IN NUMBER)
      RETURN NUMBER;

   PROCEDURE izvedi_postopek (p_postopek_id IN NUMBER);
END pkg_load_util;
/
```

Telo paketa:

```
CREATE OR REPLACE PACKAGE BODY dw_load.pkg_load_util
AS
   FUNCTION st_obdelanih_vrstic_opis (p_st_obdelanih_vrstic IN NUMBER)
      RETURN VARCHAR2
   IS
   BEGIN
      RETURN TO_CHAR (p_st_obdelanih_vrstic) || ' obdelanih vrstic.';
   END;

   FUNCTION st_vrstic_temp_tab (p_procedura_id IN NUMBER)
      RETURN NUMBER
   IS
      l_ret           NUMBER;
      l_temp_tabela   VARCHAR2 (100);
      l_sql           VARCHAR2 (1024);
   BEGIN
      SELECT temp_tabela
        INTO l_temp_tabela
        FROM polnjenje procedura p, polnjenje izvor v
       WHERE p.izvor_id = v.izvor_id AND procedura_id = p_procedura_id;

      BEGIN
         l_sql    := 'select count(*) from ' || l_temp_tabela;

         EXECUTE IMMEDIATE l_sql
                     INTO l_ret;
      EXCEPTION
         WHEN OTHERS
         THEN
            l_ret    := 0;
      END;

      RETURN l_ret;
   END;

   PROCEDURE poslji_email (p_skrbnik_vir_id NUMBER, p_polnjenje_id NUMBER)
   IS
   BEGIN
      -- implementacija pošiljanja emaila skrbniku vira podatkov!
      NULL;
   END;

   PROCEDURE izvedi_postopek (p_postopek_id IN NUMBER)
```

```
IS
   l_polnjenje_id           NUMBER;
   l_st_vrstic_za_obdelat   NUMBER;
   l_st_napak               NUMBER;
   l_skrbnik_id             NUMBER;
   l_sql                    VARCHAR2 (1024);
   l_status_korak           VARCHAR2 (10);
   l_message                VARCHAR2 (255);
   prekini_postopek         EXCEPTION;
BEGIN
   -- DOLOČANJE NOVEGA POLNJENJE_ID-ja
   SELECT seq_polnjenje_id.NEXTVAL
     INTO l_polnjenje_id
     FROM DUAL;

   -- IZVAJANJE POSAMEZNIH KORAKOV POSTOPKA
   FOR r_rec IN (SELECT   POST.postopek_id, kor.st_koraka,
                          kor.procedura_id, proc.procedura_ime,
                          proc.procedura_opis, izv.izvor_id,
                          izv.temp_tabela, izv.izvorna_tabela,
                          izv.skrbnik_id
                     FROM polnjenje_izvor izv,
                          polnjenje_procedura proc,
                          polnjenje_korak kor,
                          polnjenje_postopek POST
                    WHERE POST.postopek_id = p_postopek_id
                      AND POST.postopek_id = kor.postopek_id
                      AND kor.procedura_id = proc.procedura_id
                      AND proc.izvor_id = izv.izvor_id
                      AND kor.aktiven = 'D'
                 ORDER BY kor.st_koraka)
   LOOP
      BEGIN
         l_skrbnik_id              := r_rec.skrbnik_id;
         l_st_vrstic_za_obdelat    :=
                                st_vrstic_temp_tab (r_rec.procedura_id);
         l_sql                     :=
               'BEGIN '
            || r_rec.procedura_ime
            || '(:polnjenje_id, :status, :message); END;';

         -- INSERT  INFO O POLNJENJU
         INSERT INTO polnjenje_info
                     (polnjenje_id, postopek_id,
                      procedura_id, st_koraka,
                      korak_opis, korak_zacetek,
                      st_vrstic_za_obdelat, dat_kre, user_kre
                     )
              VALUES (l_polnjenje_id, r_rec.postopek_id,
                      r_rec.procedura_id, r_rec.st_koraka,
                      r_rec.procedura_opis, SYSDATE,
                      l_st_vrstic_za_obdelat, SYSDATE, USER
                     );

         EXECUTE IMMEDIATE l_sql
                     USING IN     l_polnjenje_id,
                           IN OUT l_status_korak,
                           IN OUT l_message;

         l_st_napak                := st_vrstic_temp_tab (r_rec.procedura_id);
         -- Če ni prišlo do izpada koraka zapišemo
         -- podatke o polnjenju koraka v polnjenje_info
         IF l_status_korak != pkg_error_util.g_status_izpad
         THEN
            IF l_st_napak = 0
            THEN
               l_status_korak    := pkg_error_util.g_status_uspeh;
            ELSE
               l_status_korak    := pkg_error_util.g_status_napake;
            END IF;

            UPDATE polnjenje_info
               SET korak_konec = SYSDATE,
                   korak_status = l_status_korak,
                   korak_sporocilo = l_message,
                   st_napak = l_st_napak
```

```
                WHERE polnjenje_id = l_polnjenje_id;
            ELSE
               -- Če je prišlo do izpada koraka prekinemo postopek
               RAISE prekini_postopek;
            END IF;
         EXCEPTION
            WHEN OTHERS
            THEN
               -- če pride do nenormalnega izpada postopka:
               pkg error util.get oracle napaka (l status korak,
                                                 l_message);
               RAISE prekini_postopek;
         END;
      END LOOP;
   EXCEPTION
      WHEN prekini_postopek
      THEN
         UPDATE polnjenje_info
            SET korak_konec = SYSDATE,
                korak_status = l_status_korak,
                korak_sporocilo = l_message,
                st_napak = -1
          WHERE polnjenje_id = l_polnjenje_id;

         poslji_email (l_skrbnik_id, l_polnjenje_id);
   END;
END pkg_load_util;
/
```

8. Viri

[1] C. Imhoff, “The Corporate Information Factory”, December 1999, DM Review Magazine. Dostopno na: http://www.dmreview.com/issues/19991201/1667-1.html

[2] W.H. Inmon, “Bottom Up Warehouse Development, Alice And The Mad Hatter”. Dostopno na: http://www.inmoncif.com/view/21

[3] W.H. Inmon, *Building the Data Warehouse*, New York: John Wiley & Sons, 1992

[4] R. Kimball, *The Data Warehouse Lifecycle Toolkit*, New York: John Wiley & Sons, 1998, pogl. uvod, 1,5,7

[5] R. Kimball, *The Data Warehouse Toolkit 2nd edition*, New York: John Wiley & Sons, 2002, pogl. 1, 9

[6] R. Kimball, *The Data Warehouse Toolkit 1st edition*, New York: John Wiley & Sons, 1996, pogl. 1, 7

[7] R. Kimball, *The Data Warehouse ETL Toolkit*, New York: John Wiley & Sons, 2004, pogl. uvod, 1

[8] R. Kimball, M. Ross, “Differences of Opinion“, Marec 2004. Dostopno na: http://www.intelligententerprise.com/info_centers/dataWarehousing/showArticle.jhtml?articleID=17800088

[9] (2005) Oracle Database Concepts 10g Release 2, pogl. 16. Dostopno na: http://download.oracle.com/docs/cd/B19306_01/server.102/b14220.pdf

[10] H. J. Watson, “Recent Developments In Data Warehousing”, *Communications of the Association for Information Systems*(Volume 8, 2001) 1-25