SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Ognjen Orel

NADZOR NAD RADOM KORISNIKA U RELACIJSKIM BAZAMA PODATAKA

MAGISTARSKI RAD

Zagreb, 2008.

Magistarski rad je izrađen na **Zavodu za primijenjeno računarstvo Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu**

Mentor: **Prof. dr. sc. Mirta Baranović**

Magistarski rad ima 90 stranica

Magistarski rad br.:

Zahvaljujem mentorici prof. dr. sc. Mirti Baranović i doc. dr. sc. Slavenu Zakošeku na pomoći i savjetima pri izradi ovog magistarskog rada.

Povjerenstvo za ocjenu magistarskog rada:

1. Prof. dr. sc. Mladen Varga – Ekonomski fakultet Sveučilišta u Zagrebu – predsjednik
2. Prof. dr. sc. Mirta Baranović – mentor
3. Doc. dr. sc. Slaven Zakošek

Povjerenstvo za obranu magistarskog rada:

1. Prof. dr. sc. Mladen Varga – Ekonomski fakultet Sveučilišta u Zagrebu – predsjednik
2. Prof. dr. sc. Mirta Baranović – mentor
3. Doc. dr. sc. Slaven Zakošek

Datum obrane: 26. studenog 2008.

Sadržaj

1. Uvod ... 4
2. Društveni aspekti snimanja traga ... 7
   2.1. Povijest snimanja traga ... 7
   2.2. Zakonske norme koje utječu na snimanje traga u bazama podataka ... 8
      2.2.1. Sjedinjene Američke Države ... 9
      2.2.2. Europa ... 10
      2.2.3. Hrvatska ... 11
3. Uvodna razmatranja o snimanju traga u bazama podataka ... 12
   3.1. Osnovni pojmovi vezani uz teoriju baza podataka ... 12
   3.2. Snimanje traga ... 14
   3.3. Raspodjela dužnosti ... 14
   3.4. Zaštita snimljenog traga ... 15
4. Metode snimanja traga u radu sustava za upravljanje bazama podataka ... 17
   4.1. Snimanje traga unutar klijentske aplikacije ... 17
   4.2. Snimanje traga unutar baze podataka ... 18
   4.3. Snimanje traga unutar sustava za upravljanje bazama podataka ... 19
      4.3.1. IBM Informix ... 19
      4.3.2. Oracle Database ... 21
      4.3.3. IBM DB2 ... 22
      4.3.4. Microsoft SQL Server ... 23
   4.4. Snimanje traga izvan sustava za upravljanje bazama podataka ... 24
   4.5. Komercijalni sustavi za snimanje traga ... 27
5. Kategorije snimanja traga u bazama podataka ... 30
   5.1. Prijava i odjava za rad s bazom podataka ... 30
   5.2. Rad s bazom podataka izvan uobičajenih obrazaca ... 30
   5.3. Podaci o klijentskim aplikacijama ... 31
   5.4. DDL naredbe ... 32
   5.5. Pogreške koje sustav za upravljanje bazama podataka prijavljuje korisnicima ... 33
   5.6. Izmjene programskog kôda pohranjenih procedura i okidača ... 33
   5.7. Izmjene podataka o korisnicima i njihovim dozvolama ... 34
   5.8. Korisnički sinonimi i replikacija baza podataka ... 34
   5.9. SQL naredbe za izmjenu podataka ... 35
   5.10. SELECT naredbe ... 35
   5.11. Snimanje izmjena definicija snimanja traga ... 36
6. Snimanje traga o izmjenama podataka prilagodbom okidača ... 37
   6.1. Motivacija ... 37
   6.2. Opis izvedbe ... 38
      6.2.1. Proširenje rječnika baze podataka ... 38
      6.2.2. Stvaranje povijesnih relacija ... 41
      6.2.3. Nadopuna okidača naredbama za snimanje traga ... 42
      6.2.4. Stvaranje povijesne baze podataka ... 45
      6.2.5. Proces backupHistory ... 47

6.2.6. Ubrzavanje čitanja snimljenog traga ... 47
6.2.7. Dozvole korištenja ... 49
6.3. Implementacija ... 50
6.4. Mogućnosti proširenja ... 55
6.4.1. Snimanje traga o pristupanju podacima ... 56
6.4.2. Definiranje operacija za koje se snima trag ... 56
6.4.3. Definiranje korisnika za koje se snima trag ... 56
6.4.4. Postavljanje uvjeta snimanja traga ... 57
6.4.5. Uklanjanje zastarjele povijesti ... 58
6.4.6. Arhiviranje zastarjele povijesti ... 59
7. Analiziranje snimljenog traga o izmjenama podataka ... 60
7.1. Pregled cjelokupne povijesti relacije ... 60
7.2. Pregled djelatnosti korisnika u vremenskom periodu ... 62
7.3. Kontrola stvarnih i deklariranih dozvola korisnika ... 64
7.3.1. Prikupljanje podataka o tranzitivnim dozvolama ... 66
7.3.2. Usporedba obavljenih operacija s deklariranim dozvolama ... 74
7.3.3. Mogućnosti proširenja ... 76
8. Prikaz primjene snimanja traga o izmjenama podataka ... 78
9. Zaključak ... 81
Literatura ... 83
Sažetak ... 88
Summary ... 90
Ključne riječi ... 92
Keywords ... 92
Životopis ... 93

1. Uvod

Podaci mogu biti vrlo vrijedna imovina, i kao takvi, predstavljaju određeni izvor rizika. To je prepoznato u brojnim tvrtkama i organizacijama te su uspostavljeni brojni mehanizmi kako bi se zaštitili njihovi podaci. U prošlosti su bili usmjereni na zabranu pristupa podacima organizacije kroz neki sigurnosni štit, kao što je vatrozid (eng. *firewall*) ili otkrivanje upada. No, tijekom vremena je otkriveno kako velike opasnosti dolaze i unutar organizacije. Bilo da je zloporaba podataka uzrokovana zlonamjernim ponašanjem ili ljudskom pogreškom, moguće su znatne štete [Mazer2005].

Kako u slučaju vanjskih napada na informacijski sustav, tako i slučaju napada koji dolaze iznutra, sigurnost sustava je od iznimne važnosti. Pod time se podrazumijeva da samo ovlašteni korisnici mogu pristupati onim podacima za koje imaju dozvole, te da s njima mogu raditi samo ono što im je dopušteno. Uz to, taj se posao mora obavljati na propisani način. Navedena ograničenja može se promatrati kao sigurnosne parametre sustava. Ukoliko se cjelokupno poslovanje sustava odvija prema ovim pravilima, onda se može reći da sustav funkcionira unutar sigurnosnih parametara.

Pitanje koje se svaki voditelj održavanja informacijskog sustava treba pitati jest, kako možemo biti sigurni da naš sustav funkcionira unutar sigurnosnih parametara? Ili, ako je to točno u ovom trenutku, kako znamo da se nešto nedopušteno nije dogodilo u nekom prijašnjem trenutku? Ili da se neće dogoditi u budućnosti? Postavljanje sigurnosnih parametara samo po sebi nije dovoljno ako se ne može provjeriti jesu li ispravni ili dovoljni.

Drugo pitanje koje se nameće jest, kako možemo znati što je potencijalni napadač sve napravio ukoliko je prošao sigurnosne provjere? Koji su podaci izmijenjeni? Kakve su bile njihove vrijednosti prije napada? Je li napadač promijenio dopuštenja korisnicima? Potrebno je imati cjelokupnu povijest svih važnih podataka, te adekvatne analize nad njima i aktualnim podacima koje mogu dati odgovore na ova pitanja.

Dalje, je li napadač promijenio naše sigurnosne postavke i na taj način omogućio daljnje napade na sustav? Sigurnosne postavke sustava su očito još jedna od kategorija čiju je povijest potrebno čuvati.

Na kraju, što je s našim privilegiranim korisnicima, administratorima sustava – oni mogu pristupiti svim podacima, izbrisati dio povijesti podataka ili je promijeniti tako da prikriju svoje djelovanje? Cjelokupni proces evidentiranja povijesti podataka treba odvojiti od administratora sustava. Ti korisnici ne trebaju imati utjecaja na samo evidentiranje povijesti niti na već evidentiranu povijest.

Kroz ova pitanja i odgovore može se zaključiti da je na razini informacijskog sustava potrebno ustanoviti kontrolni mehanizam koji će omogućiti evidenciju povijesti podataka sustava, nadzor korisničkih akcija i sigurnosnih postavki, te naknadnu analizu tih podataka. Posebnu pozornost treba posvetiti korisničkoj djelatnosti (bez obzira na to radi li se o stvarnom korisniku ili o automatiziranom procesu), obzirom da je to način na koji dolazi do izmjena u sustavu. Sve ovo treba biti uspostavljeno na

način koji neće privilegirati nijednog korisnika sustava, što podrazumijeva i administratore sustava. Ovaj mehanizam naziva se praćenje ili snimanje traga (eng. *auditing*) u informacijskim sustavima. Snimanje traga, dakle, ima dvije sastavnice: prikupljanje i organiziranje snimljenih podataka [Jajodia1995], te analiziranje snimljenih podataka. Iako je osnovna namjena ovih analiza otkrivanje narušavanja sigurnosnih postavki [Lunt1993, Lin1994], njih je moguće koristiti i za poboljšanje performansi aplikativnih rješenja, forenzične i statističke izvještaje o korištenju sustava. Uz snimanje traga uvijek se veže i uloga osobe zadužene za analizu snimljenog traga (eng. *auditor*), u principu neovisna o informacijskom sustavu kojeg analizira.[1]

Osim sigurnosnih implikacija, snimanje traga u bazama podataka dobiva sve više na značaju i zbog povećane brige o privatnosti podataka. Tijekom proteklih godina raste svjesnost o potrebi zaštite osobnih podataka zbog njihove povećane dostupnosti, ali i sve više mogućnosti njihove zloporabe. Širom svijeta države donose nove zakonske norme vezane uz zaštitu osobnih podataka koje, implicitno, sadržavaju i odredbe o snimanju traga, a sve više su osviješteni i „obični“ građani. U [Alawi2006] su autori istraživanjem potvrdili teze:

- napretkom novih tehnologija raste briga osoba za privatnost njihovih podataka,
- održavanje privatnosti podataka je jedan od ciljeva osoba zaduženih za analizu snimljenog traga,
- učinkovita kontrola u bilo kojoj organizaciji znači bolju zaštitu osobnih podataka.

