UNIVERZITET U BEOGRADU

FAKULTET ORGANIZACIONIH NAUKA

Dragoljub Krneta

AUTOMATIZACIJA FIZIČKOG PROJEKTOVANJA SKLADIŠTA PODATAKA PROŠIRENOG DATA VAULT PRISTUPA

doktorska disertacija

Beograd, 2014

UNIVERSITY OF BELGRADE

FACULTY OF ORGANIZATIONAL SCIENCES

Dragoljub Krneta

PHYSICAL DATA WAREHOUSE DESIGN AUTOMATION AN EXPANDED DATA VAULT APPROACH

Doctoral Dissertation

Belgrade, 2014

Podaci o mentoru i članovima komisije

Mentor:

Prof. dr Zoran Marjanović, redovni profesor
Univerzitet u Beogradu, Fakultet organizacionih nauka

Članovi komisije:

Prof. dr Milija Suknović, redovni profesor
Univerzitet u Beogradu, Fakultet organizacionih nauka

Prof. dr Vladan Jovanović, redovni profesor
Georgia Southern University, Allen E. Paulson College of Engineering and Information Technology, USA

Datum odbrane: _____

Automatizacija fizičkog projektovanja skladišta podataka proširenog Data Vault pristupa

Rezime:

Skladište podataka predstavlja podlogu za implementaciju sistema poslovne inteligencije, integraciju podataka i podršku poslovnom odlučivanju. Za skladišta podataka uglavnom se izgrađuje dimenzioni model kojim se dobija bolja mogućnost vizualizacije podataka. Arhitektura skladišta podataka podrazumijeva definicije osnovnih elemenata skladišta podataka i opisuje kako su ti elementi izgrađeni i povezani. Kod arhitekture skladišta podataka danas postoje dva dominantna pristupa. Jedan pristup, poznat je pod nazivom CIF (Corporate Information Factory) arhitektura, zastupa Bill Inmon koji smatra da skladište podataka treba da bude normalizovano i da sadrži atomske podatke u trećoj normalnoj formi. Drugi pristup, poznat poda nazivom BUS arhitektura, zastupa Ralph Kimbal koji tvrdi da i samo skladište treba da ima višedimenzionalnu strukturu. Data Vault koncept skladišta podataka koji je osmislio Daniel Linstedt, predstavlja detaljno orijentisani, istorijski praćen i jedinstveno povezani skup normalizovanih tabela koji podržavaju jedan ili više funkcionalnih područja poslovanja. To je hibridni pristup koji obuhvata najbolje osobine između 3. normalne forme (3NF) i zvijezda šeme. Data Vault koncept naglašava potrebu da se ostavi trag odakle i kad su podaci došli u bazu podataka, te omogućuje lake promjene strukture skladišta podataka u skladu sa promjenama u poslovnom okruženju.

Skladišta podataka se izrađuju različitim metodološkim pristupima. Međutim, problem je što proces dizajniranja skladišta podataka i izrade dimenzionog modela nije automatizovan. U disertaciji je sagledan problem dizajniranja skladišta podataka na osnovu šire literature, te istražena mogućnost automatizacije fizičkog projektovanja skladišta podataka baziranog na Data Vault konceptu uz minimalnu interakciju sa korisnikom.

U radu je predstavljen osnovni algoritam za automatizaciju fizičkog projektovanja skladišta podataka, baziranog na Data Vault konceptu, koji je zasnovan na modelu metapodataka transakcionih izvora podataka i modelu pravila. Algoritam za direktni pristup fizičkog projektovanja skladišta podataka formalizuje, generalizuje i u velikoj mjeri automatizuje proces fizičkog projektovanja skladišta podataka na osnovu fizičkog modela struktuiranih, polustruktuiranih i jednostavnijih nestruktuiranih izvora podataka. Relacioni model je zajednički činilac i za Data Vault implementaciju i za izvorne podatke i zato se svi tipovi najprije abstrahuju metmodelom. Naziv ***direktni pristup*** *(engl. direct approach) upravo znači da sa se u predloženom pristupu direktno dolazi do fizičkog modela Data Vaulta (engl. Physical Data Vault - PDV) na osnovu fizičkog modela izvora podataka u relacionom obliku. Konceptualizacija metapodataka predstavljena je fizičkim modelom podataka koji se uz manje izmjene može koristiti u projektovanju individualnih skladišta podataka i predstavlja osnovni metodološki doprinos rada. Osim toga, rad daje cjelokupan integrisan pristup automatizaciji projektovanja skladišta podataka jer pored automatizacije projektovanja samog skladišta podataka, pristup omogućuje projektovanje data martova, i projektovanje ETL procesa (iz izvora u skladište podataka, te iz skladišta podataka u data martove). Na ovaj način dobijamo integraciju svih potrebnih aktivnosti koje su potrebne za realizaciju kompletnog skladišta podataka.*

Na osnovu rezultata istraživanja u disertaciji urađen je prototip aplikacije za podršku fizičkom projektovanju skladišta podataka. Simulacija algoritma, uz podršku prototipa aplikacije provedena je na stvarnom slučaju u oblasti zdravstva i zdravstvenog osiguranja i daje dobre rezultate.

Ključne riječi: *Skladište podataka, metamodel, fizički model, arhitektura skladišta podataka, dimenzioni model, Data Vault, automatizacija projektovanja.*

Naučna oblast: *Informacioni sistemi i tehnologije*

Physical Data Warehouse Design Automation
an Expanded Data Vault Approach

Abstract:

Data warehouse is a basis for the implementation of business intelligence systems, data integration and decision support. Usually, a dimensional model is built for data warehouse in order to get better data visualization. Data warehouse architecture represents definitions of basic data warehouse elements and describes the connection and structure of the elements. There are two dominant approaches in data warehouse architecture. First one is known as CIF (Corporate Information Factory) architecture and it is represented by Bill Inmon who believes that data warehouse needs to be normalized and needs to contain atomic data in third normal form. Second approach, known as BUS architecture, is represented by Ralph Kimball who believes that warehouse should also have multi-dimensional structure. Data Vault concept of data warehouse which was designed by Daniel Linstedt and represents detail oriented, historically followed and uniquely connected group of normalized tables which support one or more functional areas of a business. That is a hybrid approach which covers the best features from third normal form (3NF) and star scheme. Data Vault concept emphasizes the need to leave a trace from where and when data was sent into database, and also enables easy changes in data warehouse structure in accordance with business environment changes.

Data warehouses are built in different methodological approaches. However, there is a problem with the fact that the process of data warehouse design and related dimensional model is not automatized. In this dissertation a problem of data warehouse design process was accessed, based on broad analysis of literature, and a possibility of automation of physical data warehouse design based on Data Vault concept, with a minimal interaction with users, researched. The work presents simple algorithm for

automation of data warehouse physical design, based on Data Vault concept, using data model with metadata transactional data sources and rules. Algorithm for direct access to physical design of data warehouse formalizes, generalizes and largely automates the process of physical design for data warehouses based on physical model of structured, semi-structured and simple non-structured data sources. Relational model is a common factor both for Data Vault implementation and for original data and that is why all types are abstracted with meta-model. The ***direct approach*** *means that Physical Data Vault - PDV is directly realized based on physical data source model in relational form. Conceptualization of metadata is presented with physical data model which, with small changes, can be used in design of individual data warehouses and represents basic methodological contribution of the work. Besides that, this work gives overall integrated approach to automation of data warehouse design because the approach enables design of data marts and of related ETL processes (from a source to data warehouse and from a data warehouse to data marts). In this way integration of all activities needed for realization for entire data warehouse is obtained.*

Based on results of the research in dissertation, a prototype of the application is made for support of physical design of data warehouse. Experiments with the algorithm, using a prototype of the design automation application, on a real case in the field of healthcare insurance, produced satisfactory results.

Key words: *Data warehouse, meta-model, physical model, data warehouse architecture, dimensional model, Data Vault, design automation.*

Scientific field: *Information systems and technology.*

Izjave zahvalnosti

Ovaj doktorat posvećujem mojoj porodici: supruzi Sanji, sinu Stefanu i kćerki Sofiji. Zahvaljujem im se na velikoj podršci i razumijevanju. Njihova beskrajna ljubav mi je pomogla da istrajem i uvijek je bila izvor moje motivacije, optimizma i težnje ka ljepšim i boljim stvarima u životu.

Zahvaljujem se ocu Urošu i majci Bosiljki na neizmjernoj ljubavi koja je bila moj najveći oslonac u životu. Zahvaljujem im se što su me naučili poštenju i što su mi uvijek naglašavali značaj porodice, obrazovanja, upornosti i strpljenja.

Želim da se zahvalim profesoru dr Zoran Marjanoviću što je prihvatio da mi bude mentor i što je moje ideje usmjerio u naučno-istraživačkom pravcu. Zahvaljujem mu se na svim savjetima, primjedbama, prijedlozima i na angažovanju tokom doktorskih studija.

Veliku zahvalnost dugujem profesoru dr Vladanu Jovanoviću, za podršku u mom radu na doktorskoj disertaciji i naročito naučnim radovima proisteklim iz teme disertacije. Zahvaljujem mu se što proširio moj pogled na *data warehouse*, te na sugestijama, angažovanju i strpljenju tokom izrade disertacije.

Zahvaljujem se profesoru dr Miliji Suknoviću na korisnim prijedlozima vezanim za sadržaj i kvalitet doktorske disertacije.

Želim da se zahvalim prof. dr Branku Perišiću i posebno doc. dr Gordani Milosavljević sa FTN Novi Sad koji su u značajnoj mjeri uticali na moju stručnu karijeru i od kojih sam mnogo naučio u oblasti projektovanja i razvoja informacionih sistema.

Zahvaljujem se mr Nebojši Niniću osnivaču i generalnom direktoru kompanije Lanaco IT što je prepoznao moj potencijal i povjerio mi organizovanje i upravljanje Sektorom za razvoj softvera u kompaniji. Zahvaljujem mu se i na svoj podršci meni i mojoj porodici.

Sadržaj

1. UVOD

U uvodnom dijelu disertacije data je formulacija problema i predmeta istraživanja, motivacija za istraživanje, opisan je cilj istraživanja i hipoteze u istraživanju, te je ukratko opisan metod i faze urađenog istraživanja. Na kraju uvodnog dijela rada dat je sažet pregled ostalih poglavlja disertacije.

1.1. Predmet i problem istraživanja

U ovom dijelu se navode definicije osnovnih pojmova od interesa (engl. *terms of reference*) iz šire oblasti istraživanja a zatim ukazuje na užu oblast istraživanja (automatizacija projektovanja skladišta podataka) i izabrani problem istraživanja, pri čijem rješavanju su ostvareni doprinosi predstavljeni ovom disertacijom. Problem istraživanja se može eksplicitno okarakterisati kao metod automatizacije fizičke integracije podataka, uz potrebno očuvanje istorije promjena.

Predmet istraživanja ovog rada u širem smislu obuhvata sljedeće oblasti: baze podataka, skladišta podataka, poslovnu inteligenciju i podršku odlučivanju. Razvojem informacionih tehnologija i Interneta i njihovom širokom primjenom, upravljačkim strukturama u kompanijama postale su dostupne velike količine informacija. Najveći dio tih podataka se nalazi u transakcionim relacionim bazama podataka. Radi donošenja pravih poslovnih odluka menadžerima su potrebne prave informacije u pravom formatu i u realnom vremenu. Zahtjevi za informacijama dodatno opterećuju transakcione sisteme. Skladišta podataka su uglavnom dizajnirana prema dimenzionom modelu koji pruža mogućnosti lakšeg pristupa podacima i vizualizacije podataka.

Skladište podataka (engl. *Data Warehouse, DW*) predstavlja podlogu za implementaciju sistema poslovne inteligencije (engl. *Business Intelligence, BI*), integraciju podataka i podršku poslovnom odlučivanju. Poslovna inteligencija je generički termin za opisivanje uticaja unutrašnjih i vanjskih informacija organizacije za donošenje boljih poslovnih odluka [KR02]. Kao specifična vrsta baza podataka, skladište podataka su se razvila krajem 80-tih i početkom 90-tih godina prošlog vijeka. Ralph Kimball je 1986. godine napravio tzv. Red Brick Systems [Kim12]. koji je bio zasnovan na relacionoj bazi podataka optimizovan za skladištenje podataka koji je u to vrijeme imao performanse i do deset puta bolje u odnosu na tadašnje baze podataka. Eftimie i Briquez su 1984. godine su predstavili tehnike modelovanja i dizajna skladišta informacija (engl. *Information Warehouse*) radi podrške menadžmentu kompanija u donošenju poslovnih odluka [EB84]. Na projektu integracije informacionog sistema brodogradilišta 3. maj u Rijeci 1986. god. i za potrebe informacionog sistema Privredne komore Jugoslavije 1988. godine, tim pod rukovodstvom Vladana Jovanovića i Zorana Marjanovića je nezavisno razvio sistem za integraciju podataka koristeći multidimenzioni dizajn[1]. Devlin i Murphy su 1988. godine objavili prvi rad koji opisuje arhitektura skladišta podataka [DM88]. Bill Inmon je 1992. godine objavio prvu knjigu o skladištenju podataka [Inm92]. Osnovni razlog koji je doprineo nastanku i razvoju skladišta podataka bila je nemogućnost tadašnjih baza podataka (engl. *Database, DB*), odnosno OLTP (engl. *On-Line Transaction Processing*) informacionih sistema da podrže sve veće zahtjeve za obradu i analizu poslovnih informacija uključujući istorijske i eksterne, te zahtjeve za estimacije i predviđanja. Skladište podataka danas ima vrlo važnu ulogu za analizu podataka i podršku odlučivanju. Prema podacima istraživačke agencije Gartner (www.gartner.com) iz kraja 2010. godine, sistemi

[1] Informacija dobijena na osnovu usmene komunikacije

poslovne inteligencije se nalaze na prvom mjestu od prvih deset tehnoloških prioriteta kompanija u SAD-u. Od 2000. godine razvoj i investicije u DW prevazišle su one u DB sistemima i taj se trend ubrzano nastavlja sa eksplozivnim razvojem storage sistema, najprije SAN (engl. *Storage Area Network*), a sada Virtualizacija (engl. *Virtualization*) i Cloud. Globalizacija poslovanja, Internet i razvoj Web baziranih aplikacija dodatno je uticao na povećanje korišćenja skladišta podataka. Popularnost skladišta podataka svakim danom je sve veća i sve više kompanija ga koristi.

Skladište podataka je predmetno orijentisana, integrisana, stalna i vremenski promjenjiva kolekcija podataka za podršku menadžmentu kod odlučivanja [Inm92]. Donošenja poslovnih odluka na osnovu raspoloživih podataka i znanja u eri globalizacije i u uslovima dinamičnog poslovnog okruženja, zauzima značajno mjesto u obezbeđivanju tržišne pozicije kompanija. Treba napomenuti da se skladišta podataka široko koriste u državnim i neprofitnim organizacijama, a takođe i za organizovanje naučnih i tehničkih podataka. U ovom radu je interes posebno usmjeren na poslovne sisteme.

U užem smislu, predmet istraživanja disertacije je proces dizajniranja, odnosno izrade modela skladišta podataka. Kod transakciono orijentisanih relacionih baza podataka (OLTP) izrađuje se najčešće normalizovan model podataka, što povlači složen način pristupa podacima. Za skladišta podataka (posebno na strani korisnika) uglavnom se izgrađuje dimenzioni model kojim se dobija bolja mogućnost vizualizacije podataka. Kod dimenzionih baza podataka relacije su podijeljene u dvije kategorije: tabelu mjera ili činjenica (engl. *fact table*) i više tabela dimenzija (eng. *dimension tables*). Tabela mjera ima dvije vrste atributa: ključne atribute koji su ujedno spoljni ključevi pomoću kojih je tabela činjenica

povezana sa tabelama dimenzija i same činjenice koje sadrže podatke koji se analiziraju. Tabele su najčešće spojene u šemu zvijezde (engl. *star schema*).
Postupak prikupljanja i obrade podataka za potrebe punjenja skladišta podataka naziva se ETL (engl. *Extraction, Transformation and Loading*) proces. Data mart (DM) je oblik skladišta podataka koji najčešće sadrži već sumirane podatke na određenim organizacionim dijelovima kompanije ili po određenoj grupi poslovnih procesa. Data mart je mjesto gdje krajnji korisnik ima direktan pristup i kontrolu njegovih analitičkih podataka. Finansije imaju svoje podatke, marketing i prodaja svoj data mart, itd. Svaki data mart obično drži znatno manje podataka nego skladište podataka [Inm02]. Kod arhitekture skladišta podataka danas postoje dva dominantna pristupa. Jedan pristup, poznat je pod nazivom CIF (engl. *Corporate Information Factory*) arhitektura, zastupa Bill Inmon koji smatra da skladište podataka treba da bude normalizovano i da sadrži atomske podatke u trećoj normalnoj formi (3NF), te da koristi višedimenzione data mart-ove. Drugi pristup, poznat poda nazivom BUS arhitektura, zastupa Ralph Kimbal koji tvrdi da i samo skladište treba da ima višedimenzionu strukturu.

Data Vault koncept skladišta podataka je osmislio Daniel Linstedt objavivši sredinom 2002. godine prvi od pet članaka pod nazivom Data Vault Series. Ostali članci su objavljeni do početka 2005. na sajtu http://www.tdan.com/view-articles/5054/. Data Vault se pojavio kao odgovor na probleme koji su pratili razvoj skladišta podataka dizajniranih u trećoj normalnoj formi i star šemi, nastojeći iskoristiti najbolje elemente jednog i drugog koncepta. Performanse i druge slabosti 3NF i zvijezda shema (kada se koristi za skladište podataka) počele su se pojavljivati u 90-im kada se količina podataka počela povećavati. Data Vault je projektovan za prevazilaženje tih nedostataka zadržavajući prednosti 3NF i zvijezda shema arhitekture [Lin1]. Data Vault predstavlja

detaljno orijentisani, istorijski praćen i jedinstveno povezani skup normalizovanih tabela koji podržavaju jedan ili više funkcionalnih područja poslovanja. To je hibridni pristup koji obuhvata najbolje osobine između 3. normalne forme i zvijezda šeme. Dizajn je fleksibilan, skalabilan, dosljedan i prilagodljiv potrebama preduzeća i organizacija [Lin1] [Lin2]. Linstedt uvodi *hub, link i satelit* entitete. *Hub* predstavlja poslovni ključ, *link* entiteti omogućuje transakcije između hubova, dok satelitski entiteti daju kontekst hub primarnom ključu.

Data Vault koncept naglašava potrebu da se ostavi trag odakle i kad su podaci došli u bazu podataka [Lin10]. Osim toga, ovaj koncept je osmišljen da se model podataka može lako promijeniti u skladu sa promjenama u poslovnom okruženju [Dam08], te kako bi se omogućilo paralelno i brzo učitavanje podataka koliko je god to moguće [HS10]. Dizajn Data Vault sistema je jednostavan sa minimalnim brojem komponenti kojeg čine *hub*, *link* i *satelit* entiteti. Kod Data Vault sistema u praksi se ponekad primjenjuje i minimalni broj pomoćnih tabela tzv. reference data.

Projekat poslovne inteligencije i projekat izgradnje skladišta podataka se u organizacijama najčešće tretiraju kao jedan projekt, jer skladište podataka podloga za korišćenje alata poslovne inteligencije. Kako sistem poslovne inteligencije i skladišta podataka uglavnom ne postoji kao gotov proizvod za svaku organizaciju ili djelatnost, proizvođači softvera nude razne tehnološke platforme, alate, znanja i preporuke za njihov razvoj. Skladišta podataka se izgrađuju različitim metodološkim pristupima, uzimajući u obzir specifičnosti svake organizacije. Metodologija izgradnje skladišta podataka (uključujući popularne kao Kimball DWLC Toolkit, sa varijantom prilagođenom za Microsoft SQL Server 2008, ali i najnovije kao [Corr11]) nije u potpunosti

formalizovana, već sadrži niz preporuka kojih bi se trebalo držati. Izgradnja skladišta podataka uglavnom počinje definisanjem korisničkih zahtjeva i upoznavanjem projektanata sistema sa njihovim željama i potrebama. Nakon analize zahtjeva korisnika i upoznavanja sa strukturom transakcione baze podataka, prelazi se na izradu logičkog modela, a zatim i fizičkog modela skladišta podataka.

Međutim, jedan problem je što proces dizajniranja skladišta podataka i izrade dimenzionog modela nije automatizovan. Dimenzioni model sa tabelama mjera i dimenzija se u većini slučajeva radi ručno na osnovu analize zahtjeva korisnika i analize transakcionog sistema, bilo da se radi o Inmon-ovoj ili Kimbal-ovoj metodologiji ili se radi o primjenama po Data Vault konceptu. Problem neautomatizovane izrade dimenzionog modela skladišta podataka daje motivaciju da se istraži mogućnost automatizacije dizajna skladišta podataka, odnosno automatskog generisanja i prepoznavanja mjera i dimenzija na osnovu dizajna i fizičkog modela transakcione baze podataka uz minimalnu interakciju sa korisnikom kroz korisnički interfejs (engl. *user interface*). Osim toga, ovakvo istraživanje otvara perspektive za istraživanje automatizacije ETL procesa u dijelu prikupljanja, obrade i učitavanja internih podataka iz transakcione baze, te posebno podršci fizičkog projektovanja DW/DM, pitanju particija po vremenu i/ili lokaciji kao i distribuciji kopija i materijalizovanim agregacijama za data martove.

Centralni **problem istraživanja** odnosi se, dakle na mogućnost automatizacije projektovanja skladišta podataka na osnovu fizičkog modela transakcione relacione baze podataka. Drugim riječima, ispitaće se mogućnost automatskog kreiranje tabela mjera i dimenzija, odnosno hub, link i satelit entiteta u Data Vault konceptu. Plan istraživanja se može vidjeti u pristupnom radu [KR12].

Lična motivacija za rad na pomenutom istraživanju proizilazi iz dugogodišnjeg rada na razvoju, implementaciji i održavanju poslovnih informacionih sistema i sistema poslovne inteligencije, te iz problema otkrivanja znanja o poslovnim procesima i potrebama korisnika za informacijama. Naime, prilikom komunikacije sa korisnicima projektanti pokušavaju da formiraju generalizacije vezane za aplikaciju koje za korisnika djeluju složeno i apstraktno. Korisnici su zbog toga postaju nezainteresovani za učešće u fazi projektovanja skladišta podataka, te se njihovi zahtjevi u potpunosti ne artikulišu u kontaktu sa projektantima skladišta podataka. Tek kad je aplikacija ili prototip aplikacije gotov, gdje korisnik može da prepozna svoje buduće radno okruženje, korisnik postaje više zainteresovan za razvoj aplikacije. Poslije toga se obično pojavljuju novi zahtjevi korisnika. Automatizacijom dizajnirajnja skladišta podataka znatno bi se ubrzao razvoj kompletnog sistema koji bi omogućio izradu prototipova aplikacije u ranoj fazi kontakta sa korisnicima, te bi dobili korisnike koji su više zainteresovani za davanje informacija. Projektanti bi na taj način lakše otkrivali znanja o poslovnim procesima i informacijama, te potrebe korisnika za informacijama.

1.2. Ciljevi istraživanja

Osnovni cilj disertacije je da se osnovu šire literature sagleda problem neautomatizovane izrade dizajna skladišta podataka, te istraži mogućnost formalizacije i automatizacije projektovanja skladišta podataka baziranog na Data Vault konceptu uz minimalnu interakciju sa korisnikom.

Ostali ciljevi doktorske disertacije su:

- ispitati mogućnost formalizacije i parametarske automatizacije ETL procesa u dijelu prikupljanja, obrade i učitavanja internih podataka iz transakcione baze,
- istražiti mogućnost uticaja formalizacije i automatizacije dizajna skladišta podataka na brzinu i kvalitet dizajna kao i dobijanja informacija o poslovnim procesima i potrebama korisnika za informacijama,
- demonstrirati skalabilnost pristupa i mogućnosti smanjenja troškova i vremena implementacije skladišta podataka,
- demonstrirati i validirati upotrebnu vrijednost automatizacije projektovanja skladišta podataka za profesionalnu praksu u domenu zdravstva.

1.3. Hipoteze u istraživanju

U radu se postavlja sledeća **generalna hipoteza** koja obuhvata preliminarno i teoretsko određenje predmeta istraživanja:

Hipoteza 1: Proces projektovanja skladišta podataka baziranog na Data Vault konceptu i izrade dimenzionog modela skladišta podataka može se formalizovati, generalizovati i u određenoj mjeri automatizovati na osnovu fizičkog modela struktuiranih, polustruktuiranih i jednostavnijih nestruktuiranih izvora podataka, uključujući i transakcione baze podataka.

Iz generalne hipoteze izvodi se sledeće **posebne hipoteza** koja obrađuje dijelove predmeta istraživanja:

Hipoteza 2: Formalizacijom, standardizacijom i automatizacijom projektovanja skladišta podataka, podiže se kvalitet DW sistema, te smanjuju vrijeme i troškovi implementacije skladišta podataka.

Hipoteza 3: Automatizacija projektovanja skladišta podataka doprinosi automatizaciji ETL procesa koji se odnosi na interne podatke iz transakcione baze podataka.

1.4. Metode istraživanja

Osnovni metod istraživanja prilikom izrade doktorske disertacije bio je sakupljanje i proučavanje dostupne literature, njena analiza i sistematizacija, a sve to s ciljem da se pokaže opravdanost i korisnost razvoja novog modela automatizacije projektovanja skladišta podataka. U izradi disertacije od opštih naučnih metoda, korišćena je metoda modelovanja i analitičko-deduktivna metoda. Metoda modelovanja je korišćena prilikom izrade modela metapodataka izvora i skladišta podataka, modela skladišta podataka i praktične provjere rezultata istraživanja. Kod analize podataka o postojećim rješenjima u oblasti skladišta podataka korištena je analitičko-deduktivna metoda. Pored navedenih opštih metoda, u dijelu izrade modela metapodataka izvora podataka, korišteni su i posebni metodološki pristupi kao što su generalizacija (apstrakcija u kojoj se skup sličnih tipova objekata tretira kao generički tip) i specijalizacija kao inverzni postupak (u kojem se za neki tip objekta definišu njegovi podtipovi sa specifičnim osobinama).

Istraživanje je provedeno u nekoliko faza. Prva faza je bila sakupljanje i proučavanje dostupne literature u oblasti projektovanja skladišta podataka, njena analiza i sistematizacija, sa akcentom na automatizaciju projektovanja skladišta podataka. Identifikovani su i kategorizovani postojeći pristupi. Sledeća faza u istraživanju je bila proučavanje Data Vault koncepta, analiza njegovih prednosti i nedostataka. Centralni dio istraživanja se odnosio na formalizaciju proširenja Data Valuta u cilju principijelnog obuhvata izvora

podataka iz izvora koji nisu relacione baze podataka i automatizaciju projektovanja proširenog Data Vaulta pristupa. U zadnjoj fazi istraživanja implementiran je prototipski alat za automatizaciju projektovanja skladišta podataka, te demonstrirana upotrebna vrijednost rezultata istraživanja na praktičnom primjeru u oblasti zdravstva.

1.5. Sadržaj disertacije

Pored uvodnog dijela, u kome je dat problem, motivacija i predmet istraživanja, ciljevi, metode i hipoteze istraživanja i sadržaj disertacije, disertacija je organizovana u dodatnih sedam dijelova od kojih svaki dio ima pripadajuća poglavlja.

U drugom dijelu disertacije razmatraju se osnovni aspekti skladišta podataka. U ovom dijelu se daje nekoliko definicija skladišta podataka, sadržaj i ciljevi izgradnje skladišta podataka, te odnos poslovne inteligencije i skladišta podataka i njihov značaj za podršku odlučivanju. Dalje se razmatraju osnovni elementi multidimenzionog dizajna, ETL procesa i načina prezentacija podataka korisniku informacija. U drugom dijelu je dat pregled i karakteristike nekoliko arhitektura skladišta podataka i osnovni aspekti kao i teorijska zasnovanost Data Vault koncepta skladištenja podataka. Osim toga, u ovom dijelu je dat i kratak osvrt na kvalitet skladišta podataka.

Najnovija istraživanja u oblasti dizajna skladišta podataka razmatraju se u trećem dijelu disertacije. Za pronalaženje literature, u ovom dijelu su korišteni najpoznatiji pretraživači naučnih radova sa konferencija i časopisa kao i sugestije projektanata i istraživača koji duže rade u DW oblasti radi kompletiranja slike i ekspertne ocjene relevantnosti smjernica prije detaljnog

čitanja i ličnog istraživanja. U ovom dijelu rada se daje kritički osvrt na dosadašnje rezultate istraživanja, identifikuje se i analizira problem automatizacije dizajna skladišta podataka, te se daje prijedlog istraživanja da bi se došlo do rješenja.

U četvrtom, u centralnom dijelu rada, se razvija direktan pristup projektovanja fizičkog modela skladišta podataka baziranog na Data Vault konceptu, te se definiše formalizacija, generalizacija i standardizacija proširenog Data Vault pristupa, pri čemu se za svaku vrstu izvora podataka standardizuje model i transformacije uz korišćenje metamodela i odgovarajućih metapodataka.

U petom dijelu rada opisan je način realizacije prototipa alata *Direct PDV* koji služi za automatizaciju projektovanja skladišta podataka baziranog na Data Vault konceptu. Alat se realizuje uz korišćenje (u kontekstu industrijski veoma relevantnog) konkretnog sistema za upravljanje bazama podataka Microsoft SQL Server 2008 R2. Postoji nekoliko razloga za odabir ovog sistema kao što su: brojnost implementacija, raspoloživost uključujući cijenu, te intenzivan razvoj (sada već verzija 2014). Tehnički odlična alternativa Oracle bila bi od većeg značaja samo za najveće i finansijski najsnažnije korisnike.

U šestom dijelu se pokazuje praktična validacija pristupa i provjera rezultata istraživanja, te razmatra upotrebna vrijednost automatizacije dizajna skladišta podataka i praktična provjera realizovanog alata *Direct PDV* u oblasti zdravstvenog osiguranja.

Zaključak, osvrt na naučne doprinose i pravce daljih istraživanja su dati u sedmom poglavlju. Na kraju rada je data korišćena literatura.

2. OSNOVNI ASPEKTI SKLADIŠTA PODATAKA

Skladište podataka je podloga za obradu podataka. To je predmetno orijentisana, integrisana, stalna i vremenski promjenjiva kolekcija podataka za podršku menadžmentu kod odlučivanja [Inm92]. Ovo je vjerovatno najstarija definicija skladišta podataka koju je dao Bill Inmon i koja još uvijek odražava suštinu skladišta podataka. Prema [Inm92], predmetna orijentisanost znači da se podaci organizuju oko predmeta i daju informacije o određenim predmetima u okviru područja kompanije; integrisana znači da se podaci skupljaju u bazu podataka iz različitih izvora i da su spojeni u koherentnu cjelinu; stalna znači da su podaci pohranjeni u skladištu stabilni i kad se jednom snime u skladište, u pravilu se ne mijenjaju; vremenska promjenljivost znači da su svi podaci u skladištu podataka identifikovani u određenom vremenskom razdoblju. Postoji još definicija skladišta podataka od kojih izdvajamo:

- Skladište podataka se definiše kao kopija transakcionih podataka specijalno struktuiranih za upite i analizu [Kim96];
- Skladište podataka predstavlja centralni izvor podataka koji su prethodno pročišćeni i transformisani, kako bi služili za potrebe podrške poslovnom odlučivanju [OBr06];
- Skladište ili stovarište podataka sadrži podatke koji su na pogodan način izvučeni iz različitih izvora podataka. Izvori podataka mogu biti relacione ili objektne baze podataka, mrežne baze i podaci u različitim datotekama [LMAB08].

Osnovna namjena skladišta podataka je analiza podataka i podrška odlučivanju, te u novije vrijeme podrška zakonskim zahtjevima u pogledu očuvanja podataka, te usklađenosti i poboljšanja upravljačkih podataka (engl. *compliance and improved data governance*) kao i podrška trajnom sistemu evidencije (engl. *persistent system of records*). Za skladište podataka postoje dva

izvora podataka, interni i eksterni podaci. Interni podaci pripadaju samoj organizaciji i generišu se najčešće putem transakcionog sistema (OLTP – *On Line Transaction Processing*). Eksterni podaci se nalaze izvan organizacije. Najvažniji cilj skladišta podataka jeste integracija eksternih i internih podataka. Kako eksterni, tako i interni podaci mogu da ne pripadaju bazama podataka. Podaci koji se nalaze u bazama podataka imaju jasnu strukturu sa tabelama, kolonama, indeksima i dr. Podaci koji se ne nalaze u bazama podataka nego u fajl sistemima odlikuju se nestruktuiranošću.

Data mart je oblik skladišta podataka koji sadrži sumirane podatke na određenim organizacionim dijelovima kompanije ili po određenoj grupi poslovnih procesa [KRR08]. Data mart je mjesto gdje krajnji korisnik ima direktan pristup i kontrolu njegovih analitičkih podataka. Finansije imaju svoje podatke, marketing i prodaja svoj data mart, itd. Svaki data mart obično drži znatno manje podataka nego skladište podataka [Inm02].

Skladište podataka nije proizvod ili grupa proizvoda, koje kompanije mogu kupiti kako bi se dobili odgovore na pitanja i poboljšali donošenja odluka. Skladište podataka može pomoći kompanijama da dobiju odgovore na pitanja radi podrške odlučivanju [IBM98]. Dok se u sistemima sa upravljanje skladištima podataka fizički formira integrisana baza podataka, medijatori su softverske komponente koje virtuelno integrišu podatke iz različitih izvora. Ponegdje se, naročito u oblasti softverskog inženjerstva, termin medijator zamenjuje sa terminom softverski posrednik. Softverski posrednik je komponenta čiji je cilj da uskladi razlike između softverskih sistema [LMAB08].

2.1. Multidimenzioni dizajn

Konceptualni dizajn predstavlja korisnički orijentisanu reprezentaciju baze podataka koji ne razmatra implementaciju. Kao rezultat ovog dizajna nastaje

konceptualni model koji sadrži informaciju o entitetima, njihovim atributima i vezama. Model se definiše kao skup iskaza o nekom sistemu [Rio05]. Okvir za razvoj softvera u kome centralno mjesto zauzima model na visokom nivou apstrakcije naziva se MDA (engl. *Model Driven Architecture* – model-bazirana arhitektura). Ova arhitektura ne zavisi od konkretne implementacione platforme (engl. *Platform Indepedent Model* – PIM). PIM se sastoji od jednog ili više vrsta dijagrama koji opisuju statičke i dinamičke osobine sistema koji se modeluje. MDA pokret je nastao pod vođstvom OMG (*Object Management Group*) grupe [OMG]. MDA pristup se koristi i kod razvoja skladišta podataka. Za izradu modela koriste se CASE (*Computer Aided Software Engineering*) alati. CASE proizvod je bilo koji programski proizvod, namenjen za podršku, ili automatizaciju makar jednog zadatka u okviru životnog ciklusa drugog programskog proizvoda [MLG04]. CASE alati koji su danas najviše u upotrebi za DB i/ili DW su: Sybase Power Designer, IBM Rational Rose, ERwin (Entity Relationship for WINdows), ORACLE Designer 2000, Microsoft Office Visio 2003, Oracle SQL Data Modeler, i dr.

Model entitet-veze [Chen76] ili neka od njegovih brojnih varijanti je jedan od najčešće korištenih konceptualnih modela za projektovanje baza podataka. Konceptualni dizajn se u principu, može izvesti pomoću dva različita pristupa, prema složenosti sistema, raspoloživim podacima i iskustva projektanta [MZ08]. Kao primarni pristup otkrivanju znanja u procesu modelovanja podataka identifikovane su dvije različite kategorije tehnika [JGC07]:

– Identifikacione tehnike. Termin identifikacija pokazuje da izvana vidljivi dokazi služe kao osnova za analizu. Ove tehnike su usmjerene na parsiranje i restrukturiranje uzoraka i definicije iz skupa podataka, uključujući i interne sheme, u terminologiji tri nivoa ANSI / SPARC arhitekture.
– Direktne tehnike modelovanja. Podrazumijevaju poznavanje cjelokupnog

sistema, a ne samo sadržaja podataka. Data mining je ovdje osnova za modelovanje podataka.

Dva generalna pristupa u akademskoj zajednici takođe se spominju kao bottom-up i top-down pristup:

- Top-down dizajn [Inm02]: zahtjevi različitih korisnika su spojeni prije nego proces dizajna počinje, a jedinstvena šema je izgrađena. Ovaj pristup može biti težak i skup za velike baze podataka i neiskusne projektante.
- Bottom-up dizajn [KR02]: odvojena šema je izgrađena za svaku grupu korisnike s različitim zahtjevima, da bi se kasnije tokom integracije te šema spojile u globalnu konceptualnu šemu za cijelu bazu podataka. Ovaj pristup se obično koristi za velike baze podataka.

Logički dizajn ima za cilj prevođenja konceptualnog prikaza baze podataka dobijenog u prethodnoj fazi u određenu primjenu modela u više DBMS (engl. *Database Management System*). Najčešći logički modeli su relacioni model i objektno-relacioni model. Fizička dizajn, čiji rezultat je fizički model podataka, koji je na najnižem nivou apstrakcije u odnosu logički i konceptualni model, ima za cilj prilagođenje logičkog prikaza baze podataka dobijenih u prethodnoj fazi za primjenu kod pojedinih DBMS platformi kao što su SQL Server, Oracle, DB2, MySQL i drugi. Prema [JGC07], najpoznatije konvencije za modelovanje su: EXPRESS (ISO, 1994), IDEF1X (IEEE, 1998), RM-OMP (ISO / IEC, 2000) i UML (ISO / IEC, 2004) koji su formalno izglasani i označeni kao standardi.

Tradicionalni OLTP sistemi su optimizovani za pružanje visokih performansi u obradi mnogo istovremenih transakcija [KR02]. Kod OLTP sistema se izrađuje normalizovan model podataka (najmanje u trećoj normalnoj formi – 3NF). Zbog ovakvog modela podataka, usložnjava se način pristupa podacima kod

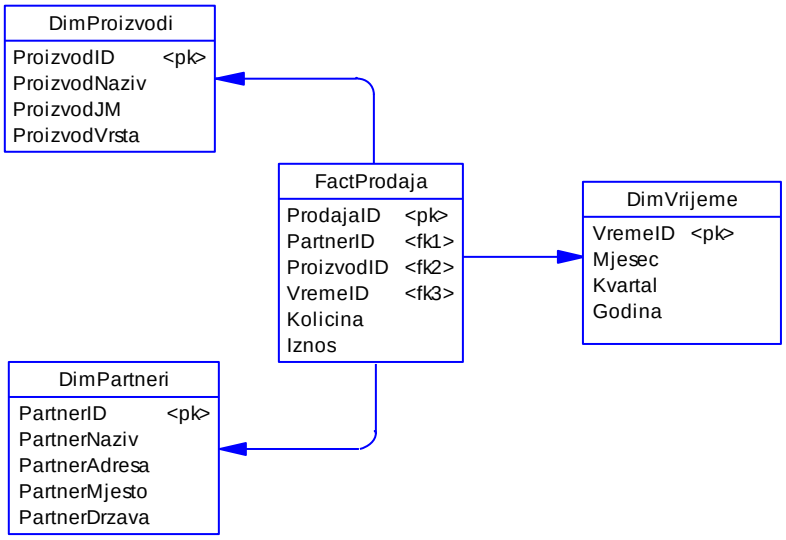
različitih upita nad podacima [KRR08]. Za razliku od normalizovanih baza podataka koje su projektovane za često dodavanje i ažuriranje podataka, dimenzione baze su projektovane za obradu ad-hoc upita i rijetko se ažuriraju nakon prenošenja u dimenzionu bazu [Rio05]. Kod dimenzionih baza podataka karakteristična šema je da su relacije podeljene u dvije kategorije: tabelu činjenica i tabele dimenzija. Izraz činjenice se koriste da predstave poslovnu mjeru. Dimenzije sadrže tekstualne opise poslovanja [KR02]. Aplikacije napravljene primjenom dimenzionog modela često se nazivaju OLAP (*On-Line Analytical Processing*) aplikacije. OLAP podrazumijeva kategoriju aplikacija i tehnologije namijenjenu za skupljanje, upravljanje, obradu i prezentaciju multidimenzionih podataka namijenjenih analizama za potrebe upravljanja [Cod93]. OLAP je prirodni produžetak skladišta podataka. Rezultati iz OLAP analize mogu biti vizualno predstavljeni, što omogućava poboljšano razumijevanja [ĆBVG]

Tabela 1. Poređenje OLTP-a i OLAP-a [KRR08] [Rio05] [KR02]

Karakteristike	OLTP	OLAP
Struktura podataka	normalizovana	dimenziona
Složenost upita	jednostavni upiti	kompleksni upiti
Namjena	transakcioni podaci	analiza i prognoza
Tipične operacije	dodavanje, izmjena, brisanje	analiza
Nivo podataka	detaljni podaci	sumarni podaci
Korisnici	većina zaposlenih	analitičari i menadžeri
Pristup podacima	čitanje i pisanje	uglavnom čitanje

Pojam OLAP prvi uveo EF Codd 1993. god. radi lakše analize informacija vezanih za proizvodnju, te radi sinteze sa podacima iz drugih dijelova kompanije. Osnovne razlike između OLTP-a i OLAP-a date su u tabeli 1.

Data mart je obično dizajniran u tzv. zvijezda shemi (engl. *star schema*) [Inm00a] [Inm92][Kim96][KR02] koja se sastoji od jedne tabele činjenicu i više tabela dimenzija koje su svi povezani sa tabelom činjenica i nisu povezani s drugom tabelom dimenzija. U nekim situacijama je korisno razdvojiti tabelu dimenziju na više tabela dimenzija, kada dobijamo pahuljastu šemu (engl. *snowflake schema*) [Inm00a][Inm92][Kim96][KR02]. U ovoj shemi podaci su normalizovani, ali je nešto teži pristup podacima. Na slici 1. je prikazana tzv. zvjezdasta šema koja prikazuje dio dimenzione baze podataka definisan pomoću jedne tabele činjenica i više tabela dimenzija koja može da se koristi u realizacija skladišta podataka za trgovačku kompaniju.

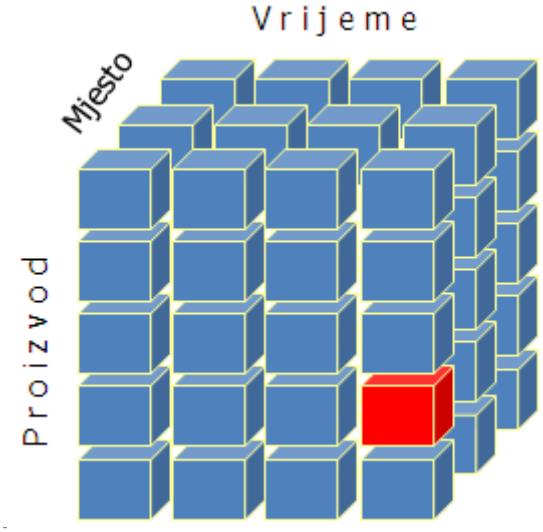


Slika 1. Zvjezdasta šema

Tabele dimenzija su: poslovni partneri, proizvodi i vrijeme. Tabela činjenica sadrži podatke o prodajama po partnerima i vremenu po proizvodima, a to su činjenice o npr. količini prodatih proizvoda i novčanim iznosima. Druga grupa atributa tabele su spoljni ključevi kojim je tabela činjenica povezana sa tabelama dimenzija. U prikazanom primjeru pahuljastu šemu bi realizovali u slučaju kada bi pored tabele DimPartneri kreirali tabelu mjesta ili država koja bi

bila sa njom povezana, gdje bi ispoštovali pravila normalizacije podataka, ali bi neznatno smanjili brzinu pristupa podacima.

Podaci iz dimenzione baze mogu zamisliti u obliku kocke (engl. *cube*). Ova kocka se generiše iz skladišta podataka i predstavlja multidemnzionu strukturu. To je struktura podataka koja mjere grupiše po hijerarhiji svake dimenzije koju korisnik želi da analizira.



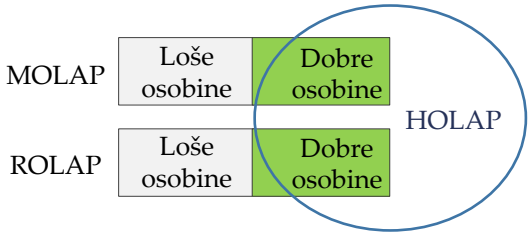
Slika 2. Struktura podataka u obliku kocke

U primjeru na slici 2. je prikazana kombinacija dimenzija: vrijeme, proizvod i geografsko područje sa mjerama količina i iznos prodatih proizvoda. Iako u strogom matematičkom smislu stranice kocke nisu jednake, u ovom slučaju se radi o pogodnoj metafori za složeni prikaza podataka za analizu. Podaci u skladištu podataka moraju biti odabrani, transformisani i učitani da bi se zadovoljili zahtjevi pojedinog data marta. Kod kreiranja kocki mnogo različitih izračunavanja mora biti izvedeno nad detaljnim podacima koji se nalaze u skladištima podataka. Proces prenosa podataka iz normalizovanog u višedimenzionalan svijet nije nimalo jednostavan [Inm02]. OLAP sistemi imaju poseban jezik za upite MDX (*Multidimensional Expressions*) koji je zasnovan na

SQL (engl. *Structured Query Language)* jeziku i koji ima mogućnost čitanja podataka iz multidimenzione kocke. MDX se izvršava u pozadini kad korisnik obavlja jednostavne operacije nad podacima "povuci i pusti" (engl. *drag & drop*). Prema načinu spremanja podataka, OLAP sistemi se mogu podijeliti na [Pon10]:

- ROLAP (*Relatinal OLAP*) sistemi imaju relacioni model baze podataka,
- DOLAP (*Desktop OLAP*) se odnosi na desktop analitičku obradu i predstavlja varijaciju ROLAP-a,
- MOLAP (*Multidimenzinal OLAP*) se odnosi na višedimenzinalnu analitičku obradu i na sisteme koji imaju multidimentioni dizajn,
- HOLAP (*Hybrid OLAP*) se odnosi na hibridnu analitičku obradu.

ROLAP sistemi imaju malu brzinu rada kod postavljenih upita, ali zauzimaju najmanje prostora u bazi podataka. MOLAP sistemi se sa druge strane odlikuju velikom brzinom rada, ali sa ograničenom količinom podataka koju mogu spremiti. HOLAP kombinuje najbolje osobine ROLAP-a i MOLAP-a.



Slika 3. Hybrid OLAP

2.2. ETL proces

Najveći dio podataka u skladište podataka dolazi iz OLTP sistema. Da bi podaci bili prebačeni iz transakcionih sistema u skladište podataka potrebno je da prođu određeni proces transformacije. Pored poslovnih podataka, vrlo važan element skladišta podataka su i metapodaci. Prema [AMA05], metapodaci su kontekstualni podaci o podacima, uključujući poslovne definicije, pravila za stvaranje podataka i formate podataka. Metapodaci se

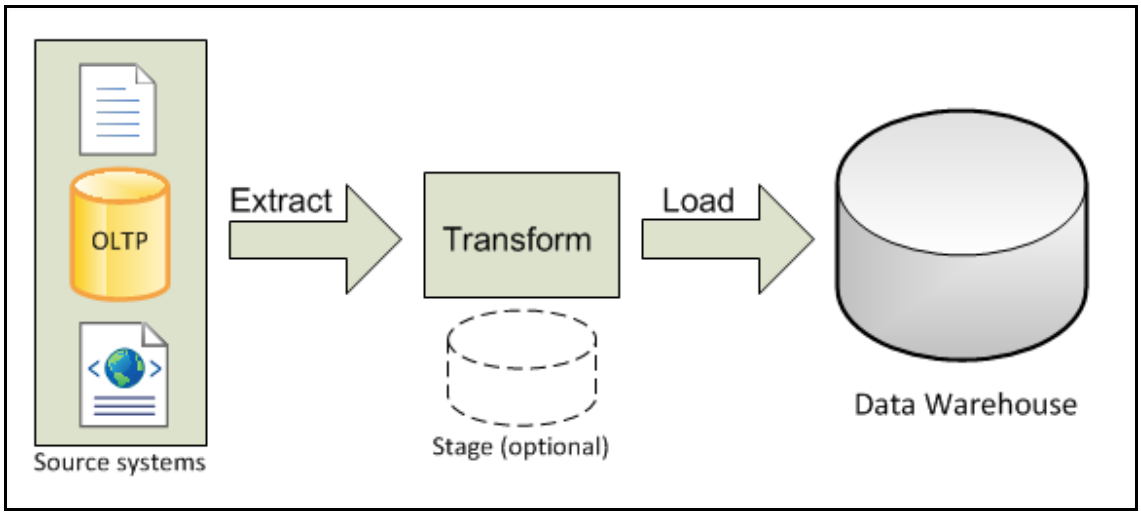
dijele na tehničke i poslovne metapodatke [MA02]. Poslovni metapodaci opisuje značenje (ili semantika) podataka i organizacionih pravila i ograničenja u vezi s podacima. S druge strane, tehnički metapodaci opisuju kako su struktuirani podaci i kako se snimaju u sistem, te aplikacije i procese koji manipulišu sa podacima [MZ08].

ETL (*Extract, Transform, Load*) proces podrazumijeva dohvatanje podataka iz jednog ili više OLTP sistema, čišćenje podataka i transformaciju u odgovarajući format i učitavanje podataka u skladište podataka [Lar09]. ETL proces reorganizuje podatke iz aplikativno orijentisane strukture u korporativnu strukturu podataka [ISN08]. ETL je skup postupaka kojima se operativni izvori podataka spremaju za skladište podataka [KR02]. Saglasno prethodnim definicijama, ETL proces se sastoji od tri faze:

1. Dohvatanje (izvlačenje) podataka iz internih i eksternih izvora podataka
2. Transformacija podataka u odgovarajući format koji je pogodan za učitavanje u skladište podataka
3. Učitavanje podataka u skladište podataka

Izvlačenje podataka je prva faza u ETL procesu. U ovoj vazi se iz više, obično heterogenih, izvora izvlače potrebni podaci. Podaci mogu biti u bazama podataka ili datotekama različitog formata. U ovoj fazi je potrebno riješiti problem interoperabilnosti između različitih sistema. Interoperabilnost podrazumijeva sposobnost informacionih i komunikacionih sistema i poslovnih procesa da podržavaju razmjenu podataka i omogućuju dijeljenje informacija i znanja [IDABC]. Procese u ovoj fazi je treba uraditi na način da što manje remete rad transakcione baze. Osim toga, potrebno je izdvojiti samo one podatke koji će se koristiti u skladištima podataka. U postupku transformacije

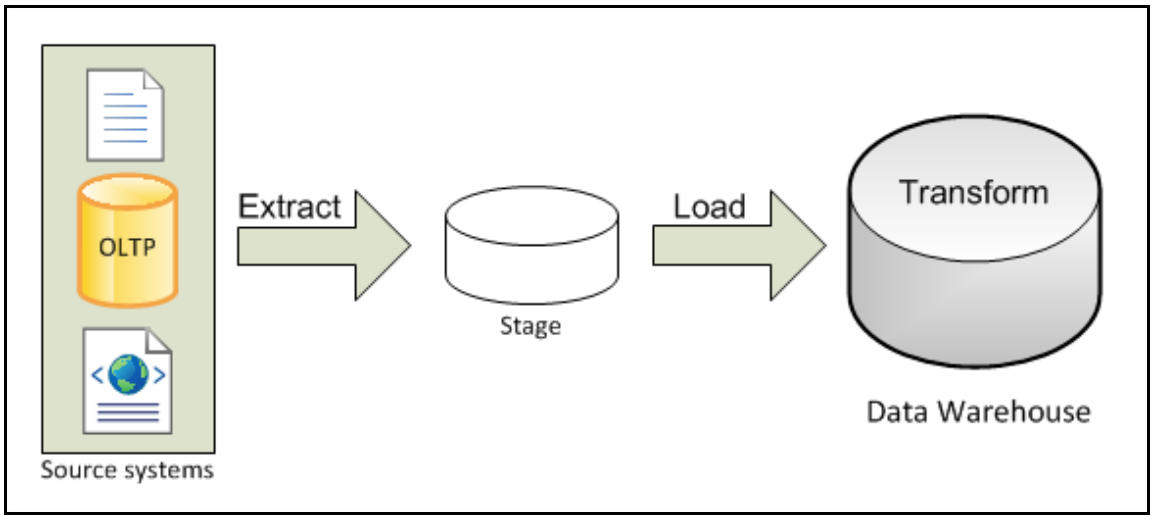
pojavljuju se problemi sa netačno unesenim podacima, nekonzistentnim vrijednostima podataka, različitim formatima podataka, pojavljivanja istih podataka sa različitim nazivima i sl. U većini slučajeva, oko 75% ETL procesa obuhvata transformacija podataka, dok je ostalih 25% obuhvata izvlačenje (ekstrakcija) i učitavanje podataka [KOBB10].



Slika 4. ETL proces

U ETL procesu mogu se kreirati prelazne (engl. *staging*) tabele ili baze podataka. To su tabele koje se kreiraju tokom transformacije podataka i u njima se čuvaju podaci u prelaznom obliku koji je adekvatan za dimenzione tabele u skladištu podataka. Drugi je pristup da se transformacija radi u memoriji, a zatim se podaci učitavaju u skladište podataka. Pristup sa prelaznim tabelama uobičajeno se primjenjuje kod velike količine podataka, dok se one ne koriste kod relativno malog broja podataka.

Postoji i alternativa ETL procesima koji su prikazani na slici 5, a to je ELT (*Extract, Load, Transform*) proces. U ovom slučaju se podaci nakon ekstrakcije najprije učitavaju u skladište podataka, a zatim se radi njihova transformacija.



Slika 5. ELT proces

ELT pristup se koristi ako imamo velika skladišta podataka, uobičajeno za MPP sisteme. Massively Parallel Processing (MPP) je grupa servera od kojih svaki imaju vlastitu memoriju, procesor i disk [Rai08]. U zadnjoj fazi ETL procesa, koja podrazumijeva učitavanje podataka, odvijaju se aktivnosti inicijalnog učitavanja podataka i tekućeg učitavanje podataka. Kod inicijalnog učitavanja podataka učitavaju se tekući i istorijski podaci iz transakcionih sistema i vanjskih izvora podataka. Nakon što su učitani svi istorijski podaci, učitavaju se tekući podaci koji će se učitavati periodično aktiviranjem odgovarajućih mehanizama u alatima koji su namijenjeni za ETL proces.

2.3. Prezentacija podataka korisniku informacija

Nakon učitavanja podataka u skladište podataka, potrebno je podatke u odgovarajućem formatu prikazati korisniku. Za prezentaciju podataka korisniku koriste se klijentski alati koji mogu biti:

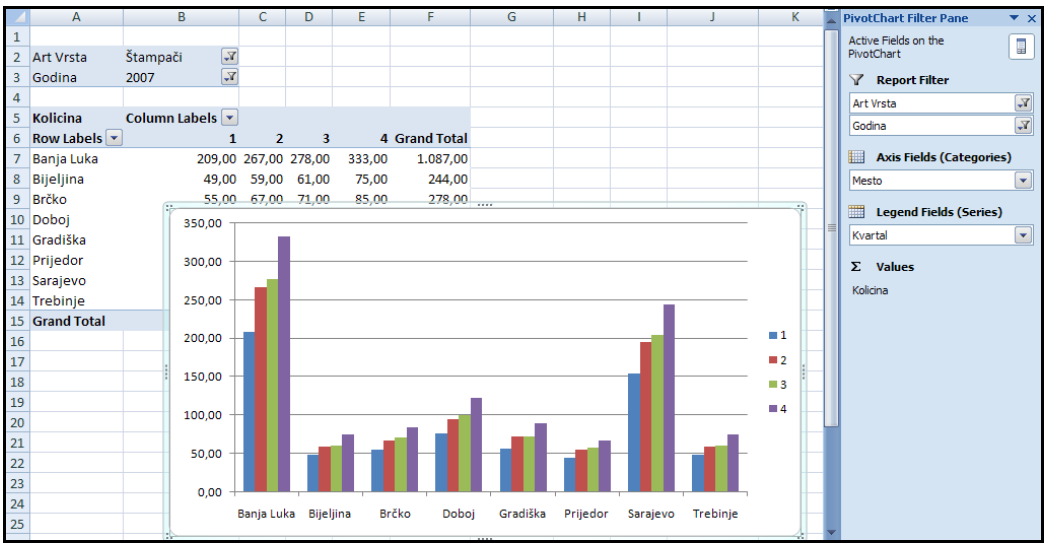
1. Alati za izvještavanje (engl. *Reporting tools*), koji koriste unaprijed poznate upite i omogućuju upravljačkim strukturama Web bazirane ili papir bazirane izvještaje koji mogu biti odštampani na štampaču. Na ovaj način se rade izvještaji za koje korisnik zna ili pretpostavlja da će mu biti potrebni. Za ove

izvještaje uglavnom se koriste alati koji su korišteni za izradu izvještaja transakcionog sistema.

2. OLAP alati, koji omogućuju kompleksne upite nad velikom količinom podataka, manipulaciju podacima i grafički prikaz podataka. OLAP alati omogućuju: biranje dimenzija, rotaciju dimenzija, drill-down (posmatranje iste mjere na nižem nivou hijerarhije, odnosno agregacije) i drill-up (posmatranje iste mjere na višem nivou hijerarhije). Podaci kroz OLAP alate se mogu istovremeno posmatrati kroz veći broj dimenzija. Ovi alati omogućuju brze ad hoc analize, u odnosu na papirne dvodimenzionalne izvještaje. Najpoznatiji OLAP alat je svakako Microsoft Excel.

3. Statistički alati koji koriste statističke metode za analizu podataka. Najpoznatiji statistički alati koji se koriste su SAS i SPSS.

4. Alati za rudarenje podataka (engl. *data mining*). Data mining ima ulogu da podrži donosioca odluka u potrazi za zakonitostima u podacima organizacije. Otkrivanje zakonitostima u podacima je slično klasičnoj statističkoj analizi pri čemu se proširuje spektrom algoritama i tehnika i to od multivariocione analize do mašinskog učenja [SD10].



Slika 6. Prezentacija podataka iz skladišta podataka pomoću MS Excel-a

2.4. Poslovna inteligencija i skladište podataka

Često se pojam skladišta podataka vezuje za pojam Poslovna Inteligencija. Ovaj pojam je prvi upotrebio 1989. godine istraživač Gartner grupe Howard Dresner da bi opisao set koncepata i metoda za unapređenje procesa odlučivanja baziranog na podacima. Do tada su donosioci odluka u kompanijama morali ručno prikupljati informacije o poslovanju. Zbog nedostataka pravih informacija, poslovne odluke su se donosile najčešće na osnovu intuicije menadžera. Postoji mnogo definicija poslovne inteligencije od kojih izdvajamo:

- Poslovna inteligencija je generički termin za opisivanje uticaja unutrašnjih i vanjskih informacija organizacije za donošenje boljih poslovnih odluka [KR02];
- Poslovna inteligencija je skup informacionih tehnologija, organizacionih pravila kao i znanja i vještina zaposlenih u organizaciji udruženih u generisanju, zapisivanju, integraciji i analizi podataka sve sa ciljem da se dođe do potrebnog znanja za donošenje odluke [SD10];
- Poslovna inteligencija podrazumijeva isporuku tačnih i korisnih poslovnih informacija za donošenje odgovarajućih poslovnih odluka [Lar09];
- Poslovna inteligencija je sticanje i iskorištavanje znanja baziranog na činjenicama u svrhu unapređivanja strateške i taktičke poslovne prednosti na tržištu [Bar97].

Sistem poslovne inteligencije predstavlja osnovnu tehnološku strukturu koja omogućuje poslovnim subjektima potpuno upravljanje informacijama pružajući im mogućnost boljeg pozicioniranja na tržištu [KR08]. Sistem poslovne inteligencije se ne isporučuje kao gotov proizvod za određene kompanije ili

određenu djelatnost. Uvođenje sistema poslovne inteligencije zahtjeva značajnu podršku unutar kompanije. Najčešće se u projekti poslovne inteligencije i izgradnje skladišta podataka posmatraju kao jedan projekt, jer skladište podataka predstavlja podlogu za korišćenje alata poslovne inteligencije.

2.5. Arhitekture skladišta podataka

IT arhitektura sistema je idejno rješenje koji opisuje strukturu i ponašanje IT sistema. Arhitektura skladišta podataka predstavlja formalni opis sistema skladišta podataka [Hua10]. Arhitektura skladišta podataka podrazumijeva definicije osnovnih elemenata skladišta podataka i opisuje kako su ti elementi izgrađeni i povezani. Osnovni elementi skladišta podataka su:

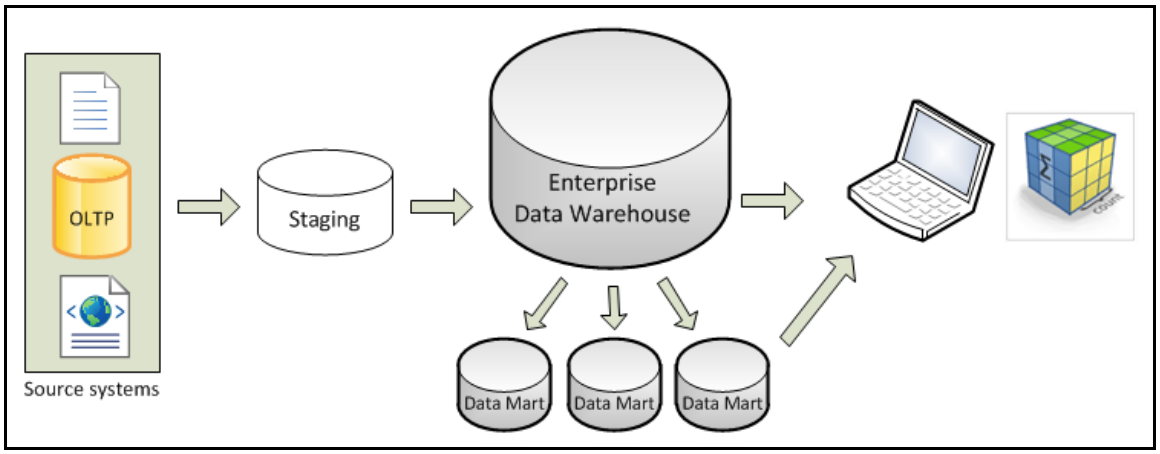
1. Izvor podataka (engl. *source data*) obuhvata interne podatke, koji se uglavnom generišu iz transakcionog sistema, i eksterne podatke koji se generišu izvan organizacije.
2. Prelazne (engl. *staging*) tabele ili baze podataka, koje se kreiraju tokom transformacije podataka i u njima se čuvaju podaci u prelaznom obliku koji je pogodan za dimenzione tabele u skladištu podataka. Pristup sa prelaznim tabelama uobičajeno se primjenjuje kod velike količine podataka, dok se one ne koriste kod relativno malog broja podataka.
3. ETL/ELT proces obuhvata dohvatanje podataka transakcionih sistema, transformisanje podataka u odgovarajući format i učitavanje podataka u skladište podataka.
4. Enterprise data warehouse podrazumijeva skladištenje podataka sa integrisanim podacima cjelokupnog preduzeća ili organizacije gdje su podaci pohranjeni i gdje je omogućen pristup podacima preko upita, izvještaja ili analitičkih aplikacija.

5. Data mart, podrazumijeva način skladištenja podataka koji sadrži sumirane podatke na određenim organizacionim dijelovima ili po određenoj grupi poslovnih procesa.
6. Alati za pristup podacima, koji korisnicima omogućuje različite poglede na podatke.

Postoji nekoliko tipova skladišta podataka koji se najviše koriste u praksi koja sadrže sve ili dio pomenutih komponenti. Prema [Inm02], postoje četiri nivoa u arhitekturi podataka:

- Operacioni nivo, sadrži primitivne podatke koji su aplikativno orijentisani.
- Skladište podataka, sadrži istorijske integrisane primitivne podatke koji se ne ažuriraju, kao i neke izvedene (engl. *derived*) podatke.
- Data mart-ovi, sadrže samo izvedene podatke i implementiraju se u skladu sa potrebama organizacionog dijela.
- Individualni nivo podataka uključuje razne heurističke analize

Ovakva arhitektura predstavlja **CIF (Corporate Information Factor) arhitekturu** koju propagira Bill Inmon i koja proizlazi iz usvajanja top-down pristupa razvoja skladište podataka. Kod ove arhitekture podaci se iz transakcionih sistema transformišu format pogodan za učitavanje u skladište podataka. Skladište podataka sadrži atomske podatke u trećoj normalnoj formi i ono predstavlja izvor za data mart-ove. Data mart-ovi najčešće imaju dimenzionu strukturu u obliku zvijezde ili pahulje. Korisnici na individualnom nivou dobijaju podatke koje prilagođavaju sopstvenom okruženju radi podrške odlučivanju.



Slika 7. Hub-and-Spoke arhitektura

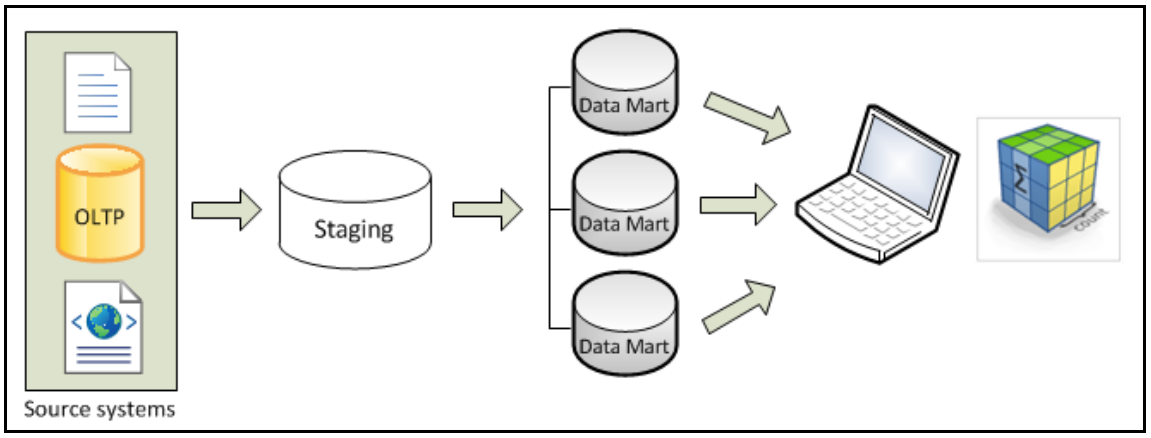
Ova arhitektura se još naziva Hub-and-Spoke, jer centralno skladište podataka predstavlja središte ili čvorište (engl. *hub*) za isporuku podataka pojedinim data mart-ovima preko veze (engl. *spoke*) sa njima.

Prema [KR02], skladište podataka treba da bude okrenuto poslovnim procesima a ne organizacionim dijelovima kompanije. Komponente skladišta podataka su:

- Izvorni operacioni sistemi, koji sadrže atomske podatke iz OLTP sistema
- Zona pripreme podatka, podrazumijeva komponentu u kojoj se vrše ETL procesi. Ovom dijelu ne pristupaju korisnici.
- Zona prezentacije podataka, skladišti podatke kako bi oni bili dostupni korisnicima za različite analize. Ovaj dio prezentacije podataka predstavlja skup integrisanih data mart-ova. Ralph Kimball insistira da se u ovom dijelu koristi višedimenzioni model podataka (i to striktno zvjezdasta šema). Data mart-ovi treba da sadrže pored sumarnih i atomske podatke da bi se izašlo u susret različitim zahtjevima korisnika. Dok Bill Inmon ne vidi potrebu za povezivanjem data mart-ova, Kimball ih povezuje preko konformisanih dimenzija i činjenica.

- Alati za pristup podacima (engl. *Data Access Tools*), čine poslednju komponentu skladišta podataka. Ovdje se ubrajaju alati za vršenje ad hoc upita, alati za kreiranje izveštaja kao i razne analitičke aplikacije.

Ovakva arhitektura čini temelj **Data mart BUS arhitekture** koja se zasniva na bottom-up pristupu razvoja skladišta podataka. Data mart BUS arhitektura omogućuje racionalnu dekompoziciju planiranja izgradnje skladišta podataka, i to tako što se najprije definiše njegova opšta arhitektura a zatim se pristupa implementaciji pojedinačnih data mart-ova. Svaka iteracija mora biti u skladu sa određenim opštim arhitekturnim okvirom.



Slika 8. Data Mart BUS arhitektura

Tabela 2. Primjer matrice BUS arhitekture

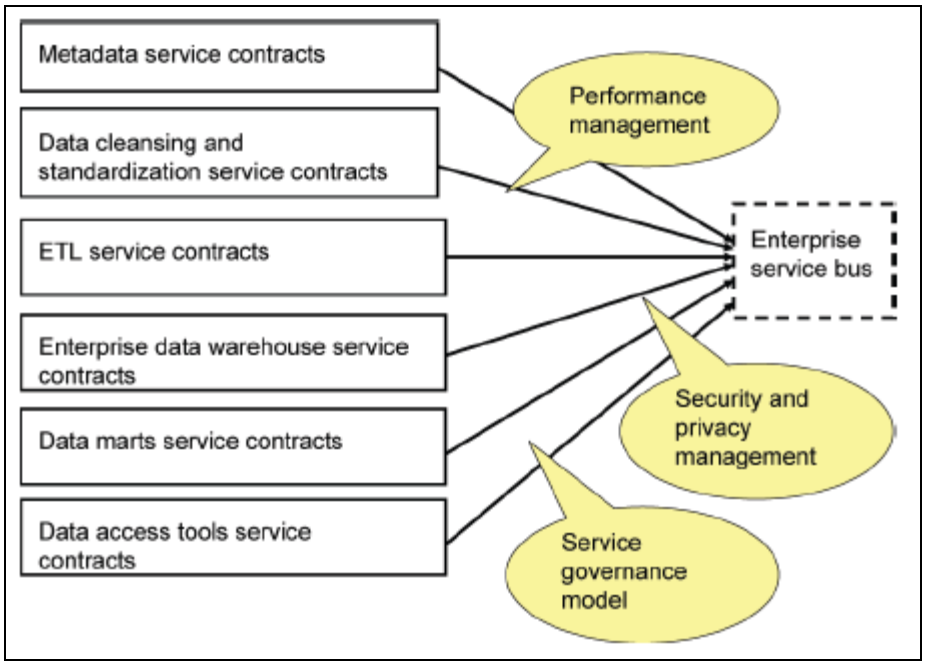
	Dimenzije				
Poslovni proces	proizvod	kvartal	dobavljač	region	maloprodaje
nabavka	X	X	X		
prodaja	X	X		X	X
finansije		X			X

Sagledavanje opšte strukture skladišta podataka realizuje se preko matrice BUS arhitekture koja u redovima prikazuje data martove (ili poslovne procese), a u kolonama prikazuje konformisane dimenzije. Dimenzija je konformisana ako ima jedinstvene nazive atributa, konzistentne ključeve i ako atributi imaju iste vrijednosti [KR02]. Konformisane dimenzije trebaju biti definisane na nivou kompanije i ona se može javiti kao dimenziona tabela u više data mart-ova. Kada se popune redovi i kolone, u ćelijama tabele se unosi X oznaka koja pokazuje koje dimenzije su značajne za pojedine poslovne procese ili data mart-ove.

Postoji i arhitektura gdje data martovi nisu međusobno povezani niti su dimenzije konformisane. Ovaj tip se naziva nezavisni (engl. *independent*) data mart i koristi se kod organizacija koje same razvijaju data martove za svoje potrebe [Pon10]. Najveći nedostatak ovakvog pristupa je nekonzistentnost podataka u data mart-ovima, što će otežati analizu podataka na nivou kompletne organizacije.

Federalna (engl. *federated*) arhitektura se zasniva na postojećoj naslijeđenoj strukturi za podršku odlučivanju u formi posebnih operativnih sistema, izdvojenih skupova podataka, primitivnih data martova i slično [Pon10]. Sistem federalnih baza podataka je kolekcija kooperativnih baza podataka koje su autonomne i eventualno heterogene [SL90]. Ova arhitektura se primjenjuje kada su kompanije puno uložile u razvoj pomenutih sistema, te im nije isplativo graditi sistem za podršku odlučivanju od početka. Podaci se u ovom slučaju integrišu kroz zajednička ključna područja globalne metapodatke i distribuirane upite. Kod ovog tipa ne postoji centralno skladište podataka na nivou organizacije.

Centralizovano skladište podataka podrazumijeva arhitekturu kod koje su se svi podaci iz organizacije nalaze na jednom centralnom skladištu podataka. U ovakvom skladištu se nalaze atomski podaci u trećoj normalnoj formi. Ova struktura nema pojedinačnih data martova. Korisnici dobijaju informacije direktno iz skladišta podataka. Aktivno skladište podataka je ono skladište podataka u kojem je prisutna on-line obrada i ažuriranje iz transakcionih sistema. Visoke performanse obrade transakcija je osobina ovog skladišta podataka [ISN08]. Njegova prednost je što su podaci iz transakcionih sistema dostupni za analizu u najkraćem mogućem roku. Iako ima određene prednosti kod ovakvog skladište podataka može se pojaviti problem održavanja podataka i integriteta transakcija. Ako se transakcija prenosa podataka iz transakcionog sistema ne izvrši, pojavljuje se problem nalaženja podataka. Osim toga, velika je cijena izgradnje takvog skladišta podataka.



Slika 9. Servisno-orijentisano skladište podataka [Hua10]

Skladište podataka se može izgraditi i kao servisno-orijentisano skladište podataka. Servis se može shvatiti kao skup dobro definisanih, labavo povezanih, interoperabilnih softverskih komponenta. Service-oriented architecture (SOA) je skup servisa koji komuniciraju jedan sa drugim. Komunikacija može uključivati bilo koje jednostavno prenošenje podataka ili može uključivati dva ili više servisa sa koordinirnim aktivnostima [SOA].

U SOA-a konceptu, tradicionalnu arhitekturu skladište podataka treba podijeliti na različite servisne komponente prema ugovora o pružanju servisa koji su navedeni u *Enterprise Service Bus* i koji sadrži sve dostupne servise preduzeća (slika 9). SOA bazirana skladište podataka omogućuje pogodnosti kao što su ponovno korišćenje, integracija i brzo reagovanje kompanije na promjene, jer različiti sistemi mogu komunicirati preko iste servisno orijentisane platforme. Kako bi se obezbjedile ove pogodnosti, servisno orijentisana skladišta podataka moraju sadržavati jedinstveni metamodel i integrisani model podataka preduzeća. Metamodel obezbjeđuje integrisanje metapodataka između skladišta podataka i ostatka IT sistema preduzeća. Interoperabilnost metapodataka skladišta podataka i podataka obezbeđuje orkestraciju različitih dijelova u arhitekturi skladišta podataka. Jedna od prednosti implementacije servisno-orijentisanih skladišta podataka je prirodna integracija ETL, baze podataka i alata poslovne inteligencije na nivou servisa [Hua10].

Arhitektura skladišta podataka nije statična. Od početka razvoja skladišta podataka ona se mijenjala u skladu sa novim istraživanjima i potrebama i zahtjevima korisnika. Bill Inmon i saradnici su 2008. objavili knjigu o novoj evoluciji skladištenje podataka: DW 2.0 The Architecture for The Next Generation of Data Warehousing. Prema [ISN08], DW 2.0, između ostalog, prepoznaje životni ciklus podataka u skladištu podataka, prepoznaje potrebu

za uključivanjem tekstualnih podataka u skladište podataka, te daje veći značaj metapodcima u skladištu podataka integrišući tehničke i poslovne metapodatke. DW 2.0 se bavi rješenjem problema skladišta podataka sa velikom količinom podataka, snimanjem podataka tokom dugog vremenskog perioda i podrškom širokoj obradi podataka. Dok se kod DW 1.0 uglavnom koristio metod vodopada kod razvoja sistema, DW 2.0 koristi spiralni model jer korisnici ne znaju šta hoće dok ne vide prototip aplikacije. Osim toga, DW 2.0 nastoji prilagoditi poslovne podatke i biti u mogućnosti iskoristiti u velikoj mjeri nestrukturirane podatke koji čine najveći dio informacija kompanije.

Prije nego se organizacija odluči za izgradnju skladišta podataka ona treba da odgovori na sledeća pitanja [Pon10]:

- Top-down ili bottom-up pristup?
- Da li se najprije izgrađuje skladište podataka ili data martovi?
- Da li najprije napraviti pilot projekat ili kompletan razvoj?
- Da li izgraditi zavisne ili nezavisne data martove?

Sva ova pitanja zahtijevaju pažljivo ispitivanje i planiranje. Organizacija treba da implementira skladište podataka koje će na najbolji način unaprijediti proces poslovnog odlučivanja.

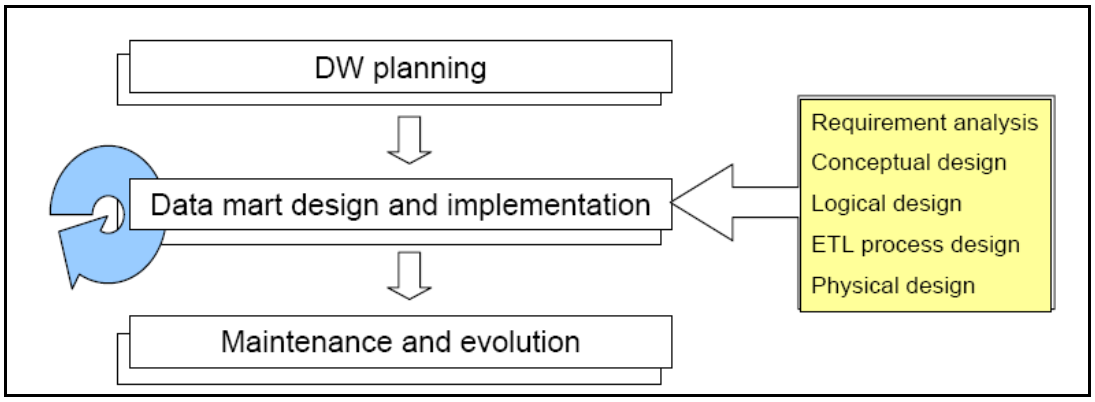
2.6. Metodologije razvoja skladišta podataka

Kako sistem poslovne inteligencije i skladišta podataka ne postoje kao završeni proizvodi za svaki djelatnost ili grupu kompanija, proizvođači softvera nude tehnološke platforme i alate za razvoj. Najvažniji ponuđači platformi i alata poslovne inteligencije na svjetskom tržištu su: Microsoft, Oracle, IBM, SAS Institute, SAP i dr. Implementacija skladišta podataka uglavnom se razlikuje od implementacije OLTP sistema. Projekt izgradnje skladišta podataka je složen, te zahtjeva veliku podršku u okviru organizacije. Kao i ostali projekti u

informacionim tehnologijama i projekt izgradnje skladišta podataka zahtjeva vrijeme i određena novčana ulaganja. Projekat izgradnje skladišta podataka se sastoji od hardvera, softvera, ljudi i organizacionih postupaka.

Kimball je 1996. predstavio višedimenzionalno modelovanje i predložio metodologije za izradu modela skladište podataka na osnovu zahtjeva korisnika, a zatim prema izvoru podataka. Faze razvoja skladišta podataka su: izbor poslovnog procesa, identifikacija dijelova procesa, izbor dimenzija i identifikacija činjenica. Zahtjevi korisnika su ključni kako za odabir poslovnog procesa, tako i za izbor dimenzija i činjenica [Kim96]. S druge strane, prema [Inm00b], zahtjeve korisnika treba razmatrati tek kad se analiziraju poslovni podaci i relevantne transakcije.

Prema [Pon10] životni ciklus projekta izgradnje skladišta podataka se može podijeliti u tradicionalne faze: plan projekta, definicija zahteva, dizajn, izgradnja, razvoj, produkcija i održavanje sistema. Prema [Gol08] proces izgradnje skladišta podataka obuhvata tri osnovne faze: planiranje skladišta podataka, dizajn i implementacija data mart-a, te održavanje i ocjena sistema. U okviru druge faze predviđene su sledeće aktivnosti: analiza zahtjeva korisnika, konceptualni i logički dizajn, ETL proces i fizički dizajn.



