UNIVERZA V LJUBLJANI

FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – praktična matematika

Rok Koren

POLNJENJE PODATKOVNIH SKLADIŠČ S POMOČJO ORODJA ORACLE WAREHOUSE BUILDER

Diplomska naloga

Ljubljana, 2010

DIPLOMSKA NALOGA :
FAKULTETA ZA MATEMATIKO IN FIZIKO

ZAHVALA

Iskreno se zahvaljujem svojemu mentorju prof. mag. Matiji Lokarju za vse strokovne nasvete in vsestransko pomoč pri izdelavi diplomske naloge. Zahvalil bi se tudi sodelavcem iz podjetja SRC, kjer sem na praktičnem izobraževanju pridobil ogromno znanja na temo te naloge.

Zahvaljujem se staršem, ki so mi omogočili študij in mi vedno stali ob strani. Hvala tudi prijateljem, ki so me spraševali kdaj bo, saj če me ne bi, verjetno še sedaj ne bi diplomiral.

Na koncu še en velik hvala moji punci Ani, ki me je spodbujala in stala ob strani, ko je bilo najtežje.

DIPLOMSKA NALOGA :
FAKULTETA ZA MATEMATIKO IN FIZIKO

Program diplomske naloge

V diplomski nalogi predstavite osnovne načine, kako lahko načrtujemo podatkovno skladišče in kako z orodjem Oracle Warehouse Builder upravljamo s podatki v podatkovnem skladišču. Pri tem se osredotočite predvsem na zajem ter transformacijo podatkov in polnjenje podatkovnega skladišča s temi podatki (t.i. postopke ETL).

Osnovna literatura:

Ralph Kimball, *The data warehouse toolkit*. 1996, USA: John Wiley & Sons, Inc.

mentor

mag. Matija Lokar

DIPLOMSKA NALOGA :
FAKULTETA ZA MATEMATIKO IN FIZIKO

Povzetek

Glavni poudarek diplomske naloge je na razvoju postopkov za zajem, transformacijo in polnjenje podatkovnega skladišča (ETL postopkov) z orodjem Oracle Warehouse Builder, ki nam omogočajo prenos podatkov v podatkovno skladišče. Da lahko to sploh počnemo, moramo najprej poznati osnove podatkovnega skladiščenja ter različne metode, kako pristopiti k izgradnji podatkovnega skladišča (poglavje 2). Potrebno je tudi znanje o tem, kako se naredi podatkovni model, ki je eden osnovnih gradnikov pri izgradnji podatkovnega skladišča (poglavje 3). Da lahko ETL postopke razvijemo z orodjem Oracle Warehouse Builder, je potrebno dobro poznavanje orodja samega (poglavje 4). Na konkretnem primeru namišljenega podjetja E-trgovina bomo postopke razvili in prikazali, kako se z njimi upravlja (poglavje 5 in 6).

Abstracts

The main focus of the diploma thesis is to develop processes for extracting, transforming and loading data warehouse (ETL processes) by using Oracle Warehouse Builder tool. These processes enable us to transfer data into the data warehouse. In order to do this, we must first know the basics of data warehousing, and various methods on how to build a data warehouse (Chapter 2). It requires the knowledge about the construction of the data model, which is one of the basic building blocks of a data warehouse (Chapter 3). In order to develop ETL processes with Oracle Warehouse Builder tool, we need to have a good understanding of the tool itself (Chapter 4). Using an imaginary company E-trgovina we will develop the processes and show how they are managed (Chapters 5 and 6).

Math. Subj. Class. (2010): 68P15, 68N15, 68P05, 68P10

Ključne besede: Oracle warehouse builder, podatkovno skladišče, podatkovni model, PL/SQL, preslikava, postopek

Keywords: Oracle warehouse builder, data warehouse, data model, PL/SQL, mapping, process flow

KAZALO

KAZALO SLIK

KAZALO TABEL

DIPLOMSKA NALOGA :
FAKULTETA ZA MATEMATIKO IN FIZIKO

SEZNAM KRATIC

- SQL – poizvedovalni jezik za delo z bazami (ang. Structured Query Language)
- PL/SQL – programski jezik (ang. Procedural Language/Structured Query Language)
- PS – podatkovno skladišče (ang. Data warehouse)
- PPS – področno podatkovno skladišče (ang. Data Mart)
- ETL – zajem, transformacija in polnjenje podatkovnega skladišča (ang. Extract, Transform, Load)

DIPLOMSKA NALOGA :
FAKULTETA ZA MATEMATIKO IN FIZIKO

1 Uvod

Večina podjetij se dandanes srečuje z velikimi poslovnimi izzivi. Da bi bila kar se da konkurenčna in bi se znala pravilno odločati ter načrtovati, potrebujejo dober sistem, ki omogoča analizo podatkov o poslovanju podjetja in analizo sprejetih odločitev. Za realizacijo takega sistema se pogosto uporablja tako imenovano podatkovno skladišče (natančneje opisano v razdelku 2.1). Podatkovno skladišče se redno osvežuje s postopki ETL (natančneje opisani v poglavju 5), ki zagotavljajo kakovost podatkov. Tako podatkovna skladišča vedno vsebujejo ažurirane podatke.

V tej diplomski nalogi bi predvsem radi predstavili načrtovanje podatkovnega skladišča, postopke polnjenja (ETL), s katerimi polnimo podatkovno skladišče in orodje Oracle Warehouse Bilder (natančneje opisano v poglavju 4). Z njim bomo razvili postopke za polnjenje podatkovnega skladišča. Kot primer bomo vzeli izmišljeno podjetje E-trgovina in na gradnji podatkovnega skladišča zanj predstavili prej omenjene postopke.

Za začetek bomo podali nekaj osnovnih definicij s področja podatkovnega skladiščenja. Pri načrtovanju le-tega bomo uporabili Kimballov pristop. Opisali bomo tudi Inmonov pristop in ga primerjali s Kimballovim ter prikazali prednosti in slabosti enega in drugega. V drugem delu naloge bomo predstavili dimenzijski model za naše izmišljeno podjetje. V tretjem delu pa bomo predstavili razvoj in nadzor nad ETL postopki za podjetje E-trgovina s pomočjo orodja Oracle Warehouse Builder.

2 Podatkovno skladiščenje

2.1 Kaj je podatkovno skladišče?

Kot za večino kompleksnejših pojmov, tudi za izraz podatkovno skladišče ne obstaja enotna definicija. Začeli bomo z eno najbolj znanih definicij, ki jo je podal Ralph Kimball, saj bomo v nalogi uporabili njegov pristop.

Podatkovno skladišče je kopija transakcijskih podatkov posebej strukturiranih za poizvedbo in analizo. [1]

Kot transakcijske podatke tu mislimo na podatke, ki jih razna podjetja oziroma ustanove ustvarijo z njihovim delovanjem. Na primer, ko gremo dvigniti denar iz bankomata, se na nekem transakcijskem sistemu banke zabeležijo podatki o dvigu denarja. Te podatke prenesemo v podatkovno skladišče. Podatke, ki so v tem skladišču, potem obdelujemo, iz njih črpamo različne informacije, delamo analize in podobno.

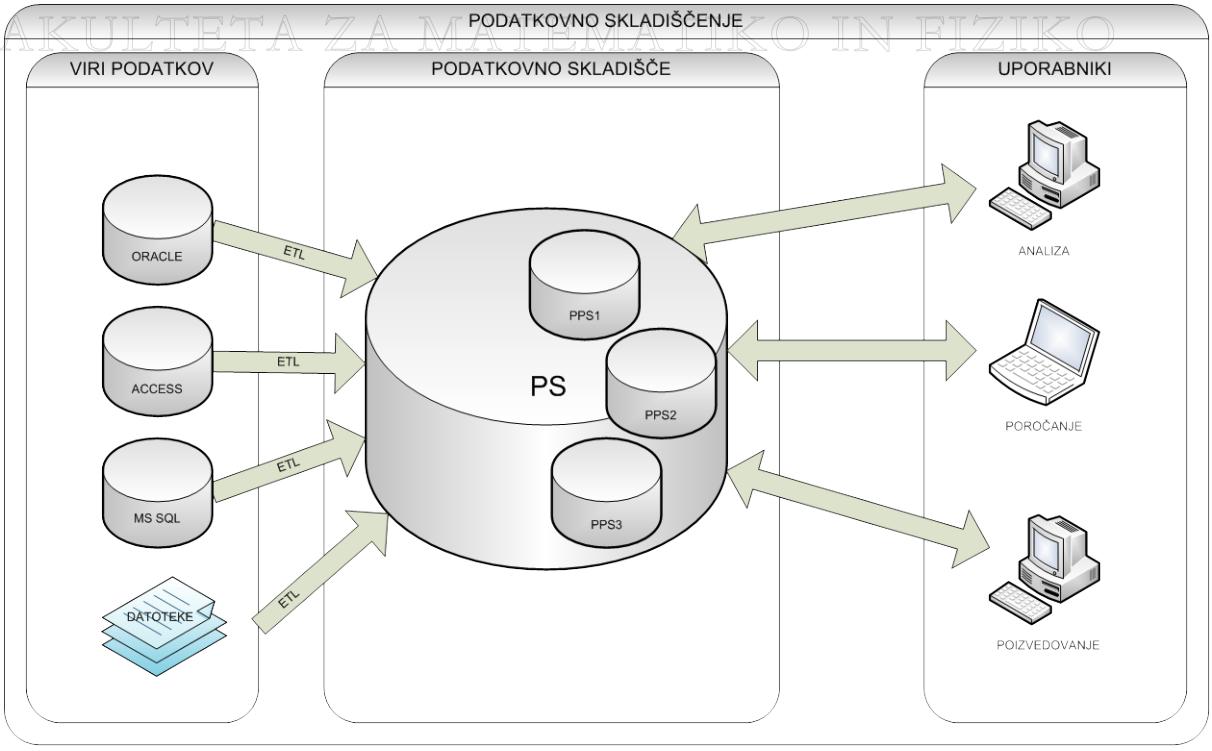
Z drugimi besedami bi lahko rekli, da je podatkovno skladišče kraj, kjer lahko ljudje dostopajo do podatkov.

Podatkovno skladišče je pravzaprav baza podatkov, ampak njegov namen je drugačen kot pri klasični podatkovni bazi. Osnovna razlika je v tem, da je podatkovna baza (mišljena kot del transakcijskega sistema) namenjena tekočemu izvajanju posla, torej operativnim podatkom, podatkovno skladišče pa temu, da vemo, kako posel izvajati in načrtovati. Že samo ime podatkovno skladišče nam pove, da v njem hranimo tudi starejše podatke. V transakcijski podatkovni bazi starih podatkov praviloma nimamo, ker bi jo s tem samo obremenjevali.

Za razliko od podatkovnega skladišča, ki predstavlja zbirko podatkov, je podatkovno skladiščenje (glej Slika 1) izraz za celoten proces dela s podatki. Prvi del tega procesa je sámo zbiranje podatkov (polnjenje skladišča), ki ga označujemo s kratico ETL (Extract Transform Load – glej razdelek 5). Sem sodi zajem, prečiščevanje, prilagajanje in dostava podatkov v podatkovno skladišče. Drugi del procesa je poizvedovanje, analiza in poročanje nad temi podatki. To je Ralph Kimball lepo strnil z naslednjimi besedami:

Podatkovno skladiščenje je sistem, ki zajame, prečisti, prilagodi in dostavi izvorne podatke v dimenzionalno podatkovno skladišče in nato podpira in izvaja poizvedbe in analize za namene odločanja. [2]

[1] Kimball, R., The Data Warehouse Toolkit. 1996, USA: John Wiley & Sons, Inc. str. 310



Slika 1 - Podatkovno skladiščenje

2.2 Zakaj imeti podatkovno skladišče?

Glavni cilj podatkovnega skladišča se je razvil iz potrebe po dostopu do velike količine neizkoriščenih podatkov, predvsem podatkov v velikih organizacijah. V poslovnem procesu je potrebno videti podatke iz različnih zornih kotov (npr. najprej kateri izdelek se je najbolje prodajal v nekem obdobju, potem pa še kateri izdelek se je najbolje prodajal glede na določen kraj …). Vsi podatki v velikem sistemu tudi niso nujno kvalitetni (npr. pri ročnem vnosu v nek sistem pride do napačnega datuma (05. 07. 0200 namesto 05. 07. 2000)). Ker so se baze hitro polnile, so se zaradi brisanja izgubljali podatki za starejša obdobja. Zaradi izvajanja poizvedb in poročanja neposredno iz transakcijskih sistemov, so slednji delovali počasneje. Želja rešiti naštete probleme nas privede do osnovnih razlogov, zakaj imeti podatkovno skladišče:

- Izvajanje analize, poizvedb in poročanja na strežnikih, ki niso del transakcijskega sistema. S tem ne obremenjujemo transakcijskega sistema, ki mora v sprejemljivem času opraviti določene operacije povezane z zapisovanjem transakcij.
- Zagotavljanje konsistentnih in kvalitetnih podatkov. Pri prenosu podatkov v podatkovno skladišče se le-ti »očistijo« in preverijo, če so dovolj dobri.

[2] Kimball, R., The data warehouse ETL toolkit. 1998, USA: Wiley Publishing, Inc. str 23

- Hranjenje podatkov tudi za daljša časovna obdobja. Zaradi hitrejšega delovanja transakcijskega sistema so starejši podatki dostikrat odstranjeni. V podatkovnem skladišču pa jih lahko brez težav hranimo.
- Preprečevanje dostopa do transakcijskih sistemov osebam, ki potrebujejo dostop do transakcijskih podatkov samo za namene poročanja in poizvedovanja.
- Zagotavljanje hitrega izvajanja poizvedb in poročil. Tehnologija, na kateri so zasnovana podatkovna skladišča, je prilagojena poizvedovanju in izdelovanju poročil.
- Zagotavljanje zaščite podatkov, kar pomeni omogočanje vpogleda v podatke le pooblaščenim osebam.

2.3 Zgodovina/razvoj podatkovnih skladišč

Prvi koraki na področju razvoja podatkovnih skladišč so se zgodili okrog leta 1960, ko sta General Mills in Dartmouth College v skupni raziskavi izrazoma »dimenzija« in »dejstvo« (ang. »dimension« in »fact«) dodala nov pomen (razlaga teh dveh pojmov sledi v razdelku 3.1). Leta 1970 sta podjetji ACNielsen in IRI predstavili sistem za informacijsko podporo prodaje, ki je vseboval določene značilnosti sistemov, ki jim danes pravimo področna podatkovna skladišča. Leta 1988 sta Barry Devlin in Paul Murphy objavila članek *An architecture for a business and information systems* v reviji *IBM System Journal*, kjer se je prvič pojavil termin »business data warehouse«. Pomemben korak pri razvoju podatkovnega skladišča je naredilo podjetje Red Brick Systems, ki je leta 1990 razvilo sistem za delo z bazami podatkov imenovano *Red Brick Warehouse*. Ta sistem je bil specializiran za izvedbo podatkovnih skladišč. Leta 1991 je podjetje Prism predstavilo programsko opremo »Prism Warehouse Manager« za izgradnjo podatkovnih skladišč. Istega leta je bila objavljena knjiga Billa Inmona – *Building the Data Warehouse*, ki velja za prvo knjigo posvečeno podatkovnemu skladiščenju. Leta 1995 je bila ustanovljena organizacija The Data Warehouse Institute. Glavni namen te organizacije je bilo promoviranje podatkovnih skladišč. Leto kasneje je Ralph Kimball objavil knjigo *The Data Warehouse Toolkit*, v kateri je strnjeno veliko stvari v vezi s podatkovnim skladiščenjem. V njej je predstavljen eden najbolj znanih pristopov, kako zgraditi podatkovno skladišče.

2.4 Osnovni elementi podatkovnega skladiščenja

Zaradi lažjega razumevanja celotnega procesa podatkovnega skladiščenja, si bomo v tem razdelku ogledali, kateri so osnovni elementi podatkovnega skladiščenja. Za vsakega bomo

navedli njegovo osnovno značilnost in nekaj praktičnih primerov. Pri tem nam ne bo toliko šlo za točne definicije, temveč bolj za praktično ilustracijo teh elementov.

2.4.1 Izvorni sistemi

To so viri podatkov za podatkovno skladišče. Ti sistemi so lahko različne baze podatkov, razne datoteke in drugi sistemi za upravljanje z informacijami. Pomembno je, da so ti sistemi dostopni v času, ko prenašamo podatke v podatkovno skladišče. Običajno nam dajejo neko trenutno sliko stanja podatkov v določenem obdobju. Podatki na teh virih so navadno zajeti avtomatsko, zato lahko pride do napak. Podatki so v različnih oblikah, strukturirani na razne načine, včasih se lahko celo ponavljajo. Zato je dobro, da podatke pred prenosom v podatkovno skladišče uredimo in prečistimo.

Primeri takih izvornih sistemov so npr. razne baze podatkov (ki jih nadziramo z različnimi sistemi: Oracle, MS SQL Server, MySQL, DB2 …), datoteke (datoteke dostopov, sprememb …, preglednice, razne tekstovne datoteke, ki jih ustvarijo različni sistemi …) in podobno.

2.4.2 Področje priprave informacij

Področje priprave informaciji je namenjeno temu, da podatke prečistimo, preoblikujemo, kombiniramo, združimo in arhiviramo. Podatke pripravimo v taki meri, da so primerni za prenos v podatkovno skladišče. Dokler so podatki v tem območju, jih uporabniki ne morejo prikazovati in poizvedovati po njih. Področje priprave informacij se nahaja med izvornimi sistemi in predstavitvenim strežnikom.

To področje služi tudi, da z ustreznimi postopki poskrbimo za poenotenje različno zapisanih istih podatkov. Če je npr. v enem viru kraj napisan kot Zg. Besnica v drugem pa kot Zgornja Besnica, poskrbimo za enoten zapis. Prav tako v tem območju preverimo, če so podatki smiselni, npr. če določen datum res obstaja. Če v podatkovnem skladišču hranimo le podatke o številu izdelkov narejenih v določenem dnevu, izvorni podatki pa vsebujejo informacije o urni proizvodnji, potem v področju priprave informacij seštejemo ustrezne podatke.

2.4.3 Predstavitveni strežnik

Predstavitveni strežnik je strojna oprema, na kateri so v podatkovnem skladišču shranjeni podatki. Ti so pripravljeni v taki obliki, da omogočajo direktno poizvedovanje končnih uporabnikov, poročanje in analiziranje.

2.4.4 Poslovni procesi

Poslovni procesi so povezani nizi poslovnih dejavnosti, ki dajejo smisel podatkovnemu skladišču. Konkretneje so to večinoma neki urejeni postopki pretoka podatkov, katerih pravila določi stranka oziroma uporabniki podatkovnega skladišča.

Primer poslovnega procesa je npr. ETL postopek, ki pripravi izvorne podatke o prodaji izdelkov v podjetju E-trgovina za podatkovno skladišče. S procesom podatke pripravimo tako, da so primerni za poročanje in planiranje. Pravila, kako naj bodo podatki urejeni, pa določi stranka.

2.4.5 Področna podatkovna skladišča

V podatkovnem skladišču hranimo množico podatkov, ki pripadajo različnim poslovnim procesom. Ker je osnova za odločanje običajno posamezni poslovni proces, so podatkovna skladišča praviloma razdeljena na področna podatkovna skladišča. To so deli celotnega skladišča. En poslovni proces običajno pripada enemu področnemu podatkovnemu skladišču.

V praksi bi lahko neka organizacija imela področna podatkovna skladišča za področje kadrov, investicij, nabav, financ …

2.4.6 Podatkovno skladišče

Podatkovno skladišče je baza podatkov, ki je osnovana tako, da je namenjena poizvedovanju, poročanju in analizi nad podatki. Podatke v podatkovnem skladišču dobimo iz področja za pripravo podatkov. Podatkovno skladišče se tudi sprotno osvežuje s kontroliranimi polnjenji, ki nadzirajo pravilnost in kakovost podatkov. Kadar imamo več področnih podatkovnih skladišč, je podatkovno skladišče združitev več PPS skupaj.

