UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

DIPLOMSKO DELO

PRIMERJALNA ANALIZA PODATKOVNIH MODELOV PODATKOVNIH SKLADIŠČ

Ljubljana, september 2016 DAVID ŽNIDARŠIČ

IZJAVA O AVTORSTVU

Podpisani David Žnidaršič, študent Ekonomske fakultete Univerze v Ljubljani, avtor predloženega dela z naslovom Primerjava podatkovnih modelov podatkovnih skladišč, pripravljenega v sodelovanju s svetovalcem red. prof. dr. Jurijem Jakličem

IZJAVLJAM

1. da sem predloženo delo pripravil samostojno;

2. da je tiskana oblika predloženega dela istovetna njegovi elektronski obliki;

3. da je besedilo predloženega dela jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem poskrbel, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam oziroma navajam v besedilu, citirana oziroma povzeta v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani;

4. da se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku Republike Slovenije;

5. da se zavedam posledic, ki bi jih na osnovi predloženega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom;

6. da sem pridobil vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v predloženem delu in jih v njem jasno označil;

7. da sem pri pripravi predloženega dela ravnal v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil soglasje etične komisije;

8. da soglašam, da se elektronska oblika predloženega dela uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;

9. da na Univerzo v Ljubljani neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve predloženega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja predloženega dela na voljo javnosti na svetovnem spletu preko Repozitorija Univerze v Ljubljani;

10. da hkrati z objavo predloženega dela dovoljujem objavo svojih osebnih podatkov, ki so navedeni v njem in v tej izjavi.

V Ljubljani, dne 29.9.2016 Podpis študenta:

KAZALO

UVOD 1

1 PODATKOVNO SKLADIŠČE IN PODATKOVNI MODELI 2
1.1 Izzivi pri tradicionalnih podatkovnih skladiščih 5
1.2 Dimenzijski model 6
1.3 Normalizirano podatkovno skladišče po Inmonu 6
1.3.1 Ravni podatkovnega skladišča 7
1.3.2 Redundanca podatkov 8
1.3.3 Granulacija in particioniranje 8
1.3.4 Gradniki modela vozlišča in priključkov 9
1.4 Model podatkovni trezor 11
1.5 Sidrni model 12

2 PRIMERJALNA PODROČJA 13
2.1 Projektni pristop in metodologija razvoja 14
2.2 Beleženje zgodovine 18
2.3 Fleksibilnost in stabilnost pri dopolnitvah 19
2.4 Kompleksnost modela in stroškovni vidik razvoja 20
2.5 Polnjenje, čiščenje, integracija in transformacija podatkov ter njihova hitrost 20
2.6 Odzivnost pri analizah končnih uporabnikov 20
2.7 Nestrukturirani podatki 21

3 DIMENZIJSKI MODEL 21
3.1 Projektni pristop in metodologija razvoja 21
3.2 Polnjenje, čiščenje, integracija in transformacija podatkov ter njihova hitrost 21
3.3 Nestrukturirani podatki 21
3.4 Beleženje zgodovine 22
3.5 Fleksibilnost in stabilnost pri dopolnitvah 22
3.6 Odzivnost pri analizah končnih uporabnikov 22
3.7 Kompleksnost modela in stroškovni vidik razvoja 23

4 MODEL VOZLIŠČA IN PRIKLJUČKOV 23
4.1 Projektni pristop in metodologija razvoja 23
4.2 Nestrukturirani podatki 25
4.3 Polnjenje, čiščenje, integracija in transformacija podatkov ter njihova hitrost 25
4.4 Beleženje zgodovine 25

4.5 Fleksibilnost in stabilnost pri dopolnitvah....25
4.6 Odzivnost pri analizah končnih uporabnikov....26
4.7 Kompleksnost modela in stroškovni vidik razvoja....26

5 PODATKOVNI TREZOR....26
5.1 Projektni pristop in razvojna metodologija....26
5.2 Beleženje zgodovine, hitrost in način ETL polnjenja....27
5.3 Fleksibilnost in stabilnost pri dopolnitvah....27
5.4 Nestrukturirani podatki....28
5.5 Virtualizirane strukture in odzivnost analiz končnih uporabnikov....28
5.6 Kompleksnost modela in stroškovni vidik razvoja....31

6 SIDRNI MODEL....31
6.1 Projektni pristop in razvojna metodologija....31
6.2 Kompleksnost modela....32
6.3 Beleženje zgodovine, hitrost in način ETL polnjenja....32
6.4 Fleksibilnost in stabilnost pri dopolnitvah....32
6.5 Nestrukturirani podatki....33
6.6 Odzivnost pri analizah končnih uporabnikov....33

7 PRIMERJAVA PODATKOVNIH MODELOV....33

SKLEP....37

LITERATURA IN VIRI....39

KAZALO TABEL

Tabela 1: Entiteta kupec....11
Tabela 2: Satelit naslov kupca....12
Tabela 3: Povezava zavarovalna polica kupca....12
Tabela 4: Primerjava modelov po kriterijih....34

KAZALO SLIK

Slika 1: Nivoji arhitekture....8
Slika 2: Vloge na zavarovalni polici....10
Slika 3: Faze kaskadnega modela....16

Slika 4: Spiralni razvojni model 17
Slika 5: Primerjava kaskadnega in spiralnega razvojnega cikla 24
Slika 6: Večtirna arhitektura z uporabo tehnik iz podatkovnega trezorja in dimenzionalnega modela 30

UVOD

Narava vseh podatkovnih skladišč je, da se spreminjajo zaradi rednega vzdrževanja in dopolnjevanja z novimi poslovnimi zahtevami, kot potrjujeta tudi Inmon (2002) in Kimball & Ross (2013). V nasprotnem primeru namreč velja, da je bil projekt vpeljave podatkovnega skladišča neuspešen, saj ga uporabniki niso sprejeli kot primarni vir informacij in zato nimajo želje po nadgrajevanju. Zadovoljstvo udeležencev pa je eno od ključnih sodil, ki jih Atkinson (1999) omenja pri presojevanju uspešnosti projekta z vidika deležnikov. Obenem pa tako Kimbal (2013), kot Inmon (2002) poudarjata, da morajo podatkovna skladišča zagotavljati stabilen in enovit vir informacij, na podlagi katerega se sprejemajo strateške odločitve in izvaja statutarno poročanje. To pa je do določene mere lahko kontradiktorno zgornji trditvi glede osnovne narave, da se podatkovna skladišča stalno razvijajo, in lahko vpliva tudi na zgodovinsko stanje podatkov. Za razrešitev tega paradoksa so se pojavili številni modeli podatkovnih skladišč in njihovi hibridi, ki bolj ali manj uspešno poskušajo razrešiti problem. Poleg navedenega je treba upoštevati še mnogo konceptualnih, tehničnih in metodoloških razlik med posameznimi podatkovnimi modeli, zato je izbira najprimernejšega modela zelo zahtevna, v kolikor je najprimernejši model dejansko sploh mogoče izbrati še pred začetkom projekta. Tradicionalni dilemi med Kimballovim dimenzionalnim modelom podatkovnega skladišča kot zbira povezanih analitičnih področij, povezanih v podatkovno skladišče, in Inmonovim večnivojskim celovitim podatkovnim skladiščem se z modernejšimi modeli in njihovimi hibridi pridružujejo še dodatne dileme. Primera modernejših modelov, ki ju bom poleg tradicionalnih primerjal v nalogi, sta še sidrni model, ki poskuša minimizirati vpliv verzioniranja podatkov na potrebe po velikosti diska in s tem pohitriti izvajanje poizvedb, ter podatkovni trezor, ki poskuša z opustitvijo transformacije in čiščenja podatkov pred polnjenjem ohraniti sledljivost podatkov ter z opustitvijo referenčnih integritet integrirati tudi nestrukturirane podatke. Vsak od modelov ima svoje prednosti in slabosti, zato je izbira prave arhitekture težka, saj se vsaj nekaj let verjetno ne bo menjala, razen če bi se projekt uvedbe podatkovnega izkazal za neuspešnega, čemur pa bi se z ustrezno prvotno izbiro lahko izognili.

Namen diplomskega dela je prispevati k razumevanju razlik med podatkovnimi modeli, ki se uporabljajo v podatkovnih skladiščih, in olajšati izbiro med njimi. Zato je prvi cilj opredelitev kriterijev, na podlagi katerih bi bilo mogoče objektivno in uporabno primerjati različne modele podatkovnih skladišč. Nato pa bom na podlagi teh kriterijev in značilnosti posameznih modelov analiziral primernost modelov podatkovnega skladišča za različne situacije.

Prvi del naloge bo temeljil na teoretični predstavitvi pojma podatkovnega skladišča, pri čemer bom uporabil tujo in domačo literaturo. Že v tem delu bom predstavil primerjavo med konceptoma obeh »očetov« tradicionalnih modelov podatkovnega skladišča, Ralpha Kimballa in Billa Inmona:

- koncept celovito normalizirano podatkovno skladišče (angl. *Information factory*), ki ga je postavil Bill Inmon, in
- koncept dimenzijskega podatkovnega modela, ki ga je postavil Ralph Kimball.

Drugi del naloge bo temeljil na evoluciji modelov podatkovnih skladišč in opredelitvi pogosto uporabljenih modernejših modelov in njihovih hibridov, ki se bo navezoval predvsem na tujo literaturo o izbranih modelih. Sodobni modeli, ki bodo podrobneje opisani, so :

- podatkovni trezor (angl. *Data Vault*), katerega avtor je Dan Linstedt,
- sidrni model (angl. *Anchor model*) – avtor Lars Rönnbäck.

V drugem delu bom naredil primerjavo teh modelov na podlagi strokovnih člankov in lastnih izkušenj na projektih podatkovnih skladišč.

1 PODATKOVNO SKLADIŠČE IN PODATKOVNI MODELI

Inmon (2002, str. 31) pravi, da je podatkovno skladišče »… tematsko usmerjena, integrirana, nespremenljiva in zgodovinsko beležena zbirka podatkov za podporo managerskim odločitvam.«

Tematska usmerjenost pomeni, da so podatki organizirani po glavnih področjih, ki so v primeru zavarovalnice lahko: polica, vloga osebe, predmet, premija, škoda (Inmon, 2002, str. 33).

Integriranost pomeni, da so v podatkovnem skladišču podatki iz različnih podatkovnih virov poenoteni z istimi šiframi. Primer šifranta spola: v enem podatkovnem viru je spol šifriran z M za moški in Z za ženski, v drugem pa X za moški in Y za ženski. Pri polnjenju podatkov v podatkovno skladišče iz vseh virov pa spol osebe poenotimo in transformiramo v enoten šifrant, ki vse šifre transformira v 1 za moški in 0 za ženski. Poenotenje zelo olajša analize končnih uporabnikov, saj izbirajo samo med dvema vrednostma (Inmon, 2002, str. 33).

Nespremenljivost pomeni, da ko so enkrat napolnjeni v podatkovno skladišče, se ne posodabljajo in zagotavljajo, da lahko vedno izvedemo isto analizo za nazaj in vedno dobimo isti rezultat (Inmon, 2002, str. 33–34). To je še posebej pomembno pri statutarnem poročanju nadzornim institucijam, še posebej v finančnem sektorju. Tako so podatki vedno shranjeni kot stanje na dan. Na ta način podatkovno skladišče zagotavlja zgodovinsko verzioniranje.

Če se v izvornih aplikacijah podatki spremenijo, se v podatkovno skladišče polnijo kot nova verzija podatkov, ki velja od določenega dne naprej. Na ta način pa podatkovno skladišče zagotavlja zgodovinsko beleženje sprememb. Časovna komponenta v podatkovnem skladišču mora biti vedno prisotna in zato vsebuje veliko večji časovni horizont dogodkov

in verzij podatkov, kot je na voljo pri poročanju na operativnem nivoju (Inmon, 2002, str. 34–35).
Ralph Kimball (2013, str. 3) pa opredeljuje podatkovno skladišče s spodaj naštetimi zahtevami.

- Sistem mora zagotavljati enostaven dostop do informacij.
- Informacije v sistemu morajo biti predstavljane konsistentno.
- Sistem mora biti prilagodljiv spremembam.
- Sistem mora zagotavljati podatke v pričakovanih časovnih okvirih.
- Sistem mora biti varna trdnjava, ki varuje informacije.
- Sistem mora služiti kot avtoritetni in verodostojni temelj za boljše sprejemanje odločitev.
- Poslovni uporabniki morajo podatkovno skladišče sprejeti, da lahko rečemo, da je uspešno.

Izhodiščna razlika pri razumevanju namena in uporabe podatkovnega skladišča med Inmonom in Kimballom je, da Inmon meni, da je ciljna skupina uporabnikov še vedno nekako IT oziroma napredni poslovni analitiki, ki sami izvajajo zahtevne analize in do določene mere posedujejo tudi tehnične veščine. Zato priporoča, da se določene ključne vsebine pripravijo vnaprej in se kasneje glede na ugotovitve in sledeče dodatne potrebe vsebina nadgrajuje kot pristop od spodaj navzgor (angl. *bottom-up*). Kimballova predvidena ciljna skupina pa so končni uporabniki, ki želijo hitro dostopati do definiranih že pripravljenih enostavnih podatkovnih rešitev. Zato priporoča, da se že na začetku naredi podrobna analiza uporabniških potreb in se glede na usklajene zahteve izvede izgradnja kot pristop od zgoraj navzdol (angl. *top-down*).

Glede na moje izkušnje sta trditvi komplementarni, ker je ustrezna izbira pristopa zelo odvisna tudi od tipa uporabnikov oziroma njihove zrelosti za podatkovno skladišče. Na podlagi izkušenj na projektih podatkovnega skladišča sem naletel na 3 glavne tipe uporabnikov. Prva, največja skupina so uporabniki statičnih periodičnih poročil, ki se lahko prilagajajo le z vnaprej pripravljenimi filtrirnimi polji, med katerimi je sicer najbolj uporabljeno filtriranje po datumski dimenziji z vidika datuma knjiženja. Druga skupina uporabnikov so analitiki, ki na podlagi izdelanega dimenzijskega modela na zahtevo in vprašanja managementa sami zaganjajo *ad hoc* poizvedbe in rezultate analizirajo z različnih vidikov. Tretja skupina pa so napredni poslovni analitiki, kontrolerji, ki samostojno raziskujejo podatke in poskušajo globlje razumeti vzroke za odmike od pričakovanih rezultatov, z namenom izboljšanja poslovanja in odpravljanja vzrokov v prihodnje ali se vsaj pripraviti nanje. Ta skupina je navadno manjša, vendar je njihov doprinos k izboljšanju poslovanja podjetja z vidika vplivanja na strateške odločitve največji. Zato pa imajo vedno nove potrebe po dodatnih informacijah, ki bi jih želeli podrobneje analizirati.

Prvi dve skupini uporabnikov zato lahko dokaj natančno določita zahteve, na podlagi katerih se lahko zgradi podatkovno skladišče po kaskadni (angl. *Waterfall*) razvojni metodi[1] in na način, da je poslovnemu uporabniku prijazno, kot opisuje Kimball. Skupina naprednih analitikov pa ne more že na začetku določiti vseh končnih zahtev, saj te nastajajo po analizi rezultatov. Še zlasti se ustvarijo apetiti po novih informacijah v prvem letu po vzpostavitvi ogrodja in glavnih vsebin podatkovnega skladišča. To pa pogosto povzroči težavo z razpoložljivostjo resursov na projektu, saj je potrebno redno usklajevati prioritete med zahtevami iz prvotnega plana implementacije področij in z novimi zahtevami, ki prihajajo od poslovnih uporabnikov, ki uporabljajo že razvita področja podatkovnega skladišča. Zato je za njihov način dela bolj primeren pristop »spiralne« metodologije, kot jo opisuje Inmon in jo bom opisal v nadaljevanju.

V moji poklicni poti sem tudi sam najprej začel z razvojem področnih dimenzionalnih modelov podatkovnih skladišč. Razlog je predvsem, da so bila enostavna tako z razvojnega vidika kot enostavna za razumevanje končnih uporabnikov, pri čemer je fleksibilnost pogledov pri uporabniških analizah velika, odzivnost poizvedb pa hitra, kar omenja tudi Jaklič (2002, str. 20).

