Evolucija skladišta podataka: pregled istraživanja, metoda i tehnika

Danijela Subotić
Sveučilište u Rijeci/ Odjel za informatiku, Rijeka, Hrvatska
dsubotic@inf.uniri.hr

Sažetak - Skladište podataka integrira brojne heterogene izvore podataka te omogućuje brzu i efikasnu analizu za potrebe poslovanja. Međutim, današnji izvori podataka često mijenjaju svoj sadržaj i strukturu, što uvelike utječe i na skladište podataka – ono uvijek mora sadržavati najnovije informacije, kako bi moglo odražavati trenutno stanje u stvarnome svijetu. Upravo iz toga razloga potrebno je ispravno upravljati svim tipovima promjena te prikladno ažurirati skladište podataka. Problem evolucije skladišta podataka promatramo odvojeno od problema evolucije baza podataka. Zahtjevi skladišta podataka su povećani u pogledu opsega i pamćenja promjena, u odnosu na baze podataka. Kod evolucije baza podataka, istraživanja su usmjerena na pamćenje stanja baze podataka. Međutim, problem koji se istražuje kod evolucije skladišta podataka je pamćenje promjena opsega te strukture podataka i meta-podataka, u dužem vremenskom periodu. Opseg skladišta podataka danas je sve širi, odnosno obuhvaća više izvora i više tipova podataka. Također, da bi skladište podataka pružilo odgovarajuću potporu procesima odlučivanja, ono mora obuhvaćati povijesne podatke i projekcije trendova, a ne samo poslijednje ažurirano stanje (kao što je slučaj kod baza podataka). Zbog ubrzanih poslovnih promjena i promjena u tehnologiji, koje obuhvaćaju sve šira područja integracije, postaje sve važnije pronaći odgovarajuće rješenje ovoga problema. U dosadašnjim istraživanjima evoluciju skladišta podataka uglavnom možemo pratiti kroz tri pristupa – evoluciju sheme, verzioniranje sheme i održavanje pogleda. Cilj ovoga rada jest prikazati pregled različitih radova i istraživanja vezanih uz problem evolucije skladišta podataka te sugerirati rješenje navedenog problema za buduća istraživanja.

Ključne riječi: evolucija skladišta podataka, evolucija sheme, verzioniranje sheme, održavanje pogleda

I. UVOD

Problem korištenja višestrukih heterogenih shema javio se već kod rada s relacijskim i objektno-orijentiranim bazama podataka. Unatoč promjenama u strukturi baze podataka, potreba za zadržavanjem postojećih podataka je ostala - imperativ je da podaci prežive promjene u shemi baze podataka. Kao rješenje ovoga problema, pojavila su se dva pristupa – evolucija sheme i verzioniranje sheme. Oba pristupa zasnivaju se na uvođenju inteligentnog upravljanja vremenskim neusklađenostima između podataka i strukture podataka. Kod evolucije sheme pohranjuje se samo trenutna verzija sheme, dok se kod verzioniranja sheme pohranjuje više odabranih verzija sheme, uz korištenje simboličkih oznaka ili, češće, oznaka datuma i vremena. Zapravo evoluciju sheme, radi jednostavnosti, možemo smatrati posebnim slučajem verzioniranja sheme. Roddick u [1] daje izvrstan pregled glavnih pitanja i karakteristika ova dva pristupa na razini relacijskih i objektno-orijentiranih baza podataka.

U slučaju skladišta podataka, u literaturi razlikujemo tri pristupa evoluciji skladišta podataka – evolucija sheme ([8]-[20]), verzioniranje sheme ([21]-[30]) i održavanje pogleda ([31]-[44]). Prva dva pristupa se zasnivaju na skladištu podataka definiranom kao višedimenzionalna shema, dok se održavanje pogleda zasniva na skladištu podataka koje je postavljeno kao skup materijaliziranih pogleda. Cilj ovoga rada jest napraviti pregled istraživanja raznih autora, vezanih uz problem evolucije skladišta podataka.

Struktura rada je sljedeća - u 2. poglavlju prikazani su i opisani različiti radovi i istraživanja vezana uz evoluciju skladišta podataka, u 3. poglavlju predstavljene su smjernice za buduća istraživanja, a u 4. poglavlju nalazi se zaključak.

II. PREGLED RADOVA I ISTRAŽIVANJA

Kao što je već spomenuto, problem evolucije sheme je prisutan i kod baza podataka, gdje su istraživanja uglavnom usmjerena na pamćenje stanja baze podataka.

Zanimljiva su i neka rana istraživanja poput [2], u kojem je predstavljen model verzija za evoluciju sheme baze podataka, [3] u kojem autori definiraju povijesni relacijski model podataka uz pripadajući skup temporalnih operatora nad oznakama datuma i vremena i intervalima, te [4] u kojem autori predstavljaju PRISM, sustav koji podupire evoluciju sheme baze podataka, tako što pruža jezik operatora za izmjenu sheme (SMO) za opis kompleksnih promjena sheme te povećava predvidivost evolucije kroz automatsku provjeru očuvanja informacija, redundantnosti i podrške upitima. Također, PRISM omogućuje automatsku migraciju podataka te potpunu dokumentaciju procesa evolucije sheme,

U [5] i [6] autori dalje proučavaju problem temporalnih baza podataka, odnosno korištenja temporalnih podataka u relacijskim bazama podataka. U [5] autori su opisali osnovne probleme temporalnih podataka te su predstavili temeljne konstrukte i operatore za rješavanje tih problema. U [6] autori su predstavili Asertivno verzioniranje, konceptualni pristup pomoću kojeg se smanjuje trošak i povećava vrijednost temporalnih podataka.

U odnosu na baze podataka, zahtjevi skladišta podataka su povećani u pogledu opsega i pamćenja

promjena te su i istraživanja problema evolucije skladišta podataka nešto šireg opsega. U [7] dan je kratki pregled istraživanja unutar navedena tri pristupa evoluciji skladišta podataka.