U poslovnom svjetlu, podaci tvrtki koji se ne nadgledaju u kontinuitetu dovode poslovanje tvrtke u opasnost. Nestvarni ili podmetnuti podaci uzrokuju kompromitirane nadzorne podatke, uništeni ugled te, konačno, gubitak korisnika. U tom kontekstu Mercuri ističe važnost provjeravanja snimljenog traga, te njegove robusnosti i dostupnosti [Mercuri2003]. Dani su primjeri u kojima je očito bilo propusta u provjerama različitih vrsta: računovodstvene pronevjere u Enronu i WorldComu, uz upitne financijske izvještaje i drugih tvrtki, među kojima su i poznati Xerox i AOL, ali i falsificiranje podataka o istraživanjima supravodljivosti jednog fizičara tijekom tri godine, te lažne novinske članke jednog slobodnog novinara.

Automatizirano snimanje traga u bazama podataka može pomoći organizacijama upravljati ovim rizicima, tako što će biti lakše identificirati sumnjive pristupe i korištenja podataka. U sklopu toga, Weber identificira sedam glavnih razloga za uspostavljanje snimanja traga u informacijskim sustavima i funkciju kontrole u organizacijama [Weber1998]:

- cijena izgubljenih podataka,
- cijena pogrešne odluke,

[1] Pojam snimanja traga u engleskom jeziku (eng. *auditing*) ima puno širu konotaciju u odnosu na isti pojam u hrvatskom jeziku. Riječ je o nadzoru, prvenstveno poslovanja tvrtki i organizacija, pri čemu ulogu nadzornika (eng. *auditor*) najčešće obavljaju druge, u pravilu neovisne tvrtke. Nadzor informacijskih sustava može biti samo dio cjelokupnog nadzora tvrtke.

- cijena računalne zlouporabe,
- vrijednost programske i sklopovske podrške i osoblja,
- velika cijena računalnih pogrešaka,
- održavanje privatnosti podataka,
- kontrola napretka uporabe računala.

Zbog svega navedenog, snimanje traga je postalo nužan način nadzora korisničke aktivnosti u informacijskim sustavima. Kako većina informacijskih sustava u pozadini ima barem jednu bazu podataka, a kako je vrlo velika većina baza podataka zasnovana upravo na relacijskom modelu podataka, ovaj rad se ograničava na snimanje traga u relacijskim bazama podataka.

Zadatak rada je istražiti načine snimanja traga, odrediti komponente sustava za upravljanje bazama podataka čiji je rad potrebno snimati, ocijeniti mogućnosti postojećih rješenja za snimanje traga, te predložiti rješenje snimanja traga korisničke djelatnosti na razini baze podataka u cilju forenzičnih i sigurnosnih analiza. Predloženo rješenje je potrebno isprobati u svakodnevnom radu informacijskog sustava, te prezentirati rezultate istraživanja.

2. Društveni aspekti snimanja traga

2.1. Povijest snimanja traga

Praćenje traga, u smislu nadzora, prvenstveno poslovnih organizacija, postojalo je i prije računalnih informacijskih sustava. Osobe zadužene za nadzor imale su dužnost izvještavati o financijskom stanju i promjenama unutar tvrtke. Postupnom informatizacijom poslovanja pojavljuje se mogućnost automatiziranja ovih poslova, ali i potreba za nadzorom rada samih informacijskih sustava, u kontekstu pristupa i uporabe podataka. Kako zbog sigurnosti informacijskog sustava, tako i zbog zaštite privatnosti podataka, s vremenom se sve više pozornosti posvećuje računalnom snimanju traga i njegovoj kontroli.

Nacionalni centar za računalnu sigurnost u SAD-u je 1983. godine definirao Kriterij za evaluaciju povjerljivih računalnih sustava (eng. *Trusted Computer System Evaluation Criteria*) s ciljem uspostavljanja metrike za evaluaciju sigurnosti računalnih sustava. Kriterij je razdijeljen u četiri odjeljka: D, C, B i A, koji su hijerarhijski poredani, tako da se oznaka A dodjeljuje sustavima najveće sigurnosti. Odjeljci C i B su podijeljeni u nekoliko razreda, koji su također hijerarhijski poredani, kako bi se moglo odrediti različite razine sigurnosti unutar ovih odjeljaka. Pri tom B3 označava razred veće sigurnosti od razreda B2.

Navedena razdioba sigurnosnih parametara također određuje i pravila za snimanje traga [NCSC1987]. Svaki viši razred u hijerarhiji dodaje nove zahtjeve na one koje postoje u prethodnom nižem razredu. U tablici su prikazani događaji koje je obvezno snimati u računalnim sustavima, te informacije koje se pri tom obvezno snimaju ili ih je preporučljivo snimati, za razrede u kojima je snimanje traga definirano (od C2 do A1).

Razred	Događaji koji se snimaju	Informacije koje se prikupljaju
C2	Obavezno: • Uporaba identifikacijskih i autentikacijskih mehanizama • Stvaranje i brisanje korisničkih objekata • Akcije koje poduzimaju računalni operatori i sistemski administratori • Svi događaji vezani uz sigurnost • Ispis	Obavezno: • Vrijeme događaja • Jedinstveni identifikator onoga tko je generirao događaj • Tip i uspješnost događaja • Izvor zahtjeva za autentikaciju • Ime objekata kojima je pristupano, stvarani su ili brisani • Izmjene koje su administratori napravili na sigurnosnim postavkama
B1	Obavezno: • Svako premošćivanje izlaznih uređaja • Promjena odredišta komunikacijskih kanala ili ulazno-izlaznih uređaja • Promjena razine osjetljivosti komunikacijskih kanala • Promjena dosega bilo kojeg	Obavezno: • Sigurnosna razina objekta Preporučljivo: • Razina osjetljivosti uzročnika događaja

	višerazinskog komunikacijskog kanala	
B2	Obavezno: • Događaji koji mogu promijeniti zaštićene kanale za pohranu	Ne dodaju se nove informacije
B3	Obavezno: • Događaji koji mogu ukazati da slijedi narušavanje sigurnosne politike sustava	Ne dodaju se nove informacije
A1	Ne dodaju se novi događaji	Ne dodaju se nove informacije

Tablica 2.1. Događaji i informacije koje se prikupljaju u različitim razredima snimanja traga

U kontekstu rada sustava za upravljanje bazama podataka, od navedenih razreda je najzanimljiviji razred C2, zbog najjednostavnije implementacije traženih zahtjeva, ali i zbog toga što ti su zahtjevi zapravo najprilagođeniji sustavu za upravljanje bazama podataka. Neki od komercijalnih sustava daju eksplicitne upute za podešavanje kako bi se postigla usklađenost s C2 razredom.

Ideja snimanja traga u samim bazama podataka postoji već dugi niz godina. Na radionici o snimanju traga 1975. godine [Berg1975] se raspravljalo o snimljenom tragu, ulozi osobe zadužene za analizu traga, sučelju baza podataka prema drugim programskim rješenjima vezanim uz praćenje traga. Između ostalih zaključaka, osobama zaduženim za analizu traga napomenuto je kako su podaci koji se čuvaju u bazama podataka osjetljiviji od redundantnih podataka koji su pohranjivani u datotekama, te da se potencijalne hazardne situacije vezane uz baze podataka moraju uključiti u redovite izvještaje i evaluacije.

2.2. Zakonske norme koje utječu na snimanje traga u bazama podataka

Iako postoji znatan broj sigurnosnih proizvoda i metodologija, sama tehnologija nije dovoljna za očuvanje sigurnosti bilo kojeg informacijskog sustava. Uz tehnologiju, potrebna je volja da se shvati važnost sigurnosti i ulaganja u sigurnosna rješenja koja će jamčiti sigurnost i privatnost podataka. Ova volja ne postoji uvijek – nekad se radi o nedostatku sredstava, a nekad jednostavno zbog činjenice da ulaganje u sigurnost ne pokazuje jasno povratak ulaganja (eng. *return of investment – ROI*).

Velike svjetske tvrtke shvaćaju da je potrebno kontinuirano ulagati u zaštitu vrijednih informacija, na isti način kao što ljudi štite i osiguravaju kuće ili automobile. Jedan incident u kojem će biti ukradene ili oštećene važne informacije može naškoditi desetogodišnjim ulaganjima u tvrtku, a ozbiljni incidenti mogu trajno naškoditi tvrtci. Ipak, za one tvrtke koje još uvijek nisu osvijestile potrebu za ulaganjem u sigurnost i zaštitu podataka, zakonodavci stvaraju veliki broj pravilnika i zakona čiji je cilj nametnuti zaštitu informacija i privatnosti. Svaki novi incident u vezi sigurnosti i zaštite podataka u kojem su oštećene tvrtke ili privatne osobe, uzrokuje povećanje broja pravila kojih se treba držati.

Važnost ovih pravilnika u aspektu snimanja traga u bazama podataka nije potrebno posebno naglašavati – svi informacijski sustavi u pozadini sadrže određeni broj relacijskih baza podataka. Zaštita informacijskog sustava se u konačnici odnosi na zaštitu relacijske baze podataka.

Pravilnici i zakoni se odnose na države u kojima se primjenjuju. Također, neki od njih se odnose samo na pojedine segmente tržišta ili samo na pojedine tvrtke, ovisno o veličini. U ovom poglavlju dan je kratak pregled stanja u SAD-u i Europi, te u Republici Hrvatskoj. Velika većina normi, uz određene specifičnosti, zapravo obvezuje tvrtke da imaju preciznu evidenciju tko je kada pristupio kojim podacima i na koji način, da o ovome izvještavaju po potrebi, te da preuzmu odgovornost ukoliko nešto krene kako nije bilo predviđeno.

2.2.1. Sjedinjene Američke Države

U Sjedinjenim Američkim Državama su na snazi razne pravne norme koje se odnose na privatnost i zaštitu podataka. Uglavnom ne postoje norme koje bi se odnosile na cijelu državu, ili na cjelokupnu djelatnost evidentiranja podataka, nego su koncentrirane na pojedine sektore ili se radi o zakonskim regulativama pojedinih država unutar SAD-a.