Slika 10. Data warehouse, faze životnog ciklusa i dizajna [Gol08]

Danas postoje, s obzirom na funkcionalne zahtjeve, sledeći pristupi kod izgradnje skladišta podataka:

1. Data-driven (koji se još naziva supply-driven) pristup [Inm00a] [WS03] [JHP04] [GTTY06] [Gol08] sa bottom-up tehnikom koji počinje i koji se zasniva na analizi operativnih izvora podataka, šeme baze podataka i identifikaciji svih raspoloživih podataka. U ovom slučaju ograničena je uloga korisnika u procesu izgradnje skladišta podataka. Ovaj pristup je moguć kod sledećih slučajeva [Gol08]:

- Detaljno se poznaju izvori ili je do njih jednostavno doći,
- Šema izvornih podataka pokazuju dobar stepen normalizacije,
- Složenost šeme izvornih podataka nije previsoka.

2. Demand-driven (koji se još naziva requirement-driven ili user-driven) pristup [Kim96] [WS03] [GTTY06] [Gol08] sa bottom-up tehnikom koji počinje od utvrđivanja zahtjeva za informacijama od poslovnih korisnika. Naglasak je na zahtjevima i učestvovanju korisnika. U ovom pristupu može postojati problem mapiranje tih zahtjeva na sa dostupnim podacima.

3. Goal-driven pristup [BU00] [Wes01] [GTTY06] se zasniva na biznis strategiji o kojoj su informacije dobijene na osnovu intervjua sa top menadžmentom kompanije. Ovaj pristup koristi top-bottom tehniku i počinje sa analizom

ključnih poslovnih procesa. Prema [WS03] [RA09], goal-driven je samo jedan od oblika Demand-driven pristupa.

4. Hibridni pristup kombinuje zahtjeve korisnika i top menadžmenta sa analizom operativnih podataka. Ovdje razlikujemo *interleaved* hibridni pristup i sekvencijalni hibridni pristup. Razlika je što se od sekvencijalnog pristupa data-driven i demand-driven paradigme primjenjuje nezavisno jedna od druge i jedna poslije druge, dok se kod *interleaved* hibridnog pristupa obe paradigme primjenjuju paralelno u cijelom procesu [RA09].

Svaka od paradigmi ima svoje prednosti i nedostatke. Hibridni pristupi nastoje riješiti nedostatke svake od paradigmi i iskoristiti njihove prednosti. Skladišta podataka se izgrađuju različitim metodološkim pristupima, uzimajući u obzir specifičnosti organizacije, te vodeći računa o očekivanoj koristi od implementacije takvog sistema. Kad organizacija donese odluku za izgradnju skladišta podataka, ona se treba odlučiti za neku od mogućih strategija uvođenje. Kod izgradnje skladišta podataka, treba imati na umu da ono nije nikad do kraja izgrađeno, već će se kontinuirano razvijati u skladu sa promjenama koje se događaju u poslovnom okruženju.

Kao i kod razvoja transakcionih baza podataka, razvoj skladišta podataka mogu pratiti određeni problemi od kojih je najznačajniji spor odgovor na promjene u poslovnom okruženju, što dovodi do prekoračenja vremena razvoja, povećanja troškova i nedostatak pravovremenih poslovnih informacija. Zbog toga neki autori [CBD10] [HS10] [Hug08] [Pon10] razmatraju korišćenje agilnih metodologija kod razvoja skladišta podataka. Agilne metodologije predstavljaju reakciju na probleme klasičnih metodologija razvoja softvera da brzo prezentuju konkretne rezultate i odgovore na promjene, što je postalo neophodno radi podrške poslovanju preduzeća koja se moraju prilagođavati

promjenama iz okruženja [CH01]. Agilne metodologije pokušavaju da postignu traženu brzinu razvoja izbjegavanjem balasta klasičnih metodologija: pretjerano planiranje, modelovanje, dokumentovanje i sve aktivnosti koje nisu u direktnoj funkciji finalnog proizvoda. Za razliku od MDA principa, modeli se formiraju samo ako su u funkciji programskog koda, najčešće za potrebe razmene ideja u okviru projektnog tima [MDP07]. Danas su mnoge organizacije prihvatile načela i prakse agilnog razvoja i ostvarile zapažen uspjeh u svojim softverskim projektima. Sve više kompanija primjenjuje agilni razvoj za skladištenje podataka i projekat poslovne inteligencije [Pon10]. Svaka faza razvoja skladišta podataka ima za cilj čišćenje grešaka iz prethodne faze. Osim toga, konačni proizvod je podložan stalnim promjenama. To ne znači da će se projekt izgradnje skladišta podataka izvoditi beskonačno jer postoje vremenska i finansijska ograničenja [CBD10]. Izgradnja i implementacije skladišta podataka može biti skup proces. Agilni pristup u razvoju skladišta podataka ne samo da nudi praktične financijske uštede, nego i skraćuje vrijeme isporuke i poboljšava kvalitet sistema [Hug08]. Ovaj pristup usklađuje razvoj skladišta podataka s potrebama kupaca i ciljevima kompanije, te predstavlja skup najbolje prakse koji omogućuju brzu i kvalitetnu izradu skladišta podataka [HS10]. U [Corr11] je opisan tzv. BEAM (*Agile dimensional modeling using Business Event Analysis & Modeling*) agilni pristup za dimenziono modelovanje koji unapređuje komunikaciju između DW dizajnera, BI stejkholdera i cjelokupnog DW/BI tima.

2.7. Prostorna i vremenska skladišta podataka

Procjenjuje se da oko 80% od podataka pohranjenih u bazama podataka ima prostornu komponentu [RBM01]. Prostorni podaci se najviše nalaze u područjima lokalne uprave, zdravstva i zaštite životne sredine. Ovi podaci mogu predstavljati objekte na površini Zemlje, kao npr. planine, gradove i

rijeke, ili geografskih pojave, kao što su temperatura, i nadmorska visina. Prostorni podaci mogu predstavljati negeografske podatke kao što su ljudsko tijelo, motor automobila i dr. Količina prostornih podataka raste znatno zahvaljujući tehnološkom napretku u područjima kao što su globalni navigacioni satelitski sistem (GNSS) i Global Positioning System (GPS) [MZ08]. Upravljanje prostornim podacima se odvija u prostornim bazama podataka ili geografskim informacionim sistemima (GIS). Kao i u slučaju baza koje sadrže poslovne podatke, baze prostornih podataka nisu pogodne za podršku odlučivanja. Zbog toga su se pojavila prostorna skladišta podataka (engl. *spatial datawarehouse*) koja čine kombinaciju prostornih baze podataka i data warehouse tehnologija [MZ08]. Prostorna skladišta podatak omogućuju značajnu podršku odlučivanju kroz analizu, vizualizaciju i manipulaciju prostornim podacima.

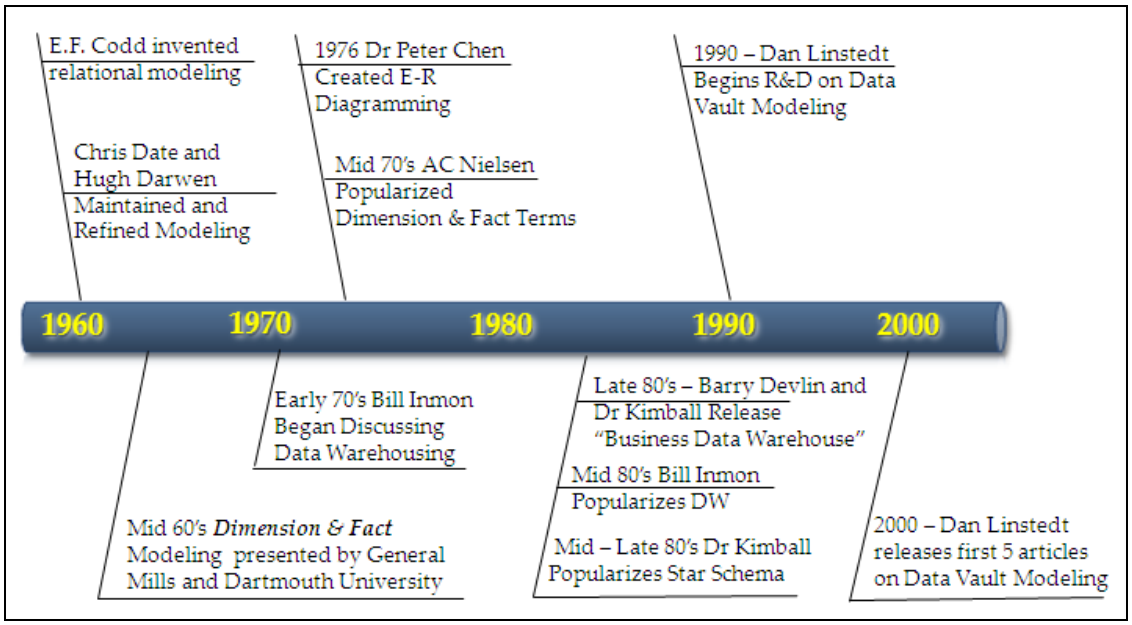
Koncepti vremena i vremena transakcija uvedeni su u SQL jezik, čija specifikacija je razvijena u 1993., a kao odgovor na prijedlog Richarda Snodgrass-a za razvoj vremenske ekstenzije za SQL [SA86]. Danas se u bazama podataka nalazi mnogo podataka koji se mijenjaju sa vremenom. Kod ovih podataka bitna je kako nova vrijednost podatka, tako i vrijeme kada se promjena dogodila (engl. *valid time*) i vrijeme kada je podatak učitan u bazu podataka (engl. *transaction time*). Naprimjer, u sistemu zdravstva bitna je informacija o određenom pregledu nekog pacijenta kao i podatak o vremenu izvršavanja svih njegovih prethodnih pregleda, te o vremenu učitavanja tih podataka u bazu podataka. Temporal baze podataka omogućuju upravljanje vremenskim informacijama. Važna karakteristika vremenskih podataka njegova granularnost ili njegov nivo detalja [MZ08] [DDL03], odnosno da li se evidencija u bazi vodi u okviru godine, kvartala, mjeseca, dana, sata ili minuta. Iako vremenske baze podataka omogućuju pregled istorijskih podataka, oni ne

nude sadržaje za podršku procesu odlučivanja kada su u pitanju velike količine podataka. Zbog toga je istraživanje na području baza podataka i vremenskih skladišta podataka dovela je do razvoja nove oblasti pod nazivom vremenske skladišta podataka (engl. *temporal data warehouse*). Temporal semantika je sastavni dio vremenskih skladišta podataka na sličan način kao što je slučaj za vremenske baze podataka [MZ08]. Model vremenskih skladišta podataka treba da predvidi kako vrijeme kada se neke promjena dogodila, tako i vrijeme kada je podatak o promjeni učitan u skladište podataka, omogućujući na taj način praćenje promjene bilo kojeg podatka u određenom vremenskom periodu.

2.8. Data Vault

Data Vault predstavlja detaljno orijentisani, istorijski praćen i jedinstveno povezani skup normalizovanih tabela koji podržavaju jedan ili više funkcionalnih područja poslovanja. To je hibridni pristup koji obuhvata najbolje osobine između 3. normalne forme (3NF) i zvijezda šeme.

Dizajn je fleksibilan, skalabilan, dosljedan i prilagodljiv potrebama preduzeća. To je model podataka koji je projektovan posebno za potrebe skladišta podataka organizacije [Lin1] [Lin2].

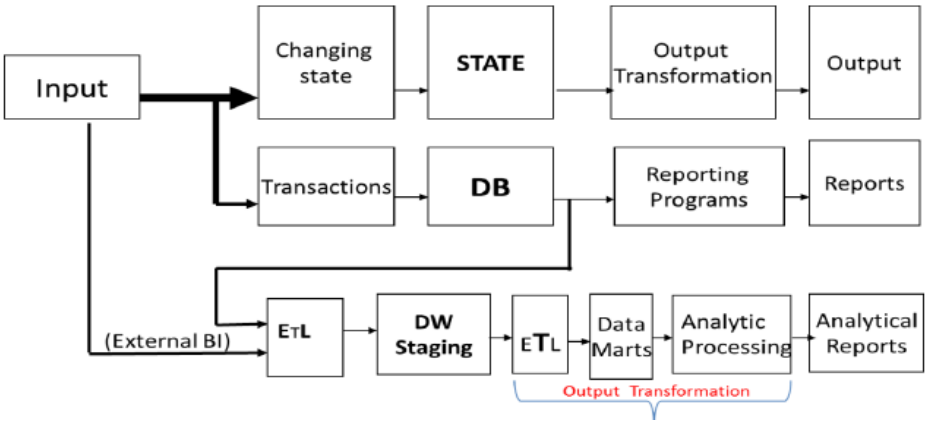


Slika 11. Razvoj Data Warehouse i Data Vault koncepta [Lin3]

Data Vault koncept je osmislio Daniel Linstedt objavivši sredinom 2002. godine prvi od pet članaka pod nazivom Data Vault Series. Ostali članci su objavljeni do početka 2005. na sajtu http://www.tdan.com/view-articles/5054/. Data Vault se pojavio kao odgovor na probleme koji su pratili razvoj skladišta podataka dizajniranih u trećoj normalnoj formi i star šemi, nastojeći iskoristiti najbolje elemente jednog i drugog koncepta. Performanse i druge slabosti 3NF i zvijezda shema (kada se koristi za skladište podataka) počele su se pojavljivati u 90-im kada se količina podataka počela povećavati. Data Vault je projektovan za prevazilaženje tih nedostataka zadržavajući prednosti 3NF i zvijezda shema arhitekture [Lin1]. Ovaj koncept je osmišljen da se model podataka može lako promijeniti u skladu sa promjenama u poslovnom okruženju [Dam08], te kako bi se omogućilo paralelno i brzo učitavanje podataka koliko je god to moguće [HS10].

Data Vault koncept naglašava potrebu da se ostavi trag odakle i kad su podaci došli u bazu podataka [Lin10]. Konceptualni dijagram na slici 12 iz [JBKP12]

[JM13] pokazuje izričito odvajanje originalnih podataka (Data Vault) i data martova, za razliku od CIF arhitekture gdje su podaci spojeni na početku.



Slika 12. Data warehouse staging i reporting [JBKP12]

State treba apstraktno shvatiti kao skup minimalnijh informacija o sistemu, a *changing state* su događaji koji utiču na sistem posmatrano kroz neku transformaciju izlaza. S druge strane iste informacije ulaze preko transakcija i trenutno ažuriraju operativnu bazu podataka koja te promjene prikazuju preko programa za izvještavanje. Na kraju skladište podataka integriše te podatke sa spoljnim informacijama kroz procese ekstrakcije i učitavanja. Poslije učitavanja u prelaznu bazu se radi transformacija i učitavanje u data mart [JBKP12]. Ovo odvajanje reverzibilne, odnosno lake transformacije bitna je ideja koja se zagovara u Data Vault konceptu prema [Lin2], [Lin10] i [LGH08]. Da bi u potpunosti obezbedili potpune informacije, u smislu istorijskih podataka o prošlosti i sadašnjosti, a potom kroz bilo koji dizajn DW sistema, moramo obraditi vremenski aspekt podataka (promjenom vrijednosti u veremenu bez anomalija, po analogiji s tradicionalnim normalnim formama) [JBKP12].

Dizajn Data Vault sistema je jednostavan sa minimalnim brojem komponenti kojeg čine hub, link i satelit entiteti.

Hub entitet u tabeli nosi minimum jedinstvene liste poslovnih ključeva (engl. *business key*). To su ključevi koje organizacija koristi u svakodnevnom poslovanju kao npr. broj (šifra) kupca, broj (šifra) zaposlenog, broj računa, i sl. Ostali atributi hub tabele uključuju [Lin2]:

- Implementacioni identifikator-ključ, (engl. *Surrogate*) ključ, opciona komponenta, eventualno pametni ključ ili sekvencijalni broj
- Datum i vrijeme učitavanja (engl. *Load Date Time Stamp*), pokazuje trenutak kada se podatak pojavio u skladištu podataka
- Izvor zapisa (engl. *Record Source*), izvorni sistem odakle je podatak učitan

Link entitet je fizički prikaz referenci stranih ključeva i više-prema-više odnosa u 3NF. Link tabela služi za predstavljanje odnosa ili transakcija između dviju ili više poslovnih komponenata (dva ili više hub-ova). Link sadrži sljedeće atribute [Lin2]:

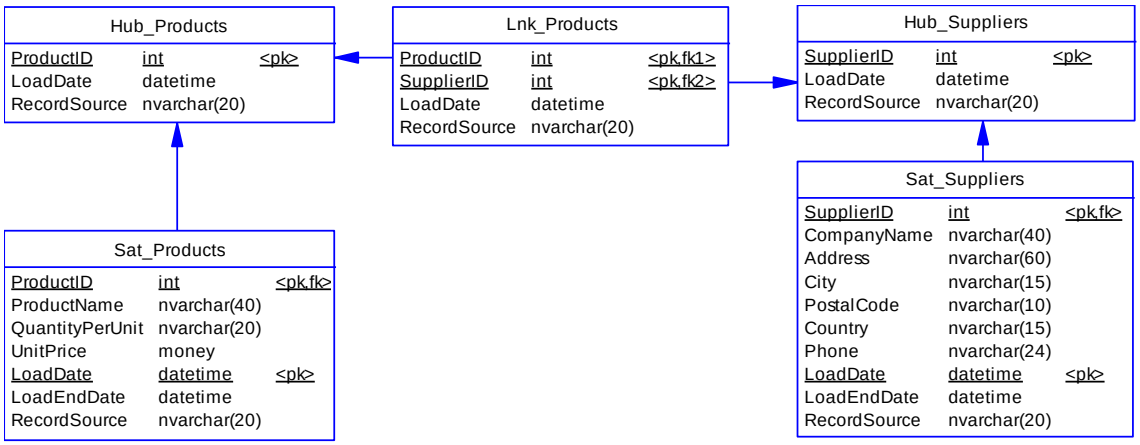
- Nadomjesni ključ, opciona komponenta, eventualno pametni ključ ili sekvencijalni broj
- Hub ključ premješten u link
- Datum i vrijeme učitavanja, pokazuje trenutak kada se podatak pojavio u skladištu podataka
- Izvor zapisa, pokazuje na izvorni sistem odakle je podatak učitan

Satelit entitet pokazuje kontekst Hub podataka. Sve informacije su podložne promjenama tokom vremena, što znači da struktura mora biti sposobna za snimanje novih ili promijenjenih podataka [Lin1]. Satelit tabela se sastoji od sljedećih atributa [Lin2]:

- Satelitski primarni ključ, hub ili link primarni ključ migriran u satelitski
- Satelitski primarni ključ, datum i vrijeme učitavanja pokazuje trenutak kada se podatak pojavio u skladištu podataka

- Satelitski opcioni primarni ključ, sekvencijalni surogat broj koji se koristi za satelite koji imaju više vrijednosti
- Izvor zapisa, pokazuje na izvorni sistem odakle je podatak učitan

Svaki hub ili link može imati više od jednog satelita. Naprimjer, ako imamo dva izvorna sistema koji sadrže podatke za kupce: prodaja i marketing. U tom slučaju možemo napraviti dva satelita, jedan za svaku izvorni sistem [CBD10]. Drugi razlog za formiranje više satelita je različita brzina promjene različitih opisnih podataka koje sadrži satelit tabela.



Slika 13. Primjer Data Vault sistema sa hub, link i satelit tabelama

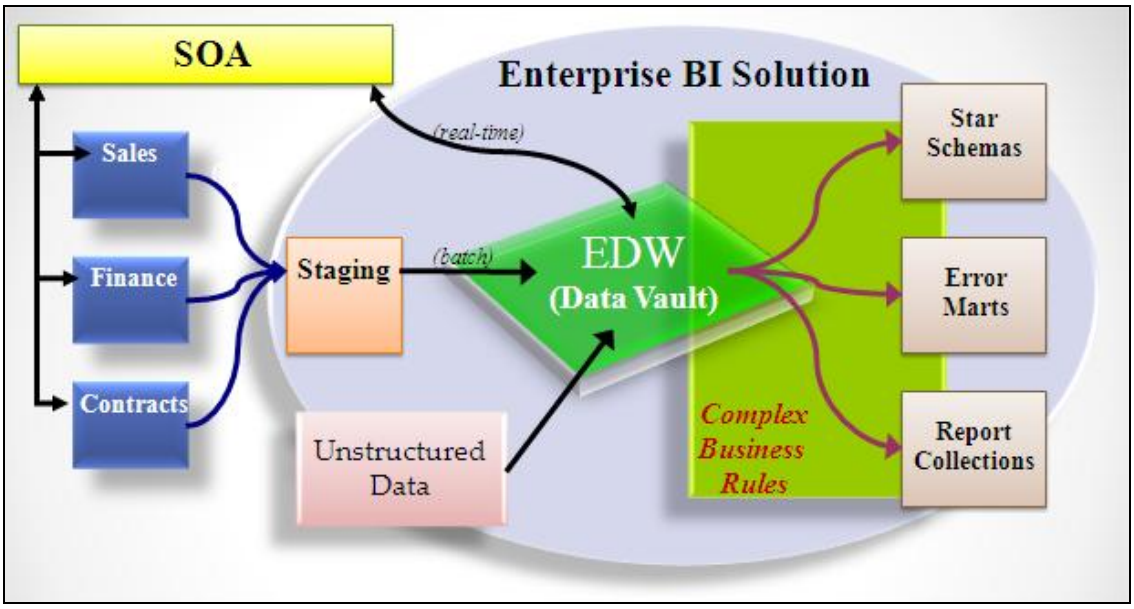
U primjeru na slici 13. prikazan je slučaj sa dvije hub tabele (Hub_Product i Hub_Supplier), odgovarajućim satelit tabelama (Sat_Products i Sat_Suppliers), te vezom između hub tabela (Lnk_Products). U ovom slučaju pretpostavka je da su poslovni ključevi stabilni (da se ne mijenjanju) i da su jedinstveni na nivou kompanije (ProductID, SupplierID), zbog čega je izostavljen surogat ključ na nivou hub tabela. Moguć je i pristup gdje će hub tabele sadržavati i dodatni surogat ključ, ali u tom slučaju nad kolonom koja označava poslovni ključ potrebno je da postoji jedinstveni (engl. *unique*) index.

Link tabela sadrži primarne ključeve hub tabela, datum i vrijeme učitavanja i izvor zapisa. Satelit tabele sadrže opisne podatke iz odgovarajućih tabela iz OLTP baze, datum i vrijeme učitavanja i izvor zapisa. Pored toga satelit tabele sadrže i kolonu LoadEndDate koji pokazuje dokad je odgovarajući slog bio važeći i koji inicijalno ima vrijednost NULL. Npr. ako se za jednog dobavljača određenog datuma promjeni adresa, telefon i fax, u tabeli Sat_Suppliers će se u koloni LoadEndDate upisati taj datum, te u tabeli dodati novi slog sa izmjenjenim podacima (LoadDate će imati novu vrijednost sa pomenutim datumom, a LoadEndDate će imati novu vrijednost NULL). Kod svih tabela Data Vault sistema, pri kreiranju primarnog ključa kreira se i jedinstveni indeks nad kolonama koje čine primarni ključ.

Data Vault ponekad sadrži i četvrti element **Refernce Table** u formi statičkog hub-a koji enkapsulira svoje satelite na tradicionalan ne-istorijski način [JPB12]. Ova tabela sadrži ime, identitet i opis. Ime se koristi za idenfifikaciju naziva, identitet je jedinstveni identifikacioni ključ unutar svake tabele, opis je string koji se koristi da opiše jedinstveni ključ unutra svake tabele [Kno12]. Na slici 14. prikazana je tipična Data Vault arhitektura koja se sastoji od: izvora podataka, staging tabela, enterprise data warehouse (izgrađene kao Data Vault) i dijela za prezentaciju podataka korisniku informacija.

Data Vault bi trebao biti izgrađen na sljedeći način [Lin3]:

1. Model hub-ova. To zahtijeva razumijevanje poslovnih ključeva i njihovo korišćenje kroz određeni opseg.
2. Model link-ova. Formiranje odnosa između ključeva - razumijevanje poslovanja u kontekstu poslovnih ključeva.
3. Model Satelita. Određivanje konteksta svakog poslovnog ključa, transakcija (linkovi) koji povezuju hubove, čime se pruža cjelovita slika poslovanja.



Slika 14. Data Vault arhitektura [Lin3]

Prema [Lin3] i [Gra11] Data Vault ima najviše smisla koristiti u slučaju distribuiranih izvora podataka. Da bi se obezbedila sljedivost podataka, Data Vault ne transformiše podatke iz različitih izvora prije nego što se učitaju u skladište, čime se omogućuje trajan sistem evidencije za razliku od CIF arhitekture gdje se podaci prije učitavanja transformišu.

U Data Vault konceptu postoje mogućnosti za reprezentaciju spoljnih izvora podataka kao što su txt datoteke i Excel fajlovi. Bez obzira na vrstu izvora, potrebno je primijeniti navedene strukture i tehnike izgradnje Data Vault sistema. U posljednjih nekoliko godina ova tehnika je nalazi sve veću primjenu. Prema http://danlinstedt.com/about/dvcustomers/, broj kompanija korisnika u SAD-u u 2010. godini nadmašio je pet stotina.

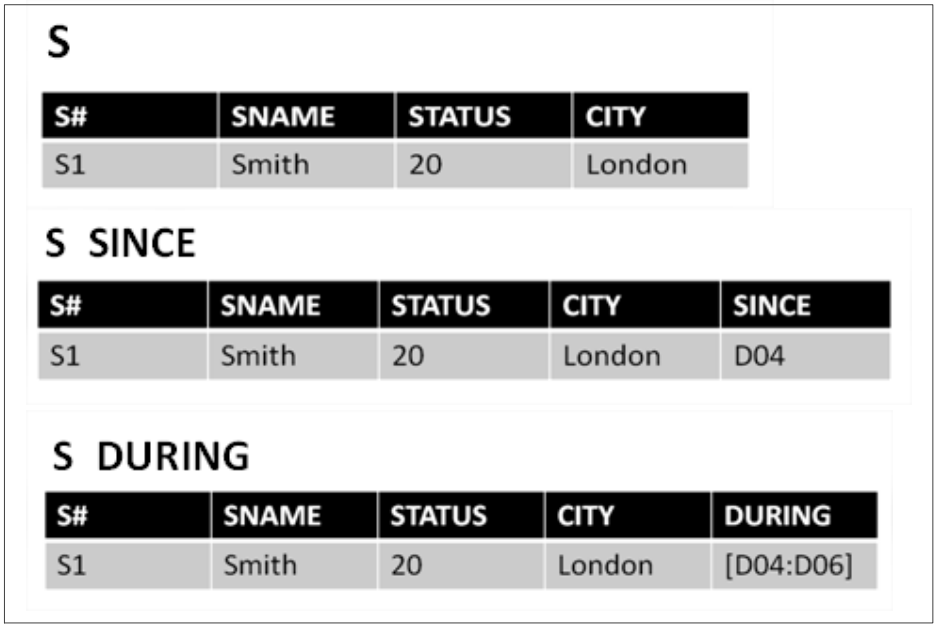
2.8.1. Teorijska osnova Data Vault modela

Prema [LMAB08], normalne forme su istorijski postepeno uvođene u teoriju relacionih baza podataka, počev od prve koju je definisao Codd 1972. godine, zatim druge i treće normalne forme, pa preko Boyce-Codd-ove, četvrte normalne forme (4NF) u kojoj su funkcionalne zavisnosti uopštene u višeznačne, do pete normalne forme (5NF) u kojoj su višeznačne zavisnosti uopštene u zavisnosti spajanja. Postavilo se pitanje postoje li dalja uopštenja zavisnosti atributa u relacijama i da li se može definisati neka nova (šesta, sedma...) normalna forma. R. Fagin je 1981. godine dao najopštiju definiciju normalne forme ključeva i domena (Domain-Key normal form – DKNF) i pokazao da je relacija koja je u DKNF ne prouzrokuje anomalije u održavanju i obrnuto, da se relacija koja ne prouzrokuje anomalije u ažuriranju nalazi u DKNF [LMAB08].

Šestu normalnu formu (6NF) u teoriju relacionih baza podataka je uveo Christopher Date da opiše baze podataka u kojima se dekomponuju relacione varijable. Dok ovaj oblik može biti nevažan za nevremenske podatke, svakako je važno za održavanja podataka koji sadrži vremenske varijable point-in-time ili interval vremena. Sa pojavom Data Warehousinga 2.0, tu je sada povećan naglasak na korišćenje potpuno temporalizovanih baza podataka u kontekstu skladištenje podataka [Kno12] [KJ12].

Christopher Date ističe nekoliko problema pri pokušaju modelovanja potpuno temporalizovanih podataka koristeći 5NF**.** Za ilustraciju najprije koristi netemporalizovanu šemu podataka zatim šemu kompatibilnu sa 5NF (tabela Supplier - Dobavljač).

Uvođenjem vremenskog atributa SINCE (vrijeme od kada dobavljač ima ugovor) šema postaje semi-temporalized. Ovakvu relaciju možemo napraviti i korištenjem atributa DURING koji će sadržavati period ugovora sa dobavljećem (slika 15 [DDL03]).



S

S#	SNAME	STATUS	CITY
S1	Smith	20	London

S SINCE

S#	SNAME	STATUS	CITY	SINCE
S1	Smith	20	London	D04

S DURING

S#	SNAME	STATUS	CITY	DURING
S1	Smith	20	London	[D04:D06]

Slika 15. Non-temporalized i Semi-temporalized Table Schema

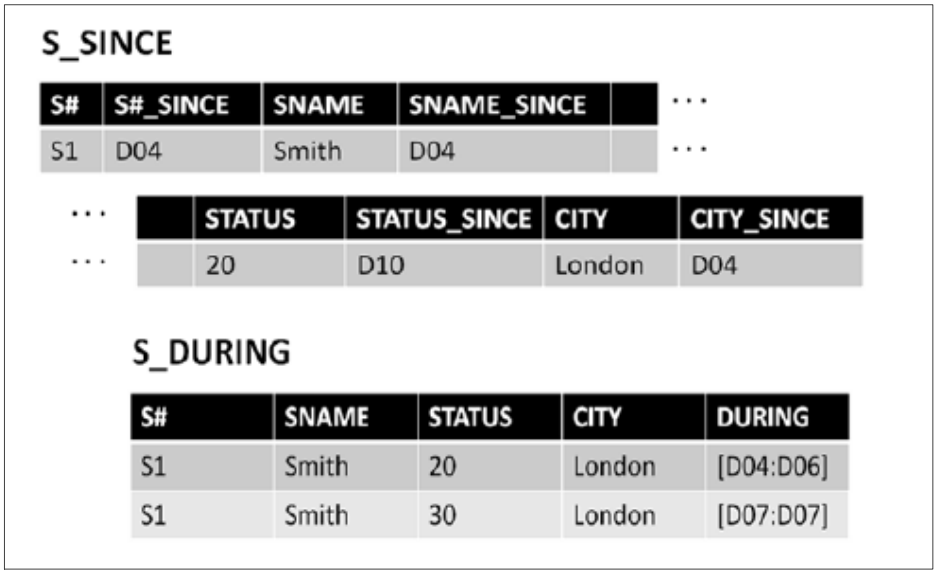
U oba slučaja, iako je vremenski element uveden u bazu relacija je loše dizajnirana iz dva razloga [DDL03]:

1. Jedan vremenski atribut nije dovoljan za modelovanje u uslovima gdje se svaki od ostalih pojedinačnih atributa mogu razlikovati nezavisno jedan od drugog tokom vremena,

2. Podaci mogu biti izgubljeni kada se jednom vremenski atributi ažuriraju. Istorijski podaci se tako ne mogu pratiti u ovom scenariju.

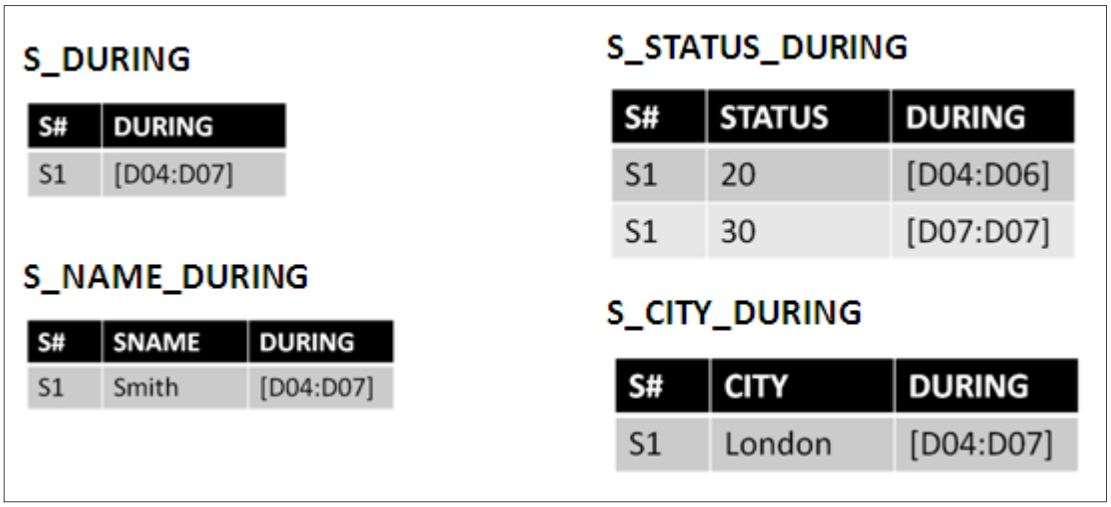
Zbog tih nedostataka, ova vrsta modelovanja se naziva "polu-temporalizing" [DDL03]. Da bi se ovo prevazišlo SINCE atribut se može dodati za svaki opisni atribut u tabeli (sljedeća slika). Rezultat je horizontalna eksplozije tabele.

Sličan problem imamo ako koristimo atribut DURING



Slika 16. Semi-temporalized Table Schema sa SINCE i DURING [DDL03]

Rješenje problema se sastoji od uvođenja vremenske odrednice za svaki atribut, odnosno vertikalne dekompozicije tabele kao što je prikazano na sljedećoj slici.



Slika 17. Potpuno temporalizovana Table Schema sa DURING [DDL03]

Dodavanjem kolona sa vremenom prvog kreiranja CREATED i zadnje izmjene sloga UPDATED dobijem potpuno temporalizovanu tabelu u 6NF sa praćenjem istorijskih podataka.

S_DURING

S#	DURING	CREATED	UPDATED
S1	[D04:D07]	D04	D07

S_NAME_DURING

S#	SNAME	DURING	CREATED	UPDATED
S1	Smith	[D04:D07]	D04	D07

S_STATUS_DURING

S#	STATUS	DURING	CREATED	UPDATED
S1	20	[D04:D06]	D04	D06
S1	30	[D07:D07]	D07	D07

S_CITY_DURING

S#	CITY	DURING	CREATED	UPDATED
S1	London	[D04:D07]	D04	D04

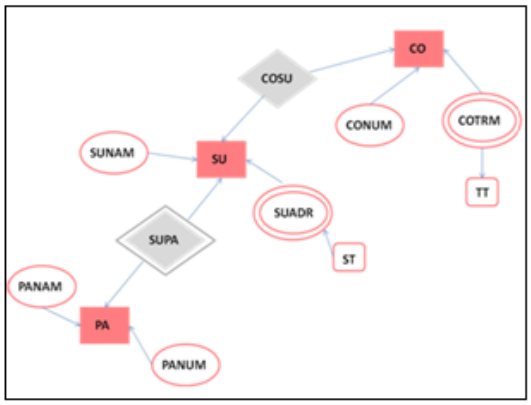
Slika 18. Potpuno temporalizovana Tabela u 6NF [DDL03]

Ovakva vertikalna dekompozicija je slična klasičnom procesu normalizacije [Kno12] i vodi nas do definicije šeste normalne forme Cristophera Date-a: „Relacija (tabela) R je u 6NF ako i samo ako ne postoji netrivijalna zavisnost spajanja, u kojem slučaju se kaže da je R nesvodljiva“ [DDL03].

2.8.2. Data Vault, Anchor modelovanje i 6NF

Anchor Modelovanje (AM) je u teoriju baza podataka uveo Lars Ronnback [Ron01] (www.anchormodeling.com). Unapređenje modela je dato u radovima Ronnbacka sa saradnicima [RRBJW, RRBJW1, Ron11]. Ova tehnika polazi od pretpostavke da se okruženje skladišta podataka stalno mijenja. Kako se anchor modelovanje zasniva na 6NF što podrazumijeva velik broj komponenti, velike promjene izvora podataka neće izazvati velike promjene modela skladišta podataka kao što je slučaj kod tradicionalnih tehnika modelovanja. Model se zasniva na četiri osnovne komponente: sidro (engl. *anchor*), atribut (engl. *attribute*), veza (engl. *tie*) i čvor (engl. *knot*). Svaka fizička tabela se dijeli na

navedene komponente, što dovodi do eksplozije tabele. Da bi se projektant lakše snalazio među mnogo tabela, uvedena su posebna pravila imenovanja [RRBJW1].



ANCHOR	ATTRIBUTE
PA-part	PANAM-PartName
SU-supplier	PANUM-PartNumber
CO-contract	SUNAM- SupplierName
	SUADR- SupplierAddress
	CONUM-ContractNumber
	COTRM-ContractTerm
TIE	**KNOT**
SUPA-SupplierPart	ST-SupplierStatus
COSU-ContractSupplier	TT-ContractTermType

Slika 19. Primjer Anchor modela [JBKP12]

Anchor se koristi za pandan hub-u kod Data Vault modela, ali sadrži samo surogat ključ koji se modeluje kao atribut sidra. Veza daje informaciju o povezanosti dva sidra. Atribut se koristi za modelovanje osobina Anchora i pored poslovnog ključa sadrži i vrijeme učitavanja. Čvor entitet sadrži zajedničke osobine ili veze.

Korelacija između osnovnih komponenti DV i AM može se predstaviti na taj način što se hub-ovi izjednačavaju sa *anchor*, link-ovi izjednačavaju do vezama (tie), Satelite izjednačavaju na Atributima i se referentne tabele sa čvorovima (knot) [JK12].

Data Vault model, u ekstremnom slučaju, u kojem se satelitske tabele sastoje od jednog atributa, postaje dizajniran u 6NF.

Tabela 3. Poređenje modela skladišta podataka [SA12]

	Star schema	CIF	Data Vault	Anchor
Normalizovanost	Niska	Srednja	Visoka	Estremna
Namjena	Front-end	Front/Back-end	Back-end	Back-end
Pristup razvoju	Top-down	Bottom-up	Bottom-up	Bottom-up
Fokus	Korisnici	Podaci	Podaci	Domen
Otpornost na promjene	Niska	Niska	Visoka	Visoka
Vrijeme razvoja	Dugo	Dugo	Kratko	Kratko
Održavanje	Kompleksno	Kompleksno	Jednostavno	Jednostavno
ETL	Vrlo kompleksan	Kompleksan	Jednostavan	Automatizovan
Auditabilnost	Nije definisana	Nije definisana	Ugrađena u model	Ugrađena u model
Veličina	Velika	Srednja	Mala	Ekstremno mala

Prema [SA12], Anchor i Data Vault modeli su dizajnirani da olakšaju back-end procese integracije podataka, dok su stariji modeli naglašavali front-end procese bliže poslovnim korisnicima.

Postojeći modeli za izgradnju skladišta podataka imaju svoje prednosti i nedostatke. U današnjem dinamičnom poslovnom okruženju, prepoznaje se rastuća potreba za fleksibilnim modelima skladišta podataka kao što su Data Vault i Anchor model. Iako ovi modeli imaju određene nedostatke, moguće je

da su u kombinaciji s modernim projektnim metodologijama, ovakvi modeli postati budućnost skladištenja podataka [SA12].

2.9. Kvalitet skladišta podataka

S obzirom da jedna od hipoteza razmatra uticaj automatizacije na kvalitet skladišta podataka, u ovom dijelu rada dat je kratak osvrt na oblast kvaliteta skladišta podataka. Mnogi autori ističu značaj kvaliteta skladišta podataka radi funkcionisanja i održavanja skladišta podataka.

Kvalitet skladišta podataka obuhvata kvalitet metapodataka i kvalitet samog skladišta podataka. Kvalitet skladišta podataka zavisi od kvaliteta izvora podataka, kvaliteta ETL procesa, te kvaliteta same baze podataka skladišta [JQJ98]. S druge strane, kvalitet informacija predstavlja kvalitet DW sistema i kvalitet prezentacije informacija. Kvalitet DW sistema podrazumijeva kvalitet DBMS-a, kvalitet modela i kvalitet podataka. U cilju procjene DBMS najčešće se koristi međunarodni ISO (*International Organization for Standardization*) standard. Ovom vrstom kvaliteta treba se baviti u fazi izbora konkretnog DBMS proizvoda [CPPS01]. Kvalitet podataka se sastoji od kvaliteta definicije podataka, kvaliteta sadržaja podataka i kvaliteta prezentacije podataka [Eng96]. U cilju postizanja što većeg kvaliteta podataka mora se posebna pažnja obratiti na ETL procese. Kvalitet modela skladišta podataka ima veliki uticaj na ukupan kvalitet informacija [CPPS01]. Kvalitet modela se posmatra sa dva aspekta, kvalitet fizičkog modela skladišta podataka i kvalitet dimenzionog modela skladišta podataka. Promjena zahtjeva korisnika, u procesu razvoja ili održavanja skladišta podataka, u najvećem broju slučajeva utiče na promjenu arhitekture skladišta podataka. S druge strane promjene u arhitekturi skladišta

podataka mogu imati uticaj na pomenute elemente kvaliteta skladišta podataka [VBQ99].

U cilju utvrđivanja kvaliteta skladišta podataka razvijeni su različiti trendovi, standardi i različite metrike za utvrđivanje kvaliteta skladišta podataka. Danas je sigurno najpoznatiji trend u oblasti menadžmenta kvalitetom Total Quality Management (TQM). TQM je filozofija poboljšanja organizacije u cilju postizanja izvrsnosti. To uključuje čitavu strukturu i rad organizacije. Vassiliadis u radu [Vass00] se daju smjernice za postizanje visokog kvaliteta skladišta podataka u kontekstu TQM-a.

Jedan od najpoznatijih standarda za ocjenu kvaliteta softvera je ISO/IEC 9126. Ovaj standard nudi šemu u dva nivoa sa karakteristikama i potkarakteristikama. Standard ISO/IEC 9126 definiše šest eksternih karakteristika kvaliteta softverskog proizvoda: funkcionalnost, pouzdanost, upotrebljivost (lakoća korišćenja), efikasnost, lakoća održavanja i portabilnost, pri čemu ne definiše konkretne metrike za njihovu ocjenu, ostavljajući time slobodu da se za svaki konkretni softverski projekat usaglase mjere kvaliteta između naručioca i isporučioca softvera [ISO9126].

S obzirom da bi analiza svih karakteristika i podkarakteristika kvaliteta softvera prevazilazila obim i temu ovog rada, u ovom dijelu rada dat je širi opis jedne od karakteristika - *lakoća održavanja*. Ova karakteristika je odabrana s obzirom da se u procesu održavanja skladišta podataka često pojavljuju novi zahtjevi korisnika softvera. Implementacija novih zahtjeva često ima uticaj na arhitekturu skladišta podataka, a promjena arhitekture može izazvati promjenu kvaliteta skladišta podataka. Održavanje aplikativnog softvera predstavlja skup svih aktivnosti neophodnih za obezbeđenje efikasne podrške softverskom

sistemu. Prema [ISO9126]: „održavanje je karakteristika kojom se utvrđuju aktivnosti potrebne da se izvrše određene ispravke i dopune softverskog proizvoda. Lakoća održavanja je vezana za napor potreban da se izvrše specificirane modifikacije, bilo u cilju otklanjanja grešaka, bilo u cilju ispunjenja dodatnih zahtjeva korisnika. Lakoća održavanja obuhvata analizu podarakteristika softvera: mogućnost analize, stabilnost, izmjenljivost i mogućnost testiranja. *Mogućnost analize* je osobina kojom se utvrđuju nepotpunosti softverskog proizvoda, uzroci i zahtjevi za izmjenama i definiše šta treba da se ispravi i dopuni. *Izmjenljivost* je osobina kojom se utvrđuje pogodnost softverskog proizvoda za otklanjanje grešaka ispravkama, doradama i izmjenama. Izmjenljivost je usko vezana sa problemima podrške isporučioca softverskog proizvoda korisniku, posebno ukoliko isporučeni proizvod zahtjeva modifikaciju da bi se udovoljilo specifičnim zahtjevima korisnika.“ *Stabilnost* je osobina kojom se mjere efekti i rizici od izmjena softverskog proizvoda, dok je *mogućnost testiranja* osobina kojom se vrši ocjena izmjenjenog softverskog proizvoda [ISO9126].

Uticaj pristupa predstavljenog u ovoj disertaciji, odnosno automatizacije projektovanja skladišta podataka, na kvalitet softvera - karakteristika lakoća održavanja biće dat u zaključnim razmatranjima.

U radu [RK07] je dat pregled osnovnih karakteristika ISO/IEC 9126 i pregled ostalih značajnijih standarda za ocjenu kvaliteta softvera, kao što su Capability Maturity Model Integration - CMMI [CMMI], Control Objectives for Information and Related Technology - CobiT [Cobit], IT Infrastructure Library - ITIL [ITIL] i Six Sigma [SixSig].

Jedna od najpoznatijih metrika za utvrđivanje kvaliteta softverskih projekata, uključujući i skladišta podataka je Goal Question Metric (GQM). GQM su osmislili Basili, Caldiera i Rombach i predstavili u radu [BCR94]. GQM (Goal/ Question/ Metric) model ima tri nivoa [BCR94]:

1. Konceptualni nivo (GOAL): Cilj se definiše za objekt iz različitih razloga, vodeći računa o različitim modelima kvaliteta i različitim tačkama gledišta. Objekti mjerenja su [BCR94]:

- proizvodi koji nastaju u životnom ciklusu softvera npr. : specifikacije, projektovanje, programi, testni podaci.,
- procesi koji obuhvataju aktivnosti vezane za vrijeme npr.: specifikovanje, projektovanje, testiranje, intervjuisanje,
- resursi upotrebljeni u procesu da bi se dobio proizvod npr.: personal, hardver, softver, radni prostor.

2. Operativni nivo (QUESTION): Niz pitanja koja se upotrebljavaju da specificiraju put za postizanje željenog cilja. Pitanja pokušavaju da karakterišu objekte merenja (proizvodi, procesi, resursi) sa respektom na selektirani nivo izlazećeg kvaliteta sa različitih tačaka gledišta [BCR94].

3. Kvantitativni nivo (METRIC): Niz podataka pridruženih svakom pitanju da bi se dobio odgovor u kvantitativnoj formi. Podaci mogu biti objektivni i subjektivni. Objektivni podaci zavise samo od objekta mjerenja ali ne i od tačke gledišta za razliku od subjektivnih koji zavise i od jedno i od drugog [BCR94].

Tabela 4. Primjer primjene GQM paradigme za definisanje opštih ciljeva kvaliteta projektovanja skladišta podataka

Ciljevi	**Pitanja**	**Metrika**
Smanjivanje rokova	Da li su zadovoljeni rokovi izrade prototipa skladišta podataka?	Vrijeme od početka projekta do funkcionalnosti prototipa
	Da li su zadovoljeni rokovi izrade konačne verzije skladišta podataka?	Vrijeme od početka projekta do završetka projekta
	Da li su zadovoljeni rokovi izmjene skladišta podataka u procesu održavanja na osnovu novih zahtjeva korisnika?	Vrijeme od usaglašavanja zahtjeva do implementacije izmjene skladišta podataka
Smanjivanje troškova	Koliko je osoba potrebno za realizaciju kompletnog skladišta podataka?	Broj čovjek/dana
	Koliki su troškovi razvoja skladišta podataka?	Cijena skladišta podataka po isporuci
	Koliki su troškovi održavanja skladišta podataka?	Mjesečna cijena održavanja
Povećanje zadovoljstva korisnika	Da li su korisnici više zainteresovani za saradnju i davanje zahtjeva nakon isporuke prototipa skladišta podataka?	Anketa o zadovoljstvu korisnika nakon isporuke i korišćenja prototipa skladišta podataka
	Koliki je broj grešaka nakon implementacije kompletnog skladišta podataka?	Broj zahtjeva za korektivno održavanje
	Koliko je vremena potrebno za realizaciju izmjena u procesu održavanja skladišta podataka?	Prosječno vrijeme rješavanja zahtjeva za izmjene skladišta podataka

Primjenom GQM paradigme mogu se odrediti osnovni ciljevi poboljšanja kvaliteta projektovanja, razvoja ili održavanja skladišta podataka, pitanja koja se postavljaju za ispunjenje navedenih ciljeva i metrika kojom se mjeri ispunjenost ciljeva projekta skladišta podataka.

Prema [Lin3] Data Vault metodologija se zasniva na CMMI standardu level 5 najboljih praksi. To uključuje više komponenti CMMI koje se kombinuju sa najboljim praksama iz Six Sigma standarda i TQM trendovima. Posebno je fokusiran na metodologije agilnog razvoja softvera. Data Vault projekti imaju kratak ciklus gdje se novi proizvod implementira svake dvije do tri sedmice.

2.10. Najznačajniji dobavljači u oblasti skladišta podataka

Velike svjetske IT kompanije prepoznale su važnost primjene koncepta poslovne inteligencije i skladišta podataka koji pružaju podršku u procesu donošenja poslovnih odluka. Koliko su kompanije iz globalnih tržišta prepoznali važnost poslovne inteligencije najbolje svjedoče rezultati istraživanja Europske unije kojim je utvrđeno da 82% evropskih kompanija s prihodom većim od 10 milijuna USD imaju interne specijalizovane odjele poslovne inteligencije radi prikupljanja i obrade relevantnih podataka o konkurenciji [IDC11]. Prema istraživanju časopisa Business Week iz 2011. godine preduzeća koja imaju model poslovne inteligencije ostvaruju i 20 posto veće prihode od konkurencije. Globalna istraživanja koja je provela istraživačka agencija Gartner pokazuju da je više od 40% investicija u informatičku tehnologiju 2012. god. bilo usmereno u rešenja za skladište podataka i poslovnu inteligenciju. Skladište podataka na tržištu prolazi kroz transformaciju s uvođenjem Big Data koncepta i logičkih skladišta podataka te potražnjom za novim tehnikama i tehnologijama u praksi. Jedan od trendova koji izaziva najveći interes je Big Data, koji prema Gartneru podrazumijeva informacioni resurs velike količine, velike brzine i velike raznovrsnosti podataka koji zahtijeva nove i inovativne metode obrade i optimizacije informacija, poboljšanje uvida u sadržaj podataka i donošenja odluka. Skladište podataka može biti bilo koje veličine.

Dimenzionisanje tradicionalnih skladišta se radi prema veličini podataka [Gar13]:

- Mala skladišta podataka su manje od 5 TB.
- Srednje velika skladišta podataka su od 5 TB do 20 TB.
- Velika skladište podataka su veća od 20 TB

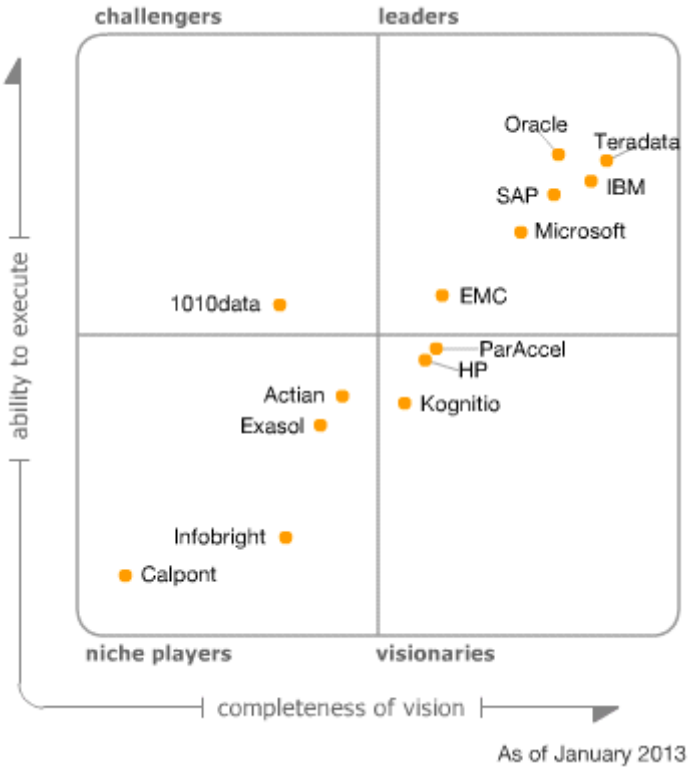
Prema izvještaju Gartnera od januara 2013. godine [Gar13], lideri na tržištu skladišta podataka su kompanije: Teradata, Oracle, IBM, EMC, SAP i Microsoft.

Teradata [Tera] ima 30-godišnje iskustvo na tržištu skladišta podataka obezbeđujući kombinaciju hardvera i posebnih analitičkih baza podataka koje pomažu kompanijama da objedine i analiziraju velike količine podataka u cilju podrške poslovnom odlučivanju

Oracle [Ora1] nudi izbor proizvoda koji omogućuju korisnicima izgradnju skladište podataka, koristeći različite konfiguracije ili uređaje za skladištenje podataka. Osim DBMS i konfiguracija, Oracle nudi tri različite Exadata branded proizvode: Oracle Exadata X2-2 za skladištenje podataka, Oracle Exadata X2-8 za cloud rješenja i Oracle Exadata Storage stalak za proširenje X2-2 za dodatnim kapacitetom pohrane.

IBM [IBM] nudi samostalna DBMS rješenja, kao i rješenja za skladištenje podataka. IBM-ov softver za skladište podataka, InfoSphere skladište, dostupan je na Unix, Linux i Windows operativni sistem. IBM je nastavio istraživanje i razvoj u oblasti skladišta podatak. IBM skladište podataka se zasniva na najboljim svjetskim praksama.

SAP kompanija [SAP] u oblasti skladišta podataka nudi proizvod SAP-In-Memory Data Fabric Solutions koje omogućuje brzu i jednostavnu analizu podataka u realnom vremenu, u kontekstu podrške poslovnom odlučivanju u svim dijelovima kompanije. SAP rješenje pojednostavljuje i ubrzava analizu velikih količina podataka.



Slika 20. Gartner-ov Magic Quadrant za januar 2013. god. [Gar13]

Microsoft [MSClo] na tržištu skladišta podataka i dalje nudi svoj SQL Server (trenutno je aktualna verzija 2014). Business Data Warehouse i Fast Track Data Warehouse za skladištenje podataka kupaca koji ne uključuje MPP DBMS. Microsoft je svoje rješenje za MPP skladištenje podataka objavio još u verziji SQL Server 2008 R2 kao Parallel Data Warehouse (PDW), Verzija SQL Servera 2014 donosi nova unapređenja u oblasti skladišta podataka. Microsoft nudi rješenja za moderna skladišta podataka zasnovana na konceptu Big Data

kombinujući tradicionalne velike količine podataka smještene u realcionim bazama podataka kao i nestruktuirane podataka kao što su Hadoop.

EMC [EMC] nudi proizvode EMC grupe sa uređajima za masivno paralelno procesiranje (MPP). Skladište podataka DBMS može da radi na Linux-u i Unix-u. EMC kroz sisteme za virtuelizaciju nudi rješenja za moderna skladišta podataka zasnovana Big Data konceptu kombinujući realcione kao i Hadop nestruktuirane podataka.

Prema Gartneru [Gar13], kao dio odgovora na dinamično ekonomsko okruženje, skladište podataka je postalo ključni element u upravljanje informacijama. Organizacije očekuju stabilne arhitekture i implementacija od dobavljača DW sistema. U 2013. godini većina velikih dobavljača je i dalje zadržala većinu udjela na tržištu. Tržište platforme DW sistema u 2013. bio je karakteristično pojačanim interesom poslovnih korisnika za jednostavniju upotrebu alata poslovne inteligencije. Jednostavnost korišćenja kao kriterijum kupovine sada nadmašuje funkcionalnost po prvi put. Pretpostavka Gartner-ovih analitičara je da će kupci tehnologije skladišta podataka i sa njim povezanog sistema poslovne inteligencije postati više konzervativni te se sve više okretati velikim i poznatijim firmama ponuđačima, što će manje kompanije staviti u nepovoljni položaj [Gar13].

3. PREGLED ISTRAŽIVANJA U OBLASTI AUTOMATIZACIJE PROJEKTOVANJA SKLADIŠTA PODATAKA

Primarni radovi u oblasti automatizacije projektovanja DW, su izdvojeni na osnovu doprinosa teoriji i praksi projektovanja skladišta podataka i broja citata prema [Harz] koji koristi Google scholar bazu radova [SchGoo].

3.1. Pregled teorijskih istraživanja u oblasti automatizacije DW dizajna

Golfarelli i drugi u radu [GMR98] predlažu konceptualni model skladišta podataka i poluautomatski sekvencijalni hibridni pristup koji polazi od E/R (*Entity/Relationship*) modela. Nakon analize šeme i izrade konceptualnog/logičkog modela predlaže se identifikacija činjenica (*fact*) koju slijedi poluautomatsko formiranje stabala atributa u skladu sa zahtjevima. Sledeći korak je identifikacija dimenzija, mjera i agregatnih funkcija nakon čega se izvodi logička i fizička šema baze skladišta podataka. Unapređenje postupka je dato u [GR09].

Boehnlein i Ulbrich u radu [BU99] primjenjuju hibridni pristup u izradi konceptualnog višedimenzionalnog skladišta podataka na osnovu transakcionog normalizovanog sistema prema Structured Entity Relationship Model-u (SERM). Kod ovog pristupa postoji predproces gdje se ER dijagram pretvara u SERM dijagram.

Phipps i Davis u radu [PD02] kao polazište uzimaju data-driven pristup, a kasnije prelazi na demand-driven (sekvencijalni hibridni pristup). Polazna šema se daje korištenjem Multidimesional E/R modela. Ovaj rad predlaže algoritam

za dobijanje konceptualne sheme od transakcione šeme. Algoritam koristi numerička polja i odnose između entiteta kao temelj za stvaranje Multidimensional E/R šeme. Algoritam se primjenjuje za svaki model podataka gdje se podaci mogu razdjeliti u numeričke, datum/vrijeme i tekstualne tipove podataka. Osim toga, ovaj pristup se može koristiti za procjenu kandidata za konceptualne šeme pomoću korisničkih upita.

Peralta i drugi u radu [PMR03] idu korak naprijed ka automatizaciji skladišta podataka baziranom na pravilima primjenom postojećih znanja u dizajnu skladišta podataka. Predložena pravila se ugrađuju u dizajn strategiju koja se pokreće u skladu sa zahtjevima i izvodom podataka, te na osnovu šeme transakcione baze. Okvir se sastoji od: pravila, smjernice za dizajn koje sadrže nefunkcionalne zahtjeve, mapiranja specifikacija koje se odnose konceptualne šeme sa šemom izvorne baza podataka, te niz šema transformacija. Ovaj koncept omogućuje primjenu postojećih dizajn tehnike, te poboljšanu produktivnost i jednostavnost.

Romero i Abelo u radu [RA07] predlažu poluautomatsku metodu prepoznavanja poslovnih višedimenzionalnih koncepata iz ontologije domen-a koji predstavljaju različite i potencijalno heterogene podataka. Uz korišćenje ontologije i alate koje ona koristi u potrazi za višedimenzionalnim uzorcima. Provedena je simulacija stvarnog slučaja za provjeru algoritam, a predstavljena je i teoretska analiza algoritma. Ovaj pristup je u stanju integrisati podatke iz heterogenih izvora koji opisuju njihove domene kroz ontologije. Jedan od najperspektivnijih područja gdje se metoda može primijeniti je semantički Web, što doprinosi vađenju i integrisanju spoljnih podataka sa Web-a.

Zepeda i drugi u radu [ZC10] predstavljaju poluautomatski proces za izgradnju DW zasnovan na MDA. Polazi se od prikupljanja i konsolidovanja zahtjeva korisnika. Ostali koraci obuhvataju prikupljanje formalnih i strukturnih informacija iz šeme izvorne baze podataka podržani od strane neke automatizacione tehnike, a u radu se koristi alat koji usmjerava dizajnera prema efikasnom rješenju u skladu sa zahtjevima korisnika.

Nazri i drugi u radu [NNH10] predlažu koncept koji se zasniva na modelu izvora podataka. Predlaže se poluautomatski alat koji koristi leksičke ontologije kao bazu znanja. Nakon što se identifikuju činjenice i dimenzije, vrši se generisanje višedimenzionalnog modela sa minimalnim učešćem korisnika. je Metoda je ilustrovana koristeći WordNet semantički leksikon kao znanje domena, pri identifikaciji mjera i dimenzija.

Tabela 5. Poređenje pristupa automatizacije DW dizajna [KJM14]

Pristup	Generisanje			Izvori podataka		Korišćenje pravila
	Koncept. DW model	Fizički DW model	Data Vault	Struktuirani	Polu-struktuirani	
[GMR98]	X	X		X		
[BU99]	X			X		
[PD02]	X			X		
[PMR03]	X			X		X
[RA07]	X	X		X	X	
[ZC10]	X			X		
[ZMA11]	X	X		X		
[NNH10]		X		X	X	
Direct Physical DV		X	X	X	X	X

Zekri i drugi, u radu [ZMA11] se uvode pristup za poluautomatsko oblikovanje multidimenzionalanog dijagrama. Conceptual Data Model (CDM) se najprije

prevodi na multivarijantni patern, a zatim se rafinira pomoću zahtjeva korisnika. Oba koraka koriste grafove kao formalizam za predstavljanje modela podataka za donošenje odluka.

3.2. Pregled realizovanih alata u oblasti automatizacije DW dizajna

Prethodno navedeni radovi u ovom poglavlju, uglavnom predstavljaju akademski doprinos. Od interesa za praktične realizacije u nastavku je dato nekoliko najznačajnijih istraživanja koja ilustruju razvoj i/ili korišćenje alata za automatizaciju dizajna skladišta podataka .

WAND alat predstavlja prototip CASE koji pomaže dizajneru u strukturiranju data marta, konceptualnom dizajnu na poluautomatski način, te definisanje konceptualne sheme i vrši logički dizajn za izradu data mart šeme [WAND].

QBX je CASE alat za dizajn data marta koji je rezultat je bliske saradnje između akademske zajednice i industrije. On podržava proces dizajna tokom idejnog rješenja, logičkog dizajna, i razvoj ROLAP data marta u obliku zvijezda ili pahulja šeme. Alat može koristiti poslovnim korisnicima da interaktivno istraživati znanja vezanih za projekat na različitim nivoima apstrakcije [QBX].

BIReady je alat za model-driven razvoj skladišta podataka. Ovaj alat je produžetak CA Erwin alata koji koristi najbolje prakse i stabilnu arhitekturu, tako da se s njim može upravljati skladištem i razvijati skladišta podataka. BIReady omogućuje automatsko generisanje skladišta podataka iz poslovnih modela. BIReady generiše i ETL procedure na osnovu najbolje prakse kao što su Corporate Information Factory (CIF), Star schema i Data Vault model [BIR].

Birst [Birst] alat automatski kreira logički dimenzioni model u star shemu. Logičke mjere se automatski analiziraju, dok se logičke dimenzije analiziraju na osnovu zahtjeva korisnika. Alat upravlja svim ključnim procesima, uključujući i surogat ključeve gdje je to potrebno, omogućuje povezanost podataka i ekstrakciju iz širokog spektra baza podataka. Birst takođe podržava rad sa svim većim relacionim sistemima za upravljanje bazama podataka, uključujući Oracle, DB2, SQLServer, MySQL, Sybase, i PostgreSQL.

Quipu [Quipu] je open source sistem za podršku projektovanja i održavanja skladišta podataka koji podržava i Data Vault arhitekture. Quipu automatizuje dizajn skladišta podataka i generiše ETL kod za popunjavanje skladišta podataka iz izvora podataka. Sa Quipu se mogu jednostavno i brzo kreirati i implementirati skladišta podataka. Prema dokumentaciji koja je dostupna na [Quipu], funkcionalnost za podršku izgradnje data marta još nije dostupna u verziji Quipu 1.1.

Pentaho Data Integration (PDI, još se naziva Kettle) [CBD10] je komponenta čija namjena je podršaka za ETL procese. Iako se ETL alati najčešće koriste u okruženjima skladišta podataka, PDI se može koristiti i za druge svrhe: migracija podataka između aplikacija ili baza podataka, izvoz podataka iz baze podataka u fajlove, masovno učitavanje podataka u baze podataka, čišćenje i integracija aplikacija. PDI je jednostavan za korišćenje i može se koristiti kao samostalna aplikacija, ili kao dio Pentaho Suite. Kao ETL alat, to je najpopularniji dostupan open source alat. Osim toga, mogućnosti transformacija PDI omogućuju manipulaciju podacima s vrlo malo ograničenja [CBD10]. Data Vault koncept i procesi nisu podržani ovim alatom [Pent].

Tabela 6. Poređenje alata za automatizaciju DW dizajna [KJM14]

DW Tool	Type of DW tools	Physical DW model	Data Vault Design	Generate ETL Code	Data Mart Design
WAND	Academic	Automatic	Not supported	Automatic	Automatic
QBX	Academic	Automatic	Not supported	Automatic	Automatic
Quipu	Open source	Automatic	Automatic	Automatic	Manual
Pentaho Kettle	Open source	Manual	Manual	Automatic	Manual
BIReady	Commercial	Automatic	Manual	Manual	Automatic
Birst	Commercial	Automatic	Not supported	Automatic	Automatic
Direct PDV	Prototype	Automatic	Automatic	Automatic	Automatic

3.3. Kritički osvrt na dosadašnja istraživanja u oblasti automatizacije dizajna skladišta podataka

Automatizacija projektovanja nije jednostavan proces, jer pored identifikacije šeme izvorne baze podataka treba formalizovati zahtjeve krajnjih korisnika. Određeni koraci ipak moraju biti ručno izvedeni, na primjer identifikacija poslovnih ključeva i poslovnih mjera.

Tabelom 5 dat je uporedni prikaz različitih pristupa uključujući predloženi na bazi Data Vaulta. U pristupu koji se predlaže u ovoj disertaciji (Direct PDV) predviđeno je automatsko generisanja fizičkog modela Data Vault-a na osnovu šeme metapodataka transakcionih struktuiranih izvora podataka (baze podataka) uzimajući (u određenoj mjeri) u obzir i polustruktuirane i nestruktuirane izvore. Pored toga, u odnosu na alternative (Tabela 4), direktni pristup automatizaciji fizičkog projektovanja skladišta realizuje se kroz korišćenje pravila.

U literaturi koja se odnosi na dizajn Data Vault sistema [Lin1, Lin2, Lin3, Dam08, HS10, Gra11, CBD10] nije prikazan konkretan postupak automatizacije dizajna skladišta podataka niti formalizacija automatizacije skladišta podataka baziranog na Data Vault pristupu. Treba imati u vidu da je većina pristupa uglavnom akademska, i da samo [GR09] ima dužu tradiciju industrijske upotrebe što je odlika i Data Valut pristupa [Lin1, Lin2, Dam08].

Tabelom 6 dat je uporedni prikaz različitih alata u oblasti automatizacije dizajna skladišta podataka uključujući predloženi na bazi Data Vaulta. Osim WAND alata, ne postoje javno dostupne informacije o formalizaciji pristupa kod realizacije alata. Između ostalog navedenog u tabeli 5, prednost predloženog alata Direct PDV u odnosu na postojeća rješenja je mogućnost grafičkog pregleda fizičkog dijagrama Data Vaulta i data marta tokom procesa dizajna skladišta podataka i procesa dizajna data marta.

4. MOGUĆNOST AUTOMATIZACIJE FIZIČKOG PROJEKTOVANJA SKLADIŠTA PODATAKA (DIREKTNI PRISTUP PROJEKTOVANJA SKLADIŠTA PODATAKA)

Organizacije izgrađuju skladišta podataka sa različitim metodološkim pristupima. Praktičan problem za sve organizacije je što proces projektovanja skladišta podataka nije u dovoljnoj mjeri automatizovan, odnosno proces automatskog generisanja i prepoznavanja mjera i dimenzija, odnosno hub, link i satelit entiteta Data Vault skladišta podataka.

Fizička dizajn, čiji rezultat je fizički model podataka, koji je na najnižem nivou apstrakcije u odnosu logički i konceptualni model, ima za cilj prilagođenja logičkog prikaza baze podataka za primjenu kod pojedinih platformi za upravljanje bazama podataka. Podaci u bazama podataka imaju jasnu strukturu sa tabelama, kolonama, indeksima i dr. To mogu biti relacione baze podataka (engl. *Relational Database Management System*, RDBMS), objektno-orijentisane baze podataka (klase i vrste) ili hijerarhijske baze podataka (stablo - struktura) [Rai08]. Šema podataka predstavlja metapodatke, odnosno podatke o podacima. Za upravljanje podacima u RDBMS, kao što je poznato, se koristi strukturni upitni jezik SQL (engl. *Structured Query Language*). SQL je jezik koji ima uporište u relacionoj algebri i/ili relacionom računu i posjeduje upitne jezike sa konstrukcijama koje su mnogo bliže korisniku nego konstrukcije same relacione algebre, odnosno računa [LMAB08]. Pojava komercijalnih sistema za upravljanje bazama podataka koje su prihvatile SQL jezik doprinijela je standardizaciji relacionog upitnog jezika preko standarda koji su objavile međunarodne agencije za standardizaciju ANSI i ISO. Iskazi (engl. *statements*) SQL jezika se mogu podijeliti u četiri kategorije [GJ00]:

- SQL iskazi koji omogućuju pretraživanje podataka u bazi podataka sa ključnom riječi SELECT
- SQL iskazi koji se koriste za promjenu podataka u bazi (engl. *Data Manipulation Language – DML*)
- SQL iskazi koji se koriste definisanje, kreiranje ili brisanje objekata baze podataka (engl. *Data Definition Language – DDL*)
- SQL iskazi koji služe za kontrolu pristupa bazi podataka (engl. *Data Control Language - DCL*)

U narednim razmatranjima SQL iskaz, koji sadrži SQL komande, je označen kao sql_statement.

U ovom poglavlju se ispituje mogućnost automatizacije:

- Fizičkog dizajna skladišta podataka baziranog na Data Vault konceptu, odnosno automatskog kreiranje hub, link i satelit entiteta na osnovu dizajna transakcione baze podataka,
- Fizičkog dizajna data martova, odnosno automatskog kreiranja fact i dimenzionih tabela iz Data Vault skladišta podataka
- ETL/ELT procesa, odnosno ekstrakcije, transformacije i učitavanja podataka u skladište podataka, te učitavanja podataka iz skladišta podataka u data martove u gornjem kontekstu, odnosno DV pristupu (engl. *pattern based ETL/ELT code generation*)

U procesu skladištenja podataka zahvatamo veliku količinu podataka koji mogu biti smješteni u različitim izvorima: ERP (engl. *Enterprise Resource Planning*) sistemi, relacione baze podataka, Excel fajlovi, DBF fajlovi, TXT i XML fajlovi ili podaci sa WEB-a. Osim toga, podaci u relacionim bazama podataka mogu biti smješteni u različite sisteme za upravljanje bazama podataka kao što su MS SQL Server, Oracle, DB2 i dr. Prema Linstedt-u, skladište podataka koje

je zasnovano na Data Vault sistemu, s obzirom na njegov koncept, najviše ima smisla koristiti u slučaju više izvora podataka.

Podaci iz ERP sistema su smješteni u baze podataka. International Data Corporation (www.idc.com), jedna od vodećih svjetskih kuća u području praćenja i analize informacionih tehnologija, ERP softver definiše kao programsku podršku za najmanje 3 od 4 segmenta poslovanja:

- financijsko poslovanje (engl. *accounting*),
- proizvodnja (engl. *manufacturing*),
- robno-materijalno poslovanje (engl. *material management/distribution*),
- upravljanje ljudskim resursima i obračun zarada (engl. *HR management, payroll*).

Iako kompanije nastoje integrisati sve svoje podatke prilikom implementacije ERP sistema, često to nije moguće praktično izvesti, zbog čega se prilikom izgradnje skladišta podataka pojavljuju i drugi izvori podataka izvan ERP sistema. Tako naprimjer u trgovini, sistem za upravljanje odnosima sa kupcima (CRM, *Customer Relationship Management*) i sistem za upravljanje lancem snabdjevanja (SCM, *Supply Chain Management*) često nisu integrisani sa ERP sistemom.

Podaci koji se nalaze u fajl sistemima odlikuju se nestruktuiranošću. Nestruktuirani podaci postoje u dva osnovna oblika: tekstualni i netekstualni. Tekstualni nestruktuirani podaci se javljaju u mnogim mjestima kao poruke e-pošte, telefonskih razgovora, PowerPoint prezentacije, i sl. Netekstualni nestrukturirani podataci se pojavljuje kao grafike, slike, dijagrami i ilustracije i dr. [ISN08]. Postojeća tehnologija omogućava lako učitavanje tekstualnih podataka u prelazne baze podataka (engl. *staging*) ili skladišta podataka, dok to

nije slučaj sa netekstualnim nestruktuiranim podacima. Tekstualni nestruktuirani podaci mogu u određenoj mjeri biti struktuirani i tada ih nazivamo polustruktuirani podaci. Tekstualni nestruktuirani podaci imaju tekst koji je napisan na slobodan način (dopis, izvještaj, zahtjev, priručnik i sl.), dok tekstualni polustruktuirani podaci imaju ponavljajući format koji omogućava lakše učitavanje u bazu podataka. Naprimjer, recept za lijekove u sistemu zdravstva ima dijelove i tekst koji se ponavlja.

Danas postoji velika količina podataka, za potrebe odlučivanja, koja je spremljena u XML (*Extensible Markup Language*) formatu podataka. Struktura XML, sastoji se od ugnježdenih definisanih tagova koji može opisati značenje samog sadržaja, koji omogućuje razmjenu podataka na webu. Kako je Internet je prerastao u globalnu platforme za e-trgovinu i razmjene informacija, korišćenje XML-a je u porastu, te velike količine podataka u XML formatu već postoje [GRV01]. XML se smatra određenom standardnom sintaksom za razmjena polustrukturiranih podataka [ABS00]. Rad [GRV01] se bavi dizajnom Data Mart-a iz XML izvora podataka. Glavni problem sa ručnim dizajnom Data Marta iz XML izvora predstavlja činjenica da zahtjevi koji izražavaju korisnici ne mogu u potpunosti biti podržani od postojećih podataka, i što proces mapiranje zahtjeva iz izvornih šema može biti vrlo složen. Predlaže se poluautomatski pristup za dizajn Data Mart-a iz XML izvora podataka. Pokazano je kako polustruktuirani izvori podataka povećavaju nivo nesigurnosti na strukturu podataka u odnosu na struktuirane izvore kao što su baze podataka. U [VBR03] je opisan pristup za dizajn web skladišta počevši od XML šeme koja opisuje operativnu izvor. Pristup se provodi u Java-based prototipu. Budući da sve potrebne informacije ne mogu zaključiti iz XML sheme, u nekim slučajevima koristi se XQuery upitni jezik. Prototip

automatizuje nekoliko dijelova procesa dizajna: pretprocesiranje XML scheme, kreiranje i transformaciju šeme grafa, pretraživanje XML dokumenata.

S obzirom na mogućnost postojanje više struktuiranih i nestruktuiranih izvora podataka, kod izgradnje Data Vault sistema neophodno je napraviti analizu svih izvora podataka. Kao rezultat ovakve analize može se pojaviti nekoliko najznačajnijih slučajeva:

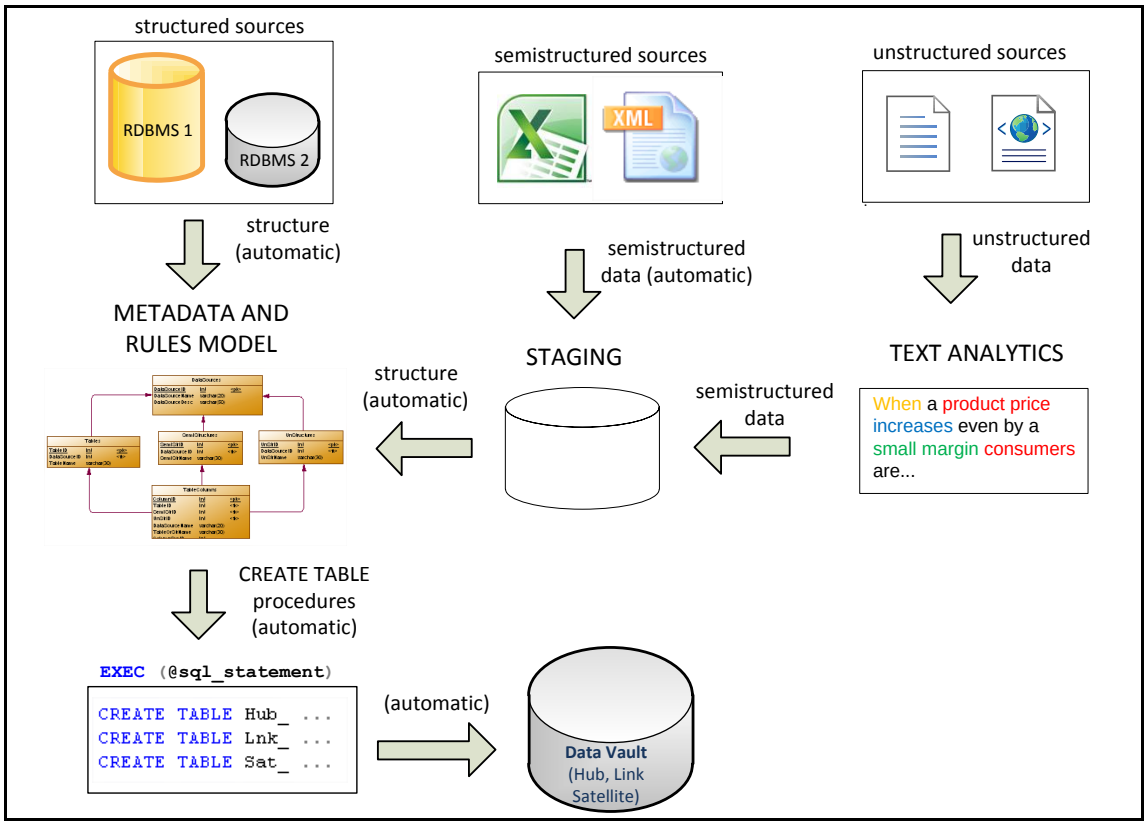
1. Svi relevantni podaci potrebni za skladište podataka smješteni su u jednom ERP sistemu ili jednoj i više transakcionih baza podataka od istog dobavljača za sisteme za upravljanje bazama podataka (npr. MS SQL Server 2000, 2005, 2008 ili 2012)
2. Svi relevantni podaci potrebni za skladište podataka smješteni su u jednoj ili više transakcionih baza podataka od različitih dobavljača za sisteme za upravljanje bazama podataka (npr. MS SQL Server, Oracle, DB2 i dr).
3. Pored podataka u bazama podataka, postoje i podaci u polustruktuiranim izvorima podataka (npr. Excel, XML)
4. Pored podataka u bazama podataka i polustruktuiranim izvorima podataka, postoje i podaci u nestruktuiranim izvrima podataka (npr. TXT)

4.1. Prijedlog originalnog pristupa za automatizaciju projektovanja skladišta podataka baziranog na Data Vault konceptu

Kod izgradnje skladišta podataka, idealna situacija je kad su svi podaci smešteni u bazama podataka, naročito ako je to na istom sistemu za upravljanje bazama podataka. Međutim, u praksi je to najrijeđi slučaj tako da je potrebno u skladište podataka integrisati podatke iz različitih baza podataka i različitih izvora podataka. U ovakvoj situaciji, izvođenje projekta izgradnje skladišta podataka može angažovati značajne ljudske resurse s obzirom da u timu

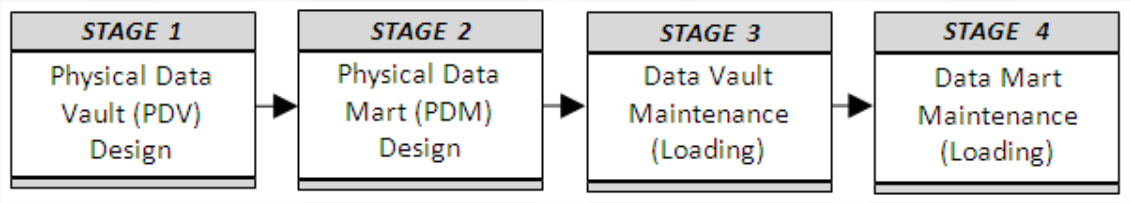
moraju postojati stručnjaci različitih znanja kao što su: administratori različitih sistema za upravljanje bazama podataka i programeri sa poznavanjem bar jednog programskog jezika (radi kreiranja objekata za konektovanje na različite baze podataka preko direktne konekcije, ODBC-a, i sl.). Osim toga, potrebna su znanja i za import polustruktuiranih i nestruktuiranih podataka u prelaznu bazu podataka. Bez obzira o kojem se od navedenih slučajeva radi, zbog dobijanja informacija o metapodacima iz različitih izvora potrebno je omogućiti pristup svim izvorima podataka koji se trebaju učitati u skladište podataka. Kod Data Vault sistema neophodno je zadržati sledljivost podataka, zbog čega ne treba raditi spajanje podataka iz različitih izvora prije učitavanja u skladište podataka. Integracija podataka će se uraditi u Data Vault sistemu preko poslovnih ključeva (ukoliko ih je moguće identifikovati u svakom izvoru podataka), što znači da će se za svaki izvor podataka, koji se tiče posmatranog poslovnog ključa, kreirati po jedna satelit tabela.

