VISOKA POSLOVNA ŠKOLA
STRUKOVNIH STUDIJA
ČAČAK

DIPLOMSKI RAD

Igradnja skladišta podataka

Mentor:_____ Student: _____
Profesor: _____ Br.Indeksa: _____

Sadržaj:

Uvod

Zadatak ovog diplomskog rada je analizirati postupke izgradnje skladišta podataka s posebnim naglaskom na Oracle-ova rješenja, te nakon izgradnje analizirati dobivene rezultate. Za izgradnju skladišta koristio sam Oracle-ov programski paket Oracle Warehouse Builder 3*i*, a kao izvor podataka za skladište poslužila mi je baza podataka uspjeha studenata za Zavod za osnove elektrotehnike i elektricka mjerenja.

Sam proces izgradnje skladišta je složen i odgovoran posao. Zadatak izgradnje skladišta podataka zahtijeva dobro poznavanje principa rada relacijskih baza podataka kao i poznavanje SQL-a i PL/SQL-a. Takoder zahtijeva dobro razumijevanje poslovnog procesa koji se želi pratiti. Osnovni problem u izgradnji skladišta podataka je pravilno odabrati podatke bitne za proces koji pratimo i raspoznati cinjenice (podatke koje cemo mjeriti kroz neko razdoblje). Model podataka koji je prikladan za dizajn skladišta podataka je dimenzijski model koji ce biti detaljno objašnjenu u ovom radu.

Proces izgradnje skladišta podataka se sastoji od nekoliko faza: faze razvoja logickog modela, faze definiranja procesa izvlacenja, transformacije i ucitavanja podataka u skladište podataka, te faze provjere ucitanih podataka. Ovisno o korištenom programskom paketu pri izradi skladišta podataka može postojati još koja faza kao što je u ovom slucaju faza konfiguracije i faza generiranja skripti. Ovaj rad je organiziran na taj nacin da se obradi svaka od tih faza u razvoju skladišta podataka.

Ovaj rad sam podijelio u dva dijela. U prvom dijelu koji se sastoji od dva poglavlja definirana su teorijska razmatranja o skladištenju podataka, dane su osnovne definicije i opisan je model podataka koji se koristi za izgradnju skladišta podataka. Takoder je opisan i proces izgradnje skladišta podataka. Drugi dio ovog rada je rezultat prakticnog rada i u njemu cu opisati kako izgraditi skladište u stvarnom svijetu. Za izgradnju sam koristio, Oracle-ov programski paket Oracle Warehouse Builder 3*i*, te su prvo opisani principi rada tog programskog paketa (3. poglavlje). U 4. i 5. poglavlju su opisani postupci u izgradnji logickog modela skladišta podataka kao i izrada mapiranja. U 6. poglavlju je opisana konfiguracija fizickih svojstava objekata u OWB-u, a u 7. poglavlju je opisano generiranje i pokretanje skripti, te testiranje ucitanih podataka.

I. DIO: TEORIJSKA RAZMATRANJA

1. Osnove skladištenja podataka

Skladištenje podataka je novi koncept koji se pojavio sredinom 90-tih godina 20. stoljeca. Razvio se zbog potrebe dobivanja informacija u kratkom vremenu, te služi kao potpora poslovnom odlucivanju. Skladište podataka sadrži golemu kolicinu podataka i omogucava da se na osnovu tih podataka dobiju kvalitetna izvješca koja pomažu odgovornim ljudima pri donošenju poslovnih odluka.

U ovom poglavlju cu dati kratki povijesni pregled razvoj skladištenja podataka kao i razloga koji su doveli do njegovog nastanka, definirat cu glavne pojmove vezane uz skladištenje podataka, definirati cu ciljeve te opisati model podataka koji se koristi u skladištima podataka.

1.1. Kratki povijesni pregled razvoja skladištenja podataka

U povijesnom pregledu razvoja skladištenja podataka treba poceti sa opisom kako se postupalo s podacima prije nastanka koncepta skladištenja. Kroz povijest sistemskog razvoja glavni naglasak je bio na unapredenju operacijskih sistema. Za takve sisteme nije prakticno zadržavati stare podatke neodredeno vrijeme, vec su se ti podaci uklanjali i spremali u arhive (vecinom su se spremali na magnetske trake, vrpce …). Takav nacin rada bio je tipican za sustave 70-tih godina koji su bili monolitni sustavi sa centraliziranim “mainframe” racunalom. Ipak takvi sustavi su glavni izvor podataka za analizu. Te sustave zovemo naslijedeni sustavi (engl. legacy systems).

80-tih godina dolazi do popularizacije osobnih racunala. Snaga tih racunala raste, uvode se nove mogucnosti, pojavljuju se graficka sucelja prema korisniku (engl. GUI – graphical user interface), jednostavnije rukovanje te stoga racunala postaju lakša za uporabu. Popularizacija osobnih racunala dovela je do toga da se poceo smanjivati jaz izmedu programera i krajnjih korisnika. Takvi sustavi su omogucili izvlacenje podataka iz naslijedenih sustava na osobna racunala. Razvili su se alati za izradu izvještaja i za analizu. Ipak ovakav nacin rada imao je manu koja se ispoljavala kroz fragmentaciju podataka na razna osobna racunala i oni su bili usmjereni prema odredenim svrhama. Takoder nije postojao standard za izvlacenje podataka na osobna

racunala. Ovakav nacin rada zahtjevao je od korisnika da poznaje strukturu dijela baze što je mnogim korisnicima bila velika zapreka.

Vrhunac sustava za analizu prije pojave skladištenja podataka bili su sustavi za potporu odlucivanju i izvršni informacijski sustavi. Iako ti sustavi zbog svoje visoke cijene nikad nisu našli široku primjenu, ipak su bili preteca skladištenja podataka.

Kao što smo vidjeli s podacima se kroz povijest postupalo razlicito. Na samom pocetku razvoja informatickih sustava podatke se cak uklanjalo sa sistema na posebne medije za pohranu podataka. Glavni uzrok za to je bila nedovoljno razvijena tehnologija. Tadašnja racunala su bila preslaba, premalog kapaciteta i nisu mogli zadovoljiti potrebe. Medutim napredak na polju elektronicke industrije doveo je do znatnog poboljšanja performansi sustava. Procesorska moc se neprekidno povecava, pojavljuju se napredne arhitekture procesora, ulazno/izlazni procesi (engl. input/output) se ubrzavaju, a što je najvažnije gustoca zapisa postaje veca, a time diskovi sve veceg kapaciteta i manjih dimenzija. Usprkos razvoju tih komponenti njihova cijena pada i mocni strojevi postaju pristupacni širem krugu ljudi, što je omogucilo drugaciji pristup pohrani podataka, kao i obradi tih podataka, a medu ostalim je omogucilo i razvoj koncepta skladištenja podataka.

Takoder se kao posljedica napretka pojavljuju sve snažnija osobna racunala koja omogucavaju razvoj klijent/server arhitekture kao i distribuiranog racunarstva. Programska podrška se takoder razvija i izgraduju se sve mocnije aplikacije koje se vrte na osobnim racunalima. Osobna racunala su za skladištenje podataka važna jer korisnici danas pristupaju podacima u skladištima podataka koristeci uglavnom osobna racunala i aplikacije na njima.

Još jedan faktor je utjecao na skladištenje podataka, a to je pojava koncepta Intraneta i korištenja web baziranih aplikacija. Putem Intraneta podaci u skladištu podataka postaju dostupni svima unutar kompanije.

Osim napretka na polju elektronicke industrije na razvoj skladištenja podataka je utjecala i sama poslovna priroda tj. promjene u poslovnoj prirodi.

Tijekom 90-tih su se dogodile velike promjene u svijetu. Komunizam se raspao, nastale su nove države koje su prešle na tržišno orjentiranu ekonomiju te su se tako stvorila nova tržišta. Ritam života se ubrzao, vrijeme je postalo izuzetno važno.

Javio se globalizacijski pokret, kompanije su prerasle granice država i pocele su se širiti po svijetu. U tom modernom nacinu rada informacija je postala izuzetno bitna i to ona informacija koja je isporucena na vrijeme, te se javila potreba za necim što se danas zove skladište podataka.

1.2. Osnovni pojmovi i znacajke skladišta podataka

1.2.1. Osnovni pojmovi

U ovom poglavlju cu navesti osnovne pojmove vezane uz skladištenje podataka, te cu dati i njihove definicije.

“Skladište podataka je baza podataka koja sadrži povijesne, nepromijenjive podatke koji su logicki i fizicki izvuceni iz raznih izvora. Ti podaci se u skladu s definiranim modelom ucitavaju u skladište i integriraju s postojecim podacima, a sve to u svrhu potpore poslovnom odlucivanju.” [4].

Kao što je vidljivo iz definicije skladište sadrži povijesne podatke i to detaljne i sumarne. Ti podaci dolaze iz raznih izvora, uglavnom operacijskih i transakcijskih baza. Bitno je naglasiti da je skladište podataka fizicki odvojeno i logicki transformirano od izvora podataka.

“Skladištenje podataka je proces integracije podataka o poslovanju neke organizacije u jednu bazu podataka iz koje krajnji korisnici mogu raditi izvješca, postavljati upite i analizirati podatke.” [4].

Treba reci da je skladištenje podataka proces koji ne završava inicijalnim ucitavanjem podataka, vec se skladište podataka osvježava novim podacima u nekim, više ili manje pravilnim, vremenskim intervalima (npr. svaki dan, tjedan, mjesec). Iz toga slijedi da je skladištenje podataka kontinuiran i dugotrajan proces.

Uz pojmove skladištenja i skladišta podataka cesto se još spominju pojmovi iskopavanja podataka (engl. data mining), OLAP (engl. On-Line Analytic Processing) i metapodaci.

“Iskopavanje podataka (engl. data mining) je proces automatskog otkrivanja prethodno nepoznatih obrazaca i odnosa medu podacima u bazi podataka.”[4].

OLAP (engl. On-Line Analytic Processing) obuhvaca skupa alata koji krajnjem korisniku pružaju potporu poslovnom odlucivanju, a temelje se na

dimenzijskom (višedimenzijskom) pristupu. Cilj takvog pristupa je omogucavanje korisniku da dobije uvid u znacenje podataka na osnovu kojeg onda korisnik donosi odluke. Takoder taj skup alata sadrži i alate koji korisnicima pomažu u prikazivanju podataka u obliku raznih grafova i tablica.

Metapodaci (podaci o podacima, engl. metadata) je izraz koji oznacava sekundarne, pomocne podatke koji sadrže informacije o podacima u skladištu podataka ili sadrže informacije kako te podatke najlakše obraditi. Metapodaci nam pomažu i u izvlacenju podataka i u rješavanju upita nad podacima.

1.2.2. Znacajke skladišta podataka

Osnovna ideja skladištenja podataka je da se u njemu pohranjuju podaci namjenjeni za izvodenje poslovne analize, a pristup tim podacima je najefikasniji ako su ti podaci odvojeni od podataka pohranjenih u operacijskim sistemima. Postoje mnogi razlozi za to razdvajanje koji su se tijekom vremena razvijali, ali što je bitno ti razlozi se razmatraju formalno prilikom izgradnje skladišta podataka.

Jedan od tih razloga za razdvajanje je cinjenica da podaci u skladište podataka mogu doci i iz više izvora (više operacijskih sistema). Ako podaci dolaze iz više izvora logicno je da se kombiniraju i spremaju samostalno i odvojeno od svojih izvora. Podaci se izvlace iz raznih izvora kako bi se mogli usporedivati. Mogucnost uspostavljanja i razumijevanja odnosa izmedu aktivnosti razlicitih organizacijskih grupa unutar kompanije se cesto istice kao najveca snaga skladišta podataka.

Takoder bitan razlog za odvajanje podataka od operacijskih sistema je cinjenica da se procesi obrade transakcije i analize podataka bitno razlikuju odnosno da postoji razlika izmedu transakcijskih (operacijskih) sistema i sistema za analizu. Transakcijski sistem (cesto se naziva i OLTP – On-Line Transaction Processing) pridaje najvecu važnost raspoloživosti i brzini obrade i ne smije se dozvoliti da analiza podataka dovede do degradacija performansi transakcijskog sistema. To je kljucni razlog razdvajanja. Transakcijski sistem i skladište podataka se takoder razlikuju i po podacima. Kod transakcijskog sistema ti podaci sadržavaju trenutne vrijednosti, oni su detaljni i izuzetno promijenjivi (tijekom svake sekunde se može obaviti i nekoliko tisuca transakcija koje mijenjaju vrijednosti tih podataka). Za razliku od transakcijskog sistema skladište podataka sadrži sumarne podatke koji su bitni za analizu, podaci su vremenski nepromijenjivi (jednom kad se ucitaju u

skladište više se ne mijenjaju). Sustavi se razlikuju i po namjeni. OLTP je namijenjen za vodenje poslovnog procesa dok je skladište namijenjeno za izvodenje procesa analize i izvještavanja. OLTP najcešce pristupa nekoliko zapisa odjednom dok skladište cita i do nekoliko milijuna zapisa istovremeno. OLTP koriste službenici za obavljanje svog posla dok skladište koriste analiticari i manageri za donošenje poslovnih odluka. OLTP i skladište su sustavi koji se razlikuju u svim bitnim znacajkama te je stoga logicno da ta dva sustava budu odvojena i fizicki i logicki.

	OLTP	SKLADIŠTE PODATAKA
Sadržaj podataka	Trenutne vrijednosti	Povijesni podaci
Vrijednost podataka	Vrlo promijenjivi podaci	Nepromijenjivi podaci
Namjena	Vodenje poslovnih operacija	Analiza i izvješcivanje
Jednica obrade	Transakcija	Upit
Korisnici	Službenici	Analiticari i manageri
Rasploživost	Vrlo bitna	Manje bitna
Izmjena podataka	Polje po polje	Nema izmjene
Radne karakteristike	Citanje/Pisanje	Citanje
Interakcija korisnika	Predodredena	Ad-hoc
Pristup zapisima	Nekoliko zapisa odjednom	Milijunima zapisa istovremeno
Fokus	Spremanje podataka	Dobivanje informacija

Slika 1.1. Razlike izmedu OLTP i skladišta podataka

Da bi se to razdvajanje postiglo potrebno je provesti logicku i fizicku transformaciju podataka s transakcijskog sistema u skladište podataka. Pritom je izuzetno bitno da svi podaci budu transformirani i integrirani u skladište podataka na konzistentan nacin. To znaci da se kroz cijelo skladište podataka moraju zadržati konzistentne konvencije imenovanja, konzistente mjerne jedinice varijabli, konzistentne strukture šifriranja, itd.

1.3. Ciljevi skladištenja podataka

Skladištenje podataka odnosno izgradnja skladišta podatka imaju svoj razlog i opravdanost. Razlog za pokretanje jednog takvog skupog i složenog projekta leži u cinjenici da ako se taj projekt dobro i strucno napravi, on omogucuje svojim korisnicima dobivanje kvalitetne informacije u trenutku što je u današnjim uvjetima poslovanja ne samo poželjno vec i neophodno. Svaki projekt, pa tako i projekt izgradnje skladišta podataka, mora zadovoljiti odredene zahtjeve (ciljeve). Neki od ciljeva koje skladišta podataka trebaju postici ili omoguciti su:

1. Skladište podataka mora omoguciti pristup podacima bitnim za neku organizaciju ili kompaniju
   - Manageri i analiticari moraju imati pristup do podataka koji su bitni za tu organizaciju. Taj pristup treba biti neposredan, brz, na zahtjev korisnika i mora omoguciti visoke performanse. Sve to korisnici moraju moci postici koristeci svoje osobno racunalo.
2. Podaci u skladištu podataka moraju biti konzistentni
   - Konzistenstnost znaci da ako dva korisnika traže isti podatak moraju dobiti isti odgovor iako su oni to tražili u razlicito vrijeme. Takoder znaci da ako podaci od jucer nisu do kraja ucitani korisnik mora biti upozoren.
3. Podaci se u skladištu podataka mogu kombinirati na sve moguce nacine (engl. dice and slice requirement)
   - To omogucava dimenzijski model
4. Skladište podataka nisu samo podaci, vec ono mora sadržavati i skup alata za postavljanje upita (engl. query tools), alata za analizu i predstavljanje informacije
   - Smatra se da je 60% potrebnog za uspjeh skladišta podataka sadržano u samim podacima, a 40% se odnosi na software za analizu.
5. Skladište podataka je mjesto gdje se objavljuju korišteni podaci
6. Kvaliteta podataka u skladištu je pokretac poslovnog restrukturiranja

1.4. Dimenzijski model

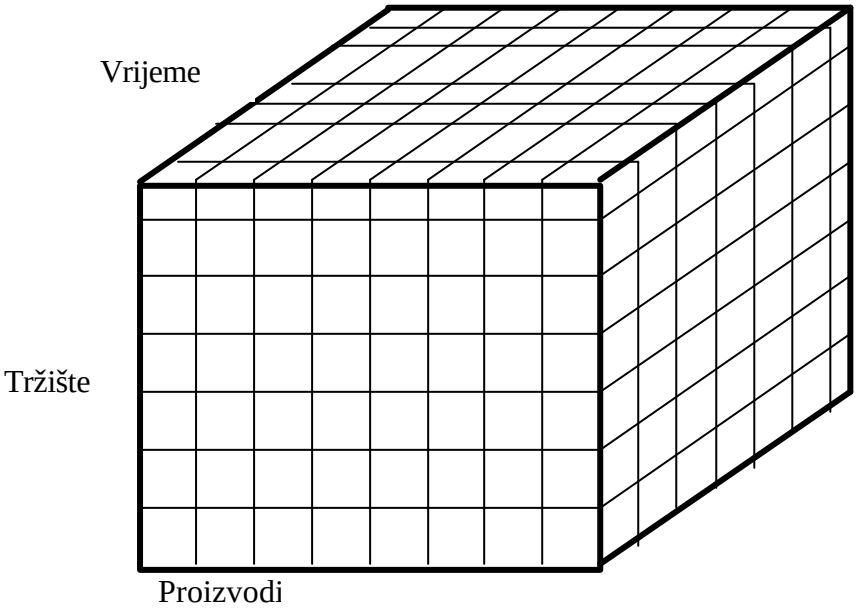
Kao što je vec spomenuto u prošlom poglavlju u transakcijskim sistemima se koristi relacijski model podataka koji je normaliziran i optimiziran za postizanje visokih brzina obrade. Takav model podataka se pokazao izvanrednim kada je rijec o transakcijskim obradama u kojima se dohvaca najviše nekoliko desetaka zapisa odjednom. Medutim za potrebe skladišta podataka, u kojima se dohvaca i do nekoliko milijuna zapisa istovremeno, taj model je neprihvatljiv. Problem leži u cinjenici da je relacijski model podataka normaliziran i kao takav je neuporabljiv za izvršavanje kompleksnih upita nad milijunima podataka. Zato se u skladištu taj model podataka zamijenjuje s dimenzijskim modelom koji je na višem stupnju apstrakcije od

relacijskog i zato je pogodniji za skladište podataka. Dimenzijski model ce biti detaljno obraden u ovom poglavlju.

1.4.1. Što je dimenzijski model

"Dimenzijski model je tehnika logickog dizajna koja teži prikazivanju podataka na standardiziran, intuitivan nacin koji omogucava pristup podacima velikom brzinom."[7].

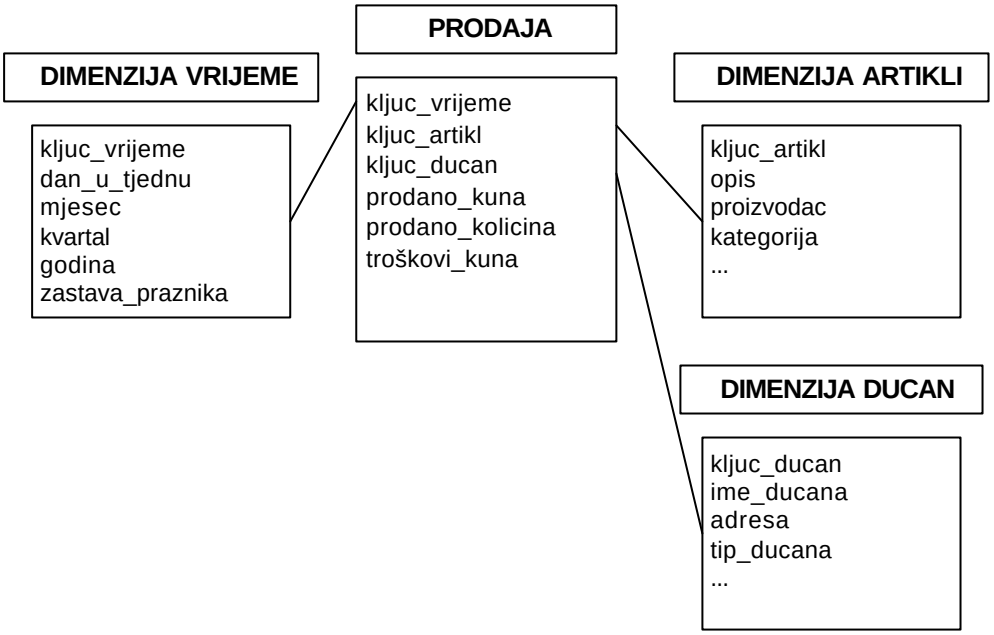
Dimenzijski model se najcešce prikazuje apstraktno kao kocka cije dimenzije predstavljaju dimenzije posla koji modeliramo, a podatak na presjeku tih dimenzija predstavlja podatak koji tražimo.



Slika 1.2. Prikaz dimenzijskog modela podataka u obliku kocke

Broj dimenzija u praksi može biti i veci od tri pa se onda govori o višedimenzionalnoj kocki. Gotovo u svim dimenzijskim modelima postoji dimenzija vremena. Organiziranje i spremanje podataka prema ovom modelu omogucuje korisnicima bolje razumijevanje podataka i omogucuje da korisnicka sucelja budu jednostavnija za korištenje, a izvedba upita na zadovoljavajucoj razini.

Struktura dimenzijskog modela se sastoji od jedne tablice sa složenim kljucem koje se naziva tablicom cinjenica (engl. fact table) i više tablica dimenzija (engl. dimensional tables) od kojih svaka ima jednostavan kljuc koji je dio složenog kljuca tablice cinjenica. Takva struktura se cesto zove zvijezda spoj (engl. star-join schema). Na slici 1.3. je prikazan tipican izgled dimenzijskog modela. O tablicama cinjenica i dimenzija bit ce malo kasnije više rijeci.



Slika 1.3. Prikaz tipicnog dimenzijskog modela

Dimenzijski model i relacijski model se razlikuju u mnogo cemu. Relacijski model je puno složeniji (tj. dijagram relacijskog modela je mnogo složeniji) od dimenzijskog modela. Mnogi dizajneri zato kažu da zbog toga dimenzijski model sadrži manje informacija i da se on koristi za sažetke više razine. Najveci autoritet u podrucju skladištenja podataka i jedan od pionira skladištenja podataka Ralph Kimball smatra da to nije tocno: “Osnovni odnos izmedu relacijskog i dimenzijskog modela je da se dijagram relacijskog modela razlaže u nekoliko dijagrama dimenzijskog modela. Dijagram relacijskog modela predstavlja svaki moguci poslovni proces u nekoj kompaniji i odnose izmedu njih i stoga je izuzetno kompleksan. Prvi korak u pretvaranju dijagrama relacijskog modela u dijagram dimenzijskog modela je razdvajanje poslovnih procesa i njihovo modeliranje zasebno. Drugi korak je nalaženje “many-to-many” odnosa i pretvaranje tih odnosa u tablice cinjenica. Ostatak se denormalizacijom pretvara u tablice dimenzija. Rezultirajuci dijagram dimenzijskog modela za relacijski model za veliko poduzece može imati 10-25 vrlo slicnih zvijezda spojeva od kojih svaki može imati 4-12 tablica dimenzija.”[7].

Uporaba dimenzijskog modela u skladištu podataka ima mnoge prednosti pred relacijskim modelom. Kao prvo, dimenzijski model je predvidljiv, standardiziran okvir. Korisnici, aplikacije, mogu pretpostaviti mnoge stvari o dimenzijskom modelu koje tada omogucuju razumljivija sucelja ili vecu efikasnost obrade. Druga prednost

leži u cinjenici da predvidljiv okvir zvijezda spoja odolijeva neocekivanim promjenama u ponašanju korisnika. Dimenzijski model je proširiv u smislu prihvacanja novih, neocekivanih podataka i novog dizajna. Sve tablice se mogu promijeniti dodavanjem novih redova, raspored im se može promijeniti, a pritom se stari podaci ne trebaju ponovno ucitati, a i aplikacije se ne trebaju mijenjati da bi podržale promjenu. Takoder snaga dimenzijskog modela je u tome da postoje standardizirani pristupi za rješavanje uobicajenih situacija u poslovnom svijetu. Svaka od tih situacija ima svoje dobre alternative (neke od tih situacija i modela su npr. sporo mijenjajuce dimenzije gdje se konstantna dimenzija kroz dugi niz vremena mijenja). Zadnja prednost leži u cinjenici da dimenzijski model omogucava kreiranje agregacijskih tablica koje ubrzavaju obradu. U zadnje vrijeme se to polje znatno razvilo pojavom odgovarajucih aplikacija i raznih utility-a (aggregation navigator).