Health Insurance Portability and Accountability Act

Health Insurance Portability and Accountability Act (HIPAA) zakon je koji je na snazi u Sjedinjenim Američkim Državama od 1996, a odnosi se na zaštitu podataka u zdravstvenom sektoru [Congress1996]. Osnovne ideje ovog zakona su smanjiti prijevare na zdravstvenom osiguranju, pojednostavljenje administracije u cilju povećavanja učinkovitosti zdravstvenog sustava te zaštita osobnih zdravstvenih podataka. Zakon se odnosi na sve sudionike u zdravstvenom sustavu – državne službe, pružatelje zdravstvenih usluga, ali i sve tvrtke koje pokrivaju zdravstvene troškove svojim zaposlenicima. Velika pozornost posvećena je privatnosti podataka pacijenata, te se za svaki slučaj „curenja“ takvih informacija predviđa novčana kazna do 250.000$ i do 10 godina zatvora za odgovornog voditelja. Također su uključeni i zahtjevi poput provjerljivosti sigurnosnih pravila, što implicira snimanje traga u bazama podataka.

Gramm-Leach-Bliley Act (GLBA)

Ovdje se radi o zakonu iz 1999. godine koji je na snazi u Sjedinjenim Američkim Državama [Congress1999]. Zakon se odnosi na poslovanje financijskih institucija, a dio zanimljiv za kontekst sigurnosti baza podataka propisuje zahtjeve za: osiguravanje sigurnosti i privatnosti korisničkih podataka, zaštitu od mogućih napada na sigurnost i integritet korisničkih podataka te zaštitu od neautoriziranog pristupa i korištenja ovih podataka.

Sarbanes-Oxley Act (SOX)

Zakon koji je u kongresu SAD-a prihvaćen 2002. godine s velikom većinom, zapravo je odgovor vlade na zabrinutost i sve veću svjesnost javnosti o upravljanju velikim tvrtkama, sukobima interesa i nedostatku transparentnog financijskog izvještavanja koji su u prošlosti uzrokovale velike štete ulagačima [Congress2002]. Odnosi se na sve tvrtke koje imaju godišnje prihode veće od 75 milijuna dolara, ali i na tvrtke koje

nisu izvorno iz SAD-a, ali obavljaju poslove u toj državi. Zbog širokog područja poslovanja na koje se odnosi, ovaj je zakon trenutno najvidljiviji u javnosti, i velike tvrtke troše prilične iznose kako bi ga ispoštovale. Pojavile su se i brojne tvrtke koje nude konzultantske usluge u cilju usuglašavanja s ovim zakonom.

Veliki dio ove norme nema veze s IT industrijom – uglavnom se radi o strožem definiranju uloga upravljačkih funkcija u tvrtkama. No, između ostalog, traži se i da menadžment potpisuje definicije i politike snimanja tragova u informacijskim sustavima, kao i uspostavu odbora za nadzor snimljenih tragova i sigurnosne analize informacijskih sustava, što se direktno odnosi i na snimanje tragova u bazama podataka i analize istih.

California Senate Bill 1386

Ova zakonska norma je vrlo važna, iako je na snazi samo na području Savezne Američke Države Kalifornije (od 2002. godine, [CS2002]). Između ostaloga, propisuje i da svaka državna agencija, osoba, ili poslovni subjekt koji vodi posao u državi a koji posjeduje osobne podatke bilo kojeg stanovnika u digitalnom obliku, mora osigurati mogućnost da sazna kada su koji podaci mogli biti korišteni od neautoriziranih korisnika. Ovaj se zakon nadovezuje na dugi niz zakonskih akata koji su na snazi u Kaliforniji, ali jasno nameće potrebu za snimanjem traga, a postoje i indikacije da se će se slična regulativa uskoro nametnuti na razini cijelog SAD-a.

2.2.2. Europa

Pravne norme Europske unije funkcioniraju na način da Europska komisija donosi razne direktive koje zemlje članice moraju implementirati kroz svoje zakone. U usporedbi s načinom zakonodavstva u SAD-u, može se primijetiti kako se u SAD-u pravne regulative donose uglavnom stihijski, prema potrebi, za razliku od EU, gdje se pokušavaju donositi ispravne direktive koje će vrijediti na razini cijele unije.

U europskom zakonodavstvu pojam *auditing* se uglavnom veže uz praćenje i evaluaciju rada tvrtki i agencija, te u tom smislu implicira snimanje traga osjetljivih podataka.

Osnova za zaštitu podataka je direktiva 95/46/EC. Puni naziv ove direktive Europske unije je: Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data [EP1995]. Radi se o direktivi koju sve članice EU moraju uključiti u svoje zakone. Ova regulativa se prvenstveno odnosi na privatnost podataka, čemu EU poklanja veliku pozornost, bilo da se radi o digitalnoj obradi ili ne, a uz privatnost su vezani i sigurnost te osiguravanje sigurnosti obrade podataka. Logičan mehanizam kontrole sigurnosti obrade podataka je i snimanje traga.

Slijedom ove direktive o zaštiti podataka, redovito nastaju nove, koje su prvenstveno usmjerene ka zaštiti građana unije u elektroničkim komunikacijama ([EP2002a], [EP2002b]). Korak dalje prema zaštiti podataka predstavlja i Konvencija vijeća Europe o zaštiti osoba u svjetlu automatskog procesiranja podataka [CE1981]. Iako su ovi dokumenti na vrlo visokoj razini u odnosu na samo snimanje traga unutar bilo kojeg informacijskog sustava, daju pozitivne smjernice svim akterima oblikovanja i korištenja tih sustava.

2.2.3. Hrvatska

U procesu pridruživanja Hrvatske Europskoj uniji, provodi se i usklađivanje zakonodavstva. Zakonske norme koje se odnose na zaštitu i sigurnost podataka već su donesene. Zakonom o zaštiti osobnih podataka [Sabor2003] osnovana je Agencija za zaštitu osobnih podataka (AZOP). Ovaj je Zakon temelj za nekoliko uredbi Vlade Republike Hrvatske, među kojima je za ovu temu najzanimljivija Uredba o načinu pohranjivanja i posebnim mjerama tehničke zaštite posebnih kategorija osobnih podataka u kojoj se među ostalim zahtijeva da svaki pristup telekomunikacijskom i računalnom sustavu za vođenje zbirki osobnih podataka mora biti automatski zabilježen korisničkim imenom, nadnevkom i vremenom prijave i odjave [Vlada2004].

Osim navedenog, posebnim zakonom [Sabor2005] je potvrđena Konvencija Vijeća Europe o zaštiti osoba u svjetlu automatskog procesiranja podataka [CE1981], a promijenjen je i tekst Ustava, u koji je uključeno i da se „zakonom uređuje zaštita podataka te nadzor nad djelovanjem informatičkih sustava u državi“ [Sabor2001].

3. Uvodna razmatranja o snimanju traga u bazama podataka

U sklopu razmatranja o snimanju traga u relacijskim bazama podataka, u ovom poglavlju dana su pojašnjenja pojmova vezanih uz baze podataka i snimanje traga. Posebna pažnja posvećena je vjerodostojnosti snimljenog traga pojašnjenjem principa raspodjele dužnosti pri snimanju traga i razmatranjima o zaštiti traga.

3.1. Osnovni pojmovi vezani uz teoriju baza podataka

Baza podataka (eng. *database*) najjednostavnije se može odrediti kao skup povezanih podataka. Jedna od mnogih definicija baze podataka bila bi: baza podataka sastoji se od skupa postojanih podataka koje koriste aplikacijski sustavi nekog projekta [Date2000]. Pri tom autor postojanost podataka opisuje kao nemogućnost nestanka podataka iz baze podataka osim eksplicitnim zahtjevom.

Sustav za upravljanje bazama podataka (SUBP, eng. *Database Management System – DBMS*), programski je proizvod koji omogućava rad s bazama podataka – definiranje, upravljanje, zauzimanje fizičkog prostora za smještaj podataka, zauzimanje računalnih resursa potrebnih za rad, omogućavanje korisnicima spajanja, posluživanje korisnika, obavljanje internih zadaća.

Poslužitelj baza podataka je računalo na kojem se (uglavnom neprekidno) izvršava sustav za upravljanje bazama podataka.

Operacijski sustav u kontekstu baza podataka, je operacijski sustav koji se izvodi na poslužitelju baza podataka, a unutar kojeg se izvršava sustav za upravljanje bazama podataka.

Administrator operacijskog sustava u kontekstu baza podataka, jest administrator samog poslužitelja baza podataka.

Administrator sustava za upravljanje bazama podataka je korisnik poslužitelja baza podataka čija je zadaća instaliranje, upravljanje i održavanje sustava za upravljanje bazama podataka.

Administrator baze podataka (eng. *Database Administrator – DBA*) je korisnik sustava za upravljanje bazama podataka koji je zadužen za definiranje, upravljanje i održavanje pojedine baze podataka unutar SUBP-a.

Relacijska baza podataka je baza podataka temeljena na relacijskom modelu podataka (eng. *relational data model*). Relacijski model podataka razvijen je 70-tih godina 20-tog stoljeća i temelji se na formalnoj matematičkoj osnovi [Codd1970]. Informacije se u relacijskom modelu opisuju kroz tvrdnje korištenjem predikatne logike. Za opis strukture podataka u relacijskom modelu koriste se elementi: entitet, atribut, domena, relacijska shema, relacija, n-torka, shema baze podataka.

Entitet (eng. *entity*) je nešto što ima suštinu ili bit i posjeduje značajke po kojima se može razlikovati od svoje okoline.

Atribut (eng. *atribute*) je karakteristika entiteta i definira se nad domenom. Atribut se također naziva i stupcem. Atributi se označavaju slovima A, B, C, ..., a skupovi atributa sa X, Y, Z, ….

Domena (eng. *domain*) je skup dopuštenih vrijednosti atributa i definira značenje podataka. Domena atributa A označava se s D = DOM(A).

Relacijska shema ili intenzija R, predstavlja imenovani skup atributa $\{A_1, A_2, \ldots, A_n\}$.

Relacija (eng. *relation*) ili ektenzija r definira se nad relacijskom shemom R i predstavlja konačan skup n-torki. Stupanj relacije je broj atributa relacije. Kardinalnost relacije je broj n-torki u relaciji. Relacija se naziva i tablicom. Relacija r definirana nad shemom $R = \{A_1, A_2, \ldots, A_n\}$ označava se s r(R) ili $r(A_1, A_2, \ldots, A_n)$.

n-torka (eng. *tuple*) t je istinita tvrdnja u kontekstu relacije i sadrži vrijednosti atributa koje odgovaraju atributima iz relacije. n-torka se također naziva i retkom. $t(A_1, A_2, \ldots, A_n)$ je n-torka definirana na shemi $R = \{A_1, A_2, \ldots, A_n\}$. t[X] je restrikcija n-torke na skup atributa X.