Z izkušnjami, pridobljenimi z implementacijo po posameznih poslovnih področjih, smo z ekipo najboljše prakse iz raznih teoretičnih predlogov nadgradili in pripravili določene preddefinirane podatkovne modele ter jih uporabili kot osnovo za naslednje projekte. Princip izgradnje se je zato iz Kimballovega pristopa od zgoraj navzdol približal Inmonovemu pristopu od spodaj navzgor, kar bi lahko poimenovali srečujoči pristop, saj upošteva oba vidika. Faza zbiranja poslovnih zahtev je zato temeljila bolj na predstavitvi preddefiniranih modelov podatkovnih področij na podatkih naročnika in predlogov dodatnih razširitev ugotovljenih z analizo vrzeli z uporabniki. Tako so tudi naprednejši uporabniki že tekom prve iteracije dobili odgovore tudi na zahtevnejša vprašanja, na katera pred analizo še niso pomislili.

Vendar sem kasneje, zlasti pri večjih naročnikih, naletel tudi na določene izzive, ki jih dimenzionalni model težko samodejno oziroma enostavno rešuje, če se dosledno upošteva osnovna vodila dimenzijskega modela, kot je npr. najnižji nivo granularnosti. Pogost izziv je bilo statutarno poročanje in zagotavljanje reproduciranja starega poročila z enakimi vrednostnimi rezultati, čeprav so se kasneje določeni vključeni podatki lahko spremenili, izboljšali, dodatno poknjižili ali celo izbrisali. To se je reševalo predvsem z dvema konceptoma – posnetek stanja in letni kumulativni podatki z zaklepanjem obdobja. Pri obeh konceptih pa je osnovna predpostavka, da se struktura in granulacija osnovne tabele dejstev ne bo bistveno spreminjala, saj je nabor dimenzij omejen na bistvene strateške poglede.

[1] Kaskadna metoda je poimenovana tudi kot slapovna ali zaporedna razvojna metoda.

Pri posnetku stanja je vodilna ideja predvsem, da se po zaključku poslovnega obdobja, ki je lahko mesec, kvartal ali leto, podatki zapišejo v posebno tabelo dejstev z dodatno datumsko dimenzijo, datum poročanja. Tabela se zato po zaključku obdobja ne briše več in se lahko samo dopolni z zapisi na nov datum poročanja. Konceptualno podobna alternativa dimenziji datuma poročanja je še scenarij oziroma verzija, ki se je, poleg poročanja o realizaciji, uporabljala predvsem pri shranjevanju različnih verzij planov z namenom, da se je vedno ohranil referenčni plan, na podlagi katerega se je ob zaključku poslovnega obdobja merilo doseganje realizacije.

Pri kumulativnih podatkih pa je namen predvsem, da se po zaključku poslovnega meseca prepreči morebitne naknadne knjižbe v zaključeno obdobje, ki bi spremenile vrednosti že objavljenega poročila. To je zagotovljeno z blokiranjem polnjenja kumulativnih vrednosti zaključenega meseca (npr. januar–februar) in odprtje novega kumulativnega meseca (npr. januar–marec), kamor se polnijo vse agregirane kumulativne vrednosti knjižb do zaprtja tega meseca, vključno z naknadnimi knjižbami v preteklo obdobje. Vrednost prometa posameznega meseca pa se izračuna v poročilu na podlagi razlike med posameznima kumulativnima mesecema. Tak način izračunavanja vrednosti posameznega meseca je v začetku leta sicer performančno potratnejši, vendar je proti koncu leta, ko se intenzivneje analizira celotno preteklo obdobje in je v ospredju napovedovanje doseganje plana celotnega leta, odzivnejši, saj so celoletni podatki že predhodno agregirani.

1.1 Izzivi pri tradicionalnih podatkovnih skladiščih

Glavni težavi tradicionalnih podatkovnih skladišč sta fleksibilnost pri razširitvah in nadgradnjah vsebine ter kompleksni ETL[2] procesi za polnjenje.

Inmonov koncept podatkovnih skladišč se sooča še z dvema izzivoma. Da bi zagotovili centralizacijo, je potrebna velika investicija vnaprej, saj se loti razvoja celotne vsebine na zalogo in se šele po izgradnji loti dodelav in razširitev glede na odkritja uporabnikov (Inmon, 2002, str. 29). Drug izziv pa je zagotavljanje ene resnice (Inmon, 2002, str. XVI), ki je vseeno lahko različna glede na čas analize in področja, v katerem so informacije pridobljene.

Kimballov koncept podatkovnih skladišč pa se sooča z anomalijami dimenzijskega modela, ki se pojavljajo pri snežinkastih »*snowflake*« shemah, kot so ogromne dimenzije in težave pri spreminjanju demografskih podatkov o osebah. Težave se pojavljajo tudi pri spreminjanju hierarhij, kjer se pojavijo potrebe po vpeljavi relacij mnogo-proti-mnogo pri npr. povezavi vodja-zaposleni, ki se lahko s časom spreminja, poleg tega pa tudi število nivojev v hierarhiji lahko variira.

[2] ETL je v podatkovnem skladišču splošno znana kratica za Extract, Transform, Load oziroma ekstrakcija, preoblikovanje in polnjenje podatkov.

Vsak od modelov, tako tradicionalnih kot modernih, ima svoje specifične značilnosti, ki imajo svoje prednosti in slabosti. Da bi modele lahko kasneje primerjali med seboj, je potrebno razumeti njihove glavne značilnosti, ki jih bom predstavil v sledečih podpoglavjih.

1.2 Dimenzijski model

Dimenzijski model, kot ga je predstavil Ralph Kimball, ali njegove variacije z različnimi manjšimi odstopanji je najbolj razširjen model podatkovnega skladišča, saj je najbolj razumljiv širokemu krogu uporabnikov in razvijalcev podatkovnih skladišč. Zaradi splošne prepoznavnosti modela bom v tem poglavju predstavil samo osnovni koncept modela in točke, ki so pomembne za primerjavo z ostalimi obravnavanimi modeli. Pri obravnavi modelov se bom namreč tudi pogosto skliceval na dimenzijski model in primerjal ali z njim nadgrajeval obravnavani model.

Če povzamem Kimballov opis, je podatkovno skladišče nabor podatkovnih struktur za prečiščevanje in predstavitev podatkov, pri čemer so operativni podatki strukturirani na način, ki omogoča hitro in enostavno analizo končnih uporabnikov (Kimball, 2013).

Usmerjeno je v zagotavljanje podatkov končnemu uporabniku. Integracija podatkov iz različnih virov se izvaja preko ETL procesov, ki polnijo analitična področja v dimenzijskem modelu.
Dimenzijskem model je torej skupina analitičnih področij, ki so sestavljeni iz tabel dejstev in konformnih dimenzijskih tabel ter tvorijo zvezdne ali kompleksneješe snežinkaste sheme.

Tabele dejstev so zelo normalizirane in so sestavljene iz relacij preko zunanjih ključev na dimenzije in mer, ki se agregirajo ob analizah, glede na nivoje in skupine v dimenzijah.

Dimenzijske tabele so zelo denormalizirane in vključujejo predvsem opisne podatke. Dimenzijske tabele, ki se uporabljajo preko več različnih tabel dejstev, so konformne dimenzije. Na ta način omogočajo povezanost celotnega podatkovnega skladišča in torej uporabo več mer iz različnih tabel dejstev v istem poročilu, ki se prekrivajo preko konformnih dimenzij (Kimball, 2013).

Zgodovina se beleži preko počasi spreminjajočih se dimenzij, ki omogočajo verzioniranje sprememb podatkov na različne načine, kot je opisano v poglavju 2.2. Podatki v tabelah dejstev so na najpodrobnejšem nivoju granulacije in se zato pri polnjenju ne agregirajo.

1.3 Normalizirano podatkovno skladišče po Inmonu

Podatkovno skladišče zajema podatke celotnega podjetja in je centralen podatkovni sistem, ki zagotavlja samo eno resnico. Bistvo podatkovnega skladišča je integracija podatkov iz različnih virov, kar je skrito očem končnih uporabnikov, saj se izvaja v ozadju. Glavni namen

je dostava informacij naprednim končnim uporabnikom v obliki področnih podatkovnih skladišč, ki služijo trenutnim zahtevam in se dopolnjujejo in spreminjajo glede na nove zahteve in spoznanja uporabnikov. Metodologija gradnje skladišča sledi pristopu od spodaj navzgor, kar pomeni, da zahteve končnih uporabnikov še niso dokončno oblikovane. Zato se podatkovno skladišče najprej dogradi s čim več podatki, nato pa končni uporabniki na podlagi rezultatov poizvedb spoznajo, kaj bi za boljše analize v trenutni situaciji še potrebovali. Na podlagi njihovih ponovnih okvirnih zahtev se začne nov krog izgradnje (Inmon, 2002).

1.3.1 Ravni podatkovnega skladišča

Kot pravi W. H. Inmon v Building the data warehouse (2002), naj bi arhitekturo podatkovnega skladišča v skladu z njegovim konceptom »*The Corporate Information Factory*« sestavljali štirje nivoji: operativni, atomarni, oddelčni/področni in individualni.

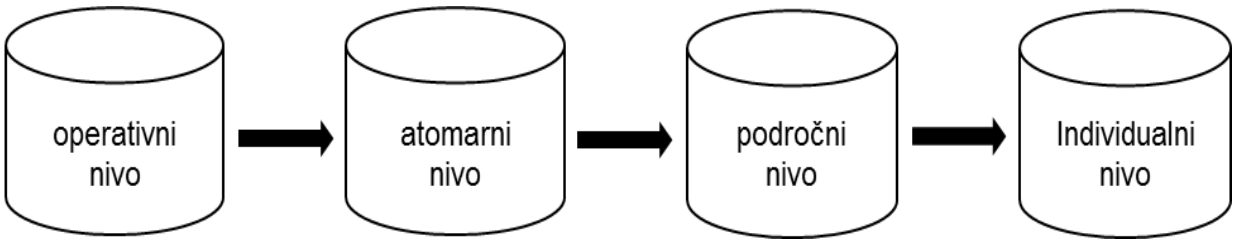
Operativni nivo je namenjen hitremu vsakodnevnemu podrobnemu trenutnemu pregledu in je zato omejen na posamezno izvorno aplikacijo kot vir podatkov.

Atomarni nivo je namenjen shranjevanju zgodovine na najbolj podrobnem nivoju po posameznih vsebinah, ki so integrirane iz več virov.

Oddelčni/področni nivo arhitekture oziroma večdimenzijski nivo je, kot že ime pove, namenjen analizam po določenih poslovnih področjih, kot so prodaja, trženje, nabava, proizvodnja, aktuariat, računovodstvo itd. Zaradi zasnove atomarnega nivoja kot osrednjega dela podatkovnega skladišča, na katerega se kot kraki navezujejo analitična področja, je model v praksi znan tudi kot model vozlišča in priključkov (angl. *hub-and-spoke*). Poleg originalnih podatkov so na tem nivoju lahko prisotne tudi izvedene informacije, ki lahko zelo pohitrijo analize pri končnih uporabnikih. Primer je izračun marže na nivoju produkta kot razlika med prodajno in nabavno vrednostjo ali povprečne vrednosti zaloge v mesecu za izračun koeficienta obračanja zalog, ki lahko bistveno skrajša ABC analizo produktov, ki se najhitreje ali najpočasneje obračajo.

Individualni nivo je namenjen *ad hoc* analizam, ki jih uporabniki na svojih osebnih računalnikih izvajajo in prilagajajo trenutnim potrebam. Ker so te analize samo trenutno zanimive in se ne izvajajo periodično oziroma se lahko v taki obliki nikoli več ne izvedejo, niso del področnega nivoja.

Slika 1: Nivoji arhitekture



Vir: W. H. Inmon, Building the data warehouse, 2002, str 39.

1.3.2 Redundanca podatkov

Večnivojska arhitektura sicer nakazuje na večjo redundanco podatkov, vendar so podatki na različnih nivojih in ponujajo različne informacije. Redundanca zato ni tako velika, saj je na operativnem nivoju npr. le trenutni podatek o kupcu, v atomarnem pa je celotna zgodovina kupca in integracija več podatkov iz operativnega nivoja. Dimenzijski nivo sicer vsebuje tako podatke iz operativnega kot atomarnega nivoja, vendar je to razumljivo, saj so podatki zaradi specifičnih zahtev posameznega oddelka denormalizirani in prilagojeni z namenom hitrih in uporabniku prilagojenih analiz. Podatki na individualnem nivoju pa so navadno v manjšem obsegu in samo začasno (Inmon, 2002, str. 17–18).

1.3.3 Granulacija in particioniranje

Inmon (2002, str. 56) poudarja pomen ustrezne granulacije podatkov in particioniranja z naslednjimi besedami: »Pogosto je rečeno, da če sta granulacija in particioniranje dobro urejena, potem večinoma tudi z vsemi ostali vidiki podatkovnega skladišča ni težav. V kolikor pa to ni skrbno urejeno, potem tudi drugi vidiki nimajo več pomena.«

Po Inmonu (2002, str. 56) je namen particioniranja fizična razdelitev podatkov v manjše obvladljive dele. Glede na moje izkušnje s podatkovnimi skladišči so pomembnejše prednosti še hitrejša reorganizacija, indeksiranje in branje podatkov. Pri tem je potrebno upoštevati tudi razliko pri klasičnih relacijskih podatkovnih bazah in sodobnimi, kot je IBM Netezza, ki ne uporablja indeksov, vendar je zato še toliko bolj pomembna organizacija in distribucija podatkov. Netezza namreč v nasprotju z delovanjem indeksov na tabelah pri klasičnih relacijskih bazah ne išče po stolpcih v indeksu, na katerih vrsticah se podatki nahajajo, temveč najprej iz nabora odstrani stolpce, kjer podatkov ni, in nato odstrani še nepotrebne vrstice glede na omejitve v »WHERE« pogojih. Tako ostane ekstremno malo podatkov za obdelavo na procesorju in zato lahko vrne rezultat poizvedbe na več TB veliki tabeli v le nekaj sekundah.

1.3.4 Gradniki modela vozlišča in priključkov

Kombinacija Inmonovega atomarnega modela kot osrednjega podatkovnega skladišča v ozadju in dimenzijskega modela s strukturo, predstavljeno že v poglavju 1.2, v ospredju namenjena končnim uporabnikom, predstavljata dobro kombinacijo celovitega podatkovnega skladišča, znanega tudi kot model vozlišča in priključkov.

Osnovne strukture v osrednjem modelu so sidra, entitete in vezi. Sama arhitektura je zelo podobna podatkovnemu trezorju, vendar z razliko, da je v Inmonovem centralnem modelu referenčna integriteta pomembna, podatki pa se uredijo že pred polnjenjem v atomarni nivo, kjer se vodi zgodovina, ki ustreza tipu 2 po Kimballovi klasifikaciji počasi se spreminjajočih dimenzij.

Sidra so osrednje tabele, ki predstavljajo glavne entitete in zajemajo samo primarne ključe teh entitet oziroma poslovne ključe, posebno polje tip, ki opredeljuje posamezno poslovno vsebino, ter tehnična polja, kot je umetni ključ, oznaka izvornega podatkovnega vira, datum in čas prvega polnjenja podatka v podatkovno skladišče in unikatno zaporedno številko polnjenja podatkovnega skladišča. Ker je edina prava lastnost iz poslovnega vidika za posamezen poslovni tip samo poslovni ključ, ni nobenega razloga, da bi se po prvem vpisu kakršen koli podatek kadar koli lahko spremenil. Zaradi osnovnega namena po beleženju zgodovine in referenčne integritete z drugimi tabelami se podatek tudi nikoli ne briše.