Evolucija sheme pretpostavlja da shema može imati samo jednu verziju u određenom vremenu – onu trenutnu. Svi podaci su pohranjeni u trenutnoj verziji te se sve promjene vrše nad trenutnom verzijom, koja se zatim transformira u novu verziju. Problem ovoga pristupa je gubitak povijesti – ne čuvaju se prethodne verzije pa prethodno dostupne strukture i podaci u novoj verziji mogu postati nedostupni. Evoluciju sheme možemo promatrati kao slabiji slučaj verzioniranja sheme.

Verzioniranje sheme pretpostavlja da više verzija sheme mogu biti važeće u različitim vremenskim intervalima. Verzioniranjem sheme čuva se povijest verzija – promjene u shemi skladišta podataka uzrokuju kreiranje nove verzije sheme, kojoj se dodjeljuje vremenska oznaka. Čuvaju se sve verzije sheme skladišta podataka.

Održavanje pogleda pretpostavlja da je skladište podataka skup materijaliziranih pogleda, direktno napunjenih podacima iz izvora podataka. Materijalizirane poglede zatim treba osvježavati i prilagođavati, ovisno o raznim promjenama u izvorima podataka.
U nastavku slijedi pregled radova i istraživanja vezanih uz navedene pristupe.

A. Evolucija sheme

Ovaj pristup odnosi se na skladište podataka definirano kao višedimenzionalna shema s tablicama činjenica i dimenzija te podatkovnim kockama. Evolucija sheme sadrži više različitih razina ažuriranja, poput ažuriranja dimenzija, instanci, razina, činjenica, atributa, podatkovnih kocki, hijerarhija, mjera, ograničenja, kvalitete ili strukture. Naravno, neke vrste ažuriranja predstavljaju više statički aspekt razvoja skladišta podataka (poput ažuriranja dimenzija), dok neke vrste ažuriranja predstavljaju više dinamički aspekt razvoja skladišta podataka (npr. ažuriranje strukture). Kada se u literaturi govori o evoluciji sheme, naglasak je na ažuriranju dimenzija, instanci, razina, hijerarhija, činjenica i atributa.

U [8] autori su, uz 5 operatora za ažuriranje dimenzija, predložili i 2 operatora za ažuriranje instanci: Generalizirati, Specijalizirati, Povezati razine, Nepovezane razine, Brisati razinu, Dodati instancu i Brisati instancu. Kako bi kontrolirali utjecaj strukturalnih promjena u dimenziji na podatkovnu kocku, autori su predložili prilagodbu podatkovne kocke nakon korištenja operatora za dodavanje i brisanje dimenzije te dodavanje i brisanje instance. Za svaki pogled podatkovne kocke, predložili su izračunavanje izraza za njegovo održavanje. U [9] autori su dodatno predložili 4 složena operatora za ažuriranje instanci.

U [10] autori su definirali formalni okvir za opis evolucije višedimenzionalnih shema, koji sadrži opis samih shema i instanci. Model je definiran na sljedeći način: MD model M je 6-torka (F, L, A, Gran, Class, Attr) gdje je F konačan skup naziva činjenica, L je konačan skup naziva razina dimenzija, A je konačan skup naziva atributa, Gran je funkcija koja povezuje činjenicu sa skupom naziva razina dimenzija, Class je relacija definirana na nazivu razine, Attr je funkcija koja veže atribut uz određenu činjenicu ili razinu dimenzije. Nakon što su definirali model, autori su predložili i skup formalnih operacija evolucije skladišta podataka. Ideja je bila da se dizajneru omogući da odredi željenu evoluciju sheme na konceptualnoj razini, nakon čega slijedi automatska prilagodba implementacije. Međutim, autori su se fokusirali samo na promjene sheme skladišta podataka koje nastaju zbog promjene poslovnih zahtjeva.

U [11] autori su predložili proširenje rada iz [8] i [9], tako što su predložili alat za vizualizaciju za dimenzije i podatkovne kocke.

U [12] autori su definirali sljedećih 6 operatora za evoluciju sheme: Dodati atribut, Brisati atribut, Preimenovati atribut, Dodati tablicu, Brisati tablicu, Preimenovati tablicu.

U [13] autori su predložili mehanizam za dobivanje logičke sheme skladišta podataka putem unaprijed definiranih transformacija koje se primjenjuju na logičke sheme izvora. Ovime se omogućuje praćenje i dokumentiranje dizajna i povezivanja (eng. *mapping*) između logičkih struktura skladišta podataka i izvora podataka. Mehanizam sadrži dvije vrste pravila – pravila konzistentnosti za osiguranje doslijednosti dobivene sheme, te strategije dizajna koje sadrže različita rješenja za dizajniranje skladišta podataka. Također, autori su predložili skup transformacija s njihovim opisima i primjerima upotrebe.

U [14] autori su predstavili AutoMed, heterogeni sustav za transformaciju i integraciju podataka, koji ima sposobnost integracije podataka preko višestrukih modela podataka. Shema se inkrementalno mijenja, kako se na nju primjenjuje niz primitivnih transformacija, a svaka primitivna transformacija dodaje, briše ili preimenuje jedan konstrukt sheme. Pomoću AutoMed-ovih transformacija autori su prikazali evoluciju sheme skladišta podataka uzrokovanu promjenama u shemama izvora, promjenama u jeziku za modeliranje, ili objema promjenama.

U [15] autori su definirali WHES (eng. *Warehouse Evolution System*) prototip koji opisuje evoluciju skladišta podataka te omogućava ažuriranje dimenzija i podatkovnih kocki. Također, proširili su SQL jezik u MDL (eng. *Multidimensional Data definition Language*) jezik. Autori su predložili 8 operatora za ažuriranje dimenzija, te odgovarajućih 8 operatora za ažuriranje podatkovnih kocki.

U [16] autori su predložili osam operatora evolucije sheme, koji su uglavnom usmjereni na ažuriranje dimenzija i razina te ažuriranje atributa dimenzija i mjera.

Predloženi operatori su sljedeći: Dodati dimenziju u činjenicu, Brisati dimenziju, Dodati razinu, Brisati razinu, Spojiti atribut s razinom dimenzije, Razdvojiti atribut od razine dimenzije, Dodati mjeru, Brisati mjeru.