2.4.7 Aplikacije za dostop končnih uporabnikov do podatkov

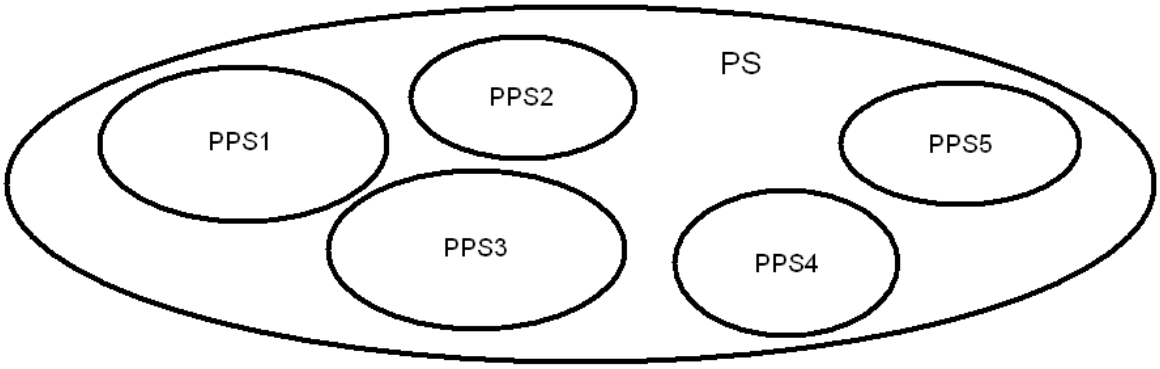
To so zbirke orodij, s katerimi omogočimo poizvedovanje, analiziranje, planiranje in poročanje nad podatki iz podatkovnega skladišča. Za povezavo med podatkovnim skladiščem in aplikacijami skrbi odjemalec (ang. client). Te aplikacije so narejene tako, da so prijazne uporabniku. Le-ta lahko z nekaj kliki pride do želenih podatkov. Ti podatki so lahko prikazani v obliki tabele, grafa, poročila … V ozadju aplikacije se izvajajo razne poizvedbe, zapisane v jeziku SQL, s katerimi dobimo želene podatke. Z uporabo aplikacij se izognemo potrebi, da bi uporabniki za dostop do podatkov morali poznati poizvedovalni jezik SQL.

2.5 Pristopi k organizaciji podatkovnih skladišč

Poznamo več različnih pristopov kako razviti podatkovno skladišče. Dva najbolj znana sta Kimballov in Inmonov. Končni cilj obeh je sestaviti celovito podatkovno skladišče (torej tako, ki pokriva celotno delovanje podjetja), zato imata oba kar nekaj skupnih točk:

- V podatkovnem skladišču morajo biti kvalitetni podatki, ki so lahko dostopni.
- Pri načrtovanju arhitekture podatkovnega skladišča je dobro zajeti organizacijo kot celoto.
- Podatki v podatkovnem skladišču morajo biti redno osveženi in dostopni v času, ko jih uporabniki rabijo.

Da bomo lažje razumeli osnovno razliko med sledečima pristopoma, je dobro poznati razmerje med podatkovnim skladiščem in področnim podatkovnim skladiščem. Matematično bi si razmerje lahko predstavljali s pomočjo množic (glej Slika 2). Pri tem so področna podatkovna skladišča podmnožice podatkovnega skladišča.

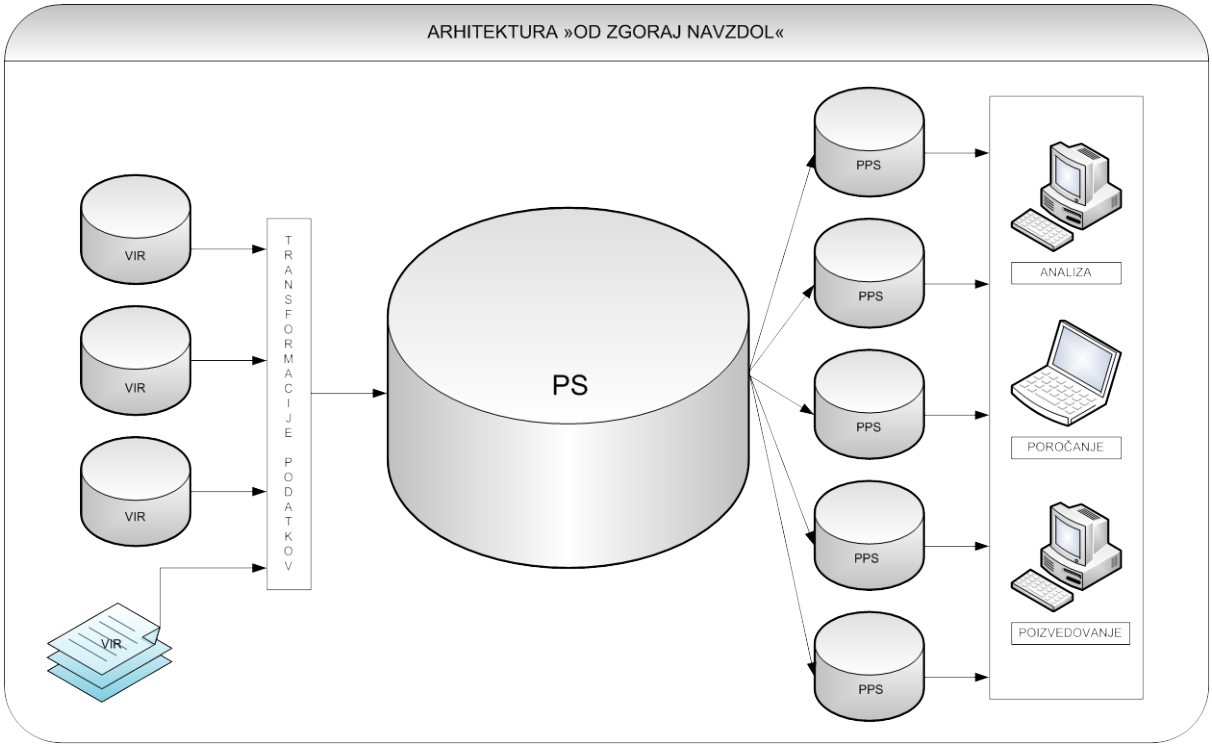


Slika 2 - Unija PPS, ki predstavljajo celovito PS

2.5.1 Inmonov pristop

Kot prvi se je pojavil Inmonov pristop, ki temelji na tako imenovani arhitekturi »od zgoraj navzdol«. Začne se z izgradnjo celovitega podatkovnega skladišča. Ko imamo podatke v tem celovitem podatkovnem skladišču, iz njega napolnimo področna podatkovna skladišča. Pri Inmonovem pristopu torej uporabljamo usklajen zajem podatkov iz izvornih sistemov. V območju priprave podatkov podatke ustrezno obdelamo in jih pripeljemo do podatkovnega skladišča. Ker so vsi podatki zbrani na enem mestu, je nato relativno enostavno napolniti

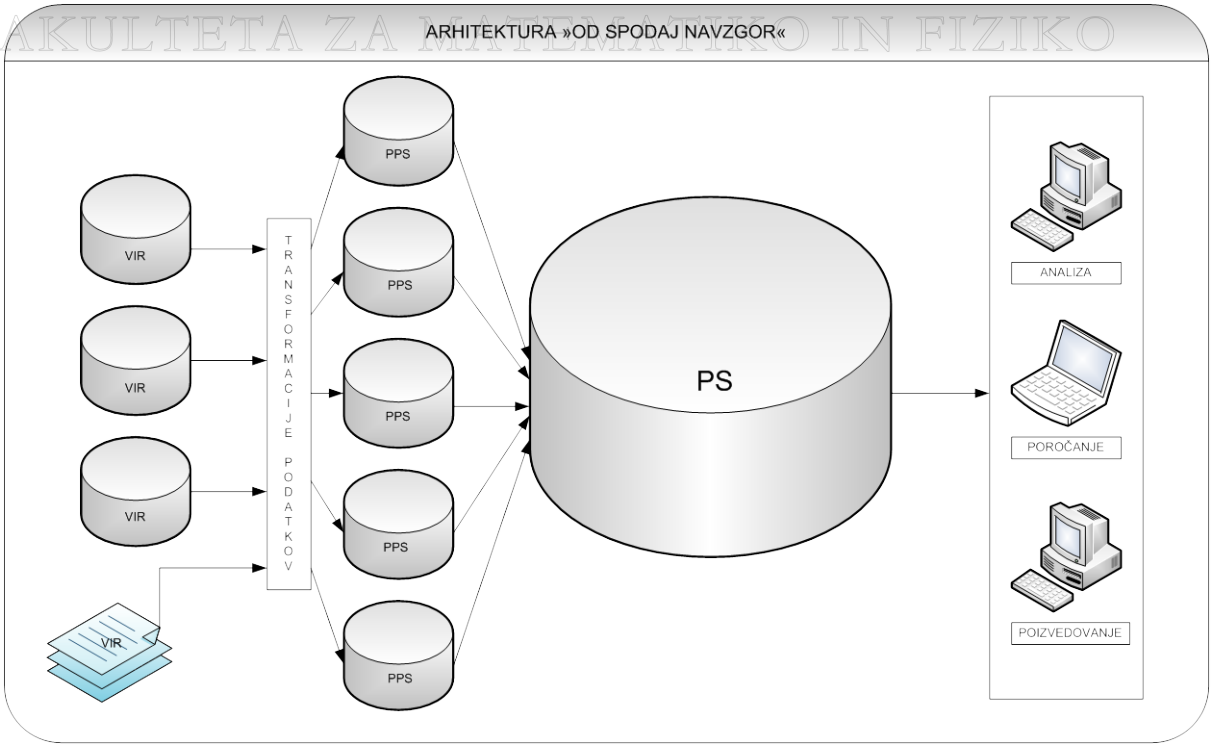
področna podatkovna skladišča. Inmonov pristop uporablja normaliziran način hranjenja podatkov v podatkovnem skladišču, kar pomeni, da ustrezna podatkovna baza zadošča tretji normalni formi. To pomeni, da neko osnovno tabelo razbijemo na eno ali več tabel, v katere prenesemo atribute, ki so odvisni od ne-ključnega atributa in le-tega postavimo za tuji ključ v osnovni tabeli. S tem si zagotovimo konsistentnost podatkov in se izognemo podvajanju le-teh. Zaradi celovitega razvoja je običajno ta način izgradnje podatkovnega skladišča dražji in časovno daljši.



Slika 3 - Arhitektura "od zgoraj navzdol"

2.5.2 Kimballov pristop

Drugi znani pristop k načrtovanju podatkovnega skladišča je Kimballov. Tu podatkovno skladišče gradimo »od spodaj navzgor«. Ta nam omogoča razvoj delnega podatkovnega skladišča, ki ga naknadno lahko tudi razširimo. Da ne bi imeli težav pri morebitni širitvi, je pomembno, da imamo pri načrtovanju podatkovnega skladišča v mislih pričakovano končno podatkovno skladišče, ki bo celovito pokrivalo vse poslovne procese. Kimball imenuje ta pristop tudi »Data Warehouse Bus Arhitecture«. Ta tako imenovana »Bus Arhitecture« vsebuje vse skupne elemente, ki jih uporabljamo v področnih podatkovnih skladiščih. Ti elementi so skladne dimenzije in merljiva dejstva (natančneje opisana v poglavju 3.1).



Slika 4 - Arhitektura "od spodaj navzgor"

2.6 Glavne razlike med Kimbalovim in Inmonovim pristopom

Na kratko si oglejmo, kakšne so glavne razlike med obema pristopoma.

KIMBALL	INMON
Postopen razvoj podatkovnega skladišča. Podatkovno skladišče razvijamo po korakih, pokrivamo en poslovni proces za drugim.	**Celovit razvoj podatkovnega skladišča.** Takoj načrtujemo podatkovno skladišče, ki pokriva vse poslovne procese podjetja.
De-normaliziran podatkovni model. Podatki so v podatkovni bazi, ki običajno zadošča drugi normalni formi. Obstajajo tudi primeri, ko imamo le delno normalizirano podatkovno bazo.	**Normaliziran podatkovni model.** Podatkovna baza mora biti v tretji normalni formi. To pomeni, da neko osnovno tabelo razbijemo na eno ali več tabel, v katere prenesemo atribute, ki so odvisni od ne-ključnega atributa in le-tega postavimo za tuji ključ v osnovni tabeli.
Cenejša realizacija. Delni razvoj omogoča vlaganje po korakih in potrebah. Če Kimballov razvoj na koncu zajame celotno rešitev, je cena najverjetneje enaka, kot pri Inmonovem.	**Dražja realizacija.** Zaradi celovite izgradnje podatkovnega skladišča je izgradnja dražja, še posebej, če za določene poslovne procese podjetje pravzaprav sploh ne potrebuje podatkovnega skadišča.
Hitreje realiziran. Ker podatkovno skladišče gradimo po področjih in včasih podjetje sploh ne	**Počasneje realiziran.** V vsakem primeru moramo zgraditi celovito podatkovno skladišče.

potrebuje celovite rešitve, hitreje razvijemo potrebno.	
Hitrejše poizvedbe. Poizvedbe so praviloma izvedena hitrejše. Razlog je uporaba dimenzijskega modela, ki je zasnovan tako, da optimizira poizvedovanje nad podatki.	**Počasnejše poizvedbe.** Podatkovni model, ki ga uporabljamo pri Inmonovem pristopu, ustvari podatkovno bazo, kjer se poizvedbe ne izvajajo optimalno.
S časom lažje pride do nekonsistentnosti podatkov. Ker podatki niso normalizirani, se isti podatki lahko nahajajo na več koncih. Zato lažje pride do tega, da ob popravkih kakšnega pozabimo popraviti.	**Lažje je ves čas zagotoviti konsistentnost podatkov.** Zaradi uporabe tretje normalne forme se podatki pojavljajo samo enkrat in tako skoraj ne more priti do razhajanj.

Tabela 1 - Razlike med Kimballovim in Inmonovim pristopom

3 Podatkovni model

Prvi korak pri načrtovanju podatkovnega skladišča je oblikovanje podatkovnega modela. Model določa logično strukturo podatkovnega skladišča in nam omogoča lažjo predstavo, kako so podatki med seboj povezani. Medsebojno odvisnost podatkov nam določajo tako imenovane dimenzije. Pri gradnji podatkovnega skladišča zelo hitro pridemo do štiri, pet ali celo večdimenzionalnosti. To pomeni, da npr. prodajo določajo štiri, pet ali več različnih lastnosti (npr. vrsta in barva izdelka, čas in kraj prodaje …). Ker je večdimenzionalnost zelo abstraktna, jo je dobro imeti vizualizirano. Ravno to nam ponudi podatkovni model.

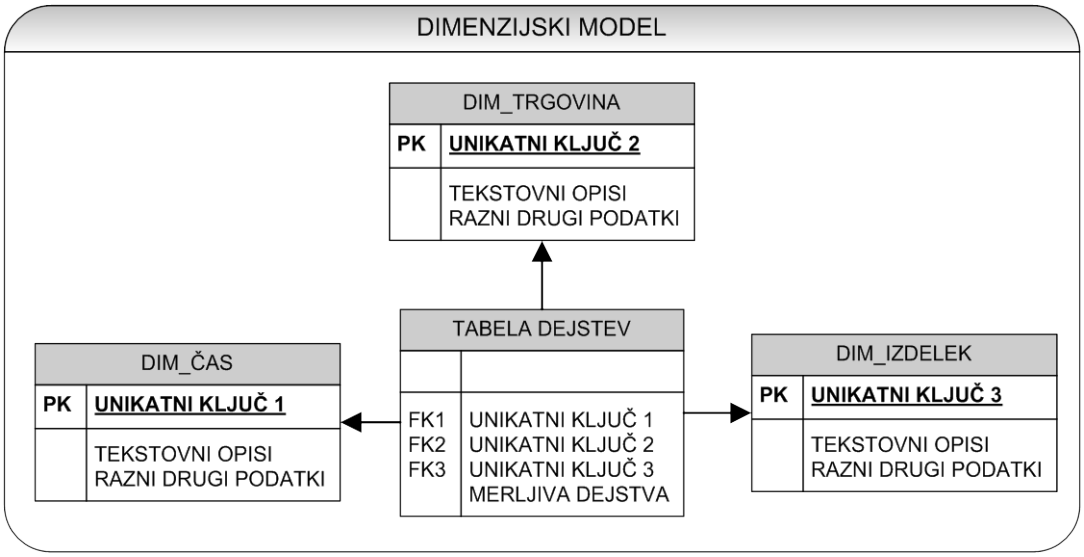
Ko zberemo vse poslovne zahteve (pogovor s stranko o tem, kaj želi) bodočih uporabnikov podatkovnega skladišča in naredimo analizo obstoječih podatkovnih virov ter dokumentacije, smo pripravljeni, da začnemo z razvojem podatkovnega modela za podatkovno skladišče. Lahko rečemo, da je ta model temeljni kamen za izgradnjo podatkovnega skladišča. Iz modela ocenimo končen obseg dela za izgradnjo podatkovnega skladišča in to, kakšne so potrebe podatkovne baze (npr. kakšno strojno opremo bomo potrebovali …).

3.1 Dimenzijski model

Dimenzijsko modeliranje je izraz za tehniko logičnega načrtovanja, ki skuša predstaviti podatke v preprosti obliki. Ta tehnika je pogosto uporabljena za izgradnjo podatkovnih skladišč, ker je njen rezultat podatkovna baza, ki optimizira (pohitri) poizvedovanje. Zato jo

bomo uporabili tudi na našem primeru pri izgradnji podatkovnega skladišča za E-trgovino (glej razdelek 3.2).

Vsak dimenzijski model je sestavljen iz ene večje tabele, imenovane tabela dejstev, in iz več manjših tabel, imenovanih dimenzijske tabele. V prenesenem pomenu bi lahko dejstva in dimenzije poimenovali srce podatkovnega modela in tudi podatkovnega skladišča. To so tabele v bazi, v katerih so spravljeni razni podatki. Tabela dejstev je vir podatkov za končne uporabnike podatkovnega skladišča. V njej se hranijo merljiva dejstva[3] določenega poslovnega procesa in povezave na dimenzijske tabele. Dimenzijske tabele so atributi, ki dajejo pomen dejstvom v tabeli dejstev. Dimenzija je lahko tudi več-nivojska, kar pomeni, da si podatke lahko ogledamo na različnih nivojih (npr. časovna dimenzija bi lahko hranila podatke na letnem, mesečnem, dnevnem nivoju ...). Vsaka tabela dejstev vsebuje večdelni ključ, preko katerega se povezujemo na dimenzijske tabele. Te pa imajo definirane enodelne primarne ključe, ki ustrezajo točno eni komponenti večdelnega ključa v tabeli dejstev. Tabela dejstev vsebuje tudi tako imenovane tuje ključe. Z njimi se zavarujemo, da ne pride do tega, da podatek v tabeli dejstev obstaja, v dimenzijskih tabelah pa ne. Povezave med tabelo dejstev in dimenzijami nam dajejo sliko podobno zvezdi. Dimenzijske tabele predstavljajo krake zvezde, tabela dejstev pa center le-te. Zaradi te oblike se je modela prijelo ime zvezdna shema.



Slika 5 - Dimenzijski podatkovni model

[3] Merljiva dejstva so numerične vrednosti. Primer bi bila cena nekega izdelka ali pa količina določenega izdelka, ki ga neka trgovina ima.

3.2 Dimenzijski podatkovni model za E-trgovino

V tem poglavju bomo sestavili podatkovni model za podjetje E-trgovina. Izvorni podatki so izmišljeni in se nahajajo v Oraclovi podatkovni bazi. Podatkovni model bo zajemal le področje prodaje izdelkov (področno podatkovno skladišče »Prodaja«). Naročnik namreč zahteva samo razvoj področnega podatkovnega skladišča za to področje. Če se bo v prihodnosti odločil za nadgradnjo podatkovnega skladišča (npr. področna podatkovna skladišča »Nabava«, »Kadri«, »Investicije« …), mu bo to omogočeno, saj Kimballov pristop to podpira.

Po priporočilih Kimballa[4] bomo do dimenzijskega modela prišli v štirih korakih:

1. določitev poslovnega procesa, ki ga bomo modelirali
2. določitev »zrna« tabele dejstev
3. določitev dimenzij
4. določitev merljivih dejstev

Najprej se odločimo, kateri poslovni proces bomo modelirali. Za primer podjetja E-trgovina si bomo izbrali poslovni proces, ki bo predstavljal prodajo izdelkov. Podatki, ki se zajemajo v tem procesu, se nahajajo na transakcijskem sistemu Oracle, ki zbira, hrani in delno že obdeluje podatke. Ti so v Oraclovih tabelah PRODAJA, ARTIKEL, TRGOVINA, BARVA_IZDELKA, SKUPINE_IZDELKOV in KATEGORIJE_IZDELKOV.