U disertaciji se predlaže direktan pristup projektovanju fizičke strukture Data Vault skladišta podataka na bazi izvornih podataka. Važan metodološki doprinos ovog pristupa je da se najprije konceptualno svi tipovi izvora podataka, na nivou metapodataka, svode na relacioni model. Osnovna ideja predloženog pristupa je data na sljedećoj slici.



Slika 21. Osnovna ideja predloženog pristupa [KJM14]

Relacioni model je zajednički činilac i za DV implementaciju i za izvorne podatke i zato se svi tipovi izvora (struktuirani i polustruktuirani) najprije abstrahuju metmodelom. Naziv „Direktni pristup“ upravo znači da sa se u predloženom pristupu direktno dolazi do fizičkog modela Data Vaulta – PDV (eng. *Physical Data Vault*) na osnovu fizičkog modela izvora u relacionom obliku. Pristup olakšava činjenica da, bez obzira u koliko izvora se nalaze operativni podaci, svaki poslovni sistem teži da integriše, barem konceptulno, osnovne podatke od šireg značaja i uspostavi kontrolu, odnosno da ima jednu primarnu bazu podataka u kojoj je smješten najveći dio podataka (a ta baza je u najvećem broju slučajeva relaciona). Sa druge strane teorijska analiza (Date, Darwen, Third manifesto 1995.) posmatra relacioni model kao generalan data model, na koji se svi ostali mogu svesti [DD95]. Predloženi pristup podrazumijeva četiri osnovne faze realizacije sistema kao što je prikazano na sljedećoj slici.

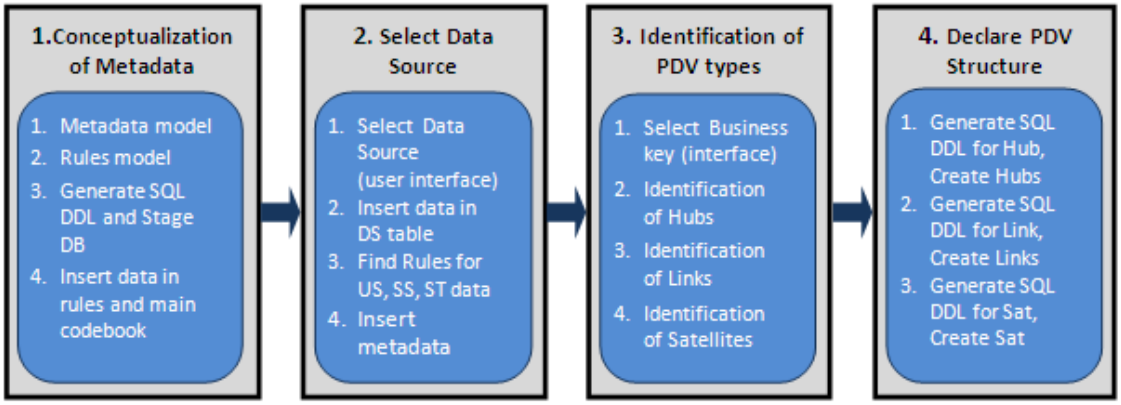


Slika 22. Faze automatizacije skladišta podataka [KJM14]

U prvoj fazi se kreira skladište podataka bazirano na Data Vault sistemu, dok se drugoj fazi projektuju data martovi. Treća faza podrazumijeva učitavanje podataka u Data Vault skladište podataka iz izvora podataka, dok se u četvrtoj fazi podaci iz skladišta podataka učitavaju u data martove. U ovom poglavlju se predlaže pristup izgradnje skladišta podataka sa visokim stepenom automatizacije uz minimalno učešće korisnika.

4.2. Direktni pristup projektovanja fizičkog modela skladišta podataka

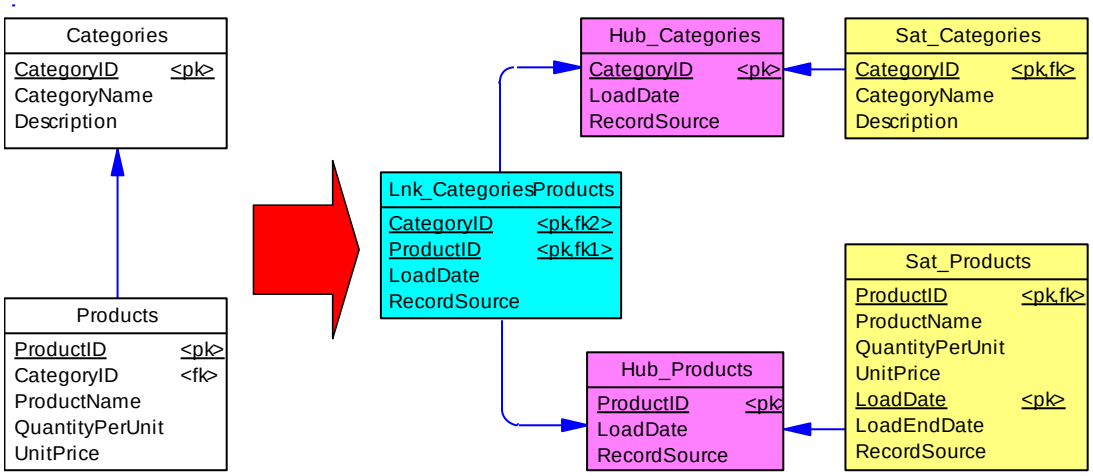
Prva faza se sastoji od četiri osnovne međufaze realizacije sistema.



Slika 23. Prva faza, automatizacija PDV [KJM14]

Direktni pristup automatizaciji fizičkog projektovanja skladišta podataka predstavljen u ovom radu podrazumijeva, između ostalog i korišćenje pravila. Neki od metapodataka su potrebni za identifikaciju veza odnosno stranih

ključeva, te će igrati ulogu u uspostavljanju pravila. Predloženi pristup će biti detaljno prezentiran sa metamodelom, algoritamima i pravilima. Na sljedećoj slici dat je jedan primjer mapiranja relacionog u Data Vault model.



Slika 24. Primjer mapiranja relacionog u Data Vault model

Da bi se razumjela detaljna pravila treba najprije analizirati generalni algoritam prepoznavanja hub, link i satelit relacija na osnovu tabela izvornih podataka. U najkraćem algoritam se sastoji od sljedećeg:

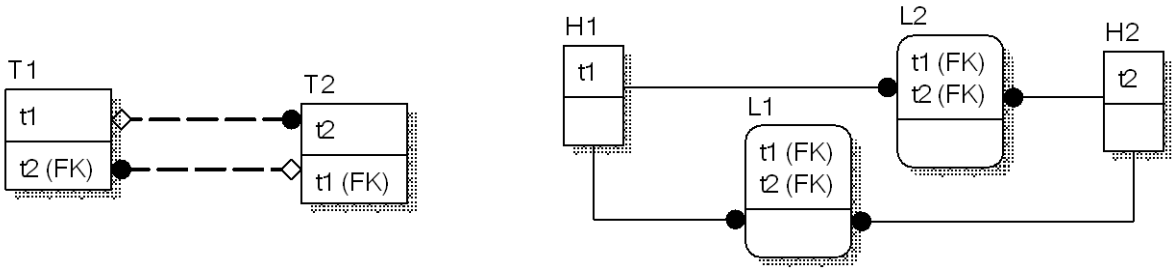
```
I- For each source:
     a) For each Table: the user confirms Hubs by
        selecting a Business Key
     b) For each Table, if not already selected as a Hub,
        create Link
     c) For each Table:
          i.for each FK create  Link
     d) for each non-key attribute: create Satellite
II-For each Hub (PK)
   a)
    i. For each Table (search for matching PK): if found,
        create Link (to Hub ad II; this only illustrate
        possible integration).
```

Efekat algoritma je kreiranje Link tabela za svaki strani ključ (osim za tabele čiji primarni ključevi se u potpunosti sastoje od stranih ključeva kao i za sve takve tabele koje će postati linkovi u koraku I-b ako ne postanu hubovi u koraku I-a). Dokaz o kompletnosti algoritma proizlazi iz pretpostavke da, bez obzira na

izvor, svaka tabela mora imati primarni ključ, kao i činjenica da su tabele prvobitno izgrađene na osnovu stvarnih podataka, tako da nema kontradikcije u pogledu cirkularne zavisnosti o drugim tablama koja bi mogla spriječiti njihovo formiranje. Kompletan algoritam se može predstaviti kako slijedi [KJM14]:

1. Every PK is either composed (partially or fully) from the PKs from other tables or is a separate PK (possibly, but rarely, composed of several fields concatenated or not. See [LGH08]).
2. Tables with separate PKs are designated as Hubs
3. Tables with fully composed PKs are designated as Links.
4. Remove all Links tables whose PK are not used as FK
5. Remove all Hubs tables whose PK are not part of any PK in any remaining table
6. Represent all remaining FK references with separate Links tables (each of those Links will now have two FK references); this is a crucial step separating mutually dependent tables
7. Represent what is left, treating it as a directed graph expressing FK references (as precedence relations) in a matrix form
    a. Nodes are tables (including Links created in step 6),
    b. Branches are FK references whose direction follows the FK reference i.e. branches are pointing to tables whose PK is referenced by each FK. The result is a connected directed graph.
8. We can fully reduce the matrix using Warshall's algorithm following precedence relations (as adjacencies) on a connected directed acyclic graph (in a finite number of steps) where the number of steps determines the length of the longest precedence chain.
9. The key for a proof that the above procedure finishes is that the graph obtained by carving initial, i.e. root, Hubs and Links must be acyclic (proof by contradiction)
    a. If the precedence (FK path) forms a cycle the table has a PK that depends on itself but this contradicts the initial requirement that tables can exists in extension, meaning that records in the tables exist i.e. not only can be formed with complete PK in the time of schema creation but also populated in the time of record creation).

Graf za ER model (ili njegova realizacija kao RM sa FK) generalno nije stablo, i može imati cikluse koji predstavljaju problem za realizaciju, dok se u Data Vault konceptu mogu realizovati samo dodavanjem linkova za svaki FK, kao u primjeru na sljedećoj slici.

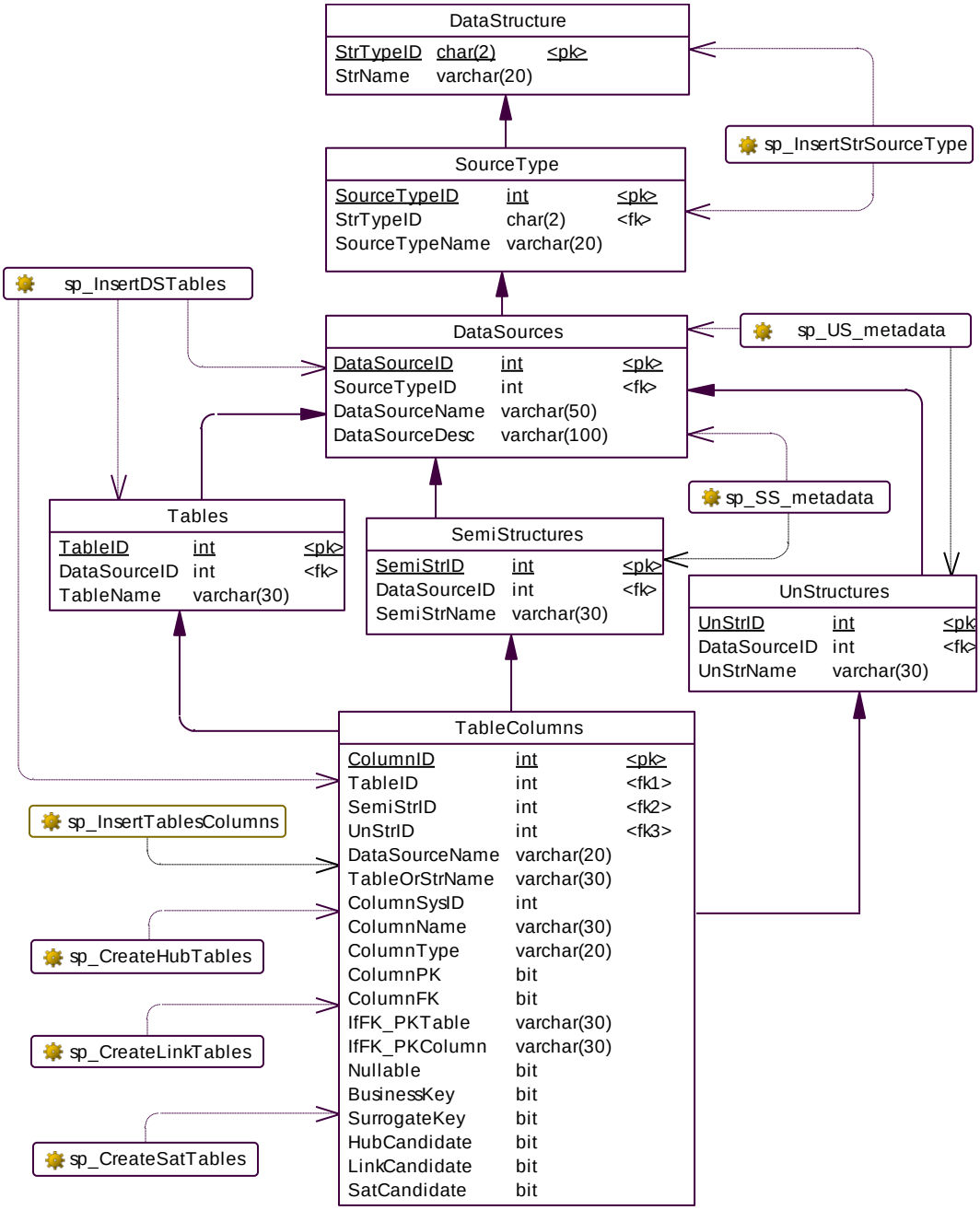


Slika 25. Dodavanje linka za svaki strani ključ

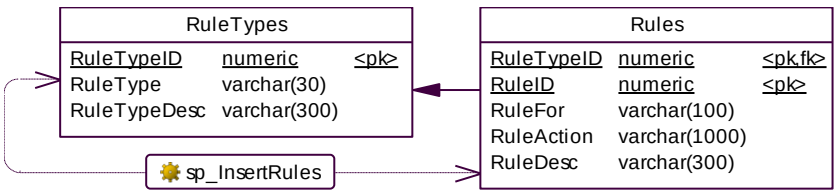
4.2.1. Konceptualizacija metapodataka

Osnovna međufaza (slika 23) je dakle **Konceptualizacija metapodataka** (engl. *Conceptualization of Metadata*) fizičkim modelom podataka koji se može koristiti u projektovanju individualnih skladišta podataka po DV konceptu.

Prva dva koraka ove međufaze podrazumjevaju: izradu modela metapodataka (slika 26) i izradu modela pravila (slika 27). Kod izrade modela metapodataka i modela pravila, razlikujemo dio modela koji je nezavisan od izvora podataka (engl. *source-idependent design*) i dio modela koji zavisi od izvora (engl. *source-specific design*). Nezavisni dio odnosi se na tabele metapodataka i pravila, a zavisni se odnosi na procedure vezane za različite izvore podataka. Namjena i način popunjavanja tabela metapodataka dat je u tabeli 7, dok je sadržaj tabela pravila dat u Dodatku A



Slika 26. Fizički model metapodataka



Slika 27. Model pravila

Modeli su urađeni korišćenjem alata Sybase Power Designer [SPD].

Tabela 7. Namjena i način popunjavanja generisanih tabela podacima

Tabela	Namjena	Način popunjavanja
DataStructure	Šifranik strukture podataka: struktuirani (ST), polustruktuirani (SS) ili nestuktuirani (US)	Pomoću skripta koji je generisan iz modela iz modela, npr: INSERT INTO DataStructure (StrTypeID, StrName) VALUES ('ST', 'Structured')
SourceType	Šifarnik izvora podataka (npr. MS SQL Server, Oracle, xls, XML, txt, i dr.)	Pomoću skripta koji je generisan iz modela iz modela, npr: INSERT INTO SourceType (SourceTypeID, StrTypeID, SourceTypeName) VALUES (1, 'ST', 'MS SQL Server')
DataSource	Izvori podataka koji će koristiti skladište podataka	Automatski, nakon identifikacije svih izvora podataka kroz korisnički interfejs.
Tables	Tabele iz operativnih baza podataka	Automatski, nakon identifikacije svih izvora podataka kroz korisnički interfejs
SemiStructures	Polustruktuirani izvori podataka (xls, XML, i dr.)	Poluautomatski, nakon importa i struktuiranja (koliko je moguće) u prelaznoj bazi podataka
UnStructures	Nestruktuirani izvori podataka (txt i dr.)	Poluautomatski, nakon tekstualne analize, importa i struktuiranja (koliko je moguće) u prelaznoj bazi podataka
TableColumns	Podaci o kolonama tabela iz baza podataka i drugih izvora koji se mogu struktuirati	Automatski, izuzev polja BusinessKey koji će se izabrati kroz korisnički interfejs
RuleTypes	Tipovi pravila za kreiranje skladišta podataka	Pomoću skripta koji je generisan iz modela
Rules	Sadrži pravila i odgovarajuće akcije ako je uslov ispunjen	Pomoću skripta koji je generisan iz modela

Tabela RuleTypes sadrži informaciju o tipovima pravila kao npr. pravila za različite izvore podataka, pravila za identifikaciju hub, link i satelit tabela, te pravila za kreiranje hub, link i satelit tabela. Tabela Rules sadrži informacije o pravilima za konkretan tip pravila i akcije koje će se izvršavati kad je određen uslov zadovoljen. Pravila u koloni Rules omogućavaju lakše izvršavanje komandi koje su spremljene u kolonu RuleAction.

Pravila su implementirana kroz korisnički definisanu funkciju koja prihvata parametar, izvršava odgovarajući SQL iskaz i vraća rezultat. Rezultat se vraća u obliku odgovarajućeg SQL iskaza ili naziva odgovarajuće uskladištene procedure. U nastavku je dat SQL kod korisnički definisane funkcije *udfGetRules*:

```
-- udfGetRules.sql
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[udfGetRules]') and xtype in (N'FN', N'IF',
N'TF'))
drop function [dbo].[udfGetRules]
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
CREATE FUNCTION udfGetRules (@RuleFor varchar(100))
RETURNS varchar (1000) AS
BEGIN
   DECLARE @RuleAction varchar (1000)
   SELECT @RuleAction = RuleAction
   FROM Rules WHERE RuleFor =  @RuleFor
   RETURN ISNULL (@RuleAction, '_')
END
```

Naprimjer za parametar *HubCandidate*, funkcija *udfGetRules* vraća rezultat u obliku SQL iskaza:

```
UPDATE TableColumns SET TableColumns.HubCandidate = 1
WHERE TableColumns.BusinessKey = 1
```

Prethodni primjer prikazuje pravilo koje opisuje skup ograničenja koja se primjenjuju za identifikaciju hub kandidata na osnovu vrijednosti kolone BusinessKey.

Treći korak u ovoj međufazi predstavlja generisanje koda za kreiranje prelazne baze podataka i kreiranje tabela i uskladištenih procedura iz modela sa slika 26 i 27. Iz predstavljenog fizičkog modela moguće je generisati skript za kreiranje tabela u prelaznoj bazi podataka i inicijalni insert podataka u tabele DataStructure i SourceType.

Četvrti korak je insert podataka u tabele pravila i osnovne šifarnike. U daljnim razmatranjima radi ilustracije praktičnih primjera koristiće se standardni SQL jezik čime se ne umanjuje opštost postupka. Insert podataka u osnovne šifarnike se radi pomoću uskladištene procedure *sp_insertStrSourceType* u tabele DataStructure i SourceType. Dio koda navedene procedure je prikazan u nastavku:

```
INSERT INTO DataStructure (StrTypeID, StrName) VALUES ('ST',
'Structured')
```

i analogno za druge izvore podataka

```
INSERT INTO SourceType (SourceTypeID, StrTypeID, SourceTypeName)
VALUES (1, 'ST', 'MS SQL Server)
```

i analogno za druge tipove izvora podataka. Inicijalni insert podataka u tabele pravila se radi pomoću uskladištene procedure iz modela *sp_InsertRules.*
Naprimjer, za struktuirane tipove pravila (analogno je za druge tipove pravila):

```
INSERT INTO RuleTypes (RuleTypeID, RuleType, RuleTypeDesc)
VALUES (1, 'Structured Source Data', 'Pravila za različite
izvore podataka kao npr. MS SQL Server, Oracle, IBM db2')
```

Nakon prve faze i kreiranja tabela i odgovarajućih procedura iz modela stvoreni su uslovi za izradu aplikacije koja omogućuje automatizaciju projektovanja fizičkog modela skladišta podataka baziranog na Data Vault konceptu, a uz minimalnu interakciju sa korisnikom. Drugu, treću i četvrtu međufazu sa slike 23 potrebno je uraditi za svaki konkretan sistem.

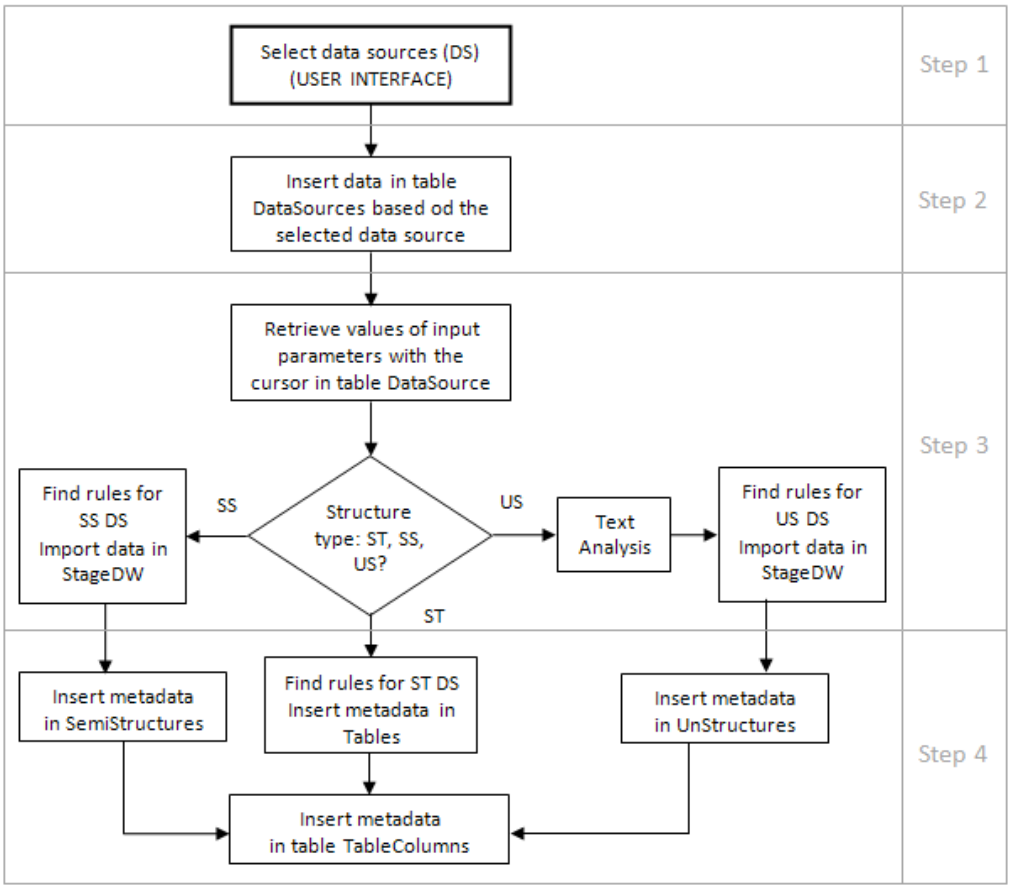
4.2.2. Identifikacija izvora podataka

Međufaza Identifikacija izvora podataka (na slici 23 označena kao *Select Data Sources*) se primjenjuje na svakom pojedinačnom primjeru skladišta podataka (dalja razrada CASE alata će uzeti u obzir iskustva sa prototipom). **Prvi korak** podrazumijeva identifikaciju konkretnih aplikacija i sistema (baza podataka, struktuiranih i nestruktuiranih izvora podataka) za skladište podataka tako što će korisnik kroz korisnički interfejs označiti konkretne izvore podataka. **Drugi korak** u ovoj fazi predstavlja učitavanje podataka u tabelu DataSources na osnovu izabranih izvora podataka.

U okviru **trećeg koraka** predviđeno je pronalaženje pravila za učitavanje metapodataka struktuiranih, polustruktuiranih i nestruktuiranih izvora podataka u odgovarajuće tabele prelazne baze podataka. Ova pravila su definisana u modelu i snimljeni u tabelu Rules. Za identifikaciju pravila potrebno je saznati tip izvora podataka, struktuirani (engl. *structured – ST*) polustruktuirani (engl. *semistructured – SS)*, nestruktuirani (engl. *unstructured - US*) iz svih izvora podataka na osnovu ulaznih parametara. Ove parametre ćemo dobiti pomoću mehanizma kursora kroz tabelu DataSources. Kursor se uvodi kao skup zapisa kome je pridružen pokazivač koji upućuje na tekući red. Komande u SQL-u sadrže iskaze koji pomjeraju pokazivač za rad sa tekućim redom. Kursor prihvata sintaksu koju određuje ISO/ANSI standard u oblasti

SQL [GJ00]. Ukoliko se radi o polustruktuiranim izvorima podataka, zbog nepostojanja jasne strukture, njihove operativne podatke je potrebno importovati u prelaznu bazu podataka radi eventualnog dodatnog strukturiranja. Polustruktuirani podaci kao npr. Excel fajlovi se mogu importovati u bazu podataka pomoću ETL procesa [MTK08]. Import metapodataka polustruktuiranih izvora podataka u prelaznu bazu podataka uradiće se pomoću uskladištene procedure *sp_SS_ImportData.* Ovoj proceduri će se na osnovu izvora podataka proslediti parametar za konkretan izvor podataka (Excel, XML, TXT i dr.).

Ukoliko se radi o nestruktuiranim izvorima podataka, potrebno je najprije identifikovati njihove metapodatke. Nestruktuirani podaci mogu biti smješteni u bazu podataka na tradicionalan način tako što se metapodaci o fajlovima, putanja do datoteke ili URL adresa, te atributi i veze između fajlova smještaju u bazu podataka. Noviji način za snimanje nestrukturiranih podataka u baze podataka je poznat kao tekstualna analiza (engl. *text analytics*). To je proces pretvaranja nestrukturiranih dokumenata u strukturirane, analizom strukture teksta u okviru dokumenta [Rai08]. Text analytics je proces omogućavanja računaru da izvlači značenje iz teksta. Tekst analiza često se provodi kao niz iterativnih procesa i potrebno ju je provesti prije učitavanja u bazu podataka [NSZ08] gde sam tekst najprije pretvaramo u oblik pogodan za analitičke obrade. Neki od postupaka prije učitavanja u bazu podataka su: *simple editing, stop-word removal, synonym replacement or concatenation, homographic resolution, thematic clustering, external glossary/taxonomy overlay, stemming, alternate spelling resolution, foreign language accommodation, direct and indirect search support*. Opis svakog od postupaka dat je u [IK12].



Slika 28. Dijagram toka druge međufaze

Četvrti korak u ovoj međufazi je popunjavanje tabela Tables, SemiStructures, UnStructures i TableColumns informacijama o tabelama i njihovim kolonama o transakcionim bazama podataka. Ovaj korak će se uraditi automatski iz generisane procedure za import podataka korišćenjem podataka iz sistemske šeme ciljnog repozitorijuma o sadržanim tabelama, kolonama tabela, indeksima, ograničenjima i relacijama za svaki konkretan sistem za upravljanje bazama podataka. Insert metapodataka u tabelu TableColumns izgleda složenije s obzirom na veći broj metapodataka i veći broj kolona. Sličan je pristup kod ostalih struktuiranih izvora podataka. Import metapodataka u tabele Tables i TableColumns uradiće se pomoću uskladištenih procedura

označene u modelu kao *sp_InsertDSTables* i *sp_InsertTablesColumns.* Ovoj proceduri će se na osnovu izvora podataka proslediti parametar za konkretan struktuirani izvor podataka (MS SQL Server, Oracle, IBM DB2 i dr.), a na osnovu pravila u tabeli Rules.

U ovoj međufazi samo prvi korak zahtjeva učešće korisnika, dok su ostali, u slučaju struktuiranih izvora podataka, u potpunosti automatizovani. Automatizaciju omogućuju pravila koja su spremljena u tabela RuleTypes i Rules U slučaju polustruktuiranih izvora podataka potrebno je djelomično učešće korisnika kod struktuiranja podataka, dok je u slučaju nestruktuiranih izvora podataka potrebno učešće korisnika i kod tekstualne analize.

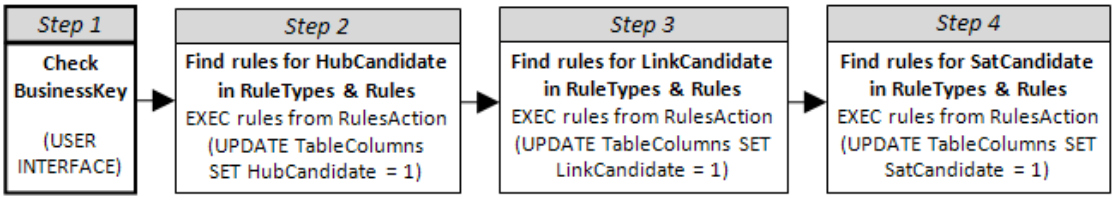
U najkraćem, algoritam za ovu međufazu se sastoji od sljedećeg [KJM14]:

```
1.  Loop over data sources (incrementally or in sets of
    DBs as sources): Select data source and Insert
    metadata in table DataSources
2.  For each row in table DataSources: find structure
    type, source type and rules
    a) For each structured type:
        i.  Insert metadata in Tables
        ii.  For each row in table Tables: insert
             metadata in table TableColumns
    b) For each semistructured type:
        i.  Extract and load data in database StageDW
        ii. Insert metadata in table SemiStructures
        iii. For each row in table SemiStructures: insert
             metadata in TableColumns
    c) For each unstructured type:
        i.  Text analytics process
        ii. Extract and load data in database StageDW
        iii.   Insert metadata in table UnStructures
        iv.  For each row in table UnStructures: insert
             metadata in TableColumns
```

4.2.3. Identifikacija PDV (Physical Data Vault) tipova

Identifikacija PDV tipova (engl. *Identification of PDV Types*) za svaki pojedinačni DW inicijalno svodi se na identifikaciju poslovnih ključeva dok se hub, link i satelit automatski generišu. Kako su, sa modela na slici 26, značenja kolona u tabelama DataStructure, SourceType, DataSource, Tables, SemiStructures, UnStructures jasna, ovdje ćemo obrazložiti samo strukturu TableColumns i namjenu pojedinih kolona (Tabela 8.). Ova tabela treba da sadrži metapodatke iz baza podataka, te metapodatke polustruktuiranih i nestruktuiranih izvora koje je moguće na odgovarajući način struktuirati. U pristupu u disertaciji predviđeno je automatsko učitavanje metapodataka u TableColumns iz svih struktuiranih izvora podataka i dijelom iz nestruktuiranih i polustruktuiranih izvora podataka. Pored podataka koji će se popuniti na osnovu šeme baze podataka, ova tabela će sadržavati indikator kolone koji će nam dati informaciju da li ona predstavlja poslovni ili surogat ključ, odnosno da li je tabela hub, link ili satelite kandidat.

Dijagram toka treće međufaze, dat je na sljedećoj slici.



Slika 29. Dijagram toka treće međufaze

Tabela 8. Struktura TableColumns tabele

Kolona	Tip podatka	Opis	Podaci se generišu	Podatke unosi korisnik
ColumnID	int	Primarni ključ tabela ID	✓	
TableID	int	Identifikator tabele iz odgovarajuće baze podataka	✓	
SemiStrID	int	Identifikator polustruktuiranog izvora podataka	✓	
UnStrID	int	Identifikator nestruktuiranog izvora podataka	✓	
DataSourceName	varchar	Ime izvora podataka	✓	
TableOrStrName	varchar	Ime tabele, polustruktuiranog ili nestruktuiranog izvora podataka	✓	
ColumnSysId	integer	Identifikator kolone posmatrane tabele	✓	
ColumnName	varchar	Ime kolone posmatrane tabele	✓	
ColumnType	varchar	Tip podatka	✓	
ColumnPK	bit	Da li je kolona primarni ključ?	✓	
ColumnFK	bit	Da li je kolona strani ključ?	✓	
Nullable	bit	Da li kolone može imati NULL?	✓	
IfFK_PKTable	varchar	Ime odgovarajuće tabele na strani primarnog ključa (ako je kolona strani ključ)	✓	
IfFK_PKColumn	varchar	Ime odgovarajuće kolone u tabeli primarnog ključa (ako je kolona strani ključ)	✓	
BusinessKey	bit	Da li je kolona poslovni ključ?		✓
SurrogateKey	bit	Da li kolona ima surogat ključ za odgovarajući poslovni ključ?	✓	
HubCandidate	bit	Da li je kolona hub kandidat?	✓	
LinkCandidate	bit	Da li je kolona link kandidat?	✓	
SatCandidate	bit	Da li je kolona satelit kandidat?	✓	

U nastavku je dat kratak opis koraka ove međufaze.

1. Identifikacija poslovnih ključeva kroz korisnički interfejs. Kroz odgovarajući korisnički interfejs, na osnovu podataka, korisnik treba da označi poslovne ključeve. Na osnovu izabranih poslovnih ključeva popuniće se kolona BusinessKey sa odgovarajućom vrijednosti (0 ili 1). Popunjavanje ove kolone moguće je automatski na osnovu pravila pohranjenog u tabeli Rules, a prema [Lin1, Lin2, Dam08, Gra11] i na osnovu BusinessKey popuniti će se kolona SurrogateKey.

2. Identifikacija hub tabela. Hub entitet u tabeli nosi minimum jedinstvene liste poslovnih ključeva. To su ključevi koje organizacija koristi u svakodnevnom poslovanju kao npr. broj (šifra) kupca, broj (šifra) zaposlenog, broj računa, i sl. [Lin2]. Hub kandidate je moguće identifikovati na osnovu popunjenih BusinessKey i SurrogateKey i na osnovu pravila za identifikaciju Hub-ova koja se nalaze u tabelama RuleTypes i Rules, a prema [Lin1, Lin2, Gra11].

3. Identifikacija link-ova. Link je fizički prikaz referenci stranih ključeva i više-prema-više odnosa u trećoj normalnoj formi [Lin2]. Linkove je moguće identifikovati pomoću tabele pravila Rules, a na osnovu tipova pravila koja se nalaze u RuleTypes koja sadrži sljedeće korake:

a) traženje tabele više-prema-više
b) postavljanje vrijednosti kolone LinkCandidate na *true* u tabelama više-prema-više
c) postavljanje vrijednosti kolone LinkCandidate na *true* za tabele koje imaju strani ključ

4. Identifikacija satelita. Satelit entitet pokazuje kontekst hub podataka i sadrži atribute koji nisu strani ili primarni ključevi [Lin1]. Sateliti se identifikuju na osnovu pravila u tabelama RuleTypes i Rules, a prema ([Lin1], [Lin2], [Gra11], [CBD10]).

4.2.4. Inicijalno deklarisanje strukture PDV

Zadnja međufaza u procesu automatizacije fizičkog projektovanja skladišta podataka predstavlja Deklarisanje PDV strukture (engl. *Declare PDV structure*). U nastavku su dati koraci ove faze.

1. Generisanje i izvršavanje skripta za kreiranje hub tabela. Nakon što su određeni poslovni ključevi, identifikovane su odgovarajuće Hub tabele koje sadrže označene poslovne ključeve (postavljanjem vrijednosti **1** u koloni HubCandidate). Zahvaljujući ovako napravljenoj i popunjenoj tabeli, moguće je generisati skript za kreiranje hub tabela u skladištu podataka zasnovanom na Data Vault konceptu, a na osnovu pravila u tabeli Rules. U okviru ovog koraka predviđeno je sledeće:

a) formiranje kursora za prolazak kroz TableColumns za HubCandidate=1
b) preuzimanje podataka iz tabele i dodjeljivanje promjenljivim
c) u svakoj iteraciji koristimo dinamički SQL da bi dopunili sql_statement
d) izvršavanje sql_statement-a, čime se kreiraju hub tabele

Radi ilustracije, u nastavku je dat SQL kod uskladištene procedure koja radi saglasno navedenim principima:

```
--sp_CreateHubTables.sql
IF EXISTS (SELECT name FROM sysobjects WHERE name =
'sp_CreateHubTables' AND type = 'P')
   DROP PROCEDURE sp_CreateHubTables
GO
--parametar @BaseDW: ime skladišta podataka
```

```
CREATE PROCEDURE sp_CreateHubTables @BaseDW varchar (100)
AS
--deklarisanje varijabli za kursor nad tabelom TableColumns
SET NOCOUNT ON
DECLARE
@TableID int, @SemiStrID int, @UnStrID int, @TableOrStrName
varchar(30, @ColumnSysID int, @ColumnName varchar(30, @ColumnType
varchar(20, @ColumnPK bit, @ColumnFK bit, @IfFK_PKTable varchar(30,
@IfFK_PKColumn varchar(30), @Nullable bit, @BusinessKey bit,
@SurrogateKey bit, @HubCandidate bit, @LinkCandidate bit,
@SatCandidate bit,@IsNull varchar (8), @CurrentTableOrStrName
varchar(100), @MaxIdColumn int, @PrimaryKeyColumn varchar(100),
@sql_statement varchar(3000)
--inicijalizacija vrijednosti za trenutnu tabelu u kursoru
SET @CurrentTableOrStrName = ''
--deklarisanje kursora za tebelu TableColumns za HubCandidate = 1
DECLARE crTableColumns CURSOR READ_ONLY FOR SELECT
    TableID, SemiStrID, UnStrID, TableOrStrName, ColumnSysID,
ColumnName, ColumnType, ColumnPK, ColumnFK, IfFK_PKTable,
IfFK_PKColumn, Nullable, BusinessKey, SurrogateKey, HubCandidate,
LinkCandidate, SatCandidate FROM TableColumns WHERE HubCandidate = 1
--otvaranje kursora
OPEN crTableColumns
FETCH NEXT FROM crTableColumns INTO @TableID, @SemiStrID, @UnStrID,
@TableOrStrName, @ColumnSysID, @ColumnName, @ColumnType, @ColumnPK,
@ColumnFK, @IfFK_PKTable, @IfFK_PKColumn, @Nullable, @BusinessKey,
@SurrogateKey, @HubCandidate, @LinkCandidate, @SatCandidate
WHILE @@FETCH_STATUS = 0
BEGIN
   --ispitujemo da li je nova tabela u kursoru
   IF @CurrentTableOrStrName <> @TableOrStrName
   BEGIN
      --upisujemo početni kod za kreiranje tabele
  SET @sql_statement = 'CREATE TABLE ' + @BaseDW + '.dbo.Hub_' +
  @TableOrStrName + '('
      --tražimo zadnju kolonu u tekućoj tabeli u kursoru
      --poslije čega kreiramo zadnja dva reda koda za LoadDate i
    RecordSource
      SELECT @MaxIdColumn = MAX(ColumnSysId) FROM TableColumns WHERE
    TableOrStrName = @TableOrStrName AND HubCandidate = 1
END
   --ispitujemo da li kolona može imati vrijednost NULL
   --u transakcionoj bazi podataka
   IF @Nullable = 1
   BEGIN
      SET @IsNull = 'NULL'
   END
   ELSE   SET @IsNull = 'NOT NULL'
   --dopunjavamo @sql_statement
  SET @sql_statement = @sql_statement + @ColumnName + ' ' +
  @ColumnType + ' ' + @IsNull + ','
```

```
    --ispitujemo da li je kolona primarni ključ
    IF @ColumnPK = 1
    BEGIN
        SET @PrimaryKeyColumn = @ColumnName
    END
   --ispitujemo da li smo došli do zadnje kolone u tekućoj tabeli da bi
    --sql komandi dodali dio za kreiranje LoadDate, RecordSource PKey
    IF @ColumnSysId = @MaxIdColumn
    BEGIN
        --dopunjavanje sql statement-a
        SET @sql_statement = @sql_statement + 'LoadDate DateTime NOT
      NULL,'
        SET @sql_statement = @sql_statement + 'RecordSource nvarchar(20)
      NOT NULL,'
        SET @sql_statement = @sql_statement + 'CONSTRAINT [PK_Hub_' +
      @PrimaryKeyColumn + ']' + 'Primary Key ' + '(' + @PrimaryKeyColumn
      + '))'
        --izvršavamo SQL komandu za kreiranje Hub tabele
        EXEC (@sql_statement)
        --PRINT @sql_statement
    END
    --postavljanje sledeće tabele za tekucu tabelu u kursoru
    SET @CurrentTableOrStrName = @TableOrStrName

     FETCH NEXT FROM crTableColumns INTO @TableID, @SemiStrID,
@UnStrID, @TableOrStrName, @ColumnSysID, @ColumnName, @ColumnType,
@ColumnPK, @ColumnFK, @IfFK_PKTable, @IfFK_PKColumn, @Nullable,
@BusinessKey, @SurrogateKey, @HubCandidate, @LinkCandidate,
@SatCandidate
END
--zatvaranje kursora
CLOSE crTableColumns
DEALLOCATE crTableColumns
GO
```

Navedena procedura generiše SQL skriptove za kreirenje svih tabela koji su označeni kao hub kandidati. U nastavku je dat primjer generisanog SQL skripta za kreiranje tabele Hub_Customers:

```
CREATE TABLE BaseDW.dbo.Hub_Customers(CustomerID nchar(5) NOT
NULL,LoadDate DateTime NOT NULL,RecordSource nvarchar(20) NOT
NULL,CONSTRAINT [PK_Hub_CustomerID]Primary Key (CustomerID))
```

U ovom slučaju pretpostavka je da je poslovni ključ CustomerID stabilan (da se ne mijenja tokom vremena), te da je jedinstven na nivou kompanije zbog čega je izostavljen surogat ključ na nivou hub tabele. Moguć je i pristup gdje će hub

tabela sadržavati i dodatni surogat ključ, ali u tom slučaju nad kolonom koja označava poslovni ključ potrebno je da postoji jedinstveni (engl. *unique*) index.
U proceduri se može predvidjeti i mogućnost eksporta generisanih skritpova u txt ili drugi format.

2. Generisanje i izvršavanje skripta za kreiranje link tabela. Kako link tabela služi za predstavljanje odnosa ili transakcija između dviju ili više poslovnih komponenata (dva ili više hub-ova), link kandidati su više prema više tabele iz transakcione baze podataka. Zahvaljujući napravljenoj i popunjenoj TableColumns tabeli, identifikovane su odgovarajuće Link tabele koje sadrže označene postavljanjem vrijednosti **1** u koloni LinkCandidate, moguće je generisati skript za kreiranje link tabela u skladištu podataka, a na osnovu pravila u tabeli Rules. U okviru ovog koraka predviđeno je sledeće:

a) formiranje kursora za prolazak kroz TableColumns tabelu za LinkCandidate=1
b) preuzimanje podataka iz tabele i dodjeljivanje promjenljivim
c) u svakoj iteraciji koristimo dinamički SQL da bi dopunili sql_statement
d) izvršavanje sql_statement-a, čime se kreiraju link tabele

3. Generisnje i izvršavanje scripta za kreiranje satelit tabela. Kako Satelit entitet pokazuje kontekst hub podataka, satelit tabela će biti kreirana za svaku hub tabelu i sadržavaće neključne atribute tabela iz transakcione baze koje su postale hub tabele. Zahvaljujući napravljenoj i popunjenoj TableColumns tabeli, identifikovane su odgovarajuće Satelit tabele koje sadrže označene postavljanjem vrijednosti **1** u koloni SatCandidate što omogućuje generisanje skripta za automatsko kreiranje satelit tabela, a na osnovu pravila u tabeli Rules. U ovom slučaju se generiše po jedna tabela za svaku hub tabelu (ili link

tabelu ukoliko tabela iz transakcione baze ima samo link tabelu). U okviru ovog koraka predviđeno je sledeće:

a) formiranje kursora za prolazak kroz TableColumns tabelu za SatCandidate = 1

c) preuzimanje podataka iz tabele i dodjeljivanje promjenljivim

d) u svakoj iteraciji koristimo dinamički SQL da bi dopunili sql_statement

e) izvršavanje sql_statement-a, čime se kreiraju satelit tabele

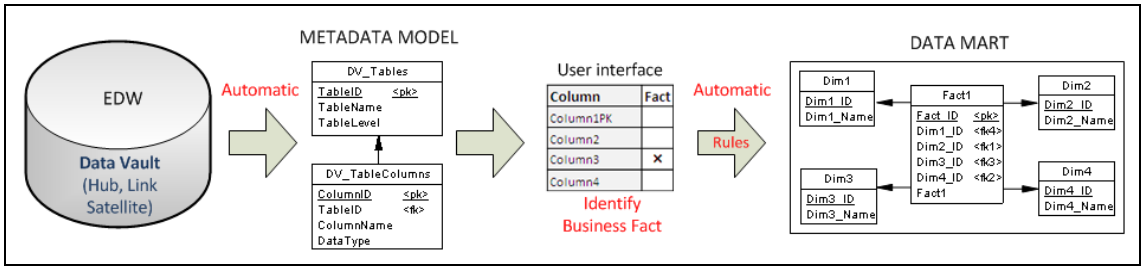
Ukoliko se u polustruktuiranom izvoru podataka, naprimjer u Excel ili TXT fajlu, nalaze podaci o kupcima koji se ne nalaze u transakcionoj bazi podataka (ostali kontakt podaci, podaci o tržišnoj poziciji, podaci o poslovanju i sl), formiraće se dodatna Satelit tabela koja će sadržavati podatke iz Excel ili TXT fajla.

U praksi je moguća i realizacija Data Vault skladišta podataka gdje svaki hub ili link može imati više od jednog satelita. Naprimjer, u slučaju da imamo iste podatke u više izvora podataka, za svaki od njih može se formirati po jedna satelit tabela. Drugi razlog za formiranje više satelita je različita brzina promjene opisnih podataka koje sadrži satelit tabela.

Opisanim postupcima u ovom poglavlju moguće je kreirati kompletno skladište podataka zasnovano na Data Vault konceptu sa minimalnim učešćem korisnika. Kod kreiranja hub, link i satelit tabela, kreira se i jedinstveni indeks nad kolonama koje čine primarni ključ. SQL kod uskladištenih procedura koje se koriste u ovoj fazi dat je na CD-u u prilogu disertacije.

4.3. Automatizacija fizičkog dizajna data martova

Data Vault model je vrlo dobro rješenje za integrisanje, spremanje i zaštitu podataka. Međutim, manje je pogodan za intenzivne upita ili izvještaje. To je razlog zašto arhitekture skladišta podataka koje koriste Data Vault obično sadrže sloj Data Mart-a gdje se nalaze jedna ili više zvijezda šema radi pristupa podacima [CBD10]. Prema [Gov10], dimenzije u star šemi nastaju iz hub i satelit tabela, a mjere (fact) iz satelit i link tabela. Pojednostavljena ideja automatizacije dizajna data marta iz Data Vaulta data je na sljedećoj slici.



Slika 30. Proces generisanja data marta iz Data Vaulta

Određivanje poslovnih mjera, dizajn fact tabela i tabela dimenzija je složen proces koji u najvećoj mjeri zahtjeva učešće korisnika. U literaturi koja se bavi automatizacijom dizajna skladišta podataka ili data mart-ova iz relacionog modela korišteno je nekoliko tehnika za identifikaciju mjera i dimenzija. U skladištima podataka, većina numeričkih atributa se nalazi u fact tabelama. Dimenzioni atributi imaju relaciju jedan-prema-više sa fact atributima i takvi odnosi su implementirani između odgovarajućih dimenzionih i fact tabela. Većina numeričkih atributa ima odnos jedan-prema-jedan sa drugim fact atributima [SB10]. Prema [Kim96], izvođenje fact tabele iz relacionog modela zasniva se na odabiru više-prema-više tabela iz ER modela koji sadrže numeričke činjenice koje nisu ključevi. Golfarelli u [GMR98], dobre kandidate za mjere vidi u entitetima koji se najčešće mijenjaju, te numeričke kolone koje se

agregiraju. Prema [PD02], numerički atributi mogu predstavljati mjere povezane sa poslovnim događajima, stoga entiteti sa najvećim brojem numeričkih atributa predstavljaju potencijalne fact tabele, dok su ostale tabele potencijalne dimenzije. Međutim, neki numerički atributi nisu baš pogodni za mjere kao što je broj telefona ili poštanski broj [ZMA11]. Rješenje ovog problema predstavlja korišćenje znanja spremljenog u WordNet semantičkom leksikonu kao bazi znanja za identifikaciju numeričkog podataka [NNH10] koji može biti poslovna mjera. Fact tabela uobičajeno ima vezu više-prema-jedan ka povezanim dimenzionim tabelama [RA07].

Kimbal u [KR02] daje smjernice za izradu dimenzionalnog modela u nekoliko oblasti: maloprodaja, popis zaliha robe, nabavka, CRM, računovodstvo, upravljanje ljudskim resursima, telekomunikacije, transport, obrazovanje, zdravstvo, osiguranje i elektronska trgovina. Prema [KR02], postoje tri fundamentalna tipa fact tabela: transaction, periodic snapshot, and accumulating snapshot. U sledećoj tabeli su date karakteristike svakog tipa. Pravilo iz [GMR98], da su dobri kandidati za mjere entiteti koji se najčešće mijenjaju, može se primijeniti samo na transakcioni tip fact tabela. Takvi bi entiteti bili naprimjer, stavke računa u maloprodaji ili veleprodaji robe.

Dimenzija određuje granularnost koja je usvojena za predstavljanje činjenica u procesu odlučivanja. Dimenzione tabele su denormalizovane i koriste činjenice od interesa za korisničke analize. U većini pomenutih oblasti, se između ostalih, najčešće koriste vremenske dimenzije sa različitim stepenom detaljnosti (godina, kvartal, mjesec).

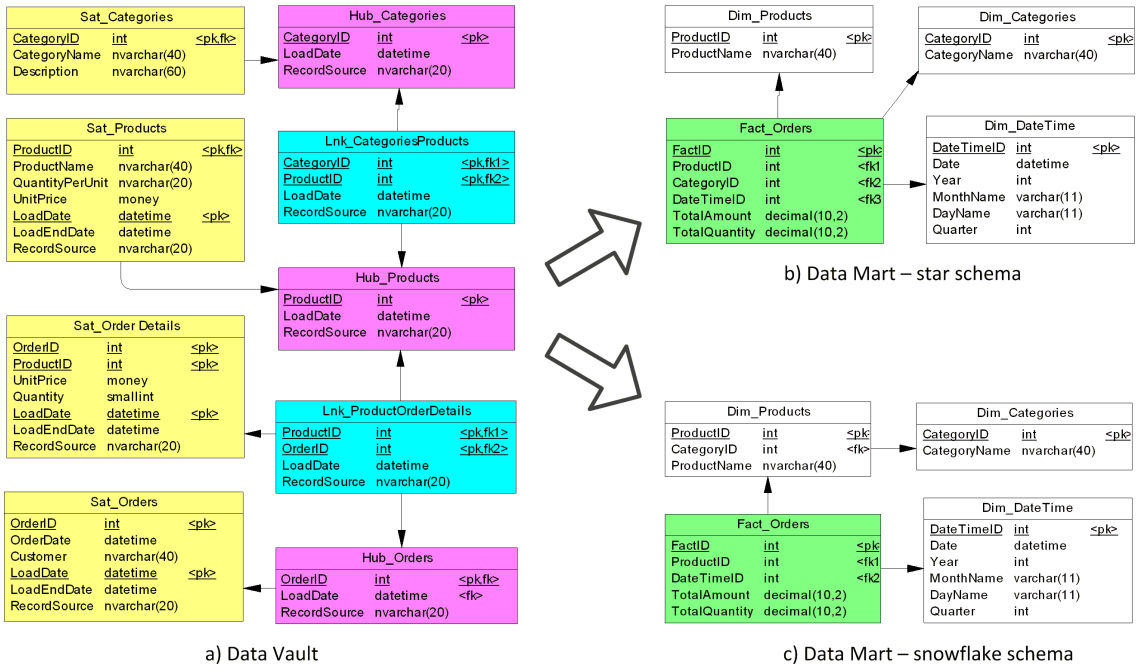
Tabela 9. Karakteristike tipova fact tabela [KR02]

CHARACTERISTIC	TRANSACTION GRAIN	PERIODIC SNAPSHOT GRAIN	ACCUMULATING SNAPSHOT GRAIN
Time period represented	Point in time	Regular predictable intervals	Indeterminate time span, typically short-lived
Grain	One per row transaction event	One row per period	One row per life
Fact table loads	Insert	Insert	Insert and update
Fact row updates	Not revisited	Not revisited	Revisited whenever activity
Data dimension	Transaction date	End-of-period date	Multiple dates for standard milestones
Facts	Transaction activity	Performance for predefined time interval	Performance over finite lifetime

Prema [MK00], entiteti u transakcionoj bazi se mogu podijeliti u transakcije, komponente i klasifikacije. Transakcioni entiteti opisuju poslovne događaje i predstavljaju najbolje kandidate za dimenzije. Entiteti iz kategorije komponente opisuju situaciju povezanu sa transakcijom kao npr. proizvod, vrijeme, lokaciju i dr. i predstavljaju dobre kandidate za dimenzije [NNH10]. Izbor dimenzija je od ključne važnosti za dizajn skladišta podataka i za ETL proces. Opšte je poznato da je vrijeme ključna dimenzija za ETL proces. Kada se dizajnira skladište podataka, vrijeme je eksplicitno predstavljeno kao relacioni atribut i stoga je očigledan kandidat za dimenziju [GMR98].

U literaturi vezanoj za skladište podataka koje je zasnovano na Data Vault konceptu, veoma malo se razmatra transformacija modela iz Data Vaulta u data martove. U pregledanoj literaturi nije nađen primjer niti formalizacija automatizacije dizajna Data Marta iz Data Vaulta. Preporuke za obrnut postupak, transformacija Data Mart-a iz Star Scheme u Data Vault model, dat su u [Lin4]. U ovom slučaju, potrebno je da se lociraju jedna ili više kolona u dimenzijama koji predstavljaju poslovne ključeve. Ukoliko su posmatrane kolone nezavisne, potrebno je napraviti odvojene hub tabele. U suprotnom će se napraviti jedna hub tabela.

Na sljedećoj slici je prikazan primjer mapiranja skladišta podataka baziranog na Data Vault konceptu na data mart u a) zvijezda šemu i b) kombinovanu šemu pahulje.



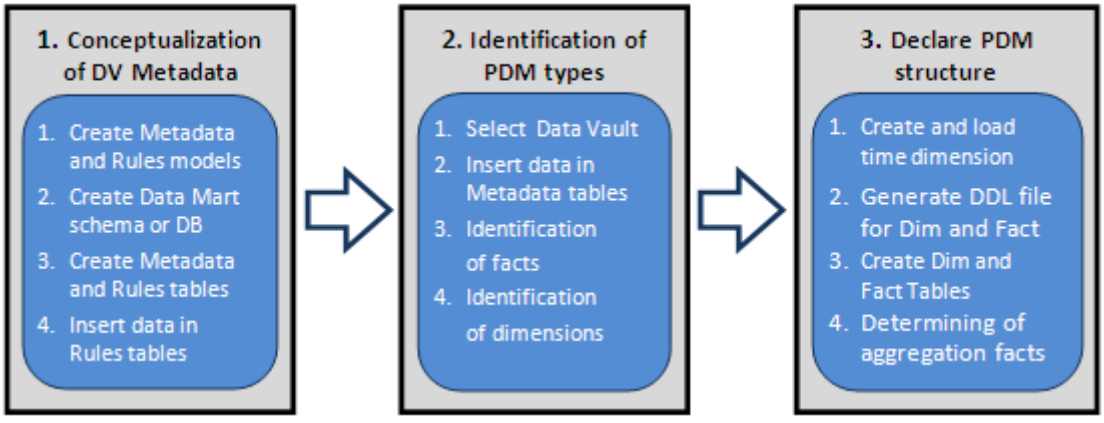
Slika 31. Primjer mapiranja Data Vaulta u data mart

Direktni pristup automatizacije projektovanja data marta predstavljen u ovom radu uključuje korištenje pravila za data mart dizajn. Radi razumjevanja

pravila, slijedi opšti algoritam za prepoznavanje činjenica i tabela mjera i odnosa zasnovanih na mapiranju Data Vault metapodataka. Uzimajući u obzir da je vremenska dimenzija je jedina dimenzija koja je skoro uvijek u svakom data martu, algoritam se sastoji od sljedećih koraka [KJM14]:

```
(i) For each Data Vault table
      (a) For each Link table
            • For each Satellite of Link: user confirms fact
              column by selected business fact
      (b) For each Hub table
            • For each busines key column of Hub: system confirms
              dimension by selected business key
            • For each Satellite of Hub: user confirms aditional
              dimension by selected other Satellite
              attribute(optional)
(ii)  Creating and filling time dimension table
(iii) For each confirmed fact
      (a) User determined aggregation of facts
```

U nastavku ovog poglavlja dat je postupak druge faze sa slike 22. koja podrazumijeva automatizaciju kreiranja data marta iz postojećeg skladišta podataka baziranog na Data Vault konceptu. Međufaze i koraci automatske realizacije data marta su dati na sljedećoj slici.

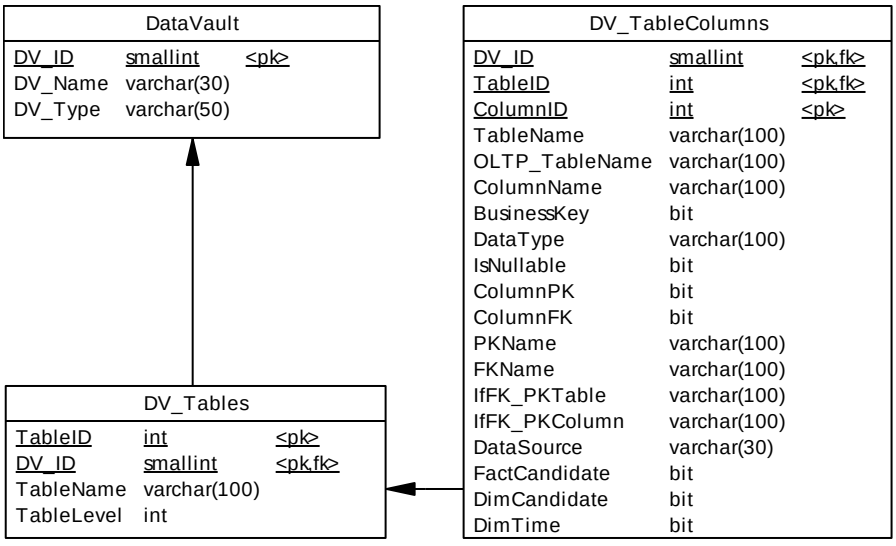


Slika 32. Međufaze i koraci u realizaciji data marta iz Data Vaulta

4.3.1. Konceptualizacija Data Vault metapodataka

Prva međufaza je **konceptualizacija DV metapodataka** (engl. *Conceptualization of DV Metadata*) koji se mogu koristiti u izradi individualnih data martova. **Prvi korak** u ovoj fazi podrazumijeva izgradnju modela metapodataka (slika 33) i modela pravila. Kod izrade modela razlikujemo dio modela koji je nezavisan od izvora podataka i dio modela koji zavisi od izvora podataka. Nezavisni dio se odnosi na tabele metapodataka i pravila, a zavisni se odnosi na procedure vezane za različite platforme skladišta podataka. Namjena i način ažuriranja tabele metapodataka prikazan je u tabeli 10.

RuleTypes tabela sadrži podatke o vrstama pravila, kao što su pravila za različite tabele Data Vault sistema, pravila za identifikaciju činjenica i dimenzija i pravila za kreiranje tabela činjenica i dimenzija. Tabela Rule sadrži informacije o pravilima za određenu vrstu podataka i postupke koje treba izvršiti kada se određeni uslov zadovoljen. Pravila u koloni Rules omogućuju lakše izvođenje naredbi koje se čuvaju u koloni RuleAction.



Slika 33. Model metapodataka Data Vaulta

U ovoj fazi mogu se koristiti i tabele pravila koja su kreirana u prvoj fazi (automatizacija kreiranja Data Vault skladišta podataka). Fizički model pravila

dat je na slici 27. Sljedeći primjer dat SQL jeziku ilustruje pravilo koje identifikuje kandidate za poslovne dimenzije na osnovu vrijednosti u koloni poslovnog ključa.

```
UPDATE DV_TableColumns SET DimCandidate = 1 WHERE
DV_TableColumns.BusinessKey = 1
```

Tabela 10. Namjena i način popunjavanja tabela metapodataka

Table	Namjena	Način popunjavanja
DataVault	Naziv Data Vault skladišta podataka kao izvor podataka za data mart	Automatski, nakon izbora skladišta podataka kroz korisnički interfejs
DV_Tables	Tabele Data Vault skladišta podataka	Automatski
DV_TableColumns	Kolone tabela Data Vault skladišta podataka	Automatski, izuzev polja koje predstavlja poslovnu mjeru (činjenicu) koje se selektuje kroz korisnički interfejs

Drugi korak uključuje kreiranje data mart šeme (ili data mart baze podataka). **Treći korak** obuhvata kreiranje tabela metapodataka i tabela pravila (ukoliko nisu kreirane u prvoj fazi), te odgovarajućih procedura. Posljednji korak ove međufaze uključuje učitavanje podataka u tabele pravila. Nakon prve međufaze, stvoreni su uslovi uslove za razvoj aplikacija koja omogućava da automatizaciju fizičkog dizajna data marta, uz minimalnu interakciju sa korisnikom. Druga i treća faza sa slike 32 treba da se urade za svaki konkretan sistem. Kako je značenje kolona u tabelama na slici 33, Data Vault i DV_Tables intuitivno i jasno, u Tabeli 11 se objašnjava struktura i namjena kolona tabele DV_TableColumns. Pristup u radu podrazumijeva automatsko učitavanje metapodataka u tabelu DV_TableColumns, izuzev kandidata za fact tabele i kandidata za povezivanje sa vremenskom dimenzijom.

Tabela 11. Struktura DV_TableColumns tabele [KJM14]:

Column	Data type	Description	Load automatic	Load through UI
DV_ID	smallint	Primary key	✓	
TableID	integer	Primary key	✓	
ColumnID	integer	Primary key	✓	
TableName	varchar	Name of Hub, Link or Sat	✓	
OLTP_TableName	varchar	Name of OLTP table	✓	
ColumnName	varchar	Column name	✓	
BusinessKey	bit	Business key column?	✓	
DataType	varchar	Data type of column	✓	
IsNullable	bit	Is column nullable?	✓	
ColumnPK	bit	Column Primary key?	✓	
ColumnFK	bit	Column Foreign key?	✓	
PKName	varchar	Primary key name	✓	
FKName	varchar	Foreign key name	✓	
IfFK_PKTable	varchar	Table name on PK side	✓	
IfFK_PKColumn	varchar	Column name on PK side	✓	
TableLevel	integer	Table hierarchial level	✓	
DataSource	varchar	Data Source name	✓	
FactCandidate	bit	Fact candidate?		✓
DimCandidate	bit	Candidate for dimension?	✓	
DimTime	bit	Candidate for time dimension?	✓	✓

4.3.2. Identifikacija PDM (Physical Data Mart) tipova

Prvi korak u ovoj međufazi podrazumjeva identifikaciju Data Vault skladišta podataka, dok drugi korak podrazumijeva insert metapodataka u odgovarajuće tabele na osnovu izabranog skladišta podataka. **Treći korak** podrazumijeva identifikaciju poslovnih mjera, dok četvrti korak predstavlja identifikaciju poslovnih dimenzija.

Uzimajući u obzir da je vremenska dimenzija jedina dimenzija koja skoro uvijek postoji u svakom data martu, jer se svaka informacije posmatra kroz neki vremenski period, u kratkim crtama algoritam ove faze se sastoji od sljedećeg:

I. For each row in DV_TableColumns table
   a. For each row in Satellite of Link: user confirms facts by selected FactCandidate column
   b. For each busines key column of Hub: system (according rules) confirms dimension by selected DimCandidate column
      - For each confirmed dimension: user choice offered dimension by selected/deselected DimCandidate column
   c. For each row in Satellite of Hub: user confirms aditional dimension by selected DimCandidate column (optional)
   d. For each row confirmed dimension
      - For each datetime dimension: user confirms time dimension by selected DimTime column

Identifikacija poslovnih mjera kroz korisnički interfejs. Kroz odgovarajući korisnički interfejs, korisnik treba da označi poslovne činjenice. Na osnovu selektovanih poslovnih mjera popuniti će se kolona FactCandidate sa odgovarajućom vrijednosti (0 ili 1). Popunjavanje ove kolone moguće je automatski na osnovu pravila pohranjenog u tabeli Rules. U pristupu datom u radu potrebno je da, kroz odgovarajući korisnički interfejs, korisnik prihvati ili ne predložene poslovne mjere (označavanjem kolone *FactCandidate* za izabrane redove u *DV_TableColumns* tabeli). Određivanjem poslovne mjere identifikuje se fact tabela. Mjera je uglavnom numerički ili podatak tipa: int, money, real, smallint, decimal [Kim96, GMR98, PD02] iz Sat_ tabela koje nisu PK ili FK [Lin4, Gov10]. Pomenute potencijalne mjere se uglavnom nalaze u tabelama na najnižem hijerarhijskom nivou (stavke narudžbe, stavke računa, stavke recepata, i sl.). u OLTP bazi. Radi lakše identifikacije mjera, u pristupu datom u tezi, korisniku se inicijalno nude samo podaci iz DV_TableColumns tabele koji imaju navedeni tip podatka, te su podaci sortirani po koloni TableLevel. Osim toga, u ovom dijelu bi korisniku bilo ponuđeno da odredi ime odgovarajuće fact tabele.

Identifikacija dimenzija. Nakon određivanja mjera, sistem na osnovu poslovnih ključeva iz hub tabela određuje tabele dimezija postavljanjem vrijednosti DimCandidate na 1 za odgovarajuće kolone iz hub i satelit tabela. Pored poslovnog ključa, tabela dimenzija treba da sadrži i opisne podatke iz odgovarajuće satelit tabele. Kroz odgovarajući korisnički interfejs, korisniku će se ponuditi da prihvati predložene dimenzije ili da izabere druge dimenzije iz odgovarajućih satelit tabela. Sistem dakle, predlaže inicijalne dimenzije: kolone BusinessKey iz Sat tabela i odgovarajući primarni ključ iz Hub tabela.

4.3.3. Deklarisanje Data Mart strukture

Zadnja međufaza u procesu automatizacije dizajna data marta iz Data Vaulta je incicijalna deklaracija PDM stukture. Ova međufaza uključuje sljedeće korake:

1. Kreiranje i učitavanje vremenske dimenzije. *P*otrebno je da korisnik identifikuje koji će se atributi u skladištu podataka koristiti za povezivanje sa tabelom vremenskih dimenzija. Vremenska dimenzija (datumska dimenzija) je jedina dimenzija koja je skoro zagarantovana da postoji u svakom data martu jer se gotovo svaki podatak posmatra kroz vremenski period. Datum je obično prva dimenzija u osnovi sortiranja baze podataka [KR02]. Poslovne mjere se posmatraju najčešće kroz godinu, kvartal, mjesec i dan. U nastavku je dat SQL skript *sp_CreateInsertDim_DateTime* koji kreira i učitava tabelu vremeskih dimenzija. Popunjavanje se vrši na osnovu posmatranog vremenskog perioda (StartDate i EndDate).

```
CREATE TABLE Dim_DateTime
(
 DateTimeID int NOT NULL IDENTITY(1, 1),
 [Date] datetime NOT NULL,
 [Month] int NOT NULL,
 [Quarter] int NOT NULL,
 [Year] int NOT NULL,
 CONSTRAINT PK_DateTime PRIMARY KEY CLUSTERED (DateTimeID)
```

```
)
-- deklarisanje varijabli za početak i kraj perioda
DECLARE @StartDate datetime
DECLARE @EndDate datetime
-- deklarisanje tekućeg datuma u petlji
DECLARE @CurrentDate datetime
SET @StartDate = '2005/01/01' -- min date from DimTime column
--(naprimjer u tabeli Racuni)
SET @EndDate = '2013/12/31' -- max date from DimTime column
--( naprimjer u tabeli Racuni)
SET @CurrentDate = @StartDate
WHILE @CurrentDate <= @EndDate
BEGIN
 -- dodavanje slogova u tabelu dimenzija Dim_DateTime
 INSERT INTO Dim_DateTime ([Date],[Month],[Year],[Quarter])
 VALUES (@CurrentDate, MONTH(@CurrentDate), YEAR(@CurrentDate),
  CASE  WHEN MONTH(@CurrentDate) IN (1, 2, 3) THEN 1
    WHEN MONTH(@CurrentDate) IN (4, 5, 6) THEN 2
    WHEN MONTH(@CurrentDate) IN (7, 8, 9) THEN 3
    WHEN MONTH(@CurrentDate) IN (10, 11, 12) THEN 4
  END
)
 -- postavljanje vrijednosti za varijablu @CurrentDate
 SET @CurrentDate = DateAdd(d, 1, @CurrentDate)
END
GO
```

2. Kreiranje DDL (Data Definition Language) fajla za fact i dimenzione tabele

a) Kada je identifikovana poslovna činjenica (popunjavanjem kolone FactCandidate sa 1), omogućeno je generisanje scripta za kreiranje fact tabele u data martu pomoću pravila pohranjenih u tabeli Rules. Ovaj korak uključuje sljedeće aktivnosti:

- formiranje kursora za prolazak kroz DV_TableColumns (Where FactCandidate=1)
- preuzimanje podataka iz tabele i dodjeljivanje promjenljivim
- u svakoj iteraciji korišćenje dinamičkog SQL-a za dopunjavanje sql_statementa za kreiranje fact tabele

b) Kolona DimCandidate će se automatski popuniti na osnovu kolone BusinessKey i na osnovu pravila snimljenog u tabeli Rules saglasno [Lin1] [Lin2] [PMR03] [ZC10]. Odgovarajuća dimenzija se identifikuje na osnovu vrijednosti 1 u koloni DimCandiadate. Nakon ove aktivnosti omogućeno je

generisanje scripta za kreiranje dimenzionih tabela. Ovaj korak uključuje sljedeće aktivnosti:

- formiranje kursora za prolazak kroz DV_TableColumns Where DimCandidate=1
- preuzimanje podataka iz tabele i dodjeljivanje promjenljivim
- u svakoj iteraciji korišćenje dinamičkog SQL-a za dopunjavanje sql_statementa za kreiranje dimenzionih tabela

3. **Kreiranje dimenzionih i fact tabela.** Poslije kreiranja DDL fajla, omogućeno je kreiranje fact i dimenzionih tabela. Najprije se izvršava SQL skript za kreiranje dimenzionih tabela, a zatim skript za kreiranje fact tabele.

Za primjer na slici 1, generisaće se sljedeći SQL kod:

```
CREATE TABLE DimPartneri (
   PartnerID            numeric(2)          not null,
   PartnerNaziv         varchar(100)        null,
   PartnerMjesto        varchar(50)         null,
   constraint PK_DIM_PARTNERI primary key (PartnerID)
)
GO

CREATE TABLE DimProizvodi (
   ProizvodID           varchar(13)          not null,
   ProizvodNaziv        varchar(50)          null,
   ProizvodJM           varchar(5)           null,
   ProizvodVrsta        char(20)             null,
   constraint PK_DIM_PROIZVODI primary key (ProizvodID)
)
GO

CREATE TABLE DimVrijeme (
   VrijemeID            numeric              not null,
   Godina               char(4)              null,
   Kvartal              numeric(1)           null,
   Mjesec               char(2)              null,
   constraint PK_DIM_VRIJEME primary key (VrijemeID)
)
GO

CREATE TABLE FactProdaja (
   ProdajaID            numeric              not null,
   ProizvodID           varchar(13)          null,
   VrijemeID            numeric              null,
   PartnerID            numeric(2)           null,
   Kolicina             decimal(12,3)        null,
   Iznos                decimal(12,3)        null,
   constraint PK_FACT_PRODAJA primary key (ProdajaID),
```

```
   constraint FK_FACT_PRO_REFERENCE_DIM_PROI foreign key (ProizvodID)
      references DimProizvodi (ProizvodID),
   constraint FK_FACT_PRO_REFERENCE_DIM_VRIJ foreign key (VrijemeID)
      references DimVrijeme (VrijemeID),
   constraint FK_FACT_PRO_REFERENCE_DIM_PART foreign key (PartnerID)
      references DimPartneri (PartnerID)
)
GO
```

4. **Određivanje agregacije mjera.** Poslije kreiranja fact i dimenzionih tabela, na kraju ove međufaze, potrebno je da korisnik izabere mjere koje će biti agregirane kao i način agregacije mjera. Za ovu namjenu kreirana je tabela FactAggregation u kojoj se nalaze informacije o načinu agregacije poslovnih činjenica (mjera).

```
CREATE TABLE FactsAggregation (
   FA_ID                int                  identity,
   FA_TableName         varchar(100)         not null,
   FA_ColumnName        varchar(100)         not null,
   FA_DataType          varchar(100)         not null,
   FA_FactName          varchar(30)          not null,
   FA_Formula           varchar(300)         not null,
   constraint PK_FACTSAGGREGATION primary key (FA_ID)
)
GO
```

Naprimjer, ukoliko imamo tabelu StavkeRacuna u kojem se nalazi količina prodatih artikala i jedinična cijena artikala, potrebno je odrediti da će se npr. količina sabirati, a iznos prodate robe računati kao količina*jedinična cijena.

ID	FactName	TableName	ColumnName	Formula
1	Sales	[Sat_Order Details]	UnitPrice	SUM [(Quantity)*(UnitPrice)]
2	TotalQuantity	[Sat_Order Details]	Quantity	SUM (Quantity)
3	TotalCount	[Sat_Orders]	OrderID	COUNT (OrderID)

Slika 34. Primjer popunjene tabele FactsAggregation

Procedure i postupci opisani u ovom poglavlju omogućuju kreiranje data martova iz skladišta podataka baziranog na Data Vault konceptu sa minimalnim učešćem korisnika. Dio prve međufaze može (kreiranje tabela

metapodataka) može da se uradi u prvoj fazi dizajna Data Vault sistema, a nakon kreiranja hub, link i satelit tabela Data Vault skladišta podataka. Radi efikasnijeg pristupa podacima, kod kreiranja tabela mjera i dimenzija kreira se i jedinstveni indeks nad kolonama koje čine primarni ključ.

4.4. Automatizacija ETL procesa

Problem neautomatizovane izrade dimenzionog modela skladišta podataka, mogućnost automatizacije dizajna skladišta podataka zasnovanog na Data Vault konceptu, odnosno automatskog generisanja i prepoznavanja mjera i dimenzija na osnovu dizajna i fizičkog modela transakcione baze podataka i skladišta podataka otvara perspektive za istraživanje automatizacije ETL procesa u dijelu prikupljanja, obrade i učitavanja internih podataka iz transakcione baze, te podrški učitavanja podataka iz hub, link i satelit tabela u tabele mjera i dimenzija. U ETL procesu, odvijaju se aktivnosti inicijalne (početne) ekstrakcije, transformacije i učitavanja podataka, te tekuće ekstrakcije, transformacije i učitavanja podataka. Kod inicijalnog ETL se trenutni i istorijski podaci iz transakcionih sistema i spoljnih izvora podataka prvi put učitavaju u skladište podataka. Nakon ovih podataka će se periodično učitavati tekući podaci aktiviranjem odgovarajućih mehanizama u alatima koji su namijenjeni za ETL proces.

U literaturi se nalazi nekoliko modela dizajna ETL-a ali svi dijele dvije vrste problema: oni zahtijevaju od dizajnera intenzivan ljudski trud, te tehnička znanja i znanja poslovnih procesa [SSC08]. Muñoz u [MMT09] predlaže pristup za automatsko generisanje koda ETL procesa korišćenjem MDA sa formalnom definicijom seta QVT (*Query, View, Transformation*) transformacije.

Problem definicije ETL aktivnosti i obezbeđenje njihove formalne konceptualne reprezentacije dato je u radu [VSS02]. Predloženi konceptualni model omogućuje: prilagođene za praćenje atributa i odgovarajuće ETL aktivnosti u ranim fazama projekta skladišta podataka, te prilagodljivost, proširivost i ponovno korišćenje obrazaca za ETL aktivnosti. Osim toga, pristup nudi mogućnost korišćenja čestih ETL aktivnosti kao npr. generisanje surogat ključeva, provjera null vrijednosti i dr.

ETL i prateći dizajnerske procesi trebali bi da daju mehanizme za automatsko generisanje ETL koda kako bi se smanjilo vrijeme razvoja složenih ETL procesa. U [MMT09] dat je pristup za automatsko generisanje koda ETL procesa. U tom cilju za modelovanje ETL procesa korišten je MDA sa formalnom definicijom skupa QVT transformacija. Kako automatizovati generički proces skladištenja podataka za praćenje poslovnih događaja iz aplikativnog okruženja prikazano je u radu [CSWD09]. Koristeći prošireni mehanizam u radu se pokazuje kako automatizovati izgradnju generičkog procesa skladištenja podataka sa dva nivoa preslikavanja koji se primjenjuju u dvije faze. Ovakav pristup korišćenja ETL tehnologije za skladištenje procesnih podataka može se posmatrati i kao korišćenje proizvoljnog ETL-a za modelovanje poslovnih procesa. Korist modelovanja ETL-a kao poslovnog procesa se ilustruje kroz izvršavanje ETL primjera, praćenje i analizu napretka ETL procesa, te rješavanje problema implementacije [CSWD09].