Vrijednost atributa je vrijednost konkretnog atributa u konkretnoj n-torki, pri čemu ta vrijednost mora pripadati domeni. Vrijednost atributa A u n-torki t se označava s t[A].

Shema baze podataka je skup relacijskih shema i označava se s $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$. Instance baze podataka **r** na shemi baze podataka **R** je skup relacija $\{r_1, r_2, \ldots, r_n\}$, pri čemu je relacija r_i definirana na relacijskoj shemi R_i.

Sustav za upravljanje relacijskim bazama podataka (eng. *Relational Database Management System – RDBMS*) je sustav za upravljanje bazama podataka temeljenima na relacijskom modelu podataka. Omogućava izvođenje naredbi SQL (*Structured Query Language*) jezika nad bazama podataka koje sadrži. SQL jezik je namijenjen manipulaciji podataka u relacijskim bazama podataka.

Pohranjena procedura (eng. *stored procedure*) je objekt u relacijskoj bazi podataka čiji se programski kôd temelji na nekom proceduralnom programskom jeziku [Stonebraker1987]. Ovaj se programski kod može izvršavati nad bazom podataka eksplicitnim pozivom u bilo kojem trenutku.

Okidač (eng. *trigger*) je objekt u relacijskoj bazi podataka čiji se programski kôd automatski izvršava kao reakcija na određeni događaj. Tipični događaji su operacije unosa, izmjene ili brisanja n-torki relacije [Dayal1995]. Programski kôd okidača može sadržavati operacije unosa, izmjene i brisanja podataka, izazivanja iznimki, ili pozive izvršavanja pohranjenih procedura koje mogu sadržavati složenu poslovnu logiku.

Pogled (eng. *view*) je objekt u relacijskoj bazi podataka definiran SELECT SQL naredbom nad relacijama baze podataka. Pogled ne sadržava konkretne podatke, nego se njegov sadržaj evaluira pri svakoj njegovoj uporabi.

3.2. Snimanje traga

Snimanje traga ili **praćenje traga** (eng. *auditing*) je evidentiranje događaja (eng. *event*) u sustavu za upravljanje bazama podataka i samoj bazi podataka. Događaji su posljedica korisničkih operacija nad sustavom ili bazom podataka, a odnose se na cjelokupnu djelatnost sustava – pristupanje i izmjene postavki, objekata u bazama podataka, korisničkih podataka, samih podataka u bazama podataka itd. Djelatnost aktivnog pregleda traga koji se upravo snima, najčešće u cilju kontrole i eventualnog upozoravanja, također ulazi u opis snimanja traga.

Snimljeni trag (eng. *audit trail*) je produkt snimanja traga – zapisane informacije o događajima koji se snimaju.

Sastavni dio snimanja traga jest i **analiza snimljenog traga**. To je djelatnost periodičnog pregleda snimljenog traga u cilju dobivanja adekvatnih informacija o uporabi sustava. Ciljevi analiza mogu biti različiti: provjera sigurnosne politike, forenzične istrage, izrada statističkih izvještaja, itd. Analize je moguće vršiti osobno direktnim pristupom snimljenom tragu, ili programski, s unaprijed pripremljenim alatima. Kako svaki sustav za snimanje traga producira različite tragove, tako se o analizama ne može određenije govoriti, jer ovise i o sustavu, ali i o željenim primjenama.

Snimanje traga nipošto ne treba promatrati kao sigurnosnu metodu. Ovdje se radi isključivo o načinu provjere funkcioniranja raznih sigurnosnih metoda sustava. U informacijskim sustavima snimanje traga treba koristiti od samog početka uporabe sustava, pogotovo nad sigurnosnim postavkama. Važno je imati kvalitetnu informaciju o promjeni istih, a informacijski sustav može biti ranjiviji pri početku uporabe zbog potencijalno nedovoljno testiranih ili neadekvatno implementiranih sigurnosnih postavki.

Vlasnici informacijskog sustava kod kojeg se primjenjuje snimanje traga također zbog toga ne smiju razviti lažni osjećaj sigurnosti. Iako postoji snimanje traga, ono neće zaštiti sustav ni na koji način ukoliko se snimljeni trag ne koristi u tu svrhu. Ovo dovodi do važnosti ispravnosti snimljenog traga. Potrebno je imati potpuno povjerenje u kompletnost i nekompromitiranost snimljenog traga. To znači da snimanje traga ne smije ovisiti o administratorima sustava i da bi analizu traga trebala obavljati neovisna osoba ili grupa. Osim toga, već snimljeni trag treba zaštititi. O ove dvije teme je riječ u sljedeća dva podpoglavlja.

3.3. Raspodjela dužnosti

Mnoge implementacije sigurnosnih politika sustava (ne samo informacijskih) u konačnici se svode na potrebu za ultimativnim povjerenjem u nadležnu osobu. U svim takvim situacijama, gdje ljudske osobine poput neiskrenosti, pohlepe, neurednosti, lijenosti itd. mogu oštetiti ili kompromitirati sustav, ustaljena je praksa raspodjela dužnosti (*segregation of duties*, *role separation*). Ideja koja stoji iza ovog principa je da se ne može vjerovati jednoj osobi ili jednoj grupi, nego se kritični proces odvija na način da je razbijen na nekoliko slojeva, pri čemu se slojevi ne preklapaju, a za svaki sloj je zadužena jedna osoba ili grupa.

U kontekstu snimanja traga u relacijskim bazama podataka, razdioba dužnosti ponajprije znači da moraju postojati osobe, koje nisu ni korisnici, niti administratori sustava, niti su članovi razvojnog tima projekta, a koje će biti zadužene za snimanje traga. Od ovih, jedna osoba ili grupa određuje što će se i kako snimati, a druga osoba ili grupa je zadužena za čitanje i analiziranje snimljenog traga. Snimljeni trag nitko ne bi trebao moći mijenjati.

Posebna pažnja treba biti posvećena administratorima sustava, koji zapravo ne bi smjeli imati veze s procesom snimanja traga. Svako uplitanje administratora u samu definiciju događaja koji se snimaju, a pogotovo u snimljeni trag dovodi u pitanje smisao snimanja traga. Za razliku od snimanja traga koje provodi sam sustav za upravljanje bazama podataka, vanjski sustavi za snimanje traga obično su izdvojeni s poslužitelja baza podataka i kao takvi imaju bolju perspektivu za ispravno implementiranje razdiobe dužnosti.

3.4. Zaštita snimljenog traga

Zaštita snimljenog traga igra važnu ulogu u cijelom informacijskom sustavu. Sustav s kompromitiranim snimljenim tragom jednako je informacijski vrijedan kao i sustav u kojem uopće nije implementirano snimanje traga.

Najjednostavniji način zaštite snimljenog traga od vanjskih utjecaja jest zapisivanje traga na medij koji omogućava jedino dopisivanje novih podataka (*Write Once Read Multiple – WORM*). Iako učinkovita, ova metoda podrazumijeva da trag koji se zapisuje nije kompromitiran u tom trenutku, te nema načina za detektiranjem kompromitiranog traga.

Prijedlog sigurnijeg rješenja dan je u [Schneier1999]. Ideja autora polazi od pretpostavke da trag nastaje na računalu u koje se nema apsolutno povjerenje, te ga je potrebno prenijeti na drugo računalo za koje se vjeruje da je sigurno. Trag se kriptiran prenosi na drugo računalo. Prvo računalo poznaje inicijalni tajni autentikacijski ključ. Zapisi koji se šalju povjerljivom računalu predstavljaju trag koji kriptiran nizom reverzibilnih transformacija ovisnih o tom autentikacijskom ključu. Nakon svakog poslanog traga se računa novi autentikacijski ključ na osnovu prethodnog, a prethodni se uništava. Na ciljnom računalu se trag dekriptira i pohranjuje. Ukoliko neki trag bude kompromitiran, to će se detektirati već prilikom procesiranja sljedećeg poslanog traga. Potencijalni napadač nije u mogućnosti čitati zaštićeni trag jer se on nalazi na povjerljivom računalu, nego mu je dostupan samo mali komad traga koji još uvijek nije upućen na drugo računalo.

Snodgrass predlaže ugradnju mehanizama unutar sustava za upravljanje bazama podataka, koji se zasnivaju na jednosmjernim kriptografskim funkcijama, a koji bi spriječili uljeza, zaposlenika, osobu zaduženu za analizu traga ili pogrešku unutar samog SUBP-a da izmijeni snimljeni trag [Snodgrass2004]. Predložena je i implementacija rješenja koja vrlo malo utječe na performanse SUBP-a, a kojom je moguće učinkovito utvrditi je li snimljeni trag kompromitiran.

Snimljeni trag, osim potrebne zaštite od vanjskih utjecaja, treba provjeravati kako bi se osigurao njegov integritet. Provjeravanje traga može se obavljati uspoređivanjem zapisanog traga i aktualnog stanja podataka, ali je moguća i posredna provjera,

usporedbom rezultata analiza snimljenog traga i stvarnoga stanja, u slučajevima u kojima je to moguće, kao i usporedbom rezultata analiza iz različitih izvora.

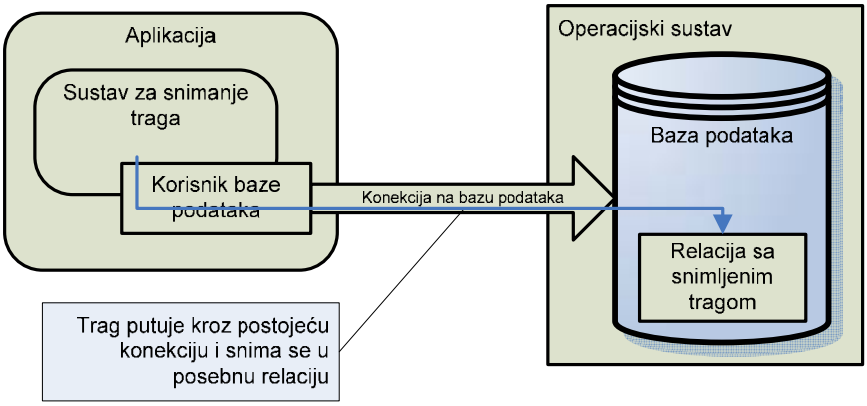
4. Metode snimanja traga u radu sustava za upravljanje bazama podataka

Način snimanja traga prvenstveno je određen mjestom gdje se trag počinje snimati. Drugi kriteriji, kao što su lokacija na koju se trag snima, format snimljenog traga, uloge korisnika pri definiranju i analiziranju snimljenog traga, mogu se shvatiti kao posljedična implementacijska rješenja uz izvorište snimanja traga. Način na koji se dolazi do informacije, traga kojeg se može snimiti, glavni je kriterij za razlučivanje metoda snimanja traga. S obzirom na to, moguće je odrediti nekoliko metoda za snimanje traga. To su:

- snimanje traga unutar klijentske aplikacije,
- snimanje traga unutar baze podataka,
- snimanje traga unutar sustava za upravljanje bazama podataka,
- snimanje traga izvan sustava za upravljanje bazama podataka.