U [17] autori su predložili formalizam za predstavljanje sheme skladišta podataka i određivanje ispravnosti operatora evolucije primijenjenih na shemi. Također, autori su predstavili alat koji brzo može kreirati shemu skladišta podataka te provjeriti valjanost operacija za evoluciju sheme, kroz upotrebu okidača i pohranjenih procedura. Predloženi su i operatori za ažuriranje proširenih hijerarhija: Višestruke hijerarhije, Ne-pokrivajuće hijerarhije, Ne-prema hijerarhije, Ne-stroge hijerarhije.

U [18] autori su dali pregled dosadašnjih istraživanja na području evolucije sheme skladišta podataka te su predložili pet operatora za ažuriranje agregirane tablice činjenica. Agregirana tablica činjenica sadrži unaprijed sastavljene izračune na najvišoj razini zrnatosti te sadrži nove metrike (agregirane činjenice ili sažete statistike) dobivene primjenom zbirnih funkcija. Autori su predložili sljedeće operatore: Kreirati agregaciju, Brisati agregaciju, Izmijeniti agregaciju, Preimenovati agregaciju, Ukloniti agregaciju.

U [19] autori su proučavali složene proširene hijerarhije te su predložili operatore evolucije i ograničenja koja treba zadovoljiti, kako bi se osigurao integritet podataka i ispravnost sheme. Za razliku od [17], ovdje autori razlikuju varijacije višestrukih hijerarhija te su predložili dodatna tri operatora za ažuriranje hijerarhija: Višestruka alternativna hijerarhija, Paralelna zavisna hijerarhija i Paralelna nezavisna hijerarhija. Definirana su i ograničenja nad navedenim vrstama višestrukih hijerarhija, koja moraju biti zadovoljena radi postizanja ispravnosti sheme.

U [20] autori proučavaju utjecaj evolucije sheme skladišta podataka na skup od njega zavisnih tržišta podataka (eng. *data mart*). Predložena je arhitektura za evoluciju skladišta podataka, koja uključuje horizontalnu i vertikalnu evoluciju. Vertikalna evolucija, između ostalog, sadrži generični model propagacije (eng. *Generic Propagation Model*, *GPM*), koji opisuje kako se promjene u shemi skladišta podataka transformiraju u promjene sheme tržišta podataka. Također, napravljen je pregled operatora za evoluciju tržišta podataka te je definiran skup pravila za propagaciju.

B. Verzioniranje sheme

Verzioniranje sheme sastoji se od prebacivanja podataka iz postojeće sheme u novu shemu te se sve promjene vrše na novoj shemi. Pritom se čuvaju sve stare verzije shema kroz korištenje vremenske ekstenzije na verziju sheme ili kroz fizičku pohranu verzija.

U [21] autori su predložili proširenje višedimenzionalnog modela – temporalni višedimenzionalni model, koji omogućuje pohranjivanje vremenskih verzija dimenzijskih podataka. Predložena su povezivanja za prijenos podataka između različitih vremenskih verzija, kako bi sustav ispravno mogao odgovoriti na upite nad različitim verzijama dimenzijskih podataka.

U [22] autori su predložili operatore za ažuriranje dimenzija (Kreirati/Brisati dimenziju, Kreirati/Brisati hijerarhiju, Kreirati/Brisati razinu, Pomaknuti razinu u hijerarhiji) i instanci (Kreirati/Brisati člana, Izmijeniti naziv ili svojstvo člana, Spojiti n članova u jednog člana, Podijeliti jednog člana u n članova, Reklasifikacija člana u dimenziji). Za svaku promjenu definira se nova, vremenski ograničena verzija.

U [23] autori su predložili petnaest operatora za izmjenu sheme skladišta podataka: Kreirati novu tablicu razine, Povezati tablicu razine s njenim pod/nad tablicama razine, Razdvojiti tablicu razine iz njene hijerarhije dimenzije, Brisati razinu iz sheme, Dodati atribut u razinu, Brisati atribut iz razine, Izmijeniti domenu atributa razine, Kreirati novu tablicu činjenica, Dodati atribut u tablicu činjenica, Povezati tablicu činjenica s dimenzijom, Brisati atribut koji nije primarni ili vanjski ključ iz tablice činjenica, Brisati vezu (vanjski ključ) između tablice činjenica i dimenzije, Brisati tablicu činjenica, Preimenovati atribut, Brisati tablicu. Autori predlažu primjenu ovih operatora na novoj verziji skladišta podataka.

U [24] autori su proučavali evoluciju hijerarhija unutar dimenzija. Autori su predložili sljedeće operatore za evoluciju sheme: DodatiA (dodavanje novog atributa u dimenziju), BrisatiA (brisanje atributa iz dimenzije), DodatiF (unos nove hijerarhije između dva atributa u dimenziji), BrisatiF (brisanje hijerarhije iz dimenzije). Također, autori su predložili spremanje uvećane sheme zajedno s novom verzijom sheme.

U [25] autori su predstavili prototip više-verzijskog skladišta podataka koje podupire promjene u strukturi sheme te „what-if“ analizu. Također, autori su prikazali korištenje tehnike dijeljenja podataka, kojom se u novoj verziji skladišta podataka pohranjuju samo oni podaci koji su novi ili izmijenjeni u odnosu na prethodnu verziju, te podaci koji su vezani uz roditeljsku verziju (a dijele ih verzije-djeca). Definirana su dva tipa verzija: prava verzija i alternativna verzija. Prava verzija prati promjene u stvarnom svijetu, tj. promjene u vanjskim izvorima podataka. Alternativna verzija prati promjene koje se događaju u „what-if“ analizi. Važno je za napomenuti da su autori postavili i vremensko ograničenje na verziju, pri čemu svaka verzija ima svoje vrijeme trajanja unutar kojega je ona važeća verzija.