1.4.2. Tablice cinjenica

Kao što je vec spomenuto dimenzijski model se implementira u strukturu znanu kao zvijezda spoj koja se sastoji od tablice cinjenica (engl. fact table) i dimenzijskih tablica (engl. dimensional tables). Tablica cinjenica je mjesto gdje se spremaju brojcani poslovni pokazatelji. Svaki od tih pokazatelja se nalazi negdje na presjeku svih dimenzija. Ti pokazatelji se nazivaju cinjenicama i otuda se tablica zove tablica cinjenica. Tipican primjer za cinjenicu je npr. atribut koji sadrži ukupnu vrijednost proizvoda prodanog odredeni dan.

Najbolje i najkorisnije cinjenice su: brojcane, kontinuirano vrednovane i zbrojive. Cinjenice trebaju biti brojcane jer su one pokazatelji nekog poslovnog procesa, a taj proces se iskazuje nekakvim iznosima (koliki je profit, koliko firma duguje, koliki su troškovi…).

Cinjenice su obicno kontinuirano vrednovane što znaci da mogu poprimati razlicite vrijednosti svaki put kad se mjere. Cinjenice i ne moraju biti kontinuirano vrednovane, vec se to pravilo o kontinuiranoj vrijednosti više koristi kao snažna preporuka dizajnerima skladišta podataka kako bi lakše razlucili cinjenice od dimenzijskih atributa.

Najbolje cinjenice su zbrojive i uvijek se teži k tome da cinjenice budu zbrojive. Razlog tomu je cinjenica da se pri gotovo svakom upitu prolazi kroz stotine, tisuce, pa cak i milijune zapisa kako bi se izgradio odgovor. Taj veliki broj zapisa se

može sažeti u nekoliko redova ako se cinjenice zbroje. Cinjenice u tablici cinjenica mogu još biti nezbrojive ili poluzbrojive. Poluzbrojive cinjenice se mogu zbrajati samo kroz neke dimenzije, dok se nezbrojive uopce ne mogu zbrajati kroz nijednu dimenziju, a to nije prihvatljivo za skladišta podataka.

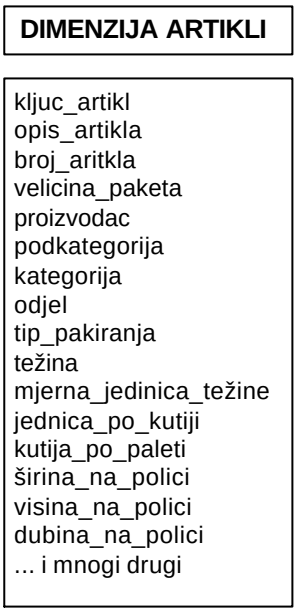
Kao i svaka tablica i tablica cinjenica mora imati kljuc. Kod nje se kljuc sastoji od više atributa. Dakle, tablica cinjenica ima složeni kljuc, i što je bitno on se sastoji od svih primarnih kljuceva dimenzijskih tablica. Zbog takve strukture u bazi nema složenih odnosa (najsloženiji je odnos jedna prema više tablica) te je stoga omogucena izvedba složenih upita. Tablica cinjenica je, uobicajeno, najveca tablica u skladištu podataka.

1.4.3. Dimenzijske tablice

Dimenzijske tablice spremaju podatke vezane za svaku pojedinu dimenziju. Dimenzije daju cinjenicama kontekst, one su prirodni poslovni parametri koji odreduju svaku cinjenicu. Dimenzije se opisuju u dimenzijskim tablicama koristeci iscrpne tekstualne opise. U dobro dizajniranom skladištu podataka dimenzijske tablice imaju vrlo velik broj atributa koji su tekstualnog oblika, a po vrijednosti su diskretni (poprimaju vrijednosti iz odredenog skupa vrijednosti), i koji kasnije služe kao izvor ogranicenja u upitima (engl. constraints) i naslova stupaca (engl. row header) u izvještajima. Jedna tipicna dimenzijska tablica izgleda kao na slici 1.4.

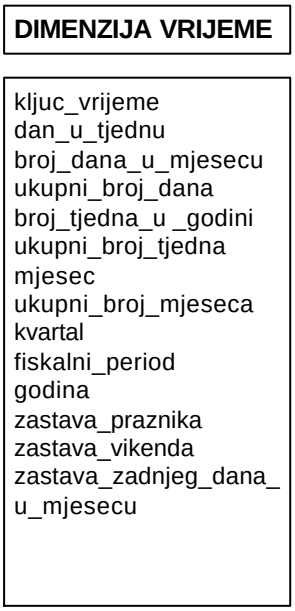
Dimenzijska tablica bi trebala imati što veci broj atributa jer se tako povecava broj ogranicenja u upitima, a time se povecava i kolicina informacija koja je korisniku dostupna. Dimenzijske tablice su denormalizirane radi jednostavnosti dizajna i ucinkovitijeg izvodenja upita. One su puno manje od tablica cinjenica pa normalizacija dimenzijskih tablica u svrhu uštede prostora nema nikakvog ucinka.

Upiti u skladištu podataka se obraduju tako da se prvo u dimenzijskim tablicama nadu sve vrijednosti kljuca koje zadovoljavaju postavljena ogranicenja, a zatim se od njih spoje sve moguce kombinacije složenih kljuceva koje ce se tražiti u tablici cinjenica. Nadeni podaci se sumiraju i grupiraju prema specifikaciji korisnika.



Slika 1.4. Jedna tipicna dimenzijska tablica

1.4.4. Dimenzija vremena



Dimenzija vremena je dimenzija koja je prisutna u svim skladištima podataka, zato što je svako skladište podataka vremenska serija snimaka stanja neke organizacije. Snimamo stanja transakcijskog sistema i spremamo ta snimljena stanja u skladište podataka kao niz slojeva podataka te je stoga svako skladište podataka vremenski niz. Nakon toga, analizirajuci podatke, kopamo kroz slojeve podataka kako bi shvatili kako je naše poduzece izgledalo u nekoj tocki vremena. Tipic na dimenzija vremena može izgledati kao na slici 1.5.

Slika 1.5. Dimenzija vremena

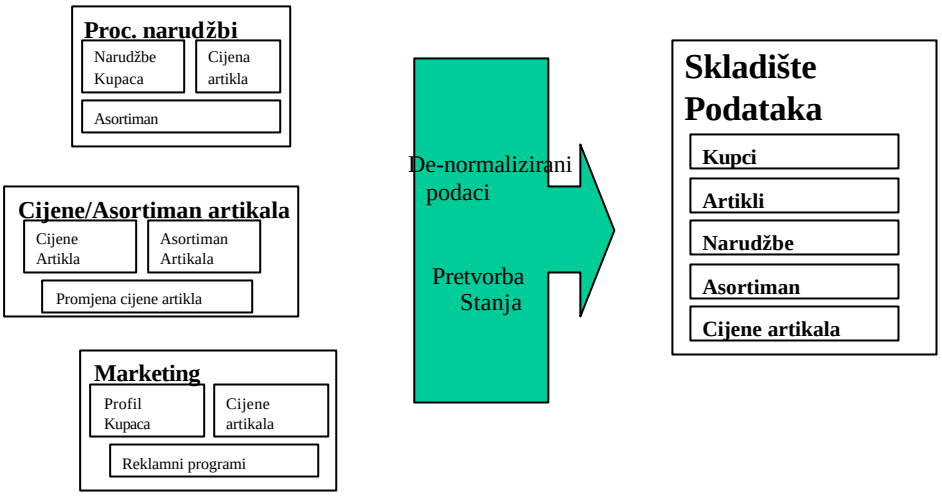
2. Izgradnja skladišta podataka

2.1. Proces izgradnje skladišta podataka

Da bi uopce došlo do izgradnje skladišta podataka potrebno je ispuniti neke zahtjeve. Kao prvo, treba postojati želja i potreba za skladištem podataka, te treba postojati netko tko ce gurati taj projekt. Sama organizacija mora biti spremna za skladište podataka, pri cemu se spremnost ocituje u postojanju poslovnog motiva, raspoloživosti podataka, spremnosti ljudi na promjene itd. Proces izgradnje zapocinje definiranjem korisnickih zahtjeva. Projektant u razgovoru s korisnicima saznaje njihove želje i potrebe. Nakon toga u razgovorima s administratorom operacijske baze saznaje strukturu baze i s kojim podacima raspolaže. Projektant mora pronaci ravnotežu izmedu želja korisnika i realnosti koju pružaju postojeci podaci.

Kada se analiziraju zahtjevi i postojeci podaci u operacijskoj bazi, prelazi se na izradu logickog modela skladišta podataka. Kao što sam vec spomenuo za skladišta podataka se koristi dimenzijski model podataka. Podaci koje cemo ucitati u skladište moraju se fizicki i logicki transformirati u odgovarajuci oblik. Pritom je izuzetno bitno da svi podaci budu transformirani i integrirani u skladište podataka na konzistentan nacin. To znaci da se kroz cijelo skladište podataka moraju zadržati konzistentne konvencije imenovanja, konzistente mjerne jedinice varijabli, konzistentne strukture šifriranja, itd.

Podaci se logicki transformiraju prilikom izvlacenja iz transakcijske baze i ucitavanja u skladište podataka. Podaci su u transakcijskoj bazi bili smješteni u sustavu visokih performansi koji sadrži relacijski model podataka optimiziran za brzo izvodenje. Podaci u skladištu podataka se modeliraju u dimenzijski model podataka koji je pogodan za izvodenje upita nad milijunima zapisa istovremeno. Takoder ti podaci moraju biti organizirani na nacin da omogucuju jednostavno proširenje i strukturirani na nacin da u njega mogu uci podaci iz više izvora. Takoder je izuzetno bitno da se model podataka slaže sa strukturom posla i da prati razvoj posla (i za to je dimenzijski model izuzetno pogodan). Podaci u skladištu podataka su uglavnom denormalizirani i oni osiguravaju staticki odnos u povijesnim podacima. Logicka transformacija prikazana je na slici 2.1.

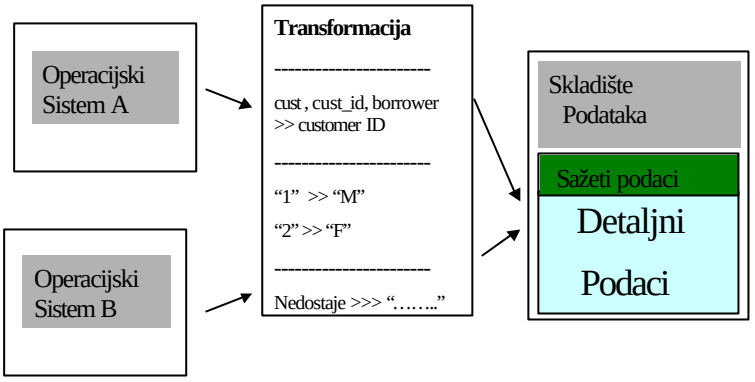


Slika 2.1. Logicka transformacija podataka

Fizicka transformacija (prikazana na slici 2.2.) homogenizira i procišcava podatke i obavlja nekoliko zadaca ciji je cilj osigurati razumijevanje pohranjenih podataka i njihovu konzistentnost u skladištu podataka. U transakcijskom sistemu za pojmove i imena atributa se koriste kripticka i teško razumljiva imena (npr. brkup) koja bi krajnjem korisniku bila teška za uporabu. Takoder razliciti izvori mogu imati i razlicita imena za isti pojam (npr. brojkup i brkup). Tijekom fizicke transformacije ti pojmovi i imena se transformiraju u jednoznacne, standardne poslovne izraze koji su razumljivi krajnjem korisniku (brkup je broj kupca).

Razliciti sustavi mogu imati za iste atribute razlicite tipove podataka i razlicit duljine (npr. jedan sistem za šifru artikla ima 12, a drugi 14 numerickih znakova). Fizickom transformacijom sve razlicite tipove podataka i duljine treba pretvoriti u jednoznacan, zahtijevani oblik.

Takoder cesti je slucaj da u razlicitim aplikacijama programeri na razlicite nacine šifriraju isti pojam ili za iste vrijednosti atributa upotrebljavaju razlicite izraze (tipican primjer je definiranje spola, jedna aplikacija spol šifrira kao M i F, druga kao X i Y, a treca kao M i Ž) i zato se fizickom transformacijom pretvaranje u zahtijevani oblik. Zadnja, ali ne najmanje važna zadaca fizicke transformacije je kako transformirati podatke koji su nekompletni, pokvareni ili cak nedostaju. Neki se takvi podaci mogu nadomjestiti nekom “default” vrijednošcu. Za one koji se ne mogu, to treba riješiti na druge nacine (upozoriti korisnika da nema tog podatka i slicno).



Slika 2.2. Fizicka transformacija podataka

Fizicka i logicka transformacija su dio procesa poznatijeg kao ETL (engl. extraction, transformation and loading) – izvlacenje, transformacija i ucitavanje podataka. Taj proces predstavlja pocetak implementacije skladišta podataka. Znaci prvo se analiziraju zahtjevi i podaci u izvorima, zatim se definira logicki model podataka, te nakon toga se prilazi implementaciji fizicke instance skladišta podataka.

Izvlacenje, transformacija i ucitavanje podataka se obavljaju pisanjem skripti u odgovarajucem programskom jeziku (SQL, PL/SQL,…), te izvodenjem tih skripti. Takve skripte se zovu mapiranja (engl. mappings). Danas postoje mnogi alati i programski paketi (kao što je i Oracle Warehouse Builder) koji omogucuju da se takve skripte “pišu” u grafickom okruženju, bez puno pisanja koda kao što cemo vidjeti u prakticnom dijelu. Tijekom ETL-a podaci se izvlace iz izvora, transformiraju se u odgovarajuci oblik, procišcavaju se, popravljaju se pogreške i zatim se ucitavaju i integriraju u skladište podataka.

Inicijalnim ucitavanjem podataka proces izgradnje skladišta nije gotov. Nakon što smo ucitali pocetne podatke naš posao ipak nije gotov jer treba osigurati ucitavanje novih podataka u nekim vremenskim intervalima. Tipican interval je dan tj. svakodnevno ucitavanje novih podataka u skladište. Taj proces se zove periodicko punjenje i osvježavanje podataka (engl. periodical loading and refreshing data). Posao projektanta je i da osigura da taj proces prolazi u redu. Proces osvježavanja podataka se u velikim sustavima automatizira. To znaci da ga pokrece i zaustavlja racunalo onako kako smo programirali, a provjeru podataka obavlja administrator skladišta podataka.

Nakon inicijalnog punjenja podataka i osiguravanja periodickog osvježavanja novim podacima, skladište je izgradeno, ali je skladište potrebno nadgledati i upravljati njime, te ako se javi potreba i restrukturirati. Korisnici ce upotrebljavati podatke iz skladišta. Iz toga možemo vidjeti da se proces skladištenja (ne izgradnje skladišta) sastoji od:

1. Izvlacenje, transformacija i ucitavanje podataka
2. Spremanje podataka i upravljanje podacima
3. Korištenje podataka kao potpora procesu poslovnog odlucivanja

2.2. Neformalna metodologija izgradnje skladišta podataka

Strucnjaci koji se vec neko vrijeme bave skladištima podataka pa tako i njihovom izgradnjom su utvrdili metodologiju koju možemo koristiti prilikom izgradnje skladišta podataka. Ipak, treba naglasiti da ta metodologija nije potpuno formalizirana, vec ona više predstavlja niz preporuka, smjernica koje bi trebalo pratiti prilikom izgradnje skladišta. Metodologija se sastoji od devet pitanja o kojima treba odluciti. Ta pitanja se još zovu i tocke odluke (engl. decision points). Tih devet tocaka odluke prilikom izgradnje cjelokupnog dimenzijskog skladišta podataka sastoje se od odlucivanja o slijedecem:

1. Koje poslovne procese treba modelirati, odnosno koje su tablice cinjenica
2. Što je srž svake tablice cinjenica
3. Koje su dimenzije svake tablice cinjenica
4. Koje su cinjenice bitne
5. Koji su dimenzijski atributi
6. Kako cemo pratiti dimenzije koje se sporo mijenjaju
7. Koje agregacije koristiti, koje su heterogene tablice, minidimenzije, nacini upita i ostali problemi fizickog spremanja podataka
8. Koliko dugo cemo cuvati podatke
9. S kojom se ucestaloscu podaci izvlace i ucitavaju u skladište

Ove odluke bi trebalo donijeti u navedenom redoslijedu. Ova metodologija je metodologija od vrha prema dnu (engl. top-down) zato što se pocinje identificiranjem

glavnih procesa u kompaniji o kojoj se prikupljaju podaci i onda se krece prema rješavanju detaljnih problema.

Najvažnija pitanja su na pocetku i o odgovorima na njih ovisi u najvecem dijelu uspjeh cijelog projekta. Najbitnije je dobro razluciti poslovne procese koje želimo pratiti i prepoznati prave cinjenice, jer o tome ovisi cijeli logicki model, a ako je on loš biti ce loše i skladište podataka.

Nakon cetvrte tocke odluke gotovi smo s logickim dizajnom i tada se pocinjemo baviti višom razinom fizickog dizajna (tocke 6 i 7). Nakon toga donosimo odluke koliko dugo cemo cuvati podatke i s kojom ucestaloscu cemo raditi redovito ucitavanje novih podataka.

Bitno je naglasiti da ove odluke nisu jednostavne i ne rade se iz nicega. Prije pocetka same izgradnje potrebno je provesti opsežne razgovore s krajnjim korisnicima i administratorima transakcijske baze kako bi spoznali korisnikove želje i potrebe i kako bi znali donijeti pravu odluku prilikom izgradnje.

Osim o ovim pitanjima treba razmišljati i o potrebnom hardware-u i odgovarajucoj programskoj podršci (kakvo racunalo, koje aplikacije za izvještavanje, itd.).

2.3. Sumarne tablice (agregacije)

Agregacije su sumarne tablice tj. tablice koje sadrže sumarne podatke. U novije doba se za takve tablice koristi izraz materijalizirani pogled (engl. materialized view). Te tablice sadrže vec pripremljene, sumarne podatke koji su rezultat neke obrade (npr. vec sumirane podatke, izracunate postotke i sl.). Takve tablice se spremaju u skladište podataka zajedno s ostalim tablicama i koriste se u svrhu poboljšavanja performansi sustava. Pravilnom uprorabom tih tablica postižu se znatna unapredenja u performansama.

Razlog zašto se spominju prilikom izgradnje skladišta je jasan. Osim logickg modela (dimenzijskih tablica i tablica cinjenica) projektant mora projektirati i agregacije koje danas predstavljaju znacajan dio skladišta. Projektant mora predvidjeti koje ce se sumarne tablice najviše koristiti, koji ce se upiti najviše postavljati nad bazom i za njih dizajnirati odgovarajuce agregacije.

Sumarne tablice ne moraju stalno postojati u skladištu. Naime one se cesto koriste da bi se riješili odredeni problemi, a kad se ti problemi riješe one više ne trebaju. Takoder se uvijek mogu dodavati nove tablice bez da to smeta radu skladišta. Sumarne tablice se automatski osvježavaju kada se podaci ucitaju u skladište i to na nacin kako ih konfiguriramo. Osvježavanje zna biti dugotrajan proces pa bi trebalo konfigurirati tablice da se osvježe na najbrži moguci nacin.

Osnovna funkcija takvih sumarnih tablica je da ubrzavaju izvodenje odredenih upita. Puno je brže odraditi upit na nekoliko stotina redaka nego na nekoliko stotina tisuca redaka. Za optimalno korištenje tih tablica potreban nam je poseban software - *aggregate navigator* (u Oracle-u se on zove Summary Advisor). Njegova uloga je da kad korisnik postavi upit on taj upit transformira kako bi se koristila najpovoljnija agregacija. On postaje važan dio skladišta podataka u današnjim sustavima jer olakšava rad s agregacijama i korisniku i administratoru. Da bi on pravilno radio potrebno je uz njega cuvati i odredenu kolicinu metapodataka. *Summary Advisor* osim što transformira upite, on takoder analizira koji se upiti najviše i najcešce koriste, te na osnovu te analize predlaže koje bi materijalizirane poglede trebalo dodati radi ubrzavanja izvodenja tih upita. Takoder, *Summary Advisor* predlaže i koje bi se materijalizirane poglede moglo izbrisati jer se više ne koriste. Na taj nam nacin *Summary Advisor* predstavlja olakšanje i pomoc pri nadgledanju rada skladišta podataka.

II. DIO: PRIMJER IZGRADNJE SKLADIŠTA PODATAKA UZ PRIMJENU PROGRAMSKOG PAKETA ORACLE WAREHOUSE BUILDER 3*i*

3. Princip rada Oracle Warehouse Buildera

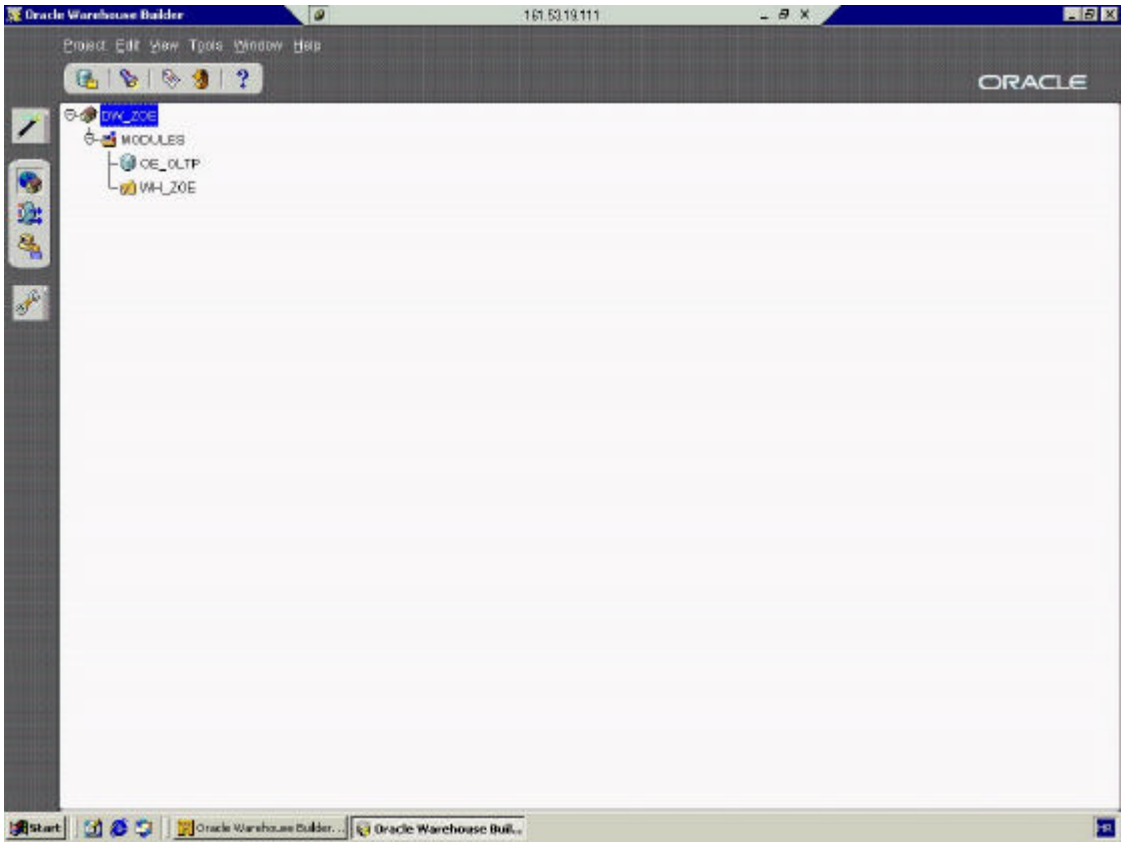
Da bi izgradili skladište podataka potrebno je, osim dizajna modela podataka, napisati i skripte, procedure, programe u raznim programskim jezicima (SQL, PL/SQL, …) ovisno o potrebi. Za to nam je potrebno odredeno vrijeme kojeg ionako uvijek imamo premalo. Da bi olakšali izgradnju skladišta podataka, mnogi proizvodaci su izradili programske pakete u kojima se može jednostavno i lako, koristeci graficko sucelje, napraviti logicki model i definirati skripte. Aplikacija, onda, umjesto nas generira kod po zadanim parametrima. To predstavlja veliku uštedu u vremenu razvoja i implementacije skladišta podataka. Jedan od takvih programskih paketa je i Oracle Warehouse Builder (skraceno OWB). U ovom poglavlju cu opisati osnovne elemente i filozofiju OWB-a, kao i njegove funkcije i nacin rada.