Okvir za praćenje evolucije ETL sloja dat je u radu [Woj11]. Strukturne promjene se prate i snimaju u bazu metapodataka. Aktivnosti u bazi su interakciji s promijenjenim EDS (*External Data Sources*), tako da se popravka aktivnosti može raditi na promijenjenoj EDS šemi. Predloženi okvir je razvijen kao modul izvan ETL-a kojem se pristupa pomoću API-ja. Doprinos ovog rada

je u značajnoj automatizaciji ETL-a i poluautomatskoj ispravci ETL grešaka. Strukturne promjene na spoljnim izvorima podataka su testirane kroz ovaj pristup i daju dobre rezultate.

Okvir za Model-driven development ETL procesa dat je u [AZMT11]. Korišćenjem platform nezavisnog modele za dizajn ETL procesa zasnovanog na standardu za modelovanje poslovnih procesa, *Business Process Modeling Notacija* (BPMN) omogućeno je kroz pristup u radu automatska izmjena ovog modela i generisanje koda za izvršavanje na konkretnoj platformi. Ovim pristupom se komplikovan dizajn i platform zavisan modeling ETL procesa znatno pojednostavljuje i automatizuje.

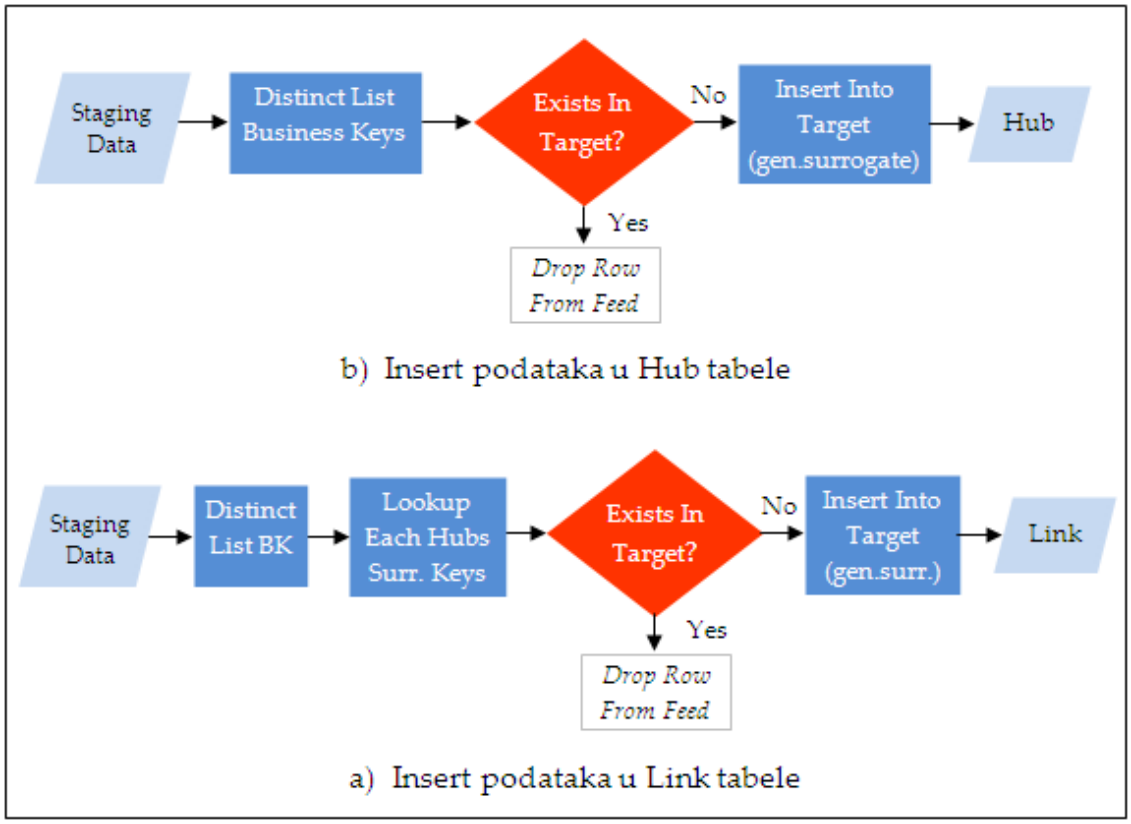
U radu [SS06] prikazan je ontološki pristup pogodan za snimanje informacije potrebnih za idejno rješenje ETL procesa. Uz odgovarajuća mapiranja smanjen je broj transformacija svakog snimanja podataka pomoću domena ontologije. Ovakvim pristupom se u određenoj mjeri automatizuje korespodencija između izvora podataka i skladišta podataka. Unapređenje ovog pristupa je dato u [SSC08], gdje se u olakšavanju komunikacije među različitim stranama koje čine poslovno i tehničko osoblje uključeno u projektovanje i razvoj u svim fazama DW projekta. Preporučuje se prirodni način komunikacije. ETL proces se predstavlja tekstualnim opisima sličnim prirodnom jeziku koji značajno doprinosi automatizaciji ETL-a. U radu se pokazuje da lingvističke tehnike mogu da se koriste u sinergiji s tim ontologijama, te se predlaže obrazac i mehanizam za prevođenje semantičkih podataka u potrebne ETL operacije.

Jovanović u [JRSA12] predstavlja alat GEM koji iz datog skupa poslovnih zahtjeva i opisa izvora podataka poluautomatski generiše ETL kod. Ovaj alat poluautomatski proizvodi višedimenzionalna ETL idejna rješenja iz datog

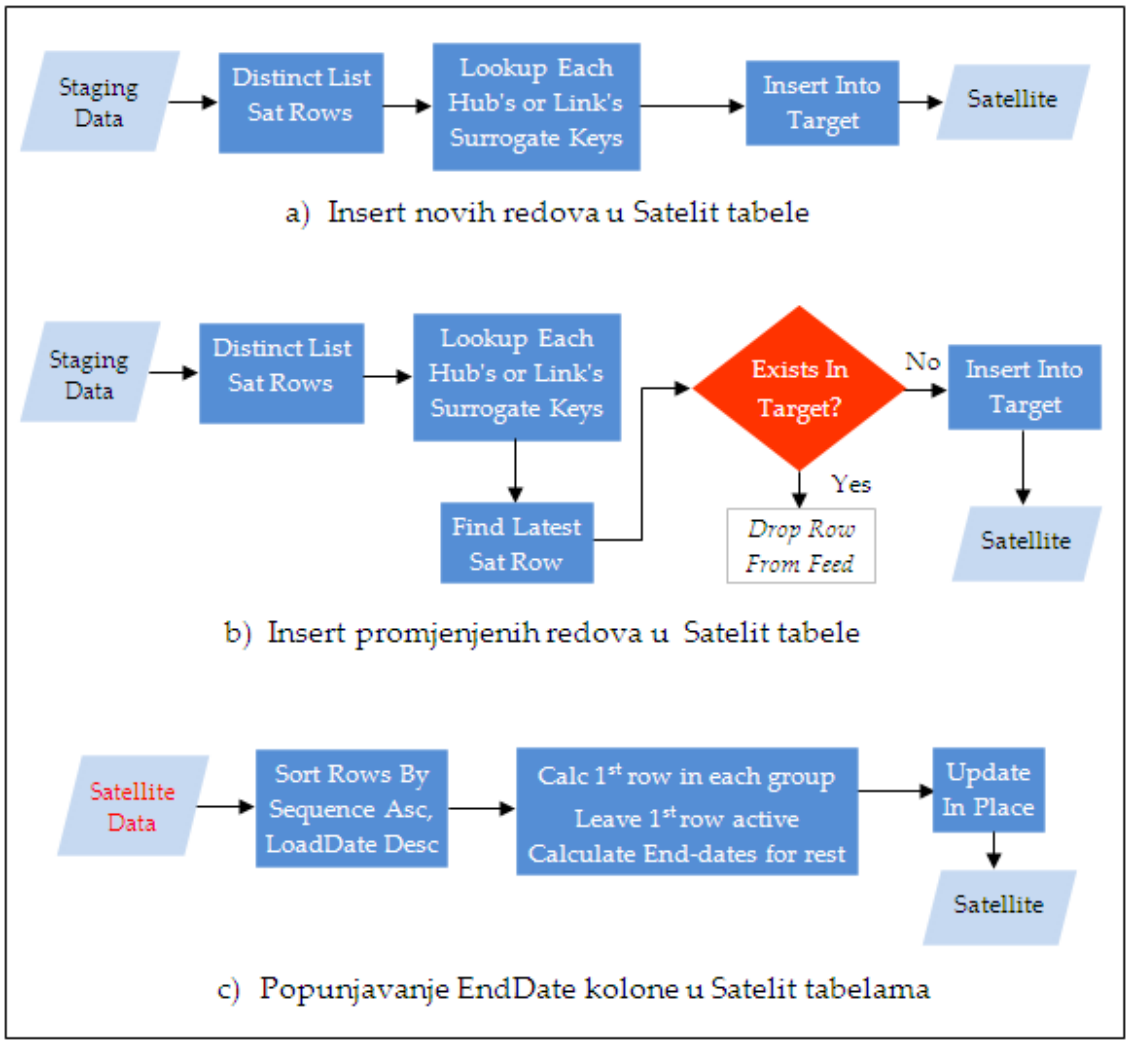
skupa poslovnih zahtjeva i opisa izvora podataka. Jovanović [JRSA12] predstavlja demand-driven pristup ETL dizajna. Svaki zahtjev se razmatra posebno generiše odgovarajući ETL proces. U ovom radu je predložen inkrementalni metod za dizajn ETL procesa na osnovu zahtjeva. Na kraju se generiše konačan ETL kod.

Jörg T. i Deßloch [JD08] daju pregled postojećeg stanja CDC (*Change Data Capture*) tehnike i istražuju ograničenja različitih pristupa. U radu se predstavlja nov pristup za generisanje inkrementalnog procese punjenja skladište podataka na osnovu deklarativne šeme mapiranja. Osnova njihovog rada je Orchid, prototip sistem razvijen u IBM Almaden Research Center, koji prevodi shemu mapiranja u ETL procese i obrnuto. Ovaj pristup ima jasne prednosti u pogledu performansi. CDC i *Change Data Application* (CDA) tehnike se koriste za snimanje promjene u izvorima i skladištu podataka, respektivno.

Linstedt u [Lin12] daje paterne za DV ETL procese u dijelu učitavanja podataka iz izvora podataka u skladište podataka. U radu se navodi da predstavljeni paterni pokrivaju 98% od ETL. Ovi paterni, prema Linstedu, omogućiće da ETL rutine (bez alata) budu ponovljive, skalabilne i sa dobrim performansama.



U dokumentu nije pomenuta mogućnost automatizacije ETL procesa kod skladišta podataka baziranog na Data Vault konceptu, niti algoritmi za njenu realizaciju. Međutim, paterni dati u radu [Lin12] i na slikama 35. i 36. daju dobru osnovu za realizaciju automatizacije ETL procesa u dijelu učitavanja podataka iz izvora podataka u skladište podataka.



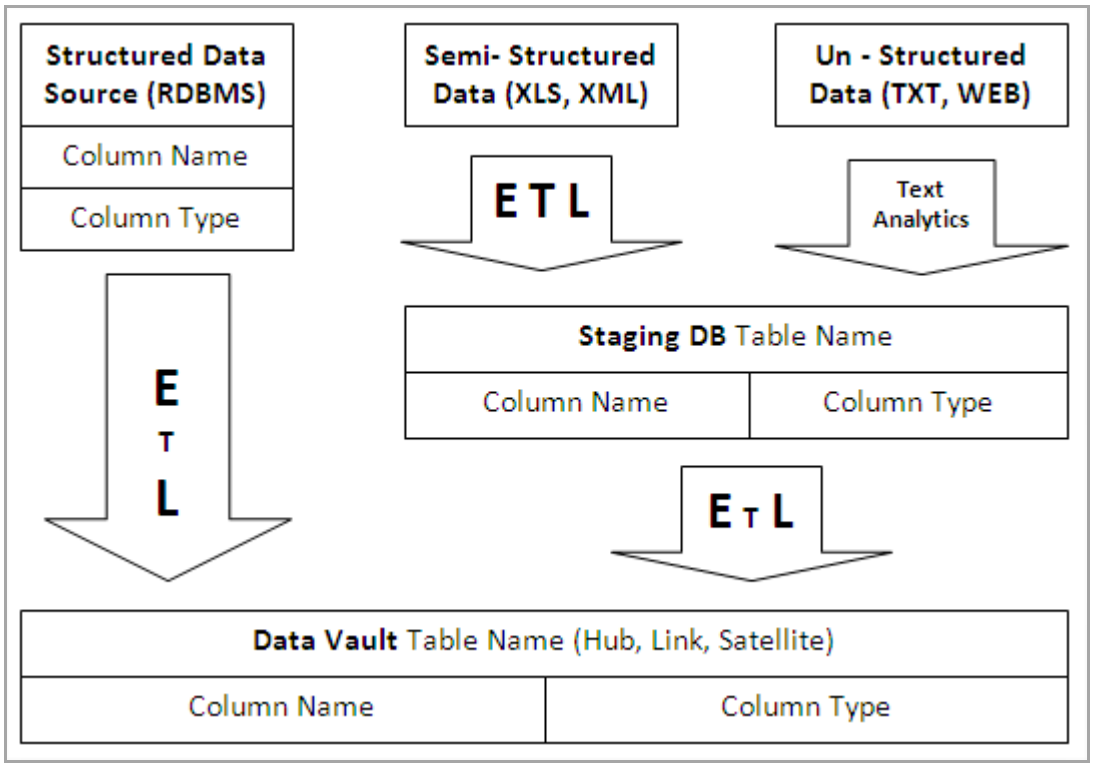
Postoji mnogo pristupa modelovanja i/ili generisanja ETL procesa i koda. Osim toga, postoje i mnogi komercijalni alati za ETL dizajn ([IBM1], [Inform], [Micro], [Oracle]). Ipak nijedan od navedenih pristupa i alata ne razmatra formalizaciju automatizacije ETL procesa skladišta podataka u kontekstu Data Vault koncepta.

4.4.1. Automatizacija ETL procesa: OLTP - DW

Pristup u ovoj disertaciji predlaže projektovanje ETL procesa u ranim fazama projektovanja skladišta podataka. Prilikom automatskog generisnja

tabela skladišta podataka iz izvora podataka, predlaže se generisanje fizičkog modela metapodataka i šeme mapiranja izvora podataka i tabela skladišta podataka baziranog na Data Vault konceptu. Isto tako predlaže se generisanje fizičkog modela metapodataka data martova prilikom automatskog kreiranja tabela mjera i dimenzija i mapiranje tabela Data Vaulta sa tabelama data marta.

U ETL (ELT) procesu kod skladišta podataka baziranog na Data Vault konceptu, podatke iz struktuiranih izvora podataka (baze podataka) moguće je direktno učitati u skladište podataka. Podatke iz polustruktuiranih izvora podataka potrebno je najprije učitati u prelaznu bazu podataka, dok podaci iz nestruktuiranih izvora podataka trebaju najprije proći kroz proces tekstualne analize radi struktuiranja i učitavanja u prelaznu bazu podataka. Ovi podaci se iz prelazne baze podataka u ETL procesu učitavaju u skladište podataka.



Slika 37. Postupak generisanja ETL koda na osnovu mapiranja metapodatka izvora i skladišta podataka baziranog na Data Vaultu

U fazi automatizacije ETL (ELT) procesa u slučaju struktuiranih izvora podataka kod skladišta podataka baziranog na Data Vault konceptu predlažu se sledeće međufaze:

1. Identifikacija metapodataka hub, link i satelit tabela. Metapodaci ovih tabela se nalaze u tabeli *DV_TableColumns* koja je kreirana u prethodnoj međufazi prilikom dizajna data marta. U najkraćem ova međufaza se sastojala od sljedećih koraka:
    - identifikacija metapodataka hub tabela
    - identifikacija metapodataka link tabela
    - identifikacija metapodataka satelit tabela
2. Ekstrakcija podataka iz izvora podataka (OLTP tabele, polustruktuirani i nestruktuirani podaci) i minimalna transformacija (ako je potrebno), gdje su predviđeni sljedeći koraci:
    - ekstrakcija podataka iz struktuiranih izvora podataka
    - ekstrakcija podataka iz polustruktuiranih izvora podataka
    - ekstrakcija podataka iz nestruktuiranih izvora podataka
3. Učitavanje (engl. *load*) podataka u hub, link i satelit tabele. U okviru ove međufaze predviđeni su sljedeći koraci:
    - učitavanje podataka u hub tabele
    - učitavanje podataka u link tabele
    - učitavanje podataka u satelit tabele

Dijagram toka se u može predstaviti kao na sljedećoj slici.

Slika 38. Dijagram toka treće faze, ETL (ELT) proces Data Vaulta

S obzirom da je prvu međufazu moguće uraditi u ranoj fazi dizajna skladišta podataka u ovom dijelu rada potrebno je opisati ostale dvije međufaze: ekstrakcija podataka iz izvora podataka i učitavanje podataka u Data Vault.

4.4.1.1. Automatizacija ELT procesa: OLTP - Data Vault

U ETL (ELT) procesu kod skladišta podataka baziranog na Data Vault konceptu, podatke iz struktuiranih izvora podataka (baze podataka) moguće je direktno učitati u skladište podataka. Podatke iz polustruktuiranih izvora podataka potrebno je najprije učitati u prelaznu bazu podataka, dok podaci iz nestruktuiranih izvora podataka trebaju najprije proći kroz proces tekstualne analize radi struktuiranja i učitavanja u prelaznu bazu podataka. Ovi podaci se iz prelazne baze podataka u ETL procesu učitavaju u skladište podataka.

Generisanje skripta za inicijalno učitavanje podataka u hub tabele obavlja se pomoću kursora (nad tabelom *DV_TableColumns*) na sledeći način i u sledećim koracima (uskladištena procedura *sp_InsertHub*):

a) formiranje kursora za prolazak kroz *DV_TableColumns* tabelu za hub tabele

b) preuzimanje podataka iz tabele i dodjeljivanje promjenljivim

c) u svakoj iteraciji dopunjavanje sql_statement-a (za insert podataka) i

d) izvršavanje sql_statement-a, čime se insertuju podaci u hub tabele

U nastavku je dat primjer generisanog SQL skripta za učitavanje podataka u tabelu Hub_Customers (iz baze Northwind kao izvora podataka):

```
INSERT INTO Hub_Customers(CustomerID,LoadDate,RecordSource)
SELECT CustomerID,GETDATE(),'Northwind' FROM
Northwind.dbo.Customers WHERE NOT EXISTS (SELECT 1 FROM
Hub_Customers WHERE
Hub_Customers.CustomerID=Northwind.dbo.Customers.CustomerID
COLLATE latin1_general_ci_as)
```

Generisanje skripta za inicijalno učitavanje podataka u link tabele obavlja se pomoću kursora (nad tabelom *DV_TableColumns*) na sledeći način i u sledećim koracima (uskladištena procedura *sp_InsertLink*):

a) formiranje kursora za prolazak kroz *DV_TableColumns* tabelu za Link tabele

b) preuzimanje podataka iz tabele i dodjeljivanje promjenljivim

c) u svakoj iteraciji dopunjavanje sql_statement-a (za insert podataka) i

d) izvršavanje sql_statement-a, čime se insertuju podaci u Link tabele

Generisanje skripta za inicijalno učitavanje podataka u link tabele obavlja se pomoću kursora (nad tabelom *DV_TableColumns*) na sledeći način i u sledećim koracima (stored procedura *sp_InsertSat*):

a) formiranje kursora za prolazak kroz *DV_TableColumns* tabelu za Link tabele

b) preuzimanje podataka iz tabele i dodjeljivanje promjenljivim

c) u svakoj iteraciji dopunjavanje sql_statement-a (za insert podataka) i

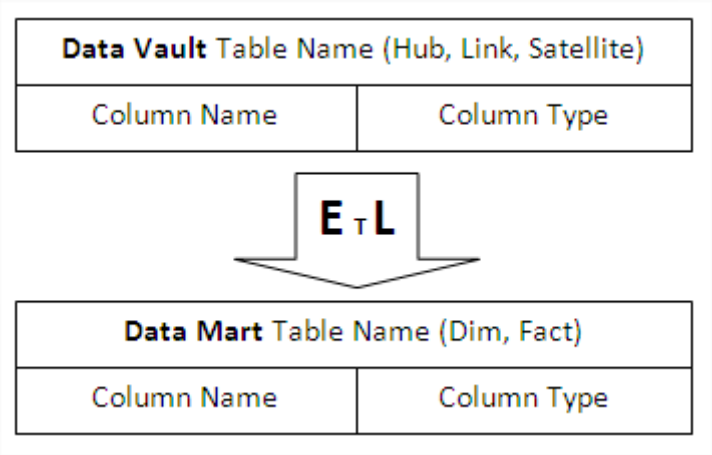
d) izvršavanje sql_statement-a, čime se insertuju podaci u Link tabele

Jedna od bitnih ideja Data Vault koncepta, da bi u potpunosti obezbedili potpune informacije istorijskih podataka o prošlosti i sadašnjosti, je da se poslije ekstrakcije podataka iz izvora podataka, radi učitavanje podataka u skladište podataka. Minimalna transformacija (ukoliko je potrebna) radi se u skladištu podataka.

Opisanim postupcima u ovom poglavlju moguće je učitati sve tabele skladišta podataka sa podacima iz izvora podataka sa minimalnim učešćem korisnika. SQL kod uskladištenih procedura koje se koriste u ovoj fazi dat je na CD-u u prilogu disertacije.

4.4.2. Automatizacija ETL procesa: DW – data mart

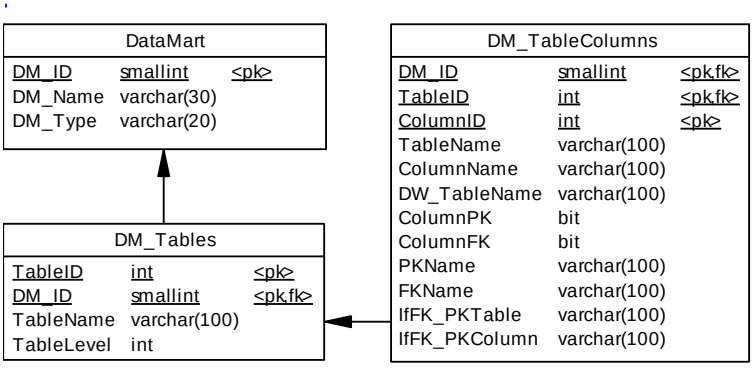
Dimenzije u star šemi nastaju iz hub i satelit tabela, a činjenice (fact) iz satelit i link tabela [Gov10]). S obzirom da se poslovni ključevi iz hub tabela i zavisni atributi iz satelit tabela transformišu u dimenzije, te da atributi koji imaju numeričke vrijednosti (prema prethodnim razmatranjima u 4.3.) postaju poslovne mjere u odgovarajućim fact tabelama.



Slika 39. Postupak generisanja ETL koda na osnovu mapiranja metapodatka skladišta podataka baziranog na Data Vaultu i metapodataka Data Marta

Identifkacija metapodataka data marta može se uraditi u ranoj fazi dizajna skladišta podataka, odnosno u fazi dizajna data martova prilikom kreiranja tabela mjera i dimenzija data marta. Kod izrade modela metapodataka data marta razlikujemo dio modela koji je nezavisan od izvora podataka i dio modela koji zavisi od izvora podataka. Nezavisni dio se odnosi na tabele metapodataka i pravila, a zavisni se odnosi na procedure vezane za različite platforme za skladište podataka. Namjena i način ažuriranja tabele metapodataka prikazan u tabeli 12.

RuleTypes tabela sadrži podatke o vrstama pravila, kao što su pravila za ekstrakciju podataka za tabele mjera i dimenzija ili pravila za učitavanje podataka u tabele mjera i dimenzija. Tabela Rule sadrži informacije o pravilima za određenu vrstu podataka i postupke koje treba izvršiti kada se određeni uslov zadovoljen. Pravila u koloni Rules omogućuju lakše izvođenje naredbi koje se čuvaju u koloni RuleAction. Kreiranje tabela metapodataka odgovarajuće procedure za insert metapodataka tabelu.



Slika 40. Model metapodataka data marta

U ovoj fazi mogu se koristiti i tabele pravila koja su kreirana u prvoj fazi (automatizacija kreiranja Data Vault skladišta podataka). Fizički model pravila dat je na slici 27.

Tabela 12. Namjena i način popunjavanja DM tabela metapodataka

Tabela	Namjena	Način ažuriranja podataka
DataMart	Naziv data marta	Automatski nakon kreiranja tabela mjera i dimenzija
DM_Tables	Tabele data marta	Automatski nakon kreiranja tabela mjera i dimenzija
DM_TableColumns	Kolone pojedinih tabela data marta	Automatski nakon kreiranja tabela mjera i dimenzija

Kako je značenje kolona u tabelama na slici 40 DataMart i DM_Tables intuitivno i jasno, u Tabeli 13 se opisuje struktura i namjena kolona tabele DM_TableColumns. Pristup u radu podrazumijeva automatsko učitavanje metapodataka u tabelu DM_TableColumns.

Tabela 13. Struktura DM_TableColumns tabele

Kolona	Tip podatka	Opis	Podaci se generišu	Podatke unosi korisnik
TableName	varchar	Ime Fact ili Dim tabele	✓	
DW_TableName	varchar	Ime odgovarajuće tabele iz skladišta podataka (hub, link ili satelit)	✓	
ColumnName	varchar	Ime kolone posmatrane tabele	✓	
ColumnId	integer	ID kolone posmatrane tabele	✓	
DataType	varchar	Tip podatka	✓	
ColumnPK	bit	Da li je kolona primarni ključ?	✓	
ColumnFK	bit	Da li je kolona strani ključ?	✓	
PKName	varchar	Ime primarnog ključa (ako je kolona PK)	✓	
FKName	varchar	Ime stranog ključa (ako je kolona FK)	✓	
IfFK_PKTable	varchar	Ime odgovarajuće tabele na strani primarnog ključa (ako je kolona strani ključ)	✓	
IfFK_PKColumn	varchar	Ime odgovarajuće kolone u tabeli primarnog ključa (ako je kolona strani ključ)	✓	

U fazi učitavanja podataka iz skladišta podataka u data mart predlažu se sljedeće međufaze:

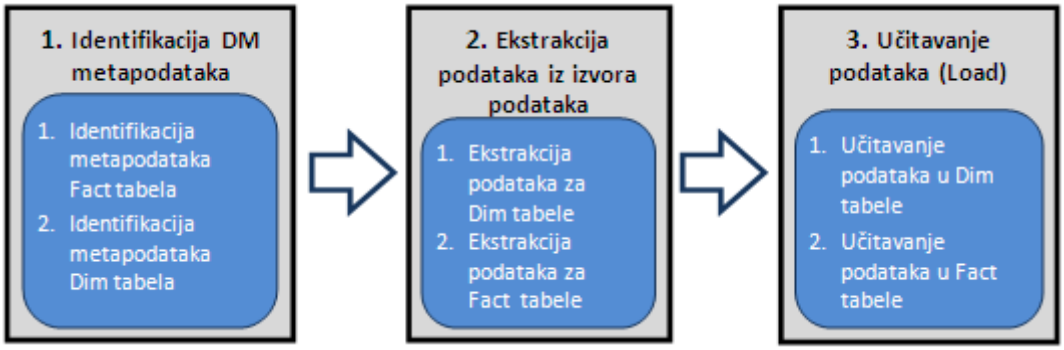
1. **Identifikacija metapodataka fact i dimezionih tabela**. Metapodaci ovih tabela se nalaze u tabeli *DM_TableColumns* koja je kreirana u prethodnoj

međufazi prilikom dizajna data marta. U najkraćem ova međufaza se sastojala od sljedećih koraka:

- identifikacija metapodataka fact tabela
- identifikacija metapodataka dimenzionih tabela

2. **Ekstrakcija podataka iz Data Vault skladišta podataka** i minimalna transformacija (ako je potrebno), gdje su predviđeni sljedeći koraci:
   - ekstrakcija podataka za tabele dimenzija
   - ekstrakcija podataka za fact tabele
3. **Učitavanje (eng. *load*) podataka u dimenzione i fact tabele**. U okviru ove međufaze predviđeni su sljedeći koraci:
   - učitavanje podataka u dimenzione tabele
   - učitavanje podataka u fact tabele

Dijagram toka se u može predstaviti kao na sljedećoj slici.



Slika 41. Dijagram toka četvrte faze, ETL (ELT) proces data marta

4.4.2.1. Automatizacija ETL procesa: Data Vault – data mart

Generisanje skripta za inicijalno učitavanje podataka u tabele dimenzija obavlja se pomoću kursora (nad tabelom *DM_TableColumns*) na sledeći način i u sledećim koracima:

a) formiranje kursora za prolazak kroz *DM_TableColumns* tabelu za dimenzione tabele
b) preuzimanje podataka iz tabele i dodjeljivanje promjenljivim
c) u svakoj iteraciji dopunjavanje sql_statement-a (za insert podataka) i
d) izvršavanje sql_statement-a, čime se insertuju podaci u dimenzione tabele

Učitavanje podataka u fact tabele daleko je složenije od učitavanja podataka u tabele dimenzija s obzirom da je potrebno korištenje agregatnih funkcija (sum, average, maximum, minimum, i dr.) i spajanje više tabela. Samim tim i generisanje skripta za automatsko učitavanje podataka u fact tabele predstavlja složeniji proces. Prema [GMR98], u nekim slučajevima agregacije nije potrebno definisati budući da je agregacija već izvršena na relacionom nivou. Na primjer, svaku instancu entiteta Prodaja u relacionoj šemi se može opisati ukupna prodaje za dati proizvod, koja uključuje i vrijeme. U ovom slučaju atributi iz relacionog modela mogu biti direktno prevedeni na atribut činjenice (fact). Kod kreiranja Data Marta, pored identifikacije mjera i dimenzija potrebno da korisnik odabere mjere koje će se agregirati i način agregacije mjera.

Sintaksa SQL izraza za inicijalno punjenje fact tabele se može definisati na sljedeći način:

```
INSERT INTO <naziv fact tabele> (<kolone fact tabele>)
SELECT  <kolone DV tabela>,
<agregatne funkcija nad kolonama DV tabela>
FROM <DV tabele koje učestvuju u punjenju fact tabele>
WHERE <uslov obuhvata podataka>
GROUP BY <kolone DV tabela nad kojima se vrši grupisanje>
```

Tabela 14. Značenja pojedinih SQL izraza kod generisanja scripta za učitavanje podataka u Fact tabele

SQL statement	Opis SQL izraza
InsertIntoStatement	<naziv fact tabele> (<kolone fact tabele>)
SelectStatement	<kolone DV tabela>
SumStatement	<agregatne funkcija nad kolonama DV tabela>
FromStatement	<DV tabele koje učestvuju u punjenju fact tabele>
WhereStatement	<uslov obuhvata podataka>
GroupByStatement	<kolone DV tabela nad kojima se vrši grupisanje>

Generisanje skripta za inicijalno učitavanje podataka u fact tabele na sledeći način i u sledećim koracima:

a) generisanje izraza FromStatement izraza na osnovu relacija između tabela kolona u Fact i Dim tabelama i odgovarajućih kolona u skladištu podataka
b) generisanje izraza WhereStatement izraza na osnovu relacija između tabela kolona u Fact i Dim tabelama i odgovarajućih kolona u skladištu podataka
c) generisanje InsertIntoStatement izraza pomoću kursora nad *DM_TableColumns* tabelom
d) generisanje SelectStatement izraza pomoću kursora nad *DM_TableColumns* tabelom
e) generisanje SumStatement izraza pomoću kursora nad *DM_TableColumns* tabelom i pomoću podataka iz tabele FactsAgregation
f) generisanje GroupByStatement izraza pomoću kursora nad *DM_TableColumns* tabelom
g) izvršavanje objedinjenog SQL izraza

Na sljedećem primjeru prikazan je SQL skript za inicijalno učitavanje podataka u fact tabelu koja je generisana iz Northwind baze. U ovom primjeru Fact tabela

je *Fact_Sat_Order_Details* a dimenzione tabele su: *Dim_Products, Dim_Customers i DimDateTime*.

```
INSERT INTO [Fact_Sat_Order Details] (ProductID, CustomerID,
DateTimeID, TotalPrice, TotalQuantity)
SELECT  [Hub_Products].ProductID, [Hub_Customers].CustomerID,
[Sat_Orders].DateTimeID,
ISNULL (SUM([Sat_Order Details].UnitPrice*[Sat_Order
Details].Quantity),0) AS TotalPrice,
ISNULL (SUM([Sat_Order Details].Quantity),0) AS TotalQuantity
FROM [Hub_Customers],[Hub_Products],[Lnk_Order
Details],[Lnk_Orders],[Sat_Order Details],[Sat_Orders]
WHERE [Hub_Customers].CustomerID = [Lnk_Orders].CustomerID AND
[Hub_Products].ProductID = [Lnk_Order Details].ProductID AND
[Lnk_Order Details].OrderID = [Sat_Order Details].OrderID AND
[Lnk_Order Details].ProductID = [Sat_Order Details].ProductID AND
[Lnk_Orders].OrderID = [Lnk_Order Details].OrderID AND
[Lnk_Orders].OrderID = [Sat_Orders].OrderID
GROUP BY  [Hub_Products].ProductID, [Hub_Customers].CustomerID,
[Sat_Orders].DateTimeID
```

Opisanim postupcima u poglavljima 4.4.2. moguće je generisati kompletan SQL kod za učitavanje podataka iz Data Vault skladišta podataka u data mart, odnosno u tabele mjera i dimenzija. Generisani kod omogućuje automatizaciju ekstrakcije podataka iz Data Vaulta učitavanje podataka u data mart. SQL kod uskladištenih procedura koje se koriste u ovoj fazi dat je na CD-u u prilogu disertacije.

5. PROTOTIP APLIKACIJE ZA PODRŠKU PROJEKTOVANJA SKLADIŠTA PODATAKA

Prema [MP04], prototipovi se koriste za kontinuirano dobijanje povratnih informacija radi daljeg profilisanja zahtjeva od strane korisnika i radi provjere koncepata. Prototipovi se primjenjuju: za demonstriranje poslovnog procesa koji se automatizuje, za ispitivanje lakoće korišćenja korisničkog interfejsa, za ispitivanje performansi i kapaciteta sistema i za ispitivanje ispravnosti razvojnih odluka i koncepata. Zahvaljujući direktnoj komunikaciji sa korisnicima baziranoj na prototipovima, nisu neophodne dugačke analize zahtjeva koje korisnici potpisuju i koje je potrebno ažurirati tokom promjena zahtjeva [Mil10] [MP04].

Razvoj prototipa aplikacije Direct PDV za podršku automatizaciji fizičkog projektovanja skladišta podataka urađen je korišćenjem sljedećih značajnijih koncepata:

- Larmanova metoda razvoja softvera
- Relacioni model podataka
- Troslojna arhitektura aplikacije
- Desktop aplikacija
- Dinamički SQL i generator programskog koda
- Data Vault (opisan u poglavlju 2.8. disertacije)
- Direktni pristup projektovanja skladišta podataka (opisan u poglavlju 4. disertacije)

5.1. Larmanova metoda razvoja softvera

Larmanova metode razvoja softvera je detaljno opisana u [Lar98]. Ova metoda je objektno orijentisana metoda razvoja softvera. Objektno orijentisana metoda omogućuje razvoj aplikacije da na jednostavan način obezbeđuje uvid u analizu i implementaciju svim učesnicima u razvoju softvera. Larmanova metoda za notacije koristi UML (*Unified Modeling Language*) dijagrame. UML predstavlja grafički jezik za specifikaciju, vizuelizaciju, konstrukciju i dokumentovanje artifakta softverskih sistema bez obzira na implementacionu platformu.

Osnovne faze Larmanove metode razvoja softvera su [Lar98][Vla04]:

1. Zahtjevi predstavljaju osobine i uslove koje sistem ili projekat mora da zadovolji. Zahtjevi se opisuju pomoću modela slučaja korišćenja (engl. *use-case model*) koji predstavljaju skup željenih korišćenja sistema od strane krajnjih korisnika. U ovoj fazi se definišu svi učesnici u sistemu i slučajevi korišćenja. Karakteristika ove faze je intenzivna saradnja između projektanta i osobe koja dobro poznaje domen problema. Ova faza najčešće počinje tekstualnim opisom zahtjeva radi formiranja slučajeva korišćenja koji se predstavljaju tekstualno i grafički. Slučajevi korišćenja definišu interakciju korisnika sa sistemom, koje opisuje skup od jednog osnovnog i više alternativnih scenarija. Scenario predstavlja skup željenih korišćenja od strane korisnika sistema [Lar98].

„Akter predstavlja spoljnog korisnika sistema. On postavlja zahtjev sistemu da izvrši jednu ili više sistemskih operacija, po unaprijed definisanom scenariju. Sistem odgovara na postavljeni zahtev aktera, šaljući mu vrijednost izlaznih argumenata kao rezultat izvršenja operacija. Akter se obično definiše kao neko

ili nešto (npr: ljudi, kompjuterski sistemi ili organizaciona jedinica) što ima ponašanje“ [Vla04].

Prema [LMAB08], model slučajeva korišćenja se koristi za izgradnju poslovnog modela sistema i za specifikaciju aplikacija. Slučaj korišćenja i način izgradnje modela slučajeva korišćenja se razlikuje za ova dva aspekta. Sa tačke gledišta analize sistema i izgradnje njegovog poslovnog modela, slučaj korišćenja se definiše kao specifikacija interakcije između sistema i jednog ili više aktera i sistema, zajedno sa opisom akcija sistema u ovoj interakciji. Sa tačke gledišta specifikacije aplikacije, slučaj korišćenja se definiše kao specifičan način na koji će akter da koristi jednu buduću aplikaciju informacionog sistema koji se projektuje. Akter se i u jednom i drugom slučaju definiše kao uloga koju igra neki entitet van sistema u jednoj ili više interakcija sa sistemom [LMAB08].

Slučajevi korišćenja (SK) se u početnim fazama razvoja softvera predstavljaju tekstualno dok se kasnije oni predstavljaju preko sekvencijalnih dijagrama, dijagrama saradnje, dijagrama prelaza stanja ili dijagrama aktivnosti. Tekstualni opis SK najčešće ima sledeću strukturu [Lar98, JBR99, Vla04]:

- Naziv SK
- Aktere SK
- Učesnike SK
- Preduslovi koji moraju biti zadovoljeni da bi SK počeo da se izvršava
- Osnovni scenario izvršenja SK.
- Postuslovi koji moraju biti zadovoljeni da bi se potvrdilo da je SK uspešno izvršen
- Alternativna scenarija izvršenja SK
- Specijalni i tehnološki zahtjevi
- Otvorena pitanja

Jednu akciju scenarija izvodi ili akter ili sistem. U tom smislu prema [Vla04]:

1. **Akter** izvodi jednu od tri vrste akcija:
   a) **Akter** **P**riprema **U**laz (**U**lazne **A**rgumente(**UA**)) za **S**istemsku Operaciju (**APUSO**)
   b) **Akter** **P**oziva sistem da izvrši **S**istemsku **O**peraciju(**APSO**). Sistemskoj operaciji se prosleđuju ulazni argumenti. Nakon toga se izvršava **S**istemska **O**peracija (**SO**) koja kao rezultat daje jedan ili više izlaznih argumenata (**IA**)
   c) **Akter** izvršava **NeS**istemsku **O**peraciju(**ANSO**)
2. Na osnovu 1.b) može se zaključiti da **Sistem** izvodi dvije akcije u kontinuitetu:
   a) Sistem izvršava **S**istemsku **O**peraciju(**SO**)
   b) Rezultat sistemske operacije (Izlazni argumenti(**IA**)) se prosleđuje do Aktera

2. Analiza. Prema [Vla04]: „Faza analize opisuje logičku strukturu i ponašanje softverskog sistema (poslovnu logiku softverskog sistema). Ponašanje softverskog sistema je opisano pomoću sistemskih dijagrama sekvenci, koji se prave za svaki slučaj korišćenja, i pomoću ugovora o sistemskim operacijama, koje se dobijaju na osnovu sistemskih dijagrama sekvenci. Slučajevi korišćenja pokazuju na koji način akter komunicira sa softverskim sistemom. U toku interakcije sa sistemom akter pokreće događaje u sistemu i traži od sistema da izvrši određene operacije.“ Da bismo ilustrovali operacije između aktera i sistema, definišemo dijagram sekvenci koji prikazuje samo izdvojeni scenario slučaja korišćenja koji se odvijaju između učesnika i sistema [Lar98][Vla04]. U fazi analize slučajevi korišćenja opisuju cjelokupan sistem, dok u fazi specifikacije daju detaljan opis jedne aplikacije [LMAB08].

Struktura softverskog sistema se opisuje pomoću konceptualnog modela. Konceptualni (domenski) model sistema opisuje konceptualne klase domena problema i asocijacije među njima. Konceptualne klase predstavljaju atribute softverskog sistema, što znači da opisuju njegovu strukturu. Konceptualni model služi kao osnova za izradu relacionog modela i šeme baze podataka [Vla04]. Realacioni model je još uvijek najpopularniji model i osnova najvećeg broja komercijalnih sistema za upravljanje bazom podataka [LMAB08].

3. **Projektovanje** prema [Lar98][Vla04]: „opisuje fizičku strukturu i ponašanje softverskog sistema (arhitekturu softverskog sistema). Projektovanje arhitekture softverskog sistema obuhvata projektovanje aplikacione logike, skladišta podataka i korisničkog interfejsa. U okviru aplikacione logike se projektuju kontroler, poslovna logika i database broker. Projektovanje poslovne logika obuhvata projektovanje logičke strukture i ponašanja softverskog sistema. Kao ulaz koriste se rezultati iz prethodne faze: konceptualni model i sistemske operacije i njihovi ugovori.“ Izlaz iz ove faze su stvarni slučajevi korišćenja, dijagrami saradnje, dijagrami sekvenci za sistemske operacije i dijagrami klasa koji obuhvataju sve klase iz aplikacije i njihove relacije i metode [Vla04].

4. **Implementacija.** U ovoj fazi se vrši kodiranje softverskog sistema u nekom od objektno-orijentisanih programskih jezika. U ovom slučaju korišćen je Microsoft C# jezik kao dio Microsoft Visual Studio .NET 2010 okruženja.
Svaka klasa iz dijagrama se implementira po unaprijed definisanom redoslijedu: prvo se implementiraju klase koje ne zavise od drugih klasa, pa se u svakom sledećem ciklusu implementiraju samo one klase koje zavise isključivo od već implementiranih. Izlaz iz ove faze je program [Vla04].

5. Testiranje. U zavisnosti od arhitekture softverskog sistema, ova faza se može podijeliti u nekoliko nezavisnih dijelova testiranja. Nezavisne jedinice testiranja su softverske komponente koje smo dobili u fazi implementacije softverskog sistema. Testiranje svake softverske komponente podrazumeva sljedeće [Lar98]:
a) test slučajeva koji opisuju šta test treba da provjeri,
b) test procedura koje opisuju kako će se izvršiti test i
c) test komponente koje treba da automatizuju test procedure ukoliko je to moguće. Nakon testiranja softverskih komponenti vrši se njihova integracija. U tom smislu se prave integracioni testovi softverskih komponenti.

5.2. Relacioni model podataka

Model podataka je specifičan teorijski okvir pomoću koga se specifikuje, projektuje i implementira neka konkretna baza podataka. Model podataka predstavlja osnovu za izgradnju sistema za upravljanje bazom podataka. Relacioni model se predstavlja preko skupa tabela (relacija). Tabele predstavljaju tipove objekata, a kolone predstavljaju atribute objekata. Vrste predstavljaju pojavljivanje objekata, odnosno veza. Jedan ili više atributa tabele jedinstveno identifikuju jednu vrstu tabele i taj atribut se naziva primarni ključ relacije. Većina postojećih baza podataka izgrađena je preko sistema za upravljanje bazama podataka zasnovanih na relacionom modelu [LMBA08].

SQL (*Structured Query Languague*) je standardni relacioni jezik baza podataka. Preko njega se definišu strukture relacionog modela (skup tabela i njihovih atributa, ključeva i drugo), skup ograničenja vrednosti atributa i dinamička pravila integriteta. Jednostavnost strukture relacionog modela (skup ravnih tabela) i jednostavan jezik za manipulaciju nad njima učinili su relacioni model najpopularnijim implementacionim modelom baza podataka [LMBA08].

5.3. Troslojna arhitektura aplikacije

Višeslojna arhitektura aplikacije (često označena kao n-slojna ili n-nivojska) je arhitektura u kojoj su sloj prezentacije, sloj obrade podataka i skladište podataka logički odvojeni. Najrasprostranjeniji oblik višenivojske arhitekture je tronivojska arhitektura, koja se smatra istovremeno i arhitekturom sistema i uzorom projektovanja [Vla04] [Sch95] [TK78].

Troslojna arhitektura se sastoji od sledećih slojeva [Vla04] [TK78]:

1. Korisničkog interfejsa koji predstavlja ulazno – izlaznu reprezentaciju softverskog sistema
2. Aplikacione logike koja opisuje strukturu i ponašanje softverskog sistema
3. Skladišta podataka koji čuva stanje atributa softverskog sistema

Aplikaciona logika se projektuje nezavisno od korisničkog interfejsa i može da ima različite ulazno-izlazne reprezentacije. Na osnovu troslojne arhitekture su napravljeni savremeni aplikacioni serveri. Aplikacioni serveri su odgovorni da obezbede servise koji će da omoguće realizaciju aplikacione logike softverskog sistema. Svaki aplikacioni server se sastoji iz tri osnovna dijela [Vla04]:

1. Dio za komunikaciju sa klijentom (kontroler)
2. Dio za komunikaciju sa bazom podataka
3. Dio koji sadrži poslovnu logiku

Aplikacija se može realizovati i u dvoslojnoj arhitekturi gdje su prezentacioni sloj (korisnički interfejs) i sloj poslovne logike objedinjeni unutar desktop aplikacije (tzv. “debeli klijent”). Ovakva aplikacija poziva funkcije biblioteke za pristup bazi podataka (ODBC, JDBC, OLEDB, ADO) koja se takođe nalaze na klijentu. Prema [LMBA08], osnovni nedostatak dvoslojne arhitekture je problem

održavanja velikog broja istih kopija aplikativnog softvera na svim klijentima. Međutim, u ovom slučaju radi se najčešće o jednom korisniku (dizajneru skladišta podataka) tako da bi se izbjegao taj nedostatak.

U izradi aplikacije se mogu koristiti uskladištene procedure koje omogućuju da se dio poslovne logike smjesti u bazu podataka. Ovakvi slučajevi se prije svega realizuju kod komplikovanijih transakcija koje zahtijevaju razne provjere i ažuriraju više od jedne tabele baze podataka. Prednost ovakve implementacije se ogleda prije svega u mogućnosti da se u slučaju promjene poslovne logike izvrši izmena samo uskladištene procedure, bez izmjene aplikacije.

5.4. Desktop aplikacija

Desktop aplikacija je aplikacije koja se pokreće na klijentskom (desktop) računaru. Desktop aplikacije imaju složeniji korisnički interfejs nego web aplikacije. Ove aplikacije uglavnom ne zahtijevaju pristup internetu mada ga mogu koristiti. Iako su danas popularnije web aplikacije, desktop aplikacije nalaze svoju primjenu u mnogim aplikativnim rješenjima, tako da još uvijek postoje klijentski programi koji zbog svojih karakteristika neće moći biti napravljeni kao web aplikacije. Karakteristike desktop aplikacija u odnosu na web aplikacije su [Micro1]:

- kontrole koje čine korisnički interfejs kod desktop aplikacija reaguju na više događaja nego kod web aplikacija,
- kod desktop aplikacija može se smanjiti broj pozivanja servera (roundtrip-ova), s tim da sličnu funkcionalnost omogućava Ajax koncept
- kod desktop aplikacija je moguće postići veći stepen interakcije sa korisnikom.

Osim toga, desktop aplikacije se koriste u slučaju kad je interfejs veoma složen pa su mu potrebne kontrole koje nisu dostupne u web aplikacijama. Osim toga, koriste se i kad se obrada podataka vrši na klijentskom računaru da bi se smanjio mrežni saobraćaj. Najznačajnija zajednička karakteristika desktop i web aplikacija je da im je dostupno optimističko zaključavanje, te da se web servisi mogu napraviti kao desktop i kao web aplikacija [Micro1].

5.5. Dinamički SQL i generator programskog koda

Dinamički SQL omogućava pisanje programa u kome SQL naredba nije poznata sve do vremena izvršenja programa. Ideja dinamičkog SQL-a se sastoji u tome da se, umjesto pisanja eksplicitne naredbe u programu, tekst SQL naredbe zapamti kao vrijednost neke promjenljive. Poslije toga se traži izvršavanje naredbe koja je sadržana u promjenljivoj [LMBA08].

Automatsko programiranje predstavlja kompjutersko programiranje u kome neki mehanizam generiše program umesto čovjeka. Generativno programiranje je programiranje koje koristi automatizovano kreiranje programskog koda preko generičkih frejmova, klasa, prototipa, skica, itd. Generisanje programskog koda je bazirano na ontološkom modelu koji se preko nekog programerskog alata ili integrisanog razvojnog okruženja (engl. *IDE*) transformiše u programski kod [BFYV]. Neki IDE sadrže razne forme generisanja programskog koda kao što su čarobnjaci (engl. *wizard*) kao niz dijaloga vodi korisnika kroz korake kojima se odabire željena akcija, snipeti (engl. *snippets*) kao relativno mali, ponovljivi dijelovi koji se umeću u veće programske module, i refaktorizacija koja podrazumijeva promjenu unutrašnje strukture programa, bez promene njegove spoljašnje funkcionalnosti u cilju poboljšanja

kvaliteta softvera, performansi, ili proširivanja funkcionalnosti. Prema [Mil10][MP04], uvođenje generatora koda može biti od višestruke pomoći:

- Tim za razvoj oslobađa straha da neće biti vremena da se napiše kod, ostaje više vremena da se posveti otkrivanju znanja o problemu i analizi korisničkih zahtjeva,
- Članovi tima su zadovoljniji jer mogu veći procenat vremena da posvete kreativnom poslu,
- Brzina rada se znatno povećava,
- Generator aplikacija forsira poštovanje programerskih i HCI (*Human Computer Interaction*) standarda, što bez takve podrške može biti veoma teško (u okviru [Els03] je istaknuto da je tim, kada su se rokovi primakli, jednostavno zanemario HCI standard, što je dovelo do nekonzistentnih izgleda ekrana)

5.6. Razvoj aplikacije Direct PDV

Na osnovu prethodnih razmatranja u ovom dijelu je opisan sam proces razvoja aplikacije Direct PDV za podršku automatizacije fizičkog projektovanja skladišta podataka. Ulaz u sistem će predstavljati model metapodataka izvora podataka, a izlaz desktop aplikacija koja će omogućiti podršku automatizaciji skladišta podataka.

5.6.1. Zahtjevi

Kod Larmanove metode razvoja softvera korisnički zahtjevi se opisuju verbalno i pomoću modela slučajeva korišćenja [Lar98].

5.6.1.1. Verbalni opis

Potrebno je napraviti aplikaciju koja će obezbjediti što veću automatizaciju u procesu projektovanja skladišta podataka baziranog na Data Vault konceptu. Aplikacija treba da ima sljedeće module:

- modul koji obezbeđuje podršku projektovanja samog skladišta podataka i koji pruža podršku automatskom generisanju ETL koda za učitavanje podataka iz izvora podataka u skladište podataka,
- modul koji obezbeđuje podršku projektovanja data marta u obliku zvijezda šeme iz skladišta podataka baziranog na Data Vault konceptu i koji pruža podršku automatskom generisanju ETL koda za učitavanje podataka iz skladišta podataka u data mart
- modul koji omogućuje kreiranje tabela skladišta podataka na osnovu polustruktuiranih izvora i učitavanje podataka iz polustruktuiranih izvora u skladište podataka i iz skladišta podataka u data mart
- modul koji omogućuje kreiranje i pregled fizičkih modela izvora podataka, skladišta podataka i data marta

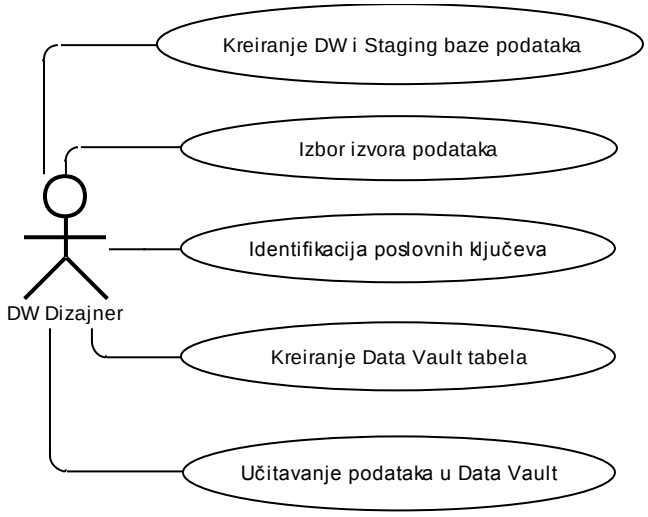
Skladište podataka treba omogućiti analizu podataka iz struktuiranih i jednostavnijih polustruktuiranih izvora podataka. Struktuirane podatke nije potrebno učitavati u prelaznu bazu nego direktno u skladište podataka. Polustruktuirane podatke je potrebno dodatno struktuirati u prelaznoj bazi podataka a zatim učitati u skladište podataka. Nestruktuirani podaci trebaju proći proces tekstualne analize, zatim se učitati u prelaznu bazu podataka radi struktuiranja, a zatim učitati u skladište podataka. Alat ne treba da ima podršku procesima tekstualne analize. Kod oba modula potrebno je omogućiti što veću automatizaciju u procesu projektovanja ETL procesa, odnosno učitavanje podataka iz izvora podataka u skladišta podataka, te iz skladišta podataka u

data martove. U drugom modulu predvidjeti mogućnost ponovnog projektovanja data martova, odnosno projektovanje više različitih data martova.

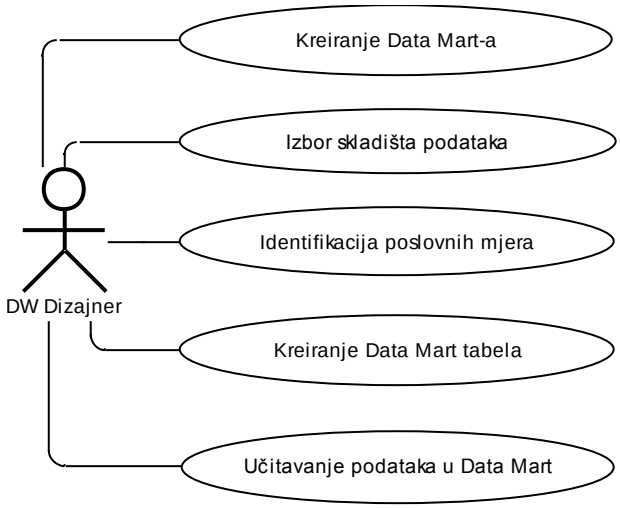
Postoje tri vrste korisnika sistema: dizajner koji projektuje skladište podataka, administrator sistema u kojem se nalaze izvori podataka i korisnik sistema koji će analizirati podatke iz skladišta podataka (najčešće menadžeri koji će na osnovu podataka donositi poslovne odluke). Administrator daje informacije o lokaciji i veličini izvora podataka, a korisnik daje informacije o poslovnim ključevima, poslovnim mjerama i dimenzijama koje su mu potrebne za poslovnu analizu radi donošenja poslovnih odluka.

5.6.1.2. Opis zahtjeva pomoću slučajeva korišćenja

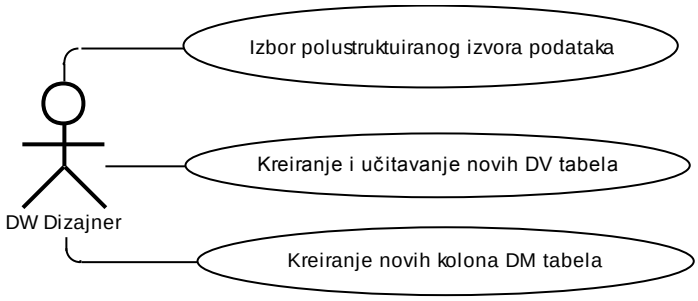
Slučaj korišćenja (SK) se opisuje preko naziva, opisa, aktera, preduslova, kao i osnovnog i alternativnih scenarija. Za pisanje slučajeva korišćenja korišćene su preporuke iz [Cock01]. Slučajevi korišćenja su podijeljeni po modulima. Na osnovu verbalnog opisa uočeni su sljedeći slučajevi korišćenja:



Slika 42. Slučajevi korišćenja 1, projektovanje Data Vault skladišta podataka



Slika 43. Slučajevi korišćenja 2, projektovanje data marta



Slika 44. Slučajevi korišćenja 3, dodavanje polustruktuiranih podataka

SK - 1.1. Kreiranje skladišta podataka i prelazne baze podataka

Naziv: Kreiranje skladišta podataka i prelazne baze podataka

Akter: Dizajner

Učesnici: Dizajner, administrator transakcione baze podataka i sistem

Opis: Dizajner pokreće proces kreiranja prelazne i baze skladišta podataka, kreiranja tabela metapodataka i tabela pravila, te pokreće učitavanje podataka u osnovne šifarnike i tabele pravila. Dizajner na osnovu informacija o veličini izvora podataka određuje veličinu prelazne baze skladišta podataka i na osnovu preporuka administratora određuje fizičke lokacije baza podataka.

Preduslovi: Iz modela na slikama 26. i 27. u poglavlju 4 disertacije kreirani su potrebni skriptovi u skladu sa izabranom platformom za realizaciju skladišta podataka (u ovom slučaju Microsoft SQL Server 2008 R2). Sistem je uključen, otvorena je forma za procese iz faze 1. Administrator je dizajneru dao informaciju o veličini izvora podataka i preporučio fizičku lokaciju baza podataka.

Osnovni scenario izvršenja SK:

1. Dizajner unosi potrebne podatke o karakteristikama baze skladišta podataka (naziv, veličina, rast, fizička lokacija) (APUSO)
2. Dizajner poziva sistem da kreira bazu skladišta podataka (APSO)
3. Sistem kreira bazu skladišta podataka (SO)
4. Sistem prikazuje poruku da je baza skladišta podataka kreirana (IA)
5. Dizajner poziva sistem da u skladištu podataka kreira procedure (čiji skriptovi su generisani iz modela) za kreiranje tabela skladišta podataka (APSO)
6. Sistem u skladištu podataka kreira procedure za kreiranje hub, link i satelit tabela (SO)
7. Sistem prikazuje poruku da su procedure za kreiranje tabela skladišta kreirane (IA)
8. Dizajner unosi potrebne podatke o karakteristikama prelazne baze podataka (naziv, veličina, rast, fizička lokacija) (APUSO)
9. Dizajner poziva sistem da kreira prelaznu bazu podataka (APSO)
10. Sistem kreira prelaznu bazu podataka (SO)
11. Sistem prikazuje poruku da je prelazna baza podataka kreirana (IA)
12. Dizajner poziva sistem da u prelaznoj bazi podataka kreira tabela metapodataka i tabele pravila (čiji skriptovi su generisani iz modela), te da učita osnovne šifarnike i tabele pravila (APSO)

13. Sistem u prelaznoj bazi podataka kreira tabela metapodataka i tabele pravila, te da učita osnovne šifarnike i tabele pravila (SO)
14. Sistem prikazuje poruku da su u prelaznoj bazi podataka kreira tabela metapodataka i tabele pravila, te da učitani osnovni šifarnici i tabele pravila (IA)

Alternativni scenario:

4.1. Ukoliko se ne može kreirati baza skladišta podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

7.1. Ukoliko ne mogu da se kreiraju procedure za kreiranje tabela skladišta podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

11.1. Ukoliko se ne može kreirati prelazna baza podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

14.1. Ukoliko ne mogu da se kreiraju tabela metapodataka i tabele pravila, te da učitaju osnovni šifarnici i tabele pravila, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 1.2. Izbor izvora podataka za skladište podataka

Naziv: Izbor izvora podataka za skladište podataka

Akter: Dizajner

Učesnici: Dizajner, administrator i sistem

Opis: Dizajner na osnovu informacija od administratora bira struktuirane izvore podataka

Preduslovi: Kreirane su prelazna i baza skladišta podataka. Kreirane su tabele metapodataka u prelaznoj bazi podataka. Sistem je uključen. Administrator je dizajneru dao pristupne podatke prema serveru gdje se nalaze izvori podataka. Prikazana je forma za konekciju prema ciljanom serveru.

Osnovni scenario izvršenja SK:

1. Dizajner upisuje pristupne podatke prema serveru gdje se nalaze izvori podataka (APUSO)
2. Dizajner poziva sistem da prikaže listu struktuiranih izvora podataka (APSO)
3. Sistem učitava dostupne struktuirane izvore podataka na izabranom serveru (SO)
4. Sistem prikazuje listu dostupnih struktuiranih izvora podataka (IA)
5. Dizajner bira struktuirane izvore podataka (APUSO)
6. Dizajner poziva sistem da snimi informaciju o izabranim struktuiranim izvorima podataka (APSO)
7. Sistem snima izabrane struktuirane izvore podataka (SO)
8. Sistem vraća poruku da su snimljeni struktuirani izvori podataka (IA)
9. Dizajner poziva sistem da izvrši učitavanje metapodataka iz struktuiranih izvora podataka (APSO)
10. Sistem vrši učitavanje metapodataka struktuiranih izvora podataka (SO)
11. Sistem vraća poruku da su učitani metapodaci struktuiranih izvora podataka (IA)

Alternativni scenario:

4.1 Ukoliko se ne mogu prikazati informacije o dostupnim struktuiranim izvorima podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

8.1. Ukoliko se ne mogu snimiti informacije o struktuiranim izvorima podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

11.1. Ukoliko se ne mogu učitati metapodaci struktuiranih izvora podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 1.3. Identifikacija poslovnih ključeva

Naziv: Identifikacija poslovnih ključeva

Akter: Dizajner

Učesnici: Dizajner, korisnik i sistem

Opis: Dizajner na osnovu informacija od korisnika selektuje kroz aplikaciju poslovne ključeve

Preduslovi: Korisnik je saopštio dizajneru poslovne ključeve. Izabrani su izvori podataka. Prikazana je forma za izbor poslovnih ključeva

Osnovni scenario izvršenja SK:

1. Dizajner bira poslovne ključeve kroz korisnički interfejs (APUSO)
2. Dizajner prosleđuje sistemu poslovne ključeve (APSO)
3. Sistem snima informaciju o poslovnim ključevima (SO)
4. Sistem vraća informaciju da su snimljeni poslovni ključevi (IA)
5. Dizajner poziva sistem da identifikuje hub tabele (APSO)
6. Sistem na osnovu poslovnih ključeva i pravila identifikuje hub tabela (SO)
7. Sistem vraća informaciju da su na osnovu poslovnih ključeva identifikovane hub tabele (IA)
8. Dizajner poziva sistem da identifikuje link tabele (APSO)
9. Sistem na osnovu identifikovanih hub tabela i pravila identifikuje link tabele (SO)
10. Sistem vraća informaciju da su identifikovane link tabele (IA)
11. Dizajner poziva sistem da identifikuje satelit tabele (APSO)
12. Sistem na osnovu identifikovanih hub i link tabela i na osnovu pravila identifikuje satelit tabele (SO)
13. Sistem vraća informaciju da su identifikovane satelit tabele (IA)

Alternativni scenario:

4.1. Ukoliko nije moguće snimiti informaciju o poslovnim ključevima sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

7.1. Ukoliko nije moguće identifikovati hub tabela, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

10.1. Ukoliko nije moguće identifikovati link tabela, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

13.1. Ukoliko nije moguće identifikovati satelit tabela, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 1.4. Kreiranje Data Vault tabela

Naziv: Kreiranje Data Vault tabela

Akter: Dizajner

Učesnici: Dizajner i sistem

Opis: Dizajner pokreće proces kreiranja Data Vault tabela (hub, link i satelit)

Preduslovi: Identifikovani su poslovni ključevi, hub, link i satelit tabele.

Prikazana je forma za kreiranje Data Vault tabela

Osnovni scenario izvršenja SK:

1. Dizajner, po potrebi, radi ručnu modifikaciju nad satelit kandidatima (APUSO)
2. Dizajner, po potrebi, poziva sistem da snimi satelit kandidate (APSO)
3. Sistem snima informaciju o modifikovanim satelit kandidatima (SO)
4. Sistem vraća informaciju da su snimljene modifikacije o satelit kandidatima (IA)
5. Dizajner poziva sistem da kreira hub tabele (APSO)
6. Sistem na osnovu identifikovanih hub tabela i pravila kreira hub tabele (SO)
7. Sistem vraća informaciju da su kreirane hub tabele (IA)
8. Dizajner poziva sistem da kreira link tabele (APSO)

9. Sistem na osnovu identifikovanih link tabela kreira link tabele (SO)
10. Sistem vraća informaciju da su kreirane link tabele (IA)
11. Dizajner poziva sistem da kreira satelit tabele (APSO)
12. Sistem na osnovu identifikovanih satelit tabela i na osnovu pravila kreira satelit tabele (SO)
13. Sistem vraća informaciju da su kreirane satelit tabele (IA)

Alternativni scenario:

4.1. Ukoliko nije moguće snimiti informaciju o modifikovanim hub, link i satelit kandidatima, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

7.1. Ukoliko nije moguće kreiranje hub tabela sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

10.1. Ukoliko nije moguće kreiranje link tabela sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

13.1. Ukoliko nije moguće kreiranje satelit tabela sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 1.5. Učitavanje podataka iz izvora podataka u Data Vault

Naziv: Učitavanje podataka u Data Vault skladište podataka

Akter: Dizajner

Učesnici: Dizajner i sistem

Opis: Dizajner otvara formu za učitavanje podataka u skladište podataka, sistem ekstrahuje podatke iz izvora podataka i učitava u skladište podataka

Preduslovi: Kreirane su tabele skladišta podataka. Prikazana je forma za ekstrakciju i učitavanje podataka u skladište podataka

Osnovni scenario izvršenja SK:

1. Dizajner poziva sistem da prikaže Data Vault metapodatke (APSO)
2. Sistem vrši učitavanje Data Vault metapodataka (SO)

3. Sistem prikazuje Data Vault metapodatke (IA)
4. Dizajner pokreće proces ekstrakcije i učitavanja podataka u hub tabele (APSO)
5. Sistem vrši ekstrakciju i učitavanje podataka u hub tabele na osnovu pravila (SO)
6. Sistem vraća informaciju da su učitani podaci u hub tabele (IA)
7. Dizajner pokreće proces ekstrakcije i učitavanja podataka u link tabele (APSO)
8. Sistem vrši ekstrakciju i učitavanje podataka u link tabele na osnovu pravila (SO)
9. Sistem vraća informaciju da su učitani podaci u link tabele (IA)
10. Dizajner pokreće proces ekstrakcije i učitavanja podataka u satelit tabele (APSO)
11. Sistem vrši ekstrakciju i učitavanje podataka u satelit tabele na osnovu pravila (SO)
12. Sistem vraća informaciju da su učitani podaci u satelit tabele (IA)

Alternativni scenario:

3.1. Ukoliko nije moguće prikazati Data Vault metapodatke, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

6.1. Ukoliko nije moguće učitavanje podataka u hub tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

9.1. Ukoliko nije moguće učitavanje podataka u link tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

12.1. Ukoliko nije moguće učitavanje podataka u satelit tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 2.1. Kreiranje data mart-a

Naziv: Kreiranje data mart-a

Akter: Dizajner

Učesnici: Dizajner, administrator transakcione baze podataka i sistem

Opis: Dizajner pokreće proces kreiranja data mart baze (ili šeme) podataka kreiranja tabela metapodataka Data Vaulta, te pokreće učitavanje podataka u tabele metapodataka Data Vaulta i tabele pravila.

Preduslovi: Iz modela na slici 33. u poglavlju 4 disertacije kreirani su potrebni skriptovi u skladu sa izabranom platformom za realizaciju skladišta podataka (u ovom slučaju Microsoft SQL Server 2008 R2). Sistem je uključen, otvorena je forma za procese iz faze 3.

Osnovni scenario izvršenja SK:

1. Dizajner poziva sistem da kreira data mart bazu (ili šemu) podataka (APSO)
2. Sistem kreira data mart bazu podataka (SO)
3. Sistem prikazuje poruku da baza kreirana (IA)
4. Dizajner poziva sistem da kreira procedure (koje su generisane iz modela) za kreiranje tabela data marta, tabele metapodataka Data Vaulta i učita tabele pravila (APSO)
5. Sistem u skladištu podataka: kreira procedure za kreiranje tabela data marta, tabele metapodataka Data Vaulta i učitava tabele pravila (SO)
6. Sistem prikazuje poruku da su procedure i tabele kreirane i učitane tabele pravila (IA)

Alternativni scenario:

3.1. Ukoliko ne može da se kreira data mart baza podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

6.1. Ukoliko sistem ne može da kreira procedure za kreiranje tabela data marta, tabele metapodataka Data Vaulta i učita tabele pravila, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 2.2. Izbor skladišta podataka

Naziv: Izbor skladišta podataka

Akter: Dizajner

Učesnici: Dizajner, administrator i sistem

Opis: Dizajner na osnovu informacija od administratora bira skladište podataka čije podatke će koristiti za data martove

Preduslovi: Kreirana je šema ili baza data marta. Kreirane su tabele metapodataka skladišta podataka. Sistem je uključen. Administrator je dizajneru dao pristupne podatke prema serveru gdje se nalazi skladište podataka. Prikazana je forma za konekciju prema ciljanom serveru.

Osnovni scenario izvršenja SK:

1. Dizajner upisuje pristupne podatke prema serveru gdje se nalaze skladište podataka (APUSO)
2. Dizajner poziva sistem da prikaže listu skladišta podataka (APSO)
3. Sistem učitava dostupna skladišta podataka na izabranom serveru (SO)
4. Sistem prikazuje listu dostupnih skladišta podataka (IA)
5. Dizajner bira skladište podataka (APUSO)
6. Dizajner poziva sistem da snimi informaciju o izabranom skladištu podataka (APSO)
7. Sistem snima izabrano skladište podataka (SO)
8. Sistem vraća poruku da su snimljene informacije o izabranom skladištu podataka (IA)
9. Dizajner poziva sistem da izvrši učitavanje metapodataka iz izabranog skladišta podataka (APSO)

10. Sistem vrši učitavanje metapodataka izabranog skladišta podataka (SO)
11. Sistem vraća poruku da su učitani meta izabranog skladišta podataka (IA)

Alternativni scenario:

4.1. Ukoliko se ne mogu prikazati informacije o dostupnim skladištima podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

8.1. Ukoliko se ne mogu snimiti informacije o izabranom skladištu podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

11.1. Ukoliko se ne mogu učitati metapodaci skladišta podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 2.3. Identifikacija poslovnih mjera

Naziv: Identifikacija poslovnih mjera

Akter: Dizajner

Učesnici: Dizajner, korisnik i sistem

Opis: Dizajner na osnovu informacija od korisnika selektuje kroz aplikaciju poslovne mjere, sistem na osnovu poslovne mjere predlaže dimenzije. Korisnik iz ponuđenog skupa dimenzija selektuje dimenzije kroz koje želi da posmatra poslovnu mjeru.

Preduslovi: Korisnik je saopštio dizajneru informaciju o poslovnim mjerama i dimenzijama. Prikazana je forma za identifikaciju poslovnih mjera i dimenzija

Osnovni scenario izvršenja SK:

1. Dizajner selektuje poslovne mjere kroz korisnički interfejs (APUSO)
2. Dizajner sistemu prosleđuje listu poslovnih mjera (APSO)
3. Sistem identifikuje tabele mjera (SO)
4. Sistem vraća informaciju da su identifikovane tabele mjere (IA)
5. Dizajner poziva sistem da identifikuje inicijalne dimenzije (APSO)

6. Sistem identifikuje inicijalne dimenzije (SO)
7. Sistem vraća inicijalnu listu identifikovanih dimenzija (IA)
8. Dizajner selektuje dimenzije koje želi da koristi iz ponuđene liste dimenzija (APUSO)
9. Dizajner sistemu prosleđuje listu dimenzija koje želi da koristi (APSO)
10. Sistem identifikuje konačnu listu dimenzija (SO)
11. Sistem vraća informaciju da su identifikovane dimenzije (IA)

Alternativni scenario:

4.1. Ukoliko nije moguća identifikacija poslovnih mjera sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

7.1. Ukoliko nije moguća identifikacija inicijalnih dimenzija, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

11.1. Ukoliko nije moguća identifikacija konačnih dimenzija, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 2.4. Kreiranje Data Mart tabela

Naziv: Kreiranje Data Mart tabela

Akter: Dizajner

Učesnici: Dizajner, korisnik i sistem

Opis: Sistem kreira tabelu vremenskih dimenzija. Dizajner određuje način agregacije mjera i upisuje ime tabele mjera. Dizajner na osnovu identifikovanih poslovnih mjera i dimenzija kreira tabele mjera i dimenzija.

Preduslovi: Identifikovane su poslovne mjere i dimenzije. Prikazana je forma za kreiranje vremenskih dimenzija, načina agregacije mjera i kreiranje poslovnih mjera i dimenzija

Osnovni scenario izvršenja SK:

1. Iz ponuđenih datumskih i vremenskih polja, dizajner označava koja će se tabela i kolona koristiti kao vremenska dimenzije (APUSO)
2. Dizajner poziva sistem da snimi vremenske dimenzije (APSO)
3. Sistem snima vremenske dimenzije (SO)
4. Sistem vraća informaciju da su snimljene vremenske dimenzije (IA)
5. Dizajner upisuje vremenski opseg kroz koji će se posmatrati vremenska dimenzija (APUSO)
6. Dizajner pokreće proces kreiranja i popunjavanja tabele vremenskih dimenzija (APSO)
7. Sistem kreira i popunjava tabelu vremenskih dimenzija (SO)
8. Sistem vraća informaciju da je kreirana i popunjena tabela vremenskih dimenzija (IA)
9. Dizajner poziva sistem da prikaže izabrane mjere (APSO)
10. Sistem vrši učitavanje izabranih mjera (SO)
11. Sistem prikazuje izabrane mjere (IA)
12. Dizajner popunjava tabelu načina agregacije mjera i upisuje ime tabele mjera (APUSO)
13. Dizajner poziva sistem da izvrši snimanje podataka u tabelu agregacija (APSO)
14. Sistem vrši upisivanje podataka o agregacijama mjera (SO)
15. Sistem vraća informaciju da je popunjena pomoćna tabela agregacija (IA)
16. Dizajner pokreće proces kreiranja tabela dimenzija i mjera (APSO)
17. Sistem kreira tabele dimenzija i mjera (SO)
18. Sistem vraća informaciju da su kreirane tabele dimenzija i mjera (IA)

Alternativni scenario:

4.1. Ukoliko nije moguće snimiti vremenske dimenzije, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

8.1. Ukoliko nije moguće kreiranje ili popunjavanje tabele vremenskih dimenzija, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

11.1. Ukoliko nije moguće prikazati izabrane mjere, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

15.1. Ukoliko nije moguće popunjavanje tabela agregacija, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

18.1. Ukoliko nije moguće kreiranje tabela dimenzija i mjera, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 2.5. Učitavanje podataka iz Data Vaulta u Data Mart

Naziv: Učitavanje podataka iz Data Vaulta u Data Mart

Akter: Dizajner

Učesnici: Dizajner i sistem

Opis: Dizajner otvara formu za učitavanje podataka u Data Mart, sistem ekstrahuje podatke iz skladišta podataka i učitava u Data Mart

Preduslovi: Kreirane su tabele Data Marta. Prikazana je forma za ekstrakciju i učitavanje podataka u Data Mart

Osnovni scenario izvršenja SK:

1. Dizajner poziva sistem da prikaže Data Mart metapodatke (APSO)
2. Sistem vrši učitavanje Data Mart metapodataka (SO)
3. Sistem prikazuje Data Mart metapodatke (IA)
4. Dizajner pokreće proces ekstrakcije i učitavanja podataka u dimenzione tabele (APSO)
5. Sistem vrši ekstrakciju i učitavanje podataka u dimenzione tabele na osnovu pravila (SO)
6. Sistem vraća informaciju da su učitani podaci u dimenzione tabele (IA)

7. Dizajner pokreće proces ekstrakcije i učitavanja podataka u tabele mjera (APSO)
8. Sistem vrši ekstrakciju i učitavanje podataka u tabele mjera na osnovu pravila (SO)
9. Sistem vraća informaciju da su učitani podaci u tabele mjera (IA)

Alternativni scenario:

3.1. Ukoliko nije moguće prikazati Data Mart metapodatke, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

6.1. Ukoliko nije moguće učitavanje podataka u dimenzione tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

9.1. Ukoliko nije moguće učitavanje podataka u tabele mjera, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 3.1. Izbor polustruktuiranog izvora podataka za skladište podataka

Naziv: Izbor polustruktuiranog izvora podataka za skladište podataka

Akter: Dizajner

Učesnici: Dizajner, administrator i sistem

Opis: Dizajner na osnovu informacija od administratora bira polustruktuirane izvore podataka

Preduslovi: Kreirane su prelazna i baza skladišta podataka. Sistem je uključen. Administrator je dizajneru dao pristupne podatke prema serveru gdje se nalaze polustruktuirani izvori podataka. Prikazana je forma za konekciju prema ciljanom serveru.

Osnovni scenario izvršenja SK:

1. Dizajner upisuje pristupne podatke prema serveru gdje se nalaze skladište podataka (APUSO)

2. Dizajner poziva sistem da prikaže listu skladišta i polustruktuiranih izvora podataka gdje će se učitati polustruktuirani podaci (APSO)
3. Sistem učitava dostupna skladišta i polustruktuirane izvore podataka na izabranom serveru (SO)
4. Sistem prikazuje listu dostupnih skladišta i polustruktuiranih izvora podataka (IA)
5. Dizajner bira skladište podataka, data mart i polustruktuirani izvor podataka (APUSO)
6. Dizajner poziva sistem da na osnovu podataka snimi informaciju o polustruktuiranom izvoru podataka (APSO)
7. Sistem snima informaciju o polustruktuiranom izvoru podataka (SO)
8. Sistem vraća poruku da su snimljeni polustruktuirani izvori podataka (IA)
9. Dizajner poziva sistem da izvrši učitavanje metapodataka iz polustruktuiranih izvora podataka (APSO)
10. Sistem vrši učitavanje metapodataka polustruktuiranih izvora podataka (SO)
11. Sistem vraća poruku da su učitani metapodaci polustruktuiranih izvora podataka (IA)

Alternativni scenario:

4.1. Ukoliko se ne mogu prikazati informacije o dostupnim skladištima podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

8.1. Ukoliko se ne mogu snimiti informacije o polustruktuiranim izvorima podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

11.1. Ukoliko se ne mogu učitati metapodaci polustruktuiranih izvora podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 3.2. Kreiranje i učitavanje novih Data Vault tabela

Naziv: Kreiranje i učitavanje novih Data Vault tabela

Akter: Dizajner

Učesnici: Dizajner i sistem

Opis: Dizajner pokreće proces kreiranja novih Data Vault tabela (uglavnom Satelita) na osnovu polustruktuiranih izvora podataka

Preduslovi: Identifikovani su skladište podataka, data mart i polustruktuirani izvor podataka. Prikazana je forma za kreiranje nove Data Vault tabele

Osnovni scenario izvršenja SK:

1. Dizajner vrši mapiranje metapodataka polustruktuiranog izvora podataka i postojećih Data Vault tabela (APUSO)
2. Dizajner poziva sistem da kreira novu Satelit tabelu (APSO)
3. Sistem kreira novu Satelit tabelu (SO)
4. Sistem vraća informaciju da je kreirana nova Satelit tabela (IA)
5. Dizajner poziva sistem da učita novu Satelit tabelu (APSO)
6. Sistem učitava podatke iz polustruktuiranog izvora u novu Satelit tabelu (SO)
7. Sistem vraća informaciju da su učitani polustruktuirani podaci u novu Satelit tabelu (IA)

Alternativni scenario:

4.1. Ukoliko nije moguće kreirati nove Data Vault tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

7.1. Ukoliko nije moguće učitati polustruktuirane podatke u nove Data Vault tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

SK - 3.3. Kreiranje i učitavanje novih kolona Data Mart tabela

Naziv: Kreiranje i učitavanje novih kolona Data Mart tabela

Akter: Dizajner

Učesnici: Dizajner i sistem

Opis: Dizajner pokreće proces kreiranja novih kolona Data Mart tabela (uglavnom tabela dimenzija) na osnovu polustruktuiranih izvora podataka

Preduslovi: Identifikovani su skladište podataka, data mart i polustruktuirani izvor podataka. Prikazana je forma za kreiranje nove Data Mart tabela

Osnovni scenario izvršenja SK:

1. Dizajner vrši mapiranje metapodataka nove Data Vault tabela kreirane iz polustruktuiranog izvora podataka i postojećih Data Mart tabela (APUSO)
2. Dizajner poziva sistem da kreira novu kolonu izabrane dimenzione tabele (APSO)
3. Sistem kreira novu kolonu izabrane dimenzione tabele (SO)
4. Sistem vraća informaciju da je kreirana nove kolona dimenzione tabele (IA)
5. Dizajner poziva sistem da učita novu kolonu dimenzione tabelu (APSO)
6. Sistem učitava podatke iz nove Satelit tabele (kreirane iz polustruktuiranog izvora podataka) u novu kolonu dimenzionu tabele (SO)
7. Sistem vraća informaciju da su učitani podaci u novu kolomu dimenzione tabele (IA)

Alternativni scenario:

4.1. Ukoliko nije moguće kreirati novu kolonu dimenzione tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

7.1. Ukoliko nije moguće učitati podatke u novu kolonu dimenzione tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

Za detaljan opis slučajeva korišćenja ponekad se prezentira korisnički interfejs preko koga će akter da koristi budući sistem. Prikaz korisničkog interfejsa, obično ekranske forme značajno olakšava definisanje i razumijevanje slučaja korišćenja [LMAB08]. Kako su sve ekranske forme prikazane u dijelu 5.6.3.3. Projektovanje, u ovom slučaju ćemo se sadržati na modelima slučajeva korišćenja.