U [26] autori su proširili rad iz [25] tako što su predložili dvije grupe operatora: operatore za promjenu sheme (petnaest operatora, uglavnom usmjereni na ažuriranje dimenzija, instanci, atributa, razina i činjenica) i operatore za strukturnu promjenu instanci dimenzija (pet

TABLICA I. **USPOREDBA RADOVA I ISTRAŽIVANJA VEZANIH UZ EVOLUCIJU I VERZIONIRANJE SHEME**

	Promjene atributa	Promjene ograničenja	Promjene kocke	Promjene dimenzije	Promjene činjenice	Promjene hijerarhije	Promjene instance	Promjene razine	Promjene mjere	Promjene agregirane činjenice	Nova verzija sheme	Prava i alternativna verzija	Promjene ETL procesa
Hurtado [8]				●			●	●					
Hurtado [9]							●						
Blaschka [10]	●			●	●			●					
Vaisman [11]			●	●			●	●					
Chen [12]	●			●	●								
Marotta [13]	●			●	●	●			●				
Fan [14]	●			●	●	●	●						
Benitez [15]	●		●	●	●			●	●				
Kaas [16]	●			●				●	●				
Banerjee [17]			●	●	●	●		●					
Meenakshi [18]										●			
Talwar [19]		●				●							
Feki [20]	●	●		●	●	●		●	●				
Eder [21]	●			●		●	●	●			●		
Body [22]	●			●		●	●	●			●		
Morzy [23]	●	●		●	●	●		●			●		
Golfarelli [24]	●			●		●		●			●		
Bebel [25]											●	●	
Bebel [26]	●			●	●		●	●			●		
Papastefanatos [27]	●	●											●
Golfarelli [28]											●		
Solodovnikova [29]											●		●
Zouari Turki [30]		●		●	●		●				●		

operatora, uglavnom usmjereni na ažuriranje razina instanci).

U [27] autori su proučavali problem izvođenja „what-if“ analize za promjene koje se događaju nad shemom izvora podataka te su predstavili model grafa koji uniformno modelira relacije, upite, poglede, ETL aktivnosti i njihova stanja. Ovaj model grafa omogućuje predviđanje učinka promjene na sveukupni sustav. Autori definiraju okvir i opći mehanizam za izvođenje „what-if“ analize nad potencijalnim promjenama izvora podataka.

U [28] autori su predstavili X-Time, istraživački prototip za upravljanje verzioniranjem shema u relacijskim skladištima podataka. X-Time je specifično orijentiran na formulaciju među-verzijskih upita te podupire definiranje i punjenje uvećanih shema i omogućava korisniku da vizualno formulira među-verzijske upite nad grafom sheme.

U [29] predstavljen je okvir za evoluciju skladišta podataka, koji podupire sljedeće promjene: unos, brisanje i preimenovanje izvorne relacije te unos, brisanje i promjena tipa izvorne relacije. Predloženi okvir automatizira evoluciju sheme skladišta podataka i kreiranje novih verzija te omogućuje prilagodbu ETL procesa i postojećih izvještaja nad shemom skladišta podataka. Za razliku od prethodnih radova, koji razmatraju samo jedan vid evolucijskih problema (promjene sheme uzrokovane promjenama poslovnih zahtjeva ili promjene sheme uzrokovane promjenama u izvorima podataka), predloženi okvir promatra oba aspekta evolucije sheme skladišta podataka.

U [30] autori su predložili model MV-DW, za upravljanje više-verzijskim skladištima podataka, zasnovan na verzioniranju sheme i instanci skladišta podataka. U svrhu očuvanja konzistentnosti, MV-DW model podupire nekoliko postojećih strukturalnih i temporalnih ograničenja te definira ograničenje temporalne zavisnosti između dimenzije i činjenice, kako bi se osigurala zbrojivost mjera kroz više dimenzijskih verzija.

C. Održavanje pogleda

Pogled predstavlja relaciju izvedenu nad osnovnom relacijom (pohranjenom u izvoru podataka), odnosno funkciju definiranu između skupa osnovnih tablica i izvedene tablice. Svaki put kad je pogled pozvan, on se izračunava ispočetka. Pogled je materijaliziran kada se njegove n-torke pohrane u bazu podataka. Kod materijaliziranog pogleda nije potrebno ponovno izračunavati funkciju, već je dovoljno samo pristupiti podacima pohranjenima u bazi podataka. Velika prednost materijaliziranog pogleda je to što omogućava brz pristup podacima. Za još brži pristup podacima, nad materijaliziranim pogledom mogu se kreirati indeksi.

Skladište podataka može biti definirano kao skup materijaliziranih pogleda nad izvorima podataka [34], [38], [40]. Problemi koji se ovdje javljaju jesu kako efikasno održavati poglede (kako prilagoditi i sinkronizirati poglede nakon promjena u definiciji ili opsegu pogleda) [31], [40] te kako efikasno odabrati

poglede za veći učinak (koje agregacije na kojoj razini povećavaju učinak s obzirom na prostorna ograničenja) [32].

Problem održavanja materijaliziranog pogleda razmatra se u odnosu na održavanje kvalitete podataka, promjene u podacima te promjene u izvorima podataka. Održavanje pogleda možemo podijeliti na dva pristupa: prilagodba pogleda i sinkronizacija pogleda. Kod prilagodbe pogleda, da bi se materijalizirani pogled prilagodio promjenama, dodaju mu se meta-podaci koji sadrže najnovije strukturalne informacije. Za razliku od prilagodbe pogleda, sinkronizacija pogleda predstavlja ponovno pisanje pogleda.

U [31] autori su opisali materijalizirane poglede, vrste primjene te probleme i tehnike za njihovo održavanje. Autori su predstavili klasifikaciju problema održavanja pogleda, prema kojoj se problem održavanja pogleda može proučavati kroz četiri dimenzije – dimenzija informacije, dimenzija promjene, dimenzija jezika i dimenzija instance. Također, opisali su nekoliko algoritama za inkrementalno održavanje pogleda.

U [33] autori su predstavili formalni model za evoluciju sheme skladišta podataka, koji ima samo dva operatora – Dodati pogled i Brisati pogled.

U [34] autor je predstavio dinamičku prilagodbu pogleda vezanu uz promjene u izvorima podataka. Ovim pristupom dobiva se na jednostavnosti – novi pogled se kreira iz starog pogleda, a ne ponovno iz različitih izvora podataka. Autor je ovaj problem proučavao iz dvije perspektive – korisnika i izvora podataka. Do korisničkih promjena nad shemom materijaliziranog pogleda dolazi kad se postave novi zahtjevi. S druge strane, izvori podataka mogu mijenjati svoju shemu, što utječe na strukturalnu konzistentnost skladišta podataka.