Entitete so tabele, namenjene beleženju lastnosti entitete, ki ji pripadajo, in evidenci zgodovinskih sprememb. Polja so lahko opisna, datumska, vrednostna ali kot reference na druge entitete. Za beleženje splošnih šifrantov je namenjena določena posebna entiteta »Code«. Za beleženje kompleksnejših šifrantov, ki se lahko med seboj povezujejo v različne časovno spreminjajoče hierarhije, je namenjena entiteta »Category«. V posamezni entitetni tabeli je lahko shranjenih več entitetnih poslovnih vsebin. Ker zgodovina sprememb sledi Kimballovemu načinu tipa 2 počasi spreminjajočih dimenzij, ima vsak poslovni zapis trajni unikatni ključ, ki je obenem v kombinaciji s poslovnim tipom, zapisan v sidru. Tako vsebuje tudi datuma začetka in konca veljavnosti in poleg njiju še datuma začetka in konca poslovne veljavnosti. Poleg unikatnega trajnega ključa je prisoten še primarni ključ tabele, ki se zapiše s sekvenco sprememb na posameznem zapisu. Za referenčno integriteto tipov skrbi posebna entiteta »Type«, ki vsebuje hierarhijo vseh poslovnih tipov vseh sider in entitet, pri čemer mora biti poimenovanje vsakega tipa unikatno, da ne pride do napačnih sklicevanj ob pripravi poizvedb.

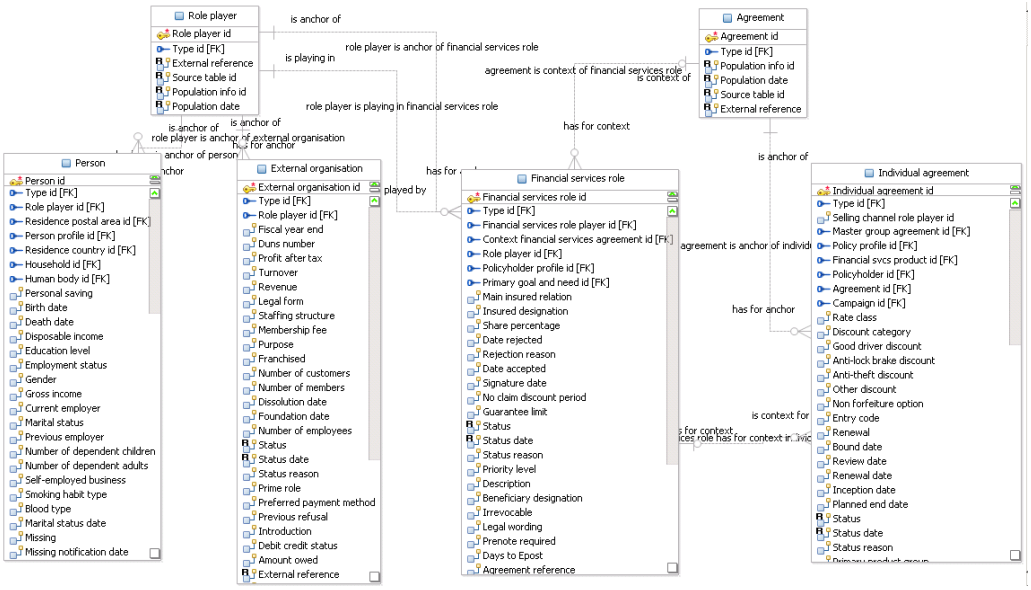
Vezi so relacijske tabele, ki povezujejo sidra med seboj ali sama nase, v kolikor se sklicuje na poslovno vsebino, ki je v istem sidru. Ker se lahko posamezna entiteta večkrat pojavi kot del vezi, model uvaja koncept vloge, ki bo opisan v nadaljevanju. Tudi vezi se lahko spreminjajo ali prekinjajo. Po je načinu beleženja zgodovine, so vezi zelo podobne strukturi počasi se spreminjajočih dimenzij tipa 2 in zato vsebujejo tudi datuma začetka in konca

veljavnosti. Poleg tehnične veljavnosti povezave vsebujejo vezi tudi dodatni datumski polji vsebinskega začetka in konca veljavnosti. Primera datumov vsebinskega začetka in konca sta datum veljavnosti police in datum prenehanja police, ki se razlikujeta od tehničnih datumov, ko je zapis v podatkovnem skladišču nastal.

Za referenčno integriteto vlog skrbi posebna entiteta »Relationship type«, ki v ločenih stolpcih opredeljuje poslovne tipe, ki jih med seboj povezuje, oznako vloge in opis vloge. Poleg tega vsebuje tudi primarni ključ, na katerega se sklicujejo vezi.

Na Slika 2 je diagram, ki sem ga oblikoval po predlogi IBM Insurance Information Warehouse, da ponazorim model vozlišč in priključkov. Prikazuje povezave vlog na polici z osebo in polico. »Role player« ali je sidro za vse osebe, tako pravne »External organisation« kot fizične »Person«, ter njihove vloge, ki so v primeru vlog na zavarovalnih policah shranjene v entiteti »Financial services role«. »Financial services role« ima torej polju »Type_id« določen tip vloge, npr. zavarovanec, in se preko polja »Financial services roleplayer id« povezuje na trajni ključ fizične ali pravne osebe. Obenem ima tudi svoj trajni ključ zapisan v »Role player id«, ki je prav tako zapisan v »Role player« sidru. Poleg tega ima še primerni ključ tabele »Financial services role id«, ki je sekvenca sprememb na tem trajnem ključu. Preko polja »Context financial services agreement id« pa se sklicuje na zavarovalno polico preko polja »Agreement id« v sidru »Agreement«. Vsi atributi zavarovalne police z vsemi zgodovinskimi spremembami se hranijo v tabeli »Individual agreement«.

Slika 2: Vloge na zavarovalni polici



Vir: IBM Corporation, IBM Insurance Information Warehouse model 8.4, 2011.

1.4 Model podatkovni trezor

Linstedt (2015a) uporabi naslednjo **definicijo** podatkovnega trezorja: »Podatkovni trezor je povezan sistem normaliziranih tabel, ki na najpodrobnejšem nivoju beleži zgodovino in hkrati zajema več poslovno-funkcijskih področij. Gre za hibridni pristop, ki združuje najboljše lastnosti zvezdaste sheme in tretje normalne forme. Arhitektura je fleksibilna, razširljiva, konsistentna in prilagodljiva potrebam podjetja. Je podatkovni model, ki je dizajniran prav za potrebe sodobnih podatkovnih skladišč.«

Osnovni namen podatkovnega trezorja, kot ga je opredelil Linstedt (2015a), je integracija različnih podatkovnih virov v enotno podatkovno skladišče, hranjenje zgodovine teh podatkov v njihovi izvirni obliki, s čimer zagotovi tudi vedno bolj pogosto zahtevo po sledenju podatkov. S tem načinom pa se tudi nekoliko oddalji od tradicionalne ideje ene same resnice, saj je zdaj resnica pogojena z eno verzijo podatkov glede na čas nastanka.

Osnovna arhitektura podatkovnega trezorja zajema središča (*hub*), satelite (*satellite*) in vezi (*link*), kar je v osnovi zelo podobno ostalim modernim podatkovnim modelom, kot sta npr. sidrni in model vozlišč in priključkov, ki upoštevata Inmonov koncept celovitega podatkovnega skladišča (Linstedt, 2015a).

Središča so osrednje tabele, ki predstavljajo glavne entitete in zajemajo samo primarne ključe teh entitet oziroma poslovne ključe, ter tehnična polja, kot je umetni ključ, oznaka izvornega podatkovnega vira, ter datum in čas prvega polnjenja podatka v podatkovno skladišče. Ker je edina lastnost s poslovnega vidika samo poslovni ključ, ni nobenega razloga, da bi se po prvem vpisu kakršen koli, tudi tehnični podatek, kadar koli lahko spremenil zaradi osnovnega namena. Po beleženju zgodovine in povezave z drugimi tabelami pa se podatek tudi nikoli ne briše (Linstedt, 2015a).

Tabela 1 predstavlja entiteto kupca s primeri zapisov kot je predvideno po modelu podatkovni trezor.

Tabela 1: Entiteta kupec

Primarni ključ tabele (sekvenca)	**Številka kupca (poslovni ključ)**	**Oznaka vira**	**Čas prvega polnjenja**
ID_KUPEC	**ST_KUPCA**	**POD_VIR**	**CAS**
1	A123456	SAP	31.03.2010 00:00:00
2	B656321	NAV	01.04.2010 00:00:00

Sateliti so tabele, namenjene beleženju lastnosti entitete, kateri pripadajo, in evidenci zgodovinskih sprememb. Večina polj v tabeli je opisnih, saj zajemajo vse lastnosti entitete, ki pokrivajo določeno področje entitete in izhajajo iz istega podatkovnega vira. Po konceptu

je struktura satelitov zelo podobna strukturi počasi se spreminjajočih dimenzij tipa 2 (Linstedt, 2015a).

Tabela 2 predstavlja satelit, ki vsebuje atribute naslova kupca s primeri zapisov kot je predvideno po modelu podatkovni trezor.

Tabela 2: Satelit naslov kupca

Primarni ključ tabele	**Pošta**	**Kraj**	**Naslov**	**Oznaka vira**	**Čas polnjenja**
ID_KUPEC	**POST**	**KRAJ**	**NASLOV**	**POD_VIR**	**CAS**
1	1000	Ljubljana	Dunajska 123	SAP	31.03.2010 00:00:00
2	1291	Škofljica	Jagrova 15	NAV	01.04.2010 00:00:00
1	1000	Ljubljana	Tržaška 76	SAP	01.10.2012 00:00:00

Povezave so relacijske tabele, ki povezujejo središča med seboj, v določenih primerih pa tudi med vezmi. Ker so vse vezi modelirane kot relacije mnogo proti mnogo, lahko v vezi zapišemo kakršne koli podatke, ne glede na referenčno integriteto (Linstedt, 2015a).

Tabela 3 predstavlja povezavo kupca na polico s primeri zapisov kot je predvideno po modelu podatkovni trezor.

Tabela 3: Povezava zavarovalna polica kupca

Primarni ključ tabele (sekvenca)	**Kupec (poslovni ključ)**	**Zavarovalna polica (poslovni ključ)**	**Oznaka vira**	**Čas polnjenja**
ID	ID_KUPEC	ID_ZAV_POL	POD_VIR	CAS
1	1	576453453466	SAP	31.03.2010 00:00:00
2	2	576453453486	NAV	01.04.2010 00:00:00
3	1	576457236562	SAP	31.03.2011 00:00:00

1.5 Sidrni model

Sidrni model so Rönnbäck, Regardt, Bergholtz, Johannesson in Wohed (2010, str. 1) povzeli z naslednjo definicijo: »… agilna tehnika modeliranja informacij, imenovana sidrni model,

ki ponuja neuničljive razširitvene mehanizme, ki omogočajo robustno in prilagodljivo upravljanje s spremembami. Glavna prednost sidrnega modela je, da spremembe v podatkovnem skladišču potrebujejo samo nove zapise in ne spremembe obstoječih.«

Glavne strukture, kot jih predstavijo Rönnbäck et al. (2010, str. 2) so sidra (angl. *anchors*), vozli (angl. *knots*), atributi (angl. *attributes*) in vezi (angl. *ties*).

Sidra predstavljajo nabor entitet, kot so kupci, produkti, dogodki, škodni spis, polica.

Namen **vozlov** je, da zajemajo manjši nabor vrednosti, ki se s časom ne spreminja. Primer je lahko spol, ki z naborom vrednosti 'moški' in 'ženska' opiše osebo in se s časom ne spreminja.

Atributi so namenjeni zajemu lastnosti sider. Obstajajo 4 tipi atributov: statični atribut (angl. *static*), zgodovinski atribut (angl. *historized*), vozelni statični atribut (angl. *knoted static*) in vozelni zgodovinski atribut (angl. *knoted historized*).

Statični atribut predstavlja lastnosti entitete, kjer ni potrebno voditi zgodovine sprememb vrednosti. Primer atributa je rojstni datum.

Zgodovinski atribut je uporabljen, kjer je potrebno voditi spremembe vrednosti atributa. Primer takega atributa je teža osebe.

Vozelni statični atribut predstavlja povezave med sidri in vozli, da lahko lastnosti entitete povežemo z naborom znanih vrednosti.

Vozelni zgodovinski atribut je uporabljen, kjer povezava entitete z vrednostjo atributa ni statična, ampak se spreminja s časom.

Vezi so povezave med dvema ali več sidri oziroma lahko tudi vozli. Podobno kot atributi so lahko statične, zgodovinske, vozelne statične in vozelne zgodovinske. Ker se lahko posamezna entiteta večkrat pojavi kot del vezi, model uvaja koncept vloge.

Primer vloge je pogost v zavarovalništvu, kjer ima lahko ista oseba na isti zavarovalni polici eno ali več vlog, ko so nosilec police, zavarovanec, plačnik, upravičenec. Podobno je oseba na škodi lahko povzročitelj, zavarovanec, oškodovanec.

2 PRIMERJALNA PODROČJA

Odločitev podjetja za podatkovno skladišče je investicija v boljše informacije, na podlagi katerih se pričakuje boljše odločitve v podjetju in na podlagi katerih bi se lahko posledično zmanjšali operativni stroški oziroma bi se lahko povečal tudi prihodek. Kot vsaka investicija ima tudi podatkovno skladišče svojo stroškovno plat in svoj ciljni donos.

Stroškovna plat projekta pogosto ni celovito ovrednotena, ker stroški nastajajo v različnih fazah življenjske dobe podatkovnega skladišča. Vendar je na začetku težko predvideti vse stroške. Zlasti pomembni in težavni so tudi oportunitetni stroški izbire med različnimi alternativami izbire modelov pri izgradnji podatkovnih skladišč. Poleg direktnega vpliva na stroške lahko vplivajo na trajanje projekta, kar vedno pomeni še dodaten strošek, ter na možnost kasnejših sprememb in historičnih analiz, za kar podatki lahko ne bodo več razpoložljivi.

Z namenom, da bi čim bolj objektivno primerjal različne podatkovne modele, sem določil glavne dejavnike, ki vplivajo na uspešnost uvedbe projekta podatkovnega skladišča, in jih ločil na dejavnike, ki imajo kratkoročen vpliv, in dejavnike, katerih vpliv se pokaže šele kasneje. Posamezne dejavnike bom podrobneje opisal v naslednjih podpoglavjih, saj bodo osnova za primerjavo vsakega od obravnavanih modelov v nadaljevanju.

Dejavniki, ki že kratkoročno in dokaj neposredno vplivajo na stroške, so predvsem:

- projektni pristop in metodologija razvoja,
- kompleksnost modela in stroškovni vidik razvoja ter
- polnjenje, čiščenje, integracija in transformacija podatkov ter njihova hitrost.

Tu so upoštevani predvsem izdatki za zunanje izvajalce, ki nastajajo že tekom implementacije osrednjega dela podatkovnega skladišča.

Kriteriji, ki na daljši rok vplivajo na stroške oziroma na kasnejšo dodano vrednost, so še:

- fleksibilnost in stabilnost pri naknadnih vsebinskih dopolnitvah,
- beleženje zgodovine podatkov,
- odzivnost in enostavnost analiz končnih uporabnikov ter fizične oziroma virtualizirane strukture ter
- nestrukturiranost podatkov.