5.6.2. Analiza

Kod Larmanove metode razvoja softvera, faza analize opisuje logičku strukturu i ponašanje softverskog sistema [Vla04]. Detaljan i formalniji opis slučajeva korišćenja može se dati i preko tzv. sistemskog dijagrama sekvenci [LMAB08].

Ponašanje softverskog sistema se opisuje pomoću [Vla04]:

- sistemskih dijagrama sekvenci, koji se prave za svaki slučaj korišćenja,
- ugovora o sistemskim operacijama, koji se dobijaju na osnovu sistemskih dijagrama sekvenci.

Struktura softverskog sistema se opisuje pomoću:

- konceptualnog modela,
- relacionog modela.

Ponašanje softverskog sistema preko sistemskih dijagrama sekvenci (DS) radi se na taj način što se za svaki slučaj korišćenja i za svaki scenario prave dijagrami i to samo za slučajeve kada akter poziva sistem da izvrši sistemsku operaciju (APSO) i kad sistem daje izlazne argumente (IA) iz sistema. Dijagrami sekvenci su dati u Dodatku B.

Prema [Lar98][Vla04], „dijagrami sekvenci se sastoje od sistemskih operacija za koje se prave ugovori. Kod definisanja a koristi se i konceptualni model.

Ugovor opisuje stanje i ponašanje sistema nakon poziva sistemske operacije. Jedan ugovor vezan je za jednu sistemsku operaciju. Ugovor uobičajeno sadrži sljedeće stavke: ime, operaciju, odgovornost, vezu sa slučajem korišćenja, preduslov i postuslov. Sistemska operacija sadrži metode i opciono ulazne ili izlazne argumente."

5.6.2.1. Rezultati analize dijagrama sekvenci

Rezultatom analize sistemskog dijagrama sekvenci (Dodatak B) dobijeno je 38 sistemskih operacija koje treba projektovati:

1. KreirajBazuPodataka (Database, ImeFajla, Velicina, MaxVelicina)
2. IzvrsiSQLScript (Database, List <SqlScript>)
3. PrikaziListuIzvoraPodataka (Server)
4. SnimiIzvorPodataka (IzvorPodataka)
5. UcitajMetaPodatke (IzvorPodataka)
6. SnimiPoslovneKljuceve (Kolona, Vrijednost)
7. IdentifikujHubTabele (Pravilo)
8. IdentifikujLinkTabele (Pravilo)
9. IdentifikujSatelitTabele (Pravilo)
10. ModifikujSatelitKandidate (Kolona, Vrijednost)
11. KreirajHubTabele (Pravilo, SkladistePodataka)
12. KreirajLinkTabele (Pravilo, SkladistePodataka)
13. KreirajSatelitKandidate (Pravilo, SkladistePodataka)
14. PrikaziDataVaultMetaPodatke(SkladistePodataka, PrelaznaBaza)
15. UcitajHubTabele (Pravilo)
16. UcitajLinkTabele (Pravilo)
17. UcitajSatelitTabele (Pravilo)
18. KreirajDMProcedureTabele (DatabaseName, List <SqlScript>)

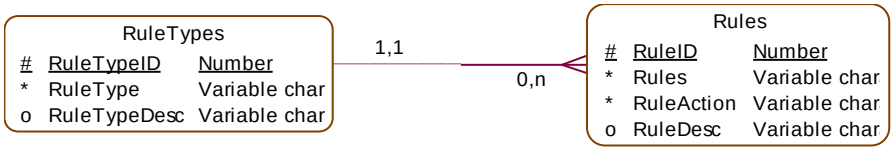
19. SnimiDataMartIzvorPodataka (SkladistePodataka)
20. UcitajDVMetaPodatke (SkladistePodataka)
21. SnimiPoslovneMjere (Kolona, Vrijednost)
22. IdentifikujInicijalneDimenzije (Pravilo)
23. SnimiKonacneDimenzije (Kolona, Vrijednost)
24. OznaciVremenskeDimenzije (Kolona, Vrijednost)
25. KreirajTabeluVremenskihDimenzija (StartDate, EndDate, DataMart)
26. PrikaziIzabraneMjere (DataMart)
27. SnimiNacinAgregacijeMjera (Kolona, FactName, Formula)
28. KreirajTabeleDimenzijaMjera(Pravilo, ImeTabeleMjera)
29. PrikaziDataMartMetaPodatke (DataMart)
30. UcitajDimTabele (Pravilo)
31. UcitajFactTabele (Pravilo, FactTabela)
32. PrikaziListuIzvoraPodataka (Server)
33. SnimiPolustruktuiraniFajl (Pravilo)
34. UcitajSSMetaPodatke (TipFajla, ImeFajla, PrelaznaBaza)
35. KreirajNovuSatelitTabelu (Pravilo, SkladistePodataka)
36. UcitajSatelitTabeluSS (Pravilo, Fajl)
37. KreirajNovuDimTabelu (Pravilo, DataMart, SatelitTabela)
38. UcitajDimTabeluSS (Pravilo, DataMart, DimTabela)

Sistemske operacije nakon izvršavanja, vratiće odgovarajući signal.

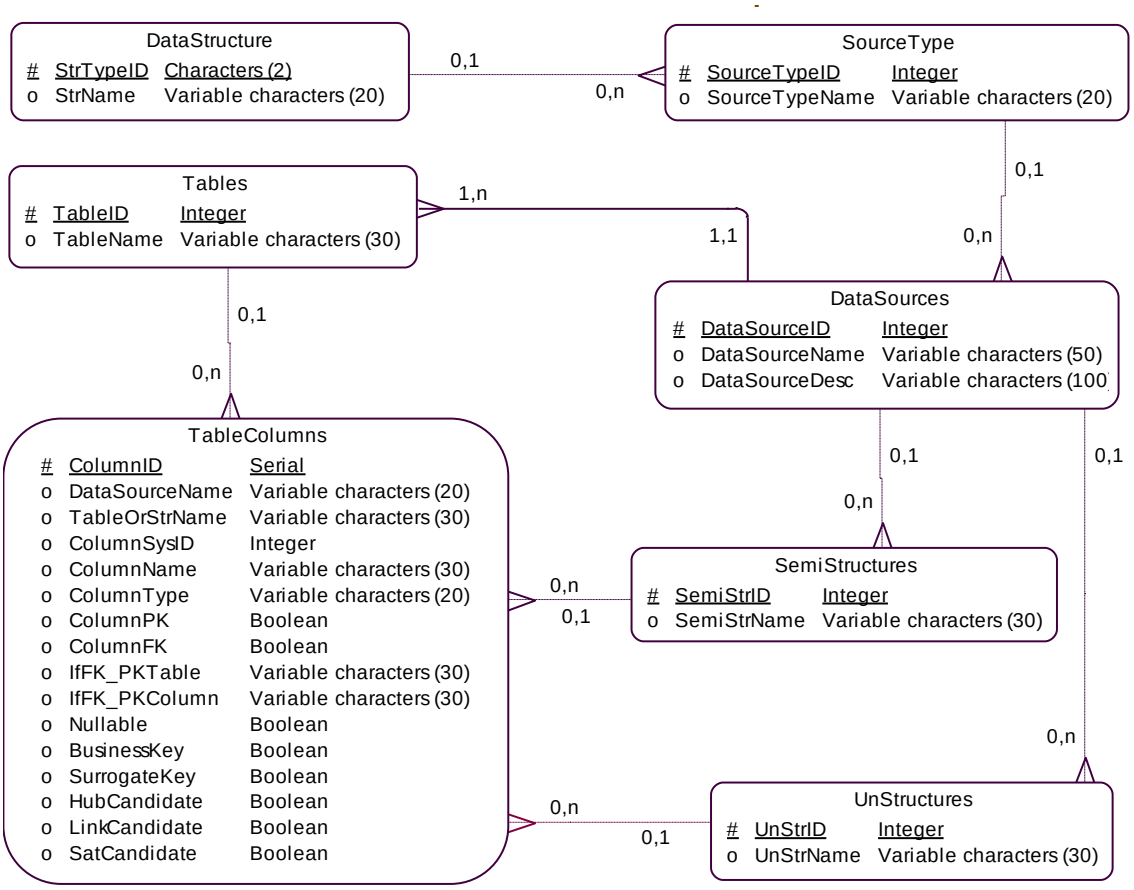
5.6.2.2. Struktura softverskog sistema

Na osnovu slučajeva korišćenja i dijagrama sekvenci radi se struktura softverskog sistema. Struktura softverskog sistema je u ovoj fazi predstavljena

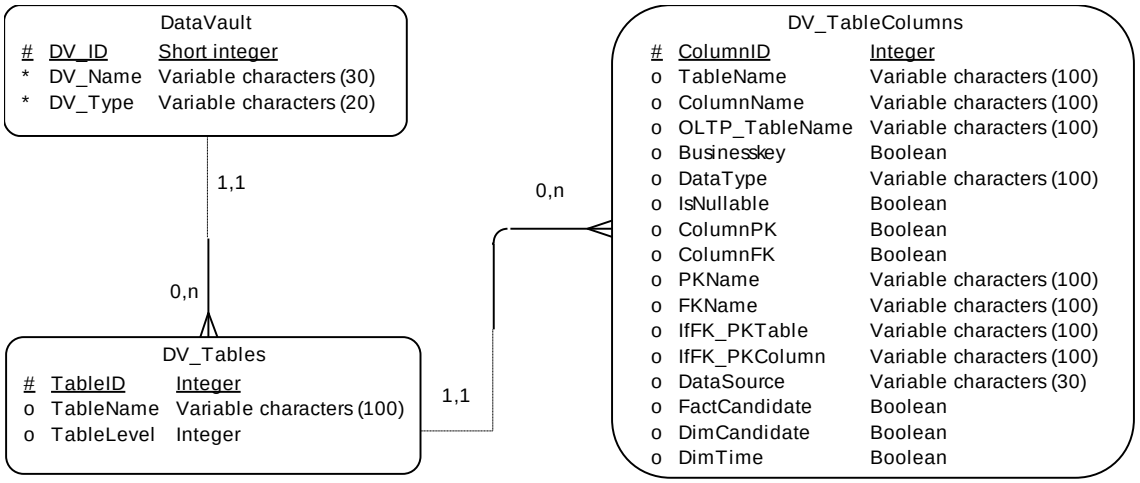
konceptualnim modelom. Na sljedećim slikama je predstavljeni su konceptualni modeli razmatranog sistema.



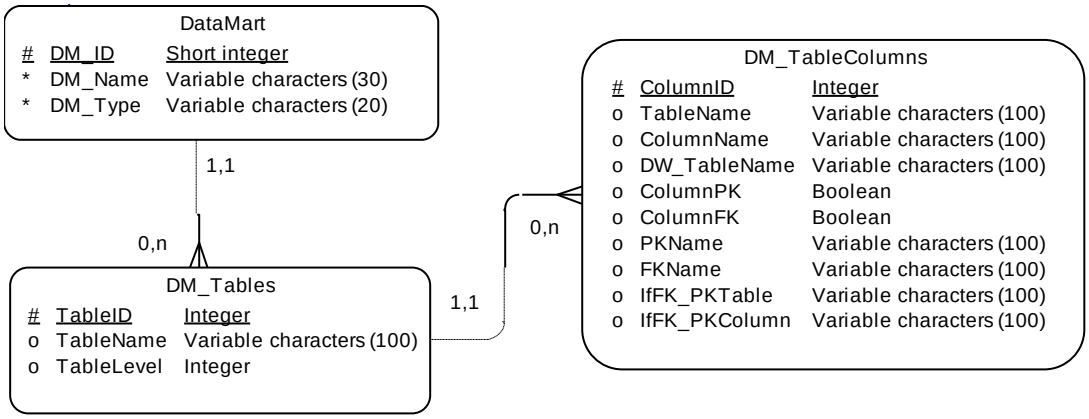
Slika 45. Konceptualni model pravila



Slika 46. Konceptualni model metapodataka izvora podataka



Slika 47. Konceptualni model metapodataka Data Vault skladišta podataka



Slika 48. Konceptualni model metapodataka data marta

Na osnovu konceptualnog modela napravljen je relacioni model iz koga se generiše baza podataka:

RuleTypes (RuleTypeID, RuleType, RuleTypeDesc)

Rules (RuleTypeID, RuleID, RuleFor, RuleAction, RuleDesc)

DataStructure (StrTypeID, StrName)

SourceType (SourceTypeID, StrTypeID, SourceTypeName)

DataSources (DataSourceID, SourceTypeID, DataSourceName, DataSourceDesc)

Tables (TableID, DataSourceID, TableName)

SemiStructures (SemiStrID, DataSourceID, SemiStrName)

UnStructures (UnStrID, DataSourceID, UnStrName)

TableColumns (TableID, SemiStrID, UnStrID, DataSourceName, TableOrStrName, ColumnSysID, ColumnName, ColumnType, ColumnPK, ColumnFK, IfFK_PKTable, IfFK_PKColumn, Nullable, BusinessKey, SurrogateKey, HubCandidate, LinkCandidate, SatCandidate)

DataVault (DV_ID, DV_Name, DV_Type)

DV_Tables (TableID, TableName, TableLevel)

DV_TableColumns (ColumnID, TableName, ColumnName, OLTP_TableName, BusinessKey, DataType, IsIdentity, ColumnPK, ColumnFK, PKName, FKName, IfFK_PKTable, IfFK_PKColumn, DataSource, FactCandidate, DimCandidate, DimTime)

DataMart (DM_ID, DM_Name, DM_Type)

DM_Tables (TableID, TableName, TableLevel)

DM_TableColumns (ColumnID, TableName, ColumnName, DW_TableName, ColumnPK, ColumnFK, PKName, FKName, IfFK_PKTable, IfFK_PKColumn)

Atributi koji predstavljaju primarne ključeve su podvučeni.

5.6.3. Projektovanje

Prema [Lar98][Vla04], u fazi projektovanja opisuje se arhitektura softverskog sistema (fizička struktura i ponašanje softverskog sistema). Projektovanje arhitekture obuhvata projektovanje aplikacione logike, skladišta podataka i korisničkog interfejsa. Korisnički interfejs se sastoji od ekranskih formi i

kontrolera korisničkog interfejsa. Ekranska forma prihvata ulazne podatke i događaje od Aktera, te ih prosleđuje kontroleru. Kontroler prihvata unijete podatke od ekranske forme, konvertuje ih u objekat odnosno ulazni argument za sistemske operacije i šalje zahtjev za izvršenje sistemske operacije. Osim toga, kontroler prihvata izlaz softverskog sistema koji nastaje kao rezultat izvršenja sistemske operacije i konvertuje objekat za prikaz na grafičkom korisničkom interfejsu [Lar98] [Vla04].

U radu je korišćena klasična troslojna arhitektura aplikacije, kao što je prikazano na sljedećoj slici.

Slika 49. Troslojna arhitektura aplikacije

5.6.3.1. Projektovanje relacionog skladišta podataka

U ovoj fazi projektovane su tabele relacionog sistema za upravljanje bazom podataka. Tabele baze podataka koje se odnose na metapodatke izvora podataka za dizajn Data Vaulta sa odgovarajućim kolonama i tipovima podataka date, tabele baze podataka koje se odnose na metapodatke Data Vaulta za dizajn Data Marta sa odgovarajućim kolonama i tipovima podataka date su u sljedeće dvije tabele.

Tabela 15. Struktura tabela metapodataka izvora podataka

Tabela	Naziv atributa	Prosto vrijedonosno ograničenje		Složeno vrijedonosno ograničenje (Međuzavisnost atributa više tabela)
		Tip podatka	Vrijednost	
DataSources	DataSourceID	int	Not null	PK_DATASOURCES primary key (DataSourceID)
	SourceTypeID	int		FK_DATASOUR_REFERENCE_SOURCETY foreign key (SourceTypeID) references SourceType (SourceTypeID)
	DataSourceName	varchar(50)		
	DataSourceDesc	varchar(100)		
Rules	RuleTypeID	numeric(18,0)	Not null	PK_RULES primary key (RuleTypeID, RuleID)
	RuleID	numeric(18,0)	Not null	FK_RULES_REFERENCE_RULETYPE foreign key (RuleTypeID) references RuleTypes (RuleTypeID)
	Rules	varchar(30)	Not null	
	RuleAction	varchar(1000)	Not null	
	RuleDesc	varchar(100)		
RuleTypes	RuleTypeID	numeric(18,0)	Not null	PK_RULETYPES primary key (RuleTypeID)
	RuleType	varchar(30)	Not null	
	RuleTypeDesc	varchar(100)		
SemiStructures	SemiStrID	int	Not null	PK_SEMISTRUCTURES primary key (SemiStrID)
	DataSourceID	int		FK_SEMISTRU_REFERENCE_DATASOUR foreign key (DataSourceID)

				references DataSources (DataSourceID)
	SemiStrName	varchar(30)		
SourceType	SourceTypeID	int	Not null	PK_SOURCETYPE primary key (SourceTypeID)
	StrTypeID	char(2)		FK_SOURCETY_REFERENCE_STRUCTUR foreign key (StrTypeID) references DataStructure (StrTypeID)
	SourceTypeName	varchar(20)		
DataStructure	StrTypeID	char(2)	Not null	PK_STRUCTURETYPE primary key (StrTypeID)
	StrName	varchar(20)		
TableColumns	ColumnID	int	Not null	PK_TABLECOLUMNS primary key (ColumnID)
	TableID	int		FK_TABLECOL_REFERENCE_TABLES foreign key (TableID) references Tables (TableID)
	SemiStrID	int		FK_TABLECOL_REFERENCE_SEMISTRU foreign key (SemiStrID) references SemiStructures (SemiStrID)
	UnStrID	int		FK_TABLECOL_REFERENCE_UNSTRUCT foreign key (UnStrID) references UnStructures (UnStrID)
	DataSourceName	varchar(20)		
	TableOrStrName	varchar(30)		
	ColumnSysID	int		
	ColumnName	varchar(30)		
	ColumnType	varchar(20)		
	ColumnPK	bit		
	ColumnFK	bit		
	IfFK_PKTable	varchar(30)		

	IfFK_PKColumn	varchar(30)		
	Nullable	bit		
	BusinessKey	bit		
	SurrogateKey	bit		
	HubCandidate	bit		
	LinkCandidate	bit		
	SatCandidate	bit		
Tables	TableID	int	Not null	PK_TABLES primary key (TableID)
	DataSourceID	int		FK_TABLES_REFERENCE_DATASOUR foreign key (DataSourceID) references DataSources (DataSourceID)
	TableName	varchar(30)		
	TableLevel	int		
UnStructures	UnStrID	int	Not null	PK_UNSTRUCTURES primary key (UnStrID)
	DataSourceID	int		FK_UNSTRUCT_REFERENCE_DATASOUR foreign key (DataSourceID) references DataSources (DataSourceID)
	UnStrName	varchar(30)		

Tabela 16. Struktura tabela metapodataka skladišta podataka

Tabela	**Naziv atributa**	**Prosto vrijedonosno ograničenje**		**Složeno vrijedonosno ograničenje (Međuzavisnost atributa više tabela)**
		Tip podatka	**Vrijednost**	
DataVault	DV_ID	smallint	Not null	PK_DATAMART primary key (DV_ID)
	DV_Name	varchar(30)	Not null	

	Dv_Type	varchar(20)	Not null	
DV_TableColumns	Dv_ID	smallint	Not null	PK_DV_TABLECOLUMNS primary key (ColumnId,TableID, DV_ID)
	TableID	int	Not null	FK_DV_TABLE_REFERENCE_DV_TABLE foreign key (TableID, DV_ID) references DV_Tables (TableID, DM_ID)
	ColumnID	int	Not null	PK_DV_TABLECOLUMNS primary key (ColumnId,TableID, DV_ID)
	TableName	varchar(100)		
	ColumnName	varchar(100)		
	OLTP_TableName	varchar(100)		
	BusinessKey	bit		
	DataType	varchar(100)		
	IsIdentity	bit		
	ColumnPK	bit		
	ColumnFK	bit		
	PKName	varchar(100)		
	FKName	varchar(100)		
	IfFK_PKTable	varchar(100)		
	IfFK_PKColumn	varchar(100)		
	DataSource	varchar(30)		
	FactCandidate	bit		
	DimCandidate	bit		
	DimTime	bit		
DV_Tables	TableID	int	Not null	PK_DV_TABLES primary key (TableID, DV_ID)
	DV_ID	smallint	Not null	PK_DV_TABLES primary key (TableID, DV_ID)
	TableName	varchar(100)		
	TableLevel	int		

Tabela 17. Struktura tabela metapodataka data marta

Tabela	**Naziv atributa**	**Prosto vrijedonosno ograničenje**		**Složeno vrijedonosno ograničenje (Međuzavisnost atributa više tabela)**
		Tip podatka	**Vrijednost**	
DataMart	DM_ID	smallint	Not null	PK_DATAMART primary key (DM_ID)
	DM_Name	varchar(30)	Not null	
	DM_Type	varchar(20)	Not null	
DM_TableColumns	DM_ID	smallint	Not null	PK_DM_TABLECOLUMNS primary key (ColumnId,TableID, DV_ID) FK_DM_TABLE_REFERENCE_DV_TABLE foreign key (TableID, DM_ID) references DM_Tables (TableID, DM_ID)
	TableID	int	Not null	
	ColumnID	int	Not null	PK_DM_TABLECOLUMNS primary key (ColumnId,TableID, DM_ID)
	TableName	varchar(100)		
	ColumnName	varchar(100)		
	DW_TableName	varchar(100)		
	ColumnPK	bit		
	ColumnFK	bit		
	PKName	varchar(100)		
	FKName	varchar(100)		
	IfFK_PKTable	varchar(100)		
	IfFK_PKColumn	varchar(100)		
DM_Tables	TableID	int	Not null	PK_DM_TABLES primary key (TableID, DM_ID)
	DM_ID	smallint	Not null	PK_DM_TABLES primary key (TableID, DM_ID)

	TableName	varchar(100)		
	TableLevel	int		

5.6.3.2. Projektovanje aplikacione logike

U programu je realizovana klasa *ClsBLL.cs* koje sadrži dio poslovne logike i sistemske operacije navedene u fazi analize. Najveći dio poslovne logike je smješten u uskladištenim procedurama i u tabelama pravila. Ovaj pristup je primjenjen iz razloga što u sistemu koji se realizuje postoje komplikovane transakcija koje zahtijevaju razne provjere i ažuriranje više od jedne tabele baze podataka. Prednost ovakve implementacije se ogleda prije svega u mogućnosti da se u slučaju promjene poslovne logike ili strukture baze podataka izvrši izmena samo uskladištene procedure ili u tabelama pravila, bez izmjene izvornog koda aplikacije.

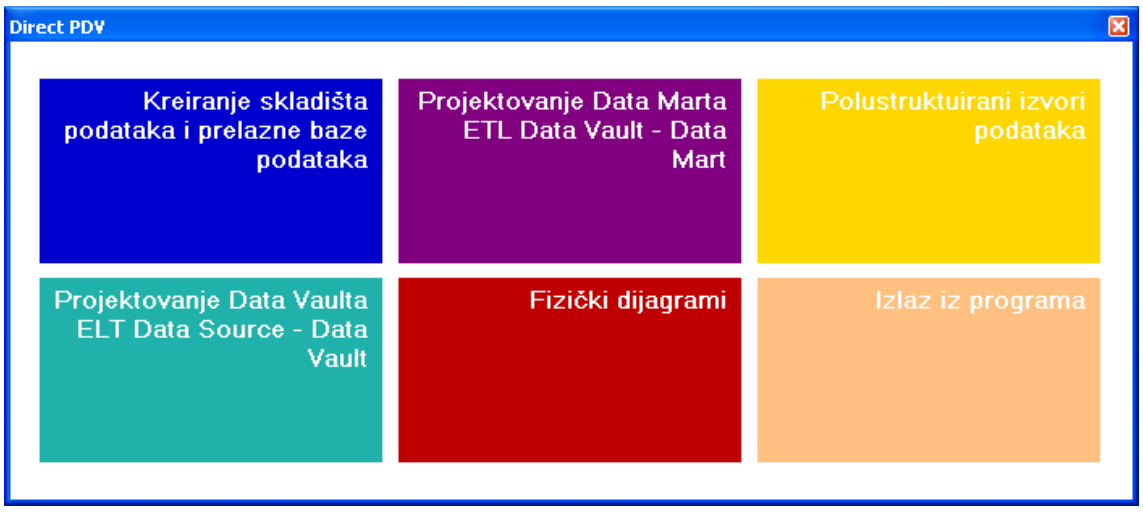
5.6.3.3. Projektovanje korisničkog interfejsa

Korisnički interfejs predstavlja realizaciju ulaza i/ili izlaza softverskog sistema i sastoji se od ekranskih formi čija je uloga da: prihvate podatke koje unosi Akter, prihvate događaje koje pravi Akter i prosleđuje sloju aplikacione logike unijete podatke [Vla04].

Na početku rada sa aplikacijom treba da se otvara Glavna ekranska forma za odabir određene aktivnosti kod projektovanja skladišta podataka. Iz glavne forme pozivaju se ostale forme za odgovarajuće slučajeve korišćenja. U slučaju uspješnog izvršenja sistemskih operacija, sistem će vratiti informaciju da je odgovarajuća operacija uspješno izvršena. U slučaju greške u izvršenju

sistemskih operacija sistem će vratiti odgovarajuću poruku o grešci. Pored glavne, predviđene su sljedeće ekranske forme:

- frm_1_1_CreateDatabases,
- frm_1_2_SelectDataSource,
- frm_1_3_SelectBusinessKeys,
- frm_1_4_CreateDataVaultTables,
- frm_1_5_ETL_DSDV,
- frm_2_1_CreateDataMart,
- frm_2_2_SelectDataVault,
- frm_2_3_SelectBusinessFact,
- frm_2_4_CreateDataMartTables,
- frm_2_5_ETL_DVDM.
- frm_3_1_AddSemiStructuredData
- frm_3_2_AddDataVaultTable
- frm_3_3_AddDimColumn
- frm_4_1_PhysicalDiagrams



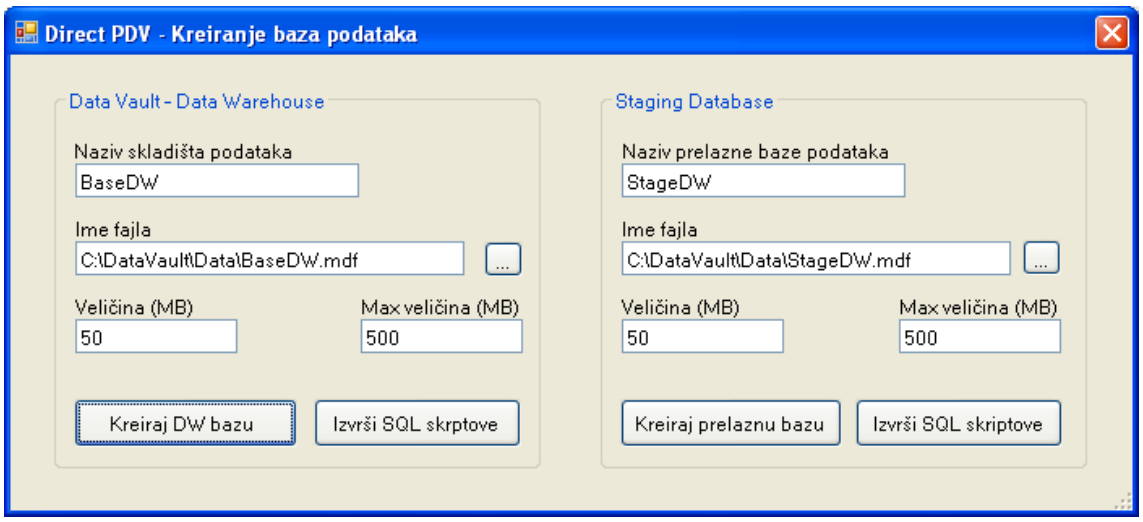
Slika 50. Glavna forma aplikacije

Slučaj korišćenja - 1.1. Kreiranje skladišta podataka i prelazne baze podataka. Forma **frm_1_1_CreateDatabases** (slika 51) treba da omogući kreiranje baze skladišta podataka i prelazne baze podataka na osnovu unijetih parametara (naziv baze, fizička lokacija, veličina i maksimalna veličina), te izvršavanje skriptova za kreiranje tabela metapodataka i pravila i popunjavanje tabela pravila.

Klikom na komandno dugme **Kreiraj DW bazu** pokreće se sistemska operacija *KreirajBazuPodataka (Database, ImeFajla, Velicina, MaxVelicina)* i kreira se baza skladišta podataka.

Klikom na komandno dugme **Kreiraj prelaznu bazu** pokreće se sistemska operacija *KreirajBazuPodataka (Database, ImeFajla, Velicina, MaxVelicina)* i kreira se prelazna baza skladišta podataka.

Klikom na komandno dugme **Izvrši SQL Skriptove** pokreće se sistemska operacija *IzvrsiSQLScript (Database, List <SqlScript>)* i izvršavaju SQL skriptovi iz zadane putanje na bazi skladišta ili prelaznoj bazi podataka.



Slika 51. Forma za slučaj korišćenja Kreiranje skladišta i prelazne baze

Slučaj korišćenja - 1.2. Izbor izvora podataka za skladište podataka. Forma **frm_1_2_SelectDataSource** (slika 52), treba da prikazuje listu izvora podataka

za skladište podataka, omogući izbor izvora podataka, snimi listu izabranog izvora podataka i učita metapodatke izabranog izvora podataka.

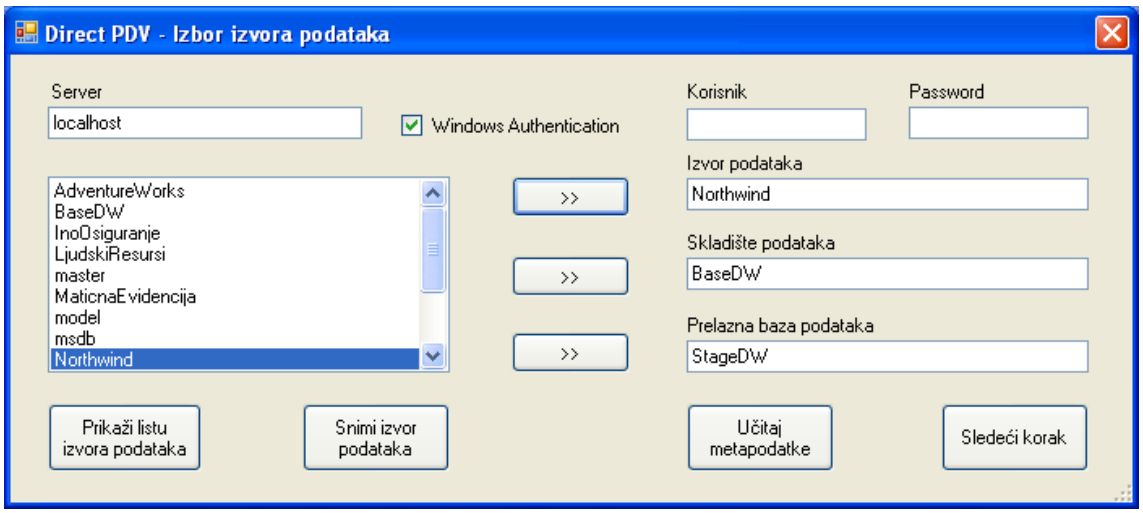
Klikom na komandno dugme **Prikaži listu izvora podataka** pokreće se sistemska operacija *PrikaziListuIzvoraPodataka (Server)* i korisniku prikazuju izvori podataka na izabranom serveru.

Klikom na komandna dugmad <strelica u desno> selektuju se izabrani izvori podataka, skladište podataka i prelazna baza podataka.

Klikom na komandno dugme **Snimi listu izvora podataka** pokreće se sistemska operacija *SnimiIzvorPodataka (IzvorPodataka)* u okviru koje se pokreće uskladištena procedura *sp_InsertDSTables.sql*, te se snima izabrani izvor podataka u tabelu DataSources.

Klikom na komandno dugme **Učitaj metapodatke** pokreće se sistemska operacija *UcitajMetaPodatke(IzvorPodataka)* u okviru koje se pokreće uskladištena procedura *sp_InsertTablesColumns.sql*, te se metapodaci snimaju u tabelu TableColumns.

Klikom na komandno dugme **Sljedeći korak** otvara se forma za sljedeći korak u procesu kreiranja skladišta podataka.



Slika 52. Forma za slučaj korišćenja Izbor izvora podataka

Slučaj korišćenja - 1.3. Identifikacija poslovnih ključeva. Forma **frm_1_3_SelectBusinessKeys** (slika 53), treba da omogući izbor i snimanje poslovnih ključeva, te identifikaciju hub, link i satelit tabela (kandidata). Forma treba da omogući prelazak na sljedeći korak u procesu kreiranja skladišta podataka.

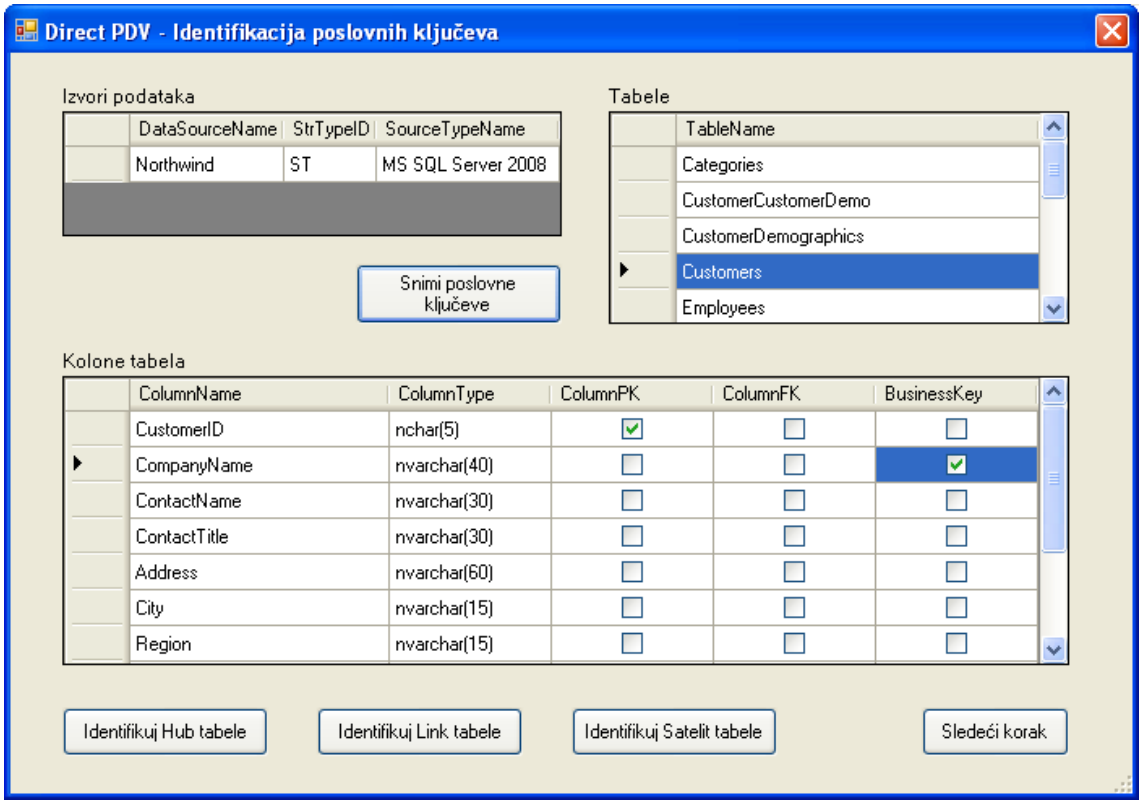
Klikom na komandno dugme **Snimi poslovne ključeve** pokreće se sistemska operacija *SnimiPoslovneKljuceve (Kolona, Vrijednost)* kojom se snima informacija o izabranim poslovnim ključevima.

Klikom na komandno dugme **Identifikuj Hub tabele** pokreće se sistemska operacija *IdentifikujHubTabele (Pravilo),* te se na osnovu pravila i poslovnih ključeva identifikuju hub tabele.

Klikom na komandno dugme **Identifikuj Link tabele** pokreće se sistemska operacija *IdentifikujLinkTabele (Pravilo),* te se na osnovu pravila i identifikovanih hub tabela, identifikuju link tabele.

Klikom na komandno dugme **Identifikuj satelit tabele** pokreće se sistemska operacija *IdentifikujSatelitTabele (Pravilo),* te se na osnovu pravila i identifikovanih hub i link tabela, identifikuju satelit tabele.

Klikom na komandno dugme **Sljedeći korak** otvara se forma za sljedeći korak u procesu kreiranja skladišta podataka.



Slika 53. Forma za slučaj korišćenja Identifikacija poslovnih ključeva

Slučaj korišćenja - 1.4. Kreiranje Data Vault tabela. Forma **frm_1_4_CreateDataVaultTables** (slika 54) treba da omogući eventualnu modifikaciju satelit kandidata, te kreiranje hub, link i satelit tabela na osnovu pravila. Forma treba da omogući povratak na prethodni korak i prelazak na sljedeći korak u procesu kreiranja skladišta podataka.

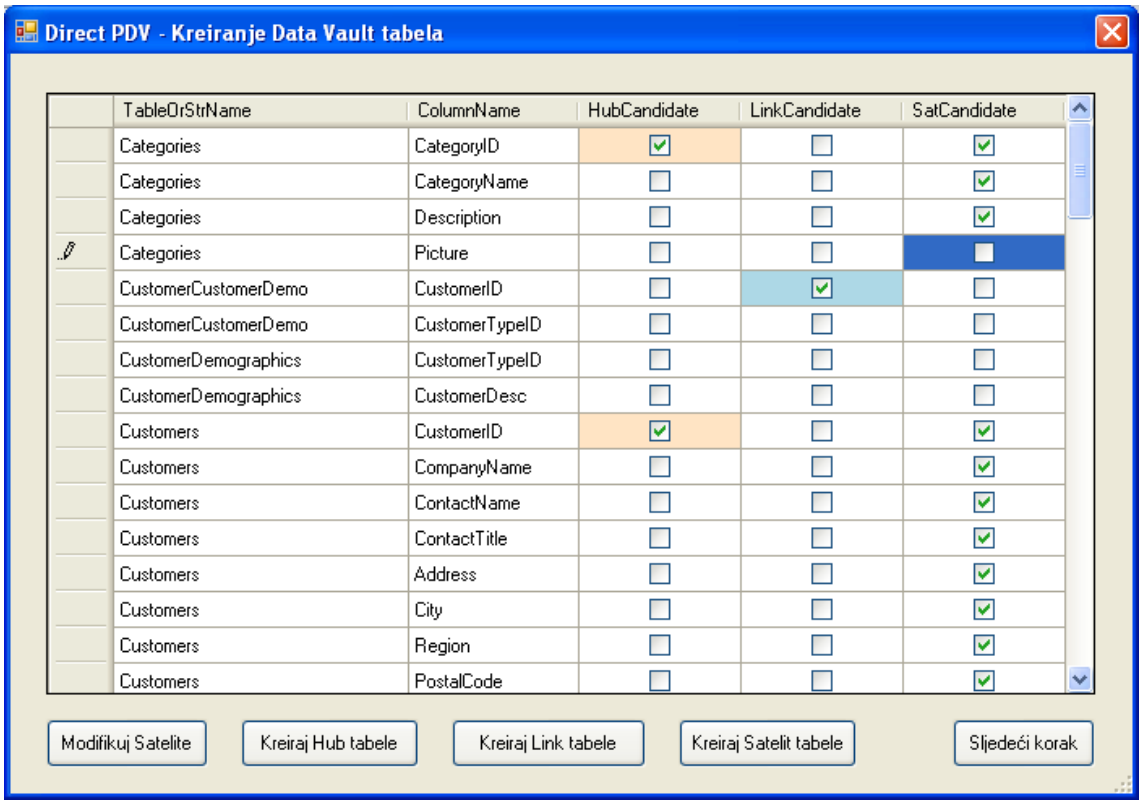
Klikom na komandno dugme **Modifikuj Satelite** pokreće se sistemska operacija *ModifikujSatelitKandidate (Kolona, Vrijednost)* kojom se snima informacija o naknadno izabranim satelit kandidatima.

Klikom na komandno dugme **Kreiraj Hub tabele** pokreće se sistemska operacija *KreirajHubTabele (Pravilo, SkladistePodataka)* u okviru koje se pokreće uskladištena procedura *sp_CreateHubTables.sql*, te se na osnovu pravila kreiraju hub tabele.

Klikom na komandno dugme **Kreiraj Link tabele** pokreće se sistemska operacija *KreirajLinkTabele (Pravilo, SkladistePodataka)* u okviru koje se pokreće uskladištena procedura *sp_CreateLinkTables.sql*, te se na osnovu pravila kreiraju link tabele.

Klikom na komandno dugme **Kreiraj Satelit tabele** pokreće se sistemska operacija *KreirajSatelitTabele (Pravilo, SkladistePodataka)* u okviru koje se pokreće uskladištena procedura *sp_CreateSatelitTables.sql*, te se na osnovu pravila kreiraju satelit tabele.

Klikom na komandno dugme **Sljedeći** korak otvara se forma za sljedeći korak u procesu kreiranja skladišta podataka.



Slika 54. Forma za slučaj korišćenja Kreiranje Data Vault tabela

Slučaj korišćenja - 1.5. Učitavanje podataka iz izvora podataka u Data Vault. Forma **frm_1_5_ETL_DSDV** (slika 55), treba da omogući prikazivanje Data

Vault metapodataka sa mapiranjem na odgovarajuće metapodatke izvora podataka, te ekstrakciju i učitavanje podataka iz izvora podataka u skladište podataka. Forma treba da ima mogućnost filtriranja hub, link i satelit tabela metapodataka. Osim toga, forma treba da ima mogućnost prikazivanja fizičkog dijagrama skladišta podataka i izvora podataka.

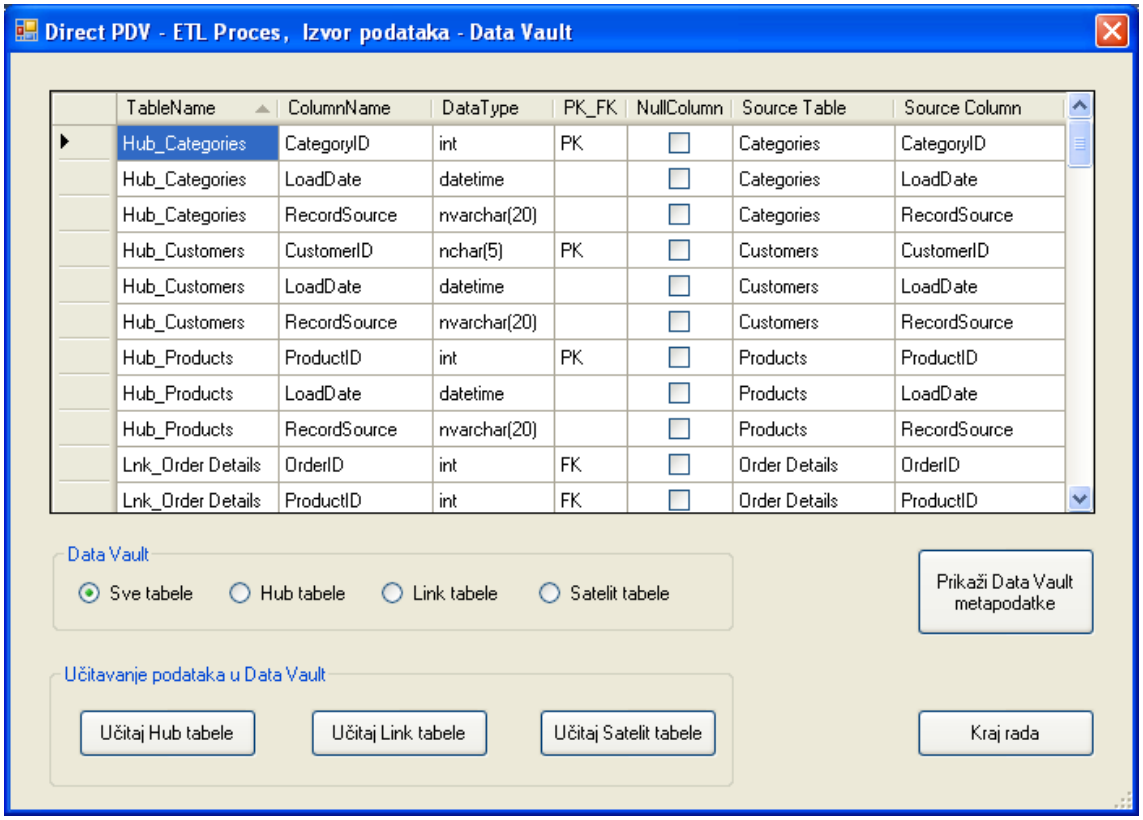
Klikom na komandno dugme **Prikaži Data Vault metapodatke** pokreće se sistemska operacija *PrikaziDataVaultMetaPodatke(SkladistePodataka,Prelaznabaza)* i korisniku prikazuju metapodaci izabranog skladišta podataka.

Klikom na komandno dugme **Učitaj Hub tabele** pokreće se sistemska operacija *UcitajHubTabele (Pravilo)* u okviru koje se pokreće uskladištena procedura *sp_InsertHub.sql,* te se na osnovu pravila učitavaju hub tabele.

Klikom na komandno dugme **Učitaj Link tabele** pokreće se sistemska operacija *UcitajLinkTabele (Pravilo)* u okviru koje se pokreće uskladištena procedura *sp_InsertLink.sql,* te se na osnovu pravila učitavaju link tabele.

Klikom na komandno dugme **Učitaj Satelit tabele** pokreće se sistemska operacija *UcitajSatelitTabele (Pravilo)* u okviru koje se pokreće uskladištena procedura *sp_InsertSatelit.sql,* te se na osnovu pravila učitavaju satelit tabele.

Klikom na komandno dugme **Kraj rada** aplikacija se vraća na Glavnu formu



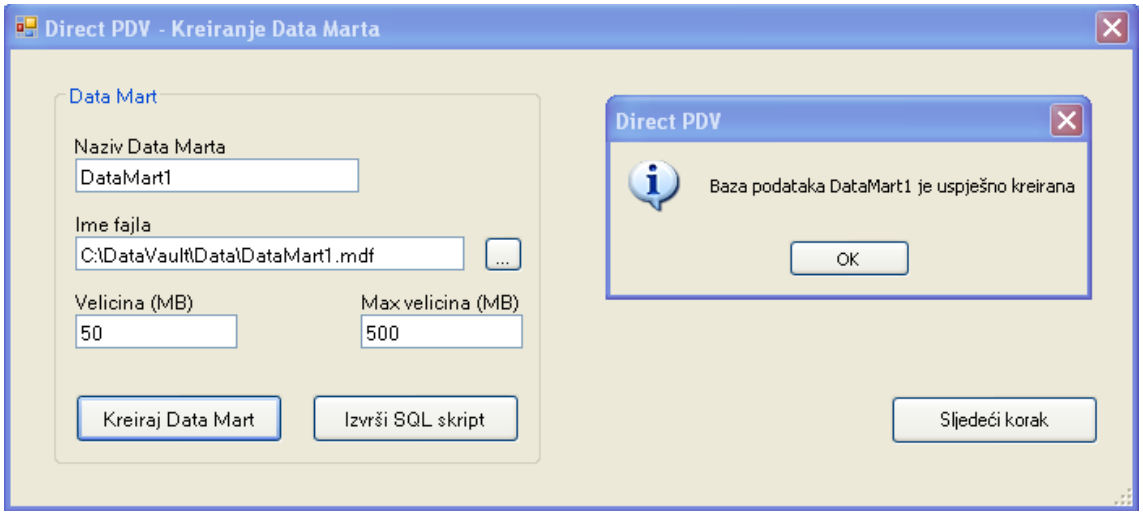
Slika 55. Forma za slučaj korišćenja Učitavanje podataka iz izvora podataka

Slučaj korišćenja - 2.1. Kreiranje Data Mart baze podataka. Forma **frm_2_1_CreateDataMart** (slika 56) treba da omogući kreiranje šeme ili Data Mart baze podataka na osnovu unijetih parametara (naziv baze, fizička lokacija, veličina i maksimalna veličina), te izvršavanje skriptova za kreiranje tabela metapodataka skladišta podataka i popunjavanje tabela pravila.

Klikom na komandno dugme **Kreiraj Data Mart** pokreće se sistemska operacija *KreirajBazuPodataka (Database, ImeFajla, Velicina, MaxVelicina)* i kreira se baza podataka Data Marta.

Klikom na komandno dugme **Izvrši SQL skript** pokreće se sistemska operacija *KreirajDMProcedureTabele (DatabaseName, List <SqlScript>)* i izvršavaju SQL skriptovi na zadatoj putanji.

Klikom na komandno dugme **Sljedeći korak** otvara se forma za sljedeći korak u procesu kreiranja Data Marta.



Slika 56. Forma za slučaj korišćenja Kreiranje data mart baze podataka

Slučaj korišćenja - 2.2. Izbor skladišta podataka. Forma **frm_2_2_SelectDataVault** (slika 57), treba da prikazuje listu izvora podataka za Data Mart, omogući izbor izvora (skladišta) podataka, snimi listu izabranog izvora podataka i učita metapodatke izabranog izvora (skladišta) podataka.

Klikom na komandno dugme **Prikaži listu izvora podataka** pokreće se sistemska operacija *PrikaziListuIzvoraPodataka (Server)* i korisniku prikazuju izvori (skladišta) podataka na izabranom serveru.

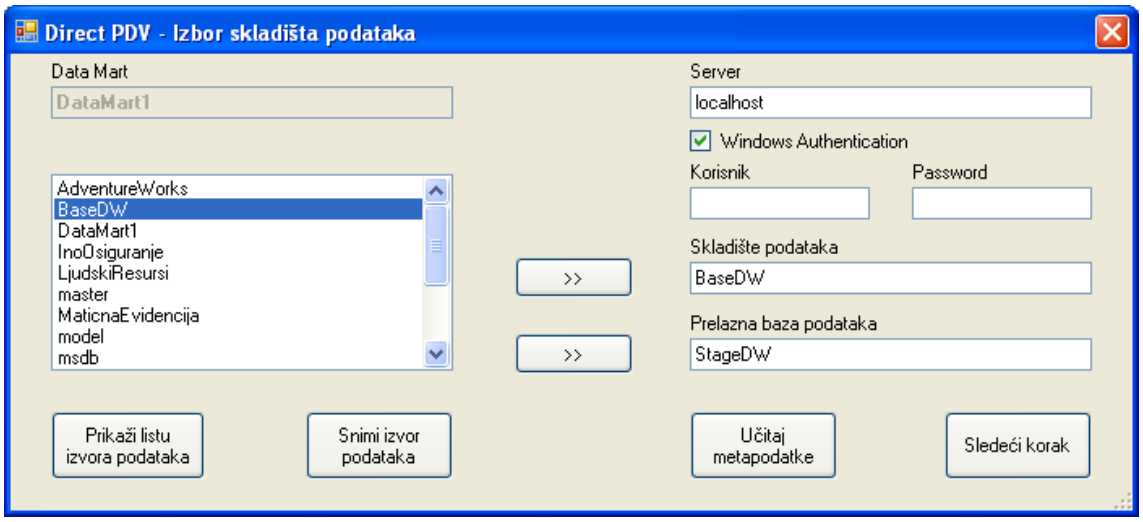
Klikom na komandno dugme <strelica u desno> selektuju se izabrani izvori podataka.

Klikom na komandno dugme **Snimi izvor podataka** pokreće se sistemska operacija *SnimiDataMartIzvorPodataka (SkladistePodataka)* u okviru koje se pokreće uskladištena procedura *sp_InsertDV_DataSource.sql*, te se snima izabrani izvor podataka u tabelu DataMart.

Klikom na komandno dugme **Učitaj metapodatke** pokreće se sistemska operacija *UcitajDVMetaPodatke (SkladistePodataka)* u okviru koje se pokreće

uskladištena procedura *sp_DV_metadata.sql*, te se metapodaci snimaju u tabelu DV_TableColumns.

Klikom na komandno dugme **Sljedeći korak** otvara se forma za sljedeći korak u procesu kreiranja skladišta podataka.



Slika 57. Forma za slučaj korišćenja Izbor skladišta podataka

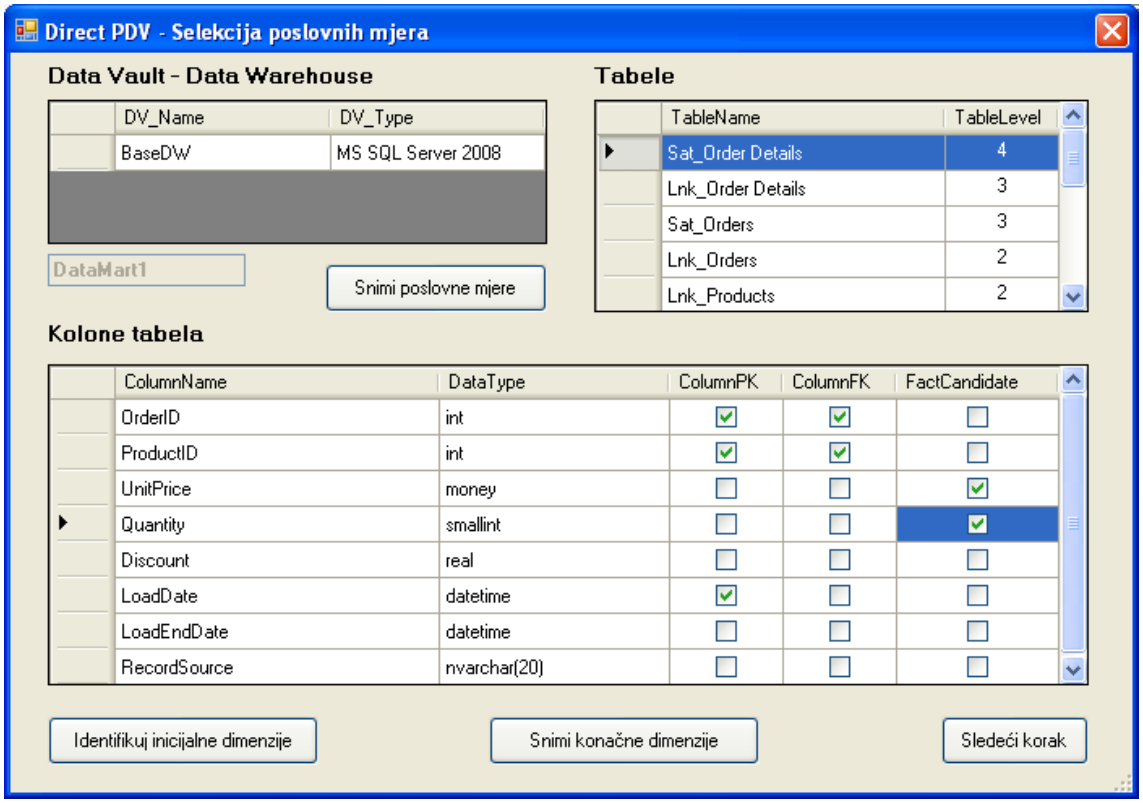
Slučaj korišćenja - 2.3. Identifikacija poslovnih mjera. Forma **frm_2_3_SelectBusinessFact** (slika 58), treba da omogući izbor i snimanje poslovnih mjera, te identifikaciju dimenzionih tabela (kandidata). Forma treba da omogući prelazak na sljedeći korak u procesu kreiranja skladišta podataka.

Klikom na komandno dugme **Snimi poslovne mjere** pokreće se sistemska operacija *SnimiPoslovneMjere (Kolona, Vrijednost)* kojom se snima informacija o izabranim poslovnim mjerama.

Klikom na komandno dugme **Identifikuj inicijalne dimenzije** pokreće se sistemska operacija *IdentifikujInicijalneDimenzije (Pravilo)*, te se na osnovu pravila i poslovnih mjera identifikuju dimenzione tabele.

Klikom na komandno dugme **Snimi konačne dimenzije** pokreće se sistemska operacija *SnimiKonacneDimenzije (Kolona, Vrijednost)* i snimaju eventualno ručno modifikovani kandidati za dimenzije.

Klikom na komandno dugme **Sljedeći korak** otvara se forma za sljedeći korak u procesu kreiranja Data Marta.



Slika 58. Forma za slučaj korišćenja Identifikacija poslovnih mjera

Slučaj korišćenja - 2.4. Kreiranje Data Mart tabela. Forma **frm_2_4_CreateDataMartTables** (slika 59) treba da omogući kreirnje tabele vremenskih dimenzija, određivanje načina agregacije mjera, upisivanje imena tabele mjera i kreiranje dimenzionih i tabela mjera. Forma treba da omogući prelazak na sljedeći korak u procesu kreiranja Data Marta.

Klikom na komandno dugme **Kreiraj vremensku dimenziju** pokreću se sistemske operacije *OznaciVremenskeDimenzije (Kolona, Vrijednost)* kojom se označavaju izabrane vremenske dimenzije i *KreirajTabeluVremenskihDimenzija (StartDate,EndDate,DataMart)*, u okviru koje se pokreće uskaldištena procedura *sp_CreateInsertDim_DateTime.sql*, a kojom se na osnovu izabrane vremenske

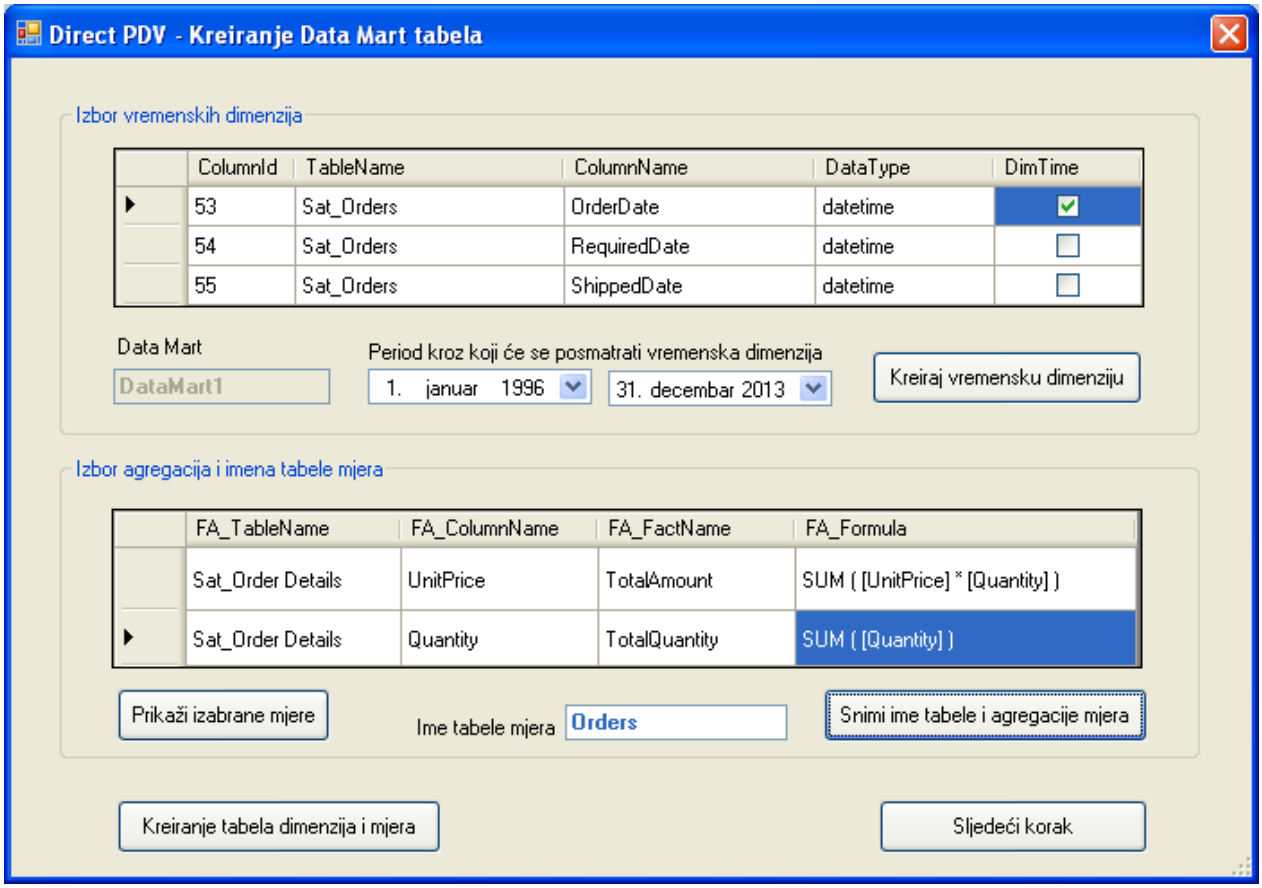
dimenzije i upisanog opsega posmatranja vremenske dimenzije kreira i popunjava tabela vremenskih dimenzija.

Klikom na komandno dugme **Prikaži izabrane mjere** pokreće se sistemska operacija *PrikaziIzabraneMjere (DataMart)* kojom se na gridu forme prikazuju izabrane mjere sa pripadajućim kolonama i tabelama. U ovom gridu korisnik treba da upiše naziv mjere i način agregacije mjera.

Klikom na komandno dugme **Snimi ime tabele i agregacije mjera** pokreće se sistemska operacija *SnimiNacinAgregacijeMjera (Kolona,FactName, Formula)* u okviru koje se upisju ime table mjera i formula za način agregacije mjera.

Klikom na komandno dugme **Kreiranje tabele dimenzija i mjera** pokreće se sistemska operacija *KreirajTabeleDimenzijaMjera (Pravilo, ImeTabeleMjera)* u okviru koje se pokreće uskladištena procedura *sp_CreateDimFactTables.sql*, te se na osnovu pravila kreiraju dimenzione i tabele mjera.

Klikom na komandno dugme **Sljedeći korak** otvara se forma za sljedeći korak u procesu kreiranja skladišta podataka.



Slika 59. Forma za slučaj korišćenja Kreiranje data mart tabela

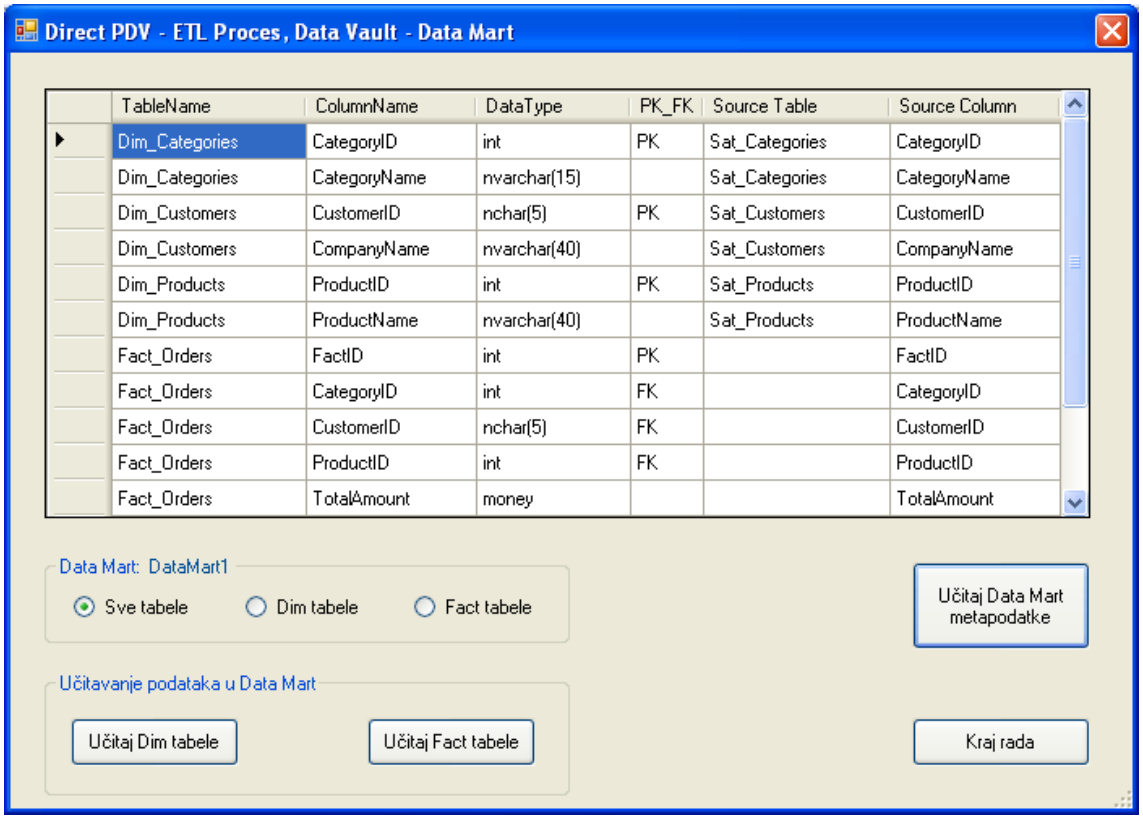
Slučaj korišćenja - SK - 2.5. Učitavanje podataka iz Data Vaulta u Data Mart. Forma **frm_2_5_ETL_DVDM** (slika 60), treba da omogući prikazivanje Data Mart metapodataka sa mapiranjem na odgovarajuće metapodatke izvora (skladišta) podataka, te ekstrakciju i učitavanje podataka iz izvora (skladišta) podataka u Data Mart. Forma treba da ima mogućnost filtriranja metapodataka dimenzionih i tabela mjera. Osim toga, forma treba da ima mogućnost prikazivanja fizičkog dijagrama skladišta podataka i Data Marta.

Klikom na komandno dugme **Prikaži Data Mart metapodatke** pokreće se sistemska operacija *PrikaziDataMartMetaPodatke(DataMart)* i korisniku prikazuju metapodaci izabranog Data Marta.

Klikom na komandno dugme **Učitaj Dim tabele** pokreće se sistemska operacija *UcitajDimTabele (Pravilo)* u okviru koje se pokreće uskladištena procedura *sp_InsertDim.sql,* te se na osnovu pravila učitavaju dimenzione tabele.

Klikom na komandno dugme **Učitaj Fact tabele** pokreće se sistemska operacija *UcitajFactTabele (Pravilo, DataMart)* u okviru koje se pokreće uskladištena procedura *sp_InsertFact.sql,* te se na osnovu pravila učitavaju tabele mjera.

Klikom na komandno dugme **Kraj rada** aplikacija se vraća na Glavnu formu



Slika 60. Forma za slučaj korišćenja Učitavanje podataka u data mart

Slučaj korišćenja - SK - 3.1. Izbor polustruktuiranog izvora podataka za skladište podataka. Forma **frm_3_1_AddSemistructuredData** (slika 61), treba da prikazuje list dostupnih skladišta podataka, Data Martova i polustruktuiranih fajlova koji će se koristiti u skladištu podataka, te njihov

izbor. Osim toga forma treba da omogući snimanje (učitavanje) polustruktuiranog fajla u prelaznu bazu podataka i učitavanje metapodatka izabranog polustruktuiranog fajla.

Klikom na komandno dugme **Prikaži listu baza podataka** pokreće se sistemska operacija *PrikaziListuIzvoraPodataka (Server)* i korisniku prikazuju izvori (skladišta) podataka na izabranom serveru.

Klikom na komandno dugme <strelica u desno> selektuje se izabrano skladište podataka.

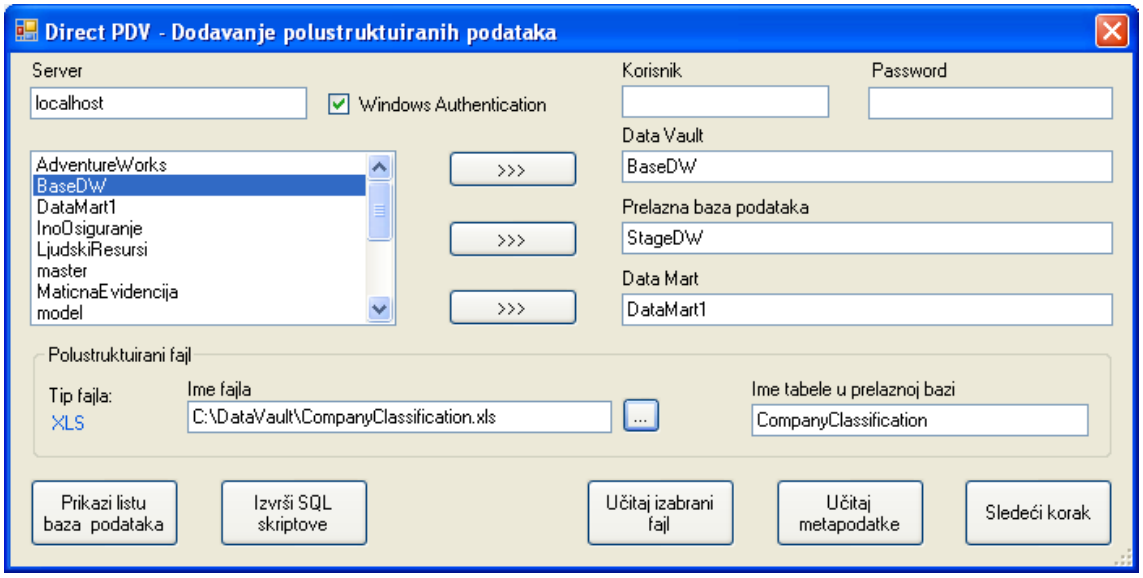
Klikom na komandno dugme <strelica u desno> (drugo) selektuje se izabrani Data Mart.

Klikom na komandno dugme **Izvrši SQL Skriptove** pokreće se sistemska operacija *IzvrsiSQLScript (Database, List <SqlScript>)* i izvršavaju SQL skriptovi iz zadane putanje na bazi skladišta ili prelaznoj bazi podataka potrebni za proces učitavanja polustruktuiranog izvora podataka u skladište podataka, kreiranje DataVault i Data Mart tabela i učitavnaje podataka u tabele.

Klikom na komandno dugme **Učitaj izabrani fajl** pokreće se sistemska operacija *SnimiPolustruktuiraniFajl (Pravilo)* u okviru koje se pokreće učitava polustruktuirani fajl u prelaznu bazu podataka u skladu sa pravilom za određenu vrstu fajla. U prelaznoj bazi kreira se materijalizovani pogled (engl. *view*) SS_Data.

Klikom na komandno dugme **Učitaj metapodatke** pokreće se sistemska operacija *UcitajSSMetaPodatke (TipFajla, ImeFajla, PrelaznaBaza)* u okviru koje se pokreće uskladištena procedura *sp_SS_metadata.sql,* te se metapodaci snimaju u tabelu SS_Metadata.

Klikom na komandno dugme **Sljedeći korak** otvara se forma za sljedeći korak u procesu kreiranja skladišta podataka.



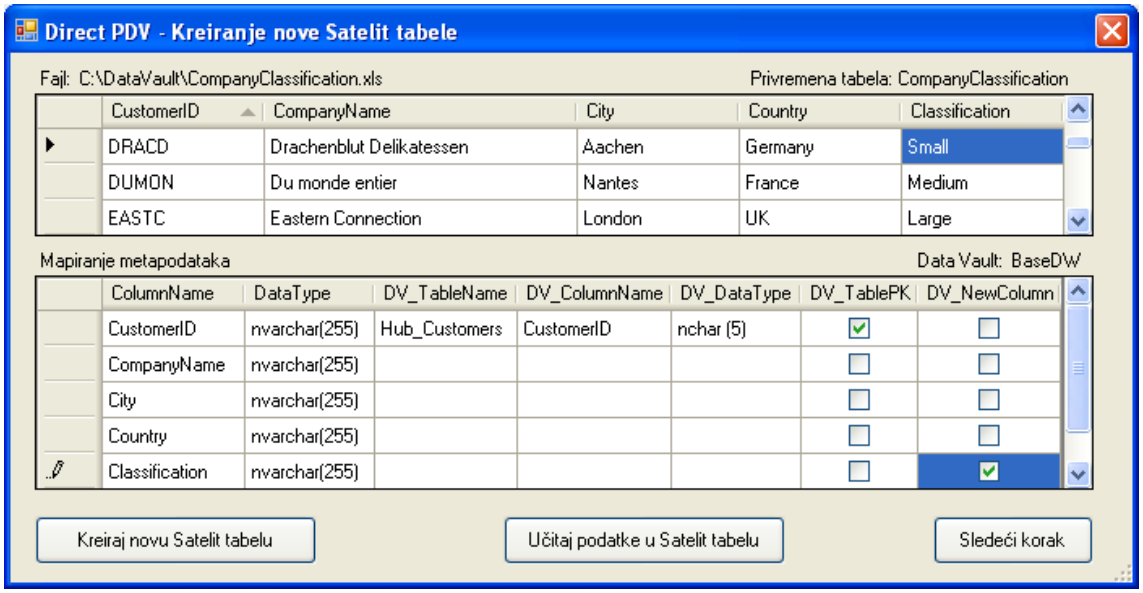
Slika 61. Forma za slučaj korišćenja Izbor polustruktuiranog izvora podataka

Slučaj korišćenja - SK - 3.2. Kreiranje i učitavanje novih Data Vault tabela. Forma **frm_3_2_AddDataVaultTable** (slika 62) treba da omogući pregled učitanog polustruktuiranog fajla i mapiranje metapodataka fajla sa metapodacima postojećih Data Vault tabela.

Klikom na komandno dugme **Kreiraj novu Satelit tabelu** pokreće se sistemska operacija *KreirajNovuSatelitTabelu (Pravilo, SkladistePodataka)* u okviru koje se pokreće uskladištena procedura *sp_CreateSatTableSS.sql*, te se na osnovu pravila kreiraju satelit tabele.

Klikom na komandno dugme **Učitaj podatke u Satelit tabelu** pokreće se sistemska operacija *UcitajSatelitTabeluSS (Pravilo, Fajl)* u okviru koje se pokreće uskladištena procedura *sp_InsertSatTableSS.sql*, te se na osnovu pravila učitavaju podaci u novu Satelit tabelu iz polustruktuiranog fajla.

Klikom na komandno dugme **Sljedeći korak** otvara se forma za sljedeći korak u procesu kreiranja skladišta podataka.

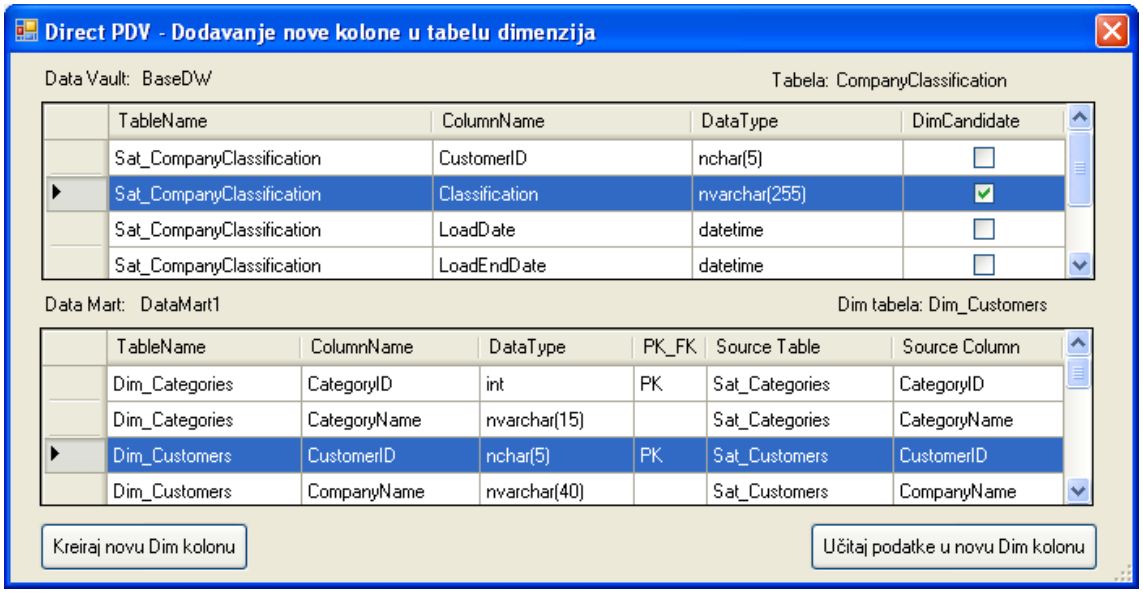


Slika 62. Forma za slučaj korišćenja Kreiranje i učitavanje novih DV tabela

Slučaj korišćenja - SK - 3.3. Kreiranje i učitavanje novih Data Mart tabela. Forma **frm_3_3_AddDataVaultColumn** (slika 63) treba da omogući pregled metapodataka nove Data Vault tabele kreirane na osnovu polustruktuiranog fajla i mapiranje metapodataka navedene tabele sa metapodacima postojećih Data Mart tabela.