V drugem koraku načrtovanja dimenzijskega modela definiramo »zrno« tabele dejstev, s katerim določimo nivo podrobnosti podatkov. Lahko bi rekli, da en zapis v tabeli dejstev predstavlja »zrno«. Priporočljivo je, da se izbere karseda podroben nivo zapisa podatkov (npr. namesto na mesečnem nivoju, si izberemo dnevni nivo zapisov v tabeli dejstev …), ker si s tem zagotovimo možnost natančnejše analize in poročila. Po pregledu izvornih podatkov se odločimo spremljati prodajo izdelkov na tedenskem nivoju. Zanimalo nas bo še, kateri izdelek je bil prodan, kje in v kateri barvi je bil prodan.

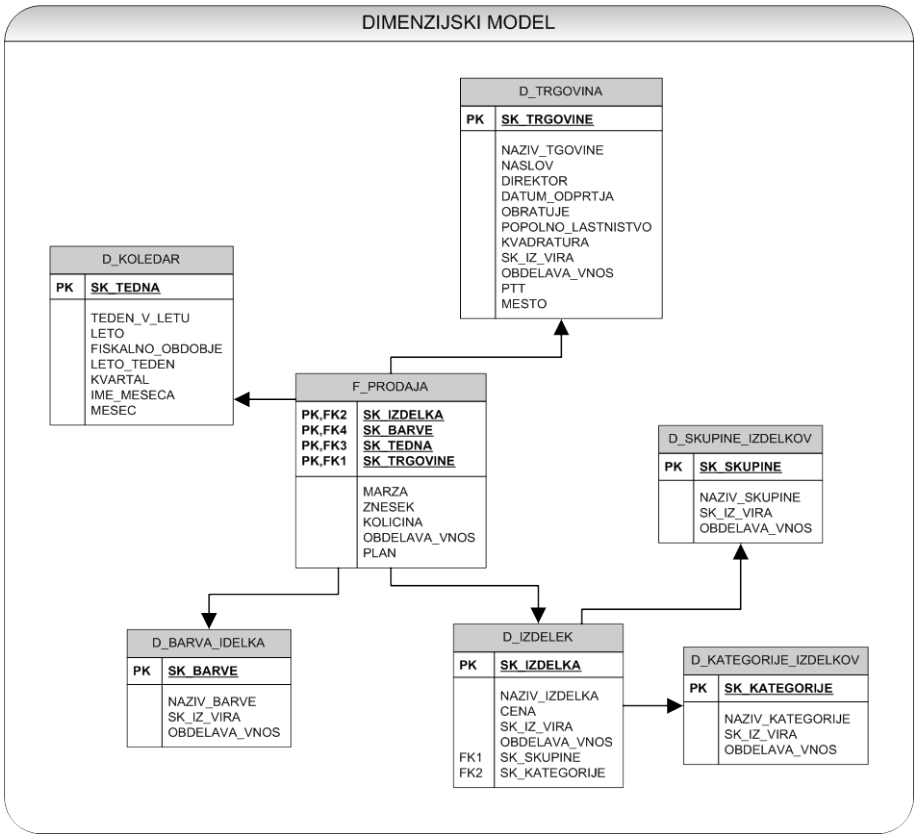
V tretjem koraku moramo opredeliti dimenzije, ki jih bomo potrebovali. Običajno nam že samo »zrno« tabele dejstev opredeli, katere dimenzije potrebujemo. Tipične dimenzije so čas, trgovina, produkt, kupec … V našem primeru E-trgovine potrebujemo dimenzije čas (D_KOLEDAR), trgovina (D_TRGOVINA), barva (D_BARVA), izdelek (D_IZDELEK), kategorije izdelkov (D_KATEGORIJE_IZDELKOV) in skupine izdelkov

[4] Kimball, R., The Data Warehouse Toolkit. 1996, USA: John Wiley & Sons, Inc. str. 194-199

(D_SKUPINE_IZDELKOV). Seveda bi lahko, glede na uporabniške zahteve, dodali še več dimenzij, če so le te na voljo na izvornih sistemih, torej če so na voljo dodatni podatki.

V zadnjem koraku moramo določiti merljiva dejstva, ki morajo ustrezati »zrnu« tabele dejstev. V konkretnem primeru bo to prodaja (F_PRODAJA), v kateri bomo hranili merljiva dejstva kot so cena izdelka, količina in marža.

Po opravljenih štirih korakih, s pomočjo primernega orodja, dobimo sliko (Slika 6) dimenzijskega podatkovnega modela za E-trgovino.



Slika 6 - Dimenzijski podatkovni model za E-trgovino

4 Oracle Warehouse Builder

Oracle Warehouse Builder (v nadaljevanju tudi OWB) je Oraclovo orodje, ki ponuja grafično okolje za izgradnjo, vzdrževanje in upravljanje z ETL postopki. Spomnimo se, da so ETL postopki tisti postopki, ki pripravijo izvorne podatke tako, da so primerni za prenos v podatkovno skladišče.

Osnovni namen orodja je, da omogoča zajem podatkov iz različnih virov in jih pripravi v taki obliki, da so primerni za prenos v podatkovno skladišče. OWB je namenjen tudi modeliranju, zagotavljanju kvalitete podatkov, pregledu izvornih podatkov in nasploh upravljanju z vsemi podatki, ki so nam na voljo. Prvotno je bilo orodje namenjeno analitikom (osebam, ki analizirajo izvorne podatke in določijo pravila za polnjenje teh podatkov v podatkovno skladišče), ker naj bi bilo zelo enostavno za uporabo. Izkazalo se je, da temu ni ravno tako, saj je potrebno kar dobro poznavanje konceptov podatkovnega skladiščenja in programskega jezika PL/SQL. Če na primer pride do težav pri razvoju preslikav, si lahko generiramo PL/SQL kodo in jo ročno poganjamo. Nekateri operatorji v preslikavah zahtevajo, da vnesemo določene PL/SQL in SQL ukaze za pravilno delovanje in podobno.

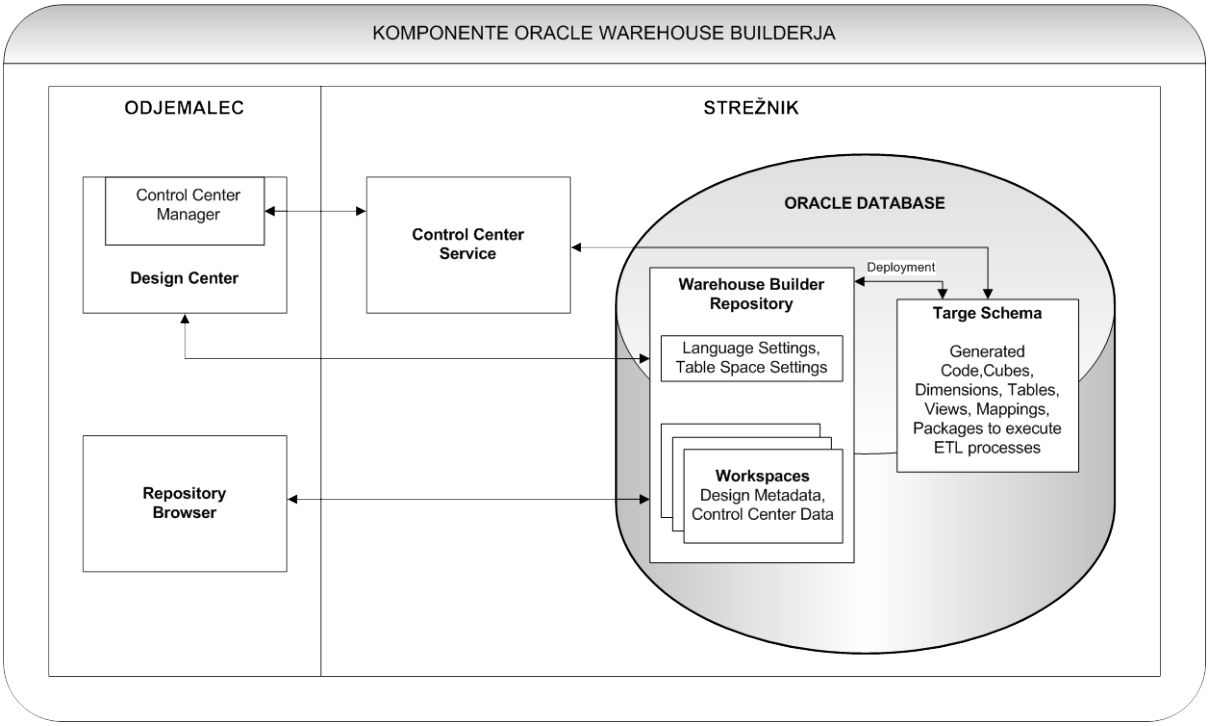
4.1 Osnovni koncept orodja OWB

Oracle Warehouse Builder je sestavljen iz niza grafičnih vmesnikov, ki uporabniku tega orodja pomagajo pri razvoju ETL postopkov. Pri razvoju rešitve se ustvari veliko baznih objektov, ki so metapodatki (torej podatki o drugih podatkih). Ti so shranjeni v shemah[5], ki jih definiramo že pri namestitvi OWB. Temu delu baze pravimo »Warehouse Builder Repository« (WBR), poznanim tudi pod imenom »Workspace«. Navadni uporabniki nimajo popolnega dostopa do tega dela baze. Zato moramo ob zagonu programa Design Center, ki predstavlja glavni grafični uporabniški vmesnik za OWB, opraviti še prijavo, da dobimo dostop do tega področja. WBR uporabljamo za uvoz tistih objektov iz baze (tabele, funkcije, procedure, paketi …), ki jih uporabljamo pri razvoju ETL postopkov. Ti objekti so sestavljeni iz tako imenovanih preslikav (ang. mappings). Preslikava je objekt, v katerem določimo, kako naj se podatki prenesejo iz izvorov v ponore. Ko preslikavo ustvarimo in preverimo njeno korektnost, jo lahko prenesemo v ciljno shemo. Tam se ustvari PL/SQL koda v obliki PL/SQL paketa. Ciljna shema je del podatkovne baze, kjer so shranjene in se izvajajo preslikave, razvite v Design Centru. Za povezavo med razvitimi objekti v Design Centru in ciljno shemo

[5] Shema je v Oraclovem pomenu besede logična struktura (objekt) v bazi za shranjevanje podatkov in metapodatkov.

skrbi Control Center servis. Ta nam omogoča, da v Design Centru razvite objekte, lahko prenesemo na ciljno shemo in jih izvajamo preko Control Centra. Prenos objektov na ciljno shemo pomeni, da se objekt (npr. preslikava, tabela …) shrani v bazo kot neka PL/SQL koda (preslikava kot PL/SQL paket, tabela kot skripta za objekt tabela …).

Orodje je namenjeno tudi nadziranju in izvajanju postopkov. To nam omogočata komponenti Control Center Manager in spletna aplikacija Repository Browser. Control Center je predvsem primeren za prenos objektov na ciljno shemo in ročnemu izvajanju postopkov ter njihov nadzor. Repository Browser pa je bolj uporaben pri pregledovanju avtomatiziranih postopkov. Primernejši je tudi za odkrivanje napak, ki se zgodijo pri njihovem izvajanju.



Slika 7 - Komponente Oracle Warehouse Builderja. Prirejeno po: Oracle warehouse builder user's guide, str. 18

4.2 Razvoj OWB-ja

Orodje Oracle Warehouse Builder je podjetje Oracle predstavilo januarja leta 2000. Takrat je izšla prva verzija, imenovana OWB 2.0.4, ki je ponujala osnovne funkcionalnosti, potrebne pri ETL postopkih. S kasnejšo verzijo 3i se je orodje zelo izboljšalo ravno na področju razvoja ETL postopkov. Leta 2003 so izdali novo verzijo 9i. Ta je že vsebovala sistem za odkrivanje napak v preslikavah (mapiranjih), možnost izgradnje ter spreminjanja postopkov, možnost uporabe stavka »merge«, možnost prenosa podatkov v več tabel hkrati, možnost pregledovanja generirane kode in podobno. Z verzijo 10gR1 je leta 2004 bilo orodje prvič

vključeno v licenco Oraclove podatkovne baze. To je pomenilo, da ga od takrat naprej ni bilo več treba plačati posebej. To je povzročilo, da se je uporaba orodja precej povečala. Nadgradnja verzije 10gR1 na 10gR2 maja 2006 je prinesla veliko novih funkcionalnosti kot so dimenzijsko modeliranje, možnost uporabe tudi drugih sistemov za delo z bazami podatkov (ne le Oraclove), možnost preverjanja obstoja podatkov na izvornih sistemih in možnost naknadne obdelave podatkov. Julija 2007 je izšla verzija 11gR1, ki je dobila še dodatne funkcionalnosti (npr. razširjen izbor povezav na druge baze uporabljene kot izvorni sistemi, osveževanje zapisov v dimenzijah tipa "Slowly Changing Dimension Type 2" in poenostavljena namestitev). Zadnja verzije OWB-ja je v trenutku pisanja diplomske naloge 11gR2 izdana septembra 2009. Ta je verjetno tudi zadnja v tej obliki. Podjetje Oracle se je namreč odločilo, da bo orodje integriralo v drugo orodje imenovano ODI EE (Oracle Data Integrator Enterprise Edition).

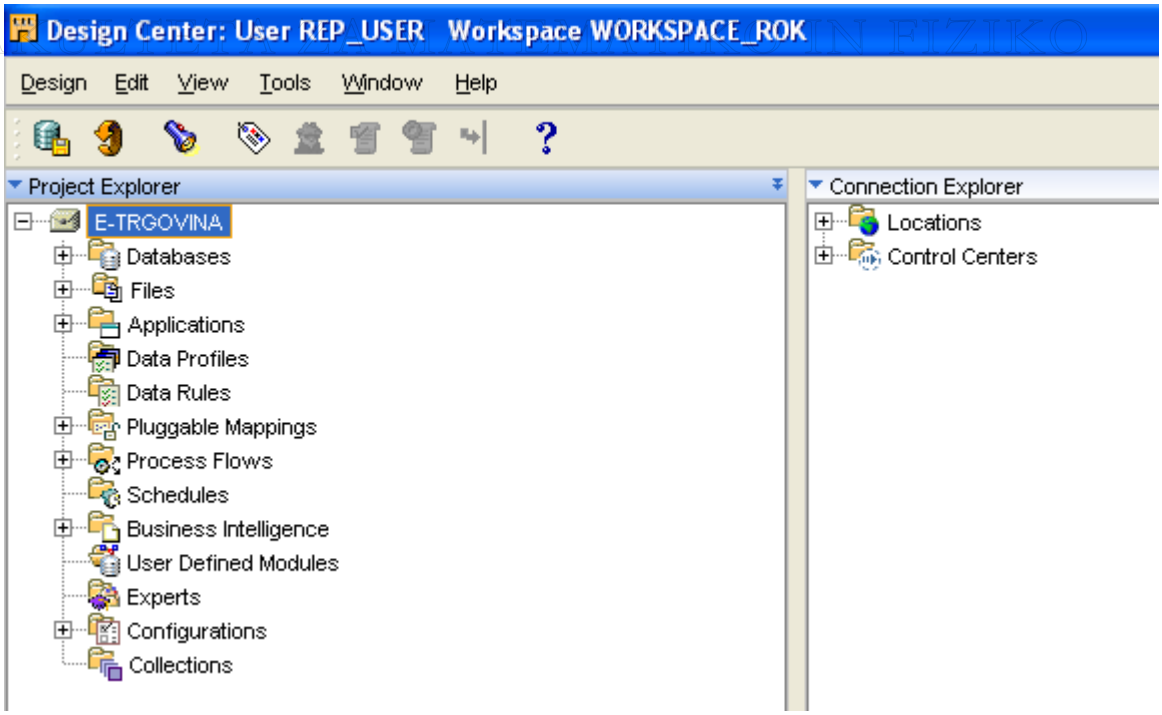
4.3 Predstavitev orodja za namen razvijanja ETL postopkov

Po uspešni prijavi se nam na zaslonu prikaže zbir komponent, ki jih orodje ponuja (glej Slika 8). V levem delu aplikacije »Project Explorer« ustvarimo tako imenovan »projekt« (na sliki se ta imenuje E-TRGOVINA). Ko ga razpremo, najdemo v njem razna področja namenjene razvoju ETL postopkov. Za nas so najbolj bistvena štiri, ki jih bom v nadaljevanju tudi natančneje opisal:

- Databases
- Files
- Process Flows
- Schedules

Tako bomo v področju Databases razvili preslikave in uvozili potrebne objekte iz baze, ki jih bomo v preslikavah potrebovali (tabele, funkcije …). Področja Files v našem primeru sicer ne potrebujemo. Drugače pa tja shranjujemo datoteke, ki nam služijo kot vir ali ponor podatkov. Za organizacijo preslikav razvitih v Databases bomo poskrbeli v področju Process Flows. Preslikave, ki ustrezajo nekemu namenu, tam združimo skupaj v postopke. Na koncu pa bomo poskrbeli, da se bodo postopki izvajali avtomatsko. To bomo uredili v področju Schedules.

V desnem delu aplikacije (»Connection Explorer«) se nahajajo komponente namenjene temu, da povemo kje so objekti, ki jih bomo uporabljali v postopkih oziroma preslikavah (npr. kje se nahajajo datoteke, ki služijo kot vir ali ponor podatkov, podatki za dostop do shem, ki so kreirane v bazi …).



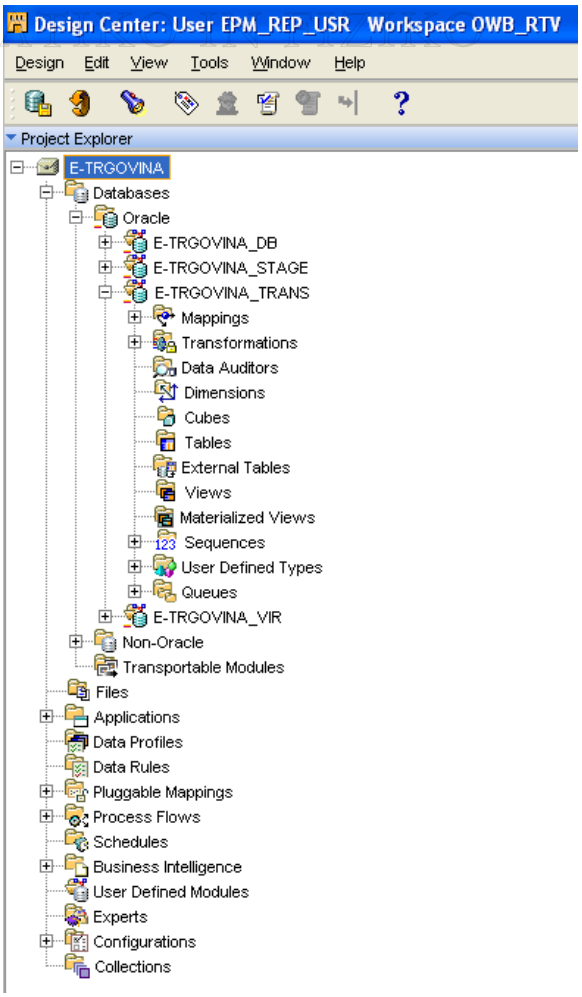
Slika 8 - Uvodno okno aplikacije

4.3.1 Databases

V tem delu bomo zbrali vse potrebne objekte (npr. tabele, poglede, funkcije, procedure, pakete, sekvence …), ki jih bomo potrebovali pri izgradnji preslikav (mapiranj). Razlog, da so preslikave v tem področju, torej v delu namenjenem podatkovni bazi je ta, da se v njihovem ozadju ne skriva nič drugega kot PL/SQL paket. Preslikave so glavni gradniki ETL postopkov (več o postopkih v razdelku 4.3.3).

V področju »Databeses« se nahajata dve opciji – »Oracle« in »Non-Oracle«. »Oracle« področje vsebuje objekte, ki se nahajajo v bazah podatkov, ki jih uporabljamo s sistemom Oracle, »Non-Oracle« pa objekte iz baz, upravljanimi z drugimi RDBMS. Omejili se bomo le na področje »Oracle«, saj bomo pri razvoju ETL postopkov za E-trgovino uporabili le Oraclovo bazo. V področju si lahko ustvarimo poljubno število modulov, ki nam pomagajo pri organizaciji metapodatkov v projektu. Običajno jih ustvarimo tako, da vsak modul pripada eni shemi v bazi (na Slika 9 se vidi module (sheme) iz naše baze: E-TRGOVINA_DB, E-TRGOVINA_STAGE, E-TRGOVINA_TRANS, E-TRGOVINA_VIR). Vsak modul vsebuje dvanajst komponent (Slika 9) namenjenih shranjevanju raznih metapodatkov (Transformations, Tables, External Tables, Views, Sequences, Mappings, Data Auditors, Dimensions, Cubes, Materialized Views, User Defined Types in Queues). Za razvoj ETL postopkov je bistvenih prvih šest.