3.1. Uvod u Oracle Warehouse Builder

Oracle Warehouse Builder je programski paket koji je napravila Oracle Corporation. Trenutno najnovija inacica je 3i. OWB je programski paket koji služi za definiranje logickog modela, implementaciju skladišta podataka kao i za nadgledanje i kontrolu rada skladišta podataka. To je integrirani skup programskih rješenja koji nam omogucava lakše dizajniranje i izgradnju skladišta podataka, ali i kasniju kontrolu rada i nadgledanje skladišta podataka.

OWB programski paket sastoji se od OWB repozitorija, OWB klijenta i OWB *Runtime*-a. Osim tih proizvoda za potrebe skladišta podataka potreban nam je i *Oracle Enterprise Manager*, te neki alat za generiranje izvještaja. Da bi instalacija bila uspješna potrebno je instalirati ove produkte pravilnim redoslijedom. To znaci da se prvo treba instalirati baza (ako vec ne postoji), zatim se instalira repozitorij u tu odgovarajucu bazu podataka, te se klijent instalira na korisnikovo osobno racunalo (može biti i više korisnika koji rade u OWB-u). OWB *Runtime* se instalira zadnji i on služi za poslove nadgledanja.

Korisnik pristupa repozitoriju preko OWB klijenta. OWB klijent predstavlja aplikaciju u kojoj korisnik obavlja sav posao, te sprema taj posao u repozitorij. Prilikom pokretanja OWB klijenta prvi put potrebno je dati informacije o imenu racunala na kojem je repozitorij, broju porta, te Oracle SID. Takoder je potrebno unijeti svoje korisnicko ime i lozinku. Prilikom pokretanja je takoder potrebno izabrati projekt na kojem cemo raditi (prilikom prvog pokretanja postoji samo prazan projekt nazvan *My Project* koji se može odabrati), ali se kasnije može prebaciti na drugi projekt. Ako smo dali dobre podatke otvara se glavni prozor OWB klijenta koji izgleda kao na slici 3.1.



Slika 3.1 Glavni prozor Oracle Warehouse Buildera

Sva akcija korisnika se odvija u grafickom sucelju koje je standardno za današnje aplikacije. Korisnikove akcije i promjene koje on unese na ekranu, ne zapisuju se automatski u repozitoriju vec je te promjene i akcije potrebno potvrditi pritiskom na tipku *Commit*. Tek tada napravljene promjene postaju važece i unose se u repozitorij.

3.2. Osnovni elementi Oracle Warehouse Buildera

Osnovni element Oracle Warehouse Buildera je projekt. Projekt se definira kao struktura repozitorija u kojoj se cuvaju formalni opisi koji definiraju skladište podataka i u kojoj OWB sprema generirane skripte korištene pri implementaciji i ucitavanju podataka. Projekt je, dakle, osnovna jedinica u Oracle Warehouse Builderu.

Svaki projekt se sastoji od jednog ili više izvorišnih modula (engl. source module) i jednog ili više odredišnih ili skladišnih modula (engl. target module, warehouse module). Odredišni ili skladišni modul je mjesto unutar OWB projekta koje organizira i sprema definicije potrebne za logicku shemu skladišta. On sadrži definicije za dimenzije, tablice cinjenica, materijalizirane poglede, obicne poglede, tablice, te za mapiranja i transformacije. Izvorišni modul je mjesto unutar OWB projekta koje organizira i sprema definicije relacijskih baza ili obicnih datoteka (engl. flat files) koje služe kao izvori podataka za skladište podataka. Definicije relacijskih baza se mogu uvesti (engl. import) iz bilo koje baze podataka (ne samo Oracle-ove). OWB koristi takozvane softverske integratore (engl. software integrators) za citanje definicija i izvlacenje podataka iz izvora. Ovisno o izvoru koristit ce se odgovarajuci integrator.

3.3. Nacin rada Oracle Warehouse Buildera

Filozofija i nacin rada Oracle Warehouse Buildera se u pocetku cini neobicnim, ali s vremenom sam shvatio da je nacin rada potpuno logican i u skladu s današnjim trendovima. Osnovni princip rada je da je izgradnja skladišta podataka podjeljena u tri dijela. Prvi dio je potpuna logicka definicija koja osim definicije logickog modela obuhvaca i logicku definiciju mapiranja podataka iz izvora. Drugi dio predstavlja konfiguraciju svih objekata definiranih na logickoj razini. Završni dio predstavlja generiranje i pokretanje skripti za stvaranje logickog modela (dimenzija, tablica cinjenica,…), te generiranje i pokretanje skripti za izvlacenje, transformaciju i ucitavanje podataka iz izvora u skladište podataka.

Logicka definicija zapocinje stvaranjem izvorišnih i odredišnih modula. Nakon definiranja izvorišnog modula, potrebno je uvesti definicije relacijske baze koja nam služi kao izvor podataka. Ako imamo više izvora, potrebno je stvoriti više izvorišnih modula (za svaki izvor podataka, potreban nam je jedan izvorišni modul).

Nakon definiranja odredišnog modula, potrebno je definirati, unutar samog modula, dimenzije, tablice cinjenica, materijalizirane poglede,… prema našem logickom modelu podataka. Takoder unutar odredišnog modula definiramo svoje transformacije i mapiranja. Osim naših transformacija u OWB-u vec postoje standardne funkcije i procedure koje možemo koristiti u svojim mapiranjima.

Nakon što smo kreirali definicije objekata potrebnih za logicki model, nakon što smo kreirali vlastite transformacije i mapiranja, i ucitali definicije izvora gotovi smo s logickom definicijom. Logicka definicija je zapisana u repozitorij, ali još nije stvoren nijedan objekt, nijedna tablica, niti je stvorena ijedna skripta. Da bi smo stvorili fizicku instancu našeg skladišta prvo je potrebno konfigurirati fizicka svojstva svakog modula, objekta, tablice, svakog mapiranja i operatora unutar mapiranja. Na taj nacin definiramo kako ce se naš logicki model fizicki kreirati, kako ce se naše skripte izvoditi itd.

Poslije konfiguracije fizickih parametara potrebno je generirati skripte za kreiranje raznih tablica, te generirati skripte za izvlacenje, transformaciju i ucitavanje podataka iz izvora u skladište. Generirane skripte zatim treba fizicki spremiti u bazu podataka i nakon toga pokrenuti. Na taj nacin se stvara fizicka instanca skladišta podataka i ucitavaju se podaci u njega. Ucitavanjem podataka skladište podataka je izgradeno.

4. Izrada logickog modela skladišta podataka

Prvi korak u izgradnji skladišta podataka je definicija zahtjeva koje skladište podataka mora ispuniti, te izrada logickog modela koji ce omoguciti zahtjevanu funkcionalnost. Model podataka koji se upotrebljava za izradu logickog modela skladišta podataka je dimenzijski model. Zahtjeve koje skladište podataka mora ispuniti definiramo u razgovoru s korisnicima koji ce to skladište upotrebljavati, te analizirajuci strukturu operacijske baze podataka koja ce služiti kao izvor podataka za skladište. Kroz razgovor s korisnicima saznajemo što njima treba i što ih zanima, a analizirajuci strukturu operacijske baze saznajemo tocno s kojim podacima raspolažemo. Na osnovu tih saznanja trebamo izraditi logicki model skladišta podataka koji ce zadovoljavati zahtjevima korisnika, a biti ce ga moguce realizirati s postojecim podacima u operacijskoj bazi.

Moj prakticni rad je izgraditi skladište podataka za Zavod za osnove elektrotehnike i elekticka mjerenja koji ce pratiti uspjeh studenata na ispitima sa tog zavoda. U daljnjem tekstu cu kroz primjere pokazati kako se jedno skladište podataka razvija i implemetira koristeci OWB.

4.1. Prikupljanje pocetnih informacija i definiranje zahtjeva

Kao što sam vec rekao proces izgradnje skladišta zapocinje prikupljanjem informacija od korisnika i definiranjem zahtjeva koje skladište mora zadovoljiti. Da bi se ti zahtjevi pravilno definirali potrebno je i dobro poznavanje poslovnog procesa koji se modelira od strane dizajnera skladišta podataka.

U ovom slucaju poslovni proces je polaganje ispita sa Zavoda za osnove elektrotehnike i elektricka mjerenja i sa Zavoda za telekomunikacije od strane studenata. Studenti upisuju odredeni predmet (predmeti koji se prate su: Osnove elektrotehnike 1, Osnove elektrotehnike 2, i Organizacija obrade podataka) koji se predaje u nekom semestru školske godine. Prilikom upisa studenti se razvrstavaju u grupe. Za svaku grupu je odredeno koji nastavnik drži predavanja, koji nastavnik drži auditorne vježbe, laboratorijske vježbe, itd. Za pohadanje nastave, izvršavanje laboratorijskih vježbi, izradu prakticnih zadataka, studenti dobivaju bodove (npr. za pohadanje nastave od 0 do 4 boda, a za laboratorijske vježbe od 0 do 15 bodova, za

konstrukcijske zadatke od 0 do 15 bodova). Tijekom semestra studenti pristupaju kontrolnim zadacama (iz Osnova elektrotehnike). Studentima koji skupe dovoljan broj bodova (više od 25 za Osnove elektrotehnike 1) iz svih ovih kategorija ponudi se ocjena i oni su oslobodeni pismenog dijela ispita i mogu pristupiti samo usmenom dijelu ispita. Ako nisu zadovoljni ponudenom ocjenom mogu pristupiti pismenom dijelu ispita. Studenti koji nisu zadovoljili moraju izaci na pismeni dio ispita. I na kontrolnim zadacama i na pismenom dijelu ispita, na zadatke se odgovara zaokruživanjem jednog od ponudenih odgovora. Da bi se prošao pismeni ispit potrebno je skupiti dovoljan broj bodova (za Osnove elektrotehnike više od 14 bodova). Studenti koji produ pismeni dio izlaze na usmeni dio ispita. Oni koji produ i usmeni dio ispita prošli su cijeli ispit, a oni koji padnu moraju ponovno i na pismeni ispit. Studenti imaju pravo izaci 8 puta na ispiti iz pojedinog predmeta s tim da ako padnu cetvrti put (komisija) moraju ponovno upisati taj predmet, a ako padnu osmi put gube pravo studiranja.

Poznavajuci proces možemo pretpostaviti neke zahtjeve, ali ipak najbolji uvid u potrebe dobivamo razgovarajuci s buducim korisnicima. Razgovor s korisnicima se obicno ponavlja nekoliko puta kako bi se potpuno razumijele potrebe i zahtjevi koje nam oni iskazuju. U ovom konkretnom slucaju “korisnike” je predstavljao mr.sc. Boris Vrdoljak, asistent na Zavodu za osnove elektrotehnike i elektricka mjerenja. S njim sam razgovarao više puta o potrebama i zahtjevima koje bi buduce skladište podataka trebalo ispuniti. Iz tih razgovora proizašlo je mnogo pitanja na koje bi naše skladište trebalo dati odgovore:

- Koliko je studenata prošlo na kojem roku ovisno o njihovom uspjehu na labosu, u pohadanju nastave…?
- Koliko studenata prode iz prvog puta, koliko iz drugog, itd.?
- Koliko studenata prode do odredenog vremena (do veljace, do travnja, lipnja,…)?
- Koliko se studenata oslobodi kolokvija?
- Koja je razdioba ocjena na pojedinom roku, ukupno i po grupama?

- Koliko se studenata oslobodi kolokvija, ali ipak izade na pismeni?
- Kako prolaze ponavljaci u odnosu na one koji su prvi put upisali taj predmet?
- Koja je razdioba odgovora po zadacima po roku?
- Koji je odgovor na kojem zadatku bio najnetocniji (gdje su studenti najviše griješili)?
- Kako su studenti prolazili na kolokviju (bodovi) u ovisnosti o uspjehu na labosu i nastavi?

Ova pitanja u stvari predstavljaju zahtjeve. Skladište podataka mora moci odgovoriti na sva ova pitanja. Stoga sam morao napraviti logicki model skladišta koji je u mogucnosti udovoljiti tim zahtjevima, ali sam isto tako morao analizirati operacijsku bazu kako bih provjerio da li uopce postoji mogucnost odgovoriti na ova pitanja tj. da li postoje svi potrebni podaci u operacijskoj bazi. U ovom slucaju odgovor je bio "da".

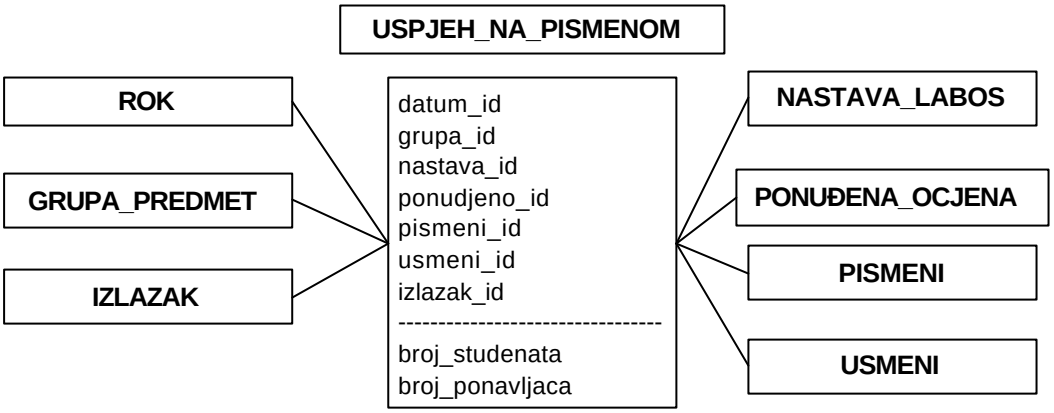
4.2. Logicki model skladišta podataka

Kao što je vec receno za logicki model podataka se upotrebljava dimenzijski model. U dimenzijskom modelu središnja tablica je tablica cinjenica u koju se spremaju podaci koje pratimo i mjerimo. Pravilan odabir zrnatosti tablice cinjenica je kljucan za uspjeh projekta izgradnje skladišta podataka. U poglavlju 2.2. je takoder opisana metodologija izrade skladišta podataka koju preporucuju strucnjaci. U njoj je takoder vidljivo da je odabir prave tablice cinjenica odnosno zrnatosti te tablice kritican (kljucan) dio izgradnje. Ako smo sad krivo napravili sve što slijedi ce isto biti krivo.

4.2.1. Tablice cinjenica

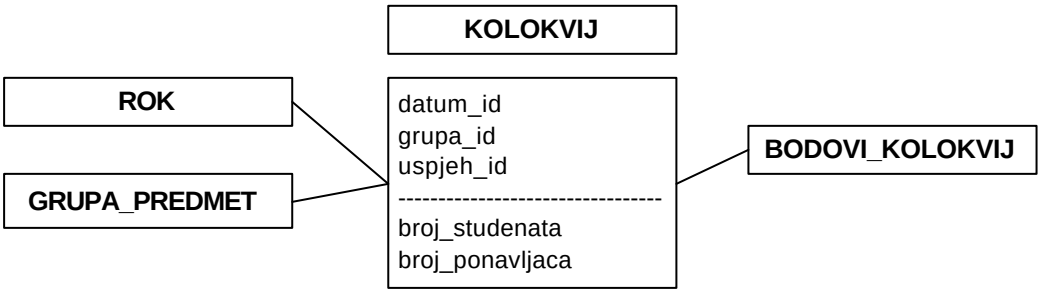
Svaka tablica cinjenica treba odgovarati jednom poslovnom procesu kojeg pratimo. Pošto u našem slucaju ima više razlicitih procesa, imat cemo i više tablica cinjenica. Pismeni ispit se bitno razlikuje od kontrolne zadace, a isto tako se pracenje pismenog ispita i kontrolne zadace bitno razlikuju od pracenja kako su studenti odgovarali na zadatke. Zbog svega navedenog odlucio sam se za cetiri tablice cinjenica.

Prva tablica cinjenica se zove USPJEH_NA_PISMENOM (slika 4.1). Kao što se vidi po imenu u toj tablici se prati uspjeh studenata na pismenim ispitima. Kratki opis ove tablice cinjenica je: “Broj studenata/ponavljaca koji su na odredenom roku iz odredenog predmeta postigli neki uspjeh (pismeni i usmeni ispit), a imali su ponudenu neku ocjenu, i odredeni broj bodova iz pohadanja nastave i laboratorijskih vježbi, te im je to bio n-ti izlazak na ispit.” Ovom tablicom cinjenica se, znaci, prati uspjeh na pismenim ispitima ovisno o svim ostalim mogucim parametrima.



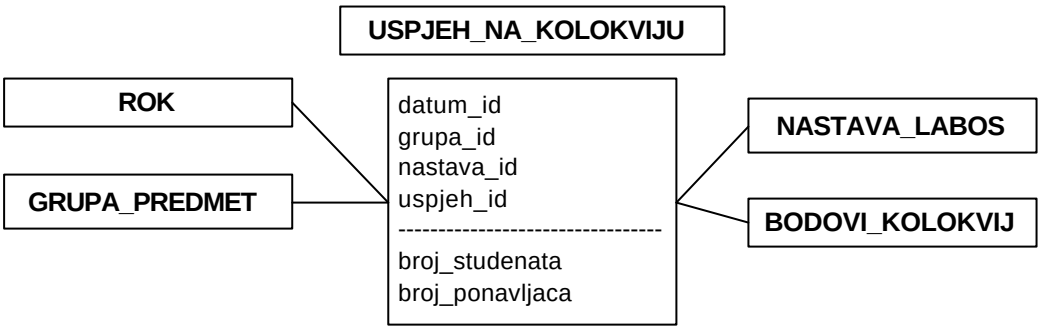
Slika 4.1. Tablica cinjenica USPJEH_NA_PISMENOM

Druga tablica cinjenica se zove KOLOKVIJ (Slika 4.2). Ona prati uspjeh studenata na kontrolnim zadacama, ali bez pokazivanja ovisnosti o postignutim bodovima iz nastave i laboratorijskih vježbi. Razlog tomu je cinjenica da ti bodovi još nisu podijeljeni prilikom pristupanja studenata kontrolnim zadacama. Opis ove tablice cinjenica je: “Broj studenata/ponavljaca koji pripadaju nekoj grupi,a na odredenom roku su iz odredenog predmeta postigli neki uspjeh (broj bodova).” Ova tablica cinjenica se popunjava neposredno nakon obrade rezultata kontrolnih zadaca i u nju ulaze samo podaci o studentima koji su pristupili kontrolnoj zadaci.



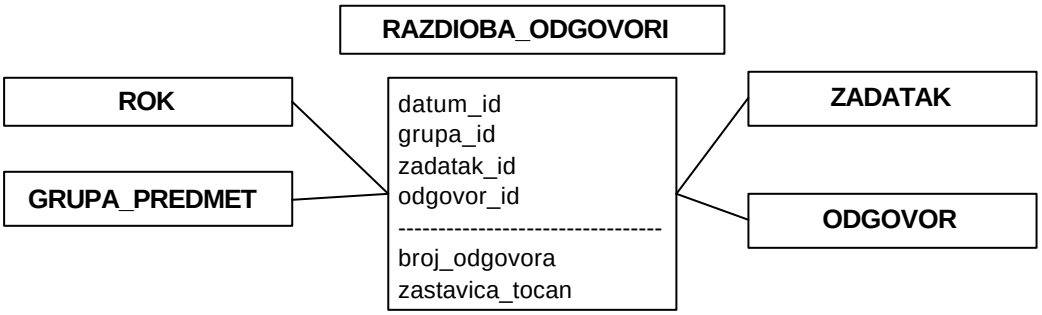
*Slika 4.2.*Tablica cinjenica KOLOKVIJ

Treca tablica cinjenica je slicna prethodnoj, a zove se USPJEH_NA_KOLOKVIJU (Slika 4.3.). Ona prati uspjeh na kontrolnim zadacama svih studenata i to u ovisnosti o postignutim rezultatima iz pohadanja nastave i obavljanja laboratorijskih vježbi. Ova tablica cinjenica se popunjava na kraju semestra kada su poznati postignuti bodovi iz relevantnih kategorija (nastava, laboratorijske vježbe, kontrolne zadace,…). Srž ove tablice cinjenica je: "Broj studenata/ponavljaca koji su na nekoj kontrolnoj zadaci iz nekog predmeta imali odredeni broj bodova, a iz nastave i laboratorijskih vježbi imaju n bodova."



Slika 4.3. Tablica cinjenica USPJEH_NA_KOLOKVIJU

Posljednja tablica cinjenica je RAZDIOBA_ODGOVORI (Slika 4.4.). Ona prati razdiobu odgovora po zadacima na nekom roku. Srž te tablice cinjenica je: "Broj nekog odgovora po grupi, po zadatku po roku iz nekog predmeta." Ova tablica cinjenica je zadužena za pracenje kako su studenti odgovarali na zadatke po rokovima.



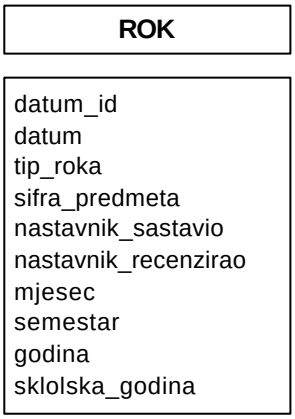
Slika 4.4. Tablica cinjenica RAZDIOBA_ODGOVORI

Zbog razlicitosti procesa koje trebamo pratiti, potreban nam je ovolik broj tablica cinjenica. Svaka tablica cinjenica prati posebne procese i sve zajedno daju cjelovitu sliku cjelokupnog procesa.

4.2.2. Dimenzijske tablice

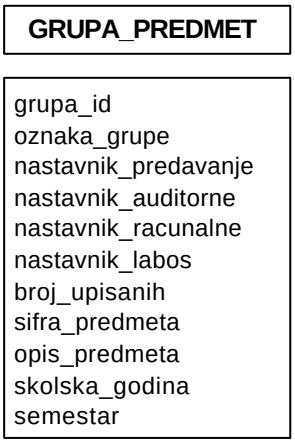
Kao što se vidi imamo deset dimenzijskih tablica: ROK, GRUPA_PREDMET, NASTAVA_LABOS, BODOVI_KOLOKVIJ, PISMENI, USMENI, PONUDJENA_OCJENA, IZLAZAK, ZADATAK i ODGOVOR. Tablice cinjenica imaju neke zajednicke dimenzije, a neke su posebne za svaku tablicu cinjenica. Pošto ima deset dimenzija, necu opisivati detaljno sve vec su opisati samo dvije koje su zajednicke svim tablicama cinjenica.

Zajednicke dimenzije su ROK i GRUPA_PREDMET. Dimenzija ROK opisuje jedan rok iz nekog predmeta. Izgleda kao na slici 4.5. Dimenzija ROK u našem skladištu predstavlja dimenziju vremena iako ona nije u pravom smislu dimenzija vremena. Ona nam omogucuje pracenje uspjeha studenata kroz vrijeme, a vremenski intervali su rokovi.



Slika 4.5. Dimenzija ROK

Dimenzija GRUPA_PREDMET opisuje grupe iz predmeta. Za svaku grupu su odredeni nastavnici koji drže predavanja, auditorne, vode laboratorijske i racunalne vježbe. Svaka grupa ima svoju oznaku i svaka grupa pripada odredenom predmetu. Niz grupa se svrstava u predmet. Dimenzija GRUPA_PREDMET prikazana je na slici 4.6.