2.1 Projektni pristop in metodologija razvoja

Projektni pristop je zelo pomemben dejavnik, ki vpliva na uspešnost projekta, še zlasti, če so v izgradnjo vključeni zunanji izvajalci. Pogosto so tendence obeh strani, tako izvajalca kot naročnika, da se zameji in fiksira obseg projekta in fiksira cena. Razlogi za to na strani naročnika so, da ima jasno sliko, kolikšen bo direkten strošek projekta, ki ga bo moral plačati izvajalcu, ter možnost pogoditi večji popust na ta strošek. Na strani izvajalca pa je tendenca, da ima s pogodbo zagotovljen posel za daljše obdobje in s tem zagotovljene redne prihodke. Težave pa lahko nastopijo, ko je obseg slabo definiran, najboljši resursi naročnika zaradi

prezasedenosti niso dovolj vključeni v projekt, njihovi namestniki in oddelek IT pa nimajo dovolj znanja in izkušenj za pripravo dobro definiranih zahtev, ker se prvič srečujejo s podatkovnim skladiščem. Kot omenja Moškon (2009, str. 9), je pred podpisom pogodbe oziroma pred začetkom izvedbe potrebna zelo dobra analiza zahtev. Tako je možnost težav sicer zmanjšana, toda ker obe strani še nista delali skupaj, je kakovost sodelovanja vseeno težko predvideti. V določenih primerih lahko tekom projekta nastopijo težave glede samega obsega projekta in spreminjanja zahtev, saj naročnik pogosto smatra, da je v obsegu vse, kar želi za nespremenljivo ceno, in da je kupil neke vrste končni produkt oziroma rešitev na ključ. Posledica so prekoračeni roki in konflikti med obema stranema, saj izvajalec ne more za dogovorjeno ceno narediti vseh sprememb. Situacija je skoraj neizogibna v primeru velikega obsega za nekajletni projekt, kjer prvotne uporabniške zahteve postanejo zastarele in jih je treba nadomestiti z aktualnejšimi.

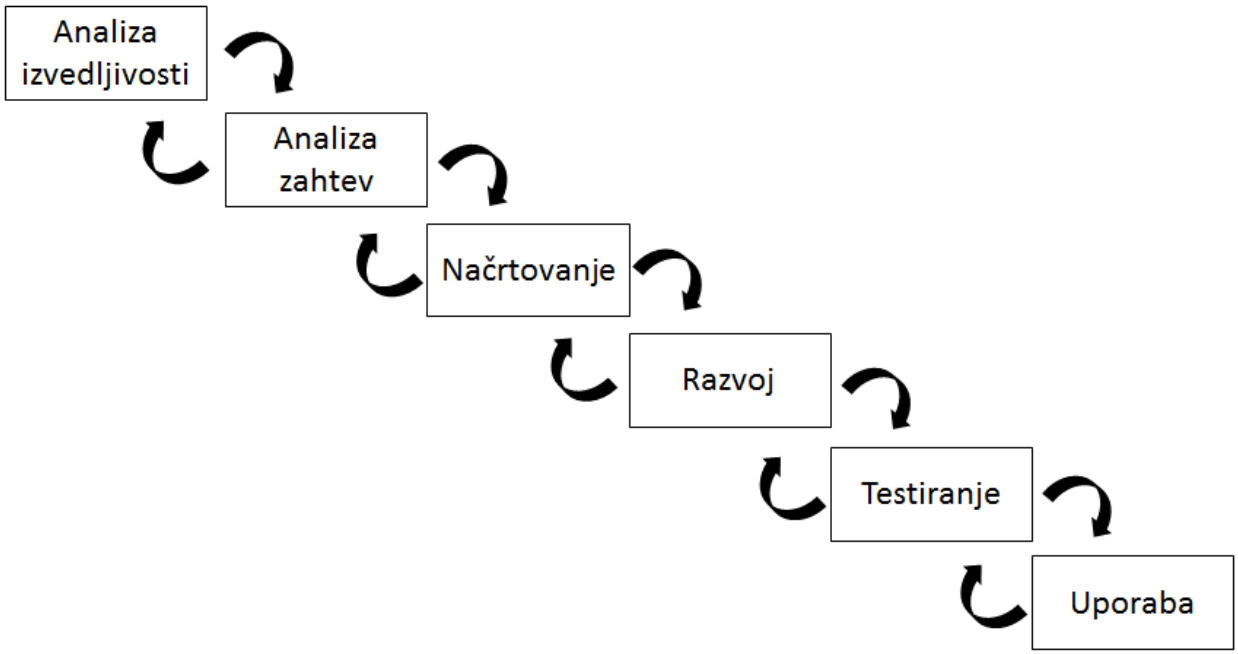
V kolikor se strani dogovorita za večfazno izvedbo, pri kateri se v prvi fazi izbere poslovno vsebino z zelo omejenim obsegom, se možnost kasnejših težav zmanjša, saj se iz začetnih težav pri skupnem delu obe strani naučita tudi, kako jih uspešno reševati. Pri naslednjih izvedbah za nove vsebine pa pretekle izkušnje upoštevata in zato izvedba napreduje bolj gladko. Dodaten strošek zaradi napak pri prvi vsebini je omejen samo na prvo implementirano vsebino in ne tudi na vse kasnejše. Planiranje izvedbe novih vsebin je zato bistveno bolj natančno tako s časovnega kot stroškovnega vidika.

Pri projektu podatkovnih skladišč je veliko odvisno tudi od izbire razvojnega cikla. Najbolj pogosto se s tradicionalnimi modeli uporabljata kaskadni razvojni cikel in spiralni razvojni cikel.

Kaskadni razvojni cikel, kot ga je opisal Royce (1970), je sestavljen iz 6 zaporednih faz razvojnega procesa, pri čemer se ob koncu vsake faze izvede validacijo in verifikacijo, kar je vhodna točka naslednje faze. Torej se vsaka naslednja faza lahko prične, ko je končana, preverjena in dobro dokumentirana predhodna faza. Razvojni cikel je tog in je zato uporaben predvsem pri projektih, kjer so uporabniške zahteve dobro znane in definirane že na začetku projekta. Na tej osnovi je mogoče že ob začetku projekta pripraviti podroben projektni plan s potrebnimi kadri, delom in trajanjem, ki se načeloma ne spreminja (Royce, 1970).
Model kaskadnega razvojnega cikla je prikazan na Slika 3 in prikazuje naslednje faze, ki se izvajajo zaporedno:

- analiza izvedljivosti,
- analiza zahtev,
- načrtovanje,
- razvoj,
- testiranje in
- uporaba

Slika 3: Faze kaskadnega modela



Vir: W. Royce, Managing the Development of Large Software Systems: Concepts and Techniques, 1970, str. 330.

Spiralni razvojni model, kot si ga je zamislil Boehm (1988, str. 7), temelji na identifikaciji in obvladovanju tveganj na projektu. Poteka iterativno, pri čemer vsaka iteracija zajema iste korake. Z oddaljenostjo od središča spirale se povečujejo kumulativni stroški, kot določa napredovanje po korakih. Model vsebuje 4 osnovne faze:

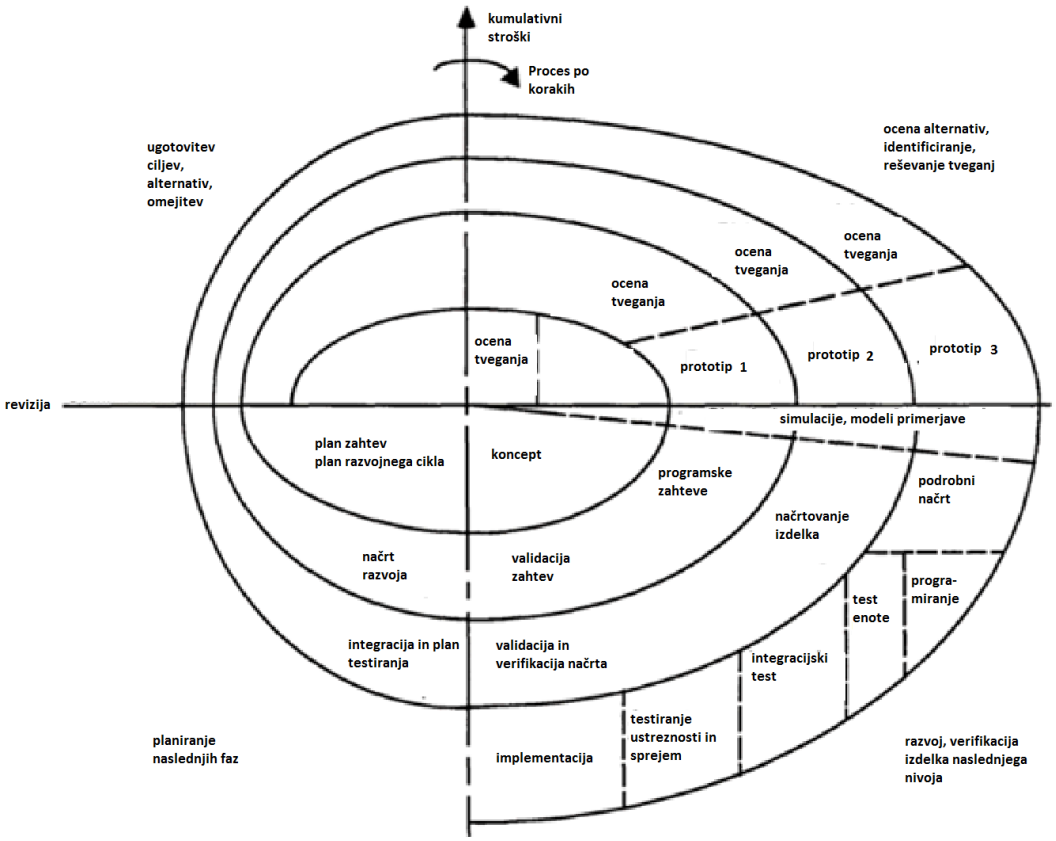
- ugotovitev ciljev, možnosti in omejitev;
- ocena možnosti ter identifikacija in reševanje tveganja glede na zastavljene cilje, ki vključuje tudi izdelavo prototipov in simulacij, dokler ne zmanjšamo tveganja na želeni nivo;
- razvoj in verifikacija lahko v zadnji verziji prototipa, ko je zadovoljil večino uporabniških in tehničnih pričakovanj, vsebujeta tudi korake iz katerega od drugih razvojnih modelov, npr. iz kaskadnega razvojnega modela;
- načrtovanje naslednje faze.

Glavna prednost spiralnega razvojnega modela je, da iz obstoječih razvojnih modelov izkorišča najboljše lastnosti in jih izboljšuje z zmanjševanjem tveganja, kot so zgodnje odpravljanje napak in opuščanje nezanimivih možnosti. Vendar pa z dodatno komponento zmanjševanja tveganj in več iteracijami povzroči tudi dodatne stroške in daljše cikle, kar je sicer z vidika fleksibilnosti sprejemljivo, v kolikor gre za interni projekt. V primeru, ko je vključen zunanji izvajalec, pa lahko v pogodbi čas in gradivo povzročita velike dodatne

izdatke v primerjavi s klasičnim kaskadnim modelom. V primeru pogodbe na ključ lahko hitro pride do konflikta z zunanjim izvajalcem, ki je zavezan s stroškovnega in terminskega vidika (Boehm, 1988).

Spiralni model je predstavljen na Slika 4 in prikazuje spiralo razvojnega procesa pri čemer so kumulativni stroški poteka prikazani na ordinatni osi.

Slika 4: Spiralni razvojni model



Vir: W.B. Boehm, A Spiral Model of Software Development and Enhancement, 1988, str. 7.

Poleg omenjenih modelov je v zadnjem času zelo pogosto uporabljen tudi agilni projektni razvojni cikel, in sicer predvsem Scrum metoda, ki podrobno določa korake za zbiranje omejenega obsega novih uporabniških zahtev, njihove analize in designa ter implementacije v majhnih koščkih oziroma sprintih, ki so izvedljivi v določenem terminskem obdobju. Prednost metode je predvsem dosledno upoštevanje prioritet in vrstnega reda, z namenom rednega nadgrajevanja obstoječih vsebin z novimi ter redno dajanje produktov končnim uporabnikom.

Schwaber in Sutherland (2011, str. 5) o Scrumu pravita: »Scrum je okvir, strukturiran za podporo razvoja kompleksnih izdelkov. Scrum je sestavljen iz Scrum ekip in z njimi

povezanih vlog, dogodkov, artefaktov in pravil. Vsaka sestavina znotraj okvirja služi posebnemu namenu in je ključna za uspeh in uporabo Scruma.«

Ključna razlika med klasičnim razvojnim ciklom, kot je kaskadni, in Scrum metodo je tudi v ekipiranju in ekipnemu načinu dela.

Schwaber in Sutherland (2011, str. 5) o Scrum ekipi pravita: »Scrum ekipo sestavljajo lastnik izdelka, razvojna ekipa in Scrum učitelj. Scrum ekipe se samoorganizirajo in so navzkrižno funkcionalne. Ekipe, ki se samoorganizirajo, same izbirajo, kako najbolje opraviti svoje delo, namesto da jim to določajo drugi, ki niso člani ekipe. Navzkrižno funkcionalne ekipe imajo vse sposobnosti, ki so potrebne za dokončanje dela in se ne zanašajo na druge, ki niso del ekipe. Ekipni model v Scrumu je zasnovan tako, da optimizira fleksibilnost, kreativnost in produktivnost. Scrum ekipe izdelke razvijajo iterativno in inkrementalno ter s tem maksimirajo priložnosti za povratne informacije. Inkrementalni razvoj »dokončanega« izdelka zagotavlja, da je potencialno uporabna verzija delujočega izdelka vedno na voljo.«

Da bi sam razvoj čim bolj potekal po planu in bi bil nadzor nad izvajanjem plana čim enostavnejši, metoda razvoj razdeli na posamezne Scrum dogodke. Schwaber in Sutherland (2011, str. 7) o Scrum dogodkih pravita naslednje: »Predpisani dogodki se v Scrumu uporabljajo za ustvarjanje stalnosti in zmanjševanje potreb po sestankih, ki s Scrumom niso določeni. Scrum uporablja časovno omejene dogodke, tako da ima vsak dogodek določeno maksimalno časovno okno. To zagotavlja ustrezno količino časa, porabljenega za načrtovanje, ne dovoljuje pa izgub v procesu načrtovanja.«

2.2 Beleženje zgodovine

Zgodovina podatkov oziroma spremembe podatkov se lahko beležijo na zelo različne načine. Kot izhodišče za primerjavo modelov bom zato vzel osnovo, ki izhaja iz dimenzijskega modela in sta jo navedla Kimball & Ross (2013, str. 53). V primeru dimenzijskega modela je namreč znanih 7 tipov počasi spreminjajočih dimenzij (angl. *slowly changing dimensions*), ki se spreminjajo skozi čas. Najbolj pogoste vsebine, ki potrebujejo beleženje zgodovine, so predvsem povezane z osebami, kupci ali zaposlenimi ter njihovimi lastnostmi in kontaktnimi podatki. Najbolj pogosti primeri so sprememba naslova, telefonske številke, stopnje izobrazbe in zakonskega statusa. Tipi, ki sta jih navedla Kimball in Ross (2013, str. 53–56), so:

Tip 0 oziroma pasivna metoda ob spremembi podatka na podatkovnem viru v dimenziji vedno ohrani le prvotno originalno vrednost.

Tip 1 oziroma posodobitev vrednosti ob spremembi podatka na podatkovnem viru prepiše vrednost v dimenziji z novo vrednostjo.

Tip 2 ob spremembi podatka na podatkovnem viru doda novo vrstico v dimenzijo z novo vrednostjo, pri čemer obdrži prejšnjo vrednost v originalnem zapisu vrstice. Ta metoda potrebuje, poleg atributov in poslovnega ključa, nujno še stolpec za dodatni ključ, ki obenem postane primarni ključ, saj se ob spremembi pojavi več zapisov z istim poslovnim ključem. Dodatno pa je potrebno tako dimenzijo opremiti še s stolpcem za verzioniranje oziroma sortiranje zgodovine sprememb ali namesto tega dva dodatna časovna, navadno datumska stolpca, v katera se zapiše datum začetka veljavnosti zapisa v prvega in datum konca veljavnosti zapisa v drugega. Pri tem se predhodno veljavni zapis zaključi z zapisom datuma spremembe, zmanjšan za en dan, v stolpec z datumom konca veljavnosti, v nov zapis pa se datum spremembe zapiše v stolpec datum začetka veljavnosti. Pri tem lahko v novem aktualnem zapisu datum konca veljavnosti ostane »NULL« ali se določi nedosegljiv datum, npr. 31. 12. 9999. Dodatno je potreben še indikator zadnjega veljavnega zapisa v ločenem stolpcu.