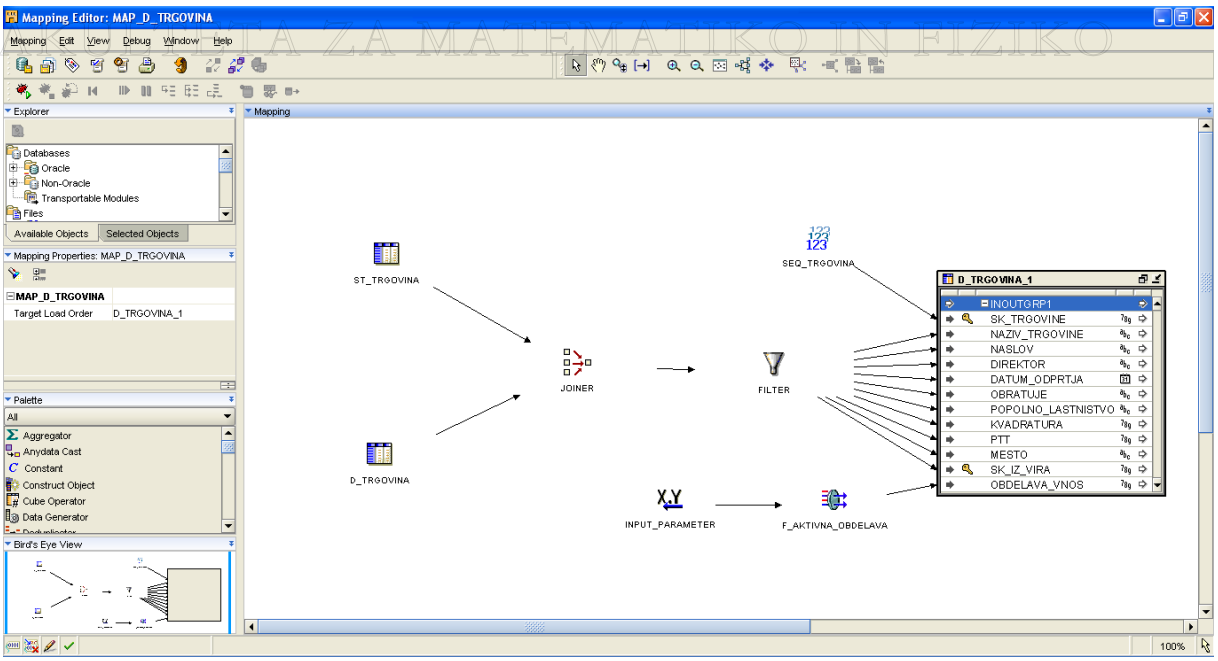
Orodje omogoča izgradnjo objektov (tabele, vpoglede, funkcije, procedure, pakete, sekvence …) v samem orodju ali uvoz le-teh iz baze. Vse te objekte pa lahko uporabimo pri izgradnji tako imenovanih preslikav (mapiranj). Kot že rečeno, v preslikavah definiramo pot podatkov iz izvorov v ponore in jih prilagodimo našim potrebam.



Slika 9 - Komponente v modulu

Ko v Design Centru (Slika 9: E-TRGOVINA -> Databases -> Oracle -> E-TRGOVINA_TRANS - Mappings) ustvarimo novo preslikavo, se nam odpre novo okno. Preslikavo ustvarimo tako, da na področje Mappings kliknemo z desnim gumbom in izberemo možnost »New«. Večino okna predstavlja delovna površina, kjer preslikavo razvijamo. Na levi strani pa imamo na voljo razne operatorje, ki nam pomagajo sestaviti preslikavo (Slika 10). Najpomembnejši operatorji so opisani v tabeli - Tabela 2. V njej je ikona, ki v orodju ponazarja operator, kratek opis in SQL oz PL/SQL ukaz, ki ustreza zapisu transformacije v programskem jeziku PL/SQL.

Kako te operatorje uporabimo, da z njimi zgradimo preslikavo, si bomo ogledali v razdelku 4.4.



Slika 10 - Primer preslikave (mapiranja)

SLIKA OPERATORJA	NAMEN	PRIPADAJOČ UKAZ V PL/SQL JEZIKU
 AGGREGATOR	»AGGREGATOR« izračuna povprečja, sešteje podatke, jih grupira in podobno.	GROUP BY, SUM, AVG, MIN, MAX …
 CONSTANT	»CONSTANT« določi razne nespremenljive zapise, ki jih lahko uporabljamo v celotni preslikavi (mapiranju) večkrat.	CONSTANT
 DEDUPLICATOR	»DEDUPLICATOR« odstrani ponavljajoče se zapise.	DISTINCT
 EXPRESSION	Operator »EXPRESSION« je uporaben za sestavljanje določenih izrazov. Npr. izraz, ki poskrbi za spremembo datuma v numerično obliko, združitev več polj v eno …	Raznorazni SQL ukazi (npr. UPPER, TO_CHAR, TRIM, TRUNC …)
 EXTERNAL_TABLE_OPERATOR	Če želimo kot vir podatkov v preslikavi (mapiranju) uporabiti določeno zunanjo tabelo (external table) iz baze, uporabimo operator »EXTERNAL_TABLE«.	EXTERNAL TABLE

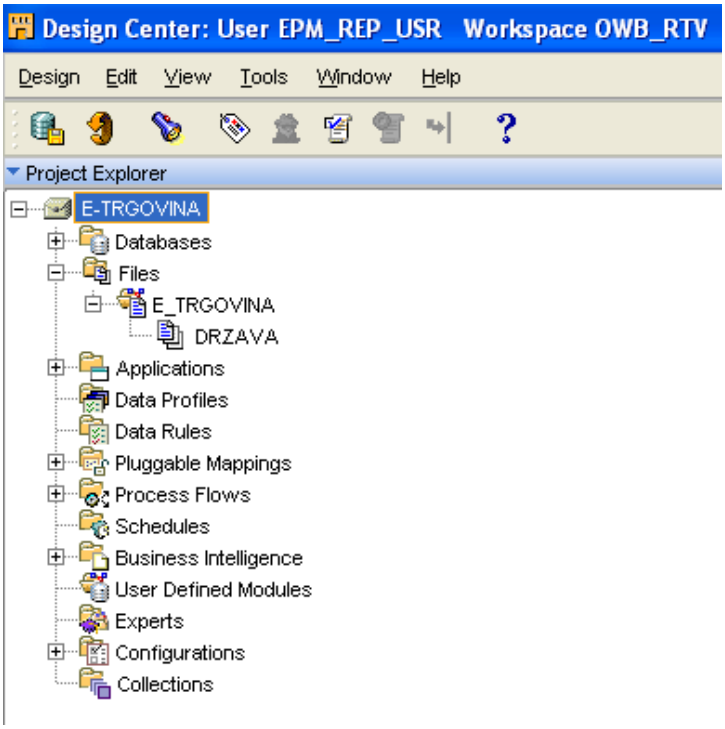
 FILTER	Z operatorjem »FILTER« dobimo samo tiste zapise, ki ustrezajo pogojem.	WHERE
 FLAT_FILE_OPERATOR	Če so izvorni podatki v navadni datoteki ali če želimo podatke zapisati v navadno datoteko, uporabimo operator »FLAT_FILE«.	
 JOINER	Operator »JOINER« poveže različne vire.	JOIN, INNER JOIN, OUTER JOIN …
 KEY_LOOKUP	Operator »KEY_LOOKUP« preko unikatnega ključa poišče podatke v tabeli ali vpogledu.	
 INPUT_PARAMETER	Operator »INPUT_PARAMETER« omogoči vnos parametra. Določimo ga preden poženemo preslikavo.	Npr. l_param IN number
 OUTPUT_PARAMETER	»OUTPUT_PARAMETER« operator služi, da določimo parameter, ki sprejme rezultat preslikave.	Npr. l_msg OUT varchar2(100)
 PIVOT	»PIVOT« operator naredi transformacijo ene vrstice v več vrstic.	PIVOT (od verzije 11g naprej)
 POST_MAPPING_PROCESS	»POST_MAPPING_PROCESS« operator izvede neko aktivnost po koncu izvajanja določene preslikave (mapiranja).	TRRIGER, PROCEDURE
 PRE_MAPPING_PROCESS	»PRE_MAPPING_PROCESS« operator izvede neko aktivnost pred začetkom izvajanja določene preslikave (mapiranja).	TRRIGER, PROCEDURE
 SEQUENCE_OPERATOR	Z operatorjem »SEQUENCE« generiramo zaporedje numeričnih vrednosti, ki jih lahko uporabimo za izgradnjo umetnega ključa v tabeli.	NEXTVAL, CURRVAL
 SET_OPERATION	»SET OPERATION« operator omogoča, da na različne načine (unija, presek …) več virov združimo v enega.	UNION, UNION ALL, INTERSECT, MINUS
 SORTER	»SORTER« operator razvrsti podatke po določenem pravilu.	ORDER BY

 SPLITTER	»SPLITTER« operator pošlje podatke iz enega vira v več različnih. Pri tem lahko nastavimo različne pogoje.	
 TABLE_OPERATOR	Če želimo kot vir ali ponor v preslikavi (mapiranju) uporabiti določeno tabelo iz baze, uporabimo operator »TABLE«.	TABLE
 TRANSFORMATION_OPERATOR	Operator »TRANSFORMATION« omogoča, da kot del transformacije uporabimo bodisi že v OWB pripravljeno transformacijo bodisi funkcijo, ki jo definiramo v jeziku PL/SQL.	FUNCTION, PACKAGE
 UNPIVOT	»UNPIVOT« operator naredi transformacijo več vrstic v eno samo.	UNPIVOT (od verzije 11g naprej)
 VIEW_OPERATOR	Če želimo kot vir podatkov v preslikavi (mapiranju) uporabiti določen pogled (view) iz baze, uporabimo operator »VIEW«.	VIEW

Tabela 2 - Predstavitev operatorjev za razvoj preslikav

4.3.2 Files

V področju »Files« imamo shranjene metapodatke o datotekah, ki nam služijo lahko kot vir ali ponor v preslikavah. Kot za razne bazne objekte, tudi za datoteke velja podobno, da jih lahko ustvarimo v orodju ali pa uvozimo že obstoječe. Obstaja še druga možnost uporabe datoteke kot vira v obliki zunanje tabele. V Oraclovi bazi lahko namreč poleg običajnih ustvarimo tudi zunanje tabele. Zunanja tabela je videti kot navadna tabela, vendar je narejena iz datoteke. V preslikavi (mapiranju) za te namene uporabljamo operatorja »flat_file« in »external_table«. Da se pri veliki količini datotek lažje znajdemo, lahko datoteke ločimo v različne module. Na sliki (Slika 11) v področju »Files« vidimo modul »E_TRGOVINA«, v njem pa se nahaja datoteka »DRZAVA«. Ta je ustvarjena le kot primer in jo v nadaljevanju ne bomo potrebovali. Ko si izberemo posamezno datoteko, določimo, kako so podatki v njej organizirani.



Slika 11 - Organizacija datotek v OWB.

4.3.3 Process flows

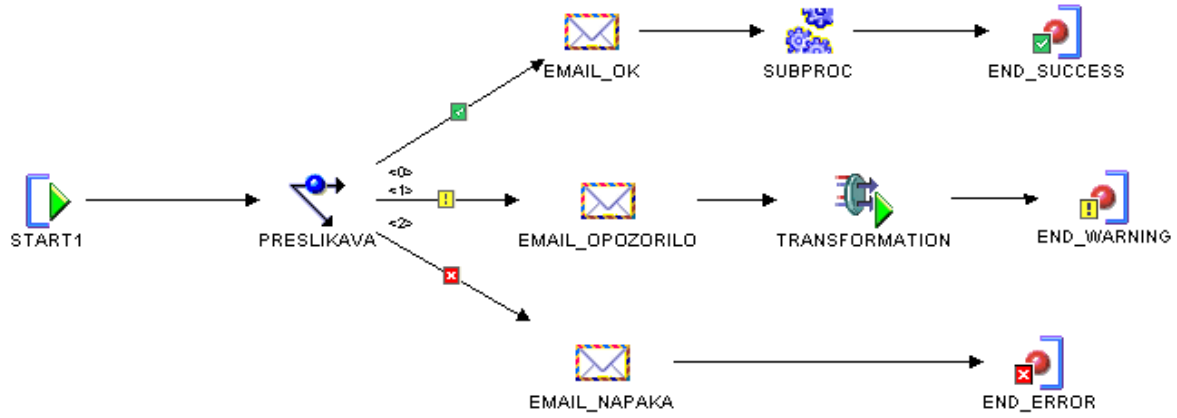
Področje »Process flows« je namenjeno razvoju postopkov, v katerih združimo sklope preslikav, postopkov in drugih funkcionalnosti. Za lažjo organizacijo postopkov je tudi tukaj možno grupiranje v različne module. Znotraj vsakega modula je možno še dodatno združevanje postopkov v tako imenovane »pakete«. V njih se nahajajo postopki (ang. process flows). Ko določen postopek razvijemo, se v ozadju ustvari XML koda, ki se ob prenosu postopka v bazo (ang. deployment) odloži na za to določeno mesto.

Postopek je sestavljen iz različnih aktivnosti, ki so med seboj povezane in določajo časovno sosledje. Te aktivnosti so lahko preslikave, procedure, postopki, itd. Povezave med aktivnostmi določajo vrstni red izvajanja določene aktivnosti ter preverijo pravilnost izvedbe

le-teh. Glavne aktivnosti so opisane v tabeli Tabela 3. V njej je za vsako aktivnost slika, ki jo na shemi označuje, in kratek opis.

Vsak postopek se začne z aktivnostjo »start« in konča z aktivnostjo »end«. Vse ostale aktivnosti dodamo glede na to, kaj naročnik želi. Za lažjo predstavo, kako je videti nek postopek, si pomagamo s sliko Slika 12. Postopek se začne z aktivnostjo »start«. Nadaljuje se z izvedbo preslikave. Od tukaj naprej pa je odvijanje postopka odvisno od tega, kako se bo preslikava izvedla:

- Če se preslikava uspešno izvede, bomo dobili sporočilo o uspešnosti na elektronski naslov. Nato se bo izvedel določen drug postopek (na sliki označen s »subproc«) in na koncu še aktivnost »end_success«.
- Če se preslikava izvede z opozorilom, bomo dobili sporočilo o tem. Nato se bo izvedla transformacija, ki je lahko neka PL/SQL procedura in na koncu aktivnost »end_warning«.
- Če pa se v preslikavi zgodi napaka, smo o tem obveščeni in končamo postopek z aktivnostjo »end_error«.



Slika 12 - Primer postopka (process flow). Prirejeno po: Oracle warehouse builder user's guide, str. 505

V primeru, da pride pri izvajanju postopka do težav, imamo več možnosti, kako to obvladovati. Ena od možnosti je, da se postopek na mestu napake ustavi in čaka, da ga pristojni, po odstranitvi težave, poženejo naprej. Druga možnost je, da se postopek enostavno ustavi in ga je potrebno še enkrat pognati od začetka. Lahko tudi nadaljujemo z izvajanjem postopka ne glede na težave in samo preko elektronske pošte obvestimo stranko, da je v izvajanju postopkov prišlo do določenih težav.

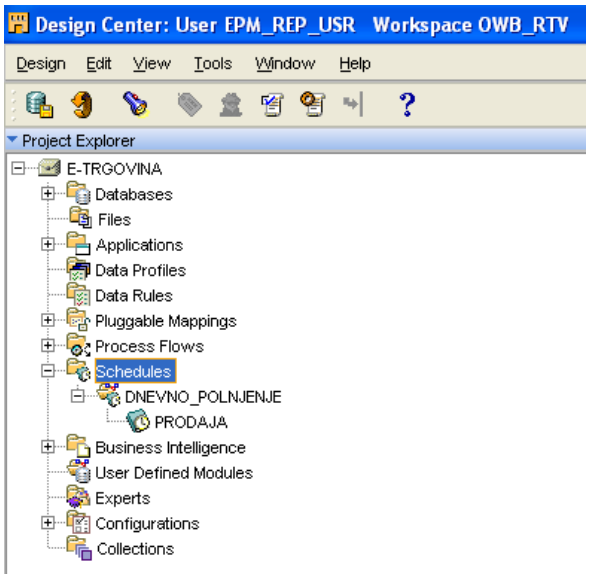
SLIKA OPERATORJA	NAMEN
 AND1	Aktivnost »AND« uporabljamo v navezi z aktivnostjo »FORK«. Ta združuje vzporedno izvajajoče se aktivnosti. Če uporabimo aktivnost »AND« se morajo vse vzporedne aktivnosti izvesti uspešno, da se postopek nadaljuje.
 EMAIL	Aktivnost »EMAIL« pošlje elektronsko sporočilo npr. o tem, ali se je neka stvar uspešno izvedla ali ne.
 END_ERROR	Z aktivnost »END_ERROR« postopek zaključimo in označimo, da je prišlo do napake.
 END_SUCCESS	Z aktivnost »END_SUCCESS« končamo postopek in označimo, da se je uspešno izvedel.
 END_WARNING	Z aktivnostjo »END_WARNING« postopek zaključimo in dodamo opozorilo, da so v njem možne napake.
 END_LOOP	Aktivnost »END_LOOP« definira zaključek zanke FOR ali WHILE.
 FORK	Aktivnost »FORK« omogoča vzporedno izvajanje aktivnosti v postopku.
 FOR_LOOP	Aktivnost »FOR_LOOP« omogoča, da v postopku določene aktivnosti ponavljamo.
 MANUAL	Aktivnost »MANUAL« ponuja možnost, da se na tem mestu izvajanje postopka prekine. Postopek potem čaka na ročno intervencijo. Pogosto to aktivnost uporabljamo tam, kjer v primeru napake želimo, da se postopek ustavi in počaka, da ga od tam naprej poženemo ročno.
 MAPPING	Aktivnost »MAPPING« v postopek doda izvedbo določene preslikave.
 OR	Aktivnost »OR« uporabljamo v navezi z vzporedno izvajajočima se aktivnostima (ustvarimo ju z aktivnostjo »FORK«). Vsaj ena od njih se mora izvesti uspešno, da se postopek nadaljuje.
 ROUTE	Aktivnost »ROUTE« omogoča dvosmerno povezavo med dvema aktivnostima. Ker puščice v postopkih ne omogočajo dvosmernih povezav, moramo v primeru, da pride do takšne situacije uporabiti aktivnost »ROUTE«.
 SQLPLUS	Aktivnost »SQLPLUS« omogoča izvajanje SQL skript.

START	Aktivnost »START« vsebuje vsak postopek. Na njej lahko določimo vhodne parametre postopka.
SUB_PROCESS	Aktivnost »SUB_PROCESS« pokliče določen drug postopek.
TRANSFORMATION	Aktivnost »TRANSFORMATION« omogoča uporabo PL/SQL funkcij in procedur.
USER_DEFINED	Aktivnost »USER_DEFINED« omogoča izvajanje raznih skript (npr. bat datoteke …)
WAIT	Aktivnost »WAIT« omogoča zakasnitev izvedbe določene aktivnosti.
WHILE_LOOP	Aktivnost »WHILE_LOOP« omogoča ponavljanje neke aktivnosti.

Tabela 3 - Aktivnosti namenjene razvoju ETL postopkov

4.3.4 Schedules

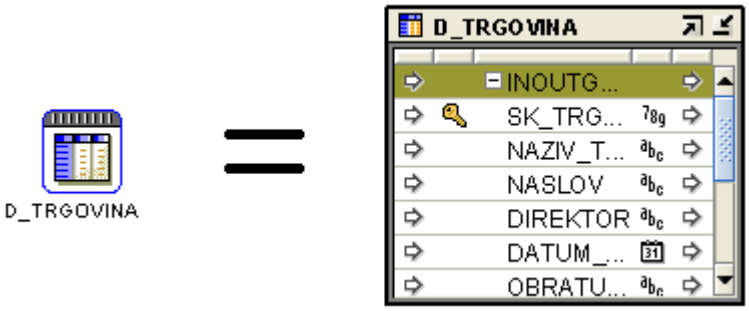
Za avtomatizacijo izvajanja postopkov je namenjeno področje »Schedules«. V fazi razvoja si ne želimo, da se postopek izvaja avtomatsko. Zato ga takrat poganjamo ročno. Ko pa je postopek enkrat končan, mu glede na potrebe določimo urnik izvajanja. Ustvarjene urnike lahko ločimo v različne module, kjer jim definiramo lokacijo. Ta nam pove, kje v bazi se bo ta objekt nahajal po prenosu (deployment). Možnost avtomatizacije postopkov je na voljo od verzije 10g dalje.