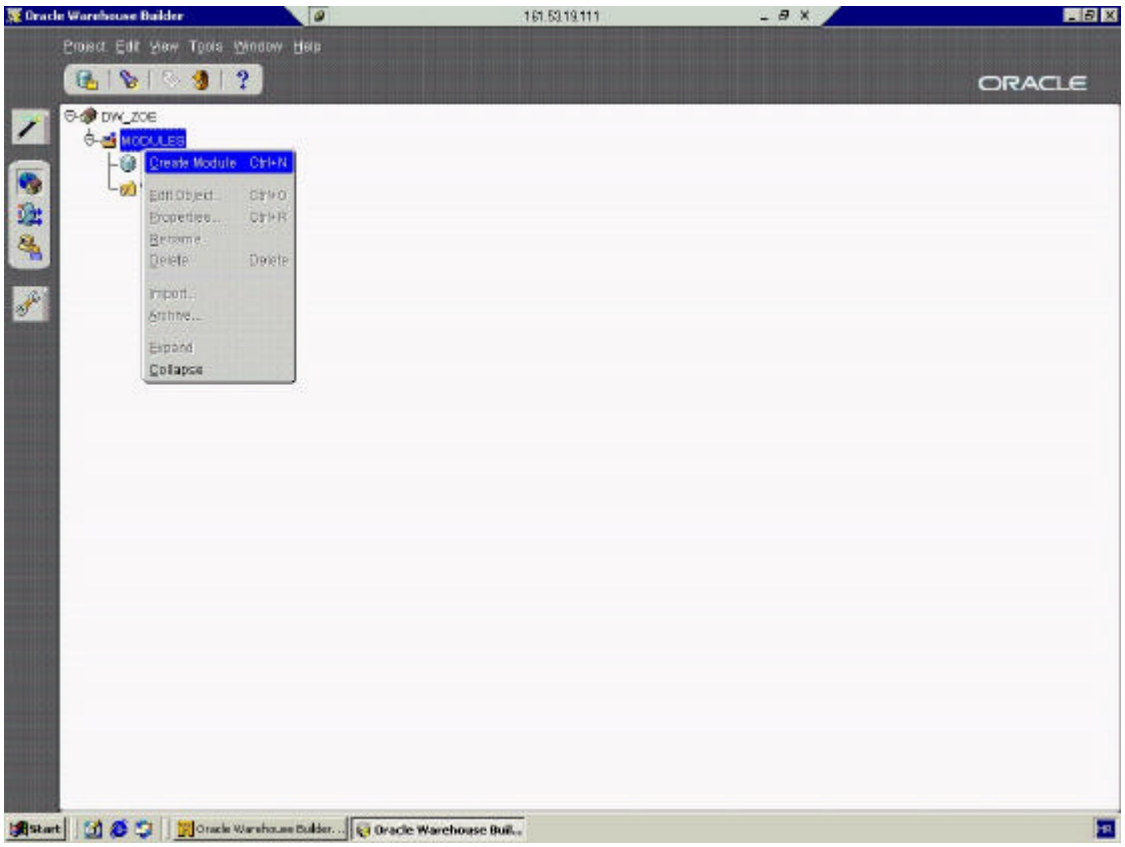
Slika 4.6. Dimenzija GRUPA_PREDMET

4.3. Izgradnja logickog modela skladišta u Oracle Warehouse Builderu

Definicija logickog modela u Oracle Warehouse Builderu se sprema u odredišni modul (target module, warehouse module), te prije kreiranja te definicije potrebno je napraviti odredišni modul u koji cemo spremiti naš logicki model.

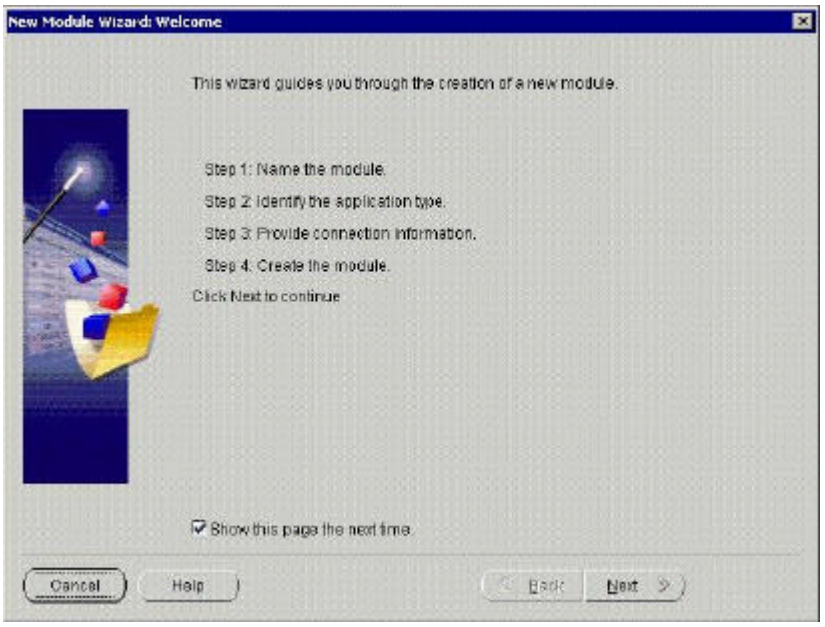
4.3.1. Stvaranje odredišnog modula

Za kreiranje svega, pa tako i odredišnog modula, u OWB-u postoje takozvani *wizardi*. *Wizardi* nas vode korak po korak u procesu stvaranja, od nas zahtjevaju potrebne podatke, te na osnovu tih podataka stvaraju traženi objekt. Da bi kreirali odredišni modul potrebno je pokrenuti *New Module Wizard*. On se pokrece tako da se odabere projekt u koji želimo spremiti taj modul, te pritiskom desne tipke miša dobijemo padajuci izbornik, na kojem odaberemo opciju *Create Module* (Slika 4.7.).



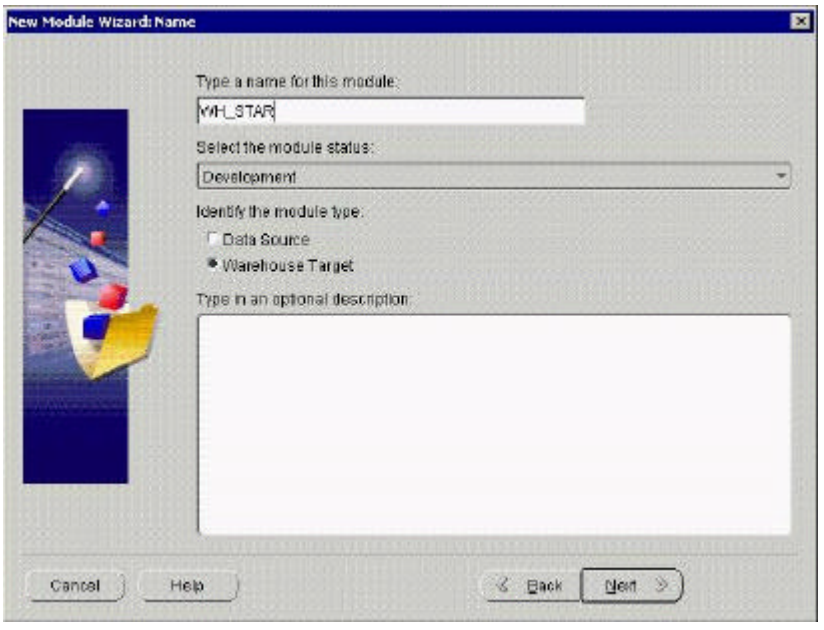
Slika 4.7 Pokretanje New Module Wizarda

Pokretanjem *New Module Wizarda* otvori se pocetni prozor koji sadrži kratki uvod i opis koraka kroz koje treba proci, te koja nas upozori koje cemo sve podatke trebati dati. (Slika 4.8.).



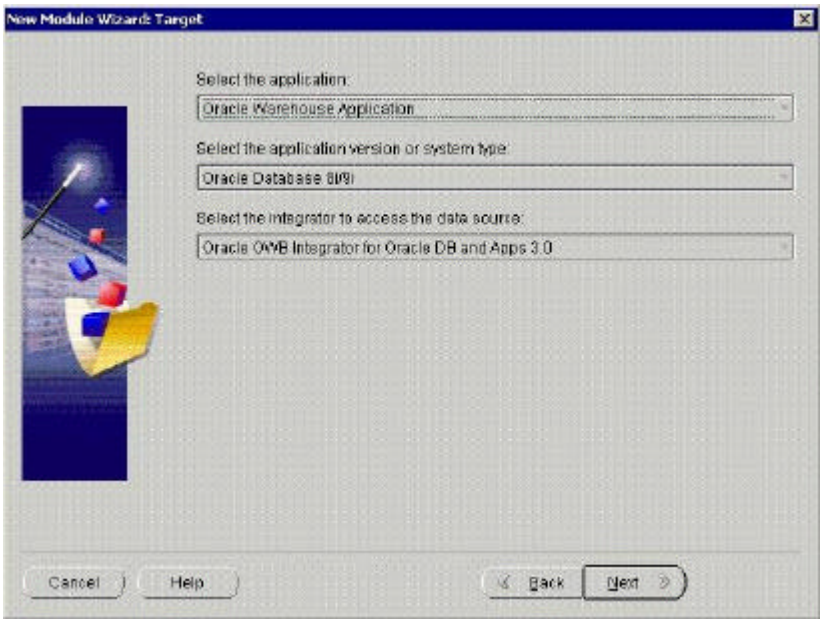
Slika 4.8. Uvodni prozor *New Module Wizarda*

U slijedecem koraku moramo imenovati modul, odrediti da li je on odredišni ili izvorišni modul, odrediti mu namjenu (za razvoj, za provjeru kvalitete ili za produkciju), te po želji možemo ukratko opisati modul. (Slika 4.9.). U našem slucaju odredili smo da je odredišni modul i da mu je namjena razvoj.



Slika 4.9. New Module Wizard: Korak 1

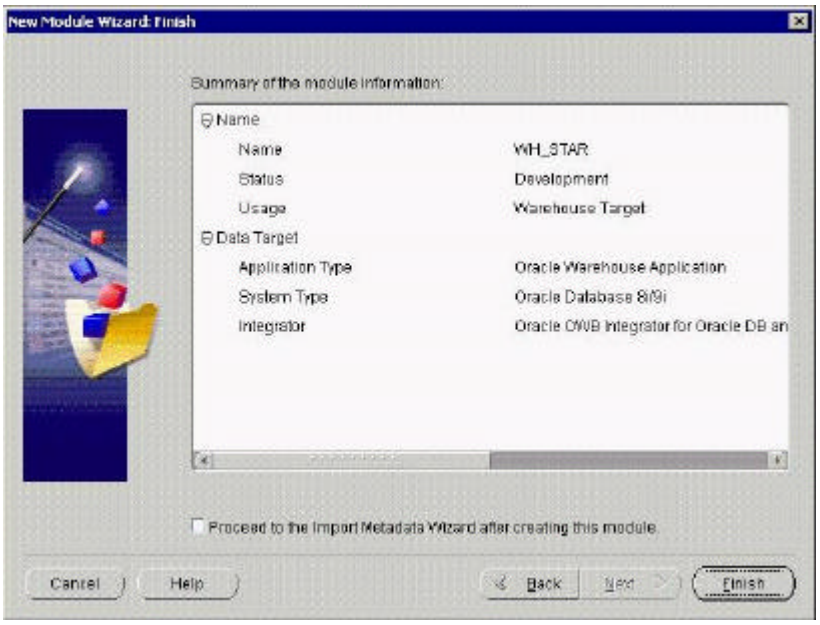
Slijedeci korak je odrediti koja ce aplikacija koristi ovaj modul, te koji ce se softverski integrator koristiti za pristupanje podacima. Ovi podaci se unose tako da se izabere jedna od ponudenih opcija sa liste. (Slika 4.10.).



Slika 4.10. New Module Wizard: Korak 2

Treci korak u stvaranju odredišnog modula je davanje informacija o linku prema bazi podataka. Ovaj korak nam treba samo ako cemo importirati definicije iz neke druge baze podataka što u nama u ovom slucaju ne treba jer cemo sami definirati svoj logicki model. Stoga preskacemo ovaj korak.

Završni prozor (slika 4.11.) nam prikazuje sažetak svih informacija koje smo unijeli tako da možemo još jednom provjeriti tocnost i da li je to ono što smo željeli. Zatvaranjem *New Module Wizarda*, OWB kreira odredišni modul u našem projektu te se ime modula pojavljuje u grani *MODULES*.



Slika 4.11. Završni dijalog *New Module Wizarda*.

Važno je napomenuti da iako smo kreirali odredišni modul podaci o tome još nisu unijeti u repozitorij. Da bi potvrdili napravljeni posao potrebno je pritisnuti tipku *Commit* koja se nalazi u gornjem desnom kutu glavnog ekrana OWB-a. Pritiskom na tipku *Commit* spremamo napravljeni posao u repozitorij.

4.3.2. Stvaranje definicija za dimenzije

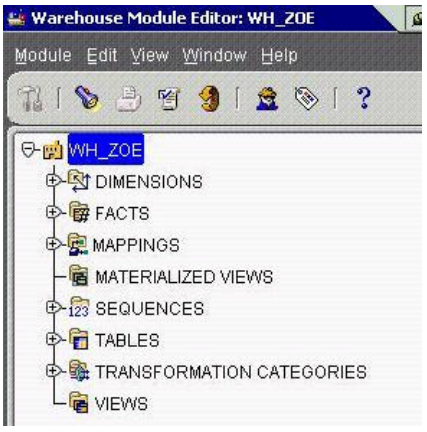
Kreiranjem odredišnog modula imamo mjesto gdje cemo spremati definicije našeg logickog modela. Slijedeci korak je kreiranje definicija za dimenzije. OWB zahtjeva da se prvo definiraju dimenzije, potom tablice cinjenica. Razlog je jasan. Tablice cinjenica referenciraju primarne kljuceve dimenzija pa je prvo potrebno kreirati primarni kljuc, a tek onda referencu tog kljuca.

Kreiranjem definicije za dimenziju ustvari kreiramo dvije definicije: jednu za dimenzijski objekt, a drugu za dimenzijsku tablicu. Dimenzijski objekt se sastoji od niza razina agregacije (engl. level of aggregation, level) i hijerarhija nad tim razinama agregacije. Razina agregacije predstavlja razinu grupiranja (npr. dan, tjedan, mjesec, godina su razine agregacije). Hijerarhije se definiraju nad razinama i definiraju roditelj-dijete odnose izmedu njih. Hijerarhije opisuju kako se razine agregacije grupiraju jedna u drugu (Primjer hijerarhije je: dan se grupira u tjedan, tjedan se grupira u mjesec, mjesec se grupira u godinu). Unutar jednog dimenzijskog objekta može biti definirano i više od jedne hijerarhije.

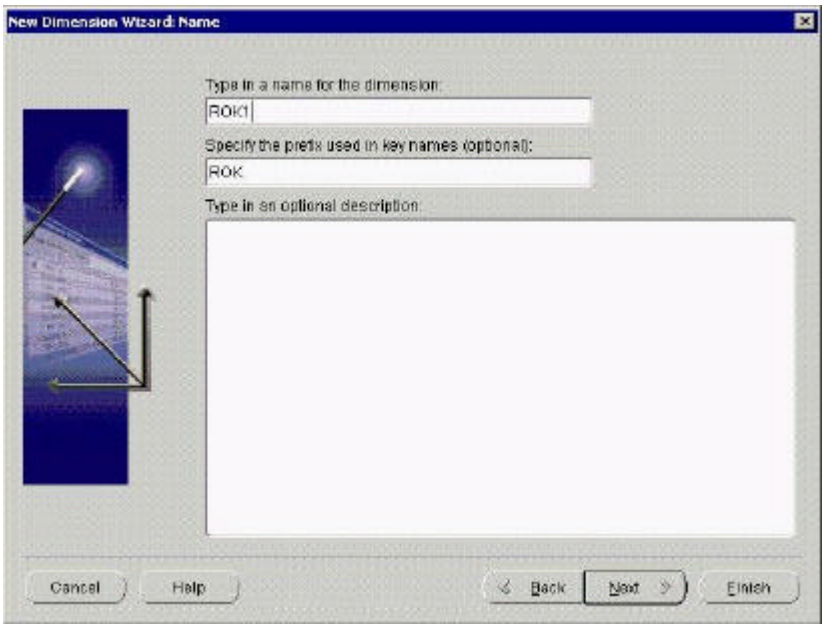
Prilikom kreiranja hijerarhije, OWB kreira identifikacijski kljuc za svaki nivo u toj hijerarhiji i jedinstveni kljuc (engl. unique key) za najniži nivo agregacije. OWB koristi identifikacijske kljuceve tijekom faze generiranja kako bi stvorio DDL skripte za kreiranje dimenzijskog objekta. Zbog postojanja tih identifikacijskih kljuceva potrebno je jako paziti prilikom kreiranja definicija za tablicu cinjenica. Naime kada se odreduje koji atributi iz dimenzije ce biti strani kljucevi u tablici cinjenica, OWB ponudi osim jedinstvenog kljuca i identifikacijske kljuceve kao kandidate za strane kljuceve. Medutim samo jedinstveni kljuc može biti strani kljuc u tablici cinjenica.

Da bi kreirali definiciju za dimenziju potrebno je pokrenuti *New Dimension Wizard*. Postoji još i *New Time Dimension Wizard* koji služi za kreiranje definicija za dimenzije vremena. I jedan i drugi *wizard* se pokrecu iz *Warehouse Module Editora* (slika 4.12.). Do njega se dolazi dvostrukim pritiskanjem lijeve tipke miša na ime odredišnog modula. Kao što se vidi na slici on ispod imena modula sadrži brojne grane. Svaka od tih grana sadrži definicije posebnih objekata (dimenzija, tablica cinjenica, mapiranja, materijaliziranih pogleda, itd.). Da bi pokrenuli *New Dimension Wizard* potrebno je oznaciti granu DIMENSIONS i pritiskom desne tipke miša otvoriti padajuci izbornik. Iz tog izbornika potrebno je odabrati *Create Dimension* i tako pokrenuti *New Dimension Wizard*.

Pokretanjem *New Dimension Wizarda* otvara se uvodni prozor koji nam ukratko opisuje korake u procesu kreiranja definicije za dimenziju, te nas upozorava na podatke koje cemo trebati unijeti. Prvi korak je unošenje podataka o imenu dimenzije, prefiksu koji ce se upotrebljavati prilikom imenovanja kljuceva, te opisa dimenzije koji nije obavezan. (slika 4.13.).

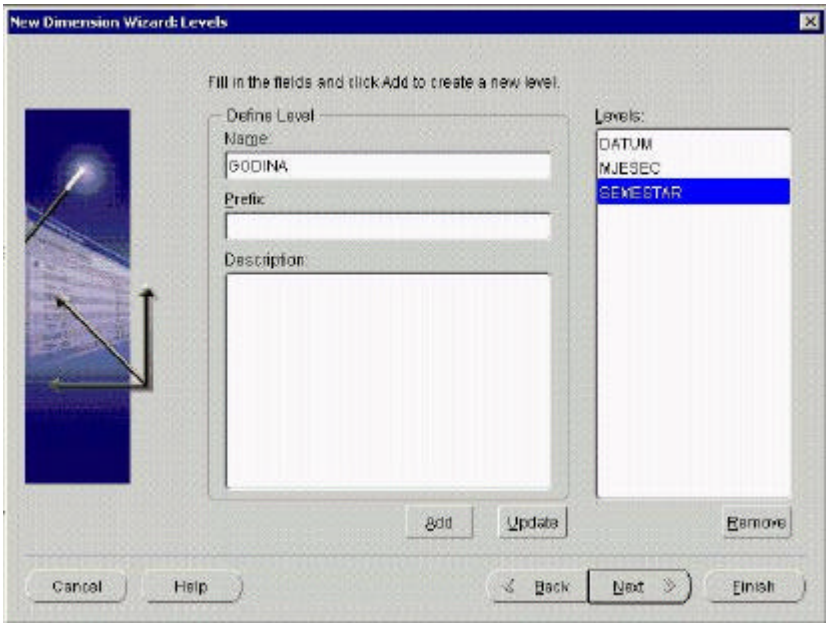


Slika 4.12. Warehouse Module Editor



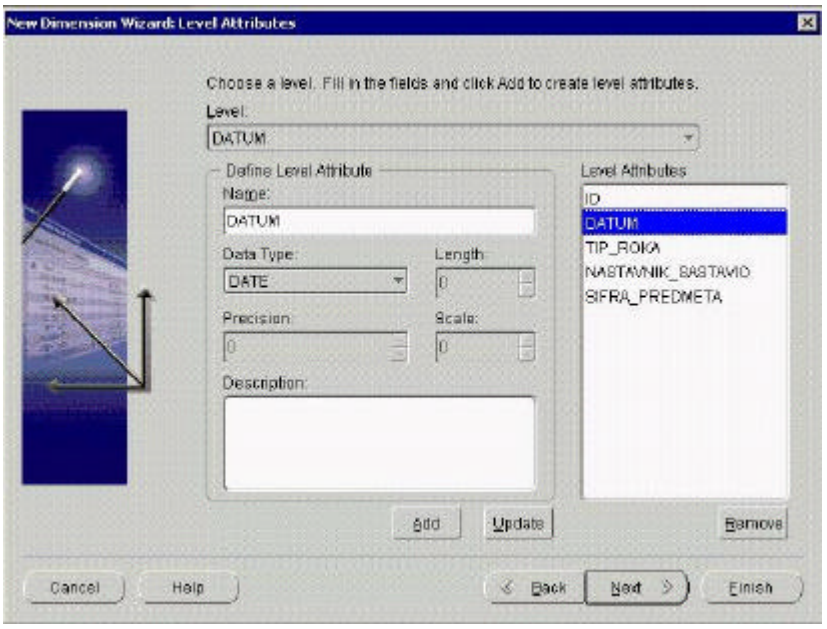
Slika 4.13. New Dimension Wizard: Korak 1

Slijedeci korak je definicija razina agregacije. Za svaku razinu agregacije potrebno je definirati njeno ime, prefiks i eventualno opis. Svaka dimenzija mora imati barem jednu razinu agregacije.



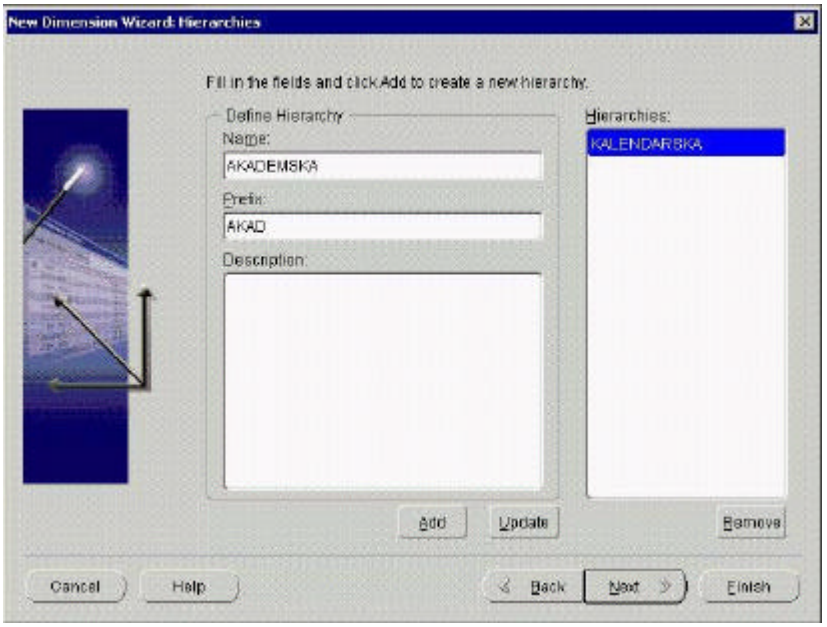
Slika 4.14. New Dimension Wizard: Korak 2

Treci korak traži da se za svaku razinu agregacije definiraju atributi te razine. Potrebno je definirati ime atributa, tip podataka, te opis.



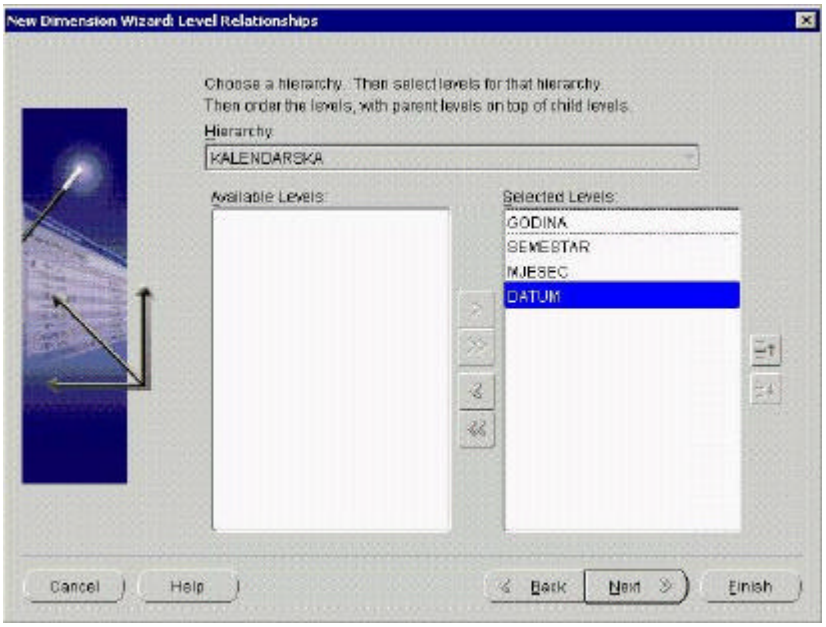
Slika 4.15. New Dimension Wizard: Korak 3

Slijedeci korak je definiranje hijerarhija. Takoder je potrebno unijeti ime, prefiks i opis za svaku hijerarhiju.