Tip 3 ob spremembi podatka na podatkovnem viru ohranja vsebino v ločenih stolpcih, pri čemer je s številom stolpcev določeno število verzij zgodovine. Tako en stolpec vedno predstavlja zadnjo vrednost, ostali pa predhodne verzije vrednosti. Navadno sta pri tem tipu 2 stolpca, pri čemer en prikazuje trenutno vrednost, en pa originalno.

Tip 4 za hranjenje zgodovine sprememb poleg osnovne tabele, ki prikazuje aktualno vrednost podatkov, kot tudi aktualni dodatni ključ kot primarni ključ, uporablja še dodatno zgodovinsko tabelo kot mini dimenzijo, ki sledi podobnemu konceptu kot tip 2. Pri tem načinu se v tabeli dejstev ohranjata oba tuja ključa, kar zagotavlja hitrejše izvajanje analiz.

Tip 5 zajema koncept tipa 4 in dodaja še dodaten stolpec v osnovno dimenzijo kot referenco na zadnje stanje v mini dimenziji.

Tip 6 združuje pristope tipov 1, 2 in 3. Na ta način se zgodovina časovno beleži z dodajanjem novih zapisov, katerih integriteta se zagotavlja z dodatnim ključem in datumoma začetka in konca veljavnosti ter indikatorjem aktualnosti zapisa kot pri tipu 2. Poleg tega je v vsaki vrstici v ločenih stolpcih zapisana aktualna vrednost atributa, ki se prepisuje kot pri tipu 1, in v dodatnem stolpcu še predhodna vrednost atributa kot pri tipu 3.

Tip 7 je hibridni, združuje tip 1 in tip 2. Do tabele dejstev je možna povezava kot tip 1, ki prikazuje samo trenutno vrednost, ali kot tip 2, ki prikazuje celotno zgodovino sprememb, saj sta v tabeli dejstev zapisana tako primarni ključ kot trajni sidrni ključ. Oba pogleda morata biti ločeno vzpostavljena v orodju za poročanje.

2.3 Fleksibilnost in stabilnost pri dopolnitvah

Fleksibilnost modela je mišljena z vidika, koliko napora je potrebno ob dodajanju popolnoma novih vsebin, novih podatkovnih virov ali samo razširjanje obstoječih vsebin z

novimi atributi. Potrebno je namreč upoštevati tudi zgodovinsko integriteto obstoječih podatkov, kateri se pridružuje oziroma jo razširjajo novi atributi s svojo zgodovinsko integriteto datumov veljavnosti. Pri določenih modelih npr. dodajanje nove vsebine ali podatkovnega vira nima nikakršnega vpliva na obstoječe vsebine, pri drugih pa so lahko potrebne kompleksnejše SQL operacije, da se poenoti zgodovinska veljavnost obeh vsebin.

2.4 Kompleksnost modela in stroškovni vidik razvoja

Kompleksnost modela je zelo pomembna z vidika, kako hitro ga lahko uporabniki spoznajo in samostojno uporabljajo. Nekateri modeli so zaradi svoje narave tako kompleksni, da je njihova dokumentacija pripravljena na specifičen način, da je sploh mogoče transparentno voditi vse vsebine v modelu in razbrati, kako so med seboj povezane. Zgradba določenih modelov, kot je npr. dimenzijski model, je že v osnovi zamišljena zelo enostavno in uporabniku prijazno, saj je model razdeljen v analitična področja po uporabniku smiselnih vsebinah. Drugi modeli pa so lahko večnivojski in z ločenim delom za beleženje zgodovine in drugim za poročanje, ki je lahko tako kompleksen, da za orodja za poročanje ni več primeren zaradi zahtevnih poizvedb iz zgodovinskih tabel in so zato potrebni še dodatni vmesni koraki in strukture za transformacijo podatkov.

2.5 Polnjenje, čiščenje, integracija in transformacija podatkov ter njihova hitrost

Modeli se med seboj lahko zelo razlikujejo po tem, v katerem koraku se zgodita čiščenje in transformacija podatkov. Nekateri modeli se že v osnovi v fazi ekstrakcije polnijo s čistimi podatki in transformirajo zaradi referenčne integritete z drugimi vsebinami, drugi pa se polnijo z nespremenjenimi in neočiščenimi podatki in se to zgodi šele v naslednjem koraku, in sicer pri pripravi struktur za poročanje, ki so namenjene končnim uporabnikom. Polnjenje podatkovnega skladišča z nespremenjenimi podatki je seveda hitrejše, kot če jih je treba prej še očistiti in transformirati, vendar je treba upoštevati celoten ETL proces in vpliv na odzivnost končnih poizvedb uporabnikov.

2.6 Odzivnost pri analizah končnih uporabnikov

Pri tradicionalnih podatkovnih modelih kot glavni gradniki prevladujejo fizične tabele dejstev in dimenzijske tabele. Pri določenih sodobnih modelih pa bi bile za namene ad hoc analiz lahko uporabljene virtualne strukture, kot so vidiki (angl. view) in materializirani vidiki (angl. *materialized view*). Ključna kriterija izbire sta čas polnjenja fizične tabele v povezavi s časovno ter strukturno fleksibilnostjo z vidika zajetih atributov v dimenziji ter odzivni čas poizvedb končnih uporabnikov. Zaradi različnih arhitektur podatkovnih skladišč so lahko razlike med modeli velike in bi lahko v določenih primerih uporabili tudi virtualne strukture.

2.7 Nestrukturirani podatki

Pri določenih podatkovnih modelih je mogoče nestrukturirane podatke neposredno zapisati v podatkovno skladišče ali pa samo uporabiti poenotene poslovne ključe in same nestrukturirane podatke pustiti v originalnem podatkovnem viru. Pri analizah se zato lahko podatki iz podatkovnega skladišča enoznačno povezujejo z nestrukturiranimi podatki.

3 DIMENZIJSKI MODEL

3.1 Projektni pristop in metodologija razvoja

Metodološki pristop razvoja je v tem primeru pristop od zgoraj navzdol, kar pomeni, da se najprej definirajo zahteve, podatkovni viri in se šele nato začneta dejanski razvoj in programiranje. Ker se zahteve navadno nanašajo na posamezno poslovno področje, je poleg Scrum metode možen tudi kaskadni razvojni model za posamezno področje, saj so zaradi fokusa na posamezno področje koraki kratki in je končni rezultat dokaj kmalu na voljo tudi končnim poslovnim uporabnikom. Zaradi zaključevanja po posameznih analitičnih področjih, ki jih je mogoče v grobem definirati že ob prvih poslovnih zahtevah, tudi izvajanje pogodbe na ključ z zunanjim izvajalcem ni problematično v primerjavi z ostalimi kompleksnejšimi podatkovnimi modeli.

3.2 Polnjenje, čiščenje, integracija in transformacija podatkov ter njihova hitrost

Tok podatkov vodi iz izvornih transakcijskih in drugih podatkovnih virov v področje priprave podatkov, ki preko ETL procedur in virtualnih pogledov transformira podatke v primerno obliko za polnjenje dimenzijskega modela, ki je množica povezanih analitičnih področij. Na dimenzijske modele se na koncu z različnimi analitičnimi orodji in aplikacijami priklapljajo poslovni uporabniki in izvajajo hitre analize.

3.3 Nestrukturirani podatki

Kimball (2012, str. 8–10) podaja naslednja priporočila za modeliranje in povezovanje nestrukturiranih podatkov v podatkovno skladišče: v vsakem nestrukturiranem podatku je možno prepoznati dimenzije in mere, čeprav na prvi pogled izgleda nemogoče. Konformne dimenzije pa so samo vezi, ki lahko združujejo različne podatkovne vire v enotno analizo. Nestrukturirani podatki iz »tweeta« lahko namreč vsebujejo osebo, čas, lokacijo, izdelek in ostale informacije, ki jih je mogoče dimenzionirati in tako preko konformnih dimenzij povezati v podatkovno skladišče. Dimenzije namreč lahko že vsebujejo ali pa jim dodamo atribute, preko katerih lahko povežemo določeno nestrukturirano informacijo z dimenzijo, četudi lahko le na zelo agregiranem nivoju. Vsako dimenzijo pa je potrebno opremiti s

trajnimi umetnimi ključi, ki jih ne more spremeniti nobeno poslovno pravilo. Lahko so preproste »integer« sekvence ali kompleksen »hash« ključ, ki zagotavlja unikatnost. Strukturirani podatki v podatkovnem skladišču naj se združujejo z nestrukturiranimi preko virtualizacije šele ob sami poizvedbi oziroma preko namenskih orodij, ki to omogočajo. Za zgodovinske spremembe nestrukturiranih podatkov je pomembno tudi , da se beležijo, in sicer na enak način kot počasi spreminjajoče se dimenzije.

Krishnan (2013, str. 204) pravi, da je zaradi masovnih podatkov (angl. *Big Data*) potrebno zelo spremeniti tudi način integracije podatkov, saj tradicionalne tehnike niso več primerne in jih je potrebno nadgraditi ali nadomestiti z ustreznejšimi, ki bodo sposobne hitro procesirati kompleksne poizvedbe na velikih količinah različno formatiranih podatkov.

3.4 Beleženje zgodovine

Kot je podrobno opisano že v poglavju 2.2, Kimball v dimenzijskem modelu loči 7 različnih tipov vodenja zgodovine. Tipa 0 in 1 ne beležita popolne vsebine sprememb, zato ne omogočata rekreiranja poročil z zgodovinskimi vrednostmi na poljuben dan v zgodovini in bi se zato večinoma izogibal njuni uporabi pri področjih, kjer je potrebno poročanje zunanjim regulatorjem oziroma je potrebna revizijska sled. Velika pomanjkljivost pri beleženju zgodovine, v primerjavi z Inmonovim osrednjim podatkovnim skladiščem, je, da dimenzijski model ne beleži nikakršne zgodovine za vsebine, ki niso bile zahtevane s strani poslovnih uporabnikov, saj sledi pristopu od zgoraj navzdol. Tako so vse spremembe na nevključenih podatkih za vedno izgubljene, v kolikor se niso vodile že na izvornem podatkovnem viru.

3.5 Fleksibilnost in stabilnost pri dopolnitvah

Dimenzijski model je vedno nadgradljiv z dodatnimi dopolnitvami z možnostjo dodatnega atributa na dimenziji, dodajanje dimenzije, dodajanja mere v tabelo dejstev ali kot izgradnja novega analitičnega področja, ki je z obstoječimi povezan preko konformnih dimenzij. Omejitev ali težava pri dopolnitvah pa se pojavi pri združevanju obstoječe zgodovine v dimenziji z obstoječo zgodovino nove vsebine oziroma novega atributa v obstoječi dimenziji, saj so potrebne kompleksnejše poizvedbe, ki uredijo preseke med zgodovinama ter poleg dimenzijske tabele posodobijo tudi tabele dejstev z novimi umetnimi ključi novonastalih zapisov v preseku zgodovin dimenzije.

3.6 Odzivnost pri analizah končnih uporabnikov

Dimenzijski model je že v osnovi načrtovan z namenom hitrih in enostavnih poizvedb, ki uporabnikom vrnejo poročilo oziroma rezultat v kar najkrajšem času. Zato je v tem pogledu daleč najhitrejši v primerjavi z alternativami. To je tudi glavna lastnost, da se dimenzijski

model pogosto uporabi v hibridnih modelih kot dodatna optimizacijska prvina, ki, poleg obstoječih značilnosti posameznega modela, izboljša tudi odzivni čas poizvedb končnih uporabnikov.

3.7 Kompleksnost modela in stroškovni vidik razvoja

Kimballov dimenzijski model je najbolj razumljiv model podatkovnega skladišča, kar je tudi eden od razlogov, zakaj je tudi najbolj razširjen. Sama zasnova je zelo enostavna, saj je razdeljen po poslovno-funkcijskih vsebinah v analitična področja, ki vsebujejo osrednje tabele dejstev z vrednostnimi podatki, na katere se vežejo dimenzije z opisnimi podatki. Polnjenje dimenzijskega modela s podatki sicer vsebuje tudi področje priprave podatkov, sicer pa se podatki transformirajo pred polnjenjem v dimenzije in tabele dejstev. Prav tako sam ETL proces ni tako dolg, kot če bi bil vključen še atomarni nivo, kar pomeni, da je razvoj hitrejši in cenejši. Vključevanje novih kadrov v projekt je zaradi lahko razumljivega modela prav tako hitrejše v primerjavi z alternativnimi modeli, kar zopet učinkuje na stroške.

4 MODEL VOZLIŠČA IN PRIKLJUČKOV

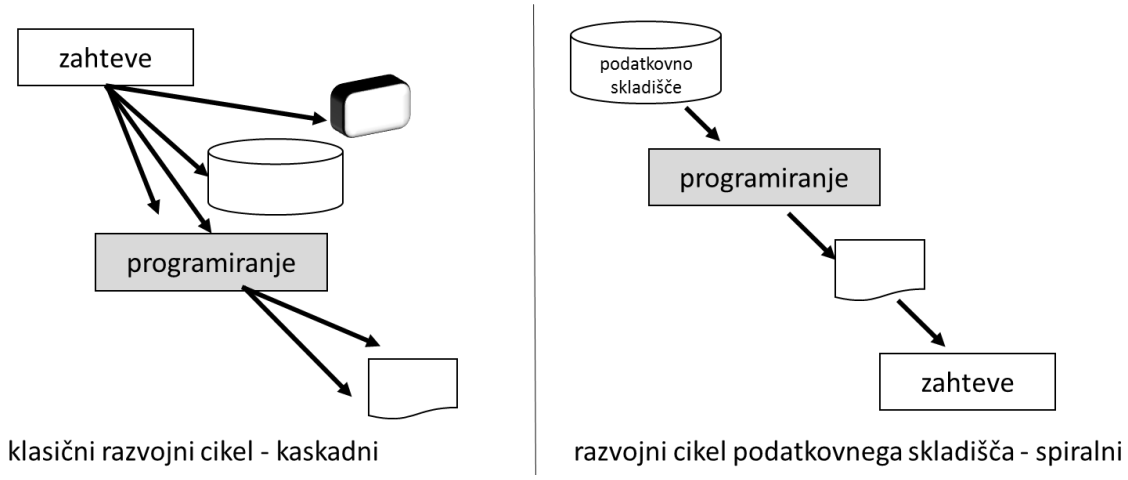
4.1 Projektni pristop in metodologija razvoja

Klasičen kaskadni razvojni cikel pri gradnji podatkovnega skladišča ni najbolj primeren, saj predvideva, da so vse zahteve že znane in podrobno definirane. Uporaben je sicer pri izgradnji operativnega nivoja, vendar za atomarni in večdimenzijski model, ki je namenjen predvsem analitikom, ki sproti dodajajo nove zahteve glede na rezultate raziskav na podlagi obstoječih zahtev. Torej klasični pristop, kjer je potrebno najprej razumeti poslovne zahteve in šele nato iti po korakih designa in razvoja, ni mogoč v začetnih fazah razvoja (Inmon, 2002, str. 19–21).

Razvojna metodologija podatkovnega skladišča navadno sledi skoraj obratnemu vrstnemu redu, kot je v kaskadni metodi, saj je podatkovno skladišče najprej implementirano in integrirano z ostalimi podatki. Nato se preverijo morebitna odstopanja v podatkih, ki jim sledi izdelava podrobnih analiz. Šele ko so rezultati analiz razumljeni, so razumljene tudi zahteve. Postopek je iterativen, saj rezultati generirajo potrebe po novih podatkih (Inmon, 2002, str. 19–21), kar začne nov cikel. Metodologija se zato imenuje tudi »spiralna« metodologija razvoja. Inmon (2002, str. 29) o tem pravi: »Pristop spiralnega razvoja predpostavlja, da so najprej do konca razviti manjši deli podatkovnega skladišča, nato pa se dodajajo novi manjši deli v naslednjih iteracijah.« Dejansko to pomeni, da končni uporabnik podatkovnega skladišča zahteva: »Daj mi, kar pravim, da želim, potem ti lahko povem, kaj res želim« (Inmon, 2002, str. 29).