Slika 13 - Urnik izvajanja postopkov.

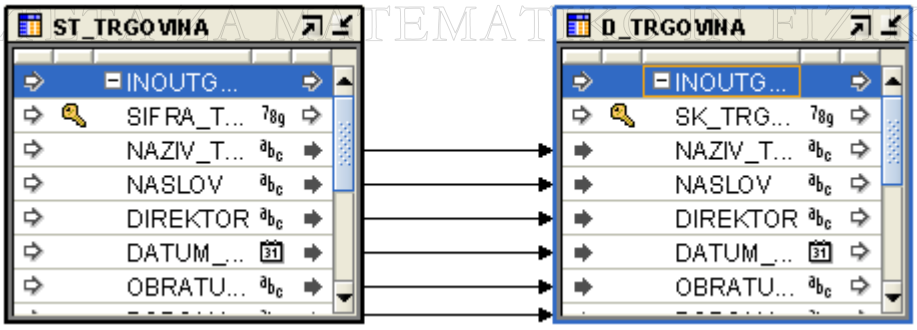
4.4 Uporaba operatorjev v preslikavah

Vsaka preslikava je sestavljena iz množice raznih operatorjev, ki so med seboj povezani. Ko imamo odprto okno, v katerem razvijamo preslikavo (urejevalnik preslikave), vidimo na levi strani okna nabor operatorjev (levo spodaj), možnost nastavitev lastnosti določenega operatorja v preslikavi (levo v sredini) in raziskovalca po objektih v Design Centru (levo zgoraj). To se lepo vidi na sliki Slika 10. Večino okna predstavlja delovna površina, kamor operatorje postavimo. To storimo tako, da operator z miško zagrabimo in ga povlečemo na delovno površino, kjer ga izpustimo. Operator se bo ob spustitvi na delovno površino razprl (»odprta« oblika operatorja). »Zaprto« in »odprto« obliko operatorja »TABLE_OPERATOR« vidimo na sliki Slika 14.



Slika 14 - "Odprt" in "zaprt" način prikaza operatorja

Nekateri operatorji imajo čarovnika, ki nas pripelje do želenih nastavitev, drugi nam omogočajo, da v njih napišemo SQL kodo. Operatorjem, ki predstavljajo ponor (končni cilj podatkov), moramo določiti, kako se bodo polnili (npr. lahko podatke najprej vse izbrišemo in nato napolnimo, lahko samo popravljamo podatke ali samo polnimo nove …). Vse te lastnosti in nastavitve določamo tako, da kliknemo na želen operator. Ko to storimo, se nam prikažejo možnosti za nastavitve le-tega v levem delu okna (»Mapping Properties«). Operatorje med seboj povežemo s puščicami, ki določijo, iz katerega polja v katero polje operatorja naj gredo podatki (Slika 15). Povezave ustvarimo tako, da zagrabimo polje, ki ga želimo preslikati in ga povežemo s poljem, v katerega ga želimo preslikati.



Slika 15 - Povezave med operatorji

4.5 Nadzor nad tokom podatkov

Ko pridemo do faze, ko so preslikave in postopki razviti, jih moramo začeti uporabljati. Za nadzor in izvajanje le-teh nam orodje ponuja dve možnosti. Prva je uporaba aplikacije Control Center, druga pa uporaba spletne aplikacije Repository Browser (natančneje sta opisani v poglavju 4.5.1). Obstajajo pa še druge kontrole za nadzor nad izvajanjem postopkov, ki jih lahko sami razvijemo (natančneje opisane v poglavju 4.5.2).

4.5.1 Control Center in Repository Browser

Aplikacija Control Center je del aplikacije Design Center. V prvi vrsti je namenjena prenosu razvitih objektov v Design Centru v bazo (»deployment«). V drugi vrsti pa je tudi zelo praktična za ročno poganjanje in spremljanje postopkov. V njej lahko tudi nadziramo ali naj se določen postopek oziroma preslikave izvaja avtomatsko. Če bi bila aplikacija Control Center še malo bolj dodelana, spletne aplikacije Repository Browser sploh ne bi potrebovali. Problem je v tem, da Control Center ne omogoča prikaza opisa napak pri avtomatiziranih postopkih. Prikaže nam samo, da se postopek ni pravilno izvedel, čas začetka postopka in čas, ko se je napaka zgodila. Druga pomanjkljivost Control Centra pa je ta, da postopke, ki so razviti tako, da se ob napaki ustavijo na mestu napake, po odpravi le-te ne moremo pognati od tam naprej. To dvoje pa nam omogoča Repository Browser. V njem so omogočene skoraj vse funkcionalnosti, kot jih ima Control Center in še druge. Opis napak je bolj podroben. Tudi ta aplikacija ima svoje pomanjkljivosti. Aplikacija ne omogoča prenosa (deployment) objektov v bazo. Prav tako je aplikacija počasna pri veliki količini razvitih objektov. To počasnost naj bi povzročala napaka v aplikaciji, ki pa naj bi v verziji 11gR2 bila odpravljena.

4.5.2 Dnevnik izvajanja postopkov

Da zagotovimo pravilno izvajanje postopkov, je na voljo še ena vrsta kontrole nad njimi. Postopke bomo spremljali tako, da bomo vsak zagnan postopek zabeležili v bazi v posebni tabeli. V njej bomo hranili podatke o tem, za kateri postopek gre, kdaj se je začel, končal, njegov status, razne parametre … S tem zagotovimo, da istočasno ne izvajamo dveh (ali več) postopkov z istim imenom, ker bi se taka postopka lahko motila. To bomo preverjali s posebno preslikavo, ki bo to tabelo tudi polnila. Ta nas bo opozorila, če bomo želeli zagnati postopek z enakim imenom, kot ga ima kak postopek, ki se že izvaja. Ker se v tabeli beležijo vsi zagnani postopki, nam ta ponuja tudi zgodovino izvajanja postopkov in možnost nastavljanja raznih parametrov (npr. za katero obdobje naj se podatki napolnijo …). V našem primeru se bo ta tabela imenovala ADM_DNEVNIK (Slika 16).

Column Name	ID	Pk	Null?	Data Type
OBDELAVA_ID	1	1	N	NUMBER
OBDELAVA_ZACETEK	2		N	DATE
OBDELAVA_KONEC	3		Y	DATE
POSTOPEK_IME	4		N	VARCHAR2 (100 Byte)
NALOZI_PODATKE_OD	5		Y	DATE

Slika 16 - administratorska tabela ADM_DNEVNIK

4.6 Prednosti in slabosti

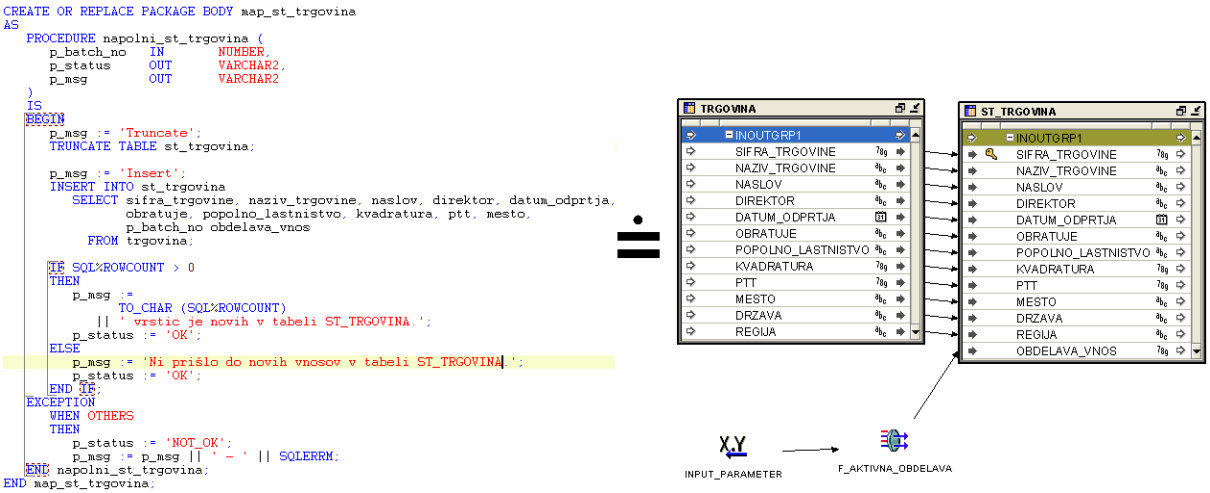
Orodje Oracle Warehouse Builder ima kar nekaj prednosti in slabosti v primerjavi z razvojem ETL postopkov v programskem jeziku PL/SQL. V podrobnosti ne bomo zahajali, ampak prednosti in slabosti le povzeli.

Prednosti:

- Zaradi grafičnih vmesnikov je postopek oziroma preslikava lažje razumljiva, ker dejansko vidimo od kje pridejo podatki (vir), kaj z njimi delamo in kam gredo (ponor) (na sliki Slika 17 vidimo razliko med zapisom preslikave, ki je razvita v jeziku PL/SQL in prikazom iste preslikave, razvite v orodju OWB).
- Slike postopkov oziroma preslikav že same po sebi predstavljajo dokumentacijo le-teh. S tem prihranimo veliko časa, ki bi ga drugače porabili za pisanje dokumentacije.
- Za izvajanje in nadzor nad postopki orodje že ponuja primerne aplikacije.
- Že samo orodje nam postavi neke standarde, po katerih naj bi vsi razvijalci delali.

Slabosti:

- Namestitev samega orodja ni najbolj enostavna.
- Program še ni dovolj preizkušen. Velikokrat v programu naletimo na napako.
- Pri kompleksnejših preslikavah je težko ugotoviti, kje je prišlo do napake.
- Za popravke preslikav ali postopkov porabimo več časa.



Slika 17 - Preslikavo za določene skupine podatkov iz vira v območje za pripravo podatkov (levo PL/SQL paket, desno OWB preslikava – oboje nam vrne isti rezultat).

5 Primer razvoja ETL postopkov

V tem poglavju si bomo natančneje ogledali razvoj ETL postopkov za podjetje E-trgovina. Zadevo bomo razdelili na štiri sklope. V prvem (razdelek 5.1) bomo sestavili postopke za zajem podatkov na viru in njihov prenos v področje za obdelavo. Ko smo podatke pripeljali v področje za obdelavo, je potrebno razviti še preslikave, ki bodo podatke prenesle v podatkovno skladišče. V drugem in tretjem sklopu (razdelek 5.2 in 5.3) bomo tako prenesli podatke iz območja za obdelavo podatkov v podatkovno skladišče (v dimenzijske tabele in tabelo dejstev). V četrtem sklopu (razdelek 5.4) pa bomo vse preslikave po sklopih združili v postopke.

5.1 Prenos podatkov iz vira v območje za obdelavo podatkov

Podatki, ki nam bodo služili kot izvor, se nahajajo v Oraclovi bazi (verzija sistema 11gR1). Pri prenosu podatkov iz vira v območje za obdelavo podatkov je priporočljivo, da podatke prenesemo brez transformiranja, ker je ravno prenos med različnimi bazami časovno najbolj potraten. Torej podatke prenesemo v območje za obdelavo podatkov v točno taki obliki kot so na viru. Tabele v območju za obdelavo bomo poimenovali enako kot so poimenovane tabele na viru, le da jim bomo dodali predpono »ST_«. Edina razlika bo ta, da v tabele dodamo dodatno polje (OBDELAVA_VNOS). To polje bo določalo številko postopka preko katerega bomo iz tabele ADM_DNEVNIK izvedeli kdaj in kateri postopek je podatke napolnil.

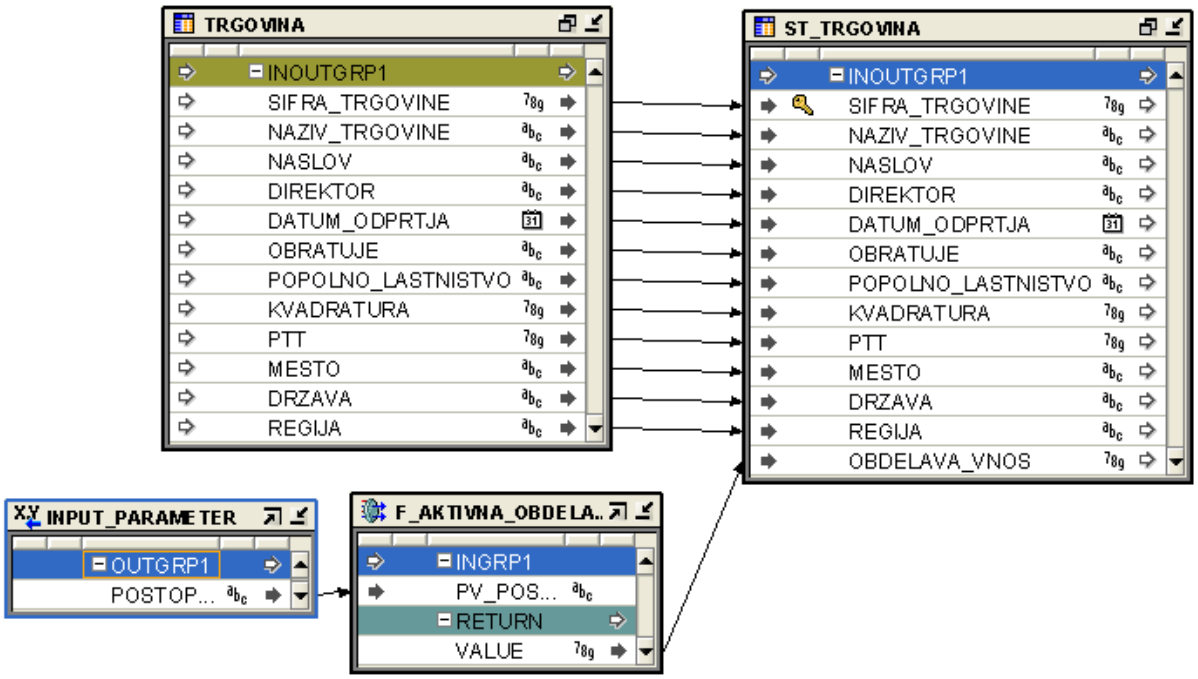
Izvorne tabele, ki jih bomo uporabili so TRGOVINA, IZDELEK, BARVA_IZDELKA, SKUPINE_IZDELKOV, KATEGORIJE_IZDELKOV in PRODAJA.

Za vsako skupino podatkov bomo naredili svojo preslikavo. Preslikave bomo poimenovali na način »MAP_« + ime ponorne tabele (npr. MAP_ST_TRGOVINA).

5.1.1 MAP_ST_TRGOVINA

Na sliki Slika 18 vidimo preslikavo, ki prenese podatke tabele TRGOVINA iz vira v področju za obdelavo podatkov. Kot je vidno na sliki, se podatki preslikajo v tabelo ST_TRGOVINA brez kakršnihkoli transformacij. Preslikava vsebuje še operatorja »INPUT_PARAMETER« in »TRANSFORMATION«. Z »INPUT_PARAMETROM« določimo ime postopka. To storimo preden poženemo preslikavo. Ta je potreben zato, da operatorju, ki mu sledi (»TRANSFORMATION« na sliki je poimenovan »F_AKTIVNA_OBDELAVA«), priskrbi ime postopka. S pomočjo tega podatka operator izvede funkcijo, ki vrne številko postopka iz tabele ADM_DNEVNIK, ki jo vpišemo v polje OBDELAVA_VNOS.

- **Vir**: TRGOVINA
- **Ponor**: ST_TRGOVINA
- **Način polnjenja**: najprej izbris vseh podatkov, nato vnos vseh podatkov

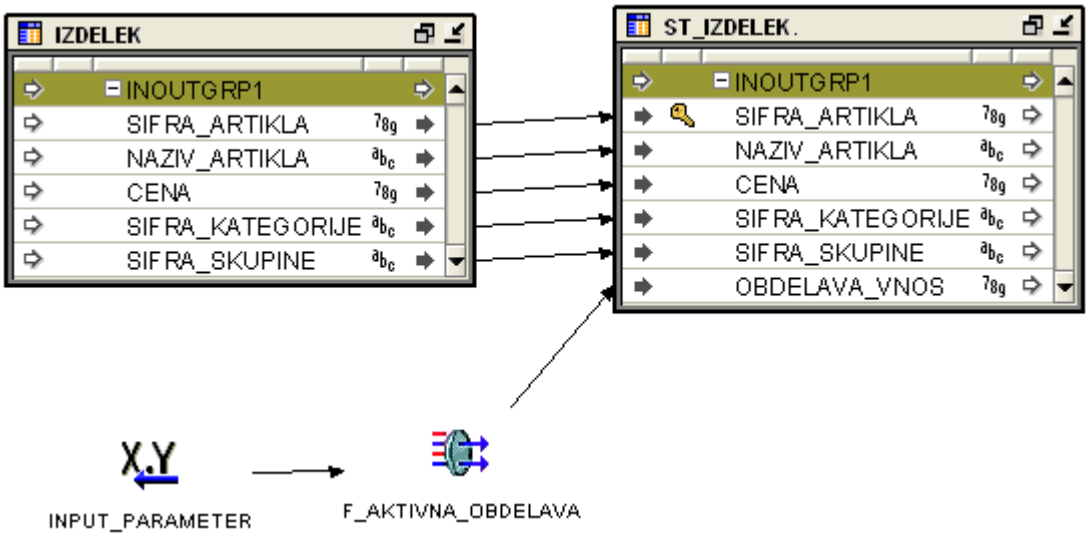


Slika 18 - Preslikava MAP_ST_TRGOVINA

5.1.2 MAP_ST_IZDELEK

Pri vseh nadaljnjih preslikavah v razdelku 5.1 je zadeva zelo podobna kot v razdelku 5.1.1. Operatorja »INPUT_PARAMETER« in »TRANSFORMATION« sta predstavljena v »zaprti« obliki. Da pridemo do iste slike, kot je v razdelku 5.1.1, dvakrat kliknemo na operatorja.