Slika 4.16. New Dimension Wizard: Korak 4

Peti korak je definiranje odnosa razina za svaku hijerarhiju. Razine unutar te hijerarhije se slažu na listu i to tako da je na vrhu liste najviša razina agregacije, a na dnu najniža.

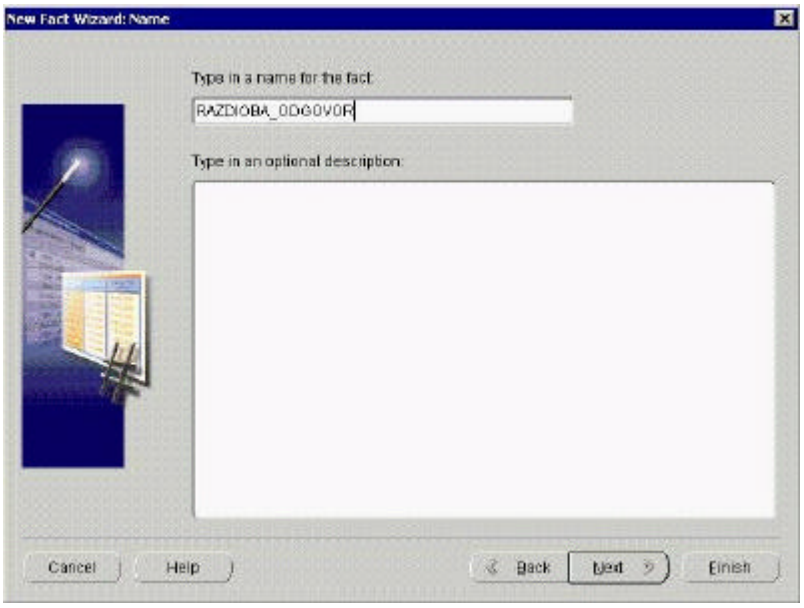


Slika 4.17. New Dimension Wizard: Korak 5

Završni dijalog prikazuje sažetak unesenih informacija, tako da možemo još jednom provjeriti tocnost danih informacija. Zatvaranjem *New Dimension Wizarda* OWB kreira definiciju za dimenzijski objekt i dimenzijsku tablicu, umece te definicije u odredišni modul i ime dimenzije se pojavljuje u navigacijskom stablu *Warehouse Module Editora*. Na taj nacin smo kreirali definiciju za jednu dimenziju. Postupak ponavljamo za svaku potrebnu dimenziju.

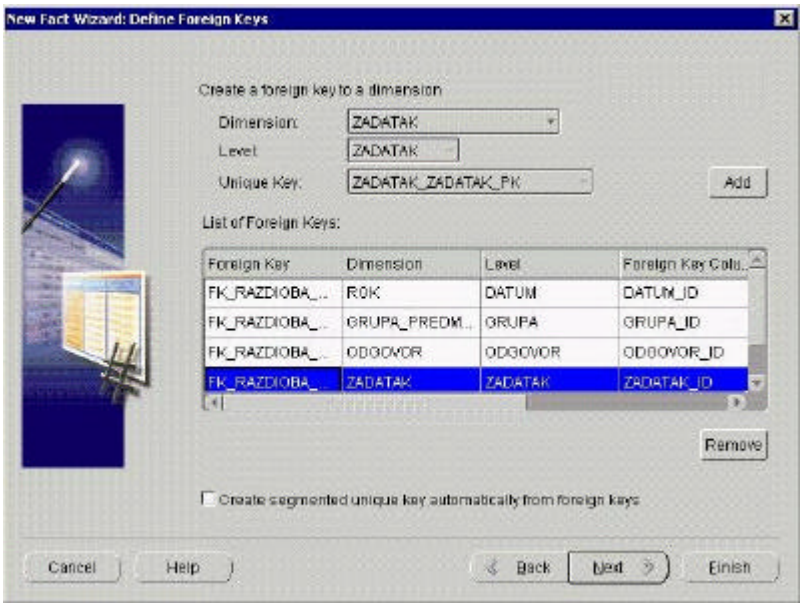
4.3.3. Stvaranje definicija za tablice cinjenica

Kad smo stvorili sve dimenzije, možemo pristupiti kreiranju definicija za tablice cinjenica. Postupak je jako slican kreiranju dimenzija, samo su podaci koje trebamo pružiti razliciti. Dakle, definiciju za tablicu cinjenica kreiramo pomocu *New Fact Wizarda* kojeg pokrecemo iz *Warehouse Module Editora*. Pokretanjem *New Fact Wizarda* otvara se pocetna stranica na kojoj se nalazi kratki opis koraka koji nas cekaju kao i upozorenje koje podatke trebamo dati. Prvi korak u kreiranju tablice cinjenica je imenovanje tablice cinjenica i davanje kratkog opisa (slika 4.18).



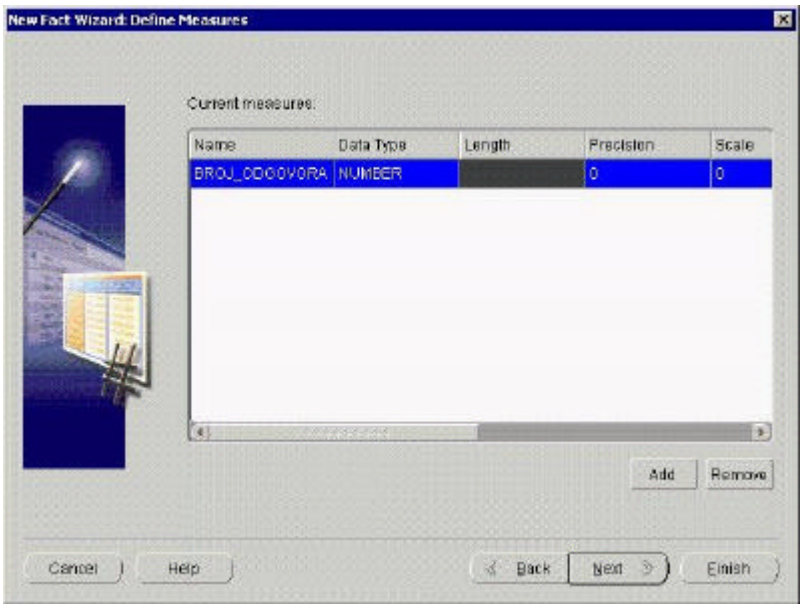
Slika 4.18. New Fact Wizard: Korak 1

Slijedeci korak je definiranje stranih kljuceva koji ce sacinjavati primarni kljuc tablice cinjenica. U ovom koraku treba jako paziti da se za strane kljuceve odaberu samo jedinstveni kljucevi iz dimenzija jer jedino oni odgovaraju svrsi.



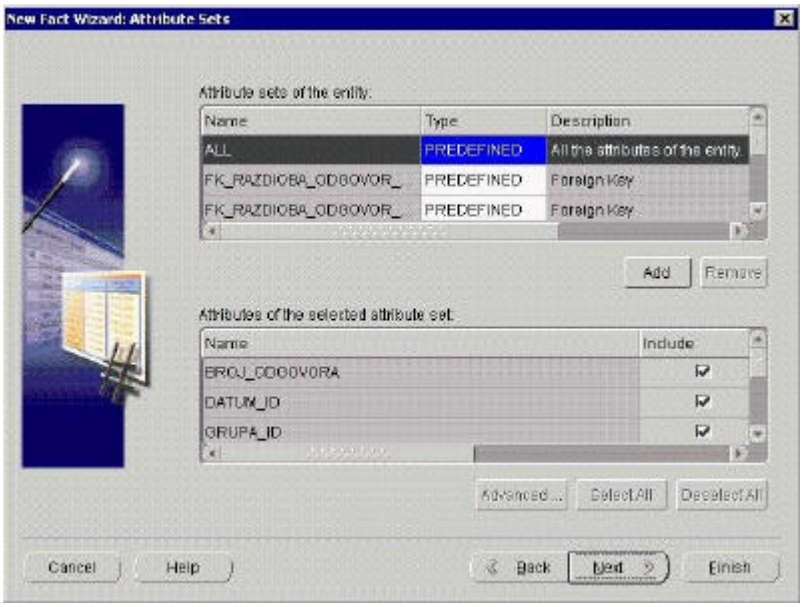
Slika 4.19. New Fact Wizard: Korak 2

Treci korak je definiranje cinjenica koje cemo pratiti (engl. facts, measures). U ovom koraku definiramo atribute koji predstavljaju cinjenice, te njihove tipove podataka.



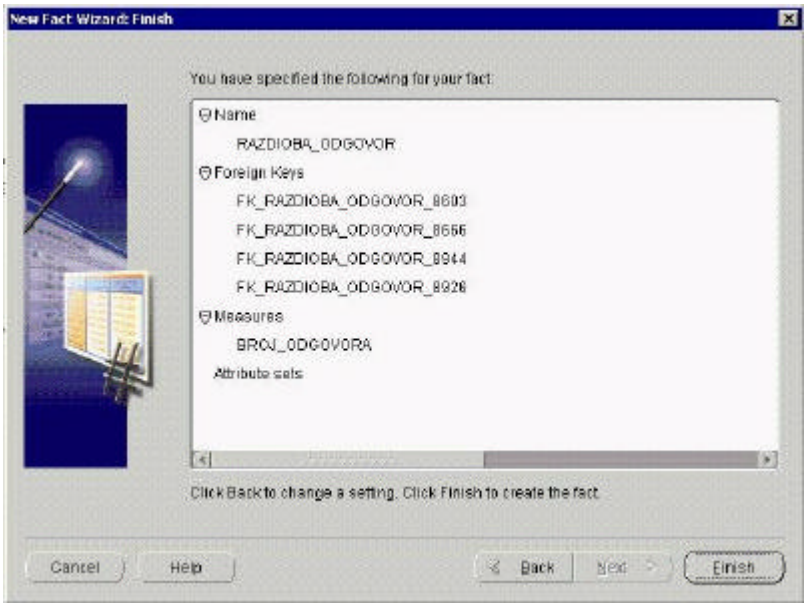
Slika 4.20. New Fact Wizard: Korak 3

Završni korak je definirati setove atributa koji ce se upotrebljavati u tablici. Postoje tri vrste setova: predefinirani, korisnicki definirani i tip most. Ja se nisam previše bavio setovima atributa, vec sam u svakoj tablici ostavio samo one predefinirane.



Slika 4.21. New Fact Wizard: Korak 4

Završni prozor je kao u svakom *wizardu* sažetak danih informacija kako bi mogli još jednom provjeriti tocnost informacija.

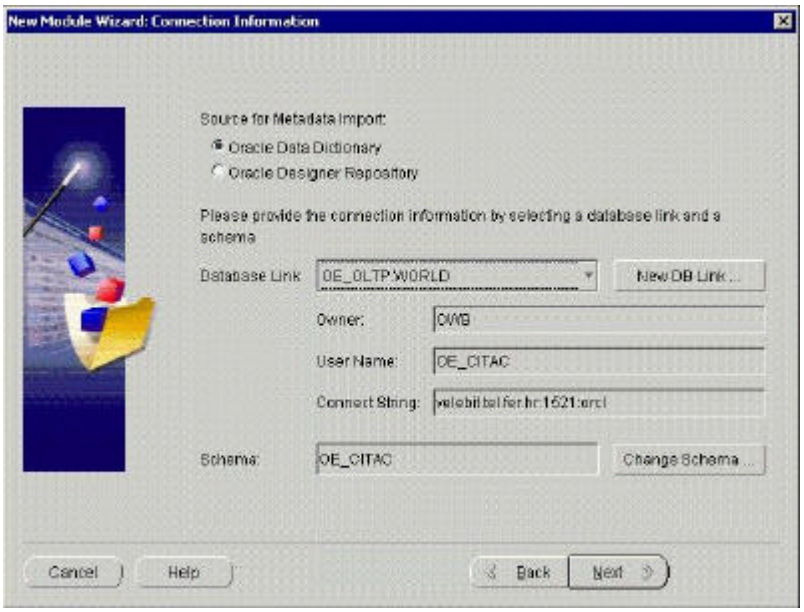


Slika 4.22. New Fact Wizard: Završni dijalog

Zatvaranjem *New Fact Wizarda*, OWB kreira tablicu cinjenica na osnovu danih informacija. Kreiranjem odgovarajucih tablica cinjenica naš logicki model je gotov. On je unesen u repozitorij (nakon pritiska tipke *Commit*). Kreirali smo definicije za dimenzije i tablice cinjenica i te definicije postoje u repozitoriju, medutim još nijedna tablica nije fizicki implementirana. Taj proces se obavlja u fazi generiranja skripti i pokretanja tih skripti.

4.3.4. Stvaranje izvorišnog modula i ucitavanje definicija izvora podataka

Dosad smo kreirali logicki model, ali još uvijek nismo definirali izvore podataka. Da bi spremili definicije za izvore podataka potreban nam je izvorišni modul (slicno kao što nam je potreban odredišni modul za spremanje definicija logickog modela). Proces kreiranja izvorišnog modula je jako slican procesu kreiranja odredišnog modula. Koristi se isti *wizard*. Razlika je samo u informacijma i opcijama koje dajemo *wizardu*. Jedina veca razlika je ta da za izvorišni modul moramo definirati valjani link prema bazi podataka koja ce nam služiti kao izvor podataka (Slika 4.23). Taj link nam omogucuje da iz te baze ocitamo definicije tablica, kljuceva, i ostalih relevantnih objekata koji nam trebaju u procesu mapiranja. Kada kreiramo izvorišni modul njegovo ime se pojavljuje u našem projektu pod granom MODULES (isto kao i odredišni modul). Takoder ako imamo više razlicitih izvora podataka, potrebno je za svaki izvor stvoriti jedan izvorišni modul.



Slika 4.23. New Module Wizard: Kreiranje linka prema bazi podataka

Nakon kreiranja izvorišnog modula potrebno je u njega ucitati definicije iz izvorišne baze podataka. Ucitavanje se obavlja izborom opcije *Import* iz padajuceg izbornika. Na taj nacin nam unutar OWB postaje vidljiva struktura izvorišne baze podataka te nam to omogucava izradu mapiranja.

5. Mapiranja

Nakon što smo definirali logicki model skladišta podataka, te ucitali definicije izvorne baze podataka, potrebno je definirati nacin na koji ce se podaci iz izvora prenijeti i ucitati u skladište podataka. Za tu svrhu nam služe mapiranja.

5.1. Opcenito o mapiranjima

Kreiranjem mapiranja definiramo proces izvlacenja, transformacije i ucitavanja podataka iz izvora u skladište podataka. Mapiranja se definiraju u skladišnom modulu. Definicija mapiranja formalno opisuje niz operacija koje izvlace podatke iz izvora podataka, transformiraju te podatke ako je to potrebno i zatim ucitavaju te podatke u odredišne tablice.

Mapiranje se u OWB-u sastoji od niza operatora. Operatori definiraju kako se ulaznim nizovima zapisa (engl. row-sets) manipulira radi dobivanja traženog izlaznog niza zapisa, a pri tom kardinalnost ulaznog niza zapisa ne mora biti jednaka kardinalnosti izlaznog niza zapisa. U OWB-u postoji nekoliko tipova operatora koji se po svojoj namjeni mogu rasporediti u: operatore za izvlacenje, operatore za ucitavanje, standardne operatore, transformacije i operatore vanjskih procesa. Operatori za izvlacenje i ucitavanje su jako slicni, a oni predstavljaju ili izvor podataka u mapiranju ili odredište podataka. Ostali operatori se mogu svrstati u grupu takozvanih *data flow* operatora koji imaju zadacu da manipuliraju podacima koji prolaze kroz njih prema zadanim parametrima. U mapiranju se koriste razliciti tipovi operatora, ali uvijek mora biti barem jedan operator izvlacenja koji ce predstavljati izvor podataka i barem jedan operator ucitavanja koji ce predstavljati odredište. Zatim se definira redoslijed tih operatora kao i njihov medusoban odnos te se na taj nacin definira ETL proces za odredenu tablicu, dimenziju ili tablicu cinjenica.

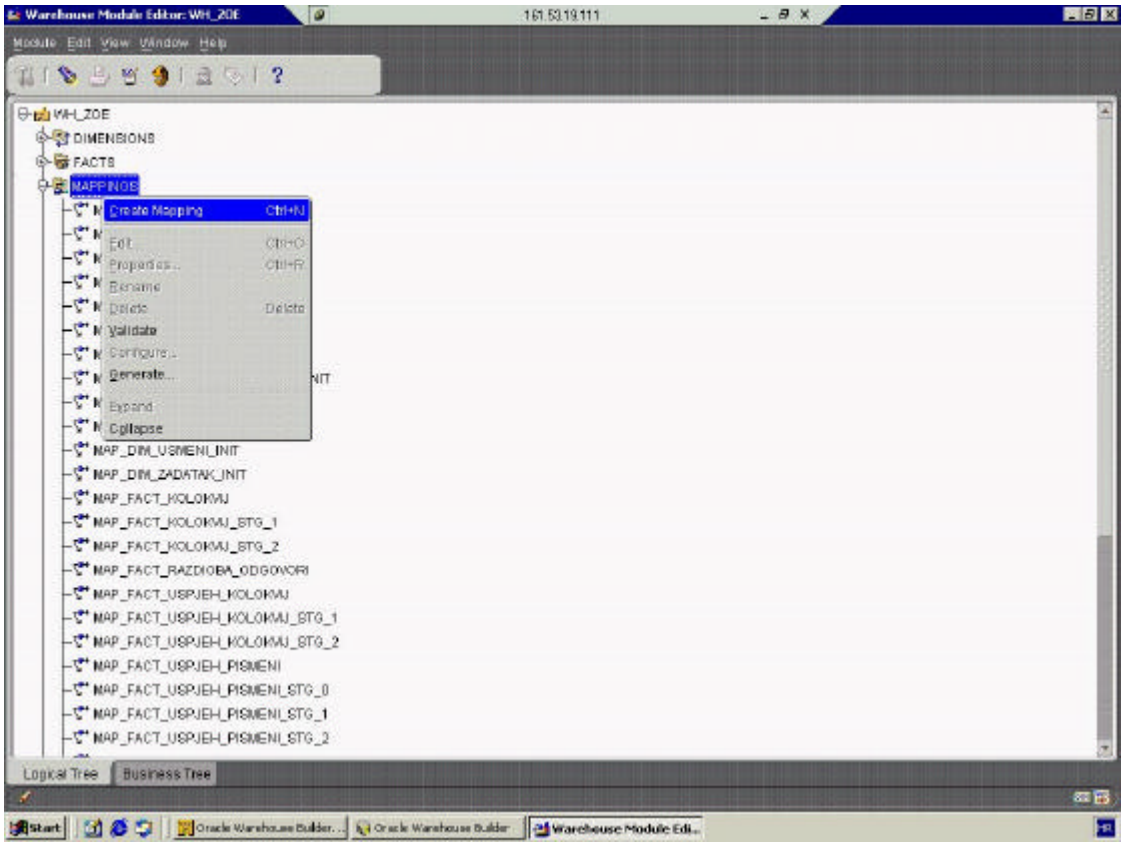
Svaki operator, ovisno o tipu, može imati jednu ili više ulaznih grupa atributa (engl. attribute groups), jednu ili više izlaznih grupa atributa, ili jednu ulazno/izlaznu grupu atributa. Tako npr. operator *Joiner*, koji je obavlja operaciju spajanja jednog ili više ulaza u jedan izlaz, može imati dvije ili više ulaznih grupa atributa i samo jednu izlaznu grupu atributa, dok npr. operator *Splitter*, koji obavlja operaciju razdvajanja jednog ulaza na više izlaza, može imati samo jednu ulaznu grupu atributa i 2 ili više izlaznih grupa atributa.

Operatori imaju tri razine: razinu operatora koja je najviša, razinu grupe atributa i razinu samog atributa koja je najniža razina. Svaka od tih razina, ovisno o tipu operatora, ima neka svojstva koja je potrebno konfigurirati da bi taj operator radio kako mi to želimo. Npr. operator *Joiner* ima na razini operatora svojstvo koje se zove uvjet spajanja (engl. join condition) u koje je potrebno navesti koji ce biti uvjet spajanja (to je u biti WHERE clause u SQL upitu) za taj operator. Konfiguracijom tih svojstava postižemo da operatori obavljaju posao koji nam treba.

Za operatore postoji još jedna bitna stvar, a zove se uskladivanje (engl. reconciliation). Znaci kad definiramo operator izvlacenja ili ucitavanja on se veže za odredenu tablicu, datoteku (engl. flat file), dimenziju ili tablicu cinjenica. Medutim nakon definiranja mapiranja u kojoj smo koristili taj operator koji je povezan za neku npr. tablicu, ta se tablica može promijeniti (dodaju se novi atributi, mijenja se struktura tablice). Takoder prilikom definiranja mapiranja možemo ustanoviti da nam neka tablica ne odgovara u potpunosti vec je potrebna promjena u strukturi te tablice. Da bi operator i nakon promjene strukture tablice za koju je vezan pratio u potpunosti tu tablicu potrebno ga je uskladiti s tom tablicom. Postoje dvije vrste uskladivanja: unutarnje i vanjsko uskladivanje. Unutarnje uskladivanje (engl. inbound reconciliation) se provodi ako se promijenila struktura tablice pa uskladujemo operator s tablicom, dok se vanjsko uskladivanje (engl. outbound reconciliation) provodi ako smo promijenili strukturu operatora, a želimo da se ta promjena ocituje i na tablici za koju je taj operator vezan.

U mapiranjima posebnu ulogu imaju i transformacije. Transformacije pretvaraju izvorne podatke u podatke koje su nam potrebne u skladištu podataka i one se pišu u PL/SQL-u. Postoje dvije vrste transformacija: funkcije i procedure. Funkcije su one transformacije koje imaju niz ulaznih parametara, a kao rezultat vracaju samo jedan, dok procedure mogu imati više ulaznih, ali i više izlaznih parametara. U OWB-u nam je na raspolaganju velik broj transformacija iz Oracle-ove knjižnice transformacija (engl. Oracle Transformation Library), ali isto tako možemo pisati svoje transformacije. Transformacije se u mapiranjima koriste preko operatora transformacija. Ti operatori se uskladuju s transformacijama na isti nacin koji je opisan za operatore izvlacenja i ucitavanja.

Mapiranje se kreira odabirom opcije *Create Mapping* iz izbornika koji se pokrene pritiskom desne tipke miša nad granom MAPPINGS *u Warehouse Module Editoru* (Slika 5.1.).



Slika 5.1 Stvaranje novog mapiranja

Nakon unošenja imena mapiranja i opisa mapiranja otvara se *Mapping Editor* u kojem definiramo naše mapiranja. *Mapping Editor* se sastoji od prazne pozadine (engl. canvas) na koju se nanose operatori, izbornika, toolbara i najvažnije palete objekata (Slika 5.2) sa koje se odabiru operatori koje cemo koristiti. Operatori koji se mogu koristiti su:

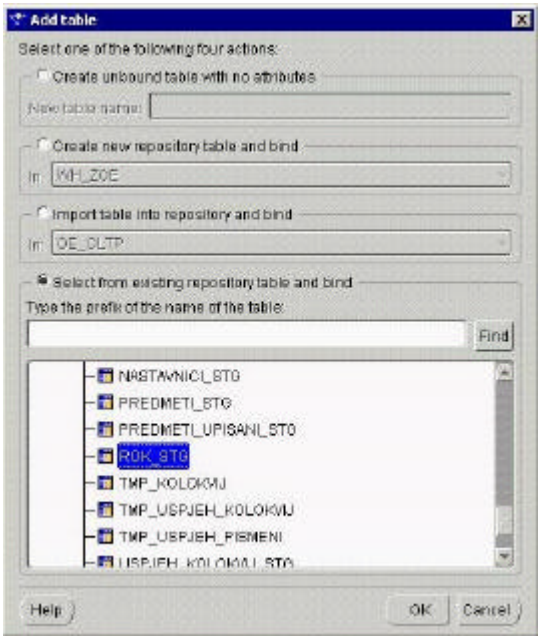
- Operatori izvlacenja i ucitavanja: Mapping Table, Mapping View, Mapping Materialized View, Mapping Sequence, Mapping Dimension, Mapping Flat File, Mapping Fact
- Standardni operatori su: Aggregator, Pre i Post-Mapping Process, Filter, Splitter, Joiner, Sorter, Deduplicator
- Operatori transforacija: Mapping Transformation, Expressions, Constants
- Operatori vanjskih procesa: Pure Integrate, Pure Extract

Slika 5.2 Paleta objekata

5.2. Primjer izgradnje mapiranja dimenzije

U ovom poglavlju cu prikazati kako se gradi mapiranje i to na primjeru mapiranja dimenzije ROK. Znaci, nakon što smo kreirali novo mapiranje otvori nam se prazan *Mapping Editor*. Naš zadatak je sada kombinacijom operatora i povezivanjem tih operatora stvoriti mapiranje koje ce obaviti izvlacenje i ucitavanje podataka. Prvi korak u izgradnji mapiranja je definiranje izvora. U ovom mapiranju kao izvore podataka koristim dvije tablice: ROK_STG i PREDAVANI_PREDMETI. ROK_STG je tablica koja je nastala mapiranjem MAP_ROK_STG, a služi kao pripremna tablica za ucitavanje podataka u dimenziju ROK.