Razlike med pristopoma, kot jih predstavi Inmon (2002, str. 22) so povzete v Slika 5, ki prikazuje različno zaporedje korakov klasičnega razvojnega cikla in razvojnega cikla podatkovnega skladišča.

Slika 5: Primerjava kaskadnega in spiralnega razvojnega cikla



Vir: W.H. Inmon, Building the data warehouse, 2002, str. 22.

Po Inmonu (2002) naj bi se skozi čas spreminjala tudi granulacija, v kateri so podatki dostopni uporabnikom. Mesečni in tedenski podatki o prodaji naj bi imeli zelo podrobno granulacijo, saj se analizirajo bistveno bolj podrobno kot 10 let stari podatki, ki so lahko agregirani na mesečnem nivoju. To nakazuje, da je bila metodologija razvita, ko so bili diski še dragi, saj argumentira, da starejše podatke lahko shranjujemo na počasnejše in cenejše diske. V sodobnih podatkovnih skladiščih, ki so vir za podrobno podatkovno rudarjenje in so vse najmanjše informacije lahko pomembne, pa podatki ostajajo na najbolj detajlni granulaciji. To je zlasti pomembno, kadar je potrebno podatke pogosto analizirati iz različnih nepredvidljivih vidikov, kar je na najnižji granulaciji podatkov najfleksibilnejše. Še posebno pri aktuarskih izračunih je zelo pomembna dolga časovna vrsta, tudi 20, 30 ali 50 let. Tako lahko npr. pravilno izračunajo zavarovalno-tehnični rezultat, saj je lahko razlika med 50-letno pobrano premijo in vsemi izplačanimi škodami v tem obdobju za določeno zavarovalno enoto le nekaj odstotna in lahko hitro postane nedobičkonosna. Delno bi lahko njegov predlog preslikali na sodobno tehnologijo v primeru strojne arhitekture, da bi novejše podatke, ki se uporabljajo pogosteje, lahko imeli v bazi v pomnilniku (angl. *in memory*), ki omogoča zelo hitre poizvedbe, starejše pa v klasični relacijski bazi. Tehnologija in način delovanja sodobnih relacijskih baz (npr. Netezza in Teradata v primerjavi z DB2 in Oracle oziroma analitičnimi DBMS in OLAP[3] različicami, kot je npr. Microsoft SQL Analysis Services) pa je zelo različna, zato podrobnejša primerjava sodobnih strojnih in programskih arhitekturnih možnosti presega namen te naloge.

[3] OLAP je kratica za online analytical processing oziroma analitična obdelava podatkov s povezavo.

4.2 Nestrukturirani podatki

Koncept podatkovnega skladišča je Bill Inmon predstavil že leta 1992 v knjigi Building the Data Warehouse, 1st Edition. Toda takrat nestrukturirani podatki še niso imeli teže, saj je bil izziv že priprava poročil in analize iz strukturiranih podatkov, zato tega koncepta še ni obravnaval. Zadnje čase pa nestrukturirani podatki predstavljajo vedno pomembnejši vir informacij, ki se je navadno analiziral v ločenih aplikacijah.

Inmon je tej temi namenil poglavje šele v knjigi Building the Data Warehouse, 4th Edition (2005), kjer je predstavil povezovanje strukturiranih in nestrukturiranih podatkov na osnovi verjetnosti ujemanja teksta, saj strukturirani podatki nimajo primarnih ključev, knjižb niti fiksnega formata, na katerega bi se lahko oprli.

4.3 Polnjenje, čiščenje, integracija in transformacija podatkov ter njihova hitrost

Sam model je večnivojski in zato poleg ekstrakcije podatkov (angl. *extract*) iz izvornih podatkovnih virov v prvem koraku predvideva tudi več nivojev transformacij (angl. *transform*) pred polnjenjem v osrednji atomarni del in nato še v področna podatkovna skladišča. V preteklosti je bila implementacija vseh nivojev pogosto prezahtevna, ker se zaradi tehničnih omejitev pri prenosu podatkov ni odvil v dovolj kratkem času (Theissen & Kraemer, 2009). Sodobna strojna oprema je sicer napredovala, vendar je celoten ETL proces, v primerjavi z ETL procesi modernejših modelov, še vedno časovno potratnejši.

4.4 Beleženje zgodovine

Kot je opisano že v poglavju 1.3.4 Gradniki modela vozlišča in priključkov na delu opisovanja entitet, zgodovina sprememb sledi konceptu Kimballovega načina tipa 2 počasi spreminjajočih dimenzij. Vsak poslovni zapis ima trajni unikatni ključ, ki je obenem v kombinaciji s poslovnim tipom zapisan v sidru. Vsebuje tudi datuma začetka in konca veljavnosti. Poleg unikatnega trajnega ključa je prisoten še primarni ključ tabele, ki se zapiše s sekvenco sprememb na posameznem zapisu. Ker je celotna zgodovina vedno zapisana v centralnem podatkovnem skladišču, je mogoče dimenzijski model vedno počistiti in ponovno napolniti z enakimi zgodovinskimi podatki. Na ta način se zaobide veliko pomanjkljivost Kimballovega podatkovnega skladišča, kjer je dodajanje novih vsebin z vidika ohranjanja in integracije obstoječe in zgodovine novih vsebin zelo zahtevno.

4.5 Fleksibilnost in stabilnost pri dopolnitvah

Temeljna značilnost osrednjega podatkovnega skladišča je, da se zapisi v osnovi ne spreminjajo, temveč samo dopolnjujejo z novo verzijo zapisa. Težava pa lahko nastopi ob

dodajanju morebitnih novih atributov, ki jih je potrebno iz izvornih sistemov napolniti v entitete, katerim bi bilo potrebno dodati nov stolpec za izbrani atribut, z vidika beleženja zgodovine pretekla ostane, kot je, vendar pa je zaradi novega atributa potrebno vse veljavne zapise ponovno dodati še z vrednostjo novega atributa. Na ta način je sicer zagotovljena fleksibilnost osrednjega modela, medtem ko je, kot je že omenjeno v predhodnem poglavju, mogoče ponovno napolniti tudi dimenzijski model. S tega vidika je celovito podatkovno skladišče zelo fleksibilno.

4.6 Odzivnost pri analizah končnih uporabnikov

Zaradi uporabe dimenzijskega modela, kot za področna podatkovna skladišča, so tudi za Inmonov model celovitega podatkovnega skladišča veljavne vse prednosti glede odzivnosti uporabniških poizvedb, ki so že opisane v poglavju dimenzijskega modela.

4.7 Kompleksnost modela in stroškovni vidik razvoja

Model vozlišča in priključkov je zamišljen kompleksnejše v primerjavi z enostavnim Kimballovim dimenzijskim modelom, saj je arhitektura večnivojska, kot je opisano že v poglavju 1.3.1. Ker je atomarni nivo organiziran po entitetah, ki v posameznih tabelah vključujejo več vsebin, ločenih s tipi, natančen vsebinski pregled iz navadnega entitetno-relacijskega diagrama ni najbolj pregleden. Zato je poleg logičnega in fizičnega modela potreben še poslovni model s povezavami na logični model. Za nove kadre na projektu to zelo vpliva, saj podaljša čas, ki je potreben za spoznavanje modela in vključenih vsebin, kar pa direktno vpliva na strošek projekta. V primerjavi s Kimballovim dimenzijskim modelom pa dodajanje nove vsebine predstavlja tudi višji strošek, saj je potrebno vsako vsebino, poleg polnjenja v sicer uporabnikom razumljiv področni nivo, dodatno polniti še atomarni nivo, kar pomeni dodatno delo in strošek na ETL procesih.

5 PODATKOVNI TREZOR

5.1 Projektni pristop in razvojna metodologija

Projektni pristop je, podobno kot pri Inmonovem konceptu, pristop od spodaj navzgor. To je logično, saj predvideva najprej vzpostavitev celovitega podatkovnega skladišča in šele nato podatkovne strukture za poročanje. Primerna razvojna metodologija bi lahko bila spiralna, saj omogoča, da bi v več ciklih vzpostavljali dodatne vsebine s podrobnostmi, vmes pa bi bili že vidni prvi rezultati. Iz istega razloga bi bila primerna tudi Scrum agilna metodologija, pri čemer prvi *sprinti* ne bi dostavljali uporabnih rezultatov za končne uporabnike, dokler osnovno ogrodje ne bi bilo vzpostavljeno. Nasprotno pa bi odsvetoval kaskadni razvojni model, saj bi faza analize in dizajna za celotno podatkovno skladišče trajala predolgo in bi se prvotne zahteve lahko preveč spreminjale in bi nastopile težave s

prevzemanjem dokončanih enot, posebno še v kombinaciji s pogodbo na ključ z zunanjim izvajalcem.

5.2 Beleženje zgodovine, hitrost in način ETL polnjenja

Glavna razlika, ki z vidika polnjenja podatkovni trezor loči od ostalih podatkovnih modelov, je, da že v prvem koraku zapisuje vse podatke ter hkrati tudi njihovo zgodovino v originalni obliki brez kakršnih koli transformacij. Po polnjenju podatkov se ti nikoli več ne brišejo, niti spremenijo. Edina sprememba na obstoječih zapisih je zaključevanje datuma veljavnosti ob novi verziji podatka. Poleg tega pa je ob dobrem definiranju metapodatkov o izvoru in ponoru podatkov in zaradi enostavnih poizvedb na originalni podatkovni vir možno pripraviti tudi generične ETL procese za avtomatski prenos podatkov v podatkovno skladišče. Zaradi zgoraj opisanega je polnjenje izredno hitro v primerjavi z drugimi podatkovnimi modeli. Pri ostalih podatkovnih modelih, kar sem opazil tudi v moji praksi, je ta del predstavljal le fazo ekstrakcije podatkov iz podatkovnega vira pred samim polnjenjem podatkovnega skladišča. Sama faza ekstrakcije podatkov je tako predstavljala kopijo podatkovnega vira oziroma samo spremembe, glede na stanje prejšnjega polnjenja podatkovnega skladišča. Le-to se je potem preko področja priprave podatkov transformiralo v končno obliko, ki se je polnila v dejansko podatkovno skladišče, kjer se je shranjevala tudi zgodovina. Na ta način se sledi klasičnemu ETL konceptu procesa ekstrakcije, transformacije in polnjenja podatkovnega skladišča. Pri podatkovnem trezorju pa se podatki transformirajo v končno, uporabnikom prijazno obliko šele na koncu in zato sledi modernejšemu in vedno bolj uporabljanemu ELT procesu polnjenja, kot ga je Linstedt opisal tudi v spletnem članku ETL is Dead! Long Live ELT (2015b). V članku opisuje, da le redka oziroma samo največja podjetja dejansko sledijo *Big Data* viziji, ki premakne fazo procesiranja podatkov na del, kjer se podatki dejansko najbolj uporabljajo, torej bliže k uporabnikom in na končne podatkovne strukture. Pri naraščanju količine podatkov namreč postaja vedno težje te količine podatkov pravočasno obdelati »na zalogo«. Zato bo klasični ETL proces v prihodnosti nadomeščen z novim ELT procesom, pri čemer bodo metapodatki in njihova sledljivost do originalnega podatkovnega vira še vedno ostali pomembni.

5.3 Fleksibilnost in stabilnost pri dopolnitvah

Model podatkovnega trezorja je od vseh opisanih modelov najbolj fleksibilen in stabilen, kar se tiče naknadnih sprememb in dopolnitev, saj samo dopolnjuje. Enakega mnenja je tudi Maja Ferle (2015, str. 28): »Prednosti podatkovnega modela podatkovnega trezorja sta v možnosti dodajanja podatkov iz več različnih virov v enotno podatkovno strukturo in zagotavljanja stabilnosti rešitve v dolgih časovnih obdobjih.«

V tradicionalnih modelih dodajanje novih atributov ali vsebin vedno povzroči dodatne operacije na obstoječih vsebinah tako na atomarnem nivoju kot na dimenzijskem, da se obe vsebini s potencialno zgodovino lahko integrirata. Pri podatkovnem trezorju pa je integracija

novih vsebin bistveno enostavnejša kot na kratko povzema tudi Ferle (2015, str, 29): »Kadar se pojavi nov vir podatkov ali sprememba v obstoječem viru podatkov, ne spreminjamo obstoječega modela podatkovnega trezorja, zato se tudi procesi polnjenja podatkov ne spremenijo. Dodamo le nove tabele, ki ustrezajo spremembam, in njihove preslikave zapišemo v metapodatke.«

5.4 Nestrukturirani podatki

Nestrukturirane vire podatkov je navadno izredno težko naložiti v podatkovno skladišče. Zaradi same zasnove s središči in sateliti pa podatkovni trezor omogoča tudi deljene podatkovne zbirke. Tako se v središča lahko zapišejo samo primarni ključi iz nestrukturiranih virov, medtem ko ostali podatki, ki bi jih polnili v satelite, lahko dejansko ostanejo na izvoru. Tako je pri analizah možno preko središč spojiti informacije iz podatkovnega skladišča z zunanjimi nestrukturiranimi podatki kot satelitom.

5.5 Virtualizirane strukture in odzivnost analiz končnih uporabnikov

Osnovno ogrodje podatkovnega trezorja pa ni najbolj prijazno končnim uporabnikom, saj vključuje veliko tabel s svojimi zgodovinskimi verzijami, sama arhitektura pa je daleč od lahko razumljivih zvezdastih struktur po poslovnih področjih, katerim je prilagojena večina orodij za poročanje. Le-ta namreč večinoma predpostavlja dimenzijski model ali kocke OLAP. Ker pa se sam model poskuša čim bolj oddaljiti od prekomernega procesiranja na zalogo, predlaga izgradnjo virtualiziranih podatkovnih struktur oziroma pogledov (angl. *view*), ki že vključujejo vso kompleksno logiko. Klasične fizične tabele pa poskuša nadomestiti z materializiranimi pogledi (angl. *materialized view*), ki se vseeno osvežujejo. Poleg tega pogledi omogočajo tudi večjo fleksibilnost in prilagodljivost v primerjavi s klasičnimi tabelami, saj se prilagoditve uredijo samo na enem mestu. Tako dograjevanje in ponovno polnjenje dimenzijskih tabel ali tabel dejstev ter prilagajanje shranjenih procedur ni več potrebno.

Virtualizirane strukture so sicer zelo uporabne pri pripravi kompleksnih *ad hoc* raziskovanj, saj v njih že pripravljena koda predstavlja dobro izhodišče za zelo napredne poslovne analitike, ki pa morajo tudi sami razpolagati z dobrim, tako tehničnim kot vsebinskim, znanjem. Pri dejanski implementaciji podatkovnega skladišča pa povsod ne bi podprl ideje virtualiziranih struktur, saj bi to lahko zelo in nepotrebno podaljšalo izvajanje poročila na uporabnikovo zahtevo. Npr. pri področju prodaje za klasične količinske in zneskovne analize po izdelkih, njihovih skupinah, kupcih, lokacijah in prodajnih poteh so analize dokaj standardizirane in zato kompleksne možnosti povezovanja in upoštevanja vseh možnih presekov zgodovinskih verzij atributov pogosto niso potrebne. Obenem pa samo zmede večino končnih uporabnikov, ki so pogosto zadovoljni samo z zadnjo verzijo podatkov. Na tem mestu bi izpostavil, da je pri materializiranih pogledih potrebno upoštevati tudi določene tehnične omejitve strojne oziroma programske opreme. Kot primer bi navedel Netezzo, ki

ne uporablja klasičnih materializiranih pogledov, sestavljenih iz več tabel, saj se lahko sklicuje na eno samo, pri čemer omeji nabor podatkov posamezne tabele z izbiro le potrebnih stolpcev oziroma z vključenimi pogoji, kar zelo pospeši izvajanje poizvedb.