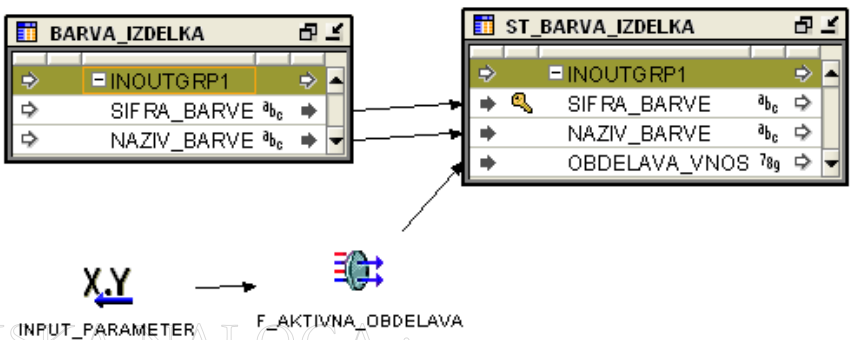
- **Vir**: IZDELEK
- **Ponor**: ST_IZDELEK
- **Način polnjenja**: najprej izbris vseh podatkov, nato vnos vseh podatkov



Slika 19 - Preslikava MAP_ST_IZDELEK

5.1.3 MAP_ST_BARVA_IZDELKA

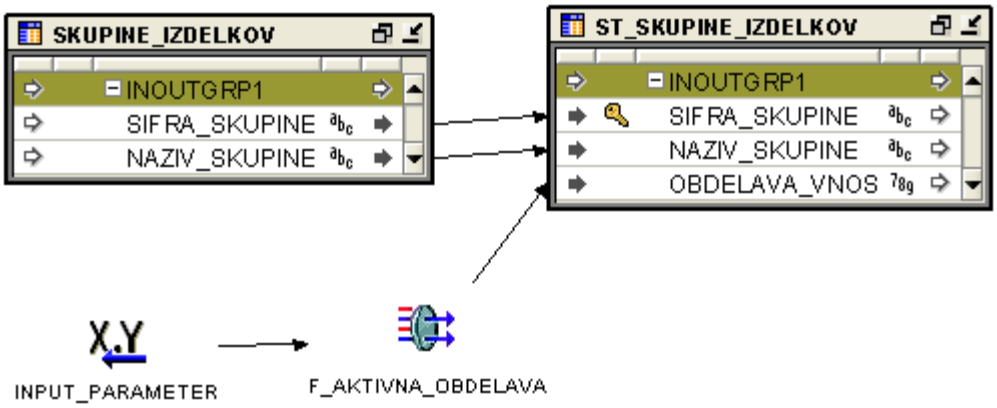
- **Vir**: BARVA_IZDELKA
- **Ponor**: ST_BARVA_IZDELKA
- **Način polnjenja**: najprej izbris vseh podatkov, nato vnos vseh podatkov



Slika 20 - Preslikava MAP_ST_BARVA_IZDELKA

5.1.4 MAP_ST_SKUPINE_IZDELKOV

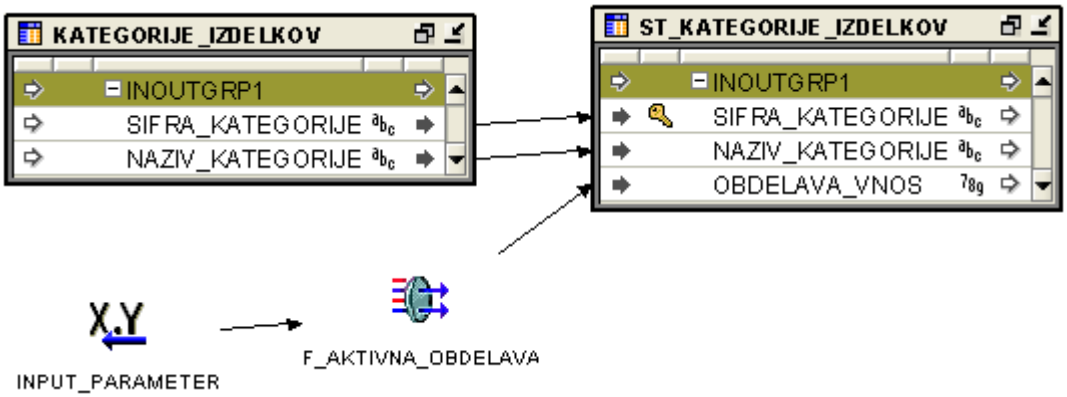
- **Vir**: SKUPINE_IZDELKOV
- **Ponor**: ST_SKUPINE_IZDELKOV
- **Način polnjenja**: najprej izbris vseh podatkov, nato vnos vseh podatkov



Slika 21 - Preslikava MAP_ST_SKUPINE_IZDELKOV

5.1.5 MAP_ST_KATEGORIJE_IZDELKOV

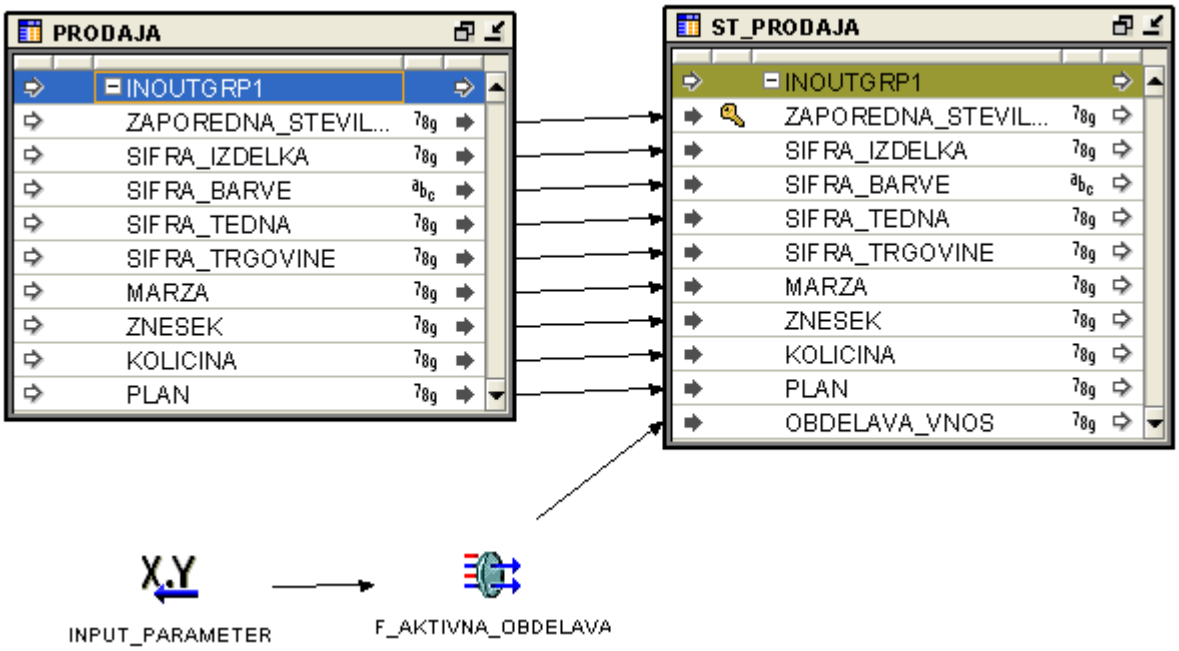
- **Vir**: KATEGORIJE_IZDELKOV
- **Ponor**: ST_KATEGORIJE_IZDELKOV
- **Način polnjenja**: najprej izbris vseh podatkov, nato vnos vseh podatkov



Slika 22 - Preslikava MAP_ST_KATEGORIJE_IZDELKOV

5.1.6 MAP_ST_PRODAJA

- **Vir**: PRODAJA
- **Ponor**: ST_PRODAJA
- **Način polnjenja**: najprej izbris vseh podatkov, nato vnos vseh podatkov



Slika 23 - Preslikava MAP_ST_PRODAJA

5.2 Prenos podatkov iz območja za obdelavo podatkov v podatkovno skladišče – dimenzijske tabele

Ko smo podatke pripeljali v področje za obdelavo, je potrebno razviti še preslikave, ki bodo podatke prenesle v podatkovno skladišče. Spodnji podrazdelki prikazujejo te preslikave. Vsem sta skupni dve stvari. Prva je ta, da v dimenzijske tabele vpeljemo tako imenovane umetne ključe, ki bodo določali enoličnost zapisov v tabeli. Te vrednosti bomo generirali s operatorjem »SEQUENCE«. Druga pa je beleženje številke postopka, ki jo dobimo preko funkcije napisane v PL/SQL programskem jeziku. Ta pogleda v tabelo ADM_DNEVNIK in tam vidi za kateri postopek gre. Princip je pri vseh preslikavah za polnjenje naših dimenzij enak. Natančnejši opis ene teh preslikav je opisan v razdelku 5.2.1, pri ostalih pa je naveden le glavni vir podatkov, ponorna tabela in način, kako se podatki polnijo v ponorno tabelo.

5.2.1 MAP_D_IZDELEK

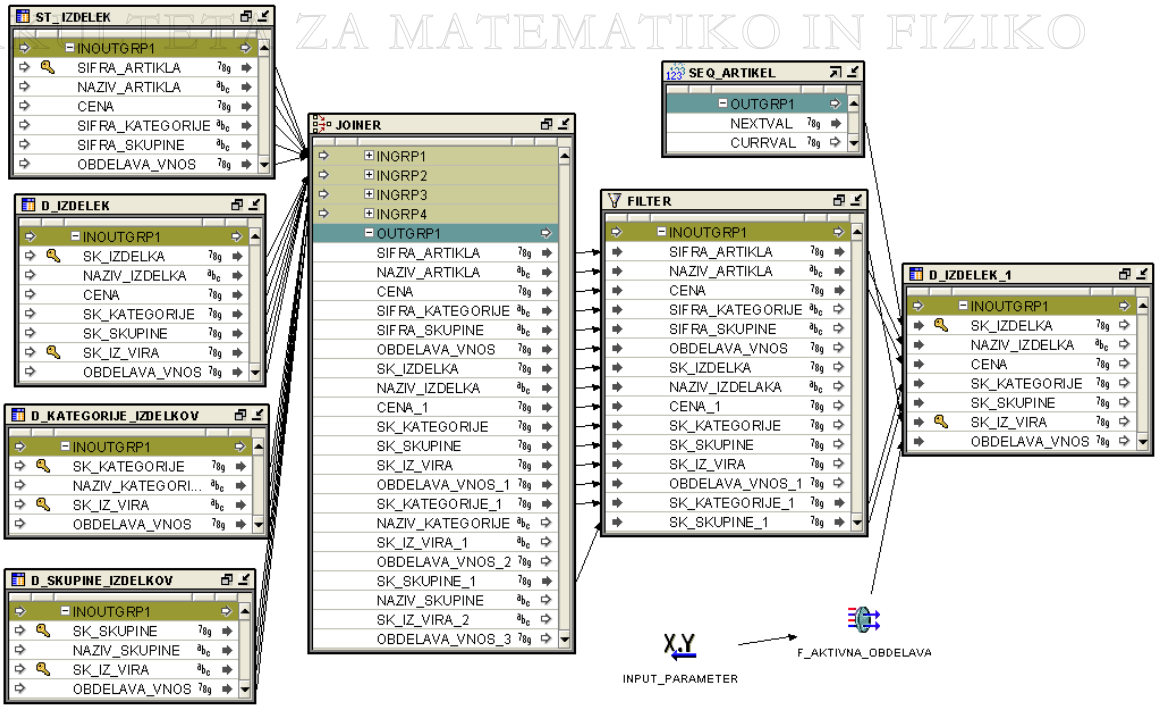
V preslikavi MAP_D_IZDELEK (Slika 24) je naš glavni vir podatkov tabela ST_IZDELEK. Ker dimenzijska tabela D_IZDELEK vsebuje tudi polja, ki služijo kot povezave na druge dimenzije, jih moramo med seboj povezati. To storimo z operatorjem »JOINER«. Našo izvorno tabelo ST_IZDELEK povežemo s tabelama D_KATEGORIJE_IZDELKOV in D_SKUPINE_IZDELKOV, ki med drugim vsebujeta tudi polja, ki služijo kot povezave na tabelo ST_IZDELEK. Tako pridemo do umetnih ključev iz D_KATEGORIJE_IZDELKOV in D_SKUPINE_IZDELKOV, ki jih potrebujemo v tabeli D_IZDELEK. Pri tem je pomembno, da sta dimenzijski tabeli D_KATEGORIJE_IZDELKOV in D_SKUPINE_IZDELKOV napolnjeni pred polnjenjem tabele D_IZDELEK.

Povezavo med tabelama ST_IZDELEK in D_IZDELEK naredimo zato, da ne prenašamo vedno vseh zapisov, ampak samo tiste, ki se na novo pojavijo oziroma se jim spremeni vrednost v polju NAZIV_IZDELKA. Te pogoje zapišemo v operatorju »FILTER«.

Z operatorjem »SEQUENCE si zagotovimo enoličnost zapisov v dimenziji D_IZDELEK.

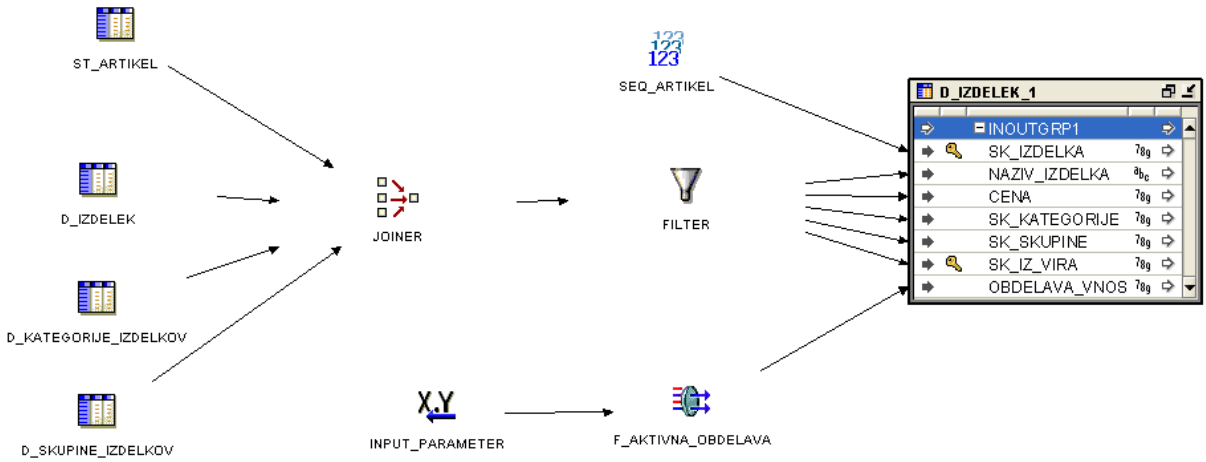
Za polnjenje polja OBDELAVA_VNOS poskrbita operatorja »INPUT_PARAMETER« in »TRANSFORMATION«. »INPUT_PARAMETER« zagotovi operatorju »TRANSFORMATION« ime postopke. Ta pokliče funkcijo razvito v jeziku PL/SQL, ki preko tabele ADM_DNEVNIK poišče številko postopka.

- **Vir**: ST_IZDELEK
- **Ponor**: D_IZDELEK
- **Način polnjenja**: najprej vnesemo zapise, ki so novi. Nato popravimo zapise, ki imajo v določenih poljih spremembe.



Slika 24 - Preslikava MAP_D_IZDELEK v »odprti obliki«

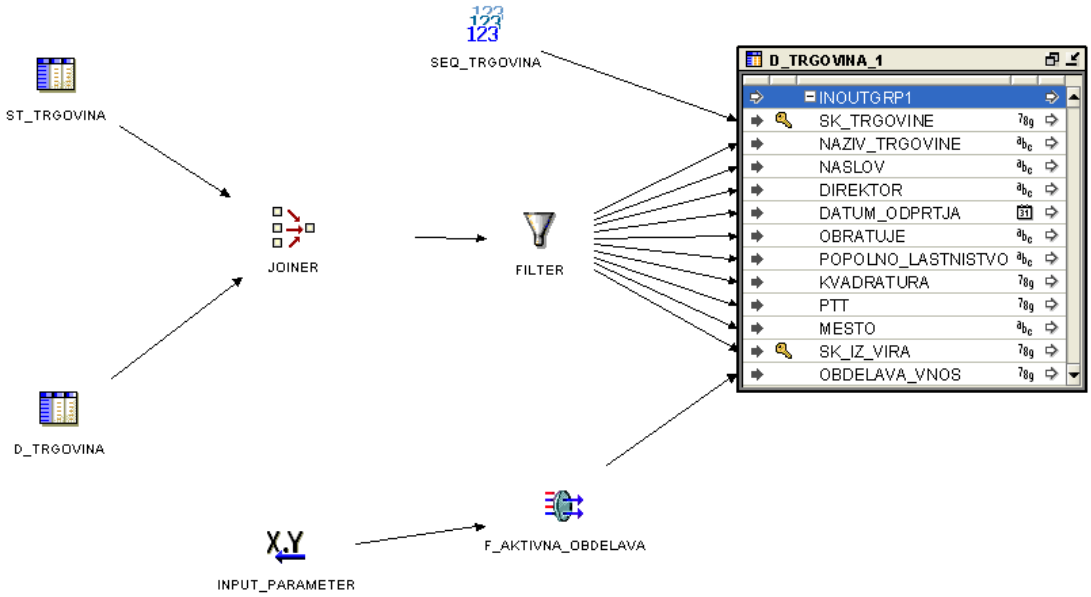
Če isto preslikavo prikažemo še v »zaprti« obliki, bo slika videti nekako tako, kot kaže Slika 25. Tudi vse nadaljnje slike preslikav v razdelku 5.2 bomo prestavili v tej tako imenovani »zaprti« obliki.



Slika 25 - Preslikava MAP_D_IZDELEK v "zaprti" obliki

5.2.2 MAP_D_TRGOVINA

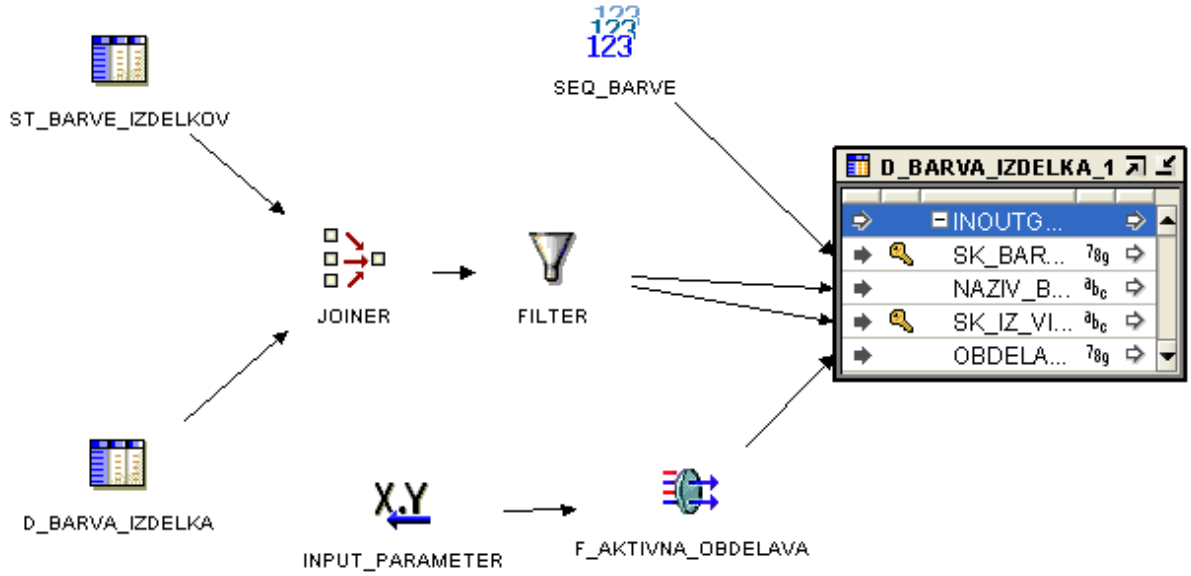
- **Vir**: ST_TRGOVINA
- **Ponor**: D_TRGOVINA
- **Način polnjenja**: najprej vnesemo zapise, ki so novi. Nato popravimo zapise, ki imajo v določenih poljih spremembe.



Slika 26 - Preslikava MAP_D_TRGOVINA

5.2.3 MAP_D_BARVA_IZDELKA

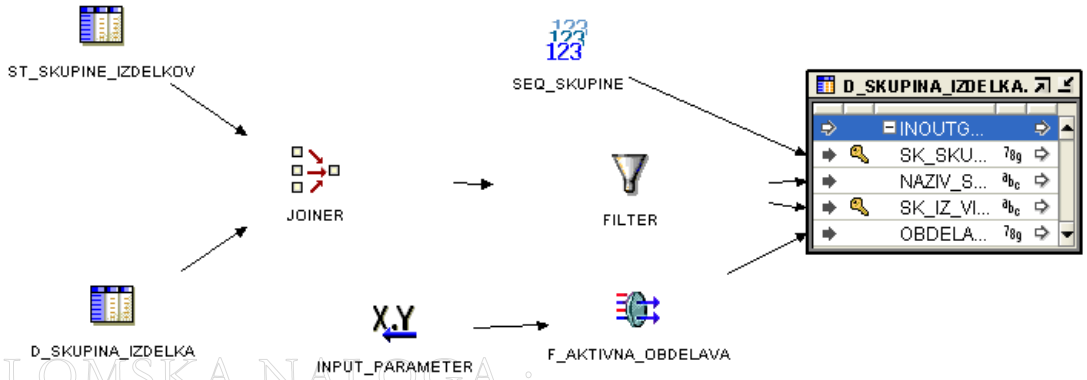
- **Vir**: ST_BARVA_IZDELKA
- **Ponor**: D_BARVA_IZDELKA
- **Način polnjenja**: najprej vnesemo zapise, ki so novi. Nato popravimo zapise, ki imajo v določenih poljih spremembe.