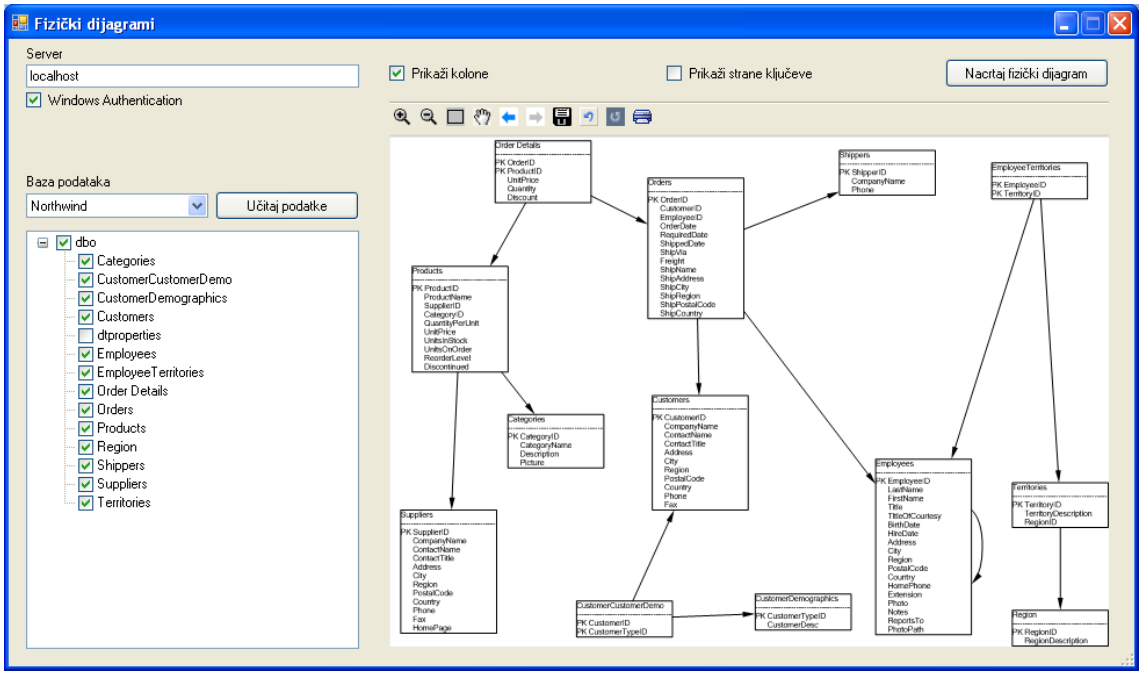
Klikom na komandno dugme **Kreiraj novu Dim kolonu** pokreće se sistemska operacija *KreirajNovuDimKolonu (Pravilo, SatTabela, DimTabela)* u okviru koje se pokreće uskladištena procedura *sp_AddDimColumnSS.sql*, te se na osnovu pravila kreira nova kolona izabrane dimenzione tabele.

Klikom na komandno dugme **Učitaj podatke novu Dim kolonu** pokreće se sistemska operacija *UcitajNovuDimKolonu (Pravilo, DataVault, SatTabela, DataMart, DimTabela)* u okviru koje se pokreće uskladištena procedura *sp_UpdateDimSS.sql*, te se na osnovu pravila učitavaju podaci u novu kolonu izabrane dimenzione tabele iz izabrane satelit tabele.



Slika 63. Kreiranje novih kolona data mart tabela

Klikom na dugme *Fizički dijagrami* na glavnoj formi, pokreće se pomoćna forma koja je napravljena za pregled fizičkih dijagrami dostupnih baza podataka potencijalnih izvora podataka. Osim toga, ova forma omogućuje pregled fizičkog modela kreiranog Data Vault skladišta podataka i fizičkog modela tabela mjera i dimenzija data marta. Ovim je omogućen pregled fizičkog dijagrama skladišta podataka u bilo kojem trenutku projektovanja skladišta bez izlaska iz programa Direct PDV ili korišćenjem drugih alata za prikaz fizičkog modela skladišta podataka. Forma **frm_4_1_PhysicalDiagrams** je prikazana na sljedećoj slici.

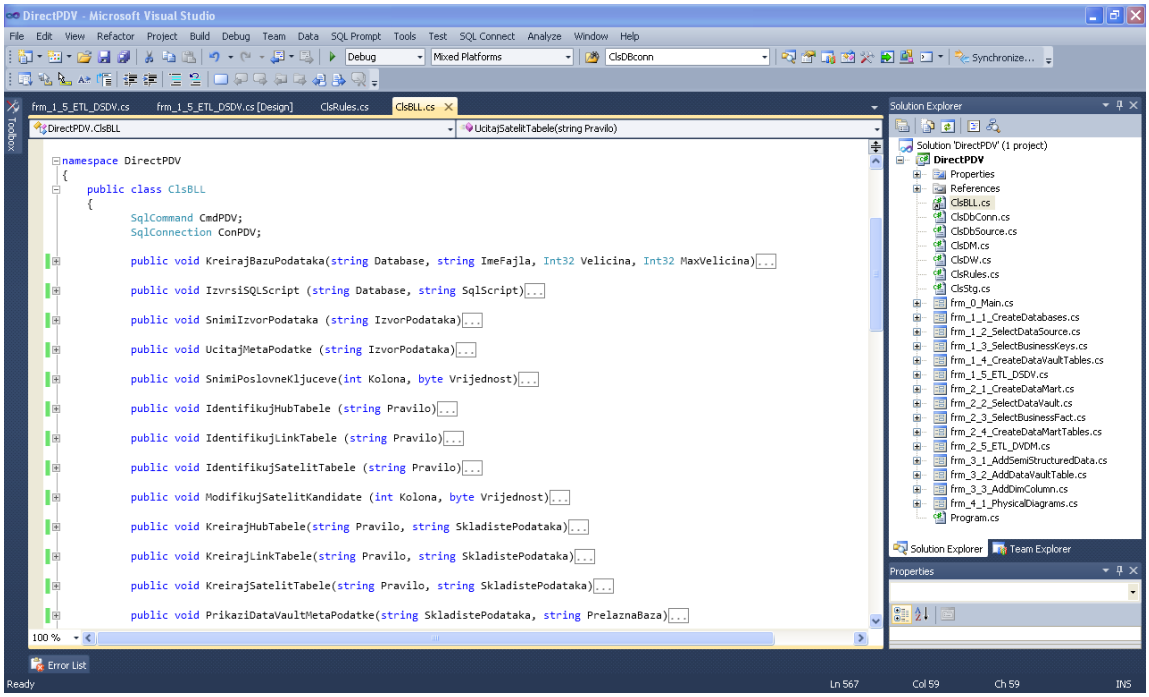


Slika 64. Forma za pregled fizičkog modela dostupnih baza podataka, uključujući i skladište podataka

Forma je urađena korišćenjem komponente Microsoft Automatic Graph Layout (MSAGL) [Msagl]. koji je dostupan na linku http://research.microsoft.com/en-us/projects/msagl/. MSAGL je .NET alat za pregled i izradu grafova koji je u Microsoftu razvio Lev Nachmanson [msagl]. MSAGL koristi principe Sugiyama scheme [STT81].

5.6.4. Implementacija

Softverski alat za podršku projektovanju skladišta podataka Direct PDV, kao rezultat ovog rada razvijen je u okruženju Microsoft Visual Studio .NET 2010, programski jezik Microsoft C#. Kao sistem za upravljanjem bazom podataka korišćen je Microsoft SQL Server 2008 R2.



Slika 65. Razvojno okruženje programa Microsoft Visual Studio 2010

Radi ilustracije u nastavku je dat dio C# koda:

a) događaja *Click* komandnog dugmeta *IdentifikujHubTabele* (koje se nalazi na formi *frm_1_3_SelectBusinessKeys.frm*)

```
private void buttonIdentifikujHubTabele_Click_1(object sender, EventArgs e)
        {
            string Pravilo = "HubCandidate";
            ClsBLL obj = new ClsBLL();
            obj.IdentifikujHubTabele(Pravilo);
        }
```

b) sistemske operacije *IdentifikujHubTabele (Pravilo)* kojoj se prosljeđuje odgovarajuće pravilo iz tabele *Rules*

```
public void IdentifikujHubTabele (string Pravilo)
            {
                //Identifikacija hub tabela na osnovu pravila (posl.ključeva)
                // Konekcioni string za prelaznu bazu podataka
                string ImeBaze = ClsStg.PrelaznaBaza;
                ClsDbConn obj1 = new ClsDbConn();
                string ConStrPDV = obj1.VratiKonString(ImeBaze);
```

```
        // Pronalaženje pravila za "HubCandidate"
        ClsRules obj2 = new ClsRules();
        string str = obj2.VratiPravilo(Pravilo);
        // sql komanda ili procedura iz tabele pravila
        ConPDV = new SqlConnection(ConStrPDV);
        CmdPDV = new SqlCommand(str, ConPDV);
        CmdPDV.CommandText = str;
        try
        {
        // otvaranje konekcije
        ConPDV.Open();
        // izvršavanje sql komande
        CmdPDV.ExecuteNonQuery();
       MessageBox.Show("Hub tabele su uspješno identifikovane",
       "Direct PDV", MessageBoxButtons.OK,
       MessageBoxIcon.Information);
        }
        catch (SqlException ex)
        {
       // obrada greške
       MessageBox.Show("Greška: Nije moguće identifikovati Hub
       tabele" + "\n" + "Originalna poruka: " + ex.Message, "Direct
       PDV", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        finally
        {
        if (ConPDV.State == ConnectionState.Open)
        {
            // zatvaranje sql konekcije
            ConPDV.Close();
        }
        }

}
```

5.6.5. Testiranje

U skladu sa arhitekturom softverskog sistema, ova faza se je podijeljena u nekoliko nezavisnih dijelova testiranja. Jedinično testiranje se odnosilo na svaki slučaj korišćenja, sistemsku operaciju i pojedinačnu ekransku formu kao softversku komponentu aplikacije. Osim toga kroz test slučajeve su testirane uskladištene procedure aplikacije.

Nakon testiranja softverskih komponenti izvršena je njihova integracija. Zadnji korak u ovoj fazi je bio integracioni test koji je podrazumijevao sljedeće aktivnosti:

- Test automatizacije kreiranja skladišta podataka baziranog na Data Vault konceptu
- Test automatizacije ETL procesa izvor podataka – skladište podataka
- Test automatizacije kreiranja Data Marta
- Test automatizacije ETL procesa skladište podataka – Data Mart
- Test automatizacije dodavanja polustruktuiranih podataka u skladište podataka
- Test kreiranja fizičkih dijagrama izvora podataka, skladišta podataka i data marta

6. PRAKTIČNA PROVJERA REZULTATA ISTRAŽIVANJA

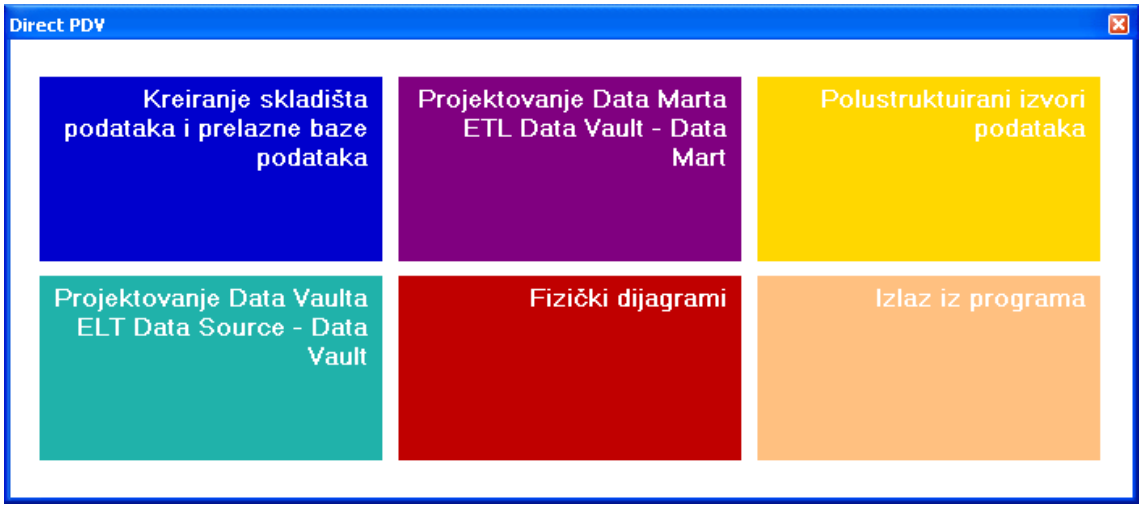
Eksperimentalna provjera rezultata istraživanja zasnovana na prototipu aplikacije Direct PDV urađena je na primjeru iz zdravstvenog osiguranja.

6.1. Zahtjevi

Zahtjevi: Fond za zdravstveno osiguranje godišnje sa apotekama sklapa ugovore o izdavanju lijekova na recept osiguranicima fonda. Lijek na recept propiše zdravstvena ustanova. Svakih petnaest ili trideset dana (u zavisnosti od ugovora), apoteka Fondu šalje račun za lijekove izdate na recept. Račun se sastoji od zaglavlja i stavki. Zaglavlje sadrži ime apoteke, datum fakture, ukupan iznos i broj recepata. Stavke fakture sadrže informaciju o izdatom lijeku, dijagnozi i osiguraniku (pacijentu) kojem je lijek izdan. Računi se dostavljaju elektronski u xml formatu. Kako se radi se o distribuiranoj bazi podataka, import podataka o receptima se radi u transakcionim bazama fonda, po organizacionim jedinicama fonda - filijalama. Broj slogova u tabeli računi i stavke računa na godišnjem nivou je nekoliko miliona. Da bi se smanjilo opterećenje transakcionog sistema, pregledali izvještaji kojima se često mijenja format i izgled, ispunili zahtjevi korisnika u pogledu izvještaja sa različitim grupisanjima i dijagramima, potrebno je realizovati skladište podataka i odgovarajući sistem poslovne inteligencije. Kod izbora arhitekture skladišta podataka, izabran je Data Vault koncept koji naglašava potrebu da se ostavi trag odakle i kad su podaci došli u bazu podataka. Osim toga, ovaj koncept je osmišljen da se model podataka može lako promijeniti u skladu sa promjenama u poslovnom okruženju. Osim podataka iz transakcionih podataka potrebno je omogućiti skladištenje podataka iz eksternih polustruktuiranih izvora podataka (Microsoft Office Excel fajl).

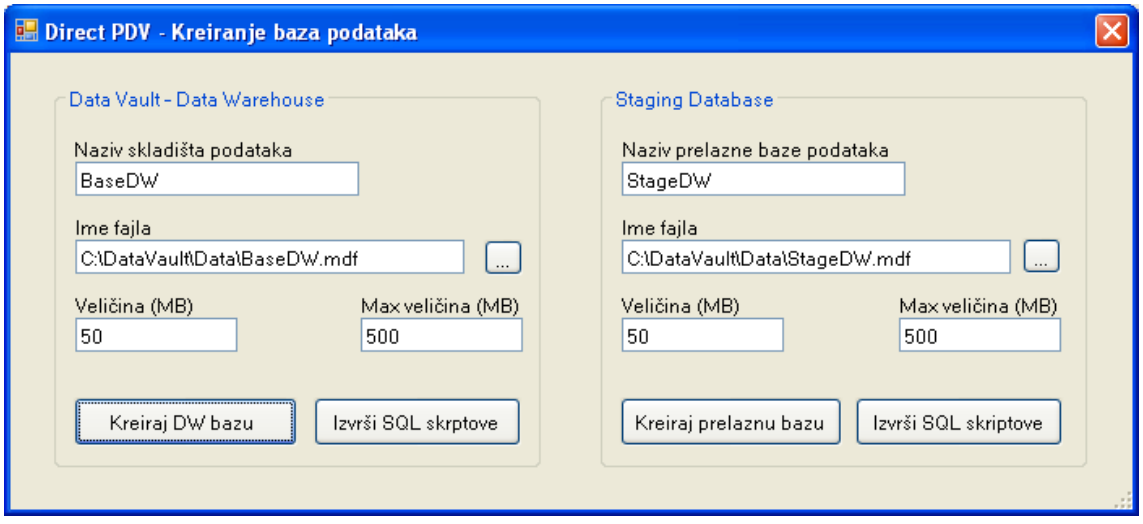
6.2. Projektovanje skladišta podataka

Proces kreiranja skladišta podataka počinje pokretanjem glavne forme aplikacije koja je prikazana na sljedećoj slici.



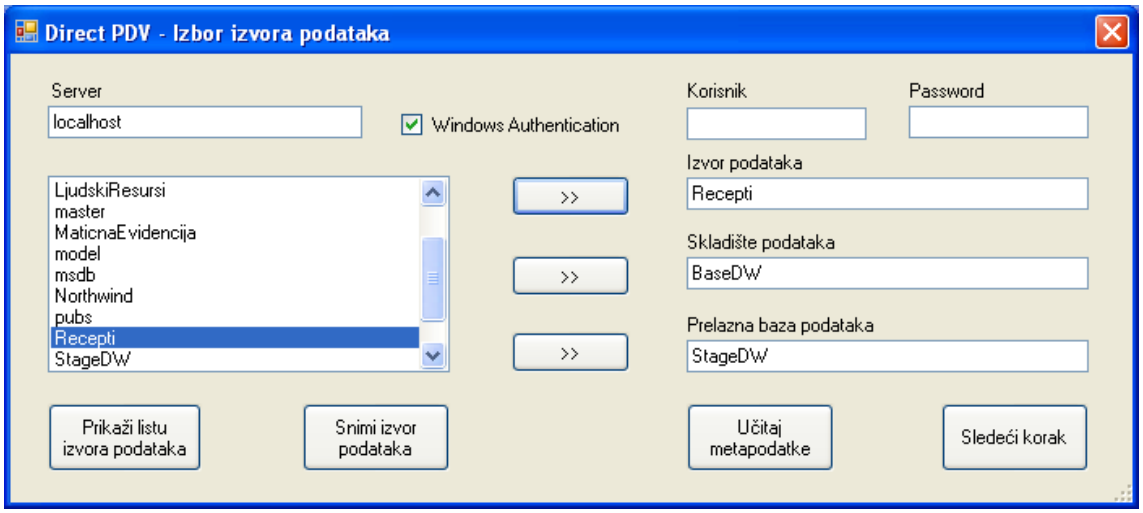
Slika 66. Glavna forma aplikacije

Bira se prva opcija **Kreiranje skladišta podataka i prelazne baze podataka**, nakon čega se otvara forma prikazana na sljedećoj slici.



Slika 67. Forma za kreiranje skladišta podataka i prelazne baze podataka

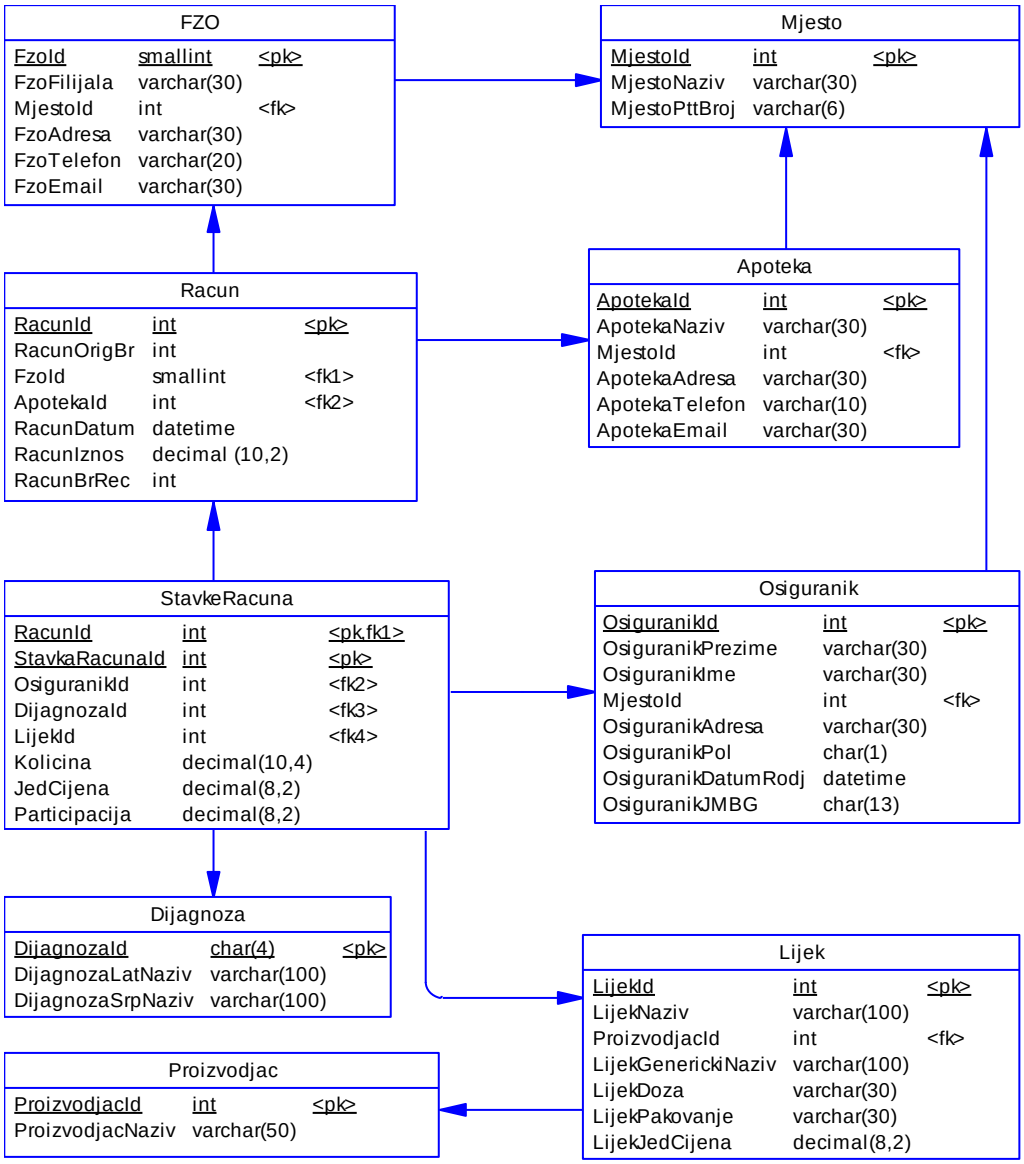
Nakon jednostavnog kreiranja skladišta podataka i prelazne baze podataka i pripadajućih procedura koje su generisane iz modela, prelazi se na drugu međufazu, izbor izvora podataka za skladište podataka. Forma za podršku ove međufaze pokreće se klikom na glavnoj formi **Kreiranje Data Vault sistema ETL Data Source – Data Vault**. Nakon ove akcije otvara se forma prikazana na skljedećoj slici.



Slika 68. Forma za izbor izvora podataka za skladište podataka

U ovom slučaju izabrana je relaciona transakciona baza podataka **Recepti**. U toku kreiranja skladišta podataka korisnik može u svakom momentu da vidi fizički model izabranog izvora podataka klikom na komandno dugme **Fizički dijagrami** na glavnoj formi.

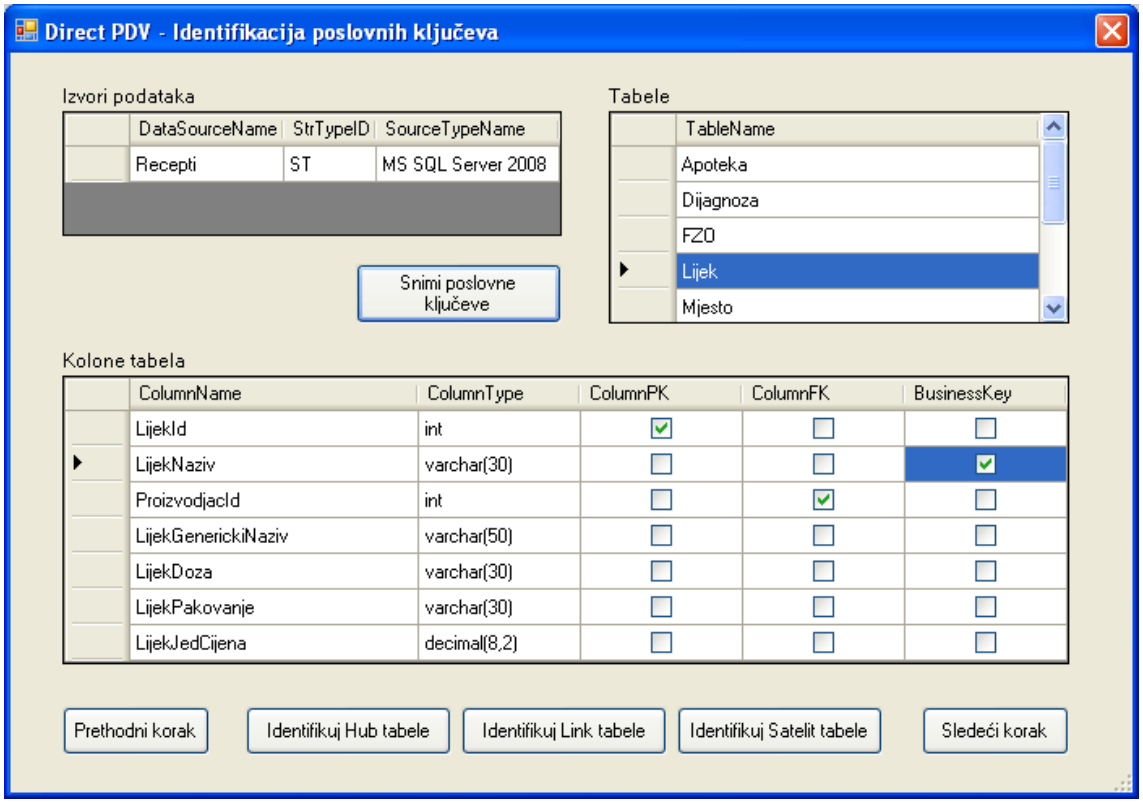
Dio fizičkog modela transakcione relacione baze podataka Recepti dat je na sljedećoj slici.



Slika 69. Fizički model transakcione baze Recepti

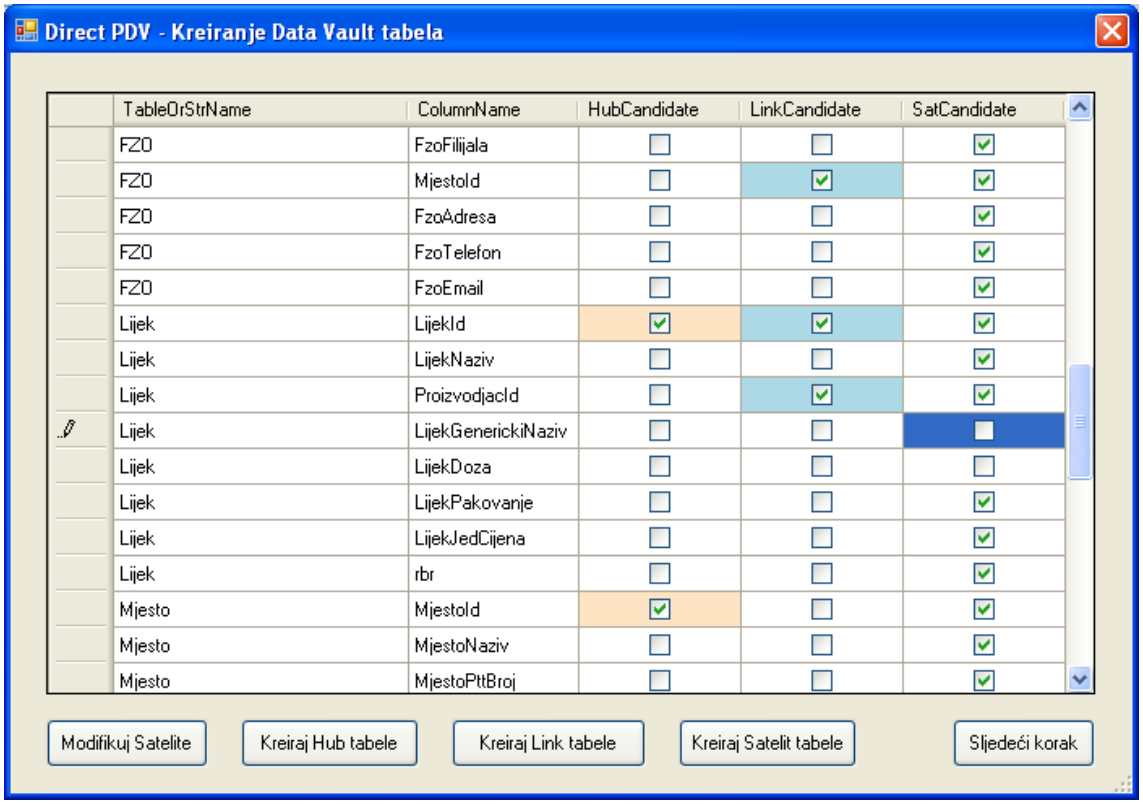
Nakon izbora i učitavanja izvora podataka, korisnik pokreće proces učitavanja metapodataka izabranih izvora podataka klikom na dugme **Učitaj metapodatke.** Klikom na dugme **Sljedeći korak,** otvara se forma za sljedeću međufazu. Na ovoj formi korisnik istovremeno na tri DataGridView komponente ima mogućnost pregleda izabranog izvora podataka, tabele izvora podataka i kolona pojedinih tabela. Na trećem DataGridView dizajner skladišta

podataka vrši izbor poslovnih ključeva čekiranjem odgovarajućih kolona tabele. Ova forma je prikazana na sljedećoj slici.



Slika 70. Forma za identifikaciju poslovnih ključeva

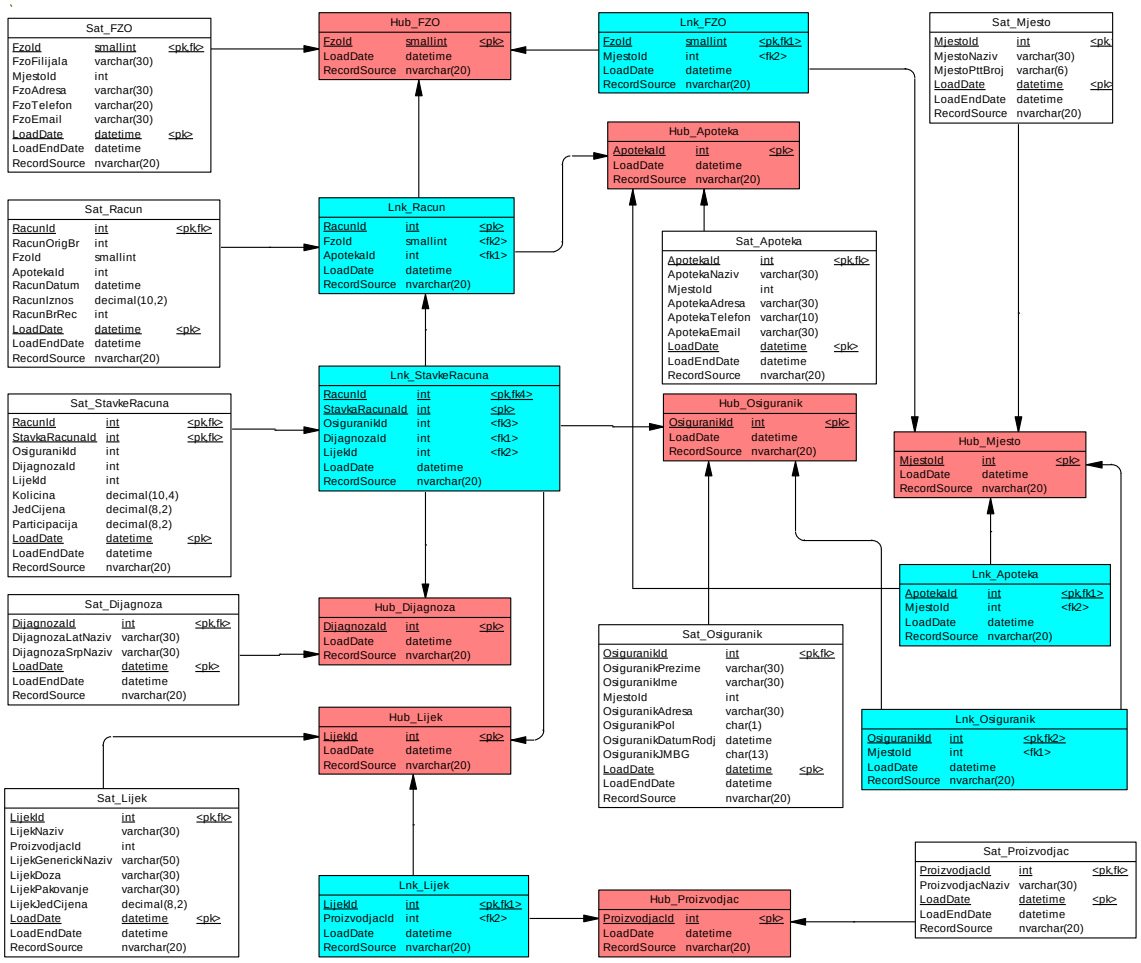
Na osnovu poslovnih ključeva i odgovarajućih pravila sistem identifikuje hub tabele, a na osnovu hub tabela i pravila za link tabele identifikuje link tabele. Na kraju ove faze korisnik na osnovu hub i link tabela, identifikuje satelit tabele. Za identifikaciju služe odgovarajuća komandna dugmad na formi. Nakon klika na dugme sljedeći, pokreće se sljedeći korak i otvara se forma za kreiranje Data Vault tabela koja je prikazana na sljedećoj slici.



Slika 71. Forma za kreiranje Data Vault tabela

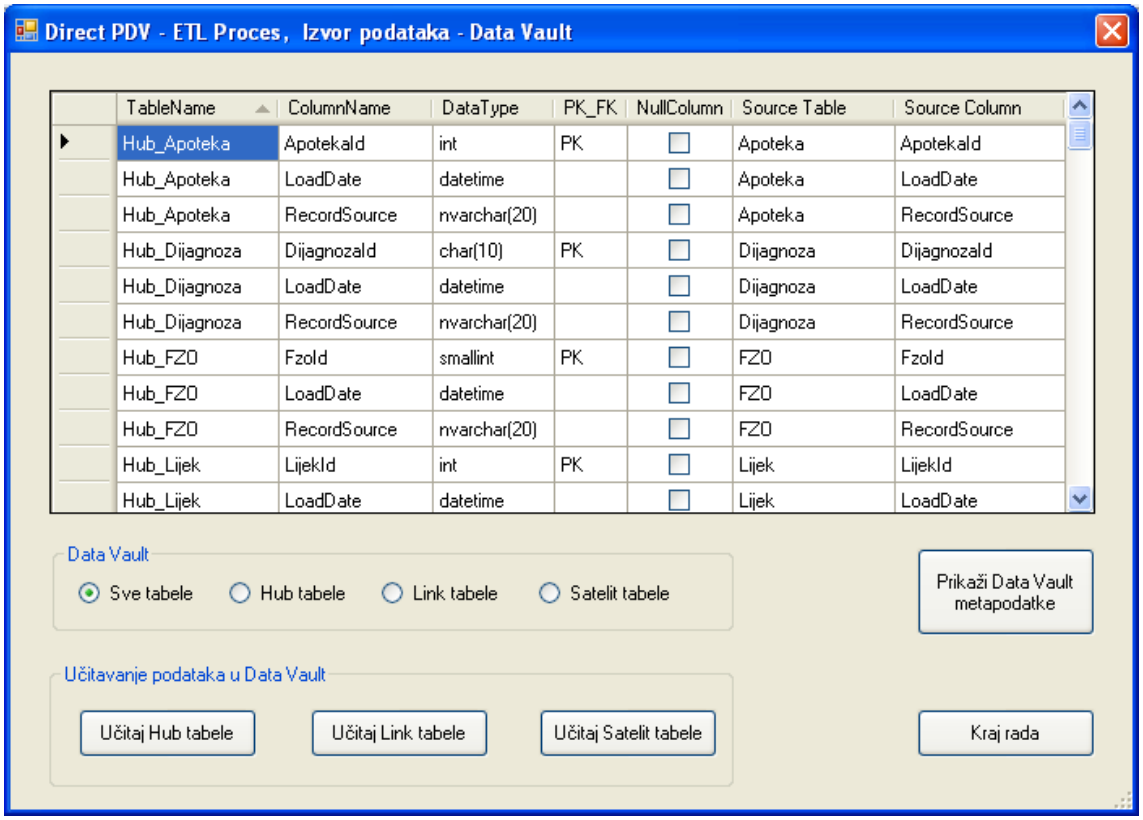
Na ovoj formi omogućena je manuelna modifikacija predloženih satelit tabela. Nakon što su snimljene manuelne modifikacije, dizajner pokreće proces kreiranja Hub, Link i Satelit tabela koje se kreiraju u bazi skladišta podataka. Ovim korakom se završava proces kreiranja skladišta podataka baziranog na Data Vault konceptu.

U toku kreiranja skladišta podataka korisnik može u svakom momentu da vidi fizički model izabranog izvora podataka klikom na komandno dugme **Fizički dijagrami** na glavnoj formi. Klikom na ovo dugme korisniku je omogućeno da vidi fizički model kreiranog Data Vault skladišta podataka. Fizički model kreiranog skladišta podataka prikazan je na sljedećoj slici.



Slika 72. Fizički model Data Vault skladišta podataka Recepti

Nakon kreiranja skladišta podataka, stvoreni su uslovi za ETL proces, odnosno ekstrakcija podataka iz izvora podataka i učitavanje podataka u Data Vault skladište podataka. Klikom na dugme Sljedeći na prethodnoj formi, otvara se forma za aktivnosti učitavanja podataka u skladište podataka koja je prikazana na sljedećoj slici.

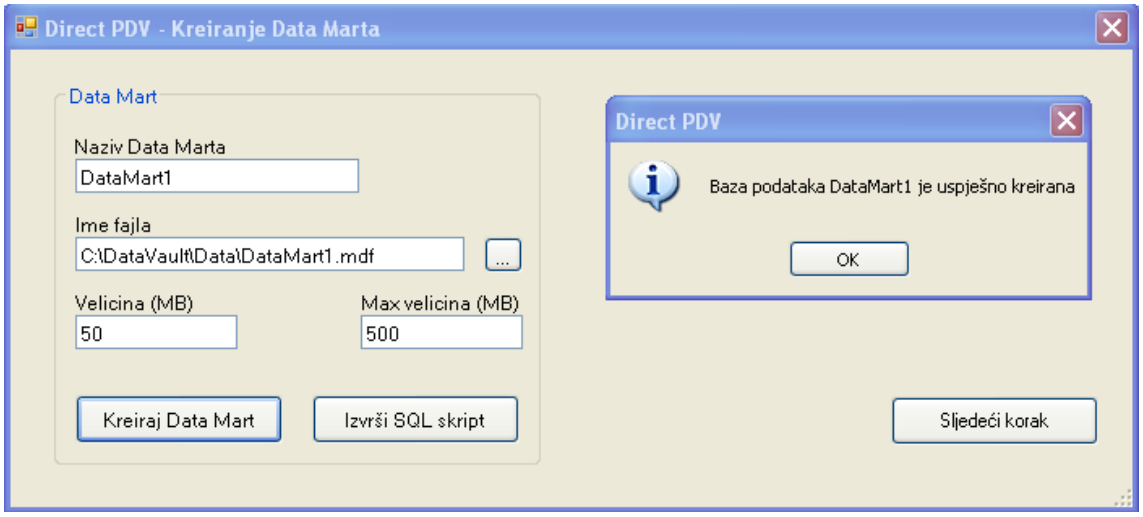


Slika 73. Forma za učitavanje podataka iz izvora podataka u Data Vault

Na ovoj dizajner skladišta podataka ima mogućnost učitavanja skladišta podataka, pregled mapiranja pojedinih tabela i kolona izvora i skladišta podataka, te učitavanje podataka u hub, link i satelit tabele (navedenim redoslijedom).

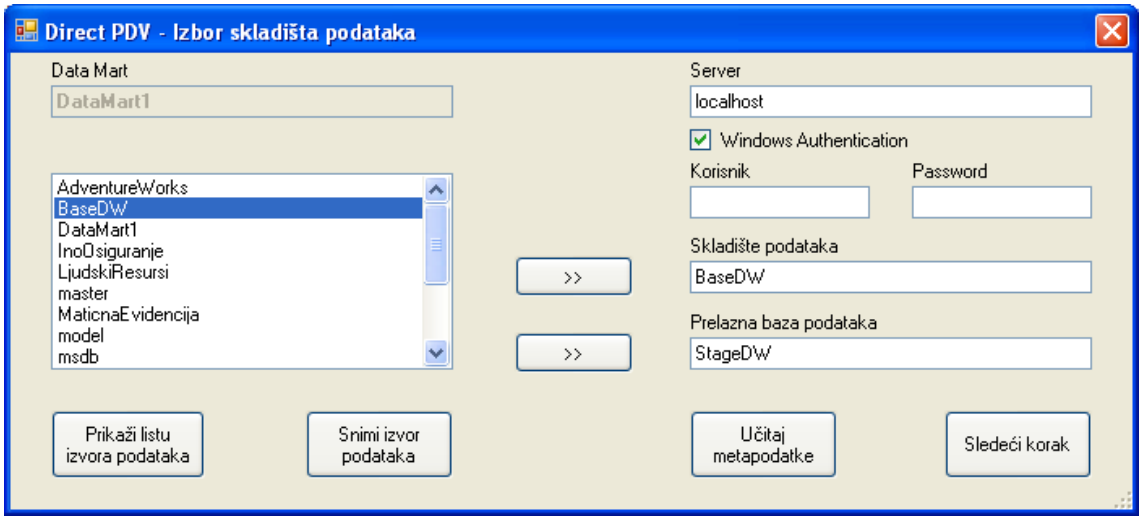
6.3. Projektovanje data marta

Proces kreiranja data Marta počinje pokretanjem glavne forme aplikacije koja je prikazana na sljedećoj slici. Bira se opcija **Kreiranje Data Mart sistema, ETL Data Source – Data Vault**, nakon čega se otvara forma prikazana na sljedećoj slici.



Slika 74. Forma za kreiranje data mart baze podataka

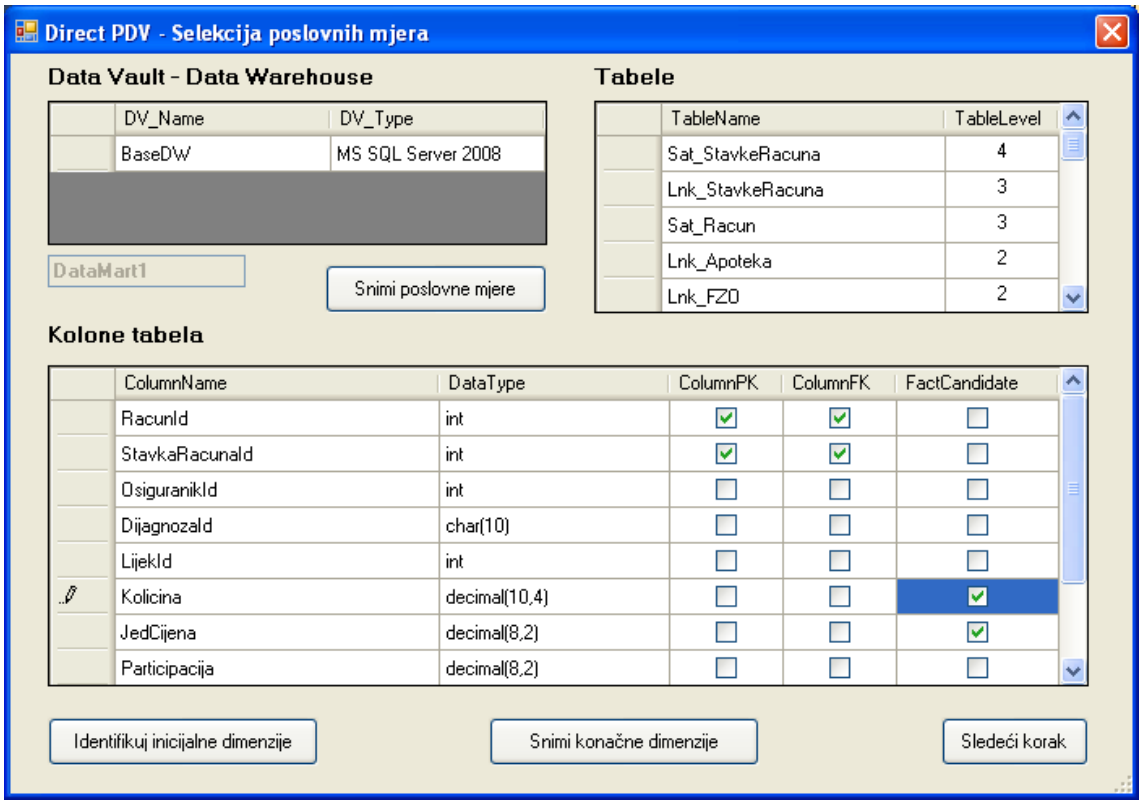
Nakon jednostavnog kreiranja Data Marta i pripadajućih procedura koje su generisane iz modela, prelazi se na drugu međufazu, izbor izvora (skladišta) podataka za Data Mart. Ova forma je prikazana na sljedećoj slici.



Slika 75. Forma za izbor skladišta podataka za data mart

U ovom slučaju izabrano je skladište podataka koje je kreirano u prethodnoj fazi sa podacima iz transakcione baze Recepti. Fizički model skladišta podataka prikazan je na slici 72.

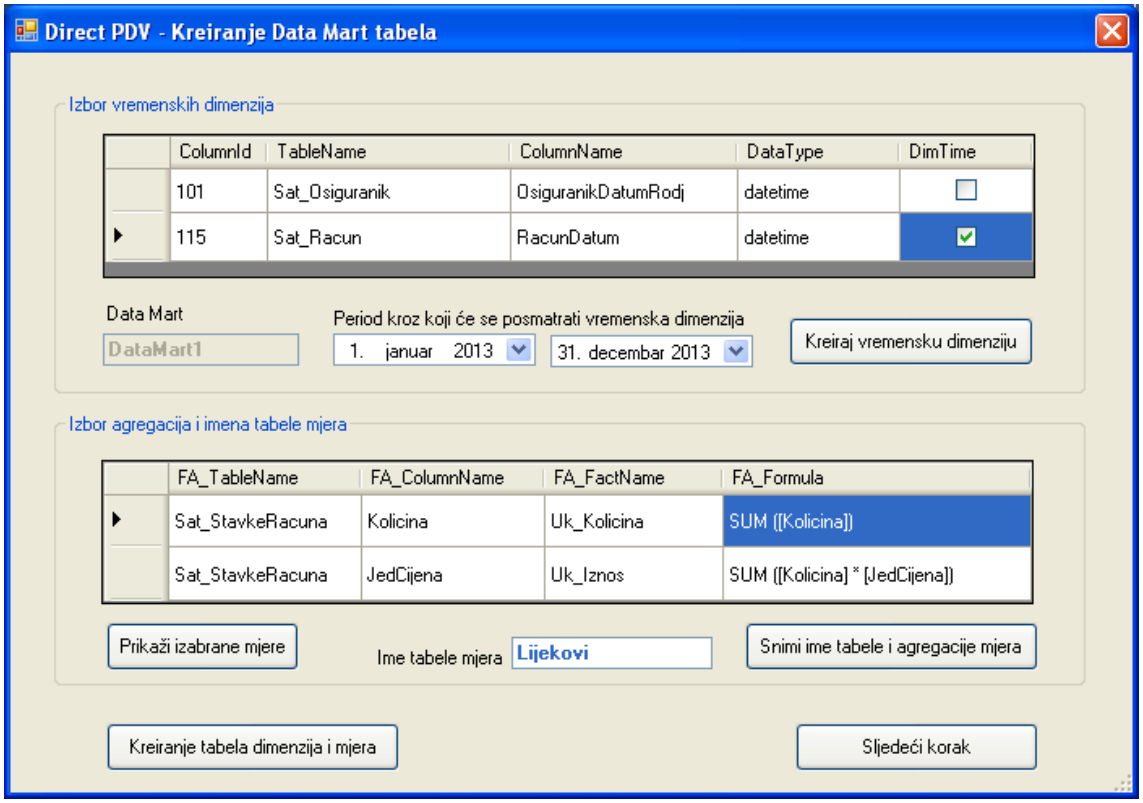
Nakon izbora i učitavanja izvora (skladišta) podataka, korisnik pokreće proces učitavanja metapodataka izabranog izvora podataka klikom na dugme **Učitaj metapodatke.** Klikom na dugme **Sljedeći,** otvara se forma za sljedeću međufazu. Na ovoj formi korisnik istovremeno na tri DataGridView komponente ima mogućnost pregleda izabranog skladišta podataka, tabele skladišta podataka i kolona pojedinih tabela. Na trećem DataGridView dizajner skladišta podataka vrši izbor poslovnih mjera čekiranjem odgovarajućih kolona tabele. Ova forma je prikazana na sljedećoj slici.



Slika 76. Forma za izbor poslovnih mjera za data mart

Na osnovu poslovnih mjera i odgovarajućih pravila sistem identifikuje tabele mjera (fact), a na osnovu tabela mjera i pravila za tabele dimenzija,sistem identifikuje inicijalne dimenzije. Na ovoj formi korisnik ima mogućnost da

proširi listu dimenzija i da ih snimi. Nakon klika na dugme sljedeći, pokreće se sljedeći korak i otvara se forma za kreiranje Data Mart tabela koja je prikazana na sljedećoj slici.

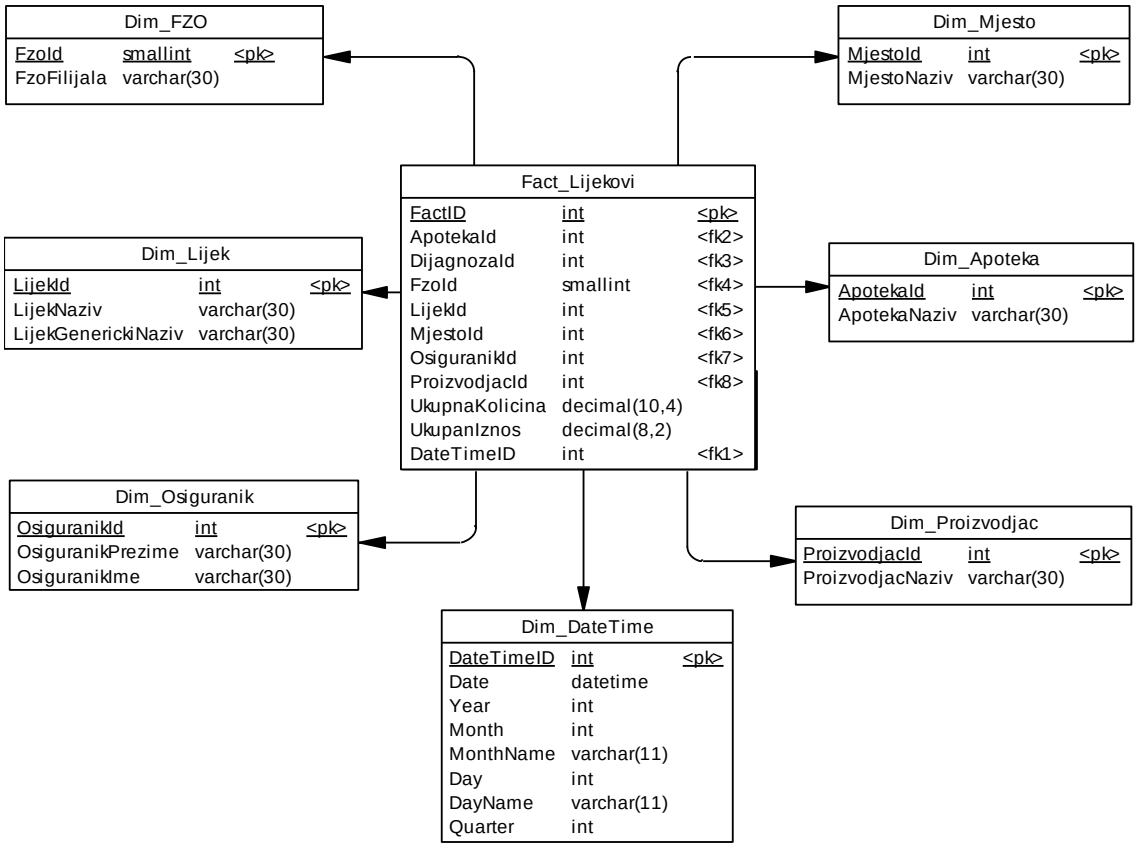


Slika 77. Forma za kreiranje data mart tabela

Klikom na komandno dugme **Kreiraj vremensku dimenziju** pokreću se sistemske operacije kojom se označavaju izabrane vremenske dimenzije i kojom se na osnovu izabrane vremenske dimenzije i upisanog opsega posmatranja vremenske dimenzije kreira i popunjava tabela vremenskih dimenzija. Klikom na komandno dugme **Prikaži izabrane mjere** pokreće se sistemska operacija kojom se na mreži forme prikazuju izabrane mjere sa pripadajućim kolonama i tabelama. U ovom gridu korisnik treba da upiše naziv mjere i način agregacije mjera. Klikom na komandno dugme **Snimi ime tabele i agregacije mjera** pokreće se sistemska operacija u okviru koje se upisuju ime tabele mjera i formula za način agregacije mjera. Klikom na komandno dugme

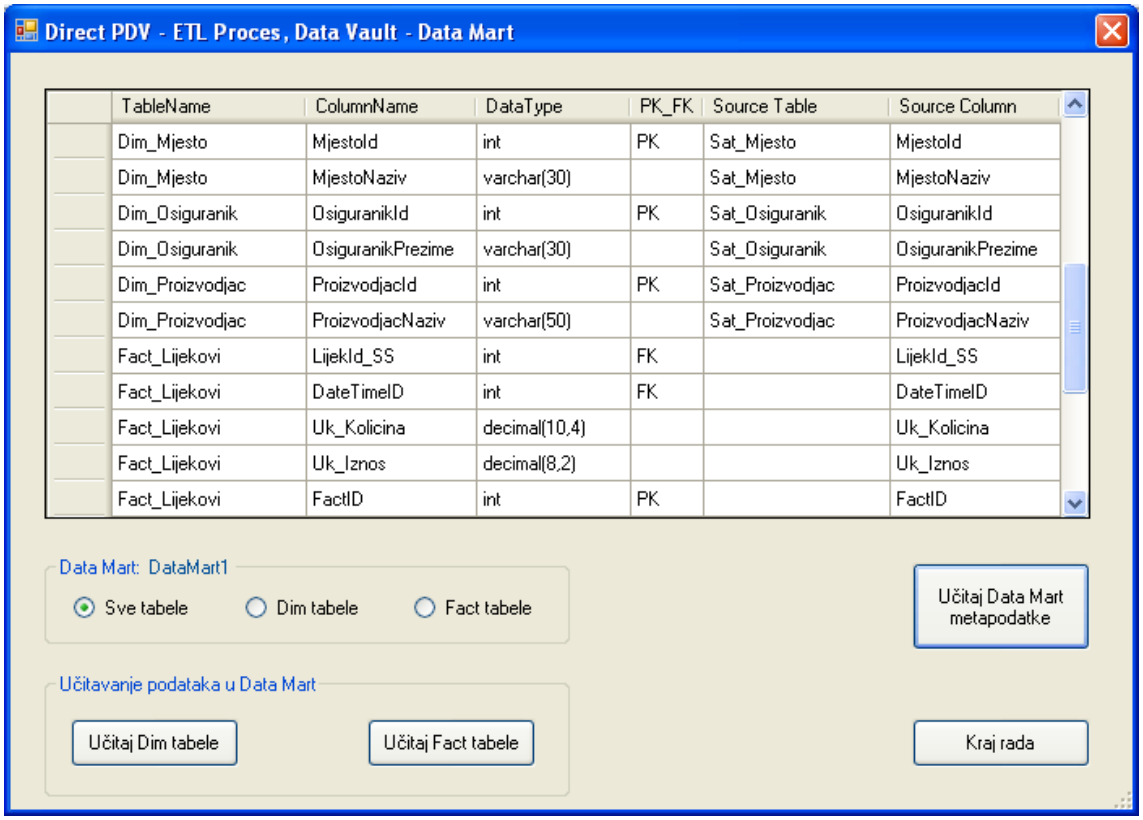
Kreiranje tabele dimenzija i mjera pokreće se sistemska operacija kojom se kreiraju dimenzione i tabele mjera. Klikom na komandno dugme **Sljedeći** otvara se forma za sljedeći korak u procesu kreiranja skladišta podataka.

U toku kreiranja Data Marta korisnik može u svakom momentu da vidi fizički model izabranog skladišta podataka klikom na komandno dugme **Fizički dijagrami** na glavnoj formi. Klikom na ovo dugme korisniku je omogućeno da vidi fizički model kreiranog Data Marta. Fizički model kreiranog Data Marta prikazan je na sljedećoj slici.



Slika 78. Fizički model data marta Lijekovi

Nakon kreiranja Data Marta, stvoreni su uslovi za ETL proces, odnosno ekstrakcija podataka iz skladišta podataka i učitavanje podataka u Data Mart.



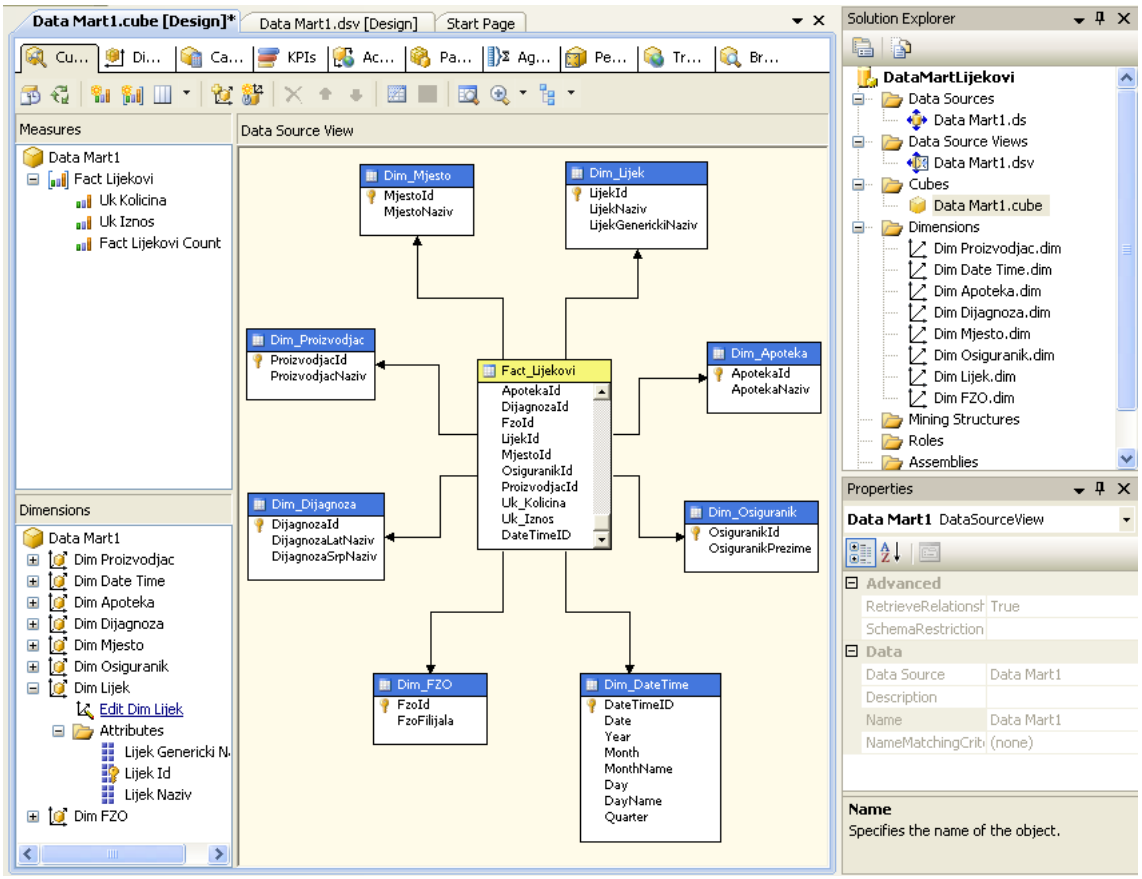
Slika 79. Forma za učitavanje podataka iz skladišta podataka u data mart

Na ovoj formi dizajner skladišta podataka ima mogućnost učitavanja metapodataka skladišta podataka, pregled fizičkog modela skladišta podataka i Data Marta, pregled mapiranja pojedinih tabela i kolona skladišta podataka i Data Marta, te učitavanje podataka u tabele dimenzija i mjera (navedenim redoslijedom).

6.4. Korišćenje podataka u sistemu poslovne inteligencije

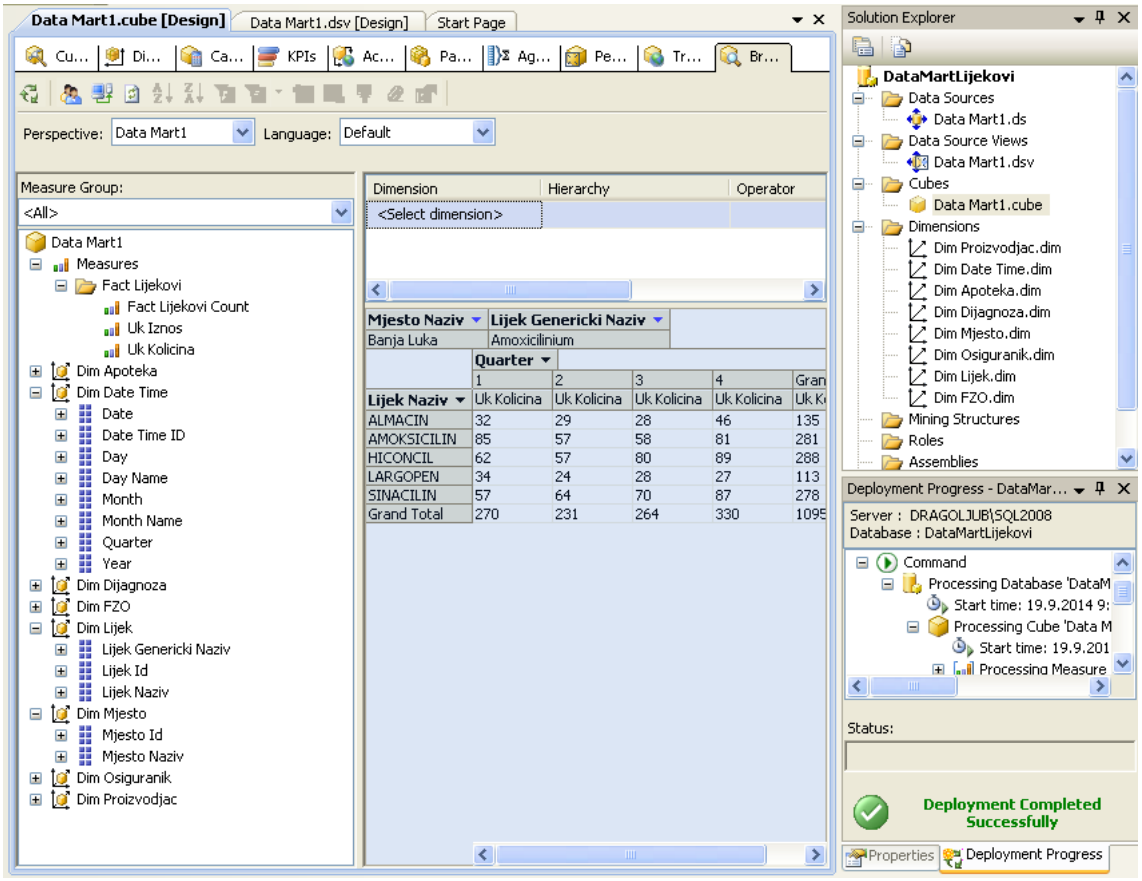
U nastavku rada je ilustrovano korišćenje podataka iz skladišta podataka u sistemu poslovne inteligencije, a radi podrške donošenju poslovnih odluka u zdravstvenom osiguranju. Postupak kreiranja kocke poslovne inteligencije prikazan je u alatu Microsoft SQL Server Business Intelligence Development Studio 2008, izborom opcije Analysis Services Project. Nakon izbor izvora

podataka, određivanja pogleda prema podacima (*Data Source View*) slijedi izbor tabele činjenica i tabela dimenzija, te kreiranje kocke. Na sljedećoj slici je prikazan prozor Solution Explorer nakon kreiranja kocke.



Slika 80. Prozor Solution Explorer nakon izrade kocke

Izborom *Browse* opcije moguće je tehnikom *drag and drop* izabrati dimenzije koje ćemo posmatrati kao redove, a koje kao kolone, te izabrati mere koje ćemo posmatrati što pokazuje sljedeća slika.

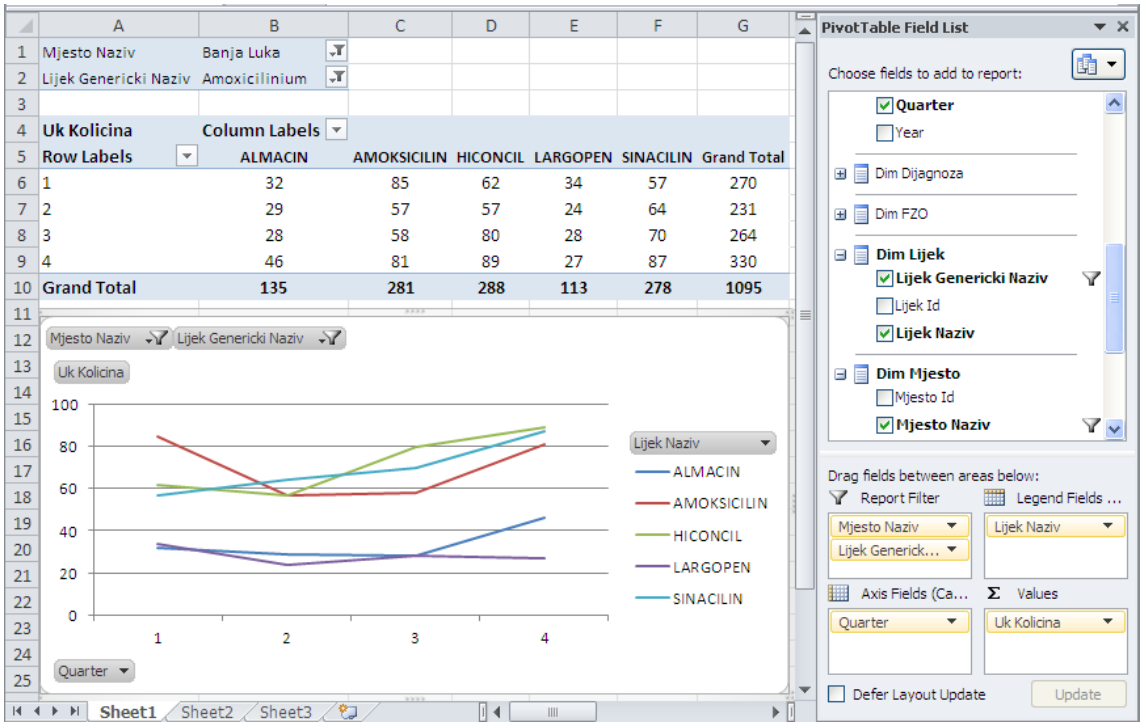


Slika 81. Prozor Browse nakon izbora mjera i dimenzija

Navedeni primjer pokazuju širok izbor načina prikazivanja mjera i dimenzija u Microsoft SQL Server Business Intelligence Development Studio 2008, što je korisna opcija za dizajnera, administratora ili programera baze podataka.

Za krajnjeg korisnika je, naravno, prihvatljivo da se podaci iz kreirane kocke čitaju sa nekim od alata namijenjenih za analizu OLAP kocke. U narednom primjeru je prikazan način povezivanja OLAP kocke iz Microsoft SQL Servera 2008. sa programom Microsoft Office Excel 2010.

Nakon konekcije na OLAP kocku, izbora obrasca izvještaja, te izbora mjera i dimenzija, dobija se radni list Excel-a sa prozorom Pivot Table Field List kao na sljedećoj slici.



Slika 82. Radni list Excel-a sa prozorom *Pivot Table Field List*

U primjeru na prethodnoj slici prikazana je analiza trenda potrošnje lijekova po kvartalima u određenom mjestu za određeni naziv, filtrirano po generičkom nazivu. Za dobijanje ovog izveštaja potrebne su dimenzije: LijekNaziv, LijekGeneričkiNaziv, Mjesto, vremenska dimenzija Kvartal i mjera Ukupna količina. Ovakav izvještaj pruža sljedeće mogućnosti:

- predviđanje potrošnje lijekova po kvartalima za narednu godinu na osnovu pregleda potrošnje po kvartalima u prethodnoj godini,
- promjene referentnog lijeka na pozitivnoj listi lijekova koji se izdaju na teret Fonda zdravstvenog osiguranja, a na osnovu troškova lijekova po pojedinom proizvođaču,
- planiranje ugovora sa apotekama i zdravstvenim ustanovama u narednom periodu na izabranom području,

- mogućnost preduzimanja preventivne zdravstvene zaštite na određenom području na osnovu potrošnje lijekova (uvrštavanjem dimenzije Dijagnoza u izvještaj)
- brže i kvalitetnije izvještavanje prema Institutu za zaštitu zdravlja i Svjetskoj zdravstvenoj organizaciji

OLAP baze podataka dizajnirane su tako da ubrzaju preuzimanje podataka. Budući da OLAP server, a ne program Microsoft Office Excel izračunava rezimirane vrednosti, kada se kreira ili mijenja izvještaj potrebno je poslati manje podataka u Excel. Ovakav pristup omogućuje da korisnik radi sa mnogo većom količinom izvornih podataka nego u slučaju da su podaci organizovani u tradicionalnu bazu podataka, odakle bi Excel preuzimao sve pojedinačne izveštaje, a zatim računao rezimirane vrijednosti [Excel]. Korišćenjem automatizacije skladišta podataka znatno se ubrzava projektovanje skladišta podataka, a time i kreiranje OLAP kocki sa podacima neophodnim za donošenje pravovremenih poslovnih odluka.

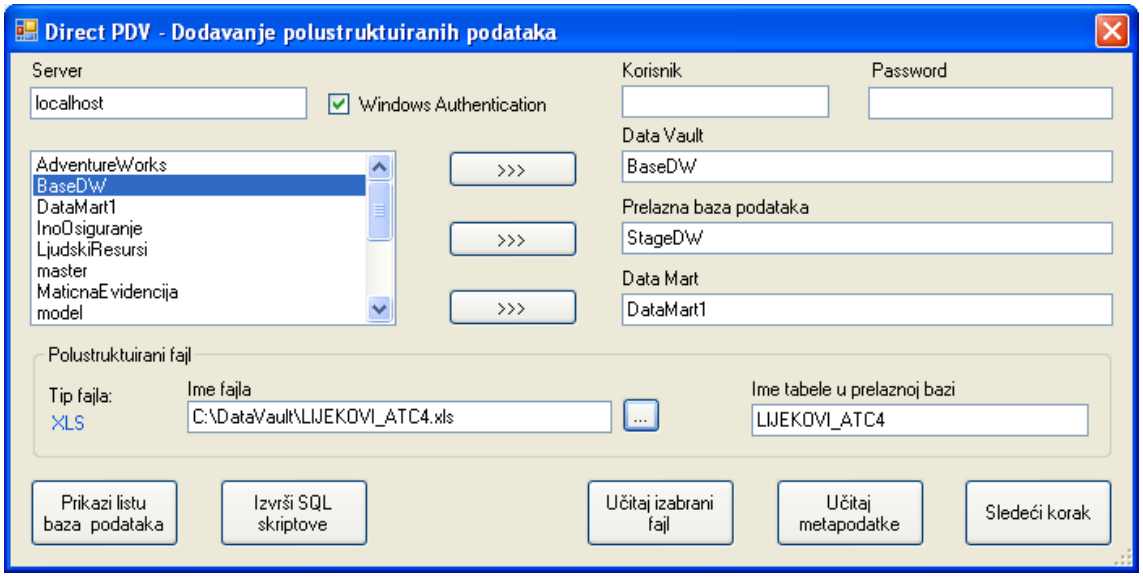
6.5. Proširenje skladišta podataka sa podacima iz eksternih polustruktuiranih izvora podataka

U praksi se često može pojaviti slučaj da su za određene poslovne analize potrebni podaci koje se ne nalaze u transakcionoj bazi podataka, niti se nalaze u organizaciji, niti su ti podaci struktuirani. U ovom slučaju razmotrićemo korišćenje eksternih podataka i polustruktuiranog izvora podataka koji se nalazi u xls formatu. Fajl LIJEKOVI_ATC4.XLS je prikazan na sljedećoj slici.

	A	B	C	D
1	**LijekId**	**LijekNaziv**	**LijekGenerickiNaziv**	**FarmakoHemijskaGrupa**
2	5	LARGOPEN	Amoksicilin	Pencilini širokog spektra
3	6	AMOXIBOS	Amoksicilin	Pencilini širokog spektra
4	156	OFTALMOL	prednizolon	Kortikosteroidi,monokomponentni
5	167	KAMIREN	doksazosin	Blokatori alfa-adrenergičkih receptora

Slika 83. Dio eksternog izvora podataka u XLS formatu

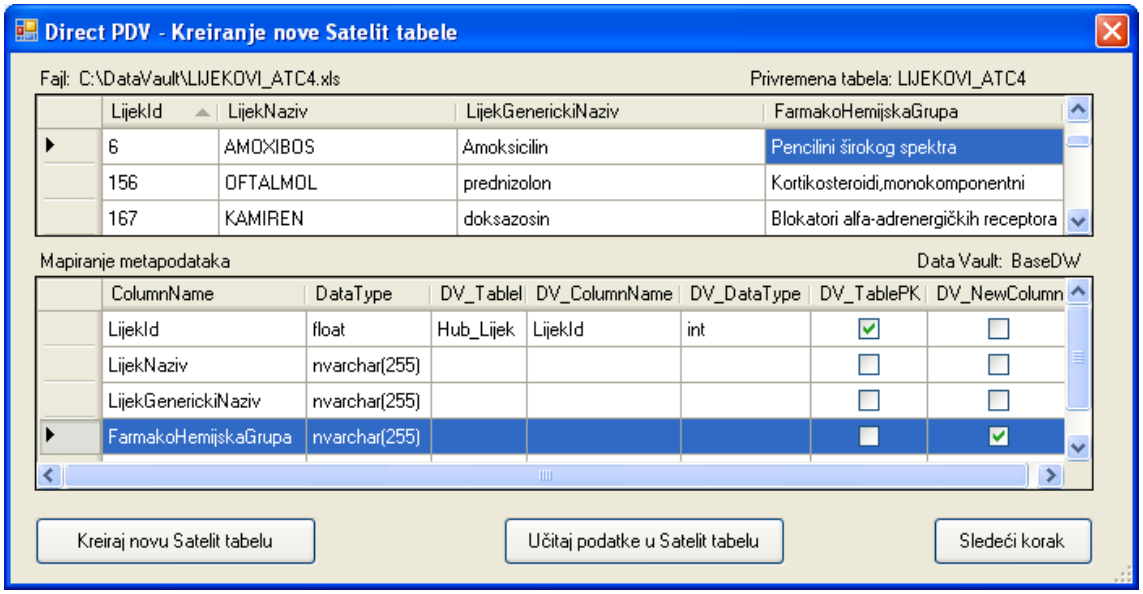
Fajl prikazuje, pored podataka koji se nalaze u transakcionoj bazi podataka (LijeId, LijekNaziv i LijekGenerickiNaziv), i podatke koji se ne nalaze transakcionoj bazi. U ovom slučaju to je podatak o farmako-hemijskoj grupi lijeka (kolona D). Prema [WHO], farmako-hemijska grupa lijeka predstavlja četvrti nivo *Anatomical Therapeutic Chemical (ATC) Classification System* koji je propisala svjetska zdravstvena organizacija (kolona C – generički naziv lijeka predstavlja peti nivo ATC-a). Ovaj podatak može da se preuzme sa web stranice svjetske zdravstvene organizacije ili Agencije za lijekove za lijekove koja odobrava korišćenje određenih lijekova na nacionalnom nivou. Podatak o farmako-hemijskoj grupi potrebno je učitati u skladište podataka (u novu satelit tabelu i povezati sa hub tabelom lijekovi), zatim u data mart (u novu kolonu tabele dimenzija) i na kraju prikazati u kocki OLAP alata radi analize podataka za podršku odlučivanju. U ovom slučaju kandidat sa primarni ključ LijekId se poklapa sa primarnim ključem tabele Hub_Lijek jer Fond zdravstveno osiguranja koristi isti šifarnik kao Agencija za lijekove. Ukoliko nije takav slučaj potrebno je LijekId prilagoditi sadržaju primarnog ključa tabele Hub_Lijek. Na sljedećoj slici prikazana je forma za učitavanje polustruktuiranog izvora u skladište podataka.



Slika 84. Forma za učitavanje polustruktuiranog izvora u Data Vault

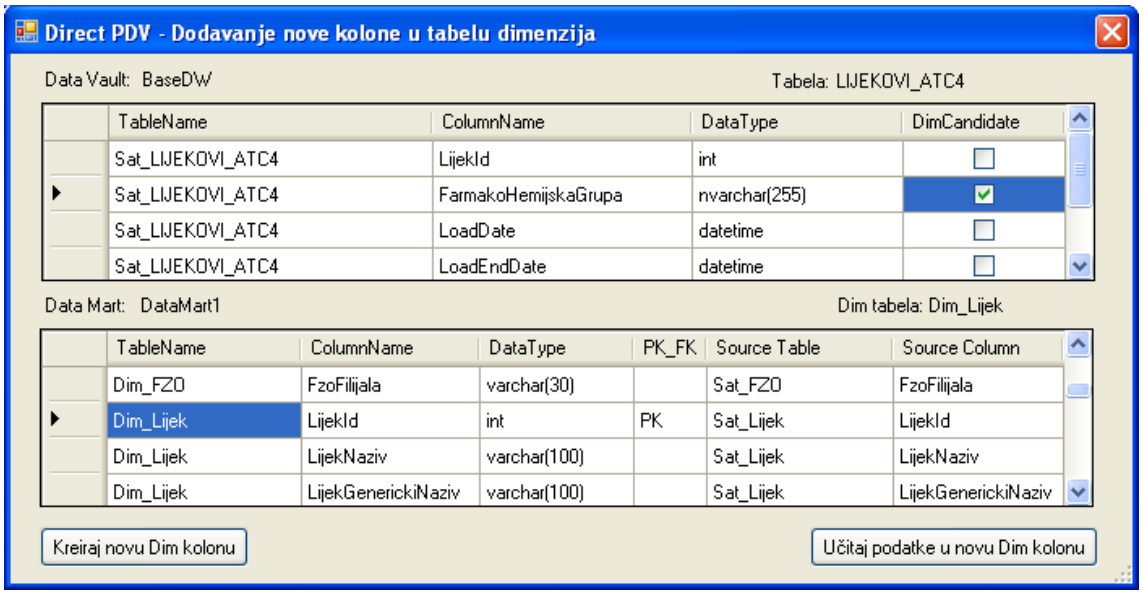
Na ovoj formi, korisnik iz ponuđenih baza podataka bira skladište podataka i data mart u koji će se učitati polustruktuirani fajl. Fajl se prethodno učitava u prelaznu bazu podataka radi eventualnog dodatnog struktuiranja. U prelaznoj bazi kreira se materijalizovani pogled (engl. *view*) SS_Data. Na ovoj formi je omogućeno i učitavanje metapodataka učitanog izvora podataka.

Na formi na sljedećoj slici omogućen je pregled učitanog polustruktuiranog fajla i mapiranje metapodataka fajla sa metapodacima postojećih Data Vault tabela, kreiranje nove satelit tabele, te učitavanje podataka u novu Satelit tabelu.



Slika 85. Forma za kreiranje nove satelit tabele

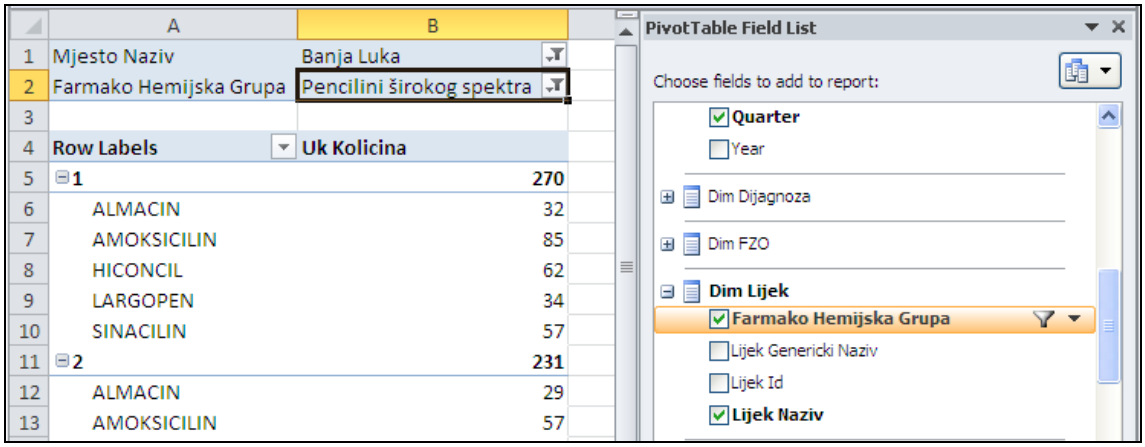
Na formi na sljedećoj slici omogućen je pregled metapodataka nove Data Vault tabele kreirane na osnovu polustruktuiranog fajla i mapiranje metapodataka navedene tabele sa metapodacima postojećih Data Mart tabela. Nakon toga kreira se nova kolona izabrane dimenzione tabele.