Virtualizirane strukture so lahko tudi samo vmesni vir pri polnjenju tabel dimenzij in tabel dejstev v zvezdastih shemah, oziroma se v primeru uporabe določenih poročilnih orodij, ki temeljijo na OLAP tehnologiji, lahko direktno povežejo na te strukture, saj si naredijo svoje interne strukture in agregacije. Vmesno polnjenje v tabele bi zato lahko bilo nepotrebno. To pa je že neke vrste hibridni model med podatkovnim trezorjem in klasičnim Kimballovim dimenzijskim modelom, ki ga je predlagal tudi Hans Michiels (2016). V njem predstavi prednosti dimenzijskega modela v primerjavi z ostalimi modeli in poudari predvsem naslednje točke, zapisane v nadaljevanju.

- Dimenzijski model je namenjen poročanju ter analiziranju podatkov po prerezih in pri tem je najboljši.
- Orodja za izdelavo poročil so za dimenzijski model bolj izpopolnjena in omogočajo več funkcionalnosti, ker je model v uporabi že desetletja.
- Performance poizvedbe so boljše v primerjavi z drugimi modeli, kar je ključno za uporabnike, ki želijo hitre odzive, da se jim ne prekine miselni tok.
- Zvezdni model je izvrsten vir za OLAP kocke pri večdimenzionalnih modelih.
- Primerjava časovnih obdobij je enostavnejša kot pri drugih modelih.
- Omogoča samopostrežno poslovno inteligenco, saj je zelo verjetno, da si bodo uporabniki lahko sami pripravili podatke v želeni obliki.

Kot glavne prednosti podatkovnega trezorja pa H. Michiels (2016) navaja:

- beleži vse podatke in zgodovino vseh sprememb med dvema polnjenjema (vse spremembe znotraj dneva so ena sprememba);
- ker so podatki enaki kot na viru z vsemi napakami in pomanjkljivostmi, brez vgrajenih poslovnih pravil, je zagotovljena sledljivost podatkov;
- omogoča integracijo z drugimi platformami (Hadoop) in decentralizirano podatkovno skladišče;
- izgradnja poteka in se lahko izvaja inkrementalno (agilno v sprintih), ker entitete nimajo direktnih povezav;
- zagotavlja dobro podporo za integracijo podatkov, ker so sateliti ločeni po virih.

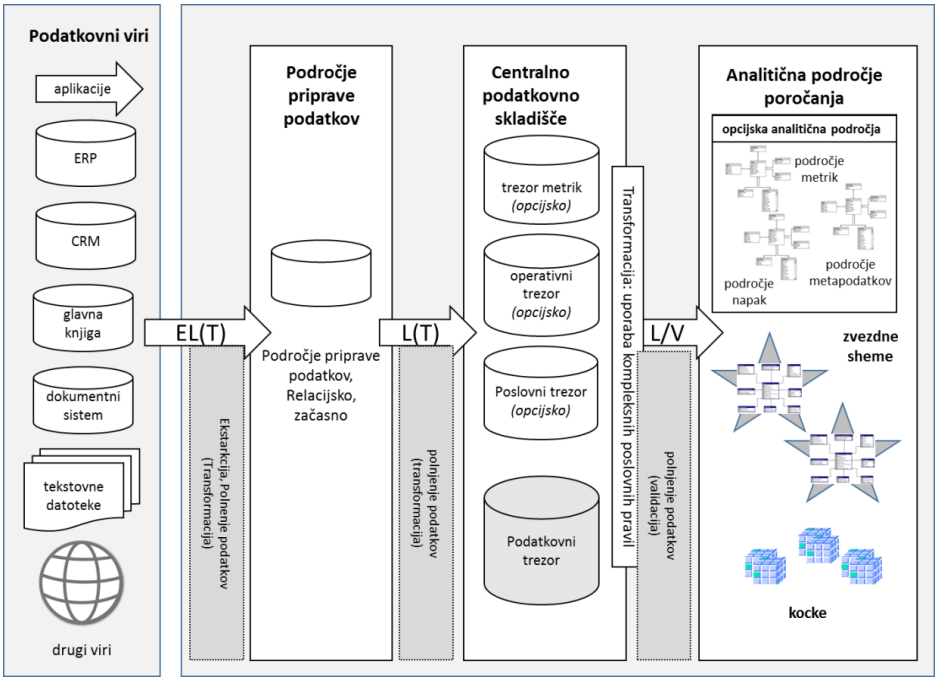
Z združitvijo podatkovnega trezorja in dimenzijskega modela pa H. Michiels (2016) ugotavlja:

- v primeru novih zahtev ali spremembe obstoječih zahtev se dimenzijski model lahko ponovno napolni iz modela podatkovnega trezorja, saj vsebuje vso zgodovino;

- zagotovljena je samo ena verzija resnice, saj v primeru več verzij dimenzijskih modelov pride do razlik zaradi različnih poslovnih pravil. Sam bi k temu dodal, da se pri tem energija porablja za dokazovanje, kdo ima prav in ne za reševanje dejanskih problemov, podatkovno skladišče pa v očeh uporabnikov izgubi na verodostojnosti;
- zgodovina ni izgubljena: z modelom podatkovnega trezorja se beleži vsa zgodovina sprememb iz podatkovnih virov, ki sicer vedno kažejo samo trenutno stanje. Brez sistema beleženja sprememb, kot je podatkovni trezor, je zgodovina izgubljena za vedno;
- dimenzijskemu modelu ni potrebno več skrbeti za dva namena – hranjenje podatkov in predstavitev informacij. Če trenutno ni zahteve po določeni vsebini v nobenem poročilu, je ta vsebina izpuščena iz zvezdaste sheme. Takoj, ko se pojavi potreba po novi vsebini, pa podatki niso na voljo in zgodovina ne obstaja. Pri kombinaciji obeh modelov se lahko vsebina z vso zgodovino enostavno doda;
- polna sledljivost je zagotovljena, saj model podatkovnega trezorja zajema nespremenjene podatke podatkovnega vira in je zato pot vseh mer in atributov na dimenzijah sledljiva do izvornih sistemov, vključno z datumi nastanka in spremembami. To omogoča, da se zavržejo morebitne obtožbe, da so podatki v podatkovnem skladišču napačni. Pokaže se lahko tudi, da je stara verzija podatkovnega skladišča, temelječa samo na zvezdasti shemi, kazala napačno.

Z večtirno arhitekturo podatkovnega skladišča je mogoče podatkovni trezor in dimenzijski model uspešno združiti.

Slika 6: Večtirna arhitektura z uporabo tehnik iz podatkovnega trezorja in dimenzionalnega modela



Vir: H. Michiels, 2016, Multi-tier architecture using both Data Vault and Dimensional Modelling techniques.

V prvem koraku se podatki iz podatkovnih virov prenesejo na področje priprave podatkov. Nato se spremembe napolnijo v »Enterprise data warehouse« atomarni nivo podatkovnega skladišča, temelječega na modelu podatkovnega trezorja. Od tu se podatki na podlagi poslovnih pravil polnijo v analitični nivo, ki temelji na dimenzijskem oziroma zvezdastem modelu, kar je prikazano tudi v Slika 6.

5.6 Kompleksnost modela in stroškovni vidik razvoja

Model je vsebinsko podobno zasnovan kot model vozlišča in priključkov ali sidrni model, saj vsebuje nivo osrednjega podatkovnega skladišča po vsebinskih entitetah in nivo poročanja. Glavna razlika pa je, da je osrednji nivo izredno enostavno zastavljen z vidika polnjenja in transformacij podatkov, saj le-teh ni. Ker je polnjenje brez transformacij tako enostavno, je ETL proces prenosa podatkov iz izvornih sistemov mogoče standardizirati, da samo na podlagi parametrov samodejno izvede preslikavo. Na ta način ni potrebno vsakega ETL procesa programirati in nastavljati ročno, temveč se uporablja generični ETL proces, kar zelo zniža stroške razvoja. Vendar je zaradi te prednosti pri polnjenju toliko bolj težavno na nivoju poročanja, ker je potrebno na tem delu podatke vseeno prečistiti, urediti in transformirati v uporabniku prijazno obliko, saj tega uporabnik ni sposoben sam. Za transformacijo podatkov pa so uporabljene virtualne strukture, ki vključujejo celotno logiko transformacij in so v uporabniku prijazni obliki, kar je nivo poslovnega podatkovnega trezorja.

6 SIDRNI MODEL

Glavne značilnosti sidrnega modela so, da se podatki, ko so enkrat v podatkovnem skladišču, ne spreminjajo več, ampak samo nastajajo nove verzije. Model je zelo agilen in razširljiv ter omogoča beleženje zgodovine z minimalno porabo prostora na disku in zato načeloma hitrejše poizvedbe.

6.1 Projektni pristop in razvojna metodologija

Projektni pristop je podoben kot pri Inmonovem celovitem podatkovnem skladišču in konceptu podatkovnega trezorja, torej od spodaj navzgor, saj predvideva najprej vzpostavitev celovitega podatkovnega skladišča in šele nato podatkovne strukture za poročanje. Zaradi tega je prav tako smiselno uporabiti spiralno ali Scrum agilno metodologijo razvoja ter v primeru zunanjega izvajalca pogodbo tipa čas in gradivo, kar je opisano že v poglavjih 4.1 in 5.1.

6.2 Kompleksnost modela

Ker je model s toliko različnimi strukturami in vlogami zelo kompleksen in nepregleden, je zelo pomembna dokumentacija modela. Rönnbäck et al. (2010) v ta namen predlagajo kar orodje, ki prikazuje diagrame s poslovno vsebino. Ker se posamezne entitete lahko pojavijo večkrat in določeni atributi pripadajo samo določenim entitetam, jih je zaradi večje preglednosti potrebno šifrirati z govorečimi šiframi, ki odražajo njihovo logično vsebino. Za to pa je potrebno definirati dogovor o načinu poimenovanja. To pomeni, da je potrebno za vsak sinonim oziroma izraz definirati okrajšavo, ki se bo uporabila tako pri kreiranju tabel kot pri poimenovanju poslovnih tipov in vlog.

6.3 Beleženje zgodovine, hitrost in način ETL polnjenja

Glavna prednost in hitrost modela se pokaže šele pri zelo velikih količinah podatkov, kjer se pridobi predvsem na zgodovinskih in vozelnih zgodovinskih atributih. Ob spremembi vrednosti posamezne lastnosti entitete se doda samo dodaten zapis za novo vrednost tega atributa, kar predstavlja bistveno manjši prirast prostora na disku, kot če bi se podvojila cela vrstica z vsemi atributi zaradi spremembe enega samega atributa, kot je to npr. pri počasi se spreminjajoči dimenziji tipa 2 v dimenzijskem modelu oziroma pri satelitih pri podatkovnem trezorju. To lahko plastično predstavim na hipotetičnem primeru beleženja zgodovine na škodnem spisu, ki ima npr. 15 razmeroma nespremenljivih opisnih atributov podatkovnega tipa varchar (256), 5 statičnih vezi na druge entitete, kot so škodni dogodek, produkt, zavarovalna podvrsta, zavarovalna enota in vrsta škode. Poleg tega pa ima še vozelni zgodovinski atribut 'Status škodnega spisa', ki ima predvidenih 10 vrednosti, preko katerih gre večinoma vsak škodni spis ali se lahko celo naknadno reaktivira in se vrne na začetek. V tem primeru bi za 10 sprememb statusa v sidrnem modelu pridelali 10 zapisov v ozki tabeli Status škodnega spisa, kar zavzame zelo malo prostora. V dimenzijskem modelu ali podatkovnem trezorju pa bi za iste spremembe 10-krat multiplicirali vse atribute na entiteti oziroma dimenziji škodni spis, kar predstavlja kar nekajkrat večji obseg podatkov. Če se sidrni model vzpostavi na Netezzi, bi bila razlika v odzivnih časih poizvedbe o času trajanja reševanja škodnega primera po statusih zelo velika v primerjavi s podatkovnim trezorjem. Na tem specializiranem strežniškem sistemu za podatkovna skladišča se poizvedbe na tabelah z manjšimi nabori stolpcev namreč močno skrajšujejo z zmanjševanjem števila stolpcev, saj operira z bistveno manjšim prostorom na disku.

6.4 Fleksibilnost in stabilnost pri dopolnitvah

Osnovni koncept sidrnega modela ja, da je razširljiv in prilagodljiv, saj je tako normaliziran, da dodajanje novih vsebin ali dopolnjevanje ne vpliva na obstoječe podatke, saj se lahko dodajajo nove vsebine kot samostojna sidra, ki jih je mogoče povezati z obstoječimi, in sicer preko novih zgodovinskih ali statičnih vezi. Atribute in dodatne šifrante pa je prav tako

mogoče prosto dodajati v vozle, saj se neodvisno od ostalih vsebin povezujejo na sidra preko zgodovinskih ali statičnih vozelnih atributov.

6.5 Nestrukturirani podatki

Sidrni model omogoča s svojo zasnovo tudi povezovanje nestrukturiranih podatkov, saj je mogoče kakršen koli tip podatka kot sidro ali kot njegovo lastnost dodati k obstoječemu ali novemu sidru.

Nestrukturirani podatki pa predstavljajo izziv, ko se nahajajo v zelo velikih količinah, kar je *Big Data* koncept, ki ga tudi Golev & Rönnbäck (2011, str. 2) opredelita s tremi V-ji: kot **velike količine podatkov** (angl *Volume*), **velika raznolikost** (angl. *Variety*), ki predstavlja različne formate oziroma nestrukturirane podatke, in **visoka hitrost** (angl. *Velocity*), ki pomeni hitrost v smislu hitrega generiranja zajemanja in uporabe podatkov. Primer takih podatkov so datoteke dnevnikov (angl. *log files*), ki na spletnih strežnikih beležijo različne aktivnosti uporabnikov, kot so obiski strani, zaporedje klikov in druge aktivnosti.

Golev & Rönnbäck (2011) v svojem članku preizkusita tudi masovne podatke na dejanskem primeru analiziranja sekvence klikov na portalu spletnega trgovca Avito. Z zelo dobrimi rezultati argumentirata izredno učinkovitost tehnike modeliranja visoko normaliziranih nestrukturiranih podatkov s sidrnim modelom, ki zaradi svojega koncepta po izredno učinkovitem načinu shranjevanja podatkov omogoča izredno hitre poizvedbe. Nestrukturirane podatke, kot del *Big Data*, je torej mogoče shranjevati in učinkovito analizirati tudi s sidrnim modelom.

6.6 Odzivnost pri analizah končnih uporabnikov

Analize končnih uporabnikov, ki bi temeljile na direktnih poizvedbah v sidrni model, bi bile nedvomno daljše, ko če bi se izvajale na Kimballovem dimenzijskem modelu v zvezdasti strukturi.

To je Němec (2012) tudi dokazal v primerjavi hitrosti izvajanja poizvedb sidrnega in dimenzijskega modela, kjer je testiral enako strukturo podatkov z istimi podatki. Prišel je do ugotovitev, da sidrni model, kljub dodani vrednosti s področja modeliranja informacij, ni dosegel hitrosti, ki jo ima dimenzijski model.

7 PRIMERJAVA PODATKOVNIH MODELOV

Uspešnost in učinkovitost podatkovnih modelov je pogosta tema zagovornikov specifičnih podatkovnih modelov, zato so bile na tem področju že narejene primerjave. Ariyachandra & Watson (2008) tako povzemata raziskavo več strokovnjakov, ki pravi, da so najbolj uspešni

projekti podatkovnih skladišč še vedno dimenzijski model, model vozlišč in priključkov in nasploh centraliziranimi modeli. Vsi vključeni v raziskavo pa se strinjajo, da je najbolj pomembno, da so prvi rezultati hitro vidni. Z vidika trajanja in stroškov najdaljši in najdražji projekti navadno temeljijo na modelu priključkov in vozlišč.