U [35] autori su definirali C-SQL (eng. *Cooperative SQL*), proširenje SQL jezika za pisanje pogleda.

U [36] autori su definirali S-SQL (eng. *Schema SQL*) za integraciju relacijskih baza podataka i meta-podataka.

U [37] autori su proučavali problem praćenja podrijetla podataka (eng. *lineage tracing*). Formalizirali su problem i razvili algoritam za praćenje podrijetla podataka za relacijske ASPJ (eng. *aggregate-select-project-join*) poglede. Također, predložili su nekoliko shema za pohranjivanje pomoćnih pogleda te su prezentirali studiju izvodivosti predloženih shema u određenim okruženjima.

U [38] autori su predstavili prototip EVE, sustav za automatiziranje ponovnog pisanja definicija pogleda. Cilj EVE sustava je rješavanje problema nefleksibilnosti pogleda, odnosno očuvanje najvećeg broja definicija pogleda po promjenama sheme u izvorima podataka. EVE sustav pomoću omotača integrira izvore podataka te pretvara modele izvora podataka u zajednički relacijski model. Autori i tvorci EVE sustava predstavili su i E-SQL jezik, prošireni SQL jezik za predstavljanje evolucije definicije pogleda te MISD, model za opis izvora podataka.

U [39] autori su predstavili formalni okvir SDCC (eng. *Schema change and Data update Concurrency Control*), kao pristup za rješavanje problema istovremenog ažuriranja sheme i podataka.

U [41] fokus je na kvaliteti podataka za vrijeme evolucije sheme. Autor je predstavio više operatora na

TABLICA II. **USPOREDBA RADOVA I ISTRAŽIVANJA VEZANIH UZ ODRŽAVANJE POGLEDA**

	Osnovno održavanje	Inkrementalno održavanje	Samoodrživo održavanje	Odabir pogleda	Select-Project_Join pogled	Meta-podaci	Implementacija i jezik	Kvaliteta
Gupta [31]	●	●			●		●	
Baralis [32]				●				
Bouzeghoub [33]	●							
Bellahsene [34]		●						
Rajaraman [35]							●	
Lakshmanan [36]						●	●	
Cui [37]		●			●			
Rundensteiner [38]							●	
Zhang [39]		●						
Rundensteiner [40]		●						
Quix [41]	●							●
Akaichi [42]							●	
Almazyad [43]		●				●		

osnovne relacije i poglede te njihov utjecaj na kvalitetu podataka. Autor je za svaki operator definirao njegov utjecaj na kvalitetu podataka, pa tako operatori mogu utjecati na točnost, cjelovitost i konzistentnost logičke sheme (Dodati osnovnu relaciju/pogled, Obrisati osnovnu relaciju/pogled, Dodati atribut u osnovnu relaciju/pogled, Obrisati atribut iz osnovne relacije/pogleda, Izmjeniti definiciju pogleda), razumljivost pogleda i njegovih atributa (Preimenovati relaciju, pogled ili atribut), razumljivost i mogućnost interpretacije podataka (Izmjeniti domenu atributa) te vjerodostojnost i konzistentnost podataka (Dodati ograničenje integriteta, Obrisati ograničenje integriteta).

U [42] autori su na temelju sustava EVE definirali MAVIE, sustav za sinkronizaciju pogleda u dinamičnijem i distribuiranom okruženju, baziran na tehnologijama mobilnih agenata. MAVIE sustav smanjuje vrijeme sinkronizacije zbog paralelnosti kod mobilnih agenata te izbjegava zasićenost mreže.

U [43] autori su predložili algoritam za inkrementalno održavanje pogleda, pri čemu su koristili koncept pohranjivanja verzija za starije verzije tablica ažuriranih na izvoru podataka te im pridruživali broj transakcije. Time su pokušali riješiti problem anomalija i nekonzistentnih promjena nad pogledima.

U [44] nalazi se dobar pregled istraživanja problema održavanja pogleda. Autori su radove koji se bave istraživanjem problema održavanja pogleda grupirali po tri pristupa – osnovno održavanje pogleda, inkrementalno održavanje pogleda i samoodrživo održavanje pogleda.

III. SMJERNICE ZA BUDUĆA ISTRAŽIVANJA

Problem evolucije sheme skladišta podataka može se promatrati i kao dvostruki problem, na razini skladišta podataka i na razini upravljanja osnovnim podacima (eng. *master data management, MDM*) ([45], [46], [47]). Iz prespektive skladišta podataka, prati se svaki događaj koji je povezan s dimenzijama. S druge strane, kod MDM se prate osnovni podaci (dimenzije), a kontekst im čine događaji. I u slučaju MDM-a, kao i kod skladišta podataka, javlja se problem evolucije sheme nakon promjena u izvorima podataka ili korisničkim zahtjevima.

Međutim, proces evolucije ili verzioniranja sheme još uvijek zahtjeva puno resursa – vremena te mrežnih i ljudskih resursa. Potrebno je uravnotežiti zahtjeve za resursima i kvalitetu provođenja evolucije skladišta podataka. Možda najveći problem kod verzioniranja sheme jest očuvanje konzistentnosti sheme i integriteta podataka, kao i izvođenje povijesnih upita nad više verzija sheme istovremeno. U pregledu dosadašnjih istraživanja prezentirani su različiti pristupi rješavanju problema evolucije skladišta podataka, uključujući razne tehnike, algoritme, algebre, modele, prototipe, metodologije i okvire, međutim i dalje ne postoji šire prihvaćeno rješenje. Neki autori definiraju metodologije za evoluciju skladišta podataka, koje su vezane uz konkretnu implementaciju, međutim one su ograničene za upotrebu.