4.1. Snimanje traga unutar klijentske aplikacije

Svaka klijentska aplikacija koja radi s udaljenom ili lokalnom bazom podataka točno zna koje podatke dohvaća ili mijenja. Nakon svakog takvog događaja je moguće snimiti trag onoga što je s podacima napravljeno, kako je prikazano na slici 4.1. Ovisno o implementaciji pojedinih aplikacija, ovo može biti vrlo složen postupak (ukoliko se na velikom broju mjesta u programskom kodu direktno pristupa bazi podataka) ili vrlo jednostavan (ukoliko su svi pristupi podacima usmjereni kroz jedno programsko sučelje koje komunicira direktno s bazom podataka). No, neovisno o tome, ovaj pristup snimanju traga je najlošiji od navedenih, i primjeren je samo u najjednostavnijim slučajevima u kojima sigurnost podataka nije od izrazite važnosti.



Slika 4.1. Snimanje traga koje započinje unutar klijentske aplikacije a trag se pohranjuje u relaciju baze podataka

Trag koji nastaje na ovaj način može se pohranjivati na neko središnje mjesto (najčešće dodatnu relaciju u radnoj bazi podataka, kojoj sve klijentske aplikacije mogu pristupiti). To implicira da klijenti mogu imati mogućnost direktnog pristupa

snimljenom tragu, ukoliko se npr. zapisivanje traga obavlja pozivom SQL naredbi nad relacijom u koju se trag snima. Ovaj postupak može se alternativno izvesti tako da se sve operacije nad bazom podataka obavljaju kroz pohranjene procedure, koje će onda sadržavati i naredbe za pisanje traga, a korisnici će imati dozvole samo nad tim procedurama. U ovom slučaju bilo bi potrebno implementirati pohranjene procedure za rad sa svim relacijama. Bilo da se radi o direktnom pristupu relacijama, ili im se pristupa kroz pohranjene procedure, ostaje činjenica da se na klijentskoj strani određuje što će biti zapisano u trag, što u svakom slučaju ostavlja mogućnost zloporabe.

Još jedna loša strana uporabe ove metode jest implicitno podrazumijevanje da se sve što se zbiva nad bazom podataka može poduzeti isključivo kroz klijentske aplikacije, tako da se u trag ne zapisuju druge djelatnosti nad bazom podataka. Ova pretpostavka je vrlo rijetko točna, čak i ako se radi o vrlo jednostavnom informacijskom sustavu, sa samo nekoliko korisnika. Administrativne akcije ili automatizirani pristupi drugih servisa česta su pojava u radu bilo koje baze podataka.

Primjerenost ove metode snimanja traga jest u potencijalnoj jednostavnosti implementacije, što ovisi o složenosti aplikativnih rješenja. Uz vrlo malo truda aplikativnih inženjera, koji se odnosi na implementiranje adekvatnog pristupa bazi podataka, moguće je imati jednostavan trag svih operacija nad bazom podataka izvedenih kroz pristupne aplikacije, uzevši u obzir ogradu o stvarnoj vrijednosti tog traga.

4.2. Snimanje traga unutar baze podataka

Trag koji nastaje snimanjem traga unutar pojedine baze podataka odnosi se samo na promatranu bazu podataka. Ovaj trag se najčešće sprema u datoteke koje se nalaze na poslužitelju baza podataka, unutar dodatnih relacija unutar promatrane baze podataka ili unutar relacija neke druge baze podataka na istom ili udaljenom poslužitelju baza podataka. Ova metoda svoje korijene vuče iz postojanja okidača nad relacijama – aktivnih objekata unutar baze podataka čiji se programski kod izvršava kao rezultat nekog događaja u bazi podataka. Kako se okidači izvršavaju samo pod određenim okolnostima, ovu se metodu ne može koristiti za snimanje svih potencijalno zanimljivih događaja u bazi podataka, već je njihova uporaba uglavnom ograničena na neke događaje izmjene podataka sadržanih u relacijama baze podataka. Neki sustavi za upravljanje bazama podataka omogućavaju implementaciju tzv. sistemskih okidača, koji se okidaju na razne događaje poput prijave korisnika za rad sa sustavom, izmjene objekata u bazi podataka i sl.

Implementacija ovakvog rješenja snimanja traga je prepuštena samim arhitektima i programerima informacijskog sustava. Najčešće je potrebno dodati programski kod koji će obaviti snimanje traga sa željenim podacima unutar svakog okidača koji se okida na događaj koji se želi pratiti – npr. izmjena podataka ili izmjena tipa podataka atributa relacije. Ukoliko okidač za promatrani događaj već ne postoji, potrebno ga je dodati u bazu podataka. Sve ovo čini ovu metodu snimanja traga priličnom složenom za implementaciju i održavanje.

Ukoliko se snimljeni trag čuva unutar relacija u radnoj ili nekoj drugi bazi podataka, tada sigurnost ovako snimljenog traga ovisi o sigurnosnim postavkama postavljenim

na razini baze podataka. Pristup tim relacijama treba biti moguć samo korisniku zaduženom za pregledavanje snimljenog traga, a bilo kakva izmjena tih podataka zabranjena svim korisnicima. Unatoč tome, dodatan oprez je potreban zbog omnipotentnih korisnika sustava – administratora baze podataka i administratora sustava za upravljanje bazama podataka. Zbog toga je ovu metodu snimanja traga potrebno kombinirati s nekom drugom, koja bi pratila aktivnost navedenih korisnika spram relacija sa spremljenim tragom, kako bi se osigurala čistoća snimljenog traga. Ukoliko se pak snimljeni trag čuva unutar neovisnih datoteka na poslužitelju baze podataka ili nekom drugom poslužitelju, tada se za taj trag primjenjuju sve sigurnosne odredbe kao i za druge zaštićene datoteke na poslužiteljima.

Dostupnost snimljenog traga ovom metodom također ovisi o mjestu na kojem se trag čuva. Ako se radi o tragu unutar neovisnih datoteka na poslužitelju, dostupnost traga je određena dostupnošću samog poslužitelja, pri čemu je raspoloživost datoteka određena sigurnosnim postavkama. U implementaciji metode koja snimljeni trag čuva u relacijama u promatranoj ili izdvojenoj bazi podataka, dostupnost traga je određena dostupnošću samog sustava za upravljanje bazama podataka odnosno baze podataka u kojoj su relacije sa snimljenim tragom. Raspoloživost ovog traga je određena gore spomenutim sigurnosnim postavkama primijenjenim nad ovim relacijama.

Ova metoda je prilično složena za implementaciju i održavanje te zahtijeva određeni angažman izvođača informacijskog sustava. Za potpunu sigurnost i nekompromitiranost snimljenog traga potrebno je kombinirati ovu metodu s nekom drugom, kojom se to može osigurati. S druge strane, ova metoda omogućava potpuno određivanje onoga što će se i kamo snimati. Prikladna je za korištenje u manjim informacijskim sustavima, ili onima sa smanjenim potrebama za snimanjem podataka. U većim sustavima ju je moguće koristiti ukoliko postoji programska podrška pomoću koje je moguće održavati cijeli sustav snimanja traga, i ako je osigurana konzistentnost snimljenog traga, kako će biti i pokazano u idućim poglavljima.

4.3. Snimanje traga unutar sustava za upravljanje bazama podataka

Komercijalni sustavi za upravljanje bazama podataka u pravilu uključuju neki oblik snimanja traga. Pri tom sam mehanizam za snimanje traga može biti implementiran kao proces koji aktivno promatra iste memorijske blokove u kojima sustav za upravljanje bazama podataka drži svoje operativne memorijske strukture, ili kao nusprodukt samog rada sustava, koji, ovisno o postavljenim parametrima na unaprijed određene lokacije sprema podatke o korištenju sustava. U nastavku ovog podpoglavlja dan je kratak pregled mogućnosti nekih komercijalnih sustava za upravljanje bazama podataka spram snimanja traga, uzevši u obzir prilagodljivost rješenja, način spremanja snimljenog traga te implementaciju razdiobe dužnosti.

4.3.1. IBM Informix

IBM Informix sustav za upravljanje bazama podataka omogućava snimanje traga na razini pojedinog poslužitelja baza podataka [IBM2007]. Za snimanje traga se koriste

dva dodatna alata isporučena s proizvodom – jedan za konfiguraciju i upravljanje procesom snimanja, a drugi za pregled snimljenog traga.

Snimanje traga obavlja se internim mehanizmima sustava za upravljanje podataka zasnovanim na praćenju zauzetih memorijskih segmenata. Postoji veliki broj događaja koje je moguće snimati, a definiranje koji će se događaji snimati se obavlja korištenjem maski za snimanje traga. Svaka maska sadrži popis događaja za snimanje traga. Moguće je definirati globalne maske, koje će vrijediti za sve korisnike koji se spoje na sustav za upravljanje bazama podataka, ili pak masku za svakog pojedinog korisnika. Globalne maske imaju predefinirana imena _default, _require i _exclude. Prilikom spajanja pojedinog korisnika na sustav za upravljanje bazama podataka, najprije se provjerava je li snimanje traga aktivno. Ukoliko jeste, sljedeća provjera jest postojanje individualne maske za tog korisnika. Ako ta postoji, tada se događaji navedeni u toj maski dodaju u popis događaja koji će se za ovog korisnika snimati. Ukoliko ova maska ne postoji, tada se provjerava postojanje maske _default, i ako ta postoji, onda se događaji iz te maske dodaju u popis onih koji će se za ovog korisnika snimati. Nakon toga se u taj popis dodaju svi događaji iz maske _require, ako ta postoji, a posljednji korak je uklanjanje svih događaja iz popisa koji su navedeni u maski _exclude. Postojanjem globalnih maski je olakšano održavanje i konfigurabilnost snimanja traga, a moguće je i stvaranje predložaka maski, na osnovu kojih je moguće jednostavno prilagođavati snimanje traga za pojedine korisnike.