Slika 27 - Preslikava MAP_D_BARVA_IZDELKA

5.2.4 MAP_D_SKUPINE_IZDELKOV

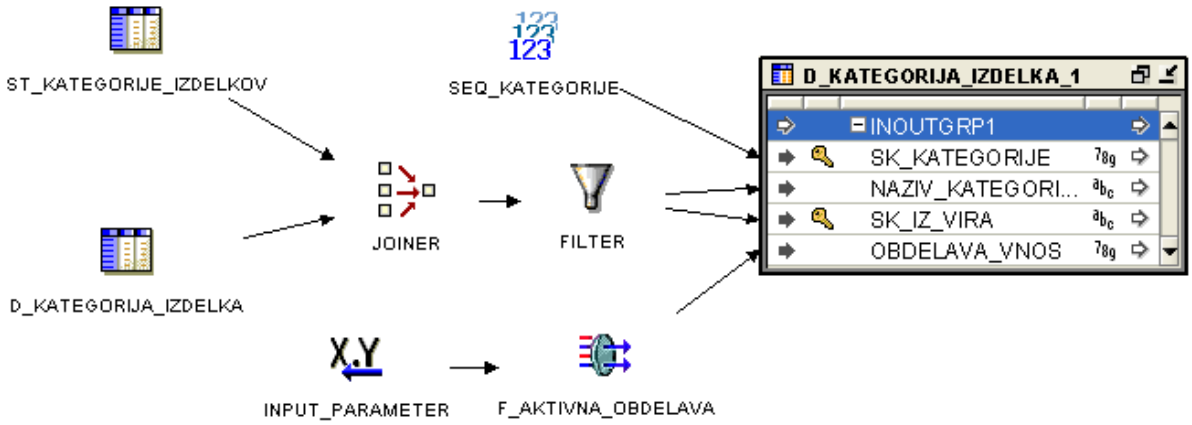
- **Vir**: ST_SKUPINE_IZDELKOV
- **Ponor**: D_SKUPINE_IZDELKOV
- **Način polnjenja**: najprej vnesemo zapise, ki so novi. Nato popravimo zapise, ki imajo v določenih poljih spremembe.



Slika 28 - Preslikava MAP_D_SKUPINE_IZDELKOV

5.2.5 MAP_D_KATEGORIJE_IZDELKOV

- **Vir**: ST_KATEGORIJE_IZDELKOV
- **Ponor**: D_KATEGORIJE_IZDELKOV
- **Način polnjenja**: najprej vnesemo zapise, ki so novi. Nato popravimo zapise, ki imajo v določenih poljih spremembe.



Slika 29 - Preslikava MAP_D_KATEGORIJE_IZDELKOV

5.2.6 D_KOLEDAR

Dimenzijsko tabelo D_KOLEDAR, v kateri bodo shranjeni razni časovni podatki (npr. teden v letu, mesec, leto, kvartal …), ni potrebno osveževati, ker so podatki v njej nespremenljivi. Torej jo bomo napolnili enkrat samkrat. To lahko napolnimo ročno ali pa si pripravimo proceduro, ki nam bo pomagala napolniti tabelo D_KOLEDAR. Polja v tej tabeli prikazuje spodnja slika (Slika 29).

Column Name	ID	Pk	Null?	Data Type
SK_TEDNA	1	1	N	NUMBER (38)
TEDEN_V_LETU	2		Y	NUMBER (38)
LETO	3		Y	VARCHAR2 (4 Byte)
FISKALNO_OBDOBJE	4		Y	VARCHAR2 (10 Byte)
LETO_TEDEN	5		Y	VARCHAR2 (7 Byte)
KVARTAL	6		Y	VARCHAR2 (2 Byte)
IME_MESECA	7		Y	VARCHAR2 (10 Byte)
MESEC	8		Y	VARCHAR2 (2 Byte)

Slika 30 - Dimenzijska tabela D_KOLEDAR

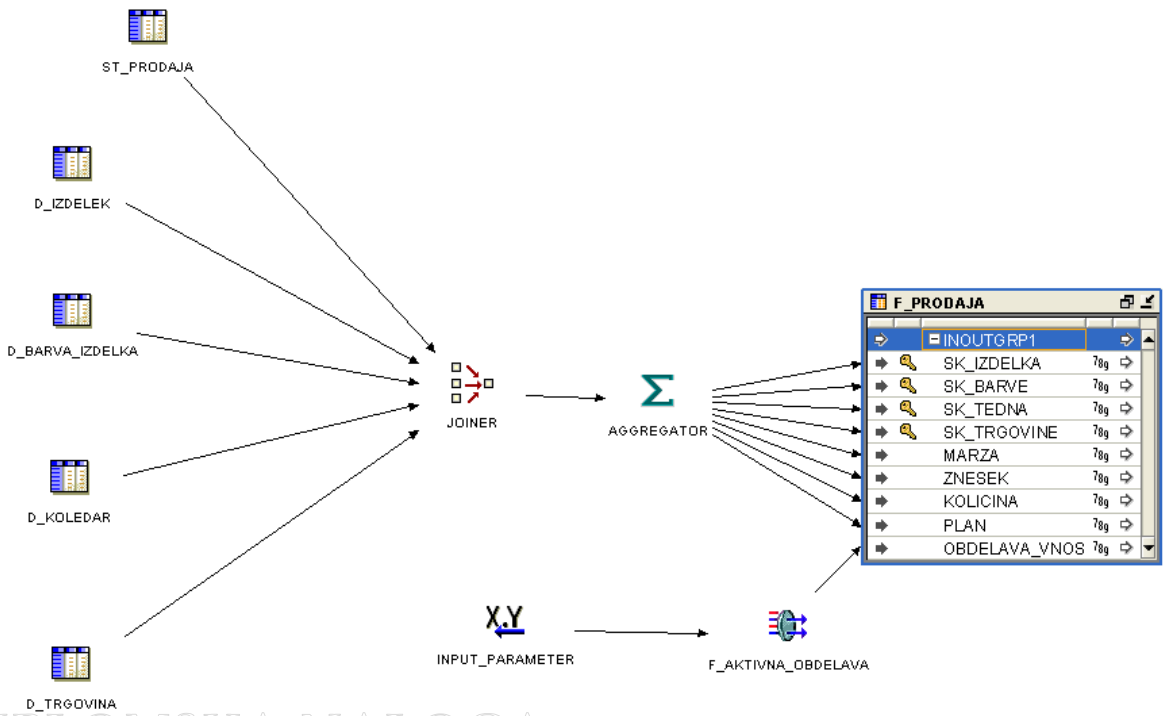
5.3 Prenos podatkov iz območja za obdelavo podatkov v podatkovno skladišče – tabela dejstev

Polnjenje tabele dejstev predstavlja združitev vseh glavnih dimenzijskih tabel z izvorno tabelo dejstev (ST_PRODAJA). »Zrno« tabele bo sestavljeno iz izdelka, časa, trgovine in barve izdelka.

5.3.1 MAP_F_PRODAJA

Preslikava se začne z združitvijo dimenzijskih tabel in izvorne tabele dejstev. To naredimo z operatorjem »JOINER«. Nato podatke grupiramo glede na »zrno« tabele dejstev (F_PRODAJA). Na primer, če se isti dan večkrat proda isti izdelek iste barve v isti trgovini, se v izvornih podatkih nahaja večkrat (v več zapisih). V tabelo dejstev (F_PRODAJA) pa se zapiše samo en zapis, ki pa ima sešteta merljiva dejstva (cena, količina, marža …). To moramo narediti zato, ker imamo na »zrnu« tabele postavljen primarni ključ. Tudi tukaj, kot v vse ostale tabele, zapišemo podatke o tem, kateri postopek je, glede na tabelo ADM_DNEVNIK, tabelo napolnil.

- **Vir**: ST_PRODAJA
- **Ponor**: F_PRODAJA
- **Način polnjenja**: najprej izbris vseh podatkov, nato vnos vseh podatkov



Slika 31 - Preslikava MAP_F_PRODAJA

5.4 Združitev preslikav v postopke

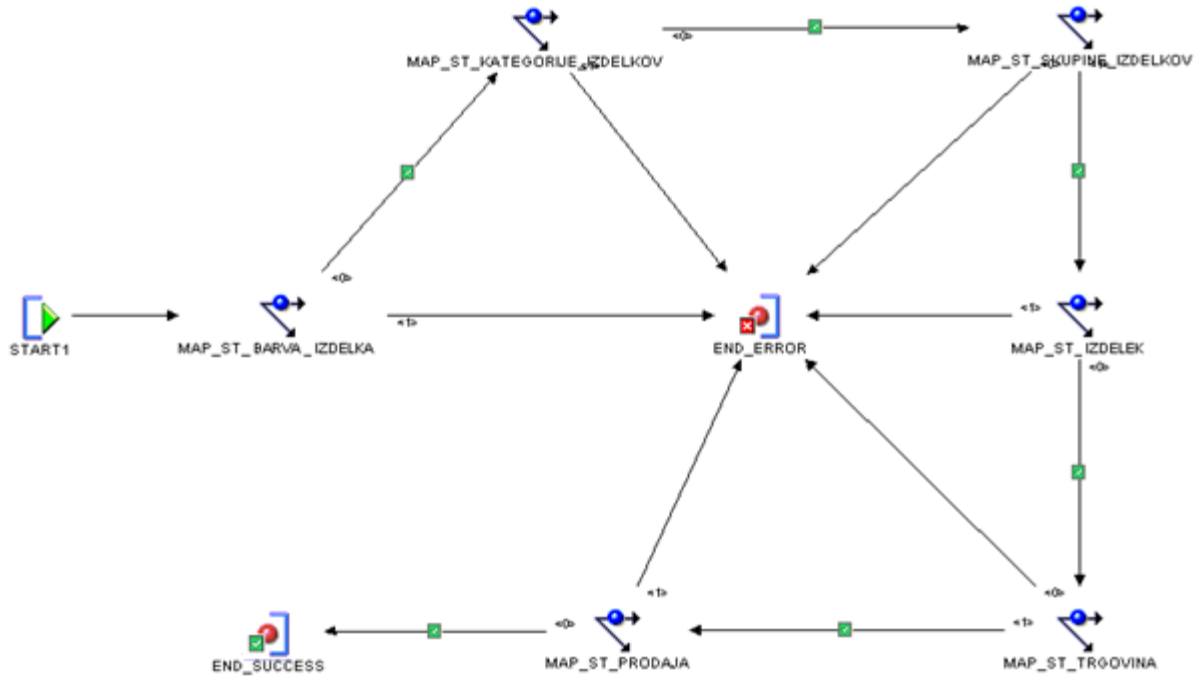
Preslikave, ki imajo neke skupne lastnosti, združimo v isti postopek. Tako bomo preslikave, ki smo jih razvili v prejšnjih razdelkih (5.1, 5.2 in 5.3), tudi združili v tri postopke. Prvi postopek (razdelek 5.4.1) bo zajel preslikave, ki prenesejo podatke iz vira v področje za obdelavo podatkov. Drugi in tretji (razdelka 5.4.2 in 5.4.3) bosta zajela preslikave, ki prenesejo podatke iz območja za obdelavo podatkov v podatkovno skladišče. Ločena sta zato, ker bomo v prvem prenašali dimenzijske podatke, v drugem pa dejstva. Vse tri postopke pa bomo združili v en postopek imenovan PF_DNEVNI (razdelek 5.4.4). Ta bo avtomatiziran in nas bo preko elektronske pošte obveščal o uspešnosti oziroma neuspešnosti izvedbe.

5.4.1 PF_NALOZI_ZAC_POD

Namen tega postopka je, da združi vse razvite preslikave, ki prenesejo podatke iz vira v področje za obdelavo podatkov. Vrstni red izvajanja preslikav ni pomemben, ker so preslikave neodvisne ena od druge. Postopek PF_NALOZI_ZAC_POD vsebuje preslikave:

- MAP_ST_BARVA_IZDELKA
- MAP_ST_KATEGORIJE_IZDELKOV
- MAP_ST_SKUPINE_IZDELKOV
- MAP_ST_IZDELEK
- MAP_ST_TRGOVINA
- MAP_ST_PRODAJA

Postopek se začne z aktivnostjo »START«. Najprej se izvede preslikava MAP_ST_BARVA_IZDELKA. Nato se zaporedoma izvajajo preslikave, ki so naštete v zgornjem seznamu. Postopek se konča se z aktivnostjo »END_SUCCESS«. V primeru, da se določena preslikava izvede neuspešno, gre postopek v aktivnost »END_ERROR« in se konča. V tem primeru za odkrivanje in odpravo napake uporabimo Control Center ali Repository Browser.



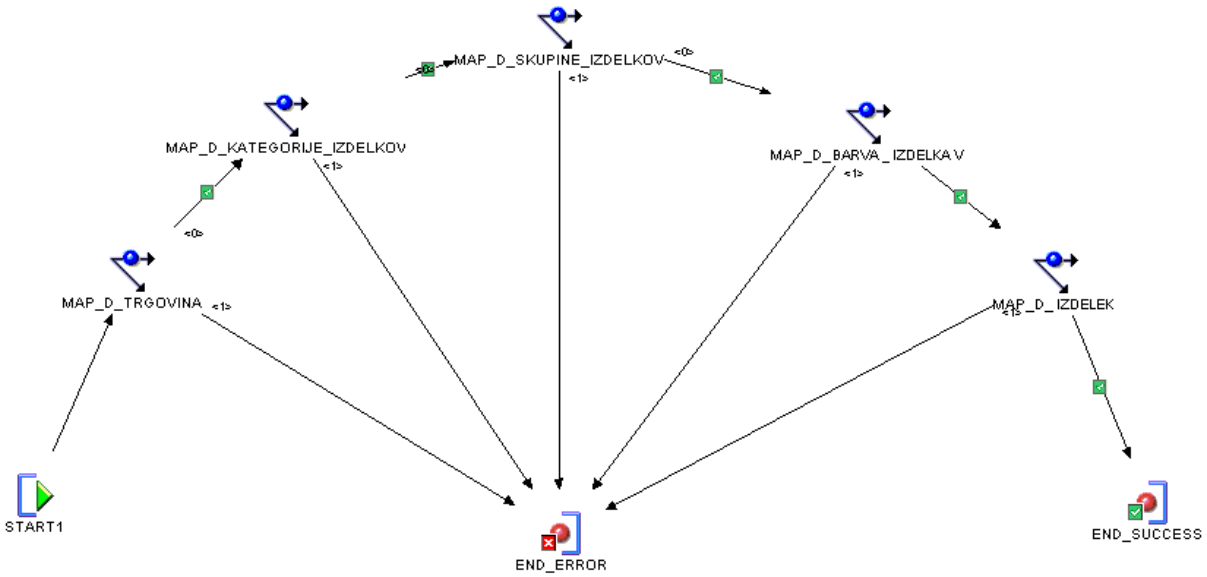
Slika 32 - Postopek PF_NALOZI_ZAC_POD

5.4.2 PF_NALOZI_DIM

Postopek PF_NALOZI_DIM, katerega namen je združiti preslikave, ki prenesejo in pripravijo podatke za dimenzijske tabele, vsebuje preslikave:

- MAP_D_TRGOVINA
- MAP_D_KATEGORIJE_IZDELKOV
- MAP_D_SKUPINE_IZDELKOV
- MAP_D_BARVA_IZDELEKA
- MAP_D_IZDELEK

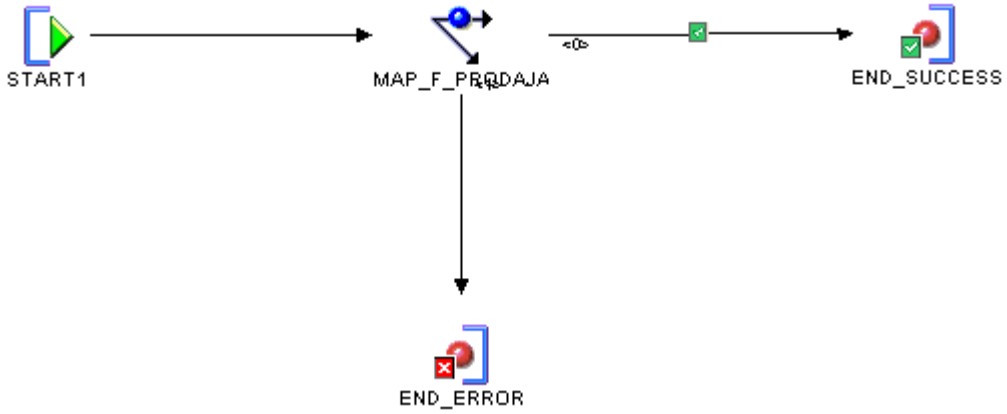
Postopek PF_NALOZI_DIM poteka na isti način, kot predhodni postopek (PF_NALOZI_ZAC_POD), samo da je tukaj pomemben vrstni red, ker je preslikava MAP_D_IZDELEK odvisna od preslikav MAP_D_KATEGORIJE_IZDELKOV in MAP_D_SKUPINE_IZDELKOV.



Slika 33 - Postopek PF_NALOZI_DIM

5.4.3 PF_NALOZI_DEJSTVA

Postopek PF_NALOZI_DEJSTVA vsebuje le preslikavo MAP_ F_PRODAJA. Zato je enostaven. Po začetku se izvede preslikava. Če se izvede uspešno, postopek končamo z aktivnostjo »END_SUCESS«, v nasprotnem primeru pa z aktivnostjo »END_ERROR«.

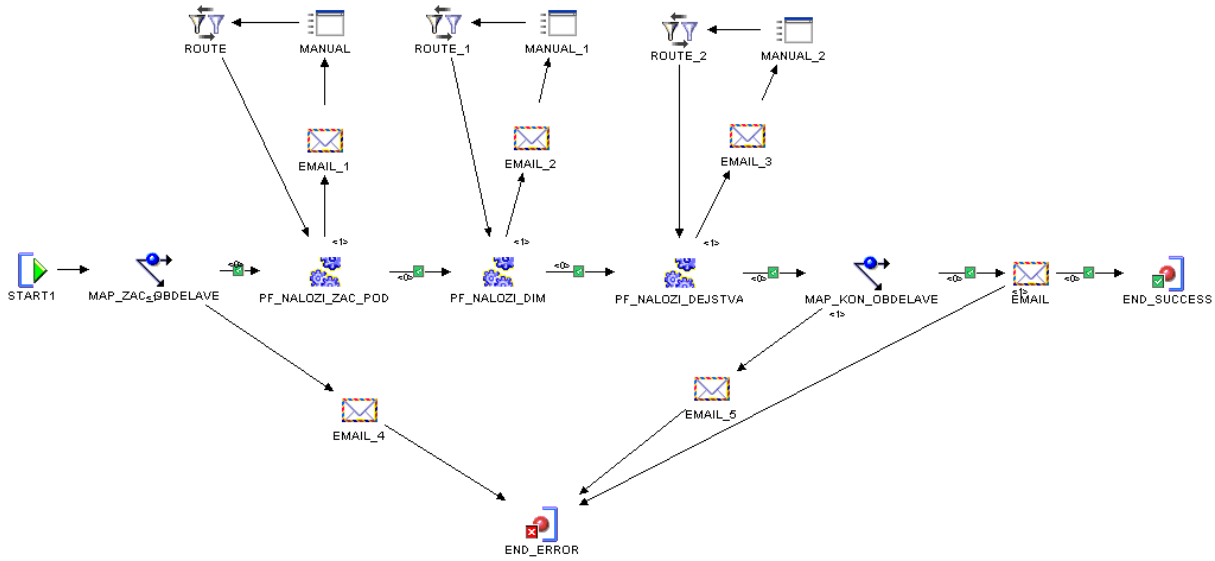


Slika 34 - Postopek PF_NALOZI_DEJSTVA

5.4.4 PF_DNEVNI

PF_DNEVNI je glavni postopek, ki združi vse tri prej razvite postopke (PF_NALOZI_ZAC_POD, PF_NALOZI_DIM in PF_NALOZI_DEJSTVA). Dodali bomo še druge funkcionalnosti, kot so obveščanje o uspešnosti oziroma neuspešnosti postopka in druge.

Na začetku se najprej izvede preslikava, ki zapiše zapis v tabelo ADM_DNEVNIK (podatki o tem, za kateri postopek gre, kdaj se je začel …). Nato se izvedejo naši trije postopki. Če se izvedejo uspešno, se za njimi izvede še preslikava, ki zapiše v tabelo ADM_DNEVNIK datum in čas zaključka postopka in nas o uspešno izvedenem prenosu podatkov v podatkovno skladišče obvesti po elektronski pošti. Če pa v katerem od treh postopkov slučajno pride do napake, smo o tem obveščeni. Izvedba postopka čaka, da jo po odpravi napake od tam naprej poženemo dalje. Obstaja tudi možnost, da pride do napake na eni od preslikav. V tem primeru se postopek konča in ga je potrebno po odpravi napake še enkrat pognati v celoti.