Dalje, javlja se problem temporalnih upita. Kako bi se postigla savršena arhivska kvaliteta sustava i podataka, ključno je pohraniti podatke u onu verziju sheme u kojoj su se oni prvi put pojavili. Mnogi autori smatraju da kod verzioniranja sheme nije potrebno razlikovati vrijeme transakcije i vrijeme važenja. Međutim, treba uzeti u obzir i mogućnost istovremeno važećih verzija, gdje je potrebno razlikovati ova dva vremena, kako bi se upiti nad više verzija mogli izvršiti nezavisno od vremena kad su oni postavljeni. Dosadašnji temporalni upitni jezici, poput TSQL2 [48], omogućili su rad s temporalnim elementima i ograničenjima, međutim ne podupiru verzioniranje u potpunosti, odnosno ne podupiru među-verzijske upite uopće, ili ih podupiru djelomično. Iako je akademska zajednica napravila korake u pravcu postavljanja i upravljanja među-verzijskim upitima, tu još ima prostora za poboljšanje (prvenstveno vezano uz softversku podršku, temporalne upitne jezike i performanse upita), kao i za pronalazak opće prihvaćenog rješenja, te u potpunosti efikasnog komercijalnog rješenja.

Najvažnije, dosadašnja istraživanja ne naglašavaju činjenicu da su zahtjevi skladišta podataka, u današnje doba, povećani u pogledu opsega i strukture podataka i meta-podataka (sve veći broj izvora podataka te više novih i različitih tipova podataka, poput npr. polustrukturiranog teksta), što dodatno zahtijeva osmišljanje novih pristupa i rješenja problema evolucije skladišta podataka.

Kao ideju za rješavanje ovoga problema, razmatram izradu meta-Data Vault modela ([49],[50]) koji će integrirati skladište podataka s MDM sustavom te na taj način omogućiti rješavanje problema evolucije sheme, promatranog kroz pristup verzioniranja sheme.

Data Vault je metoda za modeliranje baza podataka koja omogućuje dugoročnu pohranu povijesnih podataka, prikupljenih iz raznovrsnih izvora podataka. Data Vault metoda zasniva se na pretpostavci da je okruženje skladišta podataka u konstantnoj promjeni te naglašava potrebu za praćenjem od kuda su došli svi podaci koji se nalaze u bazi podataka, kroz empirijski definiran skup meta-podataka. To znači da se svakom retku u Data Vault bazi podataka dodaju atributi izvor_zapisa i datum_punjenja. Na taj način se omogućuje praćenje vrijednosti natrag do izvora. Dalje, prema Data Vault metodi ne postoji razlika između dobrih i loših podataka – pohranjuju se svi podaci u svako vrijeme, bez obzira jesu li prilagodljivi poslovnim pravilima. Također, model se zasniva na šestoj normalnoj formi, što kao rezultat daje visoko dekomponiranu implementaciju. Eksplicitno se razdvajaju strukturalni podaci od opisnih atributa, bez obzira dolaze li oni iz istog izvora. Ovime se dobiva prilagodljivost modela promjenama u poslovnom okruženju. Svaka promjena je implementirana u model kao nezavisno proširenje postojećeg modela, što znači da promjene ne utječu na trenutne aplikacije koje se zasnivaju na prethodnim modelima. Ovo također znači da se sve verzije aplikacije mogu zasnivati na istoj,

razvijajućoj bazi podataka. Sve verzije modela postoje kao podskup Data Vault modela. I na kraju, Data Vault metoda omogućuje paralelno punjenje, što utječe na smanjenje troškova vremena i resursa.

Praćenje podrijetla podataka, povijesni upiti i rad s više verzija sheme istovremeno više ne bi predstavljali problem, radi upotrebe Data Vault modela. Također, zbog dugoročne pohrane povijesnih podataka i praćenja podrijetla podataka, olakšalo bi se čuvanje integriteta skladišta podataka, pri čemu bi ono stvarno sadržavalo „jednu verziju istine“ [51] te bi se moglo koristiti kao sustav evidencije baze podataka (eng. *system of records*) [52]. Međutim, ova tema nije u obujmu ovoga rada – o njoj će više riječi biti u daljnjim radovima.

IV. ZAKLJUČAK

Skladišta podataka iz raznih izvora prikupljaju brojne interne i proizvodne podatke, arhivirane podatke te vanjske informacije. Ti izvori podataka nezavisni su od skladišta podataka (iako su u njega integrirani) te mogu u svakom trenutku promijeniti svoju shemu, neovisno o samome skladištu podataka. Također, korisnici često mogu mijenjati svoje zahtjeve, vezane uz analizu i izvještavanje. Skladište podataka mora moći prihvatiti i prilagoditi se tim promjenama.

U ovome radu predstavljen je pregled različitih istraživanja u području evolucije skladišta podataka. Istraživanja su grupirana prema tri pristupa – evolucija sheme, verzioniranje sheme i održavanje pogleda. Također, predstavljene radove često možemo promatrati i kao istraživanja koja se fokusiraju na upravljanje promjenama u skladištu podataka, istraživanja koja se fokusiraju na upravljanje promjenama podataka u tržištu podataka (promjene u činjenicama i dimenzijama) te istraživanja koja proučavaju upravljanje promjenama sheme u tržištu podataka (promjene nastale zbog rastućih poslovnih zahtjeva). Analiza literature pokazuje da se autori uglavnom usmjeravaju na jedan aspekt evolucije skladišta podataka, i to uglavnom na upravljanje promjenama podataka u tržištu podataka, gdje se može reći da postoji opći konsenzus o mogućim rješenjima za upravljanje promjenama kod dimenzionalnih podataka. Što se tiče ostalih aspekata evolucije skladišta podataka, prvenstveno upravljanja promjenama u shemi skladišta i tržišta podataka, predloženi su neki zanimljivi pristupi, koje vrijedi dodatno proučavati. Međutim, ne postoji šire prihvaćeno rješenje te opći okvir za upravljanje promjenama u shemi.

Ono što se iz navedenog pregleda može zaključiti je to da se problem evolucije skladišta podataka još nije uspio riješiti na odgovarajući način.