Raspodjela uloga pri snimanju traga implementirana je postojanjem posebnih uloga koje imaju korisnici na razini operacijskog sustava. Standardna uloga je administrator sustava za upravljanje bazama podataka (eng. *Database Server Administrator – DBSA*), čija je zadaća konfiguriranje, održavanje i prilagodba sustava. Ova uloga postoji uvijek, i ukoliko druge uloge nisu implementirane na pojedinom sustavu, objedinjene su u ovoj. Ukoliko druge uloge (korisnici) postoje na sustavu, tada je utjecaj ove uloge na sigurnost sustava određen samo prilikom zadavanja početnih vrijednosti pri instalaciji sustava.

Osoba zadužena za sigurnost sustava za upravljanje bazama podataka (eng. *Database System Security Officer – DBSSO*) je ona osoba koja je zadužena za sve rutinske poslove održavanja sigurnost poslužitelja baza podataka. U kontekstu snimanja traga u bazama podataka, ti poslovi uključuju i konfiguraciju i održavanje maski za snimanje traga.

Osoba zadužena za analizu snimljenog traga (*Audit Analysis Officer – AAO*) je osoba koja obavlja poslove konfiguracije snimanja traga, te čitanje i analizu snimljenih tragova. Snimljeni trag se sprema u čitljive datoteke na poslužitelju baza podataka na mjesto kojem samo ovaj korisnik može pristupiti. Ukoliko ovaj korisnik ne postoji na operacijskom sustavu, onda snimljenom tragu može pristupiti samo administrator sustava za upravljanje bazama podataka.

Nedostatak ovakvog načina snimanja traga očituje se u nemogućnosti odabira objekata u bazi podataka na koje će se snimanje odnositi. Odabirom događaja i korisnika za koje se trag snima postiže se snimanje traga za svaki takav događaj koji će korisnik pokrenuti, bez obzira o kojoj se eventualnoj n-torki, relaciji ili bazi podataka na promatranom poslužitelju radi. Na primjer, ukoliko se za korisnika *ivan*

prati događaj čitanja podataka (SELECT naredba), vrlo velika količina snimljenog traga će biti zabilježena, jer će se trag snimiti za svaku n-torku koju korisnik *ivan* pročita iz bilo koje relacije. Ukoliko je za pregled traga zanimljiva samo jedna određena relacija, ovakav će način snimanja, osim prevelike količine snimljenog traga, rezultirat i prevelikom količinom nepotrebnih podataka. Ovaj nedostatak čini snimanje traga unutar ovog sustava za upravljanje bazama podataka uporabljivim samo za određene događaje, koji se ne pojavljuju često i odnose na sustav ili izmjenu objekata u bazi, dok ga se ne može uporabiti za najčešće događaje – čitanje i izmjenu podataka.

4.3.2. Oracle Database

Oracle Database omogućava dva načina snimanja traga [Oracle2007b] – standardni (eng. *standard auditing*) i detaljni (eng. *fine-grained*). Standardni se način koristi za praćenje SQL naredbi, dozvola, objekata u bazi podataka i mrežne aktivnosti. Detaljni način omogućava definiranje snimanja traga prema sadržaju, koristeći istinitost zadanog uvjeta, a također je omogućeno i postavljanje upozoravanja u slučajevima kršenja uvjeta postavljenih nad podacima. Nadgledanje rada administratora sustava za upravljanje bazama podataka obavlja se na poseban način – pisanjem posebnog dnevnika kojem bi pristup jedini morao imati administrator operacijskog sustava, kojeg se smatra korisnikom od povjerenja.

Snimljeni trag se sprema u dvije relacije koje su dio rječnika podataka, a u nekim slučajevima i u datoteci unutar operacijskog sustava. Sve DML operacije nad ovim relacijama se također zapisuju u njih, i to bez obzira je li snimanje traga uključeno.

Unutar standardnog načina snimanja traga omogućeno je:

- snimanje traga pojedinih SQL naredbi, na razini svih ili određenih korisnika baze podataka,
- snimanje traga događaja vezanih uz dozvole korisnika nad objektima u bazi podataka, na razini svih ili određenih korisnika baze podataka,
- snimanje traga naredbe koja se izvodi na određenom objektu u bazi podataka, primjenjivo na sve korisnike baze podataka,
- snimanje traga akcija koje se izvode u srednjem sloju aplikacija u ime klijenta,
- snimanje traga pogrešaka u mrežnim protokolima ili u mrežnom sloju.

Detaljni način snimanja traga omogućava snimanje čitanja podataka, DML operacija, te davanja i ukidanja dozvola nad relacijama baze podataka. Definiranje ovakvog načina snimanja obavlja se koristeći dodatni SQL paket (DBMS_FGA). Prednost ovakvog načina snimanja jest mogućnost samostalnog određivanja uvjeta pod kojima će se trag snimati. Upozoravanje se može uspostaviti na način da se napiše pohranjene procedure koje će se izvoditi u određenim situacijama.

Svaki korisnik može sam uključiti snimanje traga za objekte koji se nalaze u njegovoj shemi. Ukoliko posjeduje adekvatne dozvole, može na ovaj način snimati i trag za druge korisnike ili sistemske događaje. Administrator sustava za upravljanje

bazama podataka može uključiti ili isključiti snimanje traga na razini cijelog sustava. Ukoliko je na ovaj način snimanje traga isključeno, nijedan se trag neće snimati. Ovo se odnosi samo na standardni, a ne i na detaljni način snimanja.

Postojanje tzv. sistemskih okidača, onih koje okidaju događaji koji nisu usko vezani uz manipulaciju podacima u relacijama, omogućava još jedan način praćenja i reagiranja na događaje pokrivene ovim okidačima. Iako se radi o načinu snimanja koji je potrebno posebno implementirati, ovi okidači mogu biti korisni ukoliko je vrijednost snimljenog traga u promatranom sustavu upravo u postojanju povijesti umjesto u sigurnosnim analizama.

Cjelokupno upravljanje snimanjem traga obavlja se korištenjem dodatnih SQL naredbi, ne postoje posebni alati za ove akcije. Razina dozvola korisnika u sustavu za upravljanje bazama podataka određuje kako on može utjecati na snimanje traga. Pregled snimljenog traga, odnosno sadržaja relacija u koje se snimljeni trag sprema, omogućen je korištenjem posebnog pogleda, tako da se snimljeni podaci filtriraju prema dozvolama korisnika koji ih promatra. Definiranje i upravljanje detaljnim načinom snimanja omogućeno je korisnicima koji imaju dozvolu za rad s paketom DBMS_FGA, a inicijalno je to samo administrator sustava za upravljanje bazama podataka.

Iako je upravljanje snimanjem traga relativno složeno i dobro zamišljeno, nedostatak je jasna raspodjela dužnosti. Uloge koje trebaju postojati prilikom definiranja i nadgledanja snimanja traga nisu posebno naznačene, i najveća razina zaštite jest obavljanje zadataka u svojstvu administratora sustava za upravljanje bazama podataka. Također, prilikom samostalne implementacije raspodjele dužnosti u djelatnosti snimanja traga, uloge se moraju osloniti samo na dozvole korisnika u sustavu za upravljanje bazama podataka.

Navedeni su nedostaci ispravljeni u dodatnom proizvodu – Oracle Audit Vault [Oracle2007a]. Uvažavajući zakonske norme i potrebe tržišta, ovaj je proizvod moćan alat za snimanje i analizu traga u informacijskim sustavima. Zamišljen kao sigurno skladište snimljenih podataka, integrira se s ostalim produktima ovog proizvođača, te skuplja snimljene tragove iz više izvora, objedinjujući ih u jedno skladište podataka nad kojim se dalje izvode analize i upozorenja. Omogućena je i interakcija u drugom smjeru, pa tako Audit Vault služi i za upravljanje snimanjem traga. Razdioba uloga je detaljno razrađena, a uloge se dodjeljuju prije instalacije pojedinih komponenti. U ovom kontekstu, moguće je definirati odvojene korisnike s ulogama: administrator sustava za snimanje, osoba zadužena za pregled traga, arhivar snimljenog traga, vlasnik baze podataka sa snimljenim tragom, administrator korisničkih računa baze podataka sa snimljenim tragom. Svakom izvoru podataka koji se prati, a za koji postoji programski agent za pretakanje podataka, moguće je dodijeliti posebni korisnički račun s ulogom koja omogućava pretakanje tih podataka.

4.3.3. IBM DB2

Podsustav za snimanje traga kojeg koristi IBM DB2 sustav za upravljanje bazama podataka (*DB2 database audit facility*, [IBM2006]) u potpunosti je izdvojen od rada SUBP-a. Jedan i drugi mogu raditi neovisno jedan od drugoga, i ukoliko SUBP nije

aktivan, a podsustav za snimanje traga jest, on će moći snimati samo događaje koji se odvijaju izvan SUBP-a (kao npr. izmjena podsustava za snimanje traga). Kada se SUBP ponovno aktivira, snimaju se i ostali dostupni događaji.

Snimljeni trag se sprema u datoteku. U istom kazalu u kojem se nalazi ta datoteka nalazi se i konfiguracijska datoteka za alat za upravljanje snimanjem traga. Pristup za čitanje i pisanje ovih datoteka inicijalno ima vlasnik instance sustava za upravljanje bazama podataka. Alat za upravljanje snimanjem traga može koristiti samo administrator sustava za upravljanje bazama podataka.

Omogućeno je snimanje nekoliko kategorija događaja:

- snimljeni trag – događaji koji obuhvaćaju izmjenu postavki podsustava za praćenje traga i pristupe zapisanom snimljenom tragu,
- provjera dozvola – događaji provjere korisničkih dozvola prilikom pokušaja manipulacije objektima u bazi podataka ili izvršavanja funkcija,
- sigurnosne postavke – događaji pri promjeni korisničkih dozvola,
- administracija – događaji izazvani administratorskim akcijama administratora sustava za upravljanje bazama podataka i administratora baza podataka,
- provjera korisnika – obavlja se pri spajanju novog korisnika na sustav ili pri dohvatu sigurnosnih postavki sustava,
- kontekst operacija – ovisno o operaciji koja se obavlja nad bazom podataka, zapisuju se informacije uz te operacije kako bi se više operacija moglo smjestiti u zajednički kontekst, radi jednostavnijeg praćenja.