Slika 35 - Postopek PF_DNEVNI

6 Izvajanje ETL postopkov za E-trgovino

Diplomsko nalogo bomo zaključili z opisom, kako napolnimo podatkovno skladišče. Predstavili bomo oba načina - s pomočjo Control Centra (razdelek 6.1) in Repository Browserja (razdelek 6.2). Za konec pa bomo v razdelku 6.3 glavni postopek avtomatizirali, tako, da se bo izvajal samodejno vsak dan.

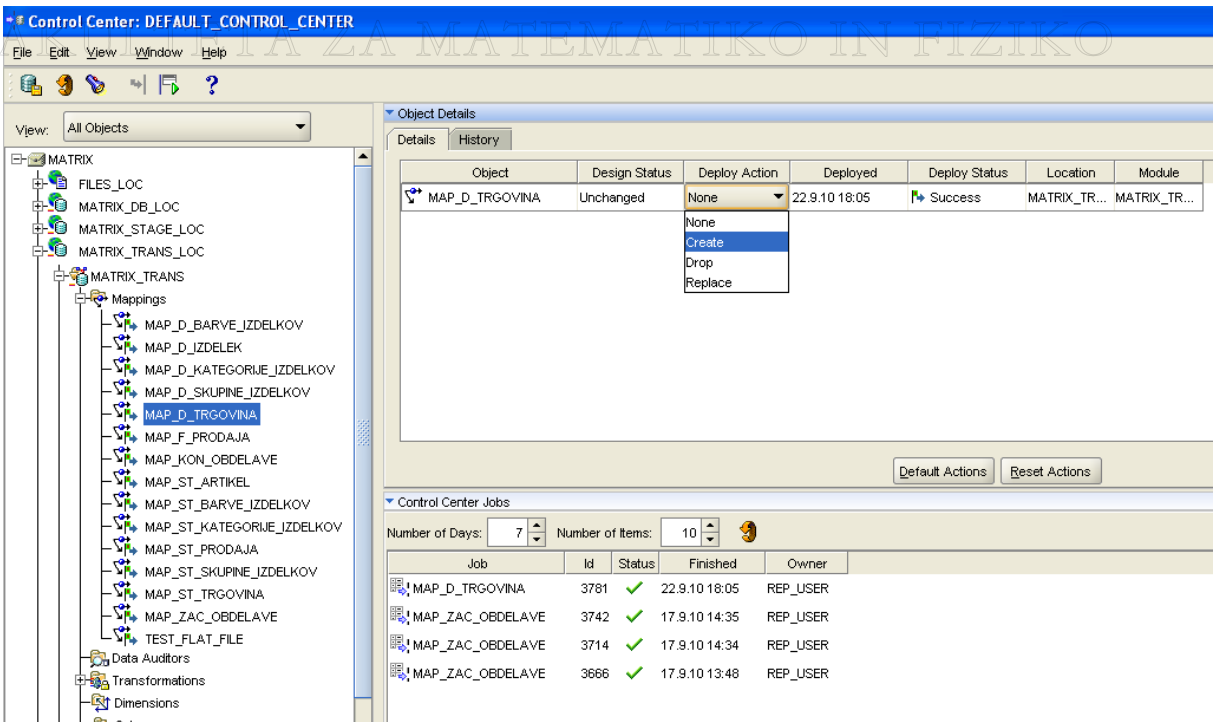
6.1 Prenos in izvedba preslikave s pomočjo Control Centra

Kot smo že povedali v razdelku 4.4.1, je uporaba Control Centra zelo praktična za prenos postopkov in preslikav v bazo (ang. deployment) ter ročno izvajanje preslikav oziroma postopkov. To nam pride prav predvsem pri razvoju, ko je potrebno stvari testirati. Kot primer si bomo ogledali, kako se naredi prenos preslikave MAP_D_TRGOVINA v bazo in kako se jo izvede tam. Za vse ostale preslikave in postopke je način prenosa in izvajanja enak.

6.1.1 Prenos preslikave MAP_D_TRGOVINA v bazo

Vsak na novo narejen ali popravljen objekt v Design Centru je potrebno prenesti v bazo. To pomeni, da ustvarimo oziroma osvežimo kodo v bazi. V tem razdelku si bomo ogledali, kako poteka prenos preslikave MAP_D_TRGOVINA v bazo.

- Odpremo Control Center iz orodja Control Center (v orodni vrstici izberemo »Tools« in nato »Control Center«).
- Poiščemo objekt, ki ga želimo prenesti (v našem primeru MAP_D_TRGOVINA).
- Želen objekt označimo tako, da orodje ve kaj narediti (na voljo so »Create«, »Replace« in »Drop«). Izberemo »Create«, kar pomeni, da želimo ustvariti nov objekt v bazi. »Replace« izberemo v primeru popravkov določenega objekta, »Drop« pa če želimo objekt izbrisati.
- Ko objekt označimo, kliknemo na gumb »Deploy« ( ).
- Odpre se novo okno, v katerem vidimo informacije o uspešnosti oziroma neuspešnosti prenosa.
- Če dobimo sporočilo, da se je uspešno preneslo, potem lahko objekt uporabljamo. Če prenos ni bil uspešen, moramo težave odpraviti in postopek ponoviti.

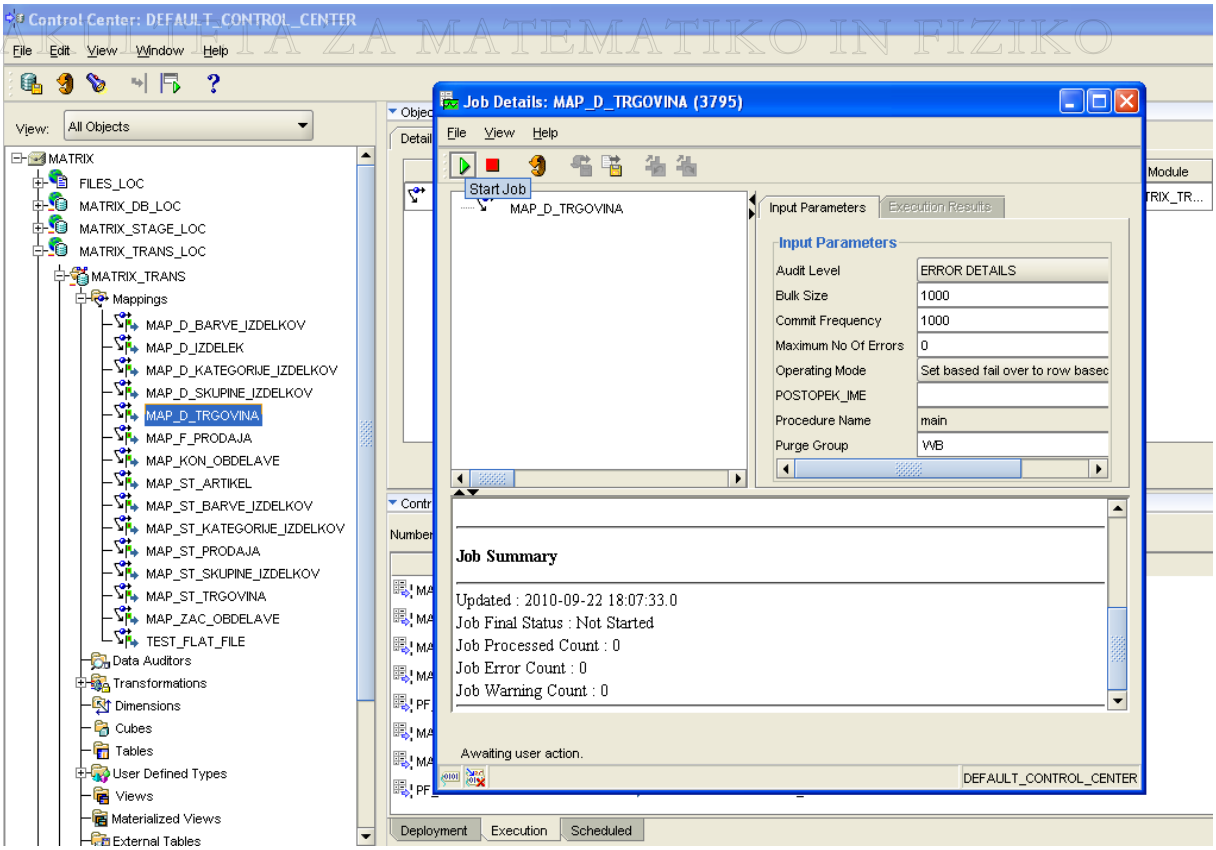


Slika 36 - Aplikacija Control Center pri prenosu preslikave v bazo

6.1.2 Izvedba preslikave MAP_D_TRGOVINA

Ko je preslikava (ali pa postopek) uspešno prenešena v bazo, smo pripravljeni, da ga uporabimo. Postopek zagona preslikave je enostaven in ne zahteva posebne razlage, zato ga navedimo v obliki alinej:

- V orodju Design Center odpremo Control Center.
- Poiščemo preslikavo MAP_D_TRGOVINA.
- Pritisnemo gumb »Start«.
- Odpre se novo okno, v katerem lahko opravimo določene nastavitve glede števila napak, ki jih dovolimo, podrobnosti beleženja napak, določili morebitne parametre …
- Po morebitni spremembi parametrov kliknemo gumb »Start Job«.
- V spodnji polovici okna lahko spremljamo uspešnost izvajanja.



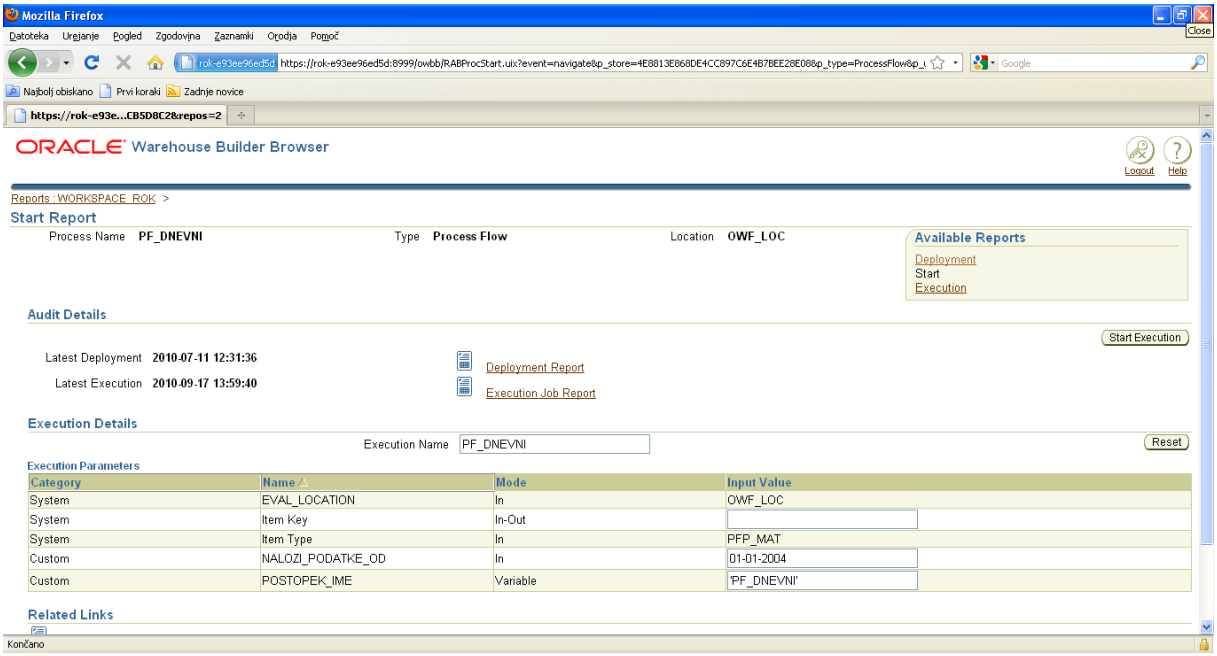
Slika 37 - Aplikacija Control Center tik pred izvedbo preslikave

6.2 Izvedba postopka PF_DNEVNI s pomočjo orodja Repository Browser

Drugi način izvajanja preslikav oziroma postopkov nam omogoča Repository Browser. Tu si bomo ogledali izvedbo postopka PF_DNEVNI. Vendar moramo najprej postopek prenesti v bazo. To spet storimo s pomočjo Control Centra, ker nam Repository Browser tega ne omogoča. Postopek PF_DNEVNI bomo izvedli preko spletne aplikacije po spodnjih korakih:

- V orodju Design Center odpremo Repository Browser.
- Opravimo prijavo, kjer pa moramo uporabiti uporabniški račun s posebnimi pravicami.
- Po uspešni prijavi, se nam odpre okno, kjer imamo na voljo sedem povezav. Izberemo povezavo »Object Summary Report«.
- Odpre se nam nova stran, kjer izberemo želen objekt (v našem primeru PF_DNEVNI).
- Spet se nam odpre nova stran, kjer desno zgoraj izberemo povezavo »Start«.
- Prikaže se nam stran, kjer lahko nastavimo morebitne parametre in pritisnemo gumb »Start execution«. S tem smo pognali postopek.

- Za spremljanje izvajanja se vrnemo na začetno stran (preko povezave »Reports : IME_WORKSPACE-a«), kjer spet izberemo povezavo »Object Summary Report«. Poiščemo pognani postopek in kliknemo povezavo »Execution«. Odpre se nam nova stran, kjer lahko spremljamo izvajanje postopka.

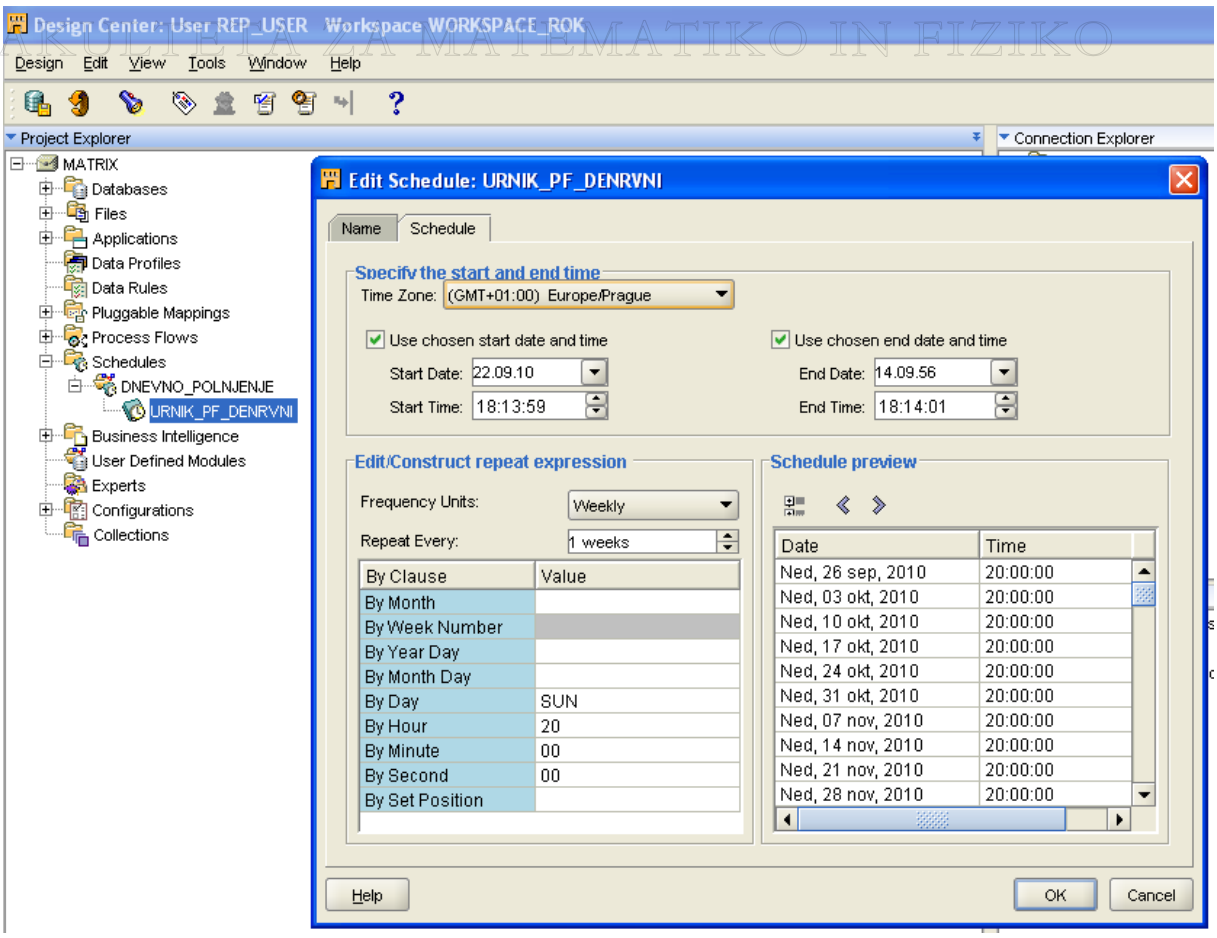


Slika 38 - Spletna aplikacija Repository Browser tik pred izvedbo postopka

6.3 Avtomatizacija postopka PF_DNEVNI

Podjetje E-trgovina si želi, da se podatki osvežujejo vsako nedeljo. Da nam ne bo potrebno postopka PF_DNEVNI, ki zajema vse potrebno, poganjati ročno vsako nedeljo, bomo ustvarili urnik, ki bo avtomatsko izvajal postopek. Pomembno je, da so izvorni podatki v času izvajanja postopka dostopni. Za to se je potrebno uskladiti s stranko.

V Design Centru ustvarimo modul imenovan DNEVNO_POLNJENJE. V njem bo urnik izvajanja za postopek PF_DNEVNI. Ta se bo imenoval URNIK_PF_DNEVNI. V Design Centru je na voljo čarovnik, ki nas po korakih vodi do uspešno ustvarjenega urnika. Ko je urnik ustvarjen, ga je potrebno prenesti v bazo in ga dodeliti želenemu postopku. Ko je urnik dodeljen, je postopek še enkrat potrebno prenesti v bazo, saj je na njem prišlo do spremembe. Za uspešno dokončanje avtomatizacije postopka, je potrebno le še zagnati urnik. To storimo v Control Centru pod zavihkom »Schedules«.



Slika 39 - Urnik izvajanja postopka PF_DNEVNI

DIPLOMSKA NALOGA :
FAKULTETA ZA MATEMATIKO IN FIZIKO

7 Zaključek

V tej diplomski nalogi smo se dodobra spoznali z ETL orodjem Oracle Warehouse builder, s katerim smo napolnili podatkovno skladišče namišljenega podjetja E-trgovina. Seveda smo se najprej morali seznaniti z osnovami podatkovnega skladiščenja in različnimi načini, kako pristopiti k izgradnji le-tega. Po preučevanju različnih pristopov smo se odločili, da bo Kimballov pristop primernejši, ker omogoča delni razvoj podatkovnega skladišča. Tako smo razvili del podatkovnega skladišča, ki zajema področje prodaje izdelkov. Če bi si želeli v podjetju E-trgovina razširiti podatkovno skladišče še z drugimi področji (npr. »Nabava«, »Kadri«, »Investicije« …), jim ta pristop to omogoča.

8 Literatura in viri

1. Kimball, R., The data warehouse toolkit. 1996, USA: John Wiley & Sons, Inc.
2. Kimball, R., The data warehouse lifecycle toolkit. 1998, USA: John Wiley & Sons, Inc.
3. Kimball, R., The data warehouse ETL toolkit. 2004, USA: Wiley Publishing, Inc.
4. Oracle warehouse builder user's guide. [PDF dokument] 15-4-2010; dostopno na: http://www.oracle.com/pls/db111/portal.portal_db?selected=6
5. A dimensional modeling manifesto. [Internetna stran] 6-8-2010; dostopno na: http://www.ralphkimball.com/html/articles_search/articles1997/9708d15.html
6. Oracle Warehouse Builder. [Internetna stran] 14-6-2010; dostopno na: http://www.oracle.com/technetwork/developer-tools/warehouse/overview/index.html
7. Oracle Warehouse Builder History. [Internetna stran] 23-8-2010; dostopno na: http://en.wikipedia.org/wiki/Oracle_Warehouse_Builder
8. Choosing Oracle Warehouse Builder. [Internetna stran] 9-9-2010; dostopno na: http://www.dbasupport.com/oracle/ora9i/ETL03_1.shtml

DIPLOMSKA NALOGA :
FAKULTETA ZA MATEMATIKO IN FIZIKO