Proces evolucije i verzioniranja sheme je zahtjevan u vidu utrošenog vremena i resursa; potrebno je automatizirati sustave za praćenje valjanosti operacija evolucije, kao i ispravnosti nove sheme, kako bi se sačuvala konzistentnost sheme i integritet podataka. Također, javlja se problem transformacije i migracije podataka između različitih verzija, problem organizacije i upravljanja meta-podacima, problem odabira i fizičke pohrane verzija, problem usklađivanja i prilagodbe postojećih upita nad novom verzijom sheme, problem korisničkih aplikacija koje također treba prilagoditi novoj shemi te problem rada s više verzija sheme istovremeno.

S druge strane, i proces održavanja i prilagodbe pogleda ima svoje, koliko mi je poznato, još neriješene probleme. Sam proces održavanja pogleda može uzrokovati zasićenost mreže, ovisno o količini pogleda koji se ažuriraju te količini sadržanih podataka. Još uvijek je učestala praksa ručno pisanje pogleda nakon promjena u izvorima. Također, predloženi pristupi za održavanje i prilagodbu pogleda i dalje su limitirani u pogledu efikasnosti i performansi, pogotovo u slučaju održavanja temporalnih pogleda nad promjenjivim, ne-temporalnim izvorima podataka, pa je i tu moguće poboljšanje.

Ovo su samo neki od problema vezanih uz navedene pristupe, međutim, jasno je da tu još ima mnogo prostora za istraživanje.

Vezano uz to, u radu su predstavljene opće smjernice za daljnje istraživanje. Prvenstveno je to ideja za istraživanje problema evolucije skladišta podataka i sustava za upravljanje osnovnim podacima (MDM), kroz upotrebu metoda i principa Data Vault modeliranja.

LITERATURA

[1] Roddick, J. F. *A Survey of Schema Versioning Issues for Database Systems*. Information and Software Technology, 37(7):383–393, 1996.

[2] Andany, J., Leonard, M., Palisser, C. *Management Of Schema Evolution In Databases.* In Proceedings of the 17th International Conference on Very Large Databases, Barcelona, 1991.

[3] Das, A.K.., Tu, S.W., Musen, M.A. *A Historical Relational Data Model for Managing Temporal Data*. Report, Stanford University, Section on Medical Informatics, San Diego, 1992.

[4] Curino, C.A., Moon, H.J., Zaniolo, C. *Graceful Database Schema Evolution:the PRISM Workbench*. In Proceedings of the Very Large Database Endowment, Auckland, New Zeland, 2008.

[5] Date, C.J., Darwen, H., Lorentzos, N. *Temporal Data & the Relational Model*. Morgan Kaufmann Publishers, Burlington, USA, 2002.

[6] Johnston, T., Weis, R. *Managing Time in Relational Databases: How to Design, Update and Query Temporal Data.* Morgan Kaufmann Publishers, Burlington, USA, 2010.

[7] Oueslati, W., Akaichi, J. *A survey on Data warehouse evolution.* International Journal of Database Management Systems (IJDMS), vol.2, no.4, 2010.

[8] Hurtado, C. A., Mendelzon, A. O., Vaisman, A. A. *Maintaining Data Cubes under Dimension Updates*. In Proceedings of the 15th International Conference on Data Engineering (ICDE), Sydney, Australia, pages 346–355. IEEE Computer Society, 1999.

[9] Hurtado, C. A., Mendelzon, A. O., Vaisman, A. A. *Updating OLAP Dimensions.* In Proceedings of the 2nd International Workshop on Data Warehousing and OLAP, Kansas City, Missouri, USA, 1999.

[10] Blaschka, M., Sapia, C., Hofling, G. *On Schema Evolution in Multi-dimensional Databases*. In 1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK 99), Florence, Italy, vol. 1676 of LNCS, pages 153–164. Springer, 1999.

[11] Vaisman, A.A., Mendelzon, A.O., Ruaro, W., Cymerman, S.G. *Supporting Dimension Updates in an OLAP Server*. In Proceedings of the CAISE02 Conference,Canada, 2002.

[12] Chen, J., Chen, S., Rundensteiner, E. *A transactional model for data warehouse maintenance*. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 247–262. Springer, Heidelberg, 2002.

[13] Marotta, A., Ruggia, R. *Data Warehouse Design: A schema-transformation approach*. 22nd International Conference of the Chilean Computer Science Society (SCCC), Copiapo, Chile, 2002.

[14] Fan, H., Poulovassilis, A. *Schema Evolution in Data Warehousing Environments – a schema transformation-based approach*. In Proceedings of Conceptual Modeling - ER, 23rd International Conference on Conceptual Modeling, Shanghai, China, 2004.

[15] Benitez-Guerrero, E., Collet, C., Adiba, M. *The WHES Approach to Data Warehouse Evolution* . e-Gn osis(online), vol.2 , 2004.

[16] Kaas, C.E., Pedersen, T.B., Rasmussen, B.D. *Schema Evolution for Stars and Snowflakes*. In Proceedings of the International Conference on Enterprise Information Systems(ICEIS 2004), Portugal, 2004.

[17] Banerjee, S., Davis, K.C. *Modeling Data Warehouse Schema Evolution over Extended Hierarchy Semantics*, S.Spaccapietra et.al (EDs): Journal on Data Semantics XIII, LNCS 5530, pp.72-96,Springer- Berlin, Heidelberg, 2009.

[18] Meenakshi, A., Gosain, A. *Schema Evolution for Data Warehouse: A Survey*. International Journal of Computer Applications, vol.22, no.6, 2011.

[19] Talwar, K., Gosain, A. *Implementing Schema Evolution in Data Warehouse through Complex Hierarchy Semantics*. International Journal of Scientific & Engineering Research, vol. 3, iss. 7, 2012.

[20] Feki, J., Taktak, S. *Impacts of Data Warehouse Evolution on Data Marts*. In Proceedings of the 13th International Arab Conference on Information Technology (ACIT), University of Balamand, Lebanon, 2012.

[21] Eder, J., Koncilla, C. *Evolution of Dimension Data in Temporal Data Warehouses*. Technical Report, 2000.

[22] Body, M. , Miquel, M., Bedard, Y., Tchounikine, A. *A Multidimensional and Multiversion Structure for OLAP Applications*. In 5th ACM International Workshop on Data Warehousing and OLAP (DOLAP 02),McLean, Virginia, USA, pages 1–6. ACM Press, 2002.