Snimanje traga korištenjem ovog podsustava može imati priličan utjecaj na performanse cijelog sustava za upravljanje bazama podataka, te je moguće odrediti hoće li se obavljati sinkrono ili asinkrono. Sinkroni način snimanja ima lošiji utjecaj na izvedbu sustava, a ukoliko se koristi asinkrono snimanje traga, koriste se posebni međuspremnici u kojima podaci ostaju do trenutka snimanja.

U ovoj implementaciji snimanja traga najjasniji je nedostatak mogućnosti implementacije raspodjele dužnosti. Sve akcije u vezi snimanja traga prepuštene su administratoru sustava za upravljanje bazama podataka ili pripadnoj grupi – određivanje postavki za snimanje, upravljanje snimanjem, kao i analiziranje snimljenog traga. Također, nije moguće definirati izričite događaje koje će sustav snimati, nego se opseg snimanja odnosi na cjelokupne prethodno navedene kategorije. U takvom okruženju, snimanje traga pojedinačnih operacija čitanja i izmjene podataka se ne bilježi svaki put, nego samo jednom – prilikom provjere dozvola korisnika za izvođenje operacije.

4.3.4. Microsoft SQL Server

U usporedbi s konkurentskim proizvodima, ovaj sustav za upravljanje bazama podataka posjeduje najslabije mogućnosti snimanja traga. SQL Server 2005 [Beauchemin2007, Ruebush2005] podržava snimanje traga na nekoliko načina, koji se međusobno nadopunjuju. Događaji pristupa objektima, provjere dozvola za

korištenje, autentikacije korisnika se bilježe u sigurnosni dnevnik operacijskog sustava – Windows Security Eventlog. Alat SQL Profiler je najvažniji alat za snimanje traga. Moguće je prilagoditi instancu sustava za upravljanje bazama podataka za praćenje željenih događaja. Snimljeni trag koji se pregledava korištenjem ovog alata također sadrži više informacija nego onaj zabilježen u sigurnosni dnevnik operacijskog sustava.

Korištenjem navedenih načina snimanja traga moguće je pokriti većinu zanimljivih događaja u sustavu za upravljanju bazama podataka, ali ne i operacije stvaranja (CREATE), izmjene (ALTER) i uništavanja (DROP) objekata u bazama podataka. Za snimanje ovih događaja, moguće je napisati programski kod za adekvatne sistemske okidače koje se može definirati - uporaba snimanja traga za DDL naredbe je ostavljena na implementaciju administratoru sustava.

Razdioba uloga nije do kraja implementirana. Za uporabu alata SQL Profiler postoji predefinirana uloga koju je moguće dodijeliti korisnicima koji nemaju administratorske ovlasti. Na ovaj način osoba zadužena za analizu snimljenog traga može imati odvojeni pristup tragu, ali ovaj se alat ujedno koristi i za definiranje samog snimanja, što ne bi trebalo biti omogućeno ovom korisniku (a isti se alat koristi i za druge operacije). Osim ovoga, administrator sustava za upravljanje bazama podataka može i definirati snimanje traga, i analizirati snimljeni trag, čak štoviše, to je njegova dužnost. Upravljanje snimanjem traga DDL naredbi ostavljeno je svim korisnicima koji smiju implementirati sistemske okidače, a na istima je i da definiraju način snimanja traga te dozvole za pristup istome.

Nepostojanje jedinstvenog načina za snimanje traga, jedinstvenog snimljenog traga, kao i nemogućnost implementacije razdiobe dužnosti čine ovaj sustav za upravljanje bazama podataka najmanje upotrebljivim od promatranih, s obzirom na potrebu kvalitetnog snimanja i analiziranja traga.

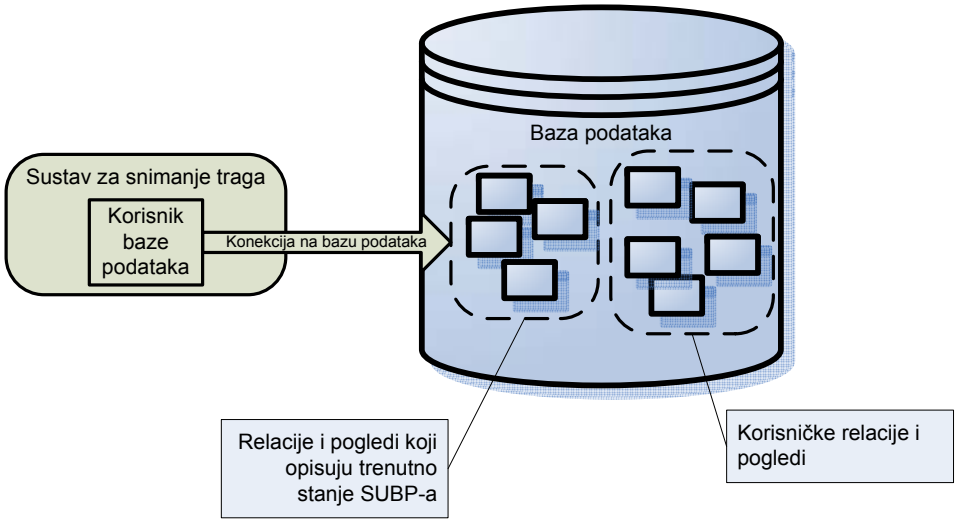
4.4. Snimanje traga izvan sustava za upravljanje bazama podataka

Svi sustavi za upravljanje bazama podataka imaju nekoliko zajedničkih arhitekturnih karakteristika: koriste mrežnu komunikaciju, međuprocesnu komunikaciju, transakcijske dnevnike (eng. *transaction logs*). Ove karakteristike omogućuju izgradnju vanjskih sustava za snimanje traga. Vanjski sustavi za snimanje traga pohranjuju snimljeni trag na izdvojene lokacije – u druge baze podataka ili datoteke koje mogu biti na samom poslužitelju baze podataka ili bolje, na nekom drugom poslužitelju.

Prema arhitekturi, ovakvi sustavi se mogu temeljiti na [Natan2005]:

- inspekciji unutarnjih podatkovnih struktura sustava za upravljanje bazama podataka
- inspekciji svih komunikacija sa sustavom za upravljanje bazama podataka
- inspekciji raznih elemenata koje tijekom rada stvara sustav za upravljanje bazama podataka.

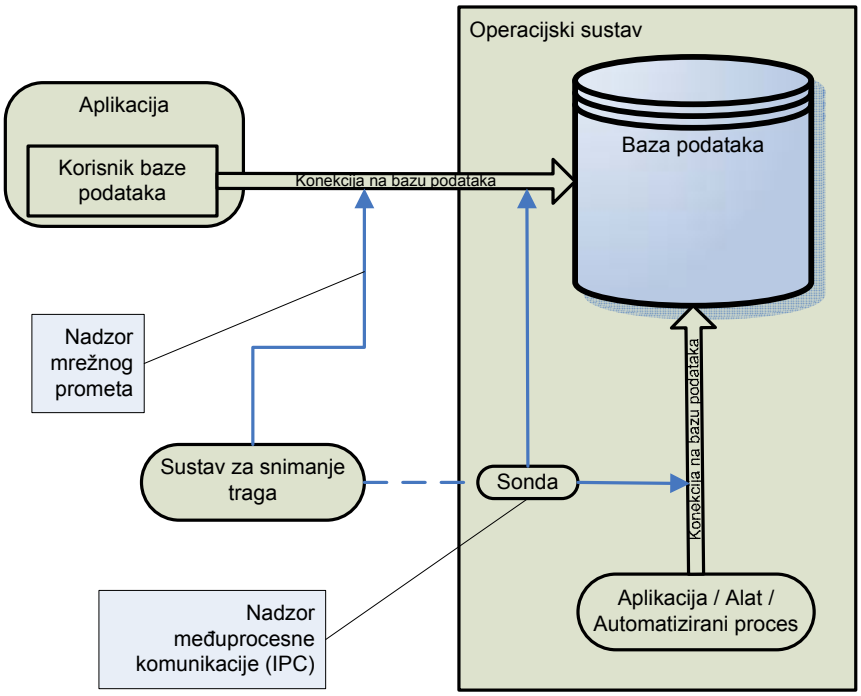
Sustavi za upravljanje bazama podataka imaju interne podatkovne strukture koje koriste za obradu naredbi, čuvanje rezultata i općenito funkcioniranje sustava. Često se tim podatkovnim strukturama može pristupiti kroz dodatne relacije i/ili poglede koji su dio rječnika podataka radne baze podataka ili dio neke posebne, sistemske baze podataka. Iza ovih relacija se kriju operativne memorijske strukture. Sadržaj i način pristupa ovim podacima je uglavnom dokumentiran i na taj način je omogućeno trećim proizvođačima da ponude vanjske sustave za snimanje traga koji će se spojiti na sustav za upravljanje bazama podataka i periodičkim upitima pratiti stanje baze podataka. Bitno je da taj period bude dovoljno mali kako bi se mogle pratiti sve izmjene, no opet ne toliko mali da bi osjetno djelovao na rad promatranog sustava. Na slici 4.2 je prikazana shema snimanja traga u bazi podataka pristupom ovim relacijama.



Slika 4.2. Snimanje traga pristupom relacijama s opisom trenutnog stanja SUBP-a

Drugi način snimanja traga vanjskim sustavom uključuje pregled komunikacija prema sustavu za upravljanje bazama podataka. SUBP djeluje kao poslužitelj, i kao takav, prima zahtjeve za spajanje, obrađuje ih i po potrebi održava. Praćenjem komunikacijskih tokova moguće je vidjeti sve što je od baze zahtijevano i što je došlo kao odgovor.