Slika 86. Forma za kreiranje nove kolone izabrane dimenzione tabele

Osim toga, ova forma posjeduje i funkcionalnost učitavanja podataka u novu kolonu izabrane dimenzione tabele iz odgovarajuće satelit tabele. U ovom slučaju to je kolona koja označava farmako-hemijsku grupu lijeka.

Nakon jednostavnog dodavanja nove kolone dimenzione tabele u OLAP kocku kroz Analysis Services, dobijamo mogućnost analize podataka po farmako-hemijskoj grupi lijeka. Na sljedećoj slici prikazan je radni list Microsoft Excela sa dodanim podacima o farmako-hemijskoj grupi lijeka.



Slika 87. Radni list Excel-a sa naknadno dodanom kolonom dimenzione tabele iz polustruktuiranog izvora podataka

6.6. Zaključak vezan za praktičnu provjeru rezultata istraživanja

Praktičnom provjerom rezultata istraživanja pokazane su sve prednosti automatizacije projektovanja skladišta podataka. Pored automatizacije projektovanja samog skladišta podataka, pristup omogućuje projektovanje data martova i projektovanje ETL procesa (iz izvora u skladište podataka, te iz skladišta podataka u data martove). Na ovaj način dobijamo integraciju svih potrebnih aktivnosti koje su potrebne za realizaciju kompletnog skladišta podataka. Osim toga, praktično je pokazano i proširenje Data Vault pristupa iz

izvora podataka koji nisu baze podataka, odnosno iz polustruktuiranih izvora podataka.

Alatom Direct PDV moguće je na veoma brz i lak način kreirati kompletno skladište podataka bazirano na Data Vault konceptu. Osim toga, alata se može koristiti i za razvoj prototipova skladišta podataka gdje korisnik može da brzo prepozna radno okruženje za analizu informacija i samim tim postaje više zainteresovan za razvoj skladišta podataka. Projektanti skladišta podataka na taj način mogu lakše otkrivali potrebe korisnika za informacijama.

7. ZAKLJUČNA RAZMATRANJA

Predmet istraživanja doktorske disertacije u širem smislu obuhvatao je sljedeće oblasti: baze podataka, skladišta podataka, poslovnu inteligenciju i podršku odlučivanju. U užem smislu, predmet istraživanja disertacije je proces dizajniranja, odnosno izrade fizičkog modela skladišta podataka. Uočen je problem što proces dizajniranja skladišta podataka i izrade dimenzionog modela nije u potpunosti automatizovan, bilo da se radi o Inmon-ovoj ili Kimbal-ovoj metodologiji ili se radi o primjenama po Data Vault konceptu. Problem istraživanja se može eksplicitno okarakterisati kao metod automatizacije fizičkog projektovanja skladišta podataka i integracije podataka, uz potrebno očuvanje istorije promjena. Osim toga, ovo istraživanje otvorilo je perspektive za istraživanje automatizacije ETL procesa u dijelu prikupljanja, obrade i učitavanja internih podataka iz transakcione baze, podršci fizičkog projektovanja data martova iz Data Vault skladišta podataka, i na kraju ETL procesu u dijelu prikupljanja podataka iz skladišta podataka kao i njihovoj distribuciji i materijalizovanim agregacijama za data martove.

U radu je predstavljen osnovni algoritam za fizički dizajn Data Vault-a zasnovan na modelu metapodataka i modelu pravila za projektovanje na osnovu izvora transakcionih podataka. Odnosi između entiteta u transakcionom sistemu i pravila za razvoj skladišta podataka baziranog na Data Vault konceptu su ključni za automatizaciju projektovanja fizičkog modela skladišta podataka. Konceptualizacija metapodataka predstavljena je fizičkim modelom podataka koji se uz manje izmjene može koristiti u projektovanju individualnih skladišta podataka i predstavlja osnovni metodološki doprinos rada.

Zahvaljujući korišćenju metapodataka za bilo koji dizajn pristup i njegovu automatizaciju, najmanje sljedeće četiri prednosti su očigledne: performanse, skalabilnost, fleksibilnost i agilnost. Kako bi se u potpunosti shvatio potencijal PDV pristupa, potrebno je prvo shvatiti ono što Data Vault kao takav (u odnosu na tradicionalne alternative, normaliziovano skladište podataka i dimenzionalni data mart) sam po sebi obezbeđuje. Razdvajanje Data Vault identiteta i linkova od atributa (satelita) po dizajnu stvara mrežu i na taj način znatno smanjuje stres pojedinačnih ekspanzije, što ide u prilog skalabilnosti. Strukturne promjene su takođe olakšane (bez brisanja), tako da je fleksibilnost dizajna osigurana. Data Vault po dizajnu podstiče veći nivo performansi (razdvajanje zavisnosti) i omogućava paralelno učitavanje podataka sve vrijeme iz svih izvora. Zahtijevajući ulaz podataka bez nepovratne promjene podataka (ELT za razliku od ETL) iz izvora podataka u bazu sirovih podataka kao stalni sistem evidencije (engl. *permanent fully auditable system of records*), Linstedt ukazuju na tipične brzine učitavanja od 100K redova po minuti kao mjerilo.

7.1. Provjera tačnosti postavljenih hipoteza

Na početku rada je postavljena sledeća **generalna hipoteza** koja je obuhvatala preliminarno i teoretsko određenje predmeta istraživanja:

Hipoteza 1: Proces projektovanja skladišta podataka baziranog na Data Vault konceptu i izrade dimenzionog modela skladišta podataka može se formalizovati, generalizovati i u određenoj mjeri automatizovati na osnovu fizičkog modela struktuiranih, uključujući i transakcione baze podataka, polustruktuiranih i jednostavnijih nestruktuiranih izvora podataka.

U radu je predstavljen algoritam za direktni pristup fizičkog projektovanja skladišta podataka koji formalizuje, generalizuje i služi kao podloga za realizaciju alata kojim je u velikoj mjeri moguća automatizacija procesa fizičkog projektovanja skladišta podataka na osnovu fizičkog modela struktuiranih, polustruktuiranih i jednostavnijih nestruktuiranih izvora podataka. Predstavljeni direktni pristup integriše sve neophodne elemente za fizičko projektovanje skladišta podataka. Na osnovu algoritma urađen je prototip aplikacije za podršku fizičkom projektovanju skladišta podataka. Eksperimentalno korišćenje algoritma, uz podršku prototipa aplikacije, je provedena na stvarnom slučaju u oblasti zdravstvenog osiguranja i daje dobre rezultate.

Iz generalne hipoteze izvedene su sledeće **posebne hipoteze** koja obrađuju dijelove predmeta istraživanja:

Hipoteza 2: Formalizacijom, standardizacijom i automatizacijom projektovanja skladišta podataka, podiže se kvalitet DW sistema, te smanjuju vrijeme i troškovi implementacije skladišta podataka.

S obzirom da procedure za kreiranje tabela skladišta podataka (hub, link i satelit) i tabele data marta (dimenzione i fact), te procedure za učitavanje podataka u navedene tabele kreirane primjenom dinamičkog SQL-a (uz podršku prototipa razvijenog alata), sadrže nekoliko stotina linija koda, očigledno je smanjenje vremena implementacije skladišta podataka. U slučaju ručnog kreiranja tabela, pa i u slučaju izrade fizičkog modela odgovarajućim CASE alatom i generisanja navedenih tabela, vrijeme bi bilo znatno duže. Isti slučaj imamo i kod ručnog pisanja ETL koda.

Automatsko generisanje koda za kreiranje tabela i koda za ETL procese, smanjuje se broj potrebnih IT stručnjaka, a time se smanjuju i troškovi implementacije skladišta podataka.

Saglasno poglavlju 2.9., ovdje razmatramo lakoću održavanja skladišta podataka kao karakteristiku kvaliteta prema standardu ISO-9126. Lakoća održavanja obuhvata analizu podkarakteristika softvera: mogućnost analize, stabilnost, izmjenljivost i mogućnost testiranja. U procesu projektovanja skladišta podataka uz podršku predstavljenog prototipa alata postoji *mogućnost analize* fizičkog modela izvora podataka, skladišta podataka i data marta kroz vizualizaciju metapodataka u obliku fizičkog dijagrama bilo u toku projektovanja ili u procesu održavanja skladišta podataka. *Izmjenljivost* je osobina kojom se utvrđuje pogodnost softverskog proizvoda za ispravkama, doradama i izmjenama. Prototip alata predstavljen u ovom radu omogućuje veoma jednostavnu *izmjenu* strukture skladišta podataka dodavanjem novih tabela ili izmjenama postojećih tabela što je pokazano u poglavljima 5 i 6 (primjer sa naknadnim dodavanjem Excel izvora podataka u skladište podataka). *Stabilnost* je osobina softvera kojom se mjere efekti i rizici od izmjena softverskog proizvoda. U poglavljima 5 i 6 pokazan je pozitivan efekat kreiranja nove tabele skladišta podataka i učitavanje podataka iz polustruktuiranom izvora sa minimalnim rizikom za konzistentnost skladišta podataka. M*ogućnost testiranja* osobina kojom se vrši ocjena izmjenjenog softverskog proizvoda. Iako ovakvu ocjenu mogu najbolje dati korisnici softvera, u primjeru u poglavlju 6. kod izmjene strukture skladišta podataka (dodavanje nove satelit tabele) interna testiranja provedena na navedenoj izmjeni pokazuju da izmjene u strukturi i procesima ekstrakcije i učitavanja ne utiču na funkcionisanje skladišta podataka. Prethodnim razmatranjima je dokazan dio hipoteze da se

formalizacijom i standardizacijom (posljedica čega je prototip alata za automatizaciju projektovanja skladišta podataka) podiže kvalitet DW sistema.

S druge strane, automatskim generisanjem tabela skladišta podataka, sa ugrađenim pravilima za nazive tabela i tipove podataka saglasno izvorima podataka dobija se konzistentnost u nazivima tabela, atributa, indeksa i tipova podataka. Sličnu situaciju imamo i kod generisanja ETL koda. Ovim se dodatno smanjuje mogućnost grešaka prilikom kreiranja skladišta podataka i ETL procesa i dodatno povećava kvalitet skladišta podataka.

Osim toga, kvalitet projektovanja DW sistema utiče na podizanje kvaliteta skladišta podataka. U poglavlju 2.9 (Tabela 4) pokazano je unapređenje kvaliteta projektovanja skladišta podataka korišćenjem GQM paradigme. Saglasno prethodnim razmatranjima, evidentno je da sve metrike iz primjera u Tabeli 4. mogu biti manje u slučaju korišćenja predloženog pristupa automatizacije skladišta podataka. Korišćenjem GQM paradigme mogu se odrediti osnovni ciljevi poboljšanja kvaliteta projektovanja, razvoja ili održavanja skladišta podataka, pitanja koja se postavljaju za ispunjenje navedenih ciljeva i metrika kojom se mjeri ispunjenost ciljeva.

Hipoteza 3: Automatizacija projektovanja skladišta podataka doprinosi automatizaciji ETL procesa koji se odnosi na interne podatke iz transakcione baze.

Bez obzira u koliko izvora se nalaze operativni podaci, svaki poslovni sistem uglavnom teži da ima jednu glavnu bazu podataka u kojoj je smješten najveći dio podataka. Najčešće je to interna transakciona baza podataka. Metapodaci te baze podataka služe kao osnova za projektovanje skladišta podataka. Prilikom

projektovanja fizičkog modela skladišta podataka, koje je bazirano na metapodacima izvora podataka, na određeni način je izvršeno mapiranje metapodataka izvora podataka i budućeg skladišta podataka. Ovo mapiranje se može iskoristiti u procesima ekstrakcije i učitavanja podataka u skladište podataka, tako da se prilikom generisanja tabela skladišta podataka može izvršiti i generisanje ETL procedura. U tom slučaju već u ranoj fazi kreiranja skladišta podataka možemo kreirati procedure za ETL procese. Na ovaj način automatizacija izrade fizičkog modela skladišta podataka doprinosi automatizaciji ETL procesa interne transakcione baze podataka. Sličnu situaciju imamo i kod automatizacije projektovanja data martova i odgovarajućih ETL procesa.

7.2. Ostvareni naučni doprinosi

Kroz istraživanja, rad i rezultate doktorske disertacije ostvareni su sljedeći naučni doprinosi:

1. Pregled trenutnog stanja u oblasti automatizacije skladišta podataka
2. Formalan opis modela i metoda koji omogućuju što veću automatizaciju izrade fizičkog modela skladišta podataka baziranog na Data Vault konceptu
3. Formalizacija proširenja Data Valuta u cilju principijelnog obuhvata izvora podataka iz izvora koji nisu samo relacione baze podataka
4. Unapređenju procesa dizajniranja skladišta podataka
5. Cjelokupan integrisan pristup automatizaciji projektovanja skladišta podataka
6. Podizanje kvaliteta projektovanja DW sistema, smanjenje troškova i vremena implementacije DW sistema

Kroz istraživanja, rad i rezultate doktorske disertacije ostvareni su i sljedeći stručni doprinosi:

1. Demonstrirana je evaluacija pristupa izradom prototipa aplikacije
2. Demonstriran je pristup u praksi kroz korišćenje prototipa aplikacije na realnom primjeru iz oblasti zdravstva i zdravstvenog osiguranja

Pregled istraživanja u oblasti automatizacije skladišta podataka (poglavlje 3), pored pregleda teoretskih istraživanja, daje se pregled realizovanih alata u oblasti automatizacije skladišta podataka. Osim toga, u poglavlju 2 dati su osnovni aspekti skladišta podataka vezani za multidimenzioni dizajn, arhitekture, metodologije razvoja, te odnos poslovne inteligencije i skladišta podataka. Ovaj dio može da se koristi kao sažet uvod u oblast skladištenja podataka.

Formalan opis modela i metoda koji omogućuju što veću automatizaciju izrade fizičkog modela skladišta podataka baziranog na Data Vault konceptu predstavlja centralni dio rada koji je dat u poglavlju 4 kao direktni pristup projektovanja skladišta podataka. Pristup je definisan i obrazložen odgovarajućim algoritmom. Najprije je određen problem (ručno projektovanje skladišta podataka), a zatim je prezentovano idejno rješenje kroz dijagrame, algoritme, modele i odgovarajuća pravila kako bi se automatizovao proces projektovanja skladišta podataka.

Formalizacija proširenja Data Valuta u cilju principijelnog obuhvata izvora podataka iz izvora koji nisu samo relacione baze podataka podrazumijevala je uključivanje polustruktuiranih izvora podataka. Formalizaciju olakšava činjenica da, bez obzira u koliko izvora se nalaze operativni podaci, svaki poslovni sistem teži da integriše, barem konceptulno, osnovne podatke od šireg

značaja i uspostavi kontrolu, odnosno da ima jednu primarnu bazu podataka u kojoj je smješten najveći dio podataka (a ta baza je u najvećem broju slučajeva relaciona). Polustruktuirani podaci se učitavaju u prelaznu bazu podataka, a zatim u skladište podataka. Nestruktuirani podaci moraju proći procese tekstualne analize, a zatim nakon struktuiranja prebacuju u prelaznu bazu podataka. Daljnji postupak je sličan kao kod polustruktuiranih podataka.

Unapređenju procesa dizajniranja skladišta podataka. Automatizacijom dizajnirajnja skladišta podataka znatno se ubrzava razvoj kompletnog sistema koji omogućuje izradu prototipa skladišta podataka u ranoj fazi kontakta sa korisnicima. Na taj način korisnici postaju više zainteresovani za davanje informacija. Projektanti skladišta podataka na taj način mogu lakše otkrivati znanja o poslovnim procesima, te o potrebe korisnika za informacijama. Pristup je moguće primijeniti na različite tipove organizacija i različite sisteme za upravljanje bazama podataka.

Cjelokupan integrisan pristup automatizaciji projektovanja skladišta podataka. Pored automatizacije projektovanja samog skladišta podataka, pristup omogućuje projektovanje data martova i projektovanje ETL procesa (iz izvora u skladište podataka, te iz skladišta podataka u data martove). Na ovaj način dobijamo integraciju svih potrebnih aktivnosti koje su potrebne za realizaciju kompletnog skladišta podataka.

Podizanje kvaliteta projektovanja DW sistema, smanjenje troškova i vremena implementacije DW sistema. Kvalitet projektovanja DW sistema pokazan je korišćenjem GQM paradigme u procesu projektovanja skladišta podataka. Primjenom GQM paradigme mogu se definisati osnovni ciljevi poboljšanja

procesa projektovanja, razvoja ili održavanja skladišta podataka, pitanja koja se postavljaju za ispunjenje tih ciljeva, te predložena metrika.

Demonstrirana je evaluacija pristupa izradom prototipa aplikacije koja je omogućila provjeru postavljenih ciljeva i hipoteza. Ova jednostavna aplikacija, nazvana *Direct PDV*, omogućuje veoma brzo projektovanje skladišta podataka uključujući i projektovanje data martova i ETL procesa.

Demonstriran je pristup u praksi kroz korišćenje prototipa aplikacije na realnom primjeru iz oblasti zdravstvenog osiguranja. Korišćenjem prototipa aplikacije pokazano je veoma brzo i efikasno projektovanje skladišta podataka, data marta i ETL procesa. U ovom dijelu je prikazan realan primjer iz zdravstvenog osiguranja, uz dodatnu analizu OLAP alatom i mogućnosti donošenja poslovnih odluka na osnovu dobijenih informacija.

7.3. Mogućnost primjene rješenja i pravci daljnjih istraživanja

Da bi pristup, koji je prezentovan u disertaciji, bio primjenjen u praksi, potrebno je da se razvije kompletna aplikacija koja će biti prilagođena projektantima skladišta podataka. To podrazumijeva daljnji razvoj i zaokruženje prototipa kao konačne aplikacije ili razvoj potpuno nove aplikacije bazirane na principima, algoritmima i slučajevima korišćenja prezentovanim u radu. Prototip aplikacije razvijen u ovom radu služi za evaluaciju predloženog pristupa direktnog projektovanja skladišta podataka i pokazuje da pristup izvodljiv u praksi. Kod korišćenja prototipa aplikacije veoma je bitno da projektanti skladišta podataka pažljivo izvrše izbor domena za koji se projektuje skladište podataka. Skladište podataka koje je zasnovano na Data Vault konceptu, najviše ima smisla koristiti u slučaju više distribuiranih izvora

podataka. Ključna poboljšanja koja su moguća na prototipu aplikacije odnose se prije svega na realizaciju funkcionalnosti koja se odnose na obuhvat složenijih polustruktuiranih i nestruktuiranih izvora podataka, te realizaciju funkcionalnosti tekstualne analize nestruktuiranih podataka. S obzirom da je prototip aplikacije predviđen samo za rad na Windows operativnom sistemu, postoji prostor za razvoj potpuno nove aplikacije, koja će se oslanjati na principe predstavljene ovim radom, a koja može da radi i na drugim operativnim sistemima.

Konceptualizacija metapodataka predstavljena fizičkim modelom podataka koji se uz manje izmjene može koristiti u projektovanju individualnih skladišta podataka, predstavlja osnovni metodološki doprinos rada. Pristup je moguće primijeniti na različite tipove organizacija i različite sisteme za upravljanje bazama podataka.

8. LITERATURA

ABS00 Abiteboul S., Buneman P., Suciu D., Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufman Publishers, 2000.

AMA05 Adelman S., Moss L., Abai M., Data Strategy, Addison Wesley, New York, 2005

AZMT11 El Akkaoui Z., Zimányi E., Mazón J.N., Trujillo J., A Model-Driven Framework for ETL Process Development, DOLAP'11, October 28, 2011, Glasgow, Scotland, UK.

Bar97 Barry D., Data Warehouse from Architecture to Implementation, Addison-Wesley, 1997.

BCR94 Basili V. R., Caldiera G., Rombach H. D., The Goal Question Metric Approach. Encyclopedia of Software Engineering - 2 Volume Set, pp 528-532, John Wiley & Sons, Inc. (1994). Available at http://www.cs.umd.edu/users/basili/papers.html

BFYV Budinsky F., Finnie M., Yu P., Vlissides J.: Automatic Code Generation From Design Patterns: http://www.research.ibm.com/designpatterns/pubs/codegen.pdf

BIR Data Warehouse Automation, http://biready.com, 2014.

Birst Save time and resources through automation, http://www.birst.com/product/technology/data-warehouse-automation, 2014

BU99 Boehnlein M., Ulbrich-vom Ende A., Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems, 2nd Int. Workshop on Data Warehousing and OLAP, 1999

BU00 Boehnlein, M., Ulbrich v.E, A., Business Process Oriented

Development of Data Warehouse Structures, Proceedings of Data Warehousing 2000, Physica Verlag, 2000

CBD10 Casters M., Bouman R., van Dongen J., Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration, Wiley Publishing, 2010

CH01 A. Cockburn, J. Highsmith, Agile Software Development: The Business Of Innovation, IEEE Computer, Sept. 2001.

Chen76 Chen P., The Entity-Relationship Model – Toward a Unified View of data, ACM Transaction on Database Systems, March 1976.

CMMI http://www.sei.cmu.edu/cmmi/

Cock01 Cocburn Alistair, Writing Effective Use Cases, AddisonWesley, 2001.

Cobit http://www.isaca.org/cobit

Corr11 Corr L., Stagnitto J., Agile Data Warehouse Design, DecisionOne Press, Leeds, 2011

Cod93 Codd E. F., The Relational Model for Database Management, Version 2. Addison-Wesley, 1993.

CPPS01 Calero C., Piattini M., Pascual C., Serrano M.A., Towards Data Warehouse Quality Metrics, Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2001), Interlaken, Switzerland, 2001

CSWD09 Castellanos M., Simitsis A., Wilkinson K., Dayal U., Automating the Loading of Business Process Data Warehouses, EDBT'09, March 24-26, 2009, Saint Petersburg, Russia

ĆBVG09 Ćamilović D., Bečejski-Vujaklija D., Gospić N., A Call Detail Records Data Mart: Data Modelling and OLAP Analysis. Computer Science and Information Systems, Issue: 12, 2009

Dam08 Damhof R., The next generation EDW, Database Magazine (Netherlands), 2008

DD95 Date C.J., Darwen H., The Third Manifesto, ACM SIGMOD Record (New York: ACM Press) 24 (1), March 1995

DDL03 Date C.J., Darwen H., Lorentzos N., Temporal Data and the Relational Model, Morgan Kaufmann, 2003.

DM88 Devlin, B., Murphy, P. (1998). An architecture for a business information system. IBM Systems Journal, 27(1), 60-80.

EB84 Eftimie R.A., Briquez V., Management DataBases Study, Geneva 1984.

Els03 Elssamadisy A., XP on a Large Project- A Developer's View, Extreme Programming Perspectives, eds. Marchesi et al., Pearson 272 Education, Indianapolis, IN. 2003.

EMC http://www.emc.com/products-solutions/index.htm

Eng96 English L., Information Quality Improvement: Principles, Methods and Management, Seminar, 5th Ed., Brentwood, TN: Information Impact International, Inc., 1996.

Excel http://office.microsoft.com/en-us/excel-help/overview-of-online-analytical-processing-olap-HP010177437.aspx

Gar13 Magic Quadrant for Data Warehouse Database Management Systems February 2013; http://www.gartner.com/technology/reprints/

GJ00 Gundelroy M., Jorden J.L., SQL Server 2000, SYBEX Inc., 2000

GMR98 Golfarelli M., Maio D., Rizzi S., Conceptual Design of Data Warehouses from E/R Schemes, System Sciences, 1998

Gol08 Golfarelli M., DEIS – University of Bologna, Data warehouse life-cycle and design, White Paper, 2008

Gov10 Govindarajan S., Data Vault Modeling The Next Generation DW Approach, http://www.globytes.com, 2010

GR09 Golfarelli M., Rizzi S., Data Warehouse Design. Modern Principles and Methodologies. McGraw-Hill, 2009.

GRV01 Golfarelli M., Rizzi S., Vrdoljak B., Data warehouse design from XML sources., Proc. DOLAP'01, Atlanta, 2001.

Gra11 Graziano K., Introduction to Data Vault Modeling, True Bridge Resources, White paper 2011.

GTTY06 Guo Y., Tang S., Tong Y., Yang D., Triple-Driven Data Modeling Methodology in Data Warehousing A Case Study, DOLAP '06 Proceedings of the 9th ACM international workshop on Data warehousing and OLAP, 2006

Harz http://www.harzing.com/pop.htm (Publish or Perish)

HS10 Hammergren T. , Simon A., Data Warehousing for Dummies, 2nd edition. JW, 2009

Hua10 Huang X., Data Warehouse Architecture: Practices and Trends, IGI Global, 2010

Hug08 Huges R., Agile Data Warehousing: Delivering World-Class Business Intelligence Systems Using Scrum and XP, Willey, 2008

IBM http://www-03.ibm.com/software/products/en/category/SWM20

IBM1 Integrate all types of data on distributed and mainframe platforms, http://www-03.ibm.com/software/products/en/ibminfodata, 2014

IBM98 Data Modeling Techniques for Data Warehousing, February 1998. http://www.redbooks.ibm.com,

IDABC European Interoperability Framework for pan-European eGovernment Services, http://ec.europa.eu/idabc/en/chapter/5883.html

IDC11 http://www.idc.com

IK12 Inmon W.H., Krishnan K., Building the Unstructured Data Warehouse, Technic Publications LLC, 2012

Inform Data Integration Agilitz, http://www.informatica.com/us/products/data-integration/enterprise/powercenter/, 2014

Inm00a Inmon W.H., Building the Data Warehouse: Gettiing Started, White Paper, BillInmon.com, 2000

Inm00b Inmon W.H., Creat ing The Data Warehouse Data Model From The Corporate Data Model, 2000

Inm02 Inmon H.W., Building the Data Warehouse, 3Ed, Wiley Computer Publishing, Indianapolis, 2002

Inm92 Inmon H. W., Building the Data Warehouse, Wiley Computer Publishing,1992.

ISN08 Inmon H.W., Strauss D., Neushloss G., DW 2.0 The Arcitecture for the Next Generation od Data Warehousing, Morgan Kaufman, 2008

ISO9126 http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749

ITIL http://www.itil-officialsite.com

JBKP12 Jovanović V., Bojičić I., Knowles C., Pavlić M., Persistent Staging Area Models for Data Warehouses, Issues in Information Systems, Volume 13, Issue 1, pp. 121-132, 2012

JBR99 Jacobson I., Booch G., Rumbaugh J., The Unified Software Development Process, Rational Software Corporation, Addison-Wesley, 1999.

JD08 Jörg, T., Deßloch, S., Towards Generating ETL Processes for Incremental Loading. In Proceedings of the 2008 international

symposium on Database engineering & applications (IDEAS '08)

JHP04 Jensen, M., Holmgren, T., Pedersen T. Discovering Multidimensional Structure in Relational Data. In Proceedings of DaWaK, 2004

JM13 Jovanovic V., Marjanovic Z., DW21 Research Program-Expectations, FON/Breza software engineering, White paper, Febrary 2013

JGC07 Jovanovic, V., Gardiner, A., Cupic, L. (2007) A taxonomy of data modeling techniques, Proceedings of the 2007 Southern Association for Information Systems Conference, 189-194.

JPB12 Jovanović V., Pavlić M., Bojičić I., Conceptual Models for DW Staging Area, MIPRO, 2012

JQJ98 Jeusfeld M.A., Quix C., Jarke M., Design and Analysis of Quality Information for Data Warehouses. Proceedings of the 17th International Conference on Conceptual Modeling (ER'98), Singapore, 1998.

JRSA12 Jovanovic P., Romero O., Simitsis A., Abelló A., Requirement-driven creation and deployment of multidimensional and ETL designs, Advanced Concept. Model, 2012

Kim96 Kimball R., The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, Inc., 1996.

Kim12 http://www.kimballgroup.com/html/about.html, 31.05.2012.

KJ12 Knowles C., Jovanovic V., Extensible Markup Languague (XML) Schemas for Data Vault Models, JCIS, 2012

KJM14 Krneta, D., Jovanović, V., Marjanović, Z., A Direct Approach to Physical Data Vault Design, Computer Science and Information

Systems, Vol. 11, No. 2, 569–599., 2014

Kno12 Knowles C., 6NF Conceptual Models and Data Warehousing 2.0, *SAIS 2012*

KOBB08 Krneta D., Opačić V., Bobanac Z., Banović N., Sistem poslovne inteligencije u Fondu zdravstvenog osiguranja, Infoteh, Jahorina, 2008.

KR12 Krneta D., Automatizacija fizičkog projektovanja skladišta podataka proširenog Data Vault pristupa, Pristupni rad na doktorskim studijama, FON, Beograd 2012

KR02 Kimball R., Ross M., The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd Edition, Wiley, 2002

KR08 Krneta D., Radosav D., Realization Business Intelligence system using MS SQL Server 2008, FIT IT Conference, Mostar 2008

KRR08 Krneta D., Radosav D., Radulović B., Realization Business Intelligence in Commerce using Microsoft Business Intelligence, SISY 2008., IEEE Catalog Number: CFP0884C-CDR, http://bmf.hu/conferences/sisy2008

Lar98 Larman Craig. Applying UML and Patterns: An introduction to object-oriented analysis and design, Second edition, Prentice Hall, New Jersy, 1998.

Lar09 Larson B., Delivering Business Intelligence with MS SQL Server 2008, Mc Graw Hill, 2009

LGH08 Linstedt D., Graziano K., Hultgren H., The Business of Data Vault Modeling, 2008

Lin1 http://danlinstedt.com/about/data-vault-basics

Lin10 Linstedt D., Super Charge your Data Warehouse, Kindle Edition, 2010

Lin12 D. Linstedt, Fundamentals Class Document, Data Vault Training, http://LearnDataVault.com/Training, 2012

Lin2 http://www.tdan.com/view-articles, Data Vault Series 1-5 by Dan E. Linstedt

Lin3 Linstedt D., Data Vault Model & Methodology, http://www.learndatavault.com, 2011

Lin4 http://danlinstedt.com/datavaultcat/from-star-schema-to-data-vault/

LMAB08 Lazarević B., Marjanović Z., Aničić N., Babarogić S., Baze podataka, Fakultet organizacionih nauka, Beograd 2008.

MA02 Moss T. L., Atre S., Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications, Addison Wesley, 2003

MDP07 Milosavljević G., Dejanović I., Perišić B., Brz razvoj adaptivnih poslovnih informacionih sistema, YuInfo, 2007

Micro SQL Server Integration Services, http://technet.microsoft.com/en-us/library/ms141026.aspx, 2014

Micro1 http://msdn.microsoft.com/en-us/library/ms973831.aspx (Designing for Web or Desktop?)

Mil10 Milosavljević G., Prilog metodama brzog razvoja adaptivnih poslovnih informacionih sistema, doktorska disertacija, Novi Sad, 2010

MK00 Moody D.L., Kortink M.R.A., From Enterprise Models to Dimensional Models: A Methodology for DW and Data Mart Design, International Workshop on Design and Management of Data Warehouse, 2000

MLG04 Mogin P., Luković I., Govedarica M., principi projektovanja baza podataka, FTN Izdavaštvo, Novi Sad 2004.

MMT09 Muñoz L., Mazón J.N., Trujillo J, Automatic Generation of ETL processes from Conceptual Models, DOLAP'09, Hong Kong, China, 2009

MP04 Milosavljevic, G., Perišić, B., A Method and a Tool for Rapid Prototyping of Large-Scale Business Information Systems. Computer Science and Information Systems, Vol. 1, No. 2, 2004

Msagl http://research.microsoft.com/en-us/projects/msagl/

MSClo http://www.microsoft.com/en-us/server-cloud/solutions/modern-data-warehouse/

MTK08 Mundy J., Thornthwaite W., Kimball R., The Microsoft Data Warehouse Toolkit Second Edition, Wiley, 2008

MZ08 Malinowski E., Zimanyi E., Advanced Data Warehouse Design, Springer, Berlin, 2008

NNH10 Nazri M.N.M., Noah S.A., Hamid Z., Using lexical ontology for semi-automatic logical data warehouse design, 5th international conference on Rough set and knowledge technology, 2010

NSZ08 Nicola M., Sommerlandt M., Zeidenstein K., From text analytics to data warehousing, http://www.ibm.com/developerworks/data/library/techarticle/dm-0804nicola/, 2008

Obr06 O'Brien J. A., Marakas G. M., Management Information Systems, McGraw-Hill, New York, 2006

OMG www.omg.org/mda

Ora1 http://www.oracle.com/us/products/database/datawarehousing/overview/index.html

Oracle Oracle Data Integrator Enterprise Edition 12c, http://www.oracle.com/us/products/middleware/data-

integration/enterprise-edition/overview/index.html, 2014

PD02 Phipps C., Davis K. C., Automating Data Warehouse Conceptual Schema Design and Evaluation. 4th International Workshop on Design and Management of Data Warehouses, 2002.

Pent De Graf K., Hollander A. Data Vault & Pentaho in HealthCare, http://www.bi-podium.nl/mediaFiles/upload/DWHgen/Pentaho_en_DV_-_KdG.pdf, 2014

PMR03 Peralta V., Marotta A., Ruggia R., Towards the Automation of Data Warehouse Design, 15th Conference on Advanced Information Systems Engineering, Velden, Austria, 2003

Pon10 Ponniah P., Data Warehousing Fundamentals For IT Professionals, Second Edition, Wiley, 2010

QBX Battaglia A., Golfarelli M., Rizzi S., QBX: A CASE tool for Data Mart design, O. De Troyer et al. (Eds.): ER 2011 Workshops, LNCS 6999, pp. 358–363, Springer-Verlag Berlin Heidelberg 2011

Quipu Data Warehouse Management, http://www.datawarehousemanagement.org/Quipu.aspx, 2014

RA07 Romero O., Abello A., Automating Multidimensional Design from Ontologies, DOLAP'07, November 2007

RA09 Romero O., Abello A., A Survey of Multidimensional Modeling Methodologies, International Journal of Data Warehousing & Mining, 5(2), 1-23, April-June 2009

Rai08 Rainardi V., Building The Data Warehouse With Examples in SQL Server, Springer, New York, 2008

RBM01 Rivest S., Bedard Y., Marchand P., Toward better suppport for spatial decision making: Defining the characteristics of spatial on-

line analytical processing (SOLAP), Geomatica 55, 2001.

Rio05 Riordan R.M., Designing effective database systems, Addison-Wasley, 2005

RK07 Radulović B., Krneta D., Modeli za ocenu kvaliteta softvera, SYMOPIS 2007., Zlatibor, ISBN: 978/86/7680/124/4

Ron01 Rönnbäck L., Anchor Modeling – A Technique for Information Under Evolution, GSE Nordic Conference, Stockholm, Sweden, 2001

RRBJW Regardt, O., Rönnbäck, L., Bergholtz, M., Johannesson, P., Wohed, P. Analysis of Normal Forms for Anchor Tables, 2010

RRBJW1 Rönnbäck, L., Regardt, O., Bergholtz, M., Johannesson, P., Wohed, P. Anchor Modeling: Naming Convention, 2010

Ron11 Rönnbäck, L. Anchor Modeling – A Technique for Information Under Evolution, GSE Nordics 2011 Conference, Stockholm, Sweden, 2011

SAP http://www.sap.com/pc/tech/data-warehousing/software/overview.html

SB10 Song J., Bao Y., Partitioned Dimension Modeling the Numerical Dimension in Data Warehouse, 12th International Asia-Pacific Web Conference, 2010

Sch95 Schulte, R., 1995: Three – Tier Computing Architectures and Beyond, Published Report Note R-401-134, Gartner Group

SchGoo http://scholar.google.com/

SD10 Suknović M., Delibašić B., Poslovna inteligencija i sistemi za podršku odlučivanju, FON Beograd, 2010

SixSig http://www.iassc.org/

SL90 Sheth P.A., Larson A.J. Federated Database Systems for Managing

Distributed, Heterogeneous, and Autonomous Databases, ACM Computing Surveys, Vol. 22, No. 3, September 1990

SA86 Snodgrass R, Ahn I., Temporal Databases, IEEE Computer 19, 1986

SOA http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

SA12 Srzentić M., Arbanas M., Extremly Normalized Data Warehouse Data, HROUG, 2012

SPD http://www.sybase.com/products/modelingdevelopment/powerdesigner

SS06 Skoutas, D., Simitsis, A., Designing ETL Processes Using Semantic Web Technologies, DOLAP, 2006.

SSC08 Simitsis A., Skoutas D., Castellanos M., Natural Language Reporting for ETL Processes, DOLAP'08, October 30, 2008, Napa Valley, California, USA

STT81 Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Trans. on Systems, Man, and Cybernetics 11(2), 109–125 (1981)

Tera http://www.teradata.com

TK98 Tsichiritzis, D., Klug, The ANSI/X3/SPARC DBMS framework, Report of the study group on database management systems. Information Systems, 3 1978

Vass00 Vassiliadis P., Data Warehouse Modeling and Quality Issues, Ph.D. Thesis, National Technical University of Athens, Greece, 2000

VBQ99 Vassiliadis P., Bouzeghoub M., Quix C., Towards Quality-Oriented Data Warehouse Usage and Evolution. In Proc. 11th Conference of Advanced Information Systems Engineering (CAiSE '99),

Heidelberg, Germany, 1999.

Vla04 Vlajić S., Projektovanje programa, FON, Beograd, 2004.

VSS02 Vassiliadis P., Simitsis A., Skiadopoulos S., Conceptual Modeling for ETL Procesaes, DOLAP'02, November 8, 2002, McLean, Virginia, USA

WAND Golfarelli M., Rizzi S., Saltarelli E.: WAND: A CASE Tool for Workload-Based Design of a Data Mart. SEBD 2002

Wes01 Westerman P., Data Warehousing, Morgan Kaufman Publishers, 2001

VBR03 Vrdoljak B., Banek M., Rizzi S., Designing Web Warehouses from XML Schemas, DaWaK 2003, LNCS 2737, Springer-Verlag Berlin Heidelberg 2003

WHO http://www.who.int/classifications/atcddd/en/

WS03 Winter, R., Strauch, B., A Method for Demand-Driven Information Requirements Analysis in DW Projects. In Proceedings of 36th Annual Hawaii International Conference on System Sciences, 2003

Woj11 Wojciechowski A., E-ETL: Framework For Managing Evolving ETL Processes, PIKM'11, October 28, 2011, Glasgow, Scotland, UK

ZC10 Zepeda L., Ceceña E., Quintero R., Zatarain R., Vega L. , Mora Z., Clemente G. G., A MDA Tool for Data Warehouse, International Conference on Computational Science and Its Applications, 2010

ZMA11 Zekri M., Marsit I., Adellatif A., A New Data Warehouse Approach Using Graph, Eighth IEEE International Conference on e-Business Engineering, 2011

Slike

Tabele

Dodatak A. Sadržaji tabela RuleTypes i Rules

U ovom dodatku su dati sadržaji tabela RuleTypes i Rules. Tabela RuleTypes sadrži informaciju o tipovima pravila kao npr. pravila za različite izvore podataka, pravila za identifikaciju hub, link i satelit tabela, te pravila za kreiranje hub, link i satelit tabela. Tabela Rules sadrži informacije o pravilima za konkretan tip pravila i akcije koje će se izvršavati kad je određen uslov zadovoljen. Pravila u koloni RuleFor omogućavaju lakše izvršavanje komandi koje su spremljene u kolonu RuleAction.

Tabela A 1. Sadržaj tabele RuleTypes

RuleTypeID	RuleType	RuleTypeDesc
1	Structured SourceData	Pravila za ucitavanje struktuiranih izvora podataka kao npr. MS SQL Server, Oracle, IBMdb2
2	Semi-Structured SourceData	Pravila za ucitavanje polustruktuiranih izvora podataka kao npr. Excel, XML
3	UnStructured SourceData	Pravila za ucitavanje nestruktuiranih izvora podataka kao npr. TXT
4	Hub Candidate	Pravila za identifikaciju Hub kandidata na osnovu Business key
5	Link Candidate	Pravila za identifikaciju Link kandidata na osnovu Hub-ova
6	Sat Candidate	Pravila za identifikaciju Sat kandidata na osnovu Hub-ova i Link-ova
7	CREATE Hub table	Pravila koja služe za generisanje SQL scripta za kreiranje Hub tabela
8	CREATE Link table	Pravila koja služe za generisanje SQL scripta za kreiranje Link tabela
9	CREATE Sat table	Pravila koja služe za generisanje SQL scripta za kreiranje Sat tabela
10	INSERT Hub table	Pravila koja služe za generisanje SQL scripta za ucitavanje podataka iz izvora podataka u Hub tabele
11	INSERT Link table	Pravila koja služe za generisanje SQL scripta za ucitavanje podataka iz izvora podataka u Link tabele
12	INSERT Sat table	Pravila koja služe za generisanje SQL scripta za ucitavanje podataka iz izvora podataka u Sat tabele
13	Dim Candidate	Pravila za identifikaciju kandidata za dimenzione tabele

14	CREATE Dim and Fact	Pravila koja služe za generisanje SQL scripta koji služi za kreiranje tabela dimenzija i mjera
15	INSERT Dim table	Pravila koja služe za generisanje SQL scripta za ucitavanje podataka iz skladišta podataka u Dim tabele
16	INSERT Fact table	Pravila koja služe za generisanje SQL scripta za ucitavanje podataka iz skladišta podataka u Fact tabele
17	CREATE Sat table SS	Pravila koja služe za generisanje SQL scripta za kreiranje Sat tabela na osnovu polustruktuiranog izvora
18	INSERT Sat table SS	Pravila koja služe za generisanje SQL scripta za ucitavanje podataka u Sat tabelu iz polustruktuiranog izvora
19	CREATE Dim Column SS	Pravila koja služe za generisanje SQL scripta za kreiranje Dim kolone na osnovu polustruktuiranog izvora
20	UPDATE Dim Column SS	Pravila koja služe za generisanje SQL scripta za ucitavanje Dim kolone iz polustruktuiranog izvora

Tabela A2. Sadržaj tabele Rules

Rule TypeID	RuleID	RuleFor	RuleAction	RuleDesc
1	1	Source-MSSQLServer	sp_InsertDSTables	Ucitavanje metapodataka podataka u tabelu DataSources i Tables za odgovarajuci izvor podataka
1	2	Metadata-MSSQLServer	sp_InsertTablesColumns	Ucitavanje metapodataka izvora podataka u tabelu TableColumns
2	1	Data-Excel	sp_SS_ImportData	Ucitavanje podataka iz Excel izvora podataka
2	2	Metadata-Excel	sp_SS_metadata	Ucitavanje metapodataka iz Excel izvora podataka
4	1	HubCandidate	UPDATE TableColumns SET SurrogateKey = 1, HubCandidate = 1 WHERE (BusinessKey = 1 OR ColumnPK = 1) AND TableID IN (SELECT TableID FROM TableColumns WHERE BusinessKey = 1) AND BusinessKey = 0	Pravilo za popunjavanje polja HubCandidate na osnovu popunjenog polja BusinessKey

5	1	LinkCandidate	sp_FindLinkCandidate	Pravilo za popunjavanje polja LinkCandidate
6	1	SatCandidate	sp_FindSatCandidate	Pravilo za popunjavanje polja SatCandidate
7	1	CreateHubTables	sp_CreateHubTables	Pravilo za generisanje SQL scripta za kreiranje Hub tabela
8	1	CreateLinkTables	sp_CreateLinkTables	Pravilo za generisanje SQL scripta za kreiranje Link tabela
9	1	CreateSatTables	sp_CreateSatTables	Pravilo za generisanje SQL scripta za kreiranje Sat tabela
10	1	InsertHubTables	sp_InsertHub	Pravilo za generisanje SQL scripta za ucitavanje podataka iz izvora podataka u Hub tabele
11	1	InsertLinkTables	sp_InsertLink	Pravilo za generisanje SQL scripta za ucitavanje podataka iz izvora podataka u Link tabele
12	1	InsertSatTables	sp_InsertSat	Pravilo za generisanje SQL scripta za ucitavanje podataka iz izvora podataka u Sat tabele
13	1	DimCandidate	UPDATE DV_TableColumns SET DimCandidate = 1 WHERE LEFT (TableName, 4) = 'Sat_' AND ColumnPK = 1 AND TableName NOT IN (SELECT TableName FROM DV_TableColumns WHERE FactCandidate = 1) AND DataType <> 'datetime'	Pravila za inicijalnu identifikaciju kandidata za dimenzione tabele
14	1	CreateDimFactTables	sp_CreateDimFactTables	Pravila koja služe za generisanje SQL scripta koji služi za kreiranje tabela dimenzija i mjera
15	1	InsertDimTables	sp_InsertDim	Pravilo za generisanje SQL scripta za ucitavanje podataka iz skladišta podataka u Dim tabele
16	1	InsertFactTables	sp_InsertFact	Pravilo za generisanje SQL scripta za ucitavanje podataka iz skladišta podataka u Fact tabele
17	1	CreateSatTableSS	sp_CreateSatTableSS	Pravilo za generisanje SQL scripta za kreiranje Sat tabela na osnovu polustruktuiranog izvora

18	1	InsertSatTableSS	sp_InsertSatTableSS	Pravila koja služe za generisanje SQL scripta za ucitavanje podataka u Sat tabelu iz polustruktuiranog izvora
19	1	CreateDimColumnSS	sp_AddDimColumnSS	Pravilo koje služi za generisanje SQL scripta za kreiranje Dim kolone na osnovu polustruktuiranog izvora
20	1	UpdateDimColumnSS	sp_UpdateDimSS	Pravilo koje služi za generisanje SQL scripta za ucitavanje Dim kolone iz polustruktuiranog izvora

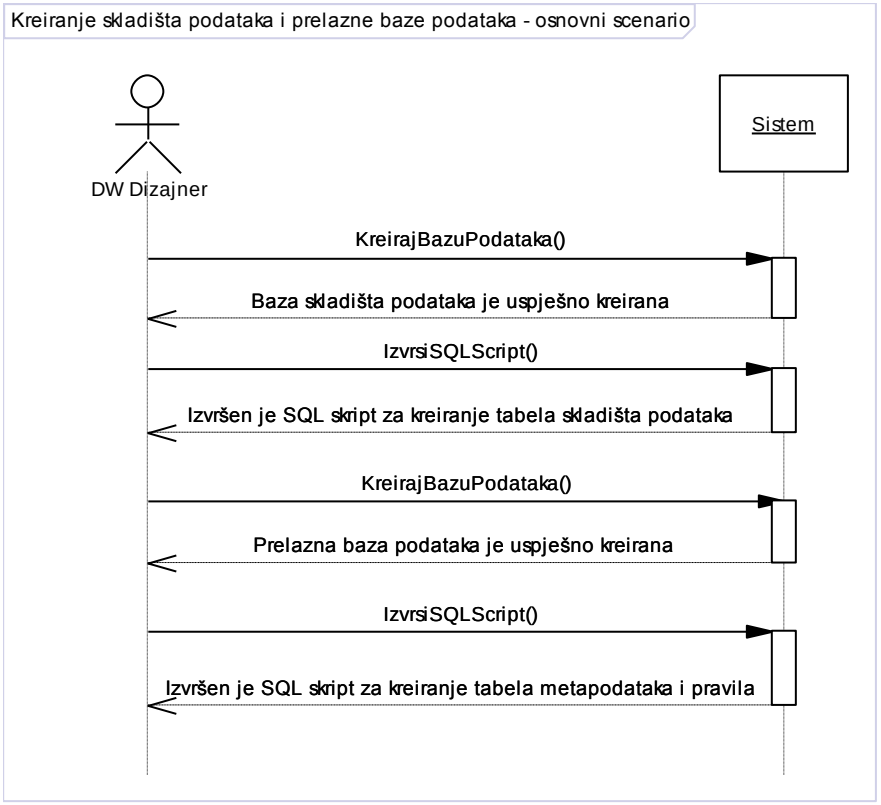
Dodatak B. Dijagrami sekvenci

Ponašanje softverskog sistema preko sistemskih dijagrama sekvenci (DS) urađeno je na taj način što je za svaki slučaj korišćenja i za svaki scenario urađeni dijagrami i to samo za slučajeve kada akter poziva sistem da izvrši sistemsku operaciju (APSO) i kad sistem daje izlazne argumente (IA) iz sistema.

DS - 1.1. Dijagram sekvenci za slučaj korišćenja SK - 1.1. Kreiranje prelazne baze i baze skladišta podataka

Osnovni scenario:

1. Dizajner poziva sistem da kreira bazu skladišta podataka (APSO)
2. Sistem prikazuje poruku da je baza skladišta podataka kreirana (IA)
3. Dizajner poziva sistem da u skladištu podataka kreira procedure (koje su generisane iz modela) za kreiranje tabela skladišta podataka (APSO)
4. Sistem prikazuje poruku da su procedure za kreiranje tabela skladišta kreirane (IA)
5. Dizajner poziva sistem da kreira prelaznu bazu podataka (APSO)
6. Sistem prikazuje poruku da je prelazna baza podataka kreirana (IA)
7. Dizajner poziva sistem da u prelaznoj bazi podataka kreira tabela metapodataka i tabele pravila, (koje su generisane iz modela), te da učita osnovne šifarnike i tabele pravila (APSO)
8. Sistem prikazuje poruku da su u prelaznoj bazi podataka kreira tabela metapodataka i tabele pravila, te da učitani osnovni šifarnici i tabele pravila (IA)



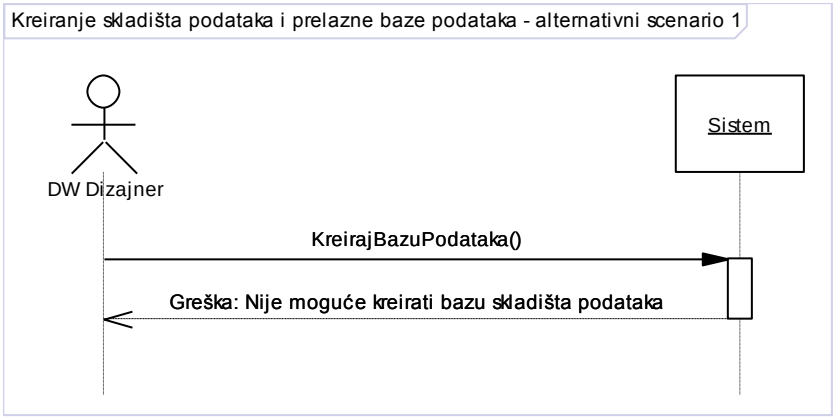
Slika B-**1**. Dijagram sekvenci za slučaj korišćenja SK - 1.1. Kreiranje prelazne baze i baze skladišta podataka, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- KreirajBazuPodataka (Database, ImeFajla, Velicina, MaxVelicina)
- IzvrsiSQLScript (Database, List <SqlScript>)

Alternativni scenario 1:

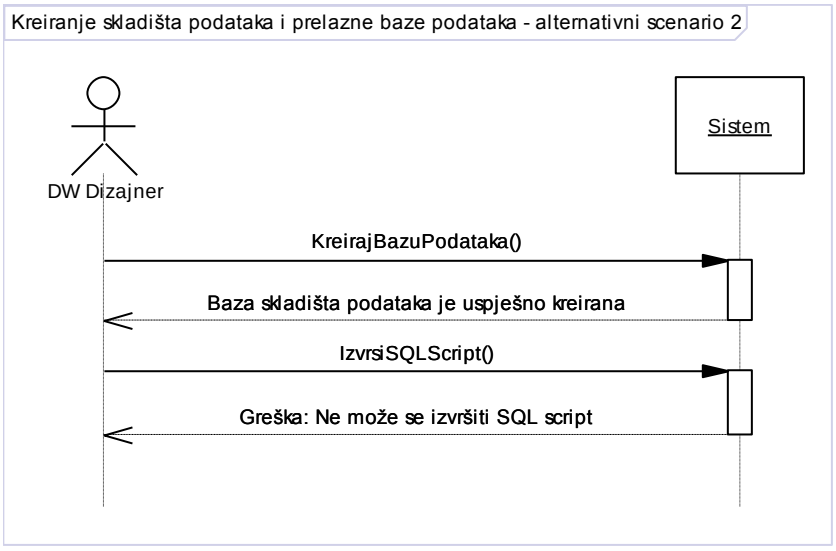
Ukoliko se ne može kreirati baza skladišta podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-2. Dijagram sekvenci za slučaj korišćenja SK - 1.1. Kreiranje prelazne baze i baze skladišta podataka, alternativni scenario 1

Alternativni scenario 2:

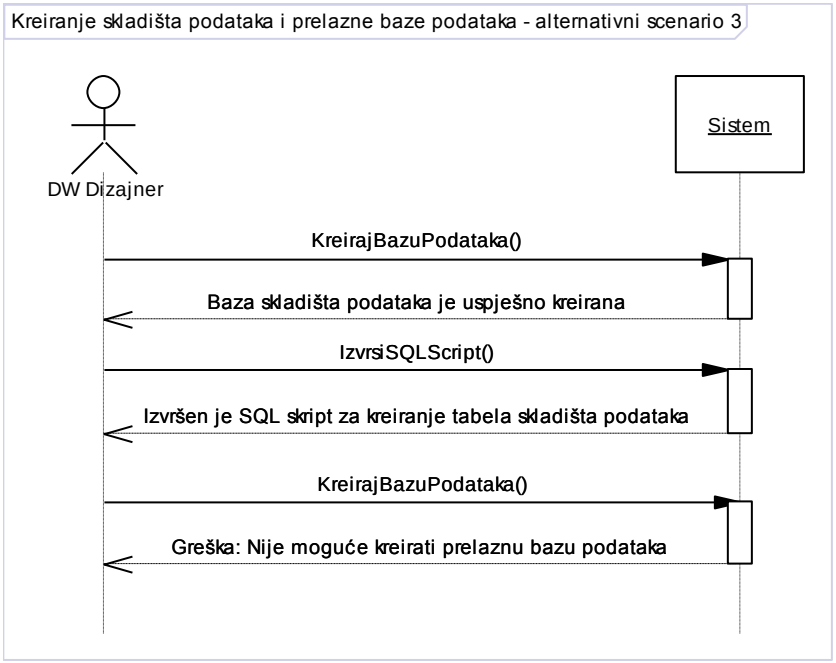
Ukoliko ne mogu da se kreiraju procedure za kreiranje tabela skladišta podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-3. Dijagram sekvenci za slučaj korišćenja SK - 1.1. Kreiranje prelazne baze i baze skladišta podataka, alternativni scenario 2

Alternativni scenario 3:

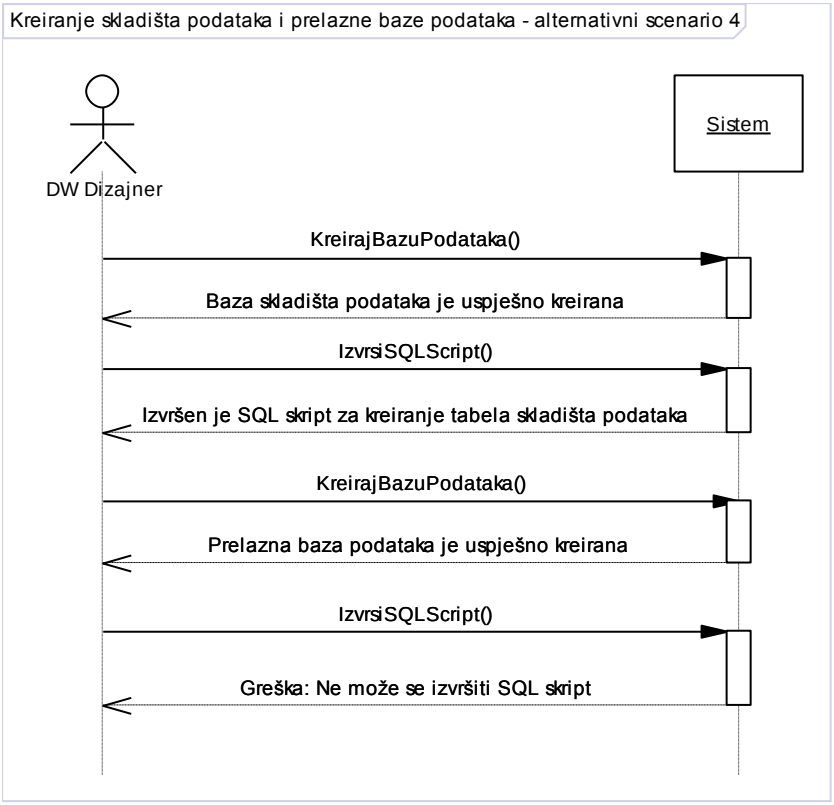
Ukoliko se ne može kreirati prelazna baza podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-4. Dijagram sekvenci za slučaj korišćenja SK - 1.1. Kreiranje prelazne baze i baze skladišta podataka, alternativni scenario 3

Alternativni scenario 4:

Ukoliko ne mogu da se kreiraju tabela metapodataka i tabele pravila, te da učitaju osnovni šifarnici i tabele pravila, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



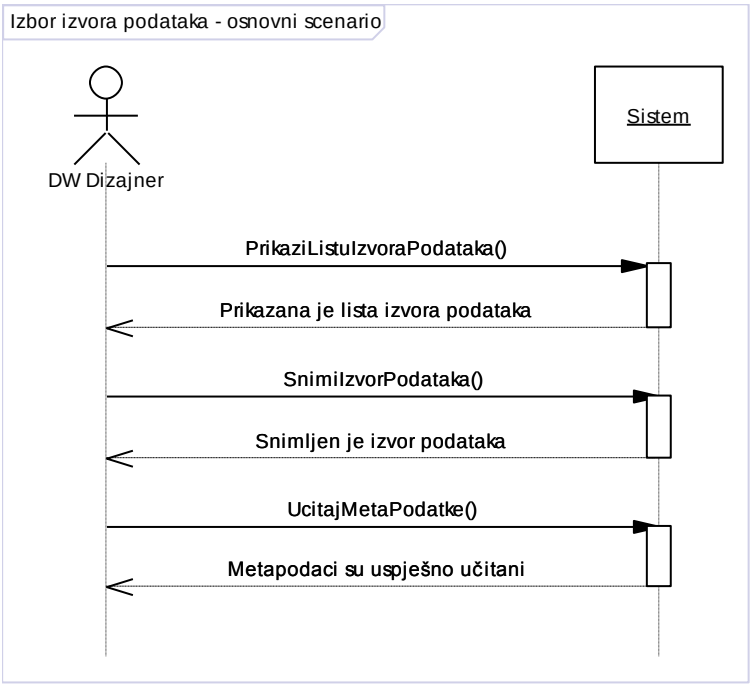
Slika B-5. Dijagram sekvenci za slučaj korišćenja SK - 1.1. Kreiranje prelazne baze i baze skladišta podataka, alternativni scenario 4

DS - 1.2. Dijagram sekvenci za slučaj korišćenja SK - 1.2. Izbor izvora podataka za skladište podataka

Osnovni scenario:

1. Dizajner poziva sistem da prikaže listu struktuiranih izvora podataka (APSO)
2. Sistem prikazuje listu dostupnih struktuiranih izvora podataka (IA)
3. Dizajner poziva sistem da snimi informaciju o izabranim struktuiranim izvorima podataka (APSO)
4. Sistem vraća poruku da su snimljeni struktuirani izvori podataka (IA)
5. Dizajner poziva sistem da izvrši učitavanje metapodataka iz struktuiranih izvora podataka (APSO)

6. Sistem vraća poruku da su učitani metapodaci struktuiranih izvora podataka (IA)



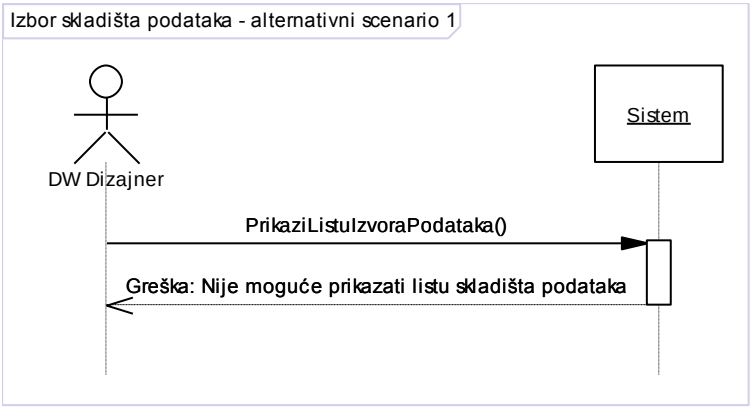
Slika B-6. Dijagram sekvenci za slučaj korišćenja SK - 1.2. Izbor izvora podataka za skladište podataka, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- PrikaziListuIzvoraPodataka (Server)
- SnimiIzvorPodataka (IzvorPodataka)
- UcitajMetaPodatke (IzvorPodataka)

Alternativni scenario 1:

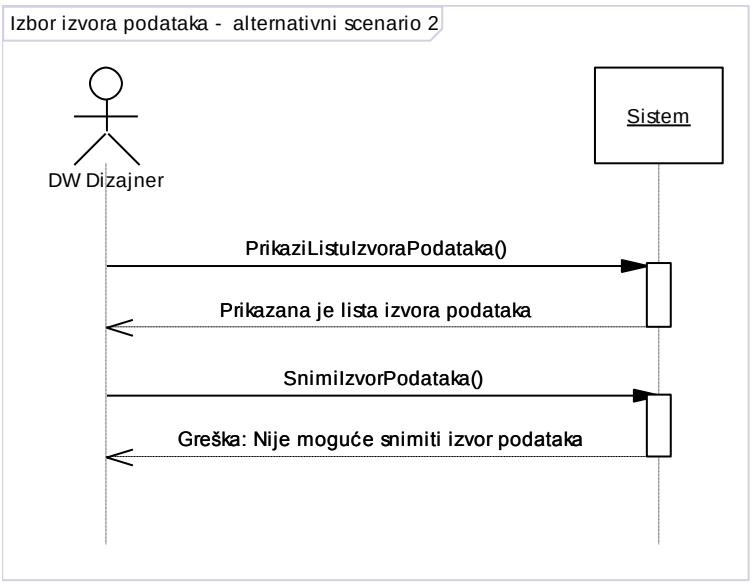
Ukoliko se ne mogu prikazati informacije o dostupnim skladištima podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-7. Dijagram sekvenci za slučaj korišćenja SK - 1.2. Izbor izvora podataka za skladište podataka, alternativni scenario 1

Alternativni scenario 2:

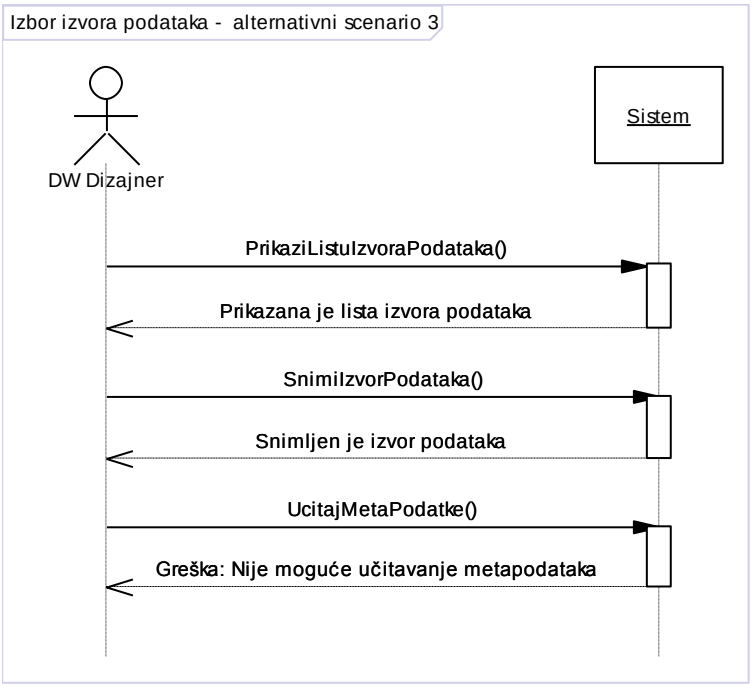
Ukoliko se ne mogu snimiti informacije o izabranom skladištu podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-8. Dijagram sekvenci za slučaj korišćenja SK - 1.2. Izbor izvora podataka za skladište podataka, alternativni scenario 2

Alternativni scenario 3:

Ukoliko se ne mogu učitati metapodaci skladišta podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

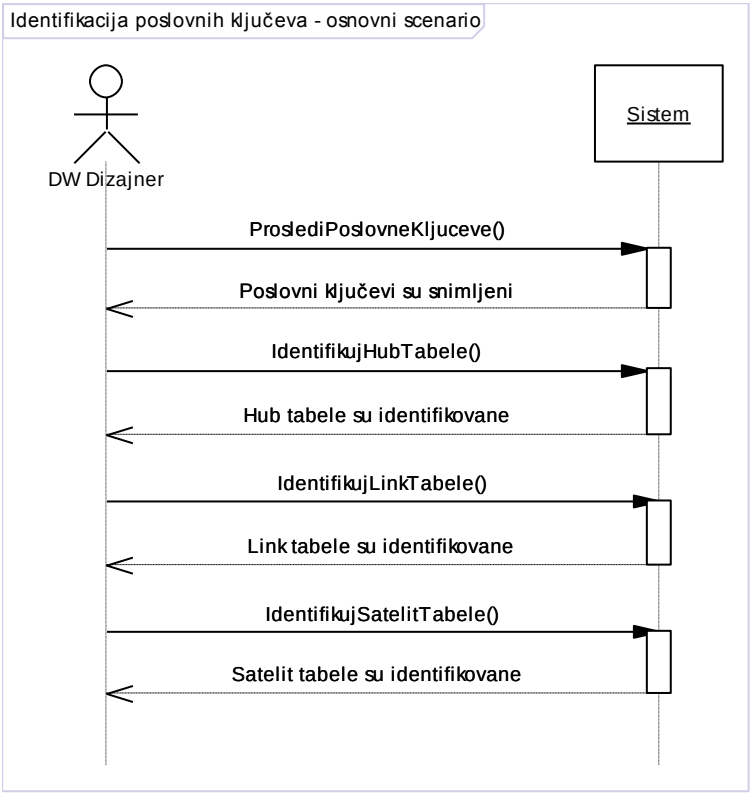


Slika B-9. Dijagram sekvenci za slučaj korišćenja SK - 1.2. Izbor izvora podataka za skladište podataka, alternativni scenario 3

DS - 1.3. Dijagram sekvenci za slučaj korišćenja SK - 1.3. Identifikacija poslovnih ključeva

Osnovni scenario:

1. Dizajner prosleđuje sistemu poslovne ključeve (APSO)
2. Sistem vraća informaciju da su snimljeni poslovni ključevi (IA)
3. Dizajner poziva sistem da identifikuje hub tabele (APSO)
4. Sistem vraća informaciju da su na osnovu poslovnih ključeva identifikovane hub tabele (IA)
5. Dizajner poziva sistem da identifikuje link tabele (APSO)
6. Sistem vraća informaciju da su identifikovane link tabele (IA)
7. Dizajner poziva sistem da identifikuje satelit tabele (APSO)
8. Sistem vraća informaciju da su identifikovane satelit tabele (IA)



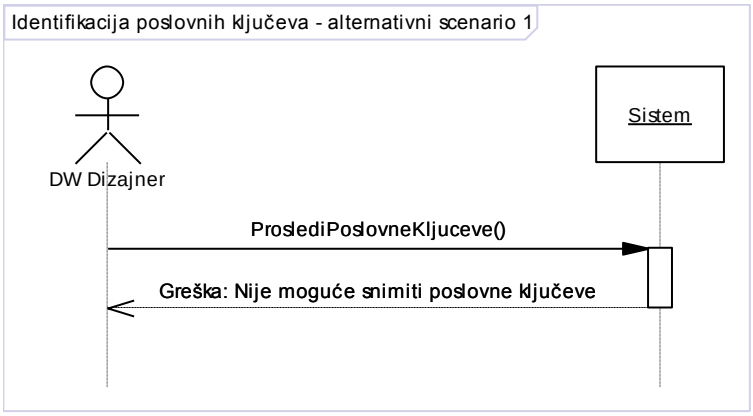
Slika B-10. Dijagram sekvenci za slučaj korišćenja SK - 1.3. Identifikacija poslovnih ključeva, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- SnimiPoslovneKljuceve (Kolona, Vrijednost)
- IdentifikujHubTabele (Pravilo)
- Identifikuj Link tabele (Pravilo)
- Identifikuj Satelit tabele (Pravilo)

Alternativni scenario 1:

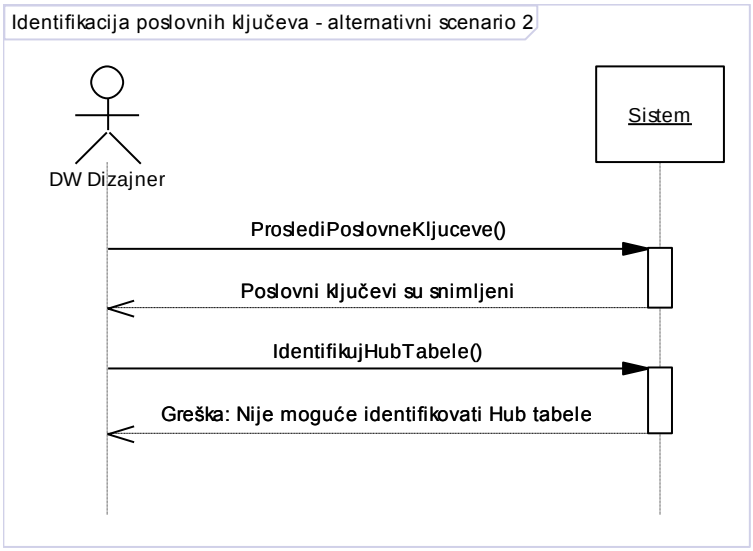
Ukoliko nije moguće snimiti informaciju o poslovnim ključevima sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-11. Dijagram sekvenci za slučaj korišćenja SK - 1.3. Identifikacija poslovnih ključeva, alternativni scenario 1

Alternativni scenario 2:

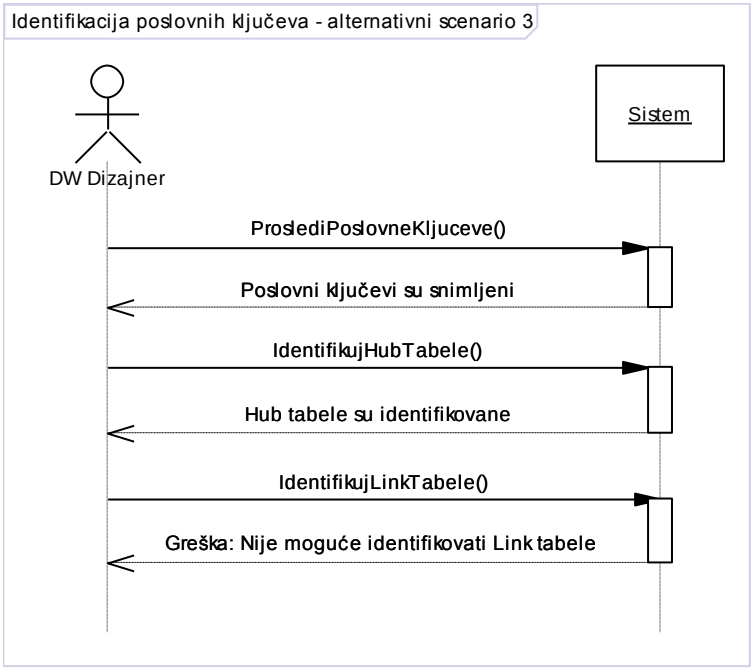
Ukoliko nije moguće identifikovati hub tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-12. Dijagram sekvenci za slučaj korišćenja SK - 1.3. Identifikacija poslovnih ključeva, alternativni scenario 2

Alternativni scenario 3:

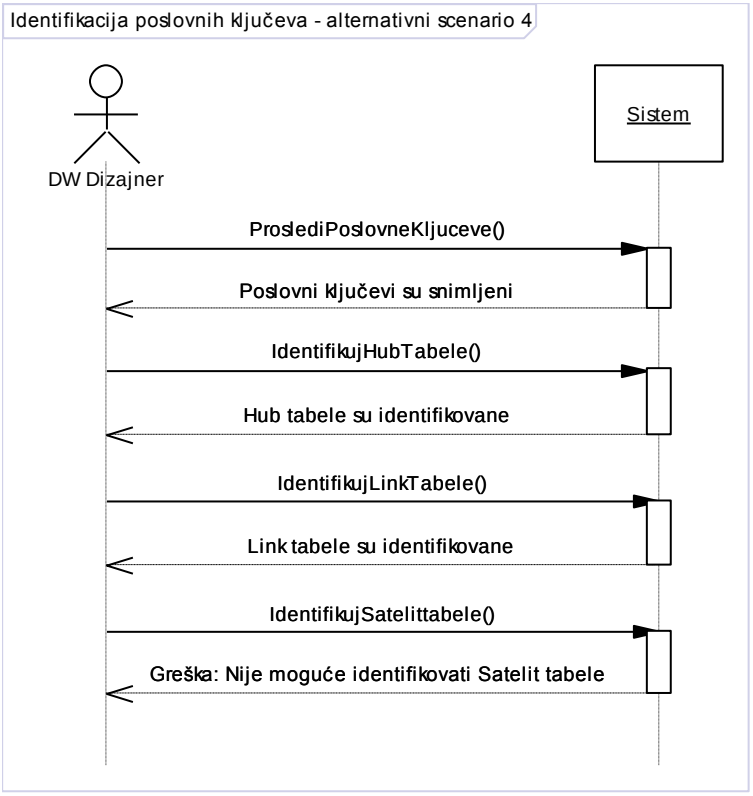
Ukoliko nije moguće identifikovati link tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-13. Dijagram sekvenci za slučaj korišćenja SK - 1.3. Identifikacija poslovnih ključeva, alternativni scenario 3

Alternativni scenario 4:

Ukoliko nije moguće identifikovati satelit tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

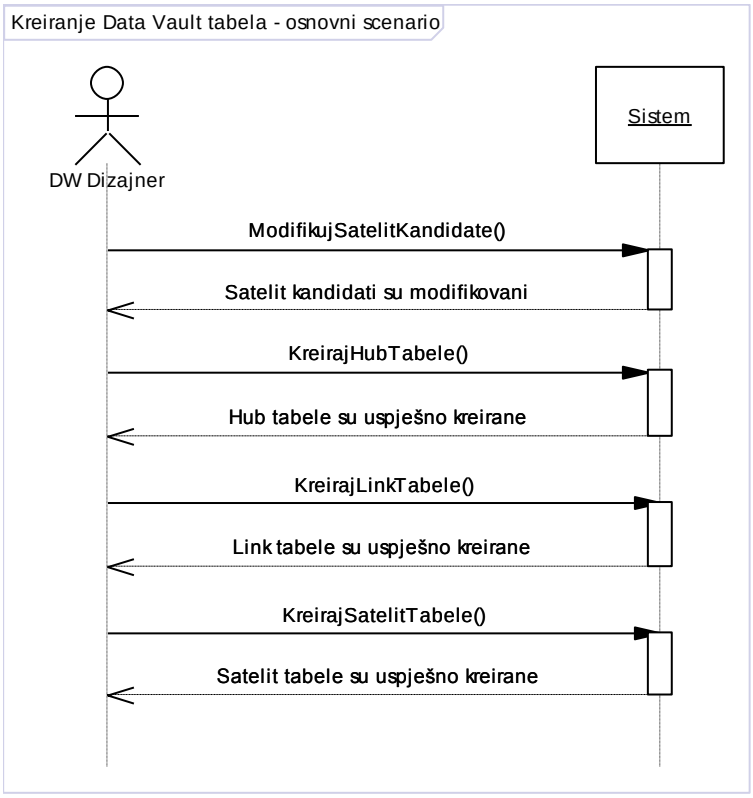


Slika B-14. Dijagram sekvenci za slučaj korišćenja SK - 1.3. Identifikacija poslovnih ključeva, alternativni scenario 4

DS - 1. 4. Dijagram sekvenci za slučaj korišćenja SK - 1. 4. Kreiranje Data Vault tabela

Osnovni scenario:

1. Dizajner, po potrebi, poziva sistem da snimi satelit kandidate (APSO)
2. Sistem vraća informaciju da su snimljene modifikacije o satelit kandidatima (IA)
3. Dizajner poziva sistem da kreira hub tabele (APSO)
4. Sistem vraća informaciju da su kreirane hub tabele (IA)
5. Dizajner poziva sistem da kreira link tabele (APSO)
6. Sistem vraća informaciju da su kreirane link tabele (IA)
7. Dizajner poziva sistem da kreira satelit tabele (APSO)
8. Sistem vraća informaciju da su kreirane satelit tabele (IA)



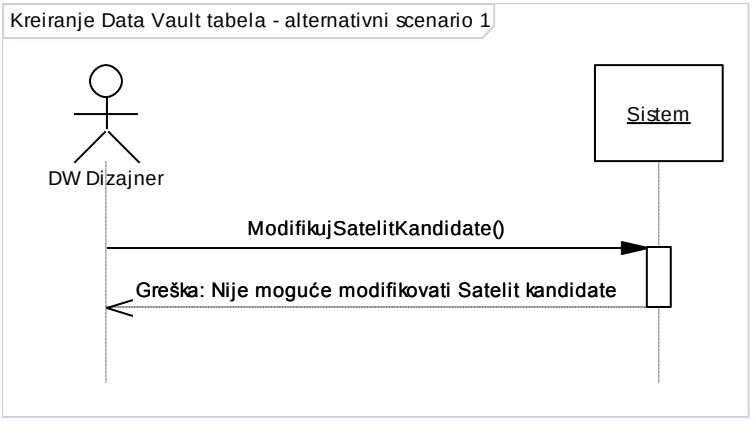
Slika B-1588. Dijagram sekvenci za slučaj korišćenja SK - 1. 4. Kreiranje Data Vault tabela, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- ModifikujSatelitKandidate (Kolona, Vrijednost)
- KreirajHubTabele (Pravilo, SkladistePodataka)
- KreirajLinkTabele (Pravilo, SkladistePodataka)
- KreirajSatelitKandidate (Pravilo, SkladistePodataka)

Alternativni scenario 1:

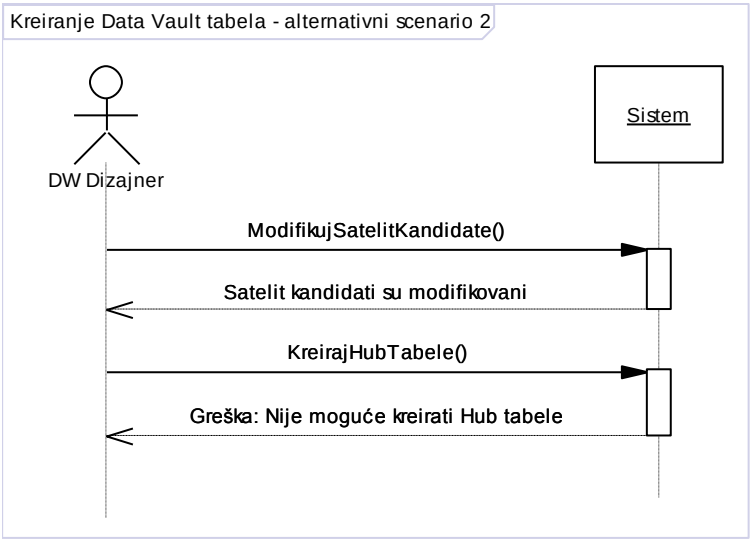
Ukoliko nije moguće snimiti informaciju o modifikovanim satelit kandidatima, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-16. Dijagram sekvenci za slučaj korišćenja SK - 1. 4. Kreiranje Data Vault tabela, alternativni scenario 1

Alternativni scenario 2:

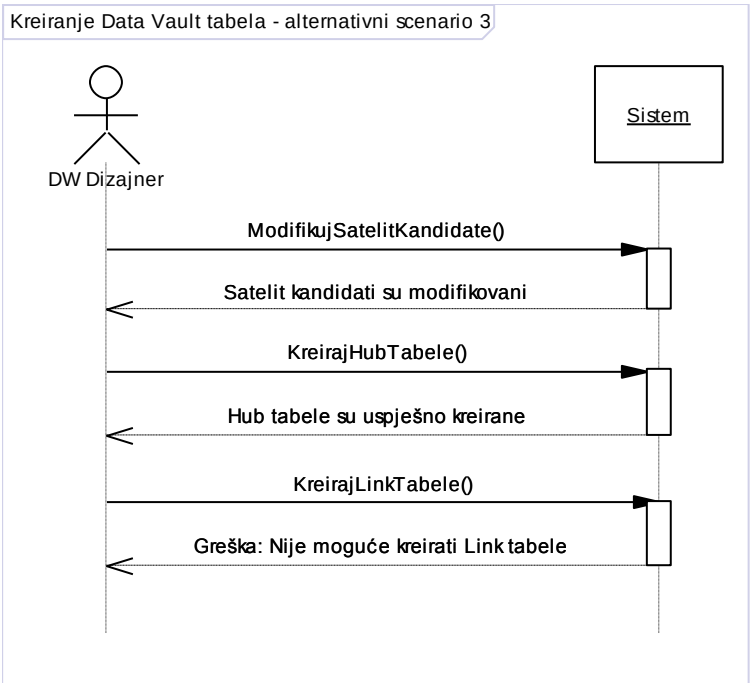
Ukoliko nije moguće kreiranje hub tabela sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-17. Dijagram sekvenci za slučaj korišćenja SK - 1. 4. Kreiranje Data Vault tabela, alternativni scenario 2

Alternativni scenario 3:

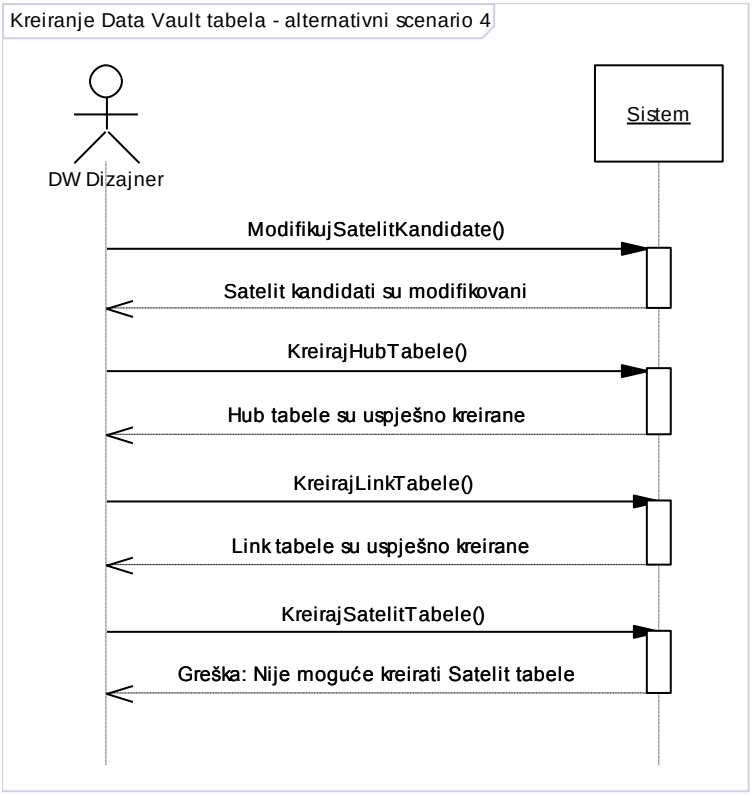
Ukoliko nije moguće kreiranje link tabela sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-18. Dijagram sekvenci za slučaj korišćenja SK - 1. 4. Kreiranje Data Vault tabela, alternativni scenario 3

Alternativni scenario 4:

Ukoliko nije moguće kreiranje satelit tabela sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



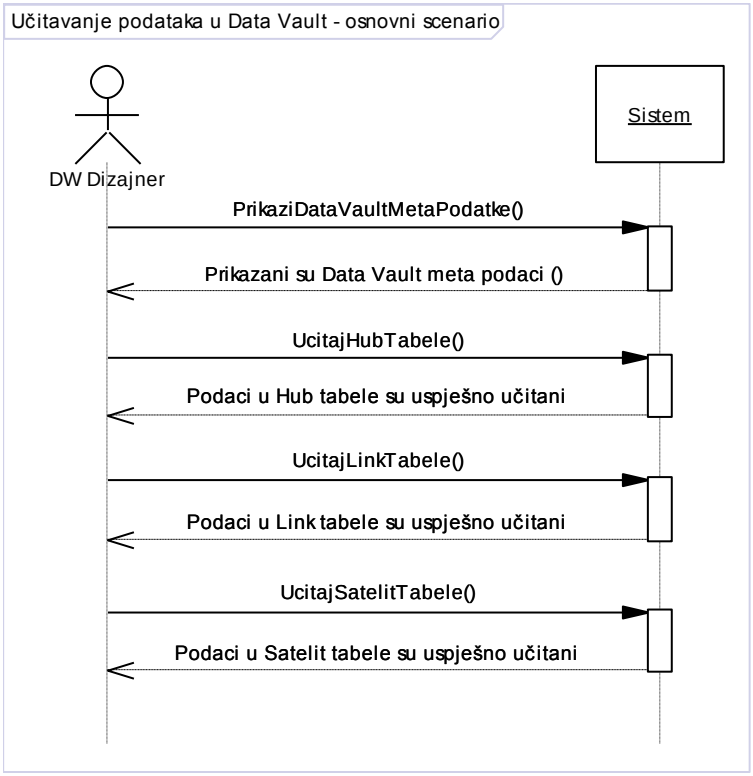
Slika B-19. Dijagram sekvenci za slučaj korišćenja SK - 1. 4. Kreiranje Data Vault tabela, alternativni scenario 4

DS - 1.5. Dijagram sekvenci za slučaj korišćenja SK - 1.5. Učitavanje podataka u Data Vault skladište podataka

Osnovni scenario:

1. Dizajner poziva sistem da prikaže Data Vault metapodatke (APSO)
2. Sistem prikazuje Data Vault metapodatke (IA)
3. Dizajner pokreće proces ekstrakcije i učitavanja podataka u hub tabele (APSO)
4. Sistem vraća informaciju da su učitani podaci u hub tabele (IA)
5. Dizajner pokreće proces ekstrakcije i učitavanja podataka u link tabele (APSO)
6. Sistem vraća informaciju da su učitani podaci u link tabele (IA)

7. Dizajner pokreće proces ekstrakcije i učitavanja podataka u satelit tabele (APSO)
8. Sistem vraća informaciju da su učitani podaci u satelit tabele (IA)



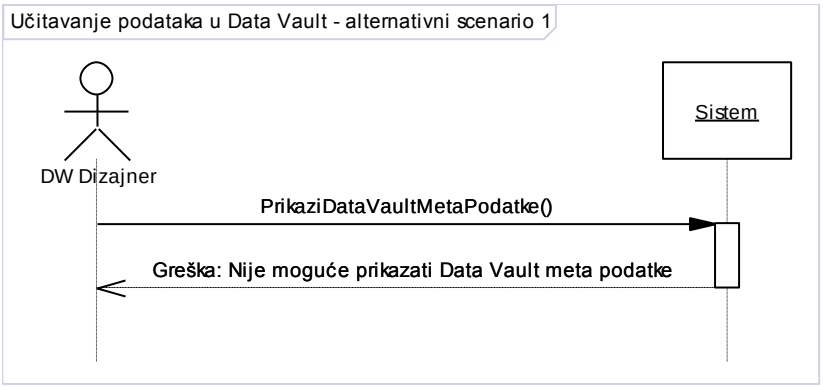
Slika B-20. Dijagram sekvenci za slučaj korišćenja SK - 1.5. Učitavanje podataka u Data Vault skladište podataka, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- PrikaziDataVaultMetaPodatke(SkladistePodataka, PrelaznaBaza)
- UcitajHubTabele (Pravilo)
- UcitajLinkTabele (Pravilo)
- UcitajSatelitTabele (Pravilo)

Alternativni scenario 1:

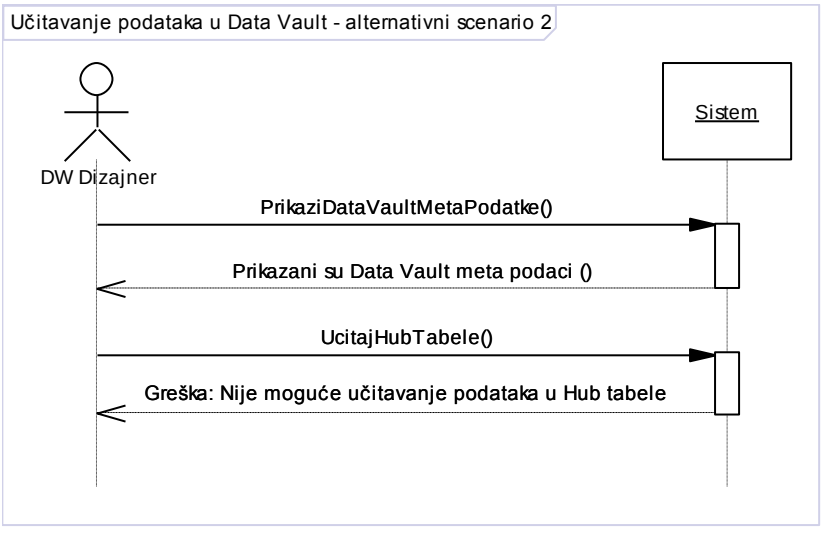
Ukoliko nije moguće prikazati Data Vault metapodatke, sistem vraća poruku o grešci, prekida se izvršenje scenarija



Slika B-21. Dijagram sekvenci za slučaj korišćenja SK - 1.5. Učitavanje podataka u Data Vault skladište podataka, alternativni scenario 1

Alternativni scenario 2:

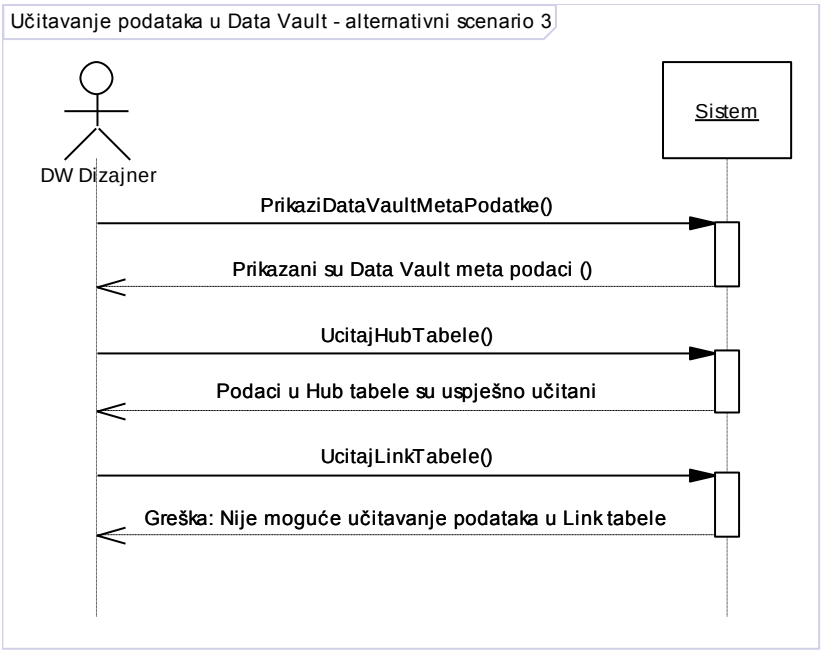
Ukoliko nije moguće učitavanje podataka u hub tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-22. Dijagram sekvenci za slučaj korišćenja SK - 1.5. Učitavanje podataka u Data Vault skladište podataka, alternativni scenario 2

Alternativni scenario 3:

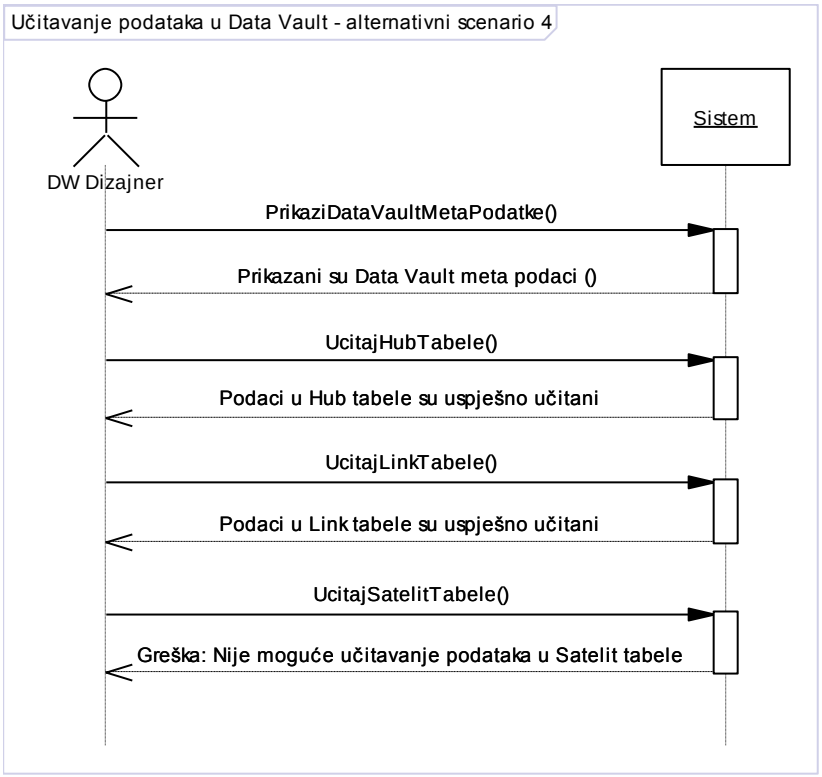
Ukoliko nije moguće učitavanje podataka u link tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-23. Dijagram sekvenci za slučaj korišćenja SK - 1.5. Učitavanje podataka u Data Vault skladište podataka, alternativni scenario 3

Alternativni scenario 4:

Ukoliko nije moguće učitavanje podataka u satelit tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

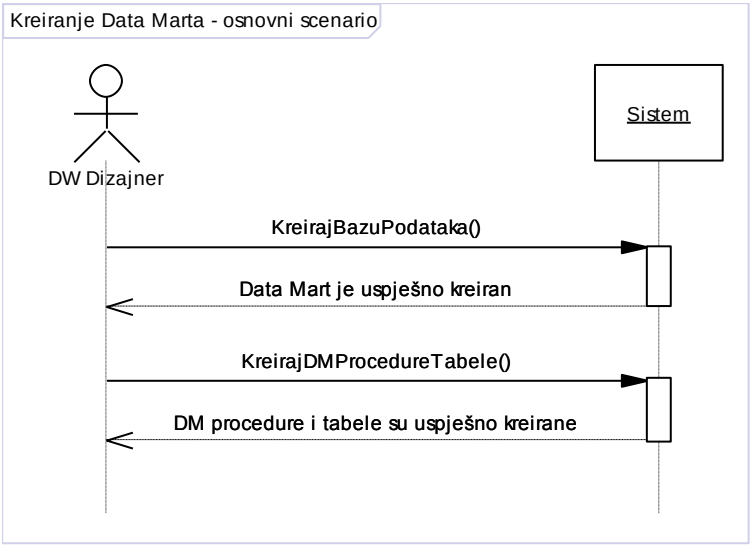


Slika B-24. Dijagram sekvenci za slučaj korišćenja SK - 1.5. Učitavanje podataka u Data Vault skladište podataka, alternativni scenario 4

DS - 2.1. Dijagram sekvenci za slučaj korišćenja SK - 2.1. Kreiranje data mart šeme ili baze podataka

Osnovni scenario:

1. Dizajner poziva sistem da kreira data mart bazu podataka (APSO)
2. Sistem prikazuje poruku da je baza podataka kreirana (IA)
3. Dizajner poziva sistem da kreira: procedure (koje su generisane iz modela) za kreiranje tabela data marta, tabele metapodataka Data Vaulta i učita tabele pravila (APSO)
4. Sistem prikazuje poruku da su procedure i tabele kreirane i učitane tabele pravila (IA)



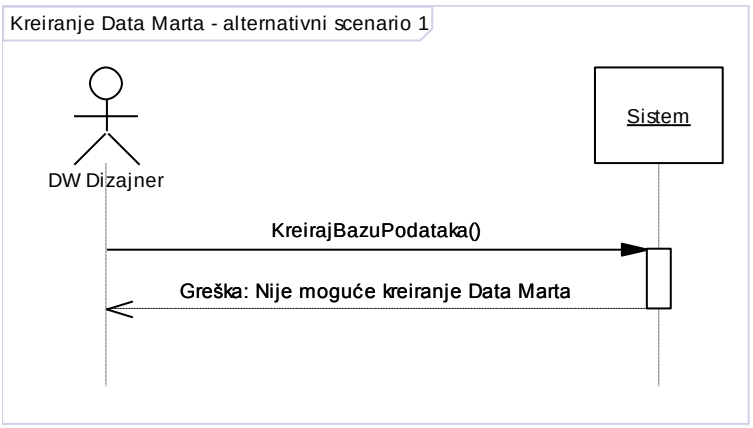
Slika B-25. Dijagram sekvenci za slučaj korišćenja SK - 2.1. Kreiranje data mart šeme ili baze podataka, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- KreirajBazuPodataka (Database, ImeFajla, Velicina, MaxVelicina)
- KreirajDMProcedureTabele (DatabaseName, List <SqlScript>)

Alternativni scenario 1:

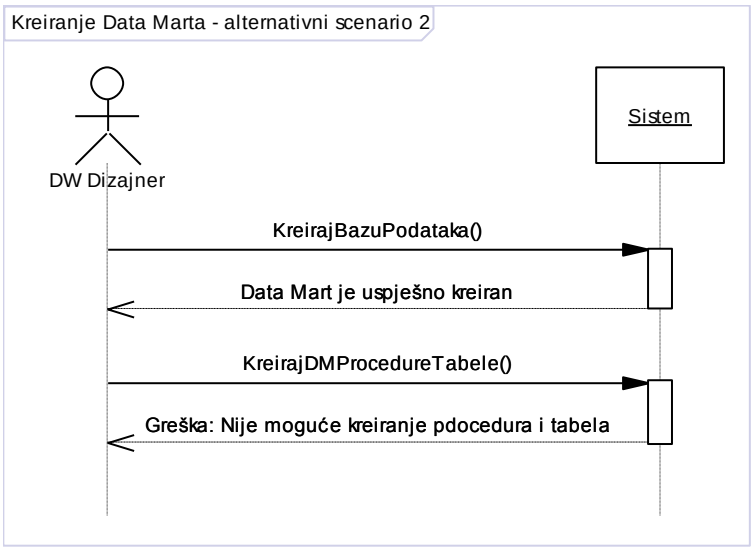
Ukoliko ne može da kreira data mart bazu podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-26. Dijagram sekvenci za slučaj korišćenja SK - 2.1. Kreiranje data mart šeme ili baze podataka, alternativni scenario 1

Alternativni scenario 2:

Ukoliko sistem ne može da kreira procedure (koje su generisane iz modela) za kreiranje tabela data marta, tabele metapodataka Data Vaulta i učita tabele pravila, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



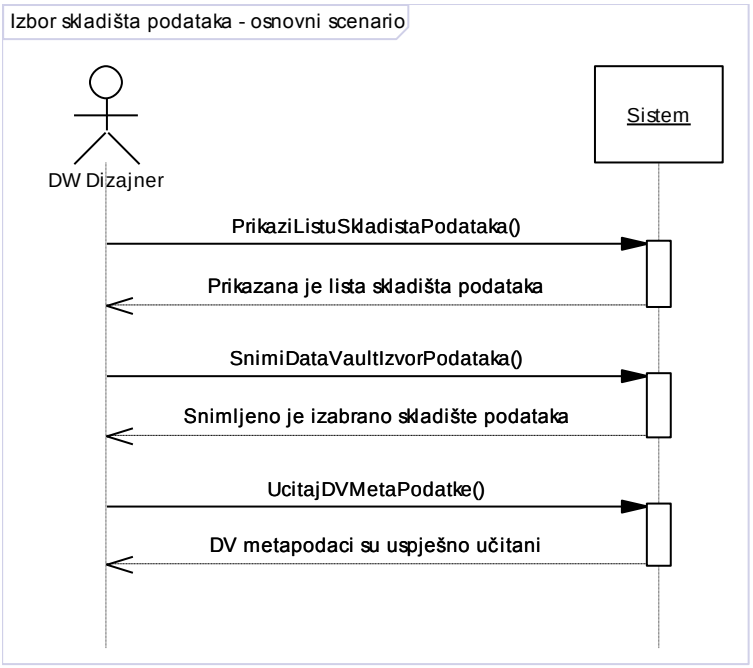
Slika B-27. Dijagram sekvenci za slučaj korišćenja SK - 2.1. Kreiranje data mart šeme ili baze podataka, alternativni scenario 2

DS - 2.2. Dijagram sekvenci za slučaj korišćenja SK - 2.2. Izbor skladišta podataka

Osnovni scenario:

1. Dizajner poziva sistem da prikaže listu skladišta podataka (APSO)
2. Sistem prikazuje listu dostupnih skladišta podataka (IA)
3. Dizajner poziva sistem da snimi informaciju o izabranom skladištu podataka (APSO)
4. Sistem vraća poruku da su snimljene informacije o izabranom skladištu podataka (IA)

5. Dizajner poziva sistem da izvrši učitavanje metapodataka iz izabranog skladišta podataka (APSO)
6. Sistem vraća poruku da su učitani meta izabranog skladišta podataka (IA)



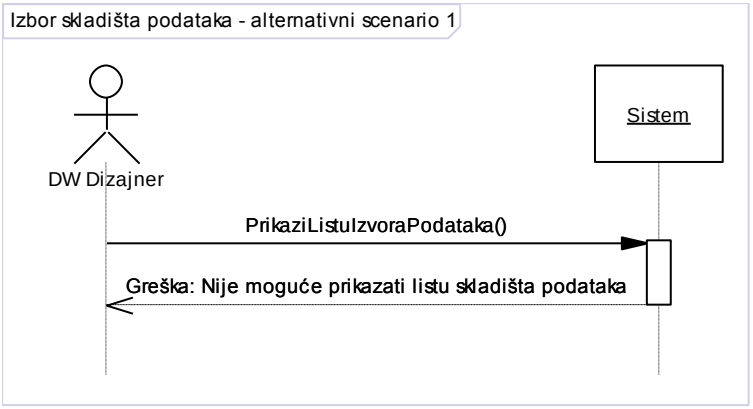
Slika B-28. Dijagram sekvenci za slučaj korišćenja SK - 2.2. Izbor skladišta podataka, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- PrikaziListuIzvoraPodataka (Server)
- SnimiDataMartIzvorPodataka (SkladistePodataka)
- UcitajDVMetaPodatke (SkladistePodataka)

Alternativni scenario 1:

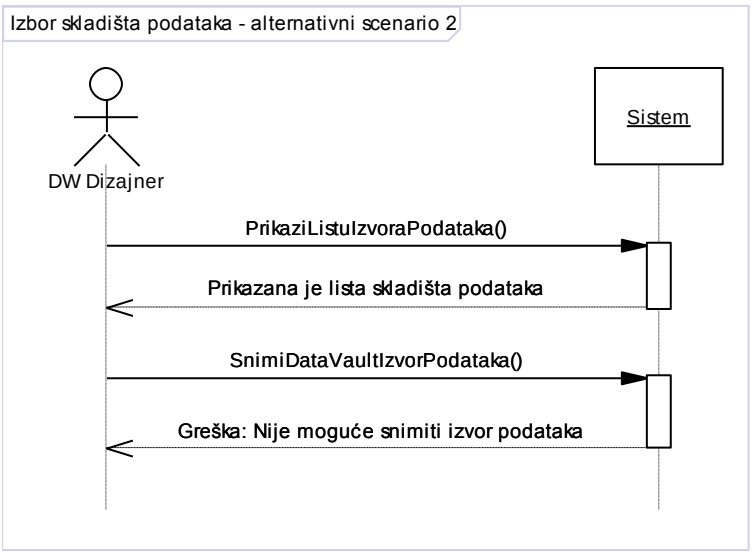
Ukoliko se ne mogu prikazati informacije o dostupnim skladištima podataka sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-29. Dijagram sekvenci za slučaj korišćenja SK - 2.2. Izbor skladišta podataka, alternativni scenario 1

Alternativni scenario 2:

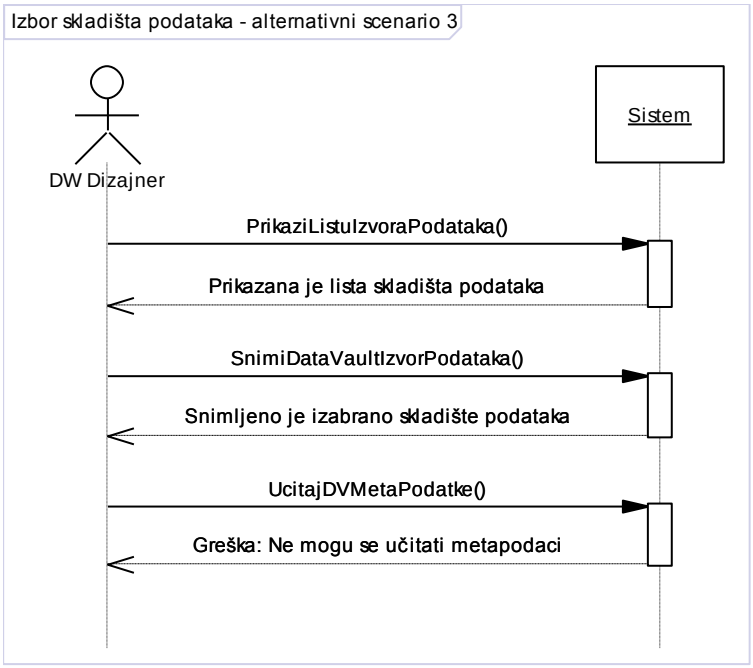
Ukoliko se ne mogu snimiti informacije o izabranom skladištu podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-30. Dijagram sekvenci za slučaj korišćenja SK - 2.2. Izbor skladišta podataka, alternativni scenario 2

Alternativni scenario 3:

Ukoliko se ne mogu učitati metapodaci skladišta podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

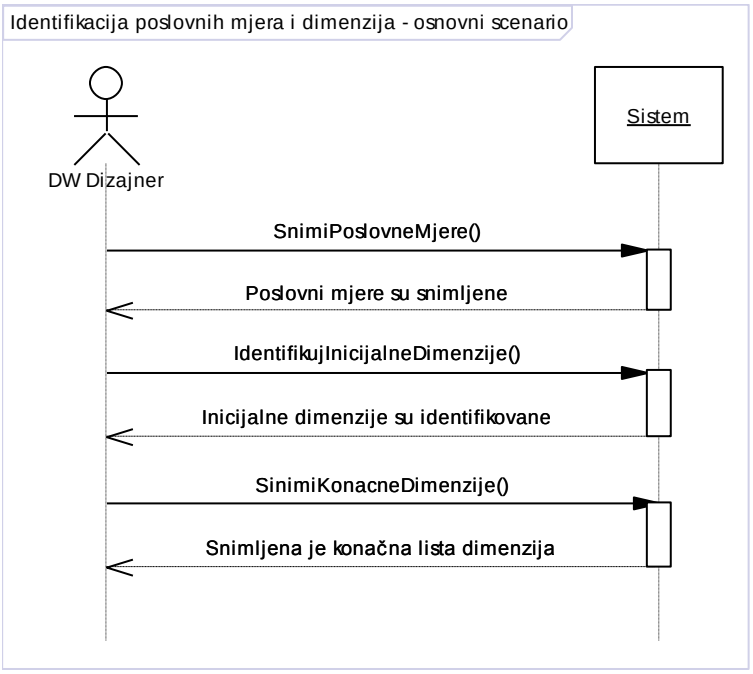


Slika B-31. Dijagram sekvenci za slučaj korišćenja SK - 2.2. Izbor skladišta podataka, alternativni scenario 3

DS - 2.3. Dijagram sekvenci za slučaj korišćenja SK - 2.3. Identifikacija poslovnih mjera i dimenzija

Osnovni scenario:

1. Dizajner sistemu prosleđuje listu poslovnih mjera (APSO)
2. Sistem vraća informaciju da su snimljene tabele mjere (IA)
3. Dizajner poziva sistem da identifikuje inicijalne dimenzije (APSO)
4. Sistem vraća listu identifikovanih inicijalnih dimenzija (IA)
5. Dizajner sistemu prosleđuje listu dimenzija koje želi da koristi (APSO)
6. Sistem vraća informaciju da je identifikovana konačna lista dimenzija (IA)



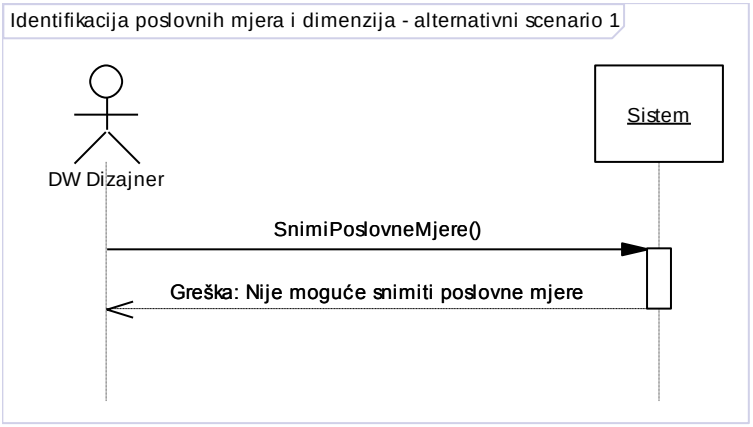
Slika B-32. Dijagram sekvenci za slučaj korišćenja SK - 2.3. Identifikacija poslovnih mjera i dimenzija, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- SnimiPoslovneMjere (Kolona, Vrijednost)
- IdentifikujInicijalneDimenzije (Pravilo)
- SnimiKonacneDimenzije (Kolona, Vrijednost)

Alternativni scenario 1:

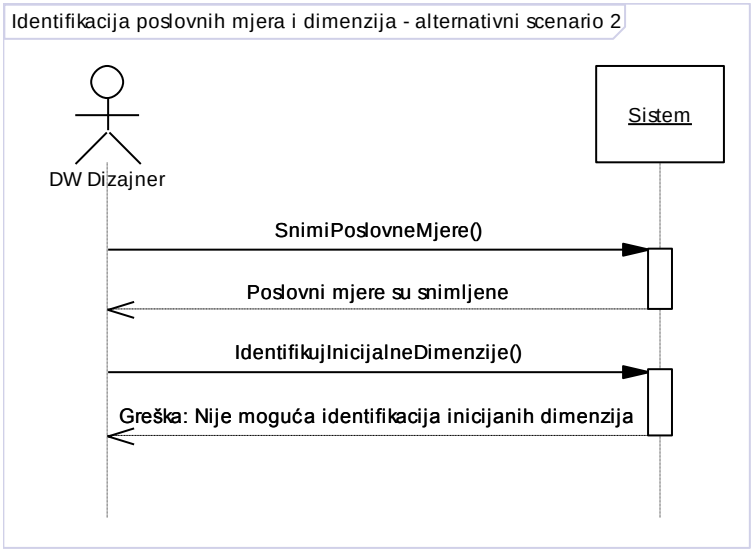
Ukoliko nije moguća identifikacija poslovnih mjera sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-33. Dijagram sekvenci za slučaj korišćenja SK - 2.3. Identifikacija poslovnih mjera i dimenzija, alternativni scenario 1

Alternativni scenario 2:

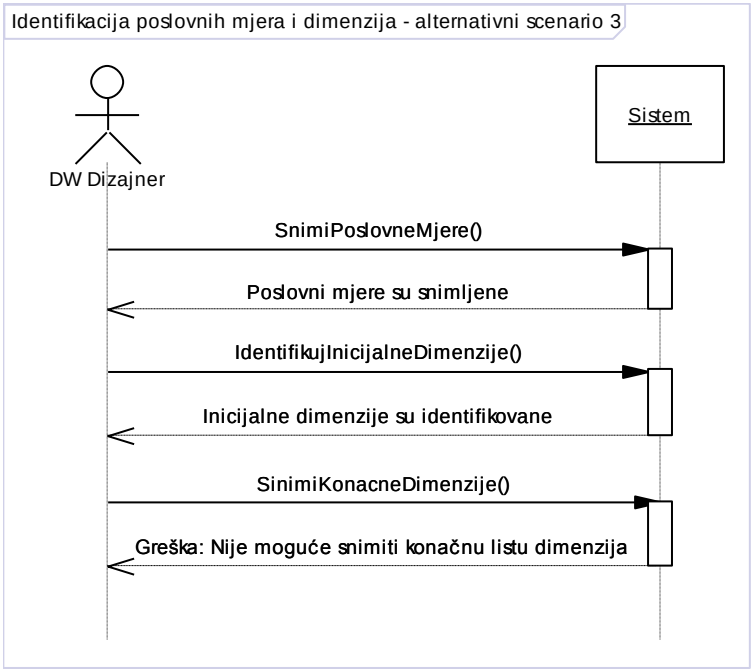
Ukoliko nije moguća identifikacija inicijalnih dimenzija sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-34. Dijagram sekvenci za slučaj korišćenja SK - 2.3. Identifikacija poslovnih mjera i dimenzija, alternativni scenario 2

Alternativni scenario 3:

Ukoliko nije moguća identifikacija konačne liste dimenzija, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



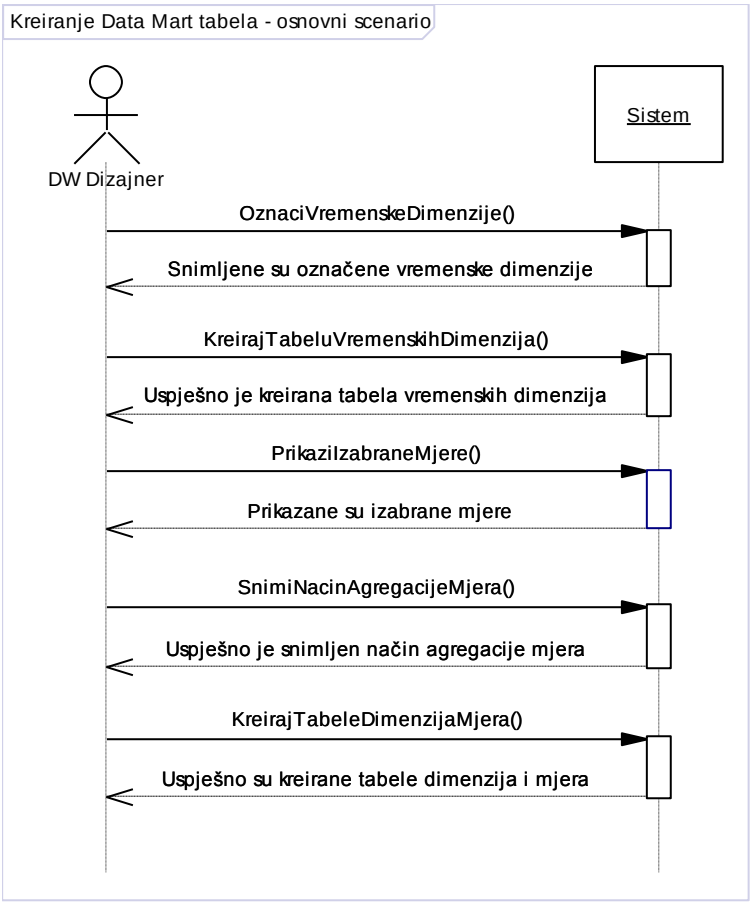
Slika B-35. Dijagram sekvenci za slučaj korišćenja SK - 2.3. Identifikacija poslovnih mjera i dimenzija, alternativni scenario 3

DS - 2.4. Dijagram sekvenci za slučaj korišćenja SK - 2.4. Kreiranje Data Mart tabela

Osnovni scenario:

1. Dizajner poziva sistem da snimi vremenske dimenzije (APSO)
2. Sistem vraća informaciju da su snimljene vremenske dimenzije (IA)
3. Dizajner pokreće proces kreiranja i popunjavanja tabele vremenskih dimenzija (APSO)
4. Sistem vraća informaciju da je kreirana i popunjena tabela vremenskih dimenzija (IA)
5. Dizajner poziva sistem da prikaže izabrane mjere (APSO)

6. Sistem prikazuje izabrane mjere (IA)
7. Dizajner poziva sistem da izvrši snimanje podataka u tabelu agregacija (APSO)
8. Sistem vraća informaciju da je popunjena pomoćna tabela agregacija (IA)
9. Dizajner pokreće proces kreiranja tabela dimenzija i mjera (APSO)
10. Sistem vraća informaciju da su kreirane tabele dimenzija i mjera (IA)



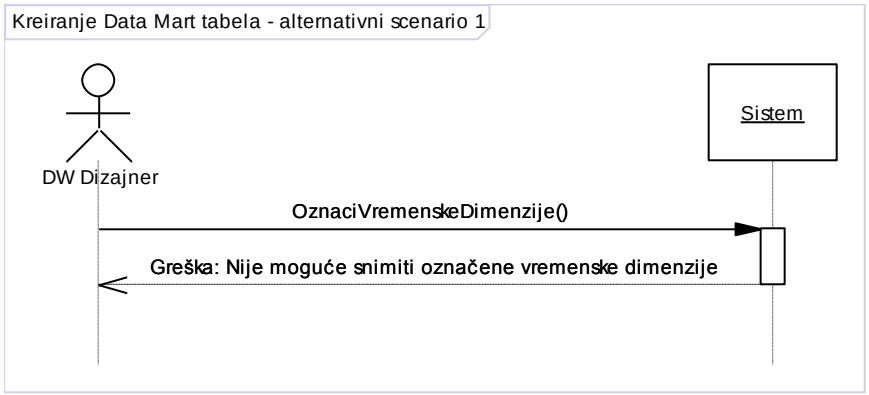
Slika B-36. Dijagram sekvenci za slučaj korišćenja SK - 2.4. Kreiranje Data Mart tabela, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- OznaciVremenskeDimenzije (Kolona, Vrijednost)
- KreirajTabeluVremenskihDimenzija (StartDate, EndDate, DataMart)

- PrikaziIzabraneMjere (DataMart)
- SnimiNacinAgregacijeMjera (Kolona,FactName, Formula)
- KreirajTabeleDimenzijaMjera(Pravilo, ImeTabeleMjera)

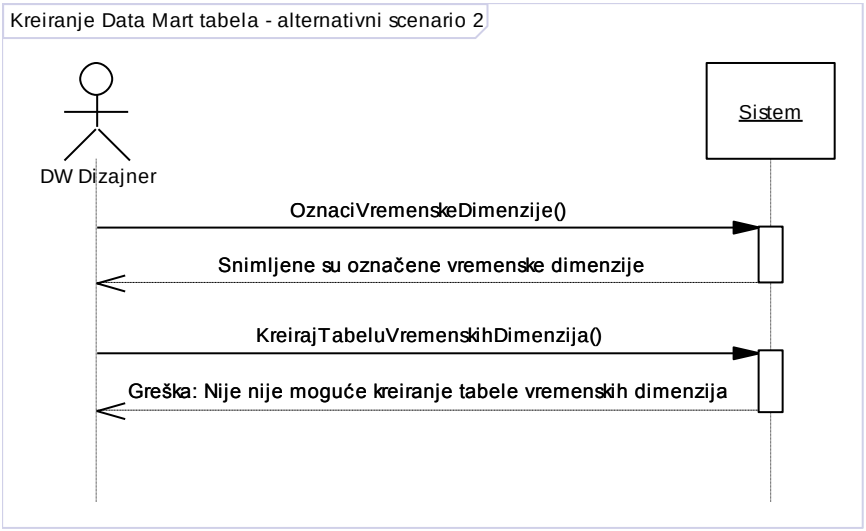
Alternativni scenario 1: Ukoliko nije moguće snimiti vremenske dimenzije, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-37. Dijagram sekvenci za slučaj korišćenja SK - 2.4. Kreiranje Data Mart tabela, alternativni scenario 1

Alternativni scenario 2:

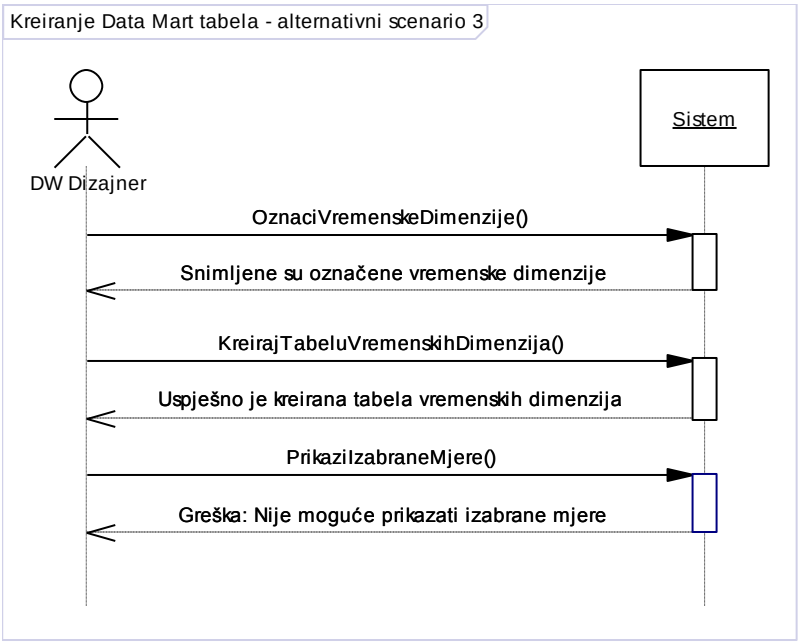
Ukoliko nije moguće kreiranje ili popunjavanje tabele vremenskih dimenzija, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-38. Dijagram sekvenci za slučaj korišćenja SK - 2.4. Kreiranje Data Mart tabela, alternativni scenario 2

Alternativni scenario 3:

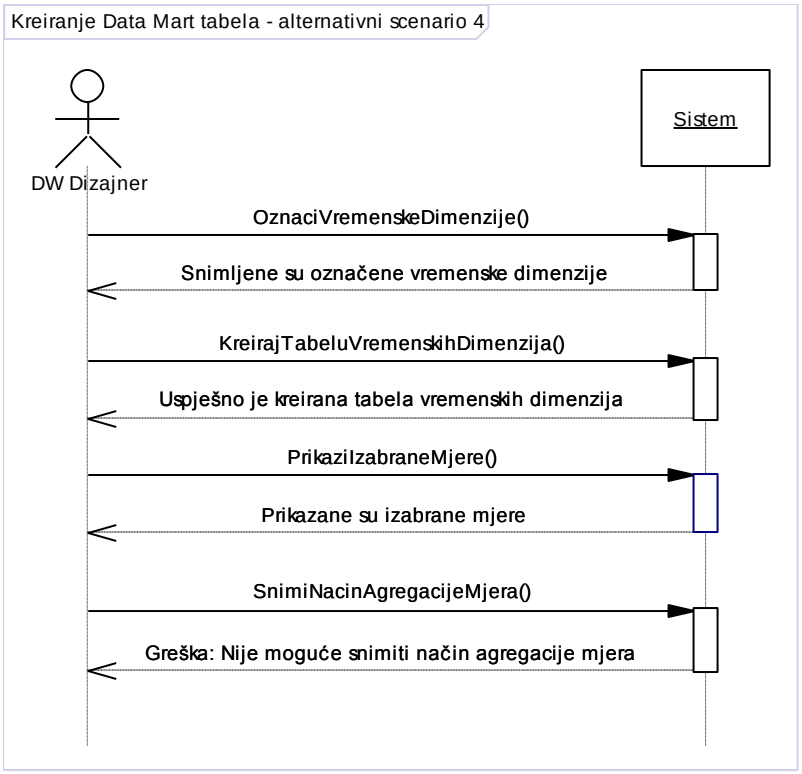
Ukoliko nije moguće prikazivanje izabranih mjera, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-39. Dijagram sekvenci za slučaj korišćenja SK - 2.4. Kreiranje Data Mart tabela, alternativni scenario 3

Alternativni scenario 4:

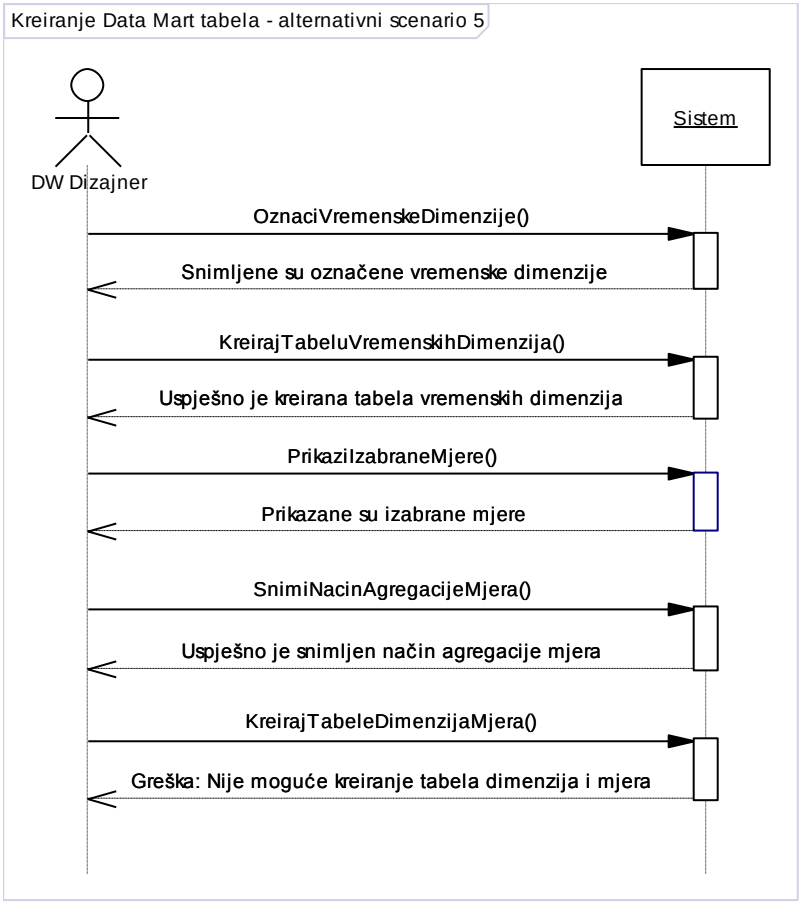
Ukoliko nije moguće popunjavanje tabela agregacija, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-40. Dijagram sekvenci za slučaj korišćenja SK - 2.4. Kreiranje Data Mart tabela, alternativni scenario 4

Alternativni scenario 5:

Ukoliko nije moguće kreiranje tabela dimenzija i mjera, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



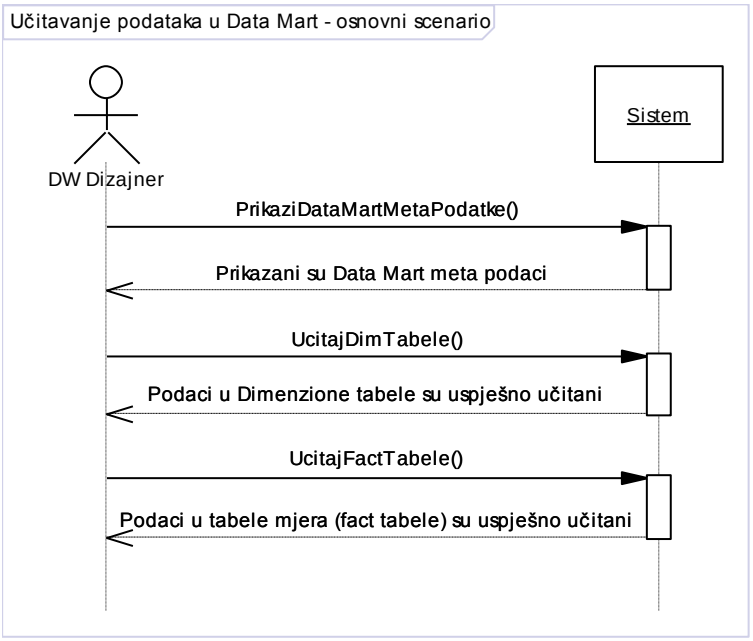
Slika B-41. Dijagram sekvenci za slučaj korišćenja SK - 2.4. Kreiranje Data Mart tabela, alternativni scenario 5

DS - 2.5. Dijagram sekvenci za slučaj korišćenja SK - 2.5. Učitavanje podataka u Data Mart

Osnovni scenario:

1. Dizajner poziva sistem da prikaže Data Mart metapodatke (APSO)
2. Sistem prikazuje Data Mart metapodatke (IA)
3. Dizajner pokreće proces ekstrakcije i učitavanja podataka u dimenzione tabele (APSO)
4. Sistem vraća informaciju da su učitani podaci u dimenzione tabele (IA)
5. Dizajner pokreće proces ekstrakcije i učitavanja podataka u tabele mjera (APSO)

6. Sistem vraća informaciju da su učitani podaci u tabele mjera (IA)



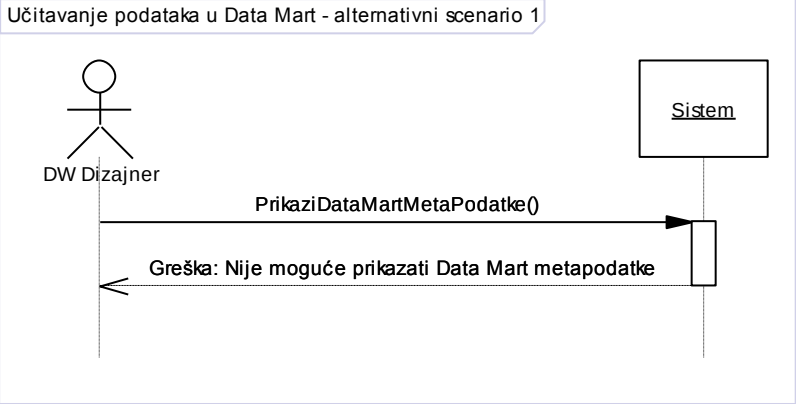
Slika B-42. Dijagram sekvenci za slučaj korišćenja SK - 2.5. Učitavanje podataka u Data Mart, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- PrikaziDataMartMetaPodatke (DataMart)
- UcitajDimTabele (Pravilo)
- UcitajFactTabele (Pravilo, FactTabela)

Alternativni scenario 1:

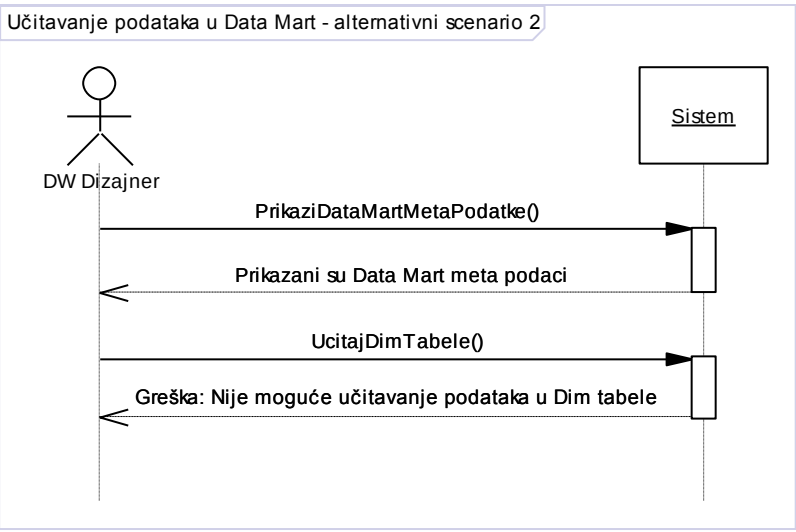
Ukoliko nije moguće prikazati Data Mart metapodatke, sistem vraća poruku o grešci, prekida se izvršenje scenarija



Slika B-43. Dijagram sekvenci za slučaj korišćenja SK - 2.5. Učitavanje podataka u Data Mart, alternativni scenario 1

Alternativni scenario 2:

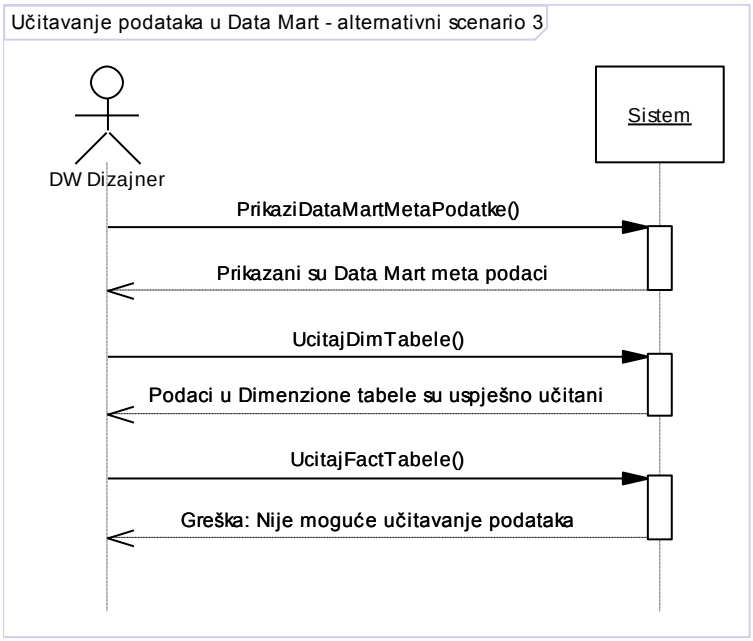
Ukoliko nije moguće učitavanje podataka u dimenzione tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-44. Dijagram sekvenci za slučaj korišćenja SK - 2.5. Učitavanje podataka u Data Mart, alternativni scenario 2

Alternativni scenario 3:

Ukoliko nije moguće učitavanje podataka u tabele mjera, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



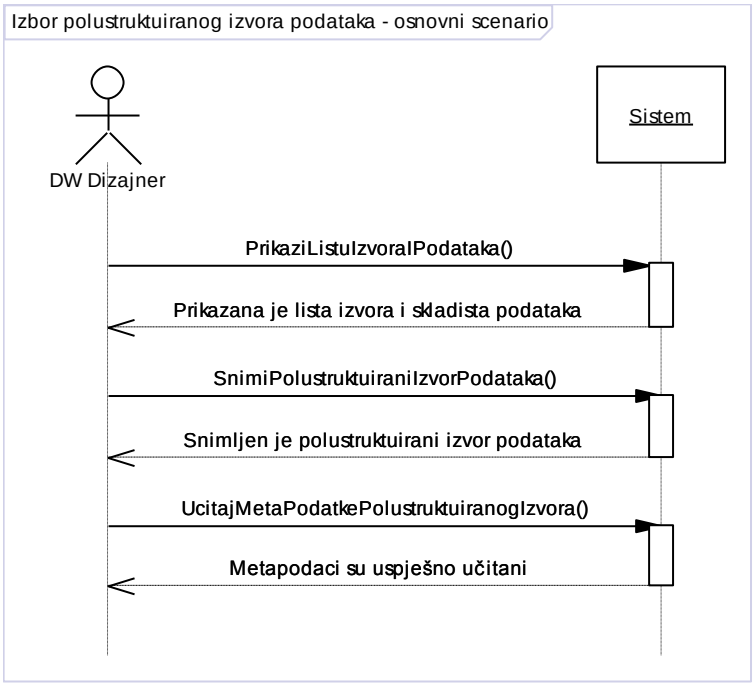
Slika B-45. Dijagram sekvenci za slučaj korišćenja SK - 2.5. Učitavanje podataka u Data Mart, alternativni scenario 3

DS - 3.1. Dijagram sekvenci za slučaj korišćenja SK - 3.1. Izbor polustruktuiranog izvora podataka za skladište podataka

Osnovni scenario:

1. Dizajner poziva sistem da prikaže listu skladišta i polustruktuiranih izvora podataka (APSO)
2. Sistem prikazuje listu dostupnih skladišta i polustruktuiranih izvora podataka podataka (IA)
3. Dizajner poziva sistem da na osnovu pravila snimi informaciju o polustruktuiranom izvoru podataka (APSO)
4. Sistem vraća poruku da su snimljeni polustruktuirani izvori podataka (IA)
5. Dizajner poziva sistem da izvrši učitavanje metapodataka iz polustruktuiranih izvora podataka (APSO)

6. Sistem vraća poruku da su učitani metapodaci polustruktuiranih izvora podataka (IA)



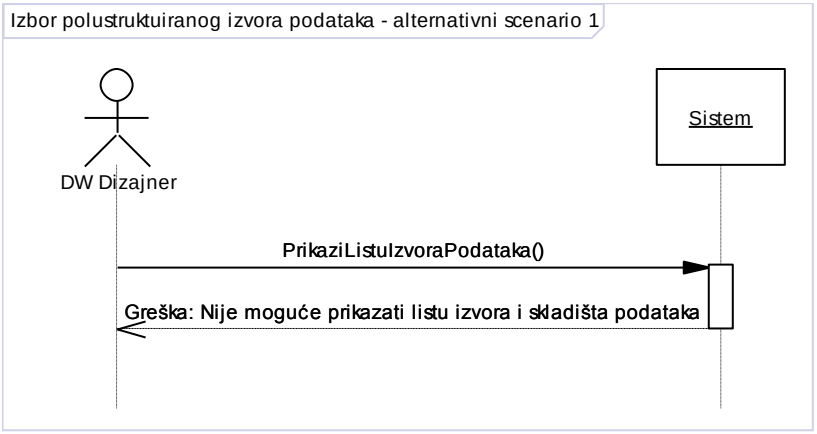
Slika B-46. Dijagram sekvenci za slučaj korišćenja SK - 3.1. Izbor polustruktuiranog izvora podataka za skladište podataka, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- PrikaziListuIzvoraPodataka (Server)
- SnimiPolustruktuiraniFajl (Pravilo)
- UcitajSSMetaPodatke (TipFajla, ImeFajla, PrelaznaBaza)

Alternativni scenario 1:

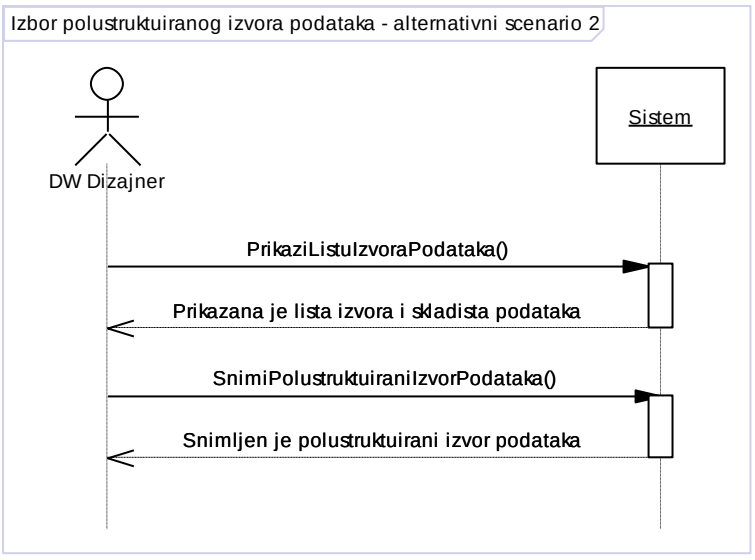
Ukoliko se ne mogu prikazati informacije o dostupnim skladištima i izvorima podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-47. Dijagram sekvenci za slučaj korišćenja SK- 3.1. Izbor polustruktuiranog izvora podataka za skladište podataka,alternativni scenario 1

Alternativni scenario 2:

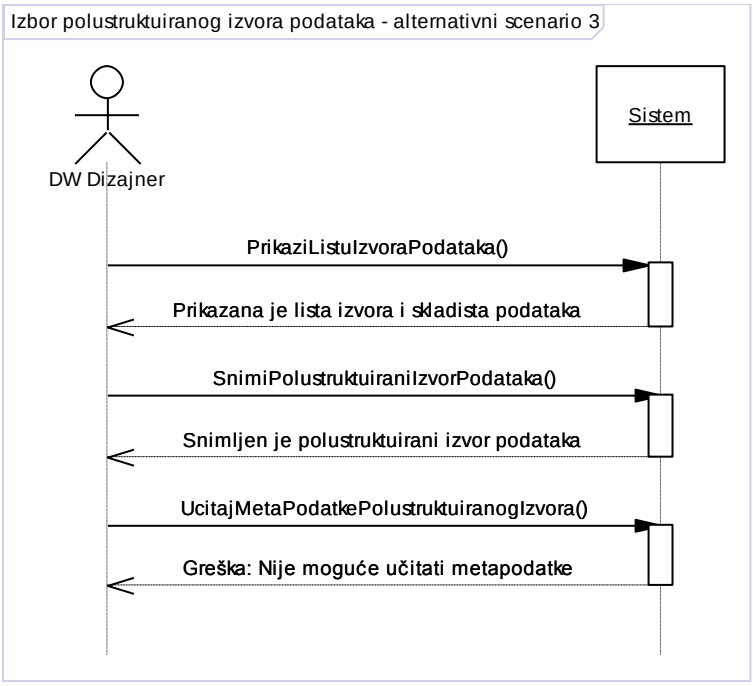
Ukoliko se ne mogu snimiti informacije o polustruktuiranim izvorima podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-48. Dijagram sekvenci za slučaj korišćenja SK- 3.1. Izbor polustruktuiranog izvora podataka za skladište podataka,alternativni scenario 2

Alternativni scenario 3:

Ukoliko se ne mogu učitati metapodaci polustruktuiranih izvora podataka, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

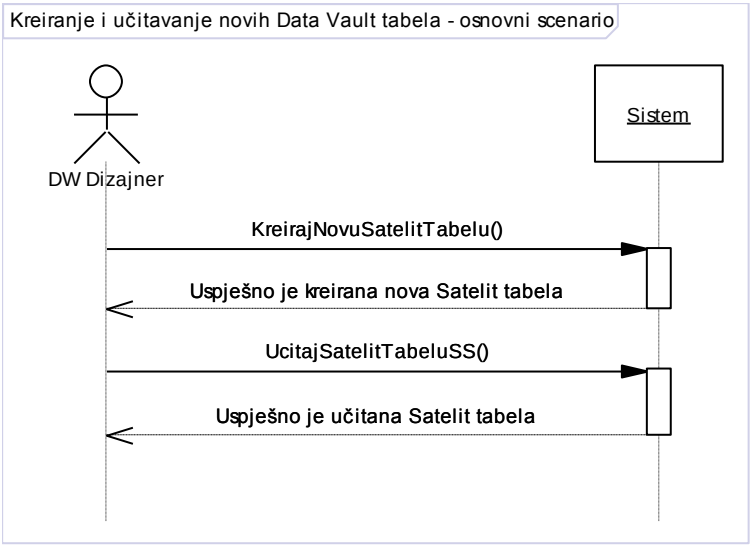


Slika B-49. Dijagram sekvenci za slučaj korišćenja SK- 3.1. Izbor polustruktuiranog izvora podataka za skladište podataka,alternativni scenario 3

DS - 3.2. Dijagram sekvenci za slučaj korišćenja SK - 3.2. Kreiranje i učitavanje novih Data Vault tabela

Osnovni scenario:

1. Dizajner poziva sistem da kreira novu Satelit tabelu (APSO)
2. Sistem vraća informaciju da je kreirana nova Satelit tabele (IA)
3. Dizajner poziva sistem da učita novu Satelit tabelu (APSO)
4. Sistem vraća informaciju da su učitani polustruktuirani podaci u novu Satelit tabelu (IA)



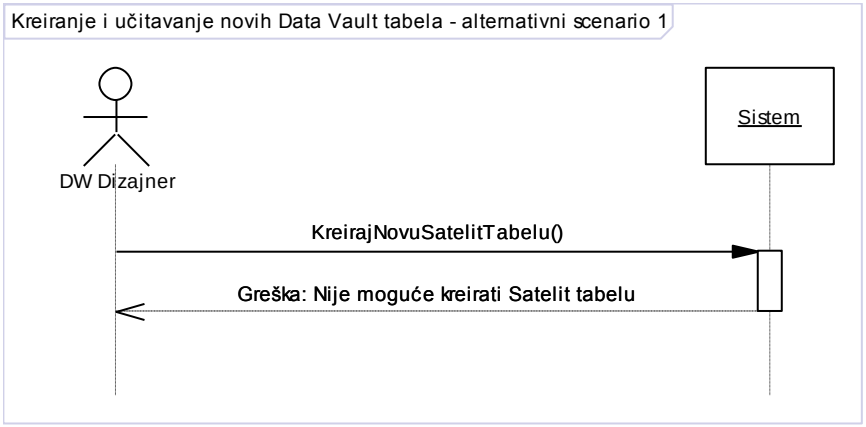
Slika B-50. Dijagram sekvenci za slučaj korišćenja SK - 3.2. Kreiranje i učitavanje novih Data Vault tabela, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- KreirajNovuSatelitTabelu (Pravilo, SkladistePodataka)
- UcitajSatelitTabeluSS (Pravilo, Fajl)

Alternativni scenario 1:

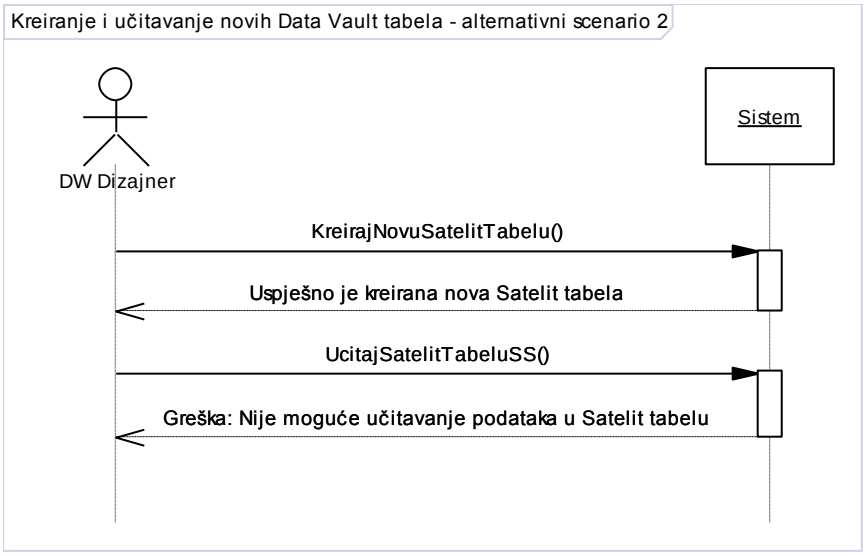
Ukoliko nije moguće kriratiu novu Satelit tabelu, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-51. Dijagram sekvenci za slučaj korišćenja SK - 3.2. Kreiranje i učitavanje novih Data Vault tabela, alternativni scenario 1

Alternativni scenario 2:

Ukoliko nije moguće učitati polustruktuirane podatke u novu Satelit tabelu, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija

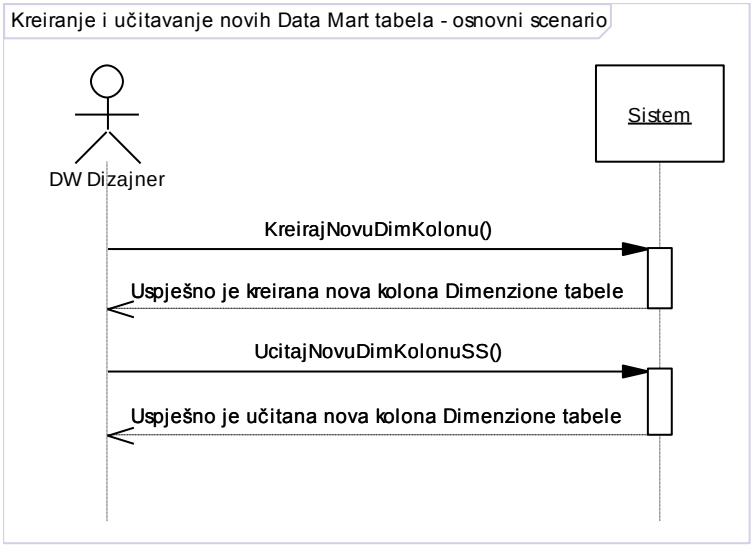


Slika B-52. Dijagram sekvenci za slučaj korišćenja SK - 3.2. Kreiranje i učitavanje novih Data Vault tabela, alternativni scenario 2

DS - 3.3. Dijagram sekvenci za slučaj korišćenja SK - 3.3. Kreiranje i učitavanje novih kolona Data Mart tabela

Osnovni scenario:

1. Dizajner poziva sistem da kreira novu kolonu izabrane dimenzione tabele (APSO)
2. Sistem vraća informaciju da je kreirana nova kolona dimenzione tabele (IA)
3. Dizajner poziva sistem da učita podatke u novu kolonu dimenzione tabele (APSO)
4. Sistem vraća informaciju da su učitani podaci u novu kolonu dimenzione tabele (IA)



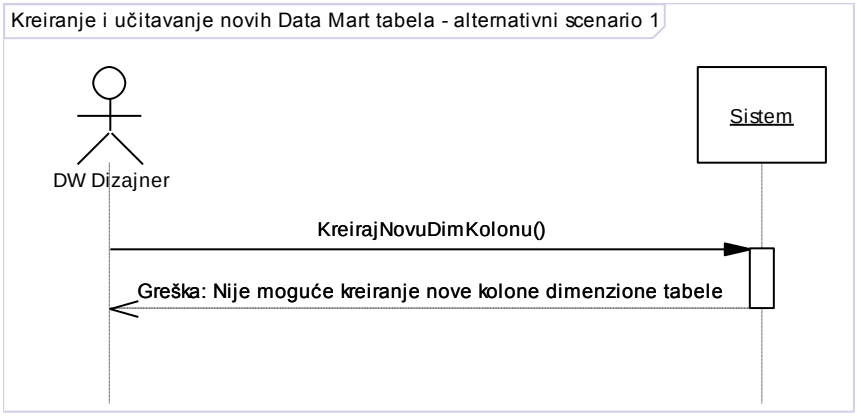
Slika B-53. Dijagram sekvenci za slučaj korišćenja SK - 3.3. Kreiranje i učitavanje novih kolona Data Mart tabela, osnovni scenario

Na osnovu dijagrama sekvenci identifikovane su sljedeće sistemske operacije:

- KreirajNovuDimKolonu (Pravilo, SatTabela, DimTabela)
- UcitajNovuDimKolonuSS (Pravilo, SatTabela, DimTabela)

Alternativni scenario 1:

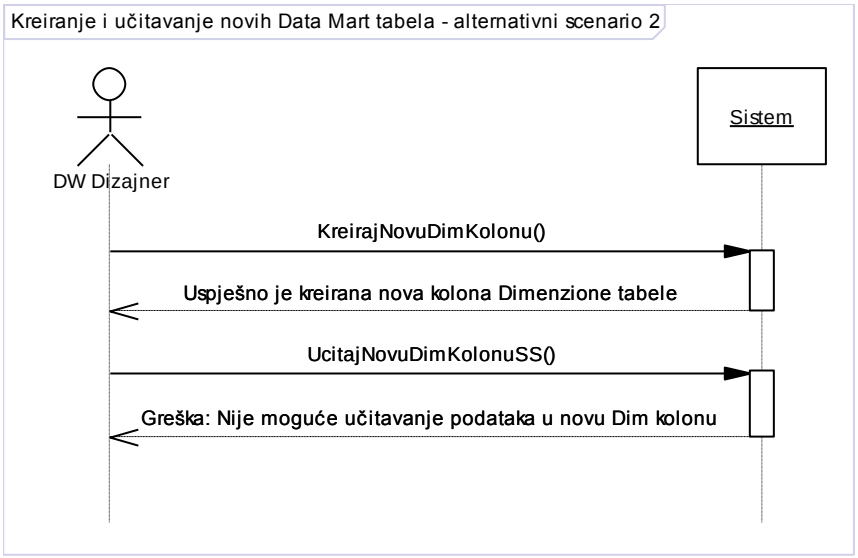
Ukoliko nije moguće kreirati novu kolonu izabrane dimenzione tabele, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-54. Dijagram sekvenci za slučaj korišćenja SK - 3.3. Kreiranje i učitavanje novih kolona Data Mart tabela, alternativni scenario 1

Alternativni scenario 2:

Ukoliko nije moguće učitati podatke u novu kolonu dimenzione tabelu, sistem vraća poruku o grešci (IA), prekida se izvršenje scenarija



Slika B-5589. Dijagram sekvenci za slučaj korišćenja SK - 3.3. Kreiranje i učitavanje novih kolona Data Mart tabela, alternativni scenario 2

Biografija autora

Dragoljub (Uroša) Krneta je rođen 06.01.1966. u selu Zebe opština Bosansko Grahovo, Bosna i Hercegovina. Osnovnu i srednju školu završio je u Bos. Grahovu kao nosilac diplome „Ognjen Prica“ (pandan diplome „Vuk Karadžić“ u Srbiji). Završio je Elektrotehnički fakultet, smjer računari i automatika, prvi stepen, na Univerzitetu u Banja Luci 1994. godine. Tehnički fakultet „Mihajlo Pupin“ u Zrenjaninu na Univerzitetu u Novom Sadu završava 2006. godine, a na istom fakultetu 2008. godine završava postdiplomske master studije sa prosečnom ocenom 9,23 i stiče zvanje Diplomirani inženjer poslovne informatike – master. Master rad u oblasti poslovne inteligencije pod mentorstvom prof. dr Dragice Radosav odbranio je sa ocenom 10. Autor ili koautor je dvanaest naučnih i stručnih radova iz oblasti razvoja softvera, baza podataka, data warehouse i sistema poslovne inteligencije, od kojih je najznačajniji rad iz oblasti koja je predmet doktorske disertacije objavljen u časopisu sa SCI liste: Krneta D., Jovanović V., Marjanović Z., "A Direct Approach to Physical Data Vault Design", Computer Science and Information Systems (ComSIS), Vol. 11, No. 2, (2014). ISSN: 1820-0214. IF (2013): 0.575

Programiranjem, razvojem softvera i bazama podataka bavi se od 1989. god. Do 2002. god. autor je dvadesetak različitih softverskih poslovnih aplikacija koje su implementirane u više od 50 preduzeća. Sredinom 2002. god. dolazi u kompaniju Lanaco IT Banja Luka na poziciju Menadžera sektora za razvoj softvera. Za vrijeme rada u Lanacu, aktivno je učestvovao svim značajnijim softverskim projektima bilo kao projekt menadžer, projektant informacionog sistema i baze podataka, sistem analitičar ili programer. Posjeduje razne sertifikate u informacionim tehnologijama od kojih su najznačajniji: Microsoft Certified Professional (MCP) i Microsoft Certified Database Administrator (MCDBA). Učesnik je mnogih međunarodnih IT konferencija od koji su najznačajnije Microsoft Worldwide Partner Conference u Denveru 2007. i u Washingtonu 2010. godine. Član je asocijacija Institute of Electrical and Electronics Engineers (IEEE) i The Data Warehousing Institute (TDWI).

Kandidat Dragoljub Krneta je upisao doktorske studije na Fakultetu organizacionih nauka Univerziteta u Beogradu školske 2009/10. Pristupni rad za doktorsku disertaciju je odbranio u julu mjesecu 2012. godine.

Prilog 1.

Izjava o autorstvu

Potpisani Dragoljub Krneta

broj indeksa: 3/2009

Izjavljujem

da je doktorska disertacija pod naslovom

Automatizacija fizičkog projektovanja skladišta podataka proširenog Data Vault pristupa

- rezultat sopstvenog istraživačkog rada,
- da predložena disertacija u celini ni u delovima nije bila predložena za dobijanje bilo koje diplome prema studijskim programima drugih visokoškolskih ustanova,
- da su rezultati korektno navedeni i
- da nisam kršio/la autorska prava i koristio intelektualnu svojinu drugih lica.

U Beogradu, 31.10.2014.

Potpis doktoranda

Prilog 2.

Izjava o istovetnosti štampane i elektronske verzije doktorskog rada

Ime i prezime autora: Dragoljub Krneta

Broj indeksa: 3/2009

Studijski program: Informacioni sistemi

Naslov rada: **Automatizacija fizičkog projektovanja skladišta podataka proširenog Data Vault pristupa**

Mentor: Prof. dr Zoran Marjanović

Potpisani Dragoljub Krneta

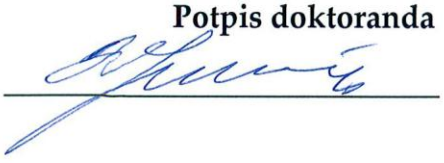
Izjavljujem da je štampana verzija mog doktorskog rada istovetna elektronskoj verziji koju sam predao za objavljivanje na portalu **Digitalnog repozitorijuma Univerziteta u Beogradu**.

Dozvoljavam da se objave moji lični podaci vezani za dobijanje akademskog zvanja doktora nauka, kao što su ime i prezime, godina i mesto rođenja i datum odbrane rada.

Ovi lični podaci mogu se objaviti na mrežnim stranicama digitalne biblioteke, u elektronskom katalogu i u publikacijama Univerziteta u Beogradu.

Potpis doktoranda

U Beogradu, 31.10.2014.



Prilog 3.

Izjava o korišćenju

Ovlašćujem Univerzitetsku biblioteku „Svetozar Marković" da u Digitalni repozitorijum Univerziteta u Beogradu unese moju doktorsku disertaciju pod naslovom:

Automatizacija fizičkog projektovanja skladišta podataka proširenog Data Vault pristupa

koja je moje autorsko delo.

Disertaciju sa svim prilozima predao sam u elektronskom formatu pogodnom za trajno arhiviranje.

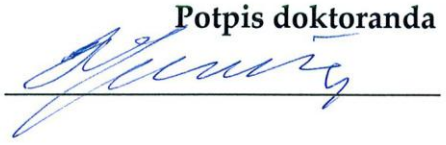
Moju doktorsku disertaciju pohranjenu u Digitalni repozitorijum Univerziteta u Beogradu mogu da koriste svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice (Creative Commons) za koju sam se odlučio/la.

1. Autorstvo
2. Autorstvo - nekomercijalno
3. **Autorstvo – nekomercijalno – bez prerade**
4. Autorstvo – nekomercijalno – deliti pod istim uslovima
5. Autorstvo – bez prerade
6. Autorstvo – deliti pod istim uslovima

(Molimo da zaokružite samo jednu od šest ponuđenih licenci, kratak opis licenci dat je na poleđini lista)

Potpis doktoranda

U Beogradu, 31.10.2014.



Kratak opis licenci

1. Autorstvo - Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence, čak i u komercijalne svrhe. Ovo je najslobodnija od svih licenci.

2. Autorstvo – nekomercijalno. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca ne dozvoljava komercijalnu upotrebu dela.

3. Autorstvo - nekomercijalno – bez prerade. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, bez promena, preoblikovanja ili upotrebe dela u svom delu, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca ne dozvoljava komercijalnu upotrebu dela. U odnosu na sve ostale licence, ovom licencom se ograničava najveći obim prava korišćenja dela.

4. Autorstvo - nekomercijalno – deliti pod istim uslovima. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence i ako se prerada distribuira pod istom ili sličnom licencom. Ova licenca ne dozvoljava komercijalnu upotrebu dela i prerada.

5. Autorstvo – bez prerade. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, bez promena, preoblikovanja ili upotrebe dela u svom delu, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca dozvoljava komercijalnu upotrebu dela.

6. Autorstvo - deliti pod istim uslovima. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence i ako se prerada distribuira pod istom ili sličnom licencom. Ova licenca dozvoljava komercijalnu upotrebu dela i prerada. Slična je softverskim licencama, odnosno licencama otvorenog koda.