[23] Morzy, T., Wrembel, R. *On Querying Versions of Multiversion Data Warehouse*. In Proceedings of the International Workshop on Data Warehousing and OLAP, DOLAP˝04, Washington, USA, 2004.

[24] Golfarelli, M., Lechtenbörger, J., Rizzi, S., Vossen, G.: *Schema Versioning in Data Warehouses*. In: Wang, S., Tanaka, K., Zhou, S., Ling, T.-W., Guan, J., Yang, D.Grandi, F., Mangina, E.E., Song, I.-Y., Mayr, H.C. (eds.) ER Workshops 2004. LNCS, vol. 3289, pages 415–428. Springer, Heidelberg 2004.

[25] Bebel, B., Eder, J., Koncilia, C. , Morzy, T., Wrembel, R. *Creation and Management of Versions in Multiversion Data Warehouse*. In 19th ACM Symposium on Applied Computing (SAC 04), Nicosia, Cyprus, pages 717–723. ACM Press, 2004.

[26] Bebel, B., Krolinkowski, Z., Wrembel, R. *Formal approach to modeling a multiversion data warehouse*. Bulletin of the Polish academy of sciences, Technical Sciences, vol. 54, no. 1, 2006.

[27] Papastefanatos, G., Vassiliadis, P., Simitsis, A., Vassiliou, Y. *What-if Analysis for Data Warehouse Evolution*. In Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery, Regensburg, Germany, 2007.

[28] Golfarelli, M., Rizzi, S. *X-Time: Schema Versioning and Cross-Version Querying in Data Warehouses*. In Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey, pages 1471-1472, 2007.

[29] Solodovnikova, D. *Data Warehouse Evolution Framework*. In Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, Moscow, Russia, 2007.

[30] Zouari Turki, I., Ghozzi Jedidi, F., Bouaziz, R. *Constraints to manage consistency in multiversion data warehouse*. In Proceedings of the 35th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2012.

[31] Gupta, A., Mumick, I. *Maintenance of Materialized Views: Problems, Techniques, and Applications*. Data Engineering Bulletin, 1995.

[32] Baralis, E., Paraboschi, S., Teniente, E. *Materialized view selection in a multidimensional database*. Proceedings Conference on Very Large Databases, Athens, Greece, 1997.

[33] Bouzeghoub, M., Kedad, Z. *A Logical Model for Data Warehouse Design and Evolution*. Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK), London, UK, pages 178-188, 2000.

[34] Bellahsene, Z. *Schema Evolution in Data Warehouses*. Knowledge and Information Systems, 4(3):283–304, 2002.

[35] Rajaraman, Y. Sagiv, J. D. Ullman. *Answering Queries Using Templates With Binding Patterns*. Proc. ACM Symp. Principles Database System, pages 105-112, 1995.

[36] Lakshmanan, L. V. S., Sadri, F. , Subramanian, I. N. *Schema SQL: a Language for Interoperability in Relational Multi-Databases Systems*. In Proceedings of the 22nd International Conference on Very Large Databases, pages 239-250, 1996.

[37] Cui, Y., Widom, J. *Practical Lineage Tracing in Data Warehouses*. In Proceedings of the 16th International Conference on Data Engineering (ICDE'00), San Diego, California, 2000.

[38] Rundensteiner, E. A. , Koeller, A., Zhang, X., Lee, A.J., Nica, A. *Evolvable View Environment EVE: A Data Warehouse System Handling Schema and Data Changes of Distributed Sources*. Proceedings of the International Database Engineering and Application Symposium (IDEAS'99), Montreal, Canada, 1999.

[39] Zhang, X., Rundensteiner, E.A. *Data Warehouse Maintenance Under Concurrent Schema and Data Updates*. In Proceedings of the International Conference on Data Engineering (ICDE'99), Sydney, 1999.

[40] Rundensteiner, E. A., Koeller, A., Zhang, X. *Maintaining Data Warehouses Over Changing Information Sources*. In Communications of the ACM, vol.43, pages 57-62, New York, USA, 2000.

[41] Quix. *Repository Support for Data Warehouse Evolution*. In Proceedings of the International Workshop DMDW, Heidelberg, Germany 2004.

[42] Akaichi, J., Oueslati, W. *MAVIE: A Mobile Agents View synchronization system*. In first international conference on the applications of digital information and web technology. Ostravem, pages 145-150, 2008.

[43] Almazyad, A.S., Siddiqui, M.K. *Incremental View Maintenance: An Algorithmic Approach*. International Journal of Electrical and Computer Sciences (IJECS-IJENS), vol.10, no. 3, 2010.

[44] Jain, H., Gosain, A. *A Comprehensive Study of View Maintenance Approaches in Data Warehousing Evolution*. ACM SIGSOFT Software Engineering Notes, vol.37, iss. 5, New York, USA, 2012.

[45] Loshin, D. *Master Data Management*. Morgan Kaufmann Publishers, Burlington, USA, 2008.

[46] Oracle. *Master Data Management*. An Oracle White paper, 2011.

[47] Ladley, J. *Data Governance: How to Design, Deploy and Sustain an Effective Data Governance Program*. Morgan Kaufmann Publishers, Burlington, USA, 2012.

[48] Snodgrass, R. T., *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.

[49] Linstedt, D. *SuperCharge Your Data Watehouse: Invaluable Data Modeling Rules to Implement Your Data Vault*. CreateSpace Independent Publishing Platform, USA, 2011.

[50] Jovanović, V., Bojičić, I. *Conceptual Data Vault Model*. In Proceedings of the Southern Association for Information Systems Conference, Atlanta, USA, 2012.

[51] Inmon, W.H., Strauss, D., Neushloss, G. *DW 2.0: The Architecture for the Next Generation of Data Warehousing*. Morgan Kaufmann Publishers, Burlington, USA, 2008.

[52] Jovanović, V., Bojičić, I., Knowles, C., Pavlic, M. *Persistent Staging Area Models For Data Warehouses*. Issues in Information Systems, vol.13, iss. 1, pages 121-132, 2012.