Glede zmagovalnega modela v raziskavi pa Ariyachandra (2013, str. 147) povzema: »Na splošno smo ugotovili, da lahko glavne arhitekture podatkovnih skladišč dostavljajo dobro kakovost informacij, sistema ter vpliv na posameznika in organizacijo. Raziskava ni našla absolutnega "zmagovalca" v "vojni arhitekturnih modelov podatkovnih skladišč", saj ne obstaja. Metrike uspešnosti izdelka so si zelo podobne med dimenzijskim modelom, modelom vozlišča in priključkov ter centraliziranih arhitektur. Podjetja lahko izberejo arhitekturo na podlagi drugih pomembnih dejavnikov kot so razpoložljivost kadrov, nujnosti potrebe po podatkovnem skladišču, strateškega vidika managementa na podatkovno skladišče, organizacijskega ustroja, kompatibilnostjo obstoječih sistemov in tehnologij, predlogov svetovalcev in drugih.«

Po podrobnih opisih značilnosti obravnavanih podatkovnih modelov in primerjavah po opredeljenih dejavnikih je smiselno glavne značilnosti povzeti čim bolj jedrnato in ponuditi hiter pregled za lažjo odločitev o potencialno primernih modelih glede na situacijo v organizaciji, ki načrtuje podatkovno skladišče. Tabela 4 ponuja pregled ugotovljenih generalnih značilnosti in priporočil, ki bi glede na pomembnost posameznih kriterijev lahko pomagala pri izbiri najprimernejšega modela.

Tabela 4: Primerjava modelov po kriterijih

	Dimenzijski model	**Celovito podatkovno skladišče**	**Podatkovni trezor**	**Sidrni model**	**Hibrid podatkovni trezor/dimenzijski model**
Projektni pristop	od zgoraj navzdol, kaskadni model, hitra uvedba	od spodaj navzgor, spiralni model, dolga uvedba	od spodaj navzgor, spiralni model ali agilna metoda scrum	od spodaj navzgor	od spodaj navzgor, spiralni model ali agilna metoda scrum, pogojno kaskadni
Tip pogodbe podizvajalca	na ključ	čas in gradivo, navadno višji stroški	čas in gradivo, pogojno na ključ v kasnejših fazah	čas in gradivo	čas in gradivo ali na ključ v kasnejših fazah

se nadaljuje

Tabela 4: Primerjava modelov po kriterijih (nad.)

	Dimenzijski model	**Celovito podatkovno skladišče**	**Podatkovni trezor**	**Sidrni model**	**Hibrid podatkovni trezor/dimenzijski model**
Zgodovina podatkov	v dimenzijah obstaja zgodovina samo za predhodno zahtevane poslovne vsebine	vsa zgodovina se vodi primarno v atomarnem delu	vsa zgodovina se vodi v atomarnem delu	vsa zgodovina se vodi v atomarnem delu	vsa zgodovina se vodi že v atomarnem delu, v dimenzijskem pa samo kjer je nujno potrebna
Dopolnitve – fleksibilnost in stabilnost	možne težave pri razširitvah dimenzij z neobstoječo zgodovino novih atributov	vsa zgodovina in dopolnitve se vodi v atomarnem delu	vsa zgodovina in dopolnitve se vodi v atomarnem delu, najbolj fleksibilen in odprt model	vsa zgodovina in dopolnitve se vodi v atomarnem delu	dimenzijski model se lahko ponovno napolni iz atomarnega tudi ob strukturnih spremembah
Kompleksnost modela	za končne uporabnike najbolj enostaven in razumljiv model	srednje kompleksno, povezovanje struktur na atomarnem nivoju, vendar enostavno na dimenzijskem	atomarni model zelo enostaven, vendar kompleksne virtualne strukture pri pripravi poročilnega področja	zelo kompleksne strukture; kompleksen koncept lahko povzroči zastoje na začetku zaradi nerazumevanja	za končne uporabnike zelo enostavno zaradi dimenzijskega modela
ETL, čiščenje in integracija	podatki se čistijo in transformiraj o pred samim polnjenjem	Podatki se transformiraj o pred samim polnjenjem	Zelo hitro pri polnjenju atomarnega dela, a brez čiščenja in transformacije, Metoda ELT z uporabo generičnega polnjenja	polnjenje je zelo hitro, odzivnost pa zelo visoka pri velikih količinah podatkov z veliko spremembami	podatki se transformirajo že pred samim polnjenjem v atomarni nivo in dodatno ob polnjenju v dimenzijski model

se nadaljuje

Tabela 4: Primerjava modelov po kriterijih (nad.)

	Dimenzijski model	**Celovito podatkovno skladišče**	**Podatkovni trezor**	**Sidrni model**	**Hibrid podatkovni trezor/dimenzijski model**
Odzivnost poročanja	najhitrejša odzivnost	zelo hitra odzivnost v področnem nivoju v dimenzijskem modelu	možna je počasnejša odzivnost pri standardnih analizah zaradi kompleksnih virtualnih pogledov, ki hkrati prečiščujejo podatke	v kolikor se nadgradi z dimenzijskim modelom, hitra odzivnost pri končnih uporabnikih, sicer lahko počasna zaradi kompleksnih poizvedb	zelo hitra odzivnost zaradi hibridnega pristopa vključitve dimenzijskega modela
Nestrukturirani podatki	predlaga dimenzioniranje nestrukturiranih podatkov ob vključitvi v podat. skladišče; predlog uporabe namenskih orodij, ki lahko analizirajo strukturirane in nestrukturirane podatke	predlaga povezovanje nestrukturiranih virov na podlagi tekstovnih algoritmov	model je najbolj primeren za skladiščenje nestrukturiranih podatkov saj je izrdeno prilagodljiv in odprt	model je zelo odprt za nestrukturirane podatke, poizvedbe pa so lahko hitre tudi na nestrukturiranih podatkih	model je zelo odprt za nestrukturirane podatke, predlaga tudi dimenzioniranje nestrukturiranih podatkov ob vključitvi v poročilni del, nova orodja za analizo strukturiranih in nestrukturiranih podatkov

Poleg predlogov modelov pri kombinacijah kriterijev je dobro opozoriti tudi na določene kombinacije, ki se jim je dobro izogniti, saj obstaja večja nevarnost za neuspešen projekt.

Primer take kombinacije je celovito podatkovno skladišče s spiralno projektno metodologijo pri vključenem zunanjem izvajalcu s projektom na ključ. Glavni razlog je, da je prva faza, ko se gradi celovit atomarni nivo podatkovnega skladišča, zelo dolga in zato dolgo časa ni pravih rezultatov. To pa lahko povzroči nezaupanje v uspešnost projekta in s tem dodatne pritiske s strani uporabnikov.

Kombinacija sidrnega modela in agilne projektne metodologije ob slabi dokumentaciji lahko povzroči prekomerno podaljšanje in višje stroške ob kasnejših dopolnitvah podatkovnega skladišča, če se vključujejo novi člani, ki potrebujejo dolgo časa, da spoznajo celoten kompleksen model.

V primeru, ko ima posamezna poslovna vsebina zelo veliko časovno neodvisnih atributov in samo en ali zelo malo število časovno zelo spremenljivih atributov, pa lahko razlike v porabi količine diskovnega prostora postanejo zelo očitne med različnimi modeli. Te razlike se potem kažejo tudi v zelo različnih odzivnih časih poizvedb ter stroških, povezanih z diskovnim prostorom, procesorsko močjo ter stroški programskih licenc, vezanih na procesorsko moč, kar pa zelo vpliva na izbiro primernega modela. Tako se lahko pojavi velika razlika med sidrnim modelom in modelom vozlišča in priključkov.

SKLEP

Prvi cilj naloge je bil definirati nabor kriterijev, ki bodo omogočili kar najbolj objektivno primerjavo različnih modelov in njihovih posebnosti. Opredeljenih sedem primerjalnih področij je uspešno pokrilo tako metodološke, tehnične kot poslovne vidike najbolj pogostih podatkovnih modelov, ki so bili analizirani v nalogi.

Glavni cilj naloge je bil preko prvotno definiranih kriterijev in dejavnikov ponuditi hitro primerjavo modelov in morebiti poiskati najustreznejši model, ki bi prevladoval pred drugimi. S priporočili in poudarjenimi glavnimi prednostmi ali slabostmi pa je namen olajšati izbiro najprimernejšega modela za izbrano organizacijo, ki se nahaja v procesu odločanja za vzpostavitev podatkovnega skladišča.

Iz naloge ugotavljam, da so si modeli po značilnostih dokaj različni in da ni mogoče z gotovostjo in enoznačno trditi, kateri je najprimernejši. Delno je mogoče predvideti, glede na splošne zakonitosti posamezne panoge, kakšno količino podatkov na posameznem področju se lahko pričakuje in kakšne podatke se navadno analizira. Za določene poslovne vsebine pa bi bilo mogoče tudi z manjšimi zadržki delno predvideti spremenljivost teh podatkov, kar pri izbiri vpliva tudi na učinkovito uporabo diskovnega prostora kot najpočasnejšo strojno komponento, ki pa je različno obremenjena glede na uporabljen model. Na podlagi teh informacij bi bil predlog lahko še bolj verodostojen.

Preko vsebine in kronološkega nadgrajevanja modelov v publikacijah ugotavljam, da se pri modernejših modelih vedno bolj osredotoča tudi na nestrukturirane podatke, ki jih je celo Inmon vključil kot poglavje v zadnjo izdajo knjige o gradnji podatkovnih skladišč. Tako bi zaradi vse večje težnje po analizi nestrukturiranih podatkov in hkrati vse hitrejši strojni opremi lahko imela v prihodnosti prednost sidrni model in podatkovni trezor kot hibridni model v kombinaciji z dimenzijskim modelom. Vendar je področje masovnih podatkov še vedno v iskanju najprimernejše platforme, saj se še vedno razvijajo nove ideje v vse smeri, tako da je nadaljnjo smer razvoja trenutno nemogoče predvideti.

V prihodnosti se bo področje poslovnih analiz razvijalo predvsem v smeri obvladovanja masovnih podatkov. Razvoj orodij za podporo odločanja gre trenutno v vse smeri, zato je težko predvideti, katera bo prava. Klasična orodja za podporo odločanja, ki temeljijo na

OLAP kockah in dimenzijskih strukturah, se bodo v prihodnosti morala prilagoditi tudi nestrukturiranim podatkom. Še večja verjetnost pa je, da jih bodo izpodrinila orodja, ki bodo podpirala hitre in kompleksne analize na nestrukturiranih podatkih in bodo lahko uporabila tudi strukturirane podatke iz podatkovnih skladišč.

Tudi razvoj podatkovnih skladišč se bo še naprej usmerjal v obvladovanje nestrukturiranih podatkov decentraliziranih podatkovnih skladišč z nestrukturiranimi podatki v zunanjih bazah in modeliranja, ki bo omogočalo čim bolj učinkovito hranjenje zgodovine, minimiziranje transformacij in procesiranja transformacij na zalogo ter hitre odzivne čase analiziranja. Na tem področju ocenjujem, da imata največjo prednost podatkovni trezor in sidrni model oziroma njune hibridne izpeljave z vključitvijo dimenzijskega modela in virtualnih struktur.

Sklepam, da je cilj naloge delno dosežen, saj ponudi pregled nabora modelov in olajša izbiro po glavnih značilnostih. Samo analizo modelov in predlog bi bilo mogoče še izboljšati z dejansko implementacijo istih poslovnih zahtev v vsakem od modelov podatkovnih skladišč in izvesti različne eksperimente, tako z merjenjem odzivnih časov enakih poročil, merjenjem časa polnjenja in transformacije podatkov v podatkovno skladišče kot merjenjem časa uvajanja novih članov na projekt ter merjenja časa in stroška za izvedbo enakih dodatnih zahtev.

Analizo bi bilo mogoče nadgraditi še z dodatnimi modeli, njihovimi hibridi ter dodatnimi kriteriji in vprašalniki s področja strategije organizacije, njene velikosti panoge, trenutno zrelostjo oddelka poslovne analitike in njenega prihodnjega razvoja ter drugih kriterijev, ki bi se pokazali za relevantne na obravnavano temo. Na podlagi vnaprej pripravljenih še podrobnejših analiz in dobrih praks po svetu, in sicer v kombinaciji z odgovori na vprašanja, bi odločevalca lahko vodila k izbiri ustreznega modela oziroma odstranila neprimerne iz nabora.

LITERATURA IN VIRI

1. Ariyachandra, T. & Watson, H. J. (2008). Technical opinion: Which data warehouse architecture is best. *Communications Of The Acm*, *51*(10), 146-147.
2. Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria, *International Journal of Project management*, *17*(6), 337-342.
3. Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. Najdeno 30. maja 2016 na spletni strani http://csse.usc.edu/TECHRPTS/1988/usccse88-500/usccse88-500.pdf
4. Ferle M. (2015, april). Podatkovni trezor, *Monitor*, str. 28-29.
5. Golov N. & Rönnbäck, L. (2011). Big Data Normalization for Massively Parallel Processing Databases. Najdeno 21. septembra 2016 na spletni strani http://www.anchormodeling.com/wp-content/uploads/2011/05/Big_Data_Normalization.pdf
6. IBM Corporation (2011). *IBM Insurance Information Warehouse model 8.4.* Armonk: IBM Insurance Information Warehouse model
7. Inmon, W. H. (2002). *Building the data warehouse* (3rd ed.). New York: John Wiley & Sons.
8. Inmon, W. H. (2005). *Building the Data Warehouse* (4th ed.). New York: John Wiley & Sons.
9. Jaklič, J. (2002). *Upravljanje in uporaba podatkov*. Ljubljana: Ekonomska fakulteta.
10. Kimball, R. (2012). Newly Emerging Best Practices for Big Data. Kimball Group. Najdeno 20. septembra 2016 na https://www.informatica.com/content/dam/informatica-com/global/amer/us/collateral/white-paper/big-data-emerging-best-practices_white-paper_2181.pdf
11. Kimball, R. & Ross, M. (2013). *The data warehouse toolkit* (Third edition). Indianapolis: John Wiley & Sons.
12. Krishnan, K. (2013). *Data Warehousing in the Age of Big Data*. Amsterdam: Elsevier.
13. Linstedt, D. (2015a, 18. marec). Data Vault Basics. Najdeno 5. aprila 2016 na spletnem naslovu http://danlinstedt.com/solutions-2/data-vault-basics/
14. Linstedt, D. (2015b, 29. maj). ETL is Dead! Long Live ELT. Najdeno 5.aprila 2016 na spletnem naslovu http://danlinstedt.com/allposts/datavaultcat/etl-is-dead-long-live-etl/
15. Michiels, H. (2016, 2. april). Data vault and dimensional modelling, a happy marriage!, *Data vault series*. Najdeno 5. aprila 2016 na spletnem naslovu http://www.hansmichiels.com/2016/04/02/data-vault-and-dimensional-modelling-a-happy-marriage-data-vault-series/
16. Moškon, S. (2009). Kako uničiti IT projekt v 10 korakih – ali revizor IS lahko to prepreči?. *17. mednarodna konferenca o reviziji in kontroli informacijskih sistemov,*

Rogaška Slatina. Najdeno 21. septembra na spletni strani http://www.vris.si/Db/vris/content/pdf/Prispevek_za_17_konferenco_StaneMoskon.pdf

17. Nemec, R. (2012). The Comparison of Anchor and Star Schema from a Query Performance Perspective. *International Journal of Computer, Electrical, Automation, Control and Information Engineering, 6*, (11), 1421-1425.
18. Rönnbäck, L., Regardt, O., Bergholtz, M., Johannesson, P., Wohed, P. (2010, 5. oktober). Anchor Modeling - Agile Information Modeling in Evolving Data Environments. Najdeno 17. aprila 2016 na spletnem naslovu http://www.anchormodeling.com/wp-content/uploads/2011/05/Anchor-Modeling.pdf
19. Royce, W. (1970). *Managing the Development of Large Software Systems: Concepts and Techniques.* Los Angeles: Western Electronic Show and Convention.
20. Schwaber, K. & Suhterland, J. (2011). Ultimativni Scrum vodič: pravila igre. Najdeno 19. maja 2016 na spletni strani http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-SI.pdf
21. Theissen M. & Kraemer E. (2009). Hub-And-Spoke: Building an EDW with SQL Server and Strategies of Implementation. Najdeno 15. junija 2016 na spletnem naslovu https://technet.microsoft.com/en-us/library/dd459147%28v=sql.100%29.aspx