Dakle, prvo trebamo definirati izvore podataka. Za definiciju izvora koristit cemo operator za izvlacenje *Mapping Table*. Odaberemo na paleti objekata operator *Mapping Table* te ga držeci lijevu tipku miša prebacimo na prazni dio ekrana, na mjesto gdje želimo da taj operator bude. Tada se otvara dijalog za dodavanje tablice koji nas pita koju tablicu i na koji nacin želimo tu tablicu dodati (slika 5.3.). Kao što se vidi postoji nekoliko opcija za dodavanje tablice: *Create unbound table with no attributes* – ovaj opcija stvara tablicu bez atributa i operator nije povezan s nijednom stvarnom tablicom, *Create new repository table and bind* – ova opcija poziva *New Table Wizard* koji omogucava stvaranje nove tablice i povezuje operator s tom tablicom, *Import table into repository and bind* – opcija omogucava ucitavanje neke tablice u repozitorij i povezivanje operatora s njom, *Select from existing repository table and bind* – ako smo vec kreirali tablicu onda ova opcija omogucuje odabir jedne od takvih tablica i povezivanje operatora s njom. Ovakav dijalog se pojavljuje za sve operatore izvlacenja i ucitavanja kao i za operatore transformacija (pojavljuje se za sve operatore koji se trebaju povezati s stvarnim fizickim objektom).

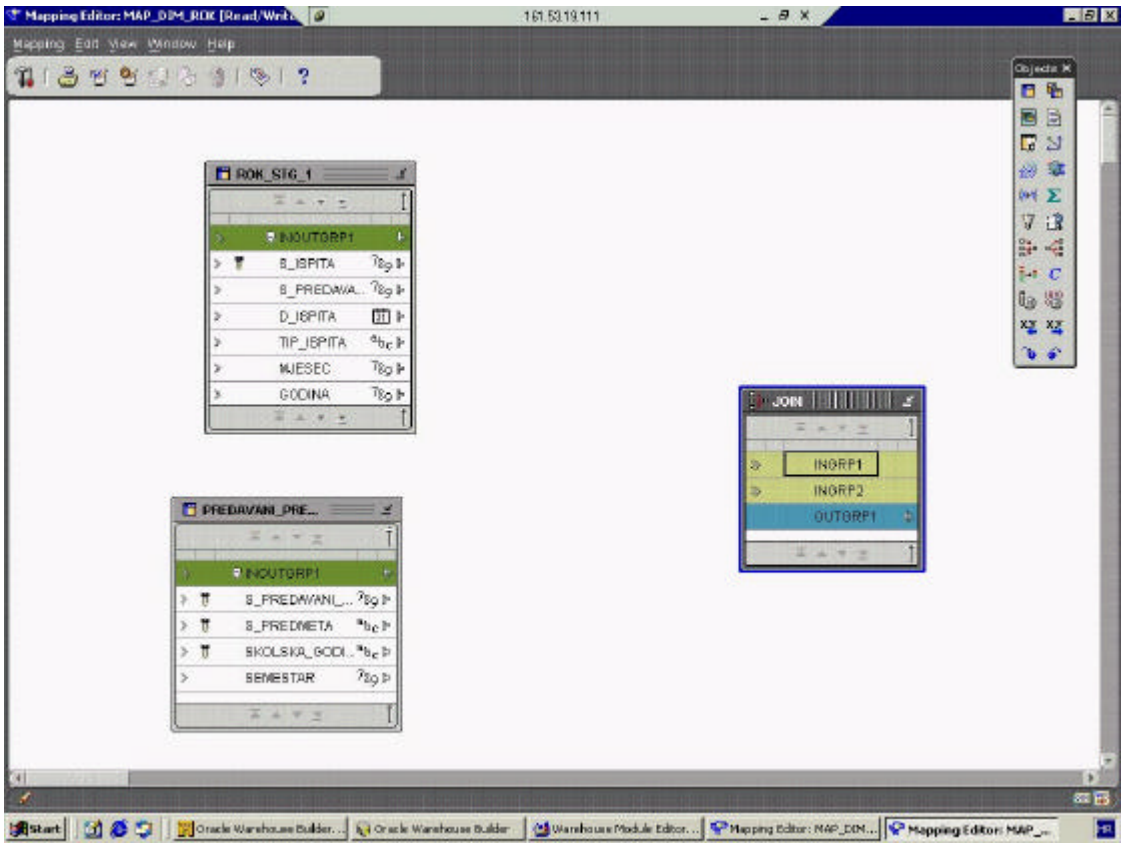


Slika 5.3 Dijalog za dodavanje tablice u mapiranju

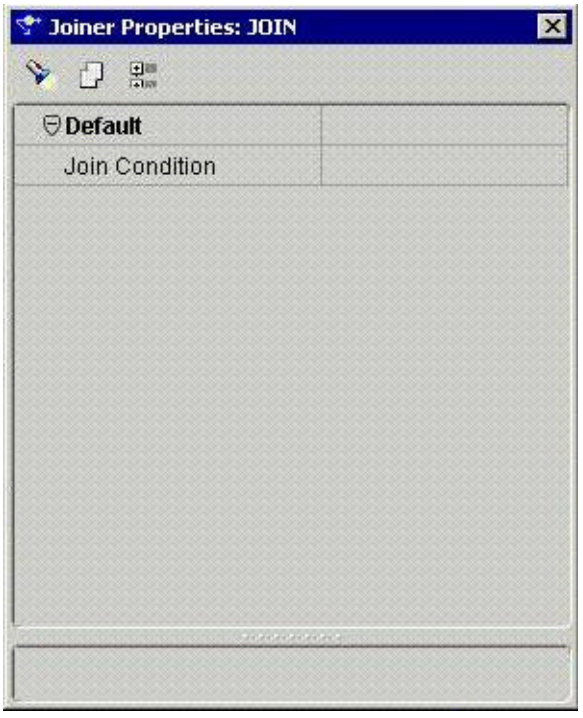
Nakon što smo dodali operator *Mapping Table*, on se pojavi na ekranu sa svojim grupama atributa i atributima kao na slici 5.4. Isti postupak ponavljamo za dodavanje tablice PREDAVANI_PREDMETI, te za dodavanje operatora *Joiner* (samo se za njega ne pojavi dijalog vec se on odmah prikaže na ekranu).

Kao što vidimo na slici operator *Joiner* ima dvije ulazne grupe atributa i jednu izlaznu. U prvu ulaznu grupu atributa se stavljaju željeni atributi iz jedne tablice, a u drugu atributi iz druge tablice. *Joiner* ce nam služiti za spajanje ove dvije tablice preko atributa S_PREDAVANI_PREDMET. Kada odredimo koji nam atributi ulaze u operator *Joiner* potrebno je definirati uvjet spajanja. Da bi ga odredili potrebno je pritisnuti desnu tipku miša nad operatorom *Joiner*. Tada se pojavi izbornik s kojeg odaberemo opciju *Operator properties* i kao rezultat dobijemo dijalog s svojstvima operatora (slika 5.5). U ovom slucaju postoji samo jedno svojstvo i to je uvjet spajanja. Pritiskanjem lijeve tipke miša nad praznim dijelom uvjeta spajanja otvara se takozvani *Expression Builder* (slika 5.6.). *Expression Builder* je dijalog u kojem možemo, bez unošenja koda, graditi razne izraze, pa tako i uvjet spajanja. Taj alat nam pomaže da bez ukucavanja koda gradimo kompleksne izraze najcešce kao dio svojstva nekog operatora kao što je i ovdje slucaj. U ovom slucaju definiramo da se ove dvije tablice spajaju po vrijednosti atributa S_PREDAVANI_PREDMET. *Expression Builder* ima tipku *Validate* koja omogucava provjeru sintakse izraza kojeg

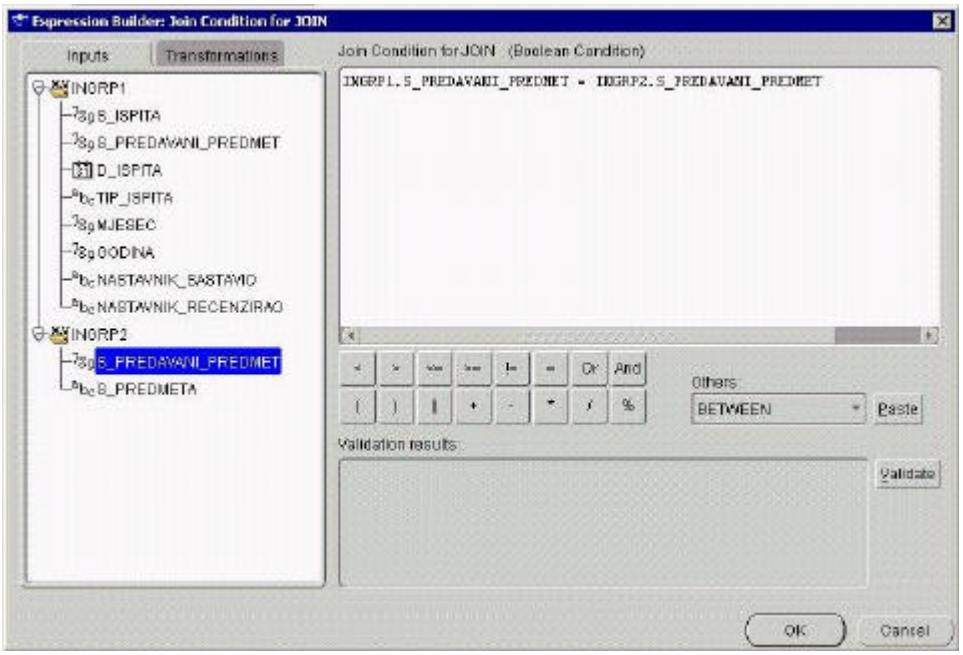
smo napravili i jako je korisna. Svaki izraz kojeg napravimo trebali bi validirati kako bi rano otkrili eventualne pogreške.



Slika 5.4 Dodani operatori na ekranu *Mapping Editora*



Slika 5.5 Svojstva operatora *Joiner*

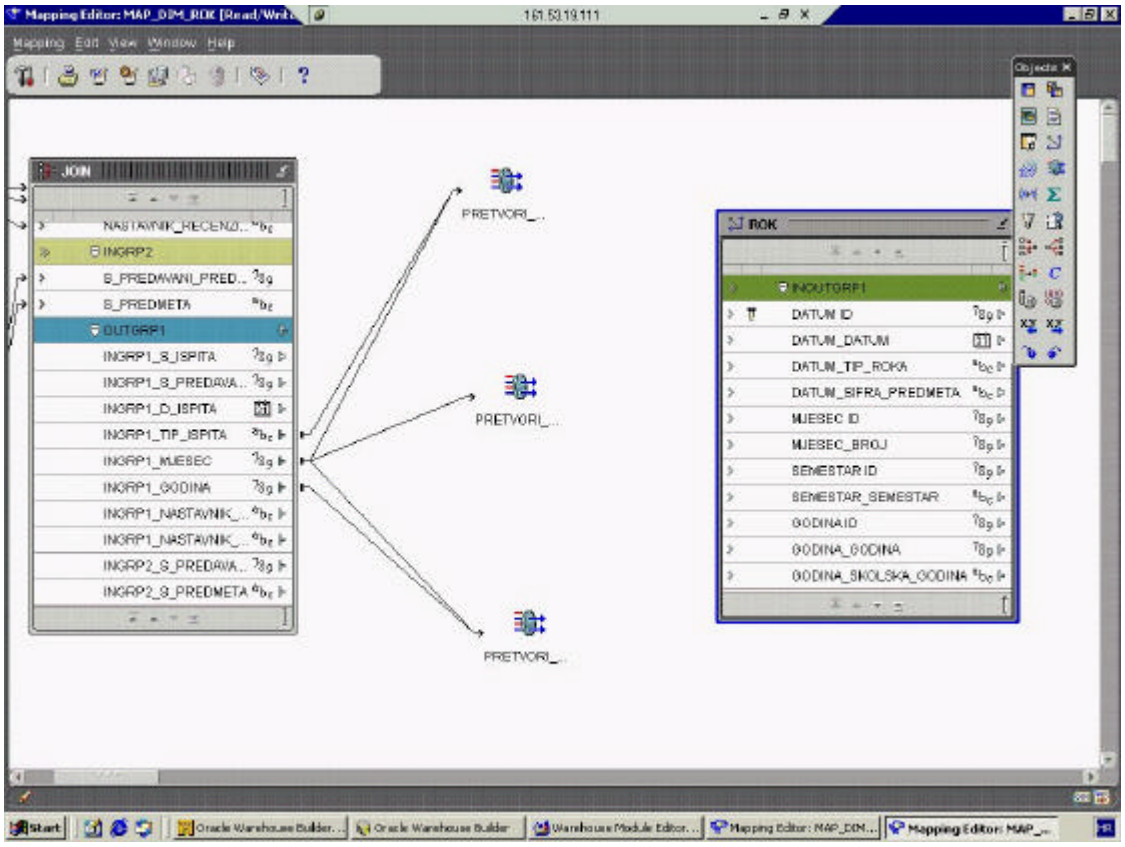


Slika 5.6 Expression Builder

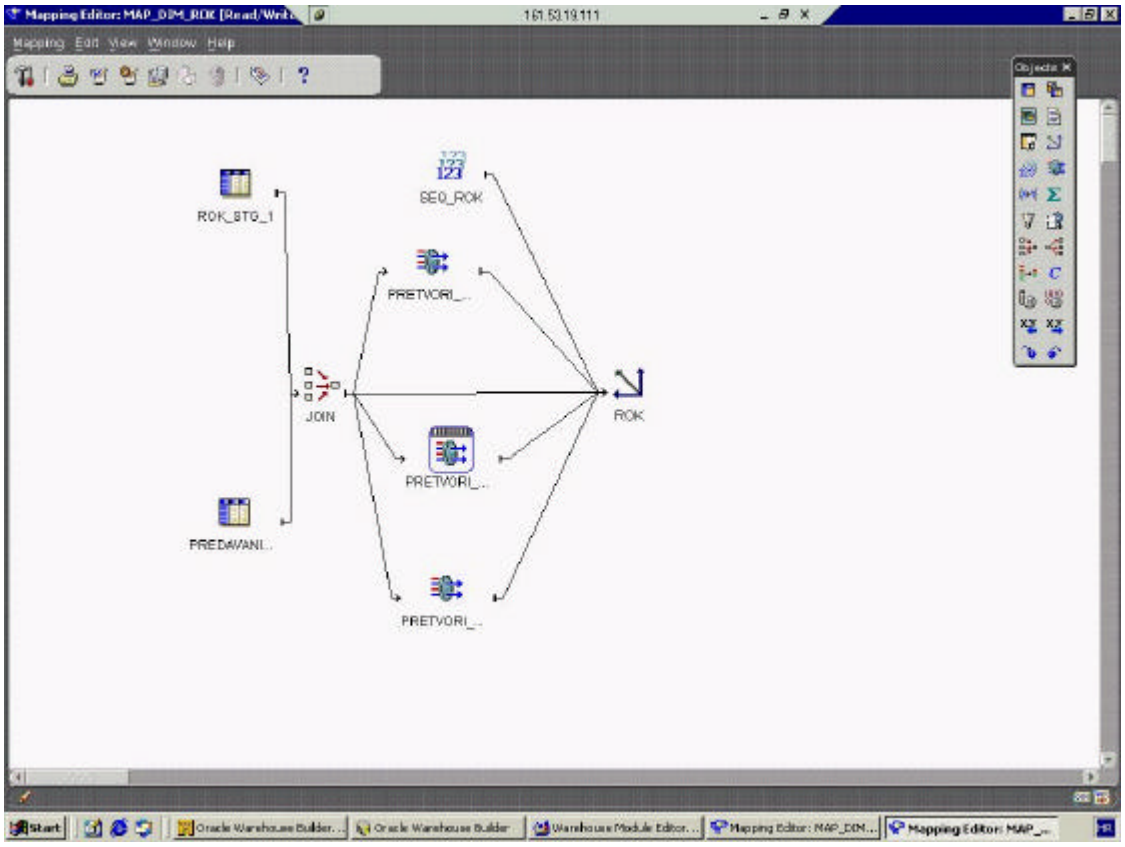
Nakon spajanja dviju tablica imamo skoro sve atribute koji su nam potrebni za ucitavanje u dimenziju ROK. Nedostaju nam podaci o školskoj godini i semestru, te je još potrebno transformirati atribut TIP_ISPITA u odgovarajuci oblik. (TIP_ISPITA ima kao moguce vrijednosti KOLOKVIJ i PISMENI, a za potrebe skladišta nam trebaju vrijedosti 1.KZ, 2.KZ, 3.KZ i PISMENI). Tu na scenu stupaju transformacije. Potrebne su nam tri. Da bi koristili transformaciju potrebno je za svaku dodati operator transformacije na ekran i povezati taj operator s odgovarajucom transformacijom. Postupak je identican dodavanju operatora *Mapping Table*.

Dodavanjem transformacija imamo sve potrebne podatke za ucitavanje u dimenziju ROK pa možemo dodati i operator ucitavanja *Mapping Dimension* koji cemo povezati s dimenzijom ROK (slika 5.7.).

Povezivanjem odgovarajucih atributa s atributima dimenzije ROK, privodimo kraju mapiranje. Završna akcija je dodavanje operatora *Sequence* koji je povezan s odgovarajucom sekvencom koja služi za generiranje primarnog kljuca. Mapiranje dimenzije ROK je gotovo i izgleda kao na slici 5.8. Da bi provjerili naše mapiranje koristimo opciju *Validate* iz izbornika *Mapping*. Opcija *Validate* provjerava naše mapiranje i traži greške u sintaksi (npr. povezivanje atributa koji imaju razlicite tipove, i sl.), te nas na te greške i upozorava. Važno je napomenuti da validiranje provjerava samo sintaksu, a ne i da li ce naše mapiranje pravilno obavljati posao. Taj dio moramo sami provjeriti.



Slika 5.7 Dodani operatori transformacija i operator dimenzije ROK

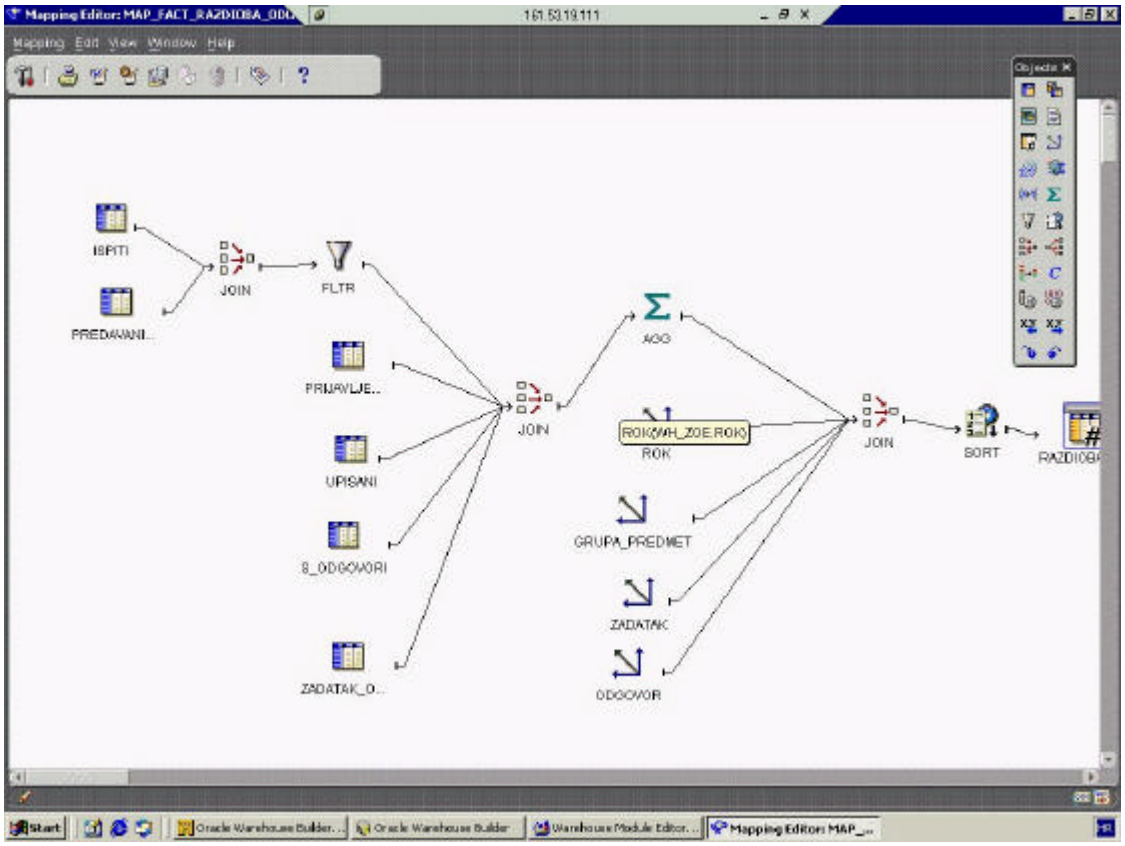


Slika 5.8 Mapiranje dimenzije ROK

5.3. Mapiranje tablice cinjenica

Postupak mapiranja tablice cinjenica je isti kao kod mapiranja dimenzija. Jedina razlika je cinjenica da je mapiranje tablice cinjenica uvijek kompleksnije, koristi se više operatora i cesto se koristi operator *Aggregator* koji može obavljati razlicite funkcije agregacija (COUNT, SUM, AVG, MAX, MIN, itd.). Ukratko cu objasniti mapiranje tablice cinjenica RAZDIOBA_ODGOVORI (slika 5.9).

Kao izvori podatka koriste se tablice: ISPITI, PREDAVANI_PREDMETI, PRIJAVLJENI_ZA_ISPIT, UPISANI, S_ODGOVORI i ZADATAK_ODGOVOR. Kao što znamo tablica cinjenica prati razdiobu odgovora po rokovima, a da bi to mogli dobiti potrebne su nam navedene tablice. Prvo spajamo tablice ISPITI i PREDAVANI_PREDMETI. Iz tih tablica dobivamo podatke o datumu ispita, tipu ispita, nastavnicima koji su ga sastavili odnosno recenzirali te o šifri predmeta iz kojeg je taj ispit. Nakon spajanja tablica provodimo filtriranje podataka. Podatke filtriramo po datumu i šifri predmeta kako bi u tablicu cinjenica upisivali podatke rok po rok. Zamišljeno je da prilikom pokretanja ove skripte korisnik koji je pokrece unosi kao parametar podatke o datumu ispita i šifri predmeta iz kojeg je taj ispit, kako bi se nakon završenog i obradenog roka unijeli podaci. Dalje spajamo sve ostale tablice kako bi dobili sve atribute koji nam trebaju. Nakon spajanja brojimo odgovore, ali tako da se brojenje grupira po roku, po grupama iz predmeta, po zadatku te po odgovoru. U tu svrhu koristimo operator *Aggregator* koji koristi funkciju COUNT, uz GROUP BY D_ISPITA, S_GRUPE, S_ZADATKA, ODGOVOR. Zadnje spajanje provodimo kako bi iz dimenzijskih tablica dobili za odgovarajuce vrijednosti atributa vrijednost primarnih kljuceva koji ce biti strani kljucevi u tablici cinjenica. Spajanjem odgovarajucih atributa s atributima u operatoru *Mapping Fact* naše mapiranje je gotovo, ali ga još treba validirati i potražiti skrivene pogreške u sintaksi. Nakon validacije mapiranje je spremno za konfiguraciju i generiranje skripti.