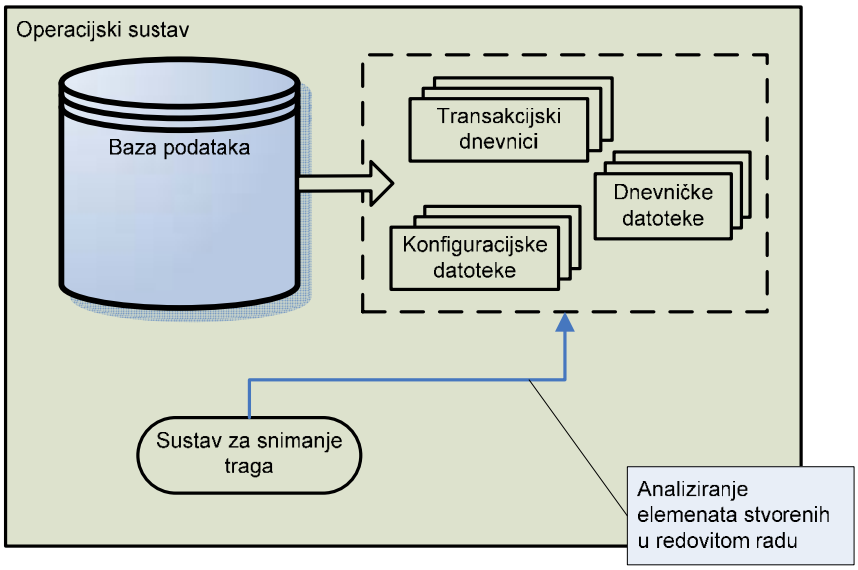
Konekcije na bazu podataka mogu biti lokalne ili mrežne. Klijenti se spajaju koristeći neki od mrežnih protokola ili mehanizme međuprocesne komunikacije (*interprocess communication* – IPC) ukoliko se radi o klijentima koji se izvode na poslužitelju baza podataka. Vanjski sustav za snimanje traga može obavljati inspekciju mrežnih paketa koji putuju konekcijama na sustav za upravljanje bazama podataka ili koristiti *sondu* unutar operacijskog sustava kojom će pratiti međuprocesnu komunikaciju, kao što je prikazano na slici 4.3.



Slika 4.3. Snimanje traga nadzorom komunikacija SUBP-s

Vanjski sustavi koji obavljaju inspekciju mrežnih paketa to mogu raditi na različite načine. Moguće je čitati pakete koji putuju mrežnom konekcijom i obavljati ad hoc analizu, ili za svaki paket kreirati njegov duplikat koji će kasnije biti analiziran. Također postoji mogućnost da se vanjski sustav za snimanje traga postavi kao most (*bridge*) između klijenata i baze podataka te na taj način sav promet doslovno putuje kroz njega, pri čemu ga se analizira.

Tijekom normalnog rada, svaki sustav za upravljanje bazama podataka stvara nekoliko elemenata koje je također moguće analizirati, što neki vanjski sustavi za snimanje traga i obavljaju. Najočitiji element za praćenje su transakcijski dnevnici. U ove dnevnike se zapisuje većina DML i DDL naredbi kako bi se prilikom eventualnog oporavka baze podataka mogle ponovno izvršiti. Vanjski sustav za snimanje traga može besprekidno čitati ove dnevnike i na taj način zabilježiti događaje koje će snimati. Također postoje i drugi elementi, kao što su dnevničke i konfiguracijske datoteke iz kojih je moguće rekreirati skoro svu aktivnost baze podataka – iskoristivost ovisi o pojedinom sustavu za upravljanje bazama podataka. Snimanje traga analiziranjem elemenata koje stvara SUBP tijekom rada je opisano na slici 4.4.



Slika 4.4. Snimanje traga analiziranjem elemenata koje pri radu stvara SUBP

Vanjske metode snimanja traga stvaraju najneovisniji trag. Utjecaj administratora baze podataka i administratora sustava za upravljanje bazama podataka na način snimanja ili na same rezultate snimanja je sveden na minimum. Sigurnost i dostupnost snimljenog traga određena je pozicijom na kojoj se taj trag nalazi, pri čemu se radi o uobičajenim načinima zaštite. Na ovaj način je olakšana razdioba dužnosti, pri čemu samo osobe zadužene za pregled traga mogu imati pristup istome.

Utjecaj ove metode na performanse sustava za upravljanje bazama podataka je vrlo mali, dok se ograničenja svode na ona koje posjeduje sam vanjski sustav za snimanje traga. Uz to, kako se radi o proizvodima trećih proizvođača, cijena je također čimbenik koji utječe na odluku implementacije snimanja traga.

Zbog velike pouzdanosti, ovu se metodu često koristi i kao dodatnu metodu snimanja traga. U informacijskim sustavima koji sadrže informacije velike važnosti, gdje je snimljeni trag važan kontrolni mehanizam, praktično je usporedno koristiti dvije metode snimanja traga. Ukoliko se paralelno koristi dvije metode za snimanje traga, tada je moguće uzastopno uspoređivati snimljene tragove i pravodobno upozoriti administratore zadužene za sigurnost sustava ukoliko dođe do razlike. To je gotovo siguran znak da je jedan od tragova kompromitiran.

4.5. Komercijalni sustavi za snimanje traga

Komercijalni sustavi za snimanje traga uglavnom koriste navedene metode snimanja traga. Veliki broj takvih sustava se zasniva na dodavanju okidača u radnu bazu podataka koji služe samo za snimanje traga, i kao takvi podržavaju sve verzije raznih sustava za upravljanje bazama podataka koje dopuštaju postojanje više okidača za istu operaciju nad relacijom. Drugi koriste vanjske metode snimanja traga. Ovdje su ukratko navedene njihove druge zajedničke karakteristike, kao i posebnosti.

Svi promatrani komercijalni vanjski sustavi za snimanje traga nude velik broj mogućnosti određivanja što će se i kako snimati. Ovi proizvodi trebaju podržavati

sve sklopovske platforme i operacijske sustave na kojima su implementirani sustavi za upravljanje bazama podataka koje podržavaju. Velika pozornost posvećuje se analizi snimljenog traga, pri čemu se uglavnom nudi određeni broj unaprijed pripremljenih analitičkih pregleda. Svi poslovi podešavanja i pregleda su implementirani u grafičkim korisničkim sučeljima. U određenim slučajevima je moguće definirati upozoravanje kako bi se odgovorne osobe moglo upozoriti na važne događaje.

DB Audit [SoftTree2005a, SoftTree2005b] je komercijalni proizvod namijenjen za rad sa sustavima za upravljanje bazama podataka: Oracle, Microsoft SQL Server, Sybase ASE i ASA te IBM DB2. Po načinu rada, ovo nije izdvojeni sustav za snimanje traga, već se radi o programskoj podršci koja se priključuje na radnu bazu podataka i u njoj implementira dodatne okidače čija je jedina namjena snimanje traga. Zbog toga može funkcionirati samo na onim verzijama navedenih sustava koji podržavaju postojanje više okidača za istu akciju nad pojedinom relacijom. Trag se snima u nove relacije unutar radne baze podataka ili u izdvojenoj bazi podataka na istom poslužitelju. U svim prikazima snimljenog traga izmjena podataka, uvijek se prikazuju vrijednosti prije i nakon izmjene, tako da je na jednostavan način moguće vidjeti povijest podataka.

Na sličan način snimanje traga obavljaju i ApexSQL Audit [ApexSQL2007], namijenjen isključivo za rad sa Microsoft SQL Server sustavom za upravljanje bazama podataka, kao i razni LogManager proizvodi, od kojih je svaki namijenjen za različiti SUBP, a podržani su Microsoft SQL Server [Upscene2006], Interbase, NexusDB i Advantage Database Server.

Lumigent Audit DB [Lumigent2007] je komercijalni vanjski sustav za snimanje traga namijenjen za rad sa sustavima za upravljanje bazama podataka: Microsoft SQL Server, Sybase ASE, IBM DB2, Oracle. Ovaj napredni vanjski sustav za snimanje traga kombinira tri načina praćenja: slušanje mrežnog prometa, čitanje transakcijskih dnevnika i uporaba unutarnjeg snimanja traga sustava za upravljanje bazama podataka, a konfiguracija svega se obavlja kroz jedno sučelje. Sav trag se smješta u zaštićeni repozitorij. Omogućava potpunu razdiobu dužnosti dodjelom različitih uloga korisnicima sustava.

Ambeo NetServer [Ambeo2007, Richardson2004] je sklopovski i programski sustav namijenjen snimanju traga u bazama podataka. Sklopovski dio sustava namijenjen je snimanju mrežnog prometa nadgledajući mrežne preklopnike ili preusmjeravajući cjelokupni promet prema i od poslužitelja baze podataka. Procesiranje podataka koje obavlja se također ne mora odraditi na drugim poslužiteljima, tako da je međudjelovanje ovog sustava s radnim sustavom svedeno na nisku razinu. Usklađen je za rad s programskom podrškom istog proizvođača, opremljenom korisnim analitičkim mogućnostima. Podržani su sljedeći sustavi za upravljanje bazama podataka: Oracle, Teradata, IBM DB2, IBM Informix, Sybase ASE, Sybase IQ, Microsoft SQL Server.

Tizor Mantra [Tizor2007] jedan je od najnaprednijih vanjskih sustava za snimanje traga. Također je zasnovan na sklopovskom rješenju, a primarni način rada je praćenje mrežnog prometa. Osim za praćenje rada poslužitelja baza podataka, namjena je proširena i s praćenjem raznih datotečnih poslužitelja. Programska

podrška omogućava definiranje pravila kojima se određuje što će se i kako snimati, a ugrađena je i dodatna inteligencija, prema kojoj sustav tijekom rada uči i prihvaća obrasce obavljanja poslova svakog pojedinog korisnika. Nakon što je obrazac prihvaćen, svako odstupanje može rezultirati upozoravanjem. Kao i kod drugih sustava, programska podrška omogućava brojne analize snimljenih tragova, alarmiranja i izvještaje. Implementirana je i potpuna razdioba dužnosti. Nedostatak ovog sustava za snimanje traga je relativno mali broj podržanih sustava za upravljanje bazama podataka, pri čemu nisu podržane sve novije verzije: IBM DB2, Microsoft SQL Server, Oracle te Sybase ASE.

5. Kategorije snimanja traga u bazama podataka

Postoji određeni broj različitih događaja koji mogu biti zanimljivi za snimanje traga. Ovi događaji mogu biti isprovocirani na različite načine, mogu se pojavljivati različitim intenzitetom, a tako i imati vrlo različito značenje za onoga tko prati snimljeni trag. Trag se za ove događaje može snimati na različite načine, navedene u prethodnom poglavlju, a neki način snimanja traga može pogodovati za praćenje određenog događaja, dok za drugi može biti apsolutno neprimjenjiv. Ovo poglavlje nudi odgovore na pitanja: što i kako snimati u radu sustava za upravljanje bazama podataka.

5.1. Prijava i odjava za rad s bazom podataka

U mnogim tvrtkama postoje sustavi praćenja tko i kada ulazi u prostorije tvrtke. Za ulaz u zaštićene prostorije, poput računalnih hala, potrebno se je svaki put autenticirati za otvaranje vrata u svakom smjeru. Podaci koji pri tom nastaju najčešće se čuvaju iz razloga da bi se, u slučaju da nešto pođe krivo, lakše moglo započeti