Slika 5.9 Mapiranje tablice cinjenica RAZDIOBA_ODGOVORI

6. Konfiguracija fizickih svojstava objekata u Oracle Warehouse Builderu

Kad smo napravili sva potrebna mapiranja dimenzija i tablica cinjenica, vrijeme je da prijedemo na drugu fazu izgradnje skladišta podataka, a to je konfiguracija fizickih parametara koji su potrebni da bi se objekti fizicki implementirali. Fizicka svojstva je potrebno konfigurirati kako bi OWB na pravilan nacin generirao potrebne skripte za definiranje objekata (DDL skripte) i izvlacenje i ucitavanje podataka (skripte pisane u raznim jezicima: SQL, PL/SQL, ABAP). Potrebno je konfigurirati fizicka svojstva svakog objekta (skladišnog modula, dimenzija, tablica cinjenica, tablica, sekvenci, mapiranja, itd.). Svaki taj objekt ima neka svojstva koja su specificna za njega, a svi objekti imaju i neka zajednicka svojstva koja treba konfigurirati.

6.1. Konfiguracija fizickih svojstava skladišnog modula

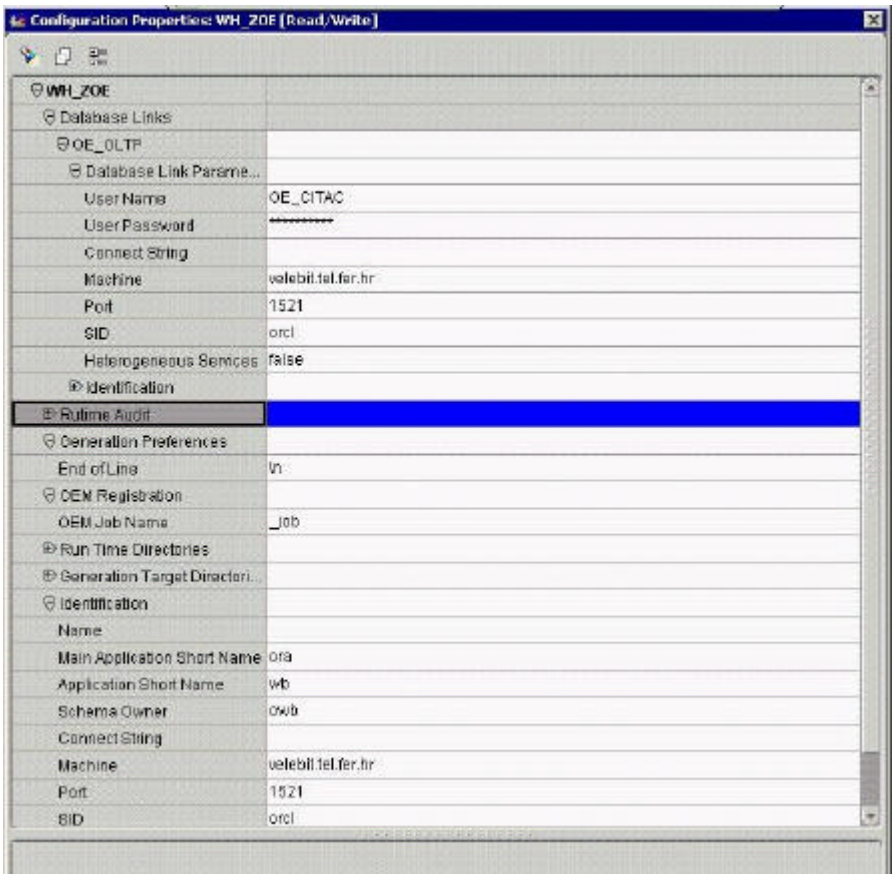
Da bi konfigurirali skladišni modul potrebno je u *Warehouse Module Editoru* oznaciti ime skladišnog modula kojeg želimo konfigurirati, te na izborniku odabrati opciju *Configure*. To otvara poseban prozor u kojem pristupamo konfiguraciji parametara. (Slika 6.1.)

Kao što se vidi na slici ima dosta parametara koje treba podesiti. Parametri su podijeljeni u kategorije. Prva kategorija parametara obuhvaca konfiguraciju veze prema bazi podataka (*Database Links*). U toj skupini potrebno je definirati korisnicko ime i lozinku, te racunalo na kojem se nalazi baza podataka, te port i SID. Veza prema bazi podataka je važna jer se prilikom svakog izvlacenja podataka upotrebljava.

Runtime Audit kategorija parametara se odnosi na razinu nadgledanja procesa u skladišnom modulu. Postoji nekoliko razina nadgledanja: Bez nadgledanja, nadgledanje na razini statistike, nadgledanje na razini pogrešaka i kompletno nadgledanje. U našem slucaju postavljeno je kompletno nadgledanje.

Generation Preferences je kategorija u kojoj se podešavaju svojstva potrebna za generirane skripte. Jedini parametar koji se podešava je *End of Line* parametar koji za Windows NT mašinu bi trebao imati vrijednost \r\n, a za UNIX \n.

OEM Registration kategorija podešava kakav sufiks ce dobiti poslovi koji su registrirani u OEM-u (*Oracle Enterprise Manageru*). Postavljena vrijednost je _job.



Slika 6.1 Konfiguracija skladišnog modula

Run Time Directories i *Generation Target Directories* kategorije su za podešavanje imena direktorija u koje ce se spremati generirane skripte i koji ce se koristiti tijekom izvodenja. Takoder se u tim kategorijama podešavaju ekstenzije za skripte (npr. za PL/SQL skriptu ekstenzija je .pls).

U I*dentification* kategoriji se nalaze parametri o samom modulu, kako se zove, tko je vlasnik, na kojem racunalu, itd.

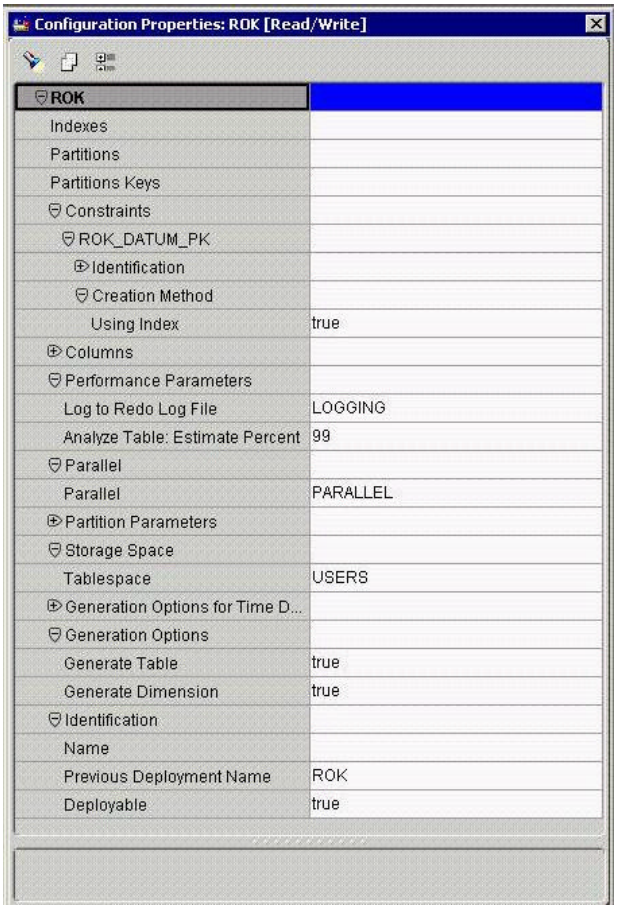
Sve ove parametre je potrebno podesiti prije generiranja skripti. Za naše skladište skladištni modul je konfiguriran kao na slici 6.1.

6.2. Konfiguracija fizickih svojstava dimenzija, tablica, i tablica cinjenica

Da bi konfigurirali dimenziju potrebno je oznaciti ime dimenzije koju želimo konfigurirati i u izborniku odabrati opciju *Configure*. Tada se takoder otvara novi prozor u kojem se nalaze konfiguracijski parametri. (slika 6.2.)

Kao što vidimo sa slike, konfiguracija dimenzije se znatno razlikuje od konfiguracije skladišnog modula što je i razumljivo. I ovdje imamo parametre

podijeljene po razlicitim kategorijama. Kategorije *Partitions*, *Partitions Keys*, *Partition Parameters* su vezane uz particije, a zbog relativno male kolicine podataka, objekti iz našeg skladišta nisu podijeljeni u particije, pa te parametre nisam podešavao.



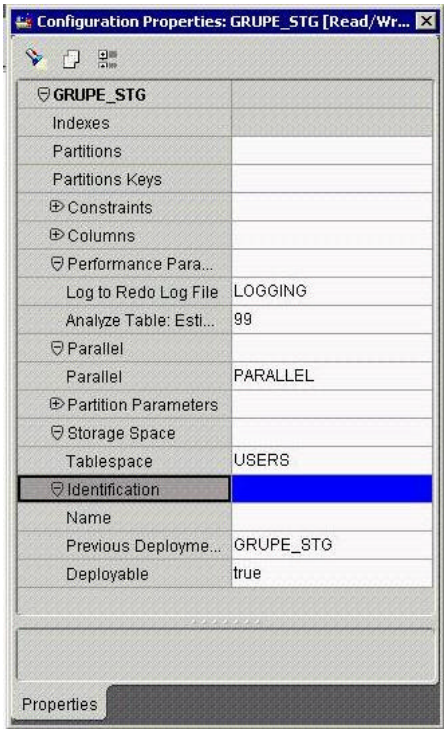
Slika 6.2 Konfiguracija dimenzije

Od ostalih kategorija najvažnije su kategorije *Performance Parameters*, *Parallel* i *Storage Space*. Podešavanjem parametara iz kategorije *Performace Parameters* i *Parallel* direktno utjecemo na brzinu izvodenja upita i ucitavanja podataka, dok podešavanjem parametra *Tablespace* iz kategorije *Storage Space* kazujemo gdje cemo fizicki smjestiti danu dimenziju.

Na slici 6.2. je prikazano kako je konfigurirana dimenzija ROK, a ostale dimenzije imaju sve bitne parametre konfigurirane isto kao i ova dimenzija.

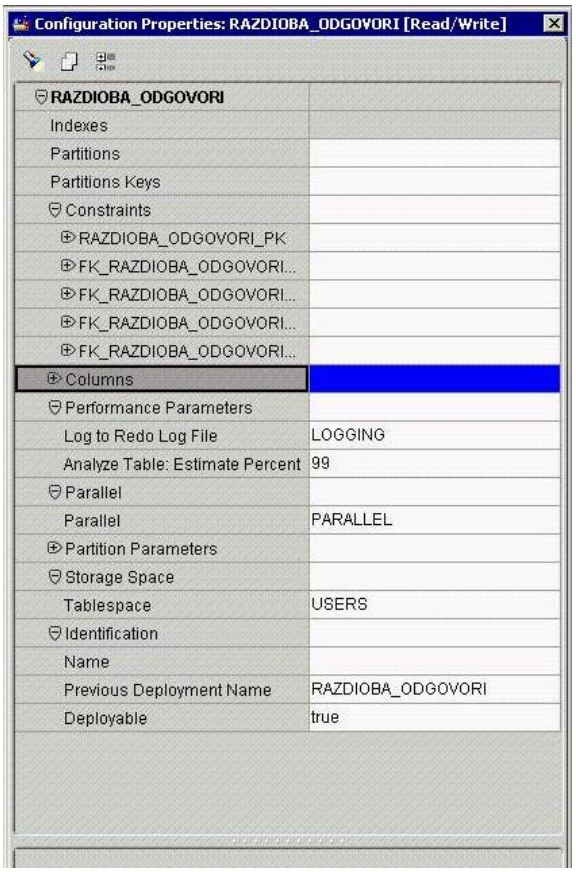
Postupak kod konfiguracija tablica je isti kao i kod konfiguracija dimenzija – odabirom opcije *Configure* s izbornika, dobivamo prozor s konfiguracijskim parametrima (slika 6.3). Konfiguracijski parametri su slicni onima kod dimenzije i

cak su isto i konfigurirani kao što se vidi na slici. Takoder su i ostale tablice slicno konfigurirane kao i ova.



Slika 6.3 Konfiguracija tablice GRUPE_STG

Konfiguracija tablica cinjenica je takoder jako slicna konfiguraciji dimenzija i tablica. Konfiguracijski parametri su prikazani na slici 6.4. Kao što se vidi na slici u konfiguraciji parametara nema bitne razlike jer su i dimenzije i tablice i tablice cinjenica po svojoj namjeni slicne, i sve se spremaju na isto racunalu istu bazu. Jedine razlike su u konfiguraciji kategorije *Index* u kojoj se konfiguriraju indeksi za zadani objekt, a opet ovisno o tipu i namjeni objekta.



Slika 6.4 Konfiguracija tablice cinjenica RAZDIOBA_ODGOVORI

Više o svim ovim parametrima se može naci u Oracle-ovim prirucnicima, posebno u *OWB User's Guide* i *Designing and Tuning Performance*.

6.3. Konfiguracija mapiranja

Konfiguracija mapiranja se jako razlikuje od konfiguracije ostalih objekata u OWB-u i ima mnogo više parametara koje treba konfigurirati. Konfiguracija mapiranja je takoder jako važna jer ona definira kako ce se podaci izvlaciti (koje ce se operacije izvoditi). Prije konfiguracije samog mapiranja potrebno je konfigurirati atribute koje ulaze u odredišni operator te odredišni operator kako bi definirali nacin na koji ce se ti atributi ucitavati u odredište.

6.3.1. Konfiguracija atributa u mapiranjima

Znaci, prije same konfiguracije mapiranja potrebno je definirati kako ce se ponašati atributi dok se ucitavaju u odredište. Slijedeci operatori sadržavaju atribute koje bi trebalo konfigurirati prije ucitavanja: Tablice, Tablice cinjenica, Dimenzije, Pogledi, Materijalizirani Pogledi.

Svaki atribut ima svoja svojstva. Pritiskanjem desne tipke miša nad odredenim atributom pojavljuje se izbornik na kojem je i opcija *Attribute properties*. Odabirom te opcije otvara se prozor sa svojstvima atributa kao na slici 6.5. Za ucitavanje podataka bitni su parametri u kategoriji *Attribute Properties*. Podešavanjem tih parametara definira se ponašanje atributa u procesu ucitavanja.

Parametar *Insert: Use for Loading* definira da li ce se atribut koristiti za ucitavanje u odredište koristeci operaciju *Insert*. Ako je postavljen na *Yes* onda ce biti ucitan u odredište u operaciji *Insert*, a ako je postavljen na *No* onda nece biti ucitan u odredište iako je u mapiranju definirano da bi se trebao ucitati u odredište.

Parametar *Update: Use for Loading* je slican prethodnom parametru, ali on definira da li ce se atribut koristiti u operaciji *Update.*

Parametri *Update: Use for matching* i *Delete: Use for matching* definiraju da li ce se za taj atribut provjeravati da li njegova vrijednost vec postoji u odredištu, te ako postoji nad tim zapisom ce se onda provesti operacija *Update* odnosno *Delete.* Znaci, prilikom ucitavanja u odredište provjerava se da li vrijednost atributa, za koji su ta dva parametra postavljeni na *Yes*, vec postoji u odredištu. Ako postoji onda ce se samo nad tim zapisom u odredišti provesti definirana operacija.

Parametar *Update: Operation* definira kako ce se racunati nova vrijednost atributa, ako se tijekom provjere pronade ista vrijednost atributa i provede se operacija *Update.* Moguce operacije su:

= odredište := izvor

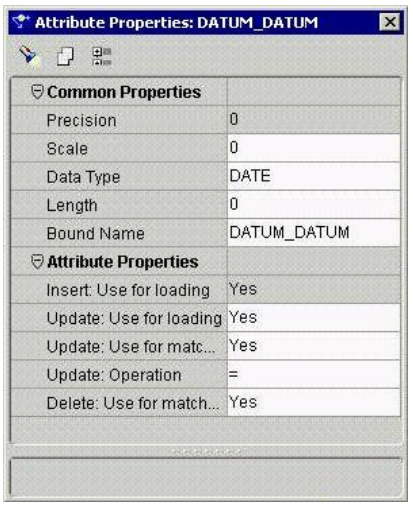
+= odredište := izvor + odredište

-= odredište := odredište – izvor

=- odredište := izvor – odredište

||= odredište := odredište||izvor

Nakon postavljanja ovih parametara može se prijeci na konfiguraciju operatora i onda na konfiguraciju samog mapiranja.



Slika 6.5 Svojstva atributa

6.3.2. Konfiguracija svojstava operatora u mapiranjima

Za svaki odredišni operator u mapiranju potrebno je podesiti njegova svojstva koja se odnose na metode ucitavanja podataka. Da bi se podesila svojstva treba otvoriti prozor sa svojstvima operatora (slika 6.6), odabirom *opcije Operator properties*. Za proces ucitavanja su bitni parametri u kategoriji *Data Entity parameters*, a posebno parametar *Loading Types* koji definira kako ce se provoditi operacija ucitavanja podataka.

Parametar *Loading Types* može imati nekoliko razlicitih vrijednosti:

INSERT – ova vrijednost definira da ce podaci biti dodani u tablicu korištenjem operacije *Insert*.

UPDATE – ova vrijednost definira da ce se podaci koji dolaze u odredište koristiti kako bi se provela operacija *Update* nad vec postojecim zapisima u odredištu.

INSERT/UPDATE – ova vrijednost definira da ce se za svaki dolazeci redak podataka, prvo obaviti operacija *Insert*. Ako se Insert zbog nekog razloga ne obavi onda ce se provesti operacija *Update*.

UPDATE/INSERT – ova vrijednost definira da ce se za svaki novi redak prvo pokušati obaviti operacija *Update*, a ako ona ne uspije onda ce se probati provesti operacija *Insert*.

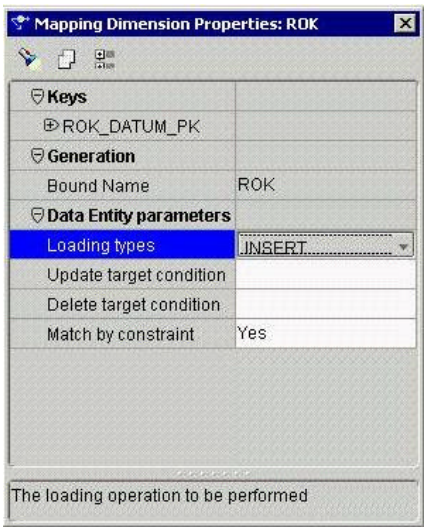
DELETE – ova vrijednost definira da ce se dolazeci podaci koristiti kako bi se odredilo koji ce zapisi iz odredišta biti brisani.

TRUNCATE/INSERT – ova vrijednost definira da ce podaci u odredištu prvo srezati, a zatim ce se novi podaci unijeti u odredište operacijom *Insert.*

DELETE/INSERT – ova vrijednost definira da ce se svi zapisi iz odredišta prvo obrisati, a zatim ce se novi podaci unijeti operacijom *Insert*.

CHECK/INSERT – ova vrijednost definira da ce se odredište provjeriti da li sadrži ijedan zapis. Ako ne sadrži onda ce se novi podaci unijeti operacijom *Insert*.

NONE – ova vrijednost definira da se nijedna operacija nece obaviti nad odredištem, ali podešavanje parametra *Loading Types* na ovu vrijednost može biti jako korisno tijekom ispitivanja jer ce se izvlacenje podataka i transformacije podataka provesti, ali podaci nece biti uneseni u odredište.



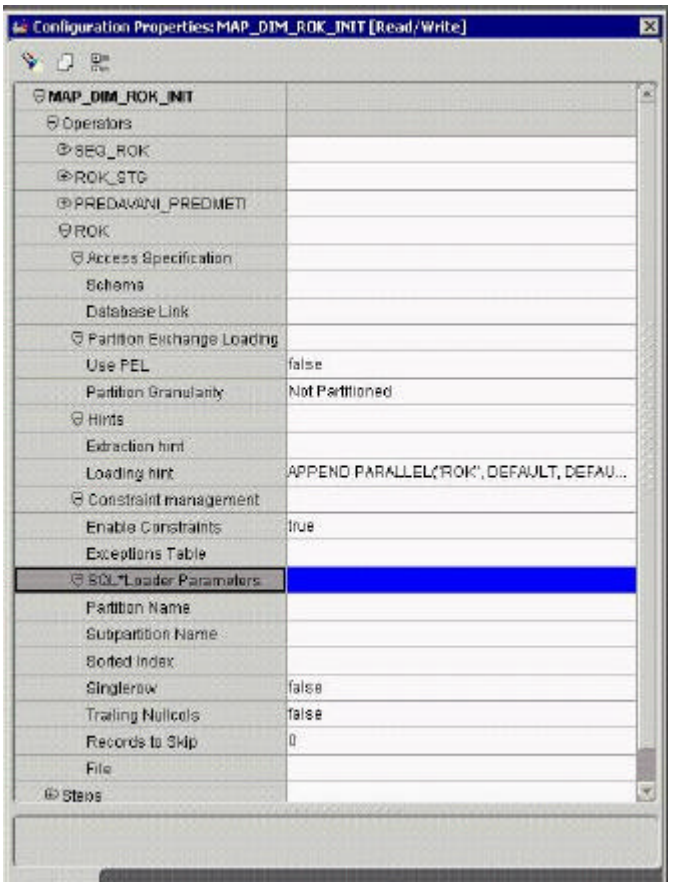
Slika 6.6 Svojstva operatora

Podešavanjem svojstava operatora, posebno parametra *Loading Types,* definiramo kako ce se podaci ucitavati u odredište. Na taj nacin, u kombinaciji s konfiguracijom parametara pojedinih atributa, odredujemo kako ce se provoditi proces izvlacenja i ucitavanja podataka u odredište.

6.3.3. Konfiguracija fizickih parametara mapiranja

Nakon što smo podesili parametre atributa i operatora u mapiranju, možemo prijeci i na konfiguraciju fizickih svojstava samog mapiranja. Da bi konfigurirali mapiranje potrebno je sa izbornika odabrati opciju *Configure*. Tada se otvara prozor koji sadrži fizicka svojstva mapiranja. Svojstva su podijeljena u dvije kategorije: *Operators* i *Steps*. Kategorija *Operators* sadrži parametre za sve operatore u mapiranju, dok kategorija *Steps* sadrži parametre koji su bitni za generiranje koda.

Na slici 6.7. su prikazani parametri iz kategorije *Operators* za jedan operator iz mapiranja MAP_DIM_ROK_INIT. Kao što vidimo za svaki operator postoji nekoliko podkategorija fizickih parametara. Od definiranja sheme u kojoj se nalazi operator do definiranja *SQL*Loader* parametara koji služe za izvlacenje podataka iz obicnih datoteka (flat files). *Partition Exchange Loading* je poseban nacin ucitavanja podataka, ali za njega je potrebno da je odredište podijeljeno po particijama, te ga nisam koristio. Podkategorija *Hints* definira sugestije koje ce koristiti optimizator za optimizaciju izvlacenja i ucitavanja podataka, a *Constraint management* podkategorija definira da li ce se koristiti ogranicenja.



Slika 6.7 Fizicki parametri operatora u mapiranju

Na slici 6.8. su prikazani parametri koji utjecu na generiranje koda, odnosno pomocu kojih definiramo strategiju generiranja koda. Postoji samo jedna kategorija ovih parametara, ali zato sadrži mnogo parametara koji su jako bitni pa cu ih i opisati:

Default audit level – ovaj parametar definira razinu nadgledanja mapiranja tijekom izvodenja. Postoje cetiri razine nadgledanja koje su iste kao i kod skladišnog

modula: bez nadgledanja, nadgledanje na razini statisticke informacije, nadgledanje na razini pogrešaka i kompletno nadgledanje.

Setting Maximum Number of Errors – ovaj parametar definira maksimalni broj pogrešaka koji ce biti dozvoljen tijekom generiranja i izvodenja ovog mapiranja. Ako se taj broj prijede, generiranje ili izvodenje se prekida.

Commit Frequency – definira ucestalost s kojom ce se potvrdivati promjene u bazi podataka. Ovdje je definirano da se nakon svakih 1000 redaka potvrduju promjene.

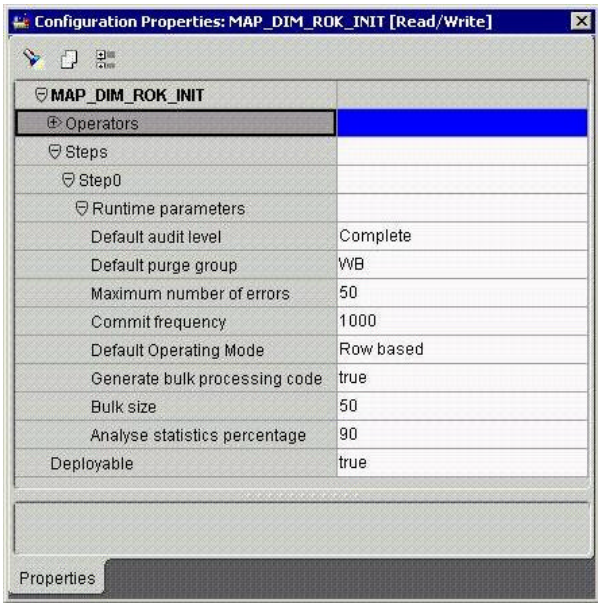
Default Operating Mode – ovaj parametar definira kako ce se generirati kod iz mapiranja. Ovdje je definirani nacin *Row based* što znaci da ce kao implementacijski jezik koristiti PL/SQL i da ce se obrada provoditi na razini retka.

Generate Bulk Processing Code – ovaj parametar ukljucuje ili iskljucuje *bulk* procesiranje. Naime, PL/SQL za izvlacenje podataka iz neke tablice koristi SQL upit. *Bulk* procesiranje znaci da ce PL/SQL koristiti jedan SQL upit za dobivanje više redaka. Ako je *bulk* procesiranje iskljuceno to znaci da ce PL/SQL za svaki redak podataka koristiti jedan SQL upit, što može jako usporiti izvodenje mapiranja.

Bulk size – definira koliko se redaka podataka dohvaca odjednom, ako je *bulk* procesiranje ukljuceno

Analyze Statistics Percentage – definira se postotak redaka koji ce se koristiti za procjenu statistike odredišnih tablica

Kada smo zadovoljni s konfiguracijom svih objekata možemo prijeci na sljedeci korak u implementaciji skladišta podataka, a to je generiranje i izvodenje skripti za definiranje objekata kao i za izvlacenje, transformaciju i ucitavanje podataka u skladište podataka.



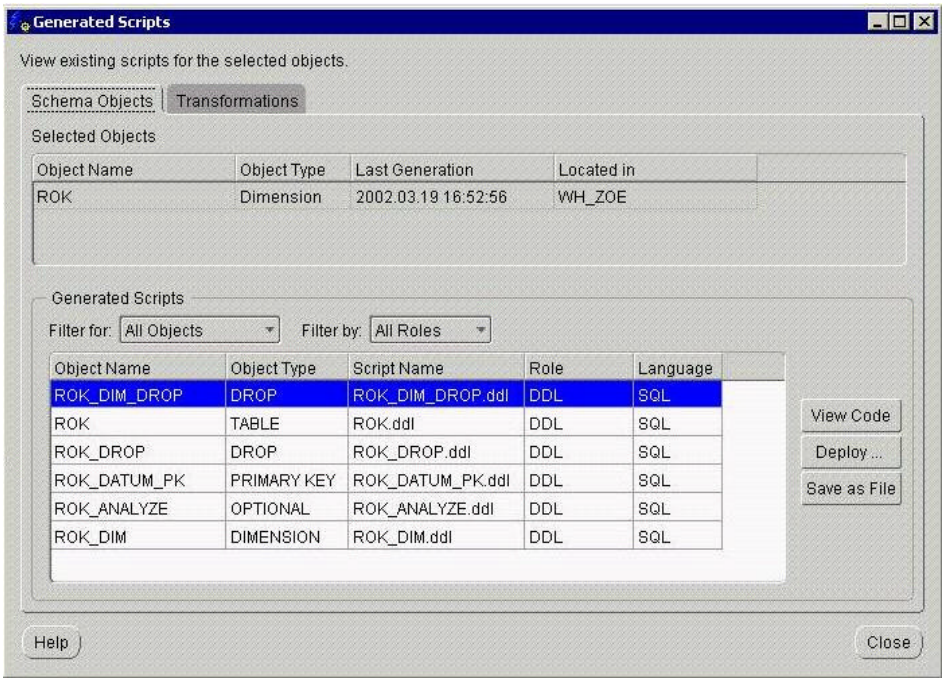
Slika 6.8 Fizicki parametri mapiranja koji utjecu na generiranje koda

7. Ucitavanje pocetnih podataka i osvježavanje skladišta podataka novim podacima

Generiranjem i pokretanjem skripti implementiramo fizicku instancu našeg skladišta podataka i pokrecemo ETL proces. Nakon pocetnog ucitavanja podataka u skladište moramo i osigurati mehanizme za periodicki punjenje i osvježavanje podataka u skladištu. Takoder moramo nakon svakog punjenja provjeriti tocnost ucitanih podataka, te ako je sve u redu objaviti te podatke kao pouzdane i tocne.

7.1. Generiranje i pokretanje skripti

Nakon definiranja logickog modela, definiranja mapiranja te konfiguriranja svih tih objekata potrebno je za svaki objekt generirati skripte. Da bi generirali skripte potrebno je pritisnuti desnu tipku miša iznad odabranog objekta, te iz izbornika odabrati opciju *Generate.* Tada nam se otvara novi prozor koji izgleda kao na slici 7.1., a sadrži informacije o generiranim skriptama. Na slici je prikazan primjer za generirane skipte za dimenziju ROK.

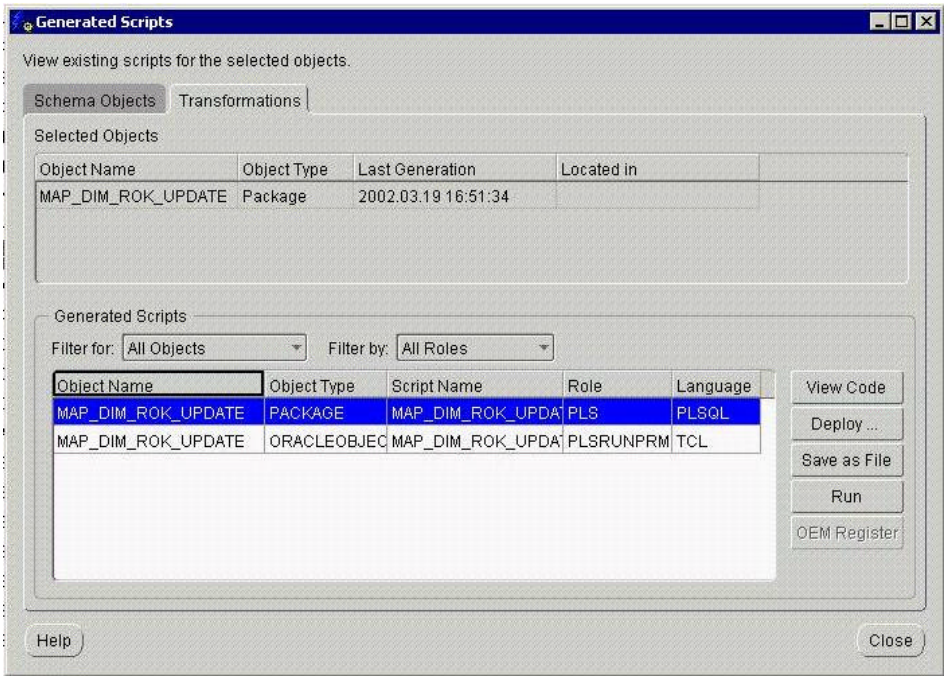


Slika 7.1 Prozor s informacijama o generiranim skriptama za dimenziju ROK

Kao što vidimo u donjem dijelu ekrana se pojavljuje informacija o imenu objekta, tipu objekta, imenu skripte, ulogu koja ima ta skripta i jeziku u kojem je napisana. Za objekte poput dimenzija, tablica cinjenica, tablica, i sl., OWB generira

takozvane DDL skripte (DDL – Data Definition Language) koje služe za kreiranje objekata. Na desnoj, donjoj strani ekrana nalaze se tri tipke od kojih su nam bitne prve dvije: *View Code* i *Deploy*. Tipka *View Code* nam omogucuje da vidimo generirani kod za odabranu skriptu, dok nam tipka *Deploy* omogucuje da tu skriptu i implementiramo na željeno racunalo gdje nam je i skladište podataka. Implementacijom DDL skripti ujedno i stvaramo fizicku instancu objekta za koji je ta skripta napisana. Oznacavanjem svih objekata na ekranu i pritiskom na tipku *Deploy* smo u stvari fizicki instancirali dimenziju ROK prema definiciji i konfiguraciji koju smo proveli ranije.

Postupak generiranja skripti za mapiranja je slican postupku generiranja skripti za ostale objekte. Ekran s informacijama o generiranim skriptama je vrlo slican onome na slici 7.1., ali ipak ima nekih razlika. Na slici 7.2. je prikazan ekran s generiranim skriptama za mapiranje MAP_DIM_ROK_UPDATE.



Slika 7.2 Ekran s informacijama o generiranim skriptama za mapiranje MAP_DIM_ROK_UPDATE

Kao što vidimo za svako mapiranje se generiraju dvije skripte: PL/SQL skripta namjenjena za izvodenje ETL procesa i TCL skripta koja služi za registraciju skripte kao posla (engl. job) u *Oracle Enterprise Manageru* ili slicnim programima.

Za PL/SQL skriptu se takoder može vidjeti generirani kod pritiskom na tipku *View Code*. Pritiskom na tipku *Deploy* ta skripta se integrira u skladište podataka, ali

da bi napravili svoj posao potrebno ju je i pokrenuti s tipkom *Run*. Pokretanjem skripte ona obavlja posao za koji je napravljena. U posebnom prozoru možemo vidjeti rezultate tog obavljenog posla, eventualne pogreške i status posla. Pokretanjem skripte se obavlja izvlacenje podataka iz definiranih izvora i punjenje odredišta tim podacima.

TCL je kontrolna skripta koja služi za registriranje posla u OEM. Kada se neka skripta registrira kao posao u OEM, tada se može pokretanje te skripte automatizirati tj. odrediti kada ce se pokrenuti. Takoder se za sve poslove mogu pomocu *Oracle Workflowa* odrediti njihove medusobne ovisnosti i na taj nacin možemo definirati kompletan proces kojeg opet možemo automatizirati pomocu OEM. Medutim za naše potrebe bio je dovoljan i samo OWB.

Ovdje takoder treba napomenuti da je redoslijed implementacije objekata bitan jer je potrebno implementirati dimenzije, prije implementacije tablice cinjenica. Generiranjem i pokretanjem svih skripti, za sve objekte i mapiranja, implementirali smo fizicku instancu našeg skladišta i napunili smo je pocetnim podacima. Medutim ostaje nam još riješiti problem periodickog osvježavanja i punjenja skladišta podataka novim podacima kao i provjere tocnosti ucitanih podataka.

7.2. Periodicko osvježavanje i punjenje skladišta podataka novim podacima

Kad smo napunili skladište podataka pocetnim podacima, potrebno je takoder osigurati mogucnost da se skladište podataka nadopunjava novim podacima. Taj proces nadopunjavanja novim podacima je možda i najosjetljiviji dio ETL procesa. To je zato što je potrebno raspoznati koje podatke vec imamo u skladištu, a koje bi trebalo ucitati u skladište što ovisno o situaciji zna biti vrlo kompliciran problem.

Medutim u našem slucaju ipak nije tako kompliciran. Pošto nemamo pravu dimenziju vremena, vec imamo dimenziju ROK koja ju zamjenjuje, moguce je da se novi podaci u tablice cinjenica unose po završetku obrade roka. Nakon unosa podataka potrebno je podatke provjeriti i objaviti do kojeg roka je tablica cinjenica valjana, tako da se zna koji je iduci rok kojeg trebamo unijeti. Na taj nacin se može osigurati nadopunjavanje skladišta podataka bez velikih problema.

Što se tice nadopunjavanja i osvježavanja dimenzijskih tablica, situacija je malo kompliciranija. Dimenzijske tablice ne možemo nadopunjavati po nekom kljucu

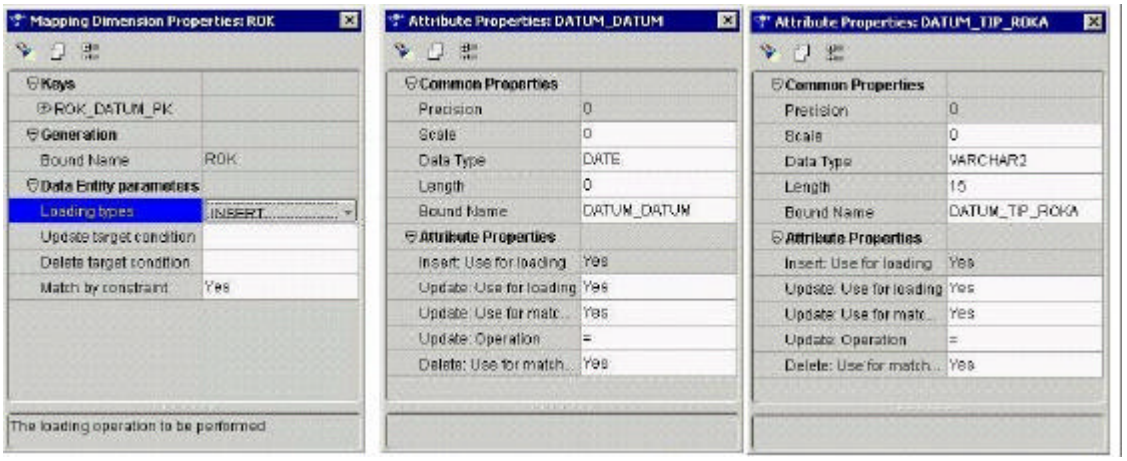
vec moramo uvijek odabirati sve moguce podatke iz izvora i onda od tih podataka odabrati one koji su nam novi i njih unijeti u dimenzijske tablice. Kako se to može napraviti pokazat cu na primjeru dimenzije ROK koja ce se ujedno i najviše nadopunjavati.

Za dimenziju ROK napravio sam dva mapiranja: MAP_DIM_ROK_INIT i MAP_DIM_ROK_UPDATE. Prvo mapiranje puni dimenziju pocetnim podacima, a drugo služi za nadopunu. Ovaj problem se mogao riješiti i samo s jednim mapiranjem, ali sam htio pokazati razliku pa sam napravio dva mapiranja. Razlika u ova dva mapiranja je u konfiguraciji atributa i odredišnog operatora. Na slici 7.3. je prikazana konfiguracija operatora dimenzije ROK, kao i konfiguracija atributa DATUM_DATUM i DATUM_TIP_ROKA u mapiranju MAP_DIM_ROK_INIT. Na slici 7.4. je su prikazane konfiguracije za isti operator i iste atribute, ali u mapiranju MAP_DIM_ROK_UPDATE.

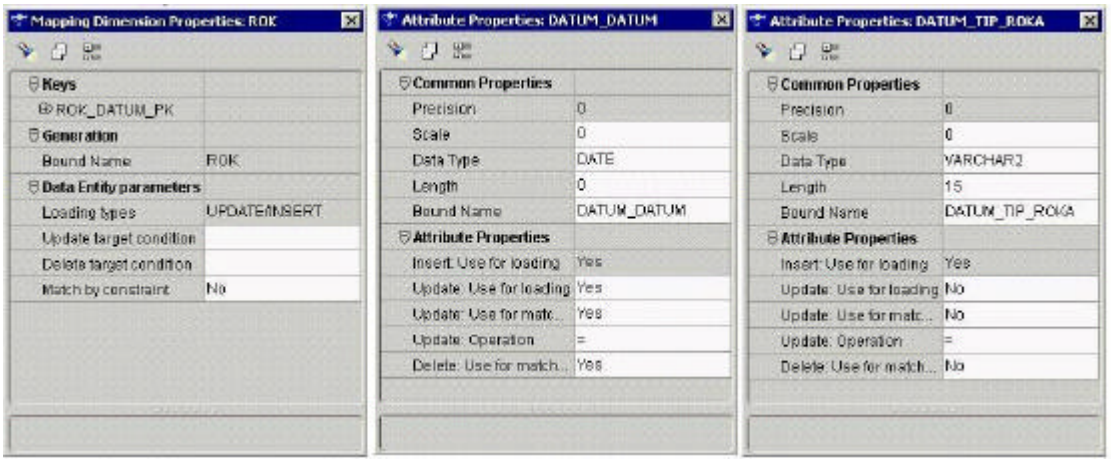
Kao što se vidi sa slika, konfiguracije operatora i atributa DATUM_TIP_ROKA su razlicite, dok je konfiguracija za atribut DATUM_DATUM ista. Prilikom konfiguriranja operatora i atributa u mapiranju MAP_DIM_ROK_INIT bilo je zamišljeno da to mapiranje puni dimenziju ROK pocetnim podacima. To znaci da prije pokretanja tog mapiranja nije bilo nikakvih podataka u dimenziji ROK. Stoga su svi podaci mogli biti umetnuti u dimenziju ROK operacijom *Insert* bez provjere da li ti podaci vec postoje, pa je i mapiranje tako konfigurirano (u konfiguraciji operatora je postavljeno da ce se izvoditi operacija INSERT).

Medutim, ako želimo nadopunjavati dimenziju, mapiranje MAP_DIM_ROK_INIT nije prikladno jer ono samo umece podatke u tablicu bez obzira da li neki podaci vec postoje ili ne. Nadopunjavanje se vrši pokretanjem mapiranja MAP_DIM_ROK_UPDATE koje je i konfigurirano u tu svrhu. Naime, u konfiguraciji operatora dimenzije ROK u tom mapiranju je u parametru *Loading Types* postavljena vrijednost UPDATE/INSERT i u parametru *Match by constraint* je postavljena vrijednost *No*. Takav postav nalaže da se za svaki redak koji ulazi u dimenziju ROK prvo pokuša provesti operacija UPDATE, a ako ona ne uspija obavlja se operacija INSERT. Za odluku koji ce redak biti podvrgnut operaciji UPDATE ne gleda se kljuc vec se ispituju atributi koji imaju ukljucenu opciju *Update: Use for matching*. U našem slucaju to je atribut DATUM_DATUM jer su rokovi poslagani po datumu pa je logicno ispitivati da li vec postoji taj datum u dimenziji. Za ostale

atribute koji ne sudjeluju u provjeri vrijednosti iskljucene su sve opcije osim *Insert: Use for Loading* tako da se oni ne ispituju, ali se umecu u dimenziju. Mapiranje radi tako da se prilikom punjenja dimenzije provjerava da li vec postoji u dimenziji redak koji ima istu vrijednost atributa DATUM_DATUM kao i nadolazeci. Ako ima onda se taj redak ne umece u dimenziju ROK, a ako nema istu vrijednost onda se umece na kraj dimenzije ROK. Ovakav postupak se može koristiti za nadopunjavanje bilo koje dimenzije.



Slika 7.3 Konfiguracija operatora i atributa u mapiranju MAP_DIM_ROK_INIT



Slika 7.4 Konfiguracija operatora i atributa u mapiranju MAP_DIM_ROK_UPDATE

7.3. Provjera ucitanih podataka

Nakon ucitavanja pocetnih podataka u skladište podataka potrebno je provjeriti tocnost ucitanih podataka. Provjera tocnosti zapravo znaci da li ucitani podaci odgovaraju stvarnim podacima koji su u operacijskoj bazi. Provjera podataka je jako bitna jer ako su ucitani podaci krivi skladište podataka ne prikazuje pravu sliku stanja.

U našem slucaju provjera ucitanih podataka je bila relativno jednostavna. Što se dimenzijskih tablica tice jednostavno sam usporedio ucitane podatke s onima u operacijskoj bazi i ustanovio da su ucitani podaci tocni. Jednostavna provjera dimenzijskih tablica je bila moguca zbog relativno malo podatka prisutnih u operacijskoj bazi pa se jednostavnim pregledavanjem moglo provjeriti da li su podaci ucitani u skladište podataka tocni.

Što se tice tablice cinjenica situacija je bila ipak malo problematicnija. Podatke u tablici cinjenica sam provjeravao u dva dijela. Prvi dio provjere je obuhvacao nadzor nad ucitavanjem u tablice cinjenica i kontrola broja redaka koji se ucitavaju u tablice cinjenica jer sam znao koliko bi se redaka trebalo ucitati za neki rok (npr. znao sam da je na prvi kolokvij izašlo 739 studenata pa sam ocekivao 739 redaka koji bi tijekom obrade morali izvuci iz operacijske baze i kasnije sažeti u manji broj i ucitati u skladište podataka). Drugi dio provjere je obavljen nakon ucitavanja, a obuhvacao je postavljanje upita za koje sam vec imao rezultate iz operacijske baze i usporedba dobivenih rezultata iz skladišta podataka s onim koje sam vec imao. Na taj nacin sam obavio provjeru ucitanih podataka u tablicama cinjenica. Zakljucak je da su podaci u skladištu dobro ucitani.

Zakljucak

U ovom diplomskom radu sam obradio kompletan postupak izgradnje skladišta podataka – od teorijskih osnova do implementacije korištenjem programskog paketa Oracle Warehouse Builder. Skladištenje podataka je koncept koji se pojavio u zadnjih nekoliko godina, a u prvom redu služi kao potpora procesa poslovnog izvješcivanja. Skladištenje podataka omogucava lakše i brže izvještavanje i efikasnije korištenje raspoloživih resursa. U ovom radu sam izradio skladište podataka koje bi trebalo služiti profesorima i asistentima sa Zavoda za osnove elektrotehnike i elektricka mjerenja u lakšem dobivanju povratnih informacija o uspjehu studenata na ispitima iz njihovih kolegija.

Sam proces izgradnje skladišta podataka je vrlo složen i zahtjevan proces, te nadasve odgovoran posao. Tijekom izgradnje skladišta potrebno je definirati od kuda ce se izvlaciti podaci (izvori), potrebno je definirati logicki model skladišta u kojeg ce se spremati izvuceni podaci. Takoder je potrebno definirati proces izvlacenja, transformacije i ucitavanja podataka iz izvora u skladište podataka (poznatiji kao ETL proces) što se radi pisanjem skripti u raznim programskim jezicima (SQL, PL/SQL,…). Nakon definiranja logickog dijela, potrebno je implementirati stvarno skladište, ucitati u njega podatke te najvažnije provjeriti da li su ti podaci tocni i da li su dobro ucitani. Proces izgradnje skladišta podataka obuhvaca najveci dio skladištenja podataka.

Za izradu skladište koristio sam Oracle-ov programski paket Oracle Warehouse Builder. Kako je OWB relativno nov programski paket s kojim nitko još nije imao previše iskustva, mnogo sam vremena potrošio proucavajuci razne prirucnike i eksperimentirajuci po ekranima OWB, ali kad sam shvatio kako radi nije više bilo problema. OWB je programski paket koji je jednostavan za korištenje, te znatno ubrzava izgradnju skladišta podataka. OWB, uz ostalo, ipak posebno olakšava definiciju ETL procesa. Naime u OWB nije potrebno pisati skripte rucno, vec OWB generira kod prema definicijama i konfiguracijama koje smo sami napravili. To omogucuje veliku uštedu u vremenu razvoja i implementacije skladišta podataka, te na taj nacin se i povecava efikasnost tima koji izraduje skladište podataka.

Što se tice ovog konkretnog rada i izgradnje skladišta za Zavod za osnove elektrotehnike i elektricka mjerenja, smatram da je projekt bio uspješan. U tome je

OWB bio od velike pomoci. Uspio sam napraviti skladište podataka i u njega ucitati tocne podatke. Ono što bi još trebalo napraviti da bi skladište bilo kompletno, je izgraditi sustav izvještavanja. Znaci, potrebno je postojece skladište nadograditi s nekim alatom za izradu izvještaja, i možda izraditi portal preko kojeg ce svi zaposlenici zavoda moci pristupiti skladištu korištenjem lokalne mreže. Bez mogucnosti dobivanja izvještaja samo skladište nema smisla ni koristi.

Na kraju mogu reci da sam zadovoljan s napravljenim. Mislim da je projekt uspio iako ce na potpunu ocjenu rada trebati pricekati neko vrijeme. Također sam zadovoljan i sa samom temom diplomskog rada i mogu reci da sam puno naucio i da ce mi to kasnije puno pomoci. Spoznao sam neke probleme koji se javljaju samo u stvarnom svijetu i ovaj projekt mi je bio veliko iskustvo i poticaj za daljnji rad i usavršavanje.

Popis literature

Knjige:

[1] R. Kimball, *The Data Warehouse Toolkit*, Wiley Computer Publishing, 1996.

[2] Oracle Corporation, *Oracle Warehouse Builder User's Guide*, Oracle Corporation, 2001.

[3] Oracle Corporation, *Data Warehousing Guide*, Oracle Corporation, 1999.

[4] Z. Skocir, I. Matasic, Vrdoljak, *Organizacija obrade podataka*, interna skripta, Zavod za telekomunikacije, 2001.

[5] Oracle Corporation, *PL/SQL User's Guide and Reference*, Oracle Corporation, 1997.

Clanci:

[6] V. R. Grupta, *An Introduction to Data Warehousing*, www.system-services.com

[7] R. Kimball, *A Dimensional Modeling Manifesto*, www.dbmsmag.com

[8] IBM Corporation, *Data Warehousing Concepts for AS/400*, IBM Corporation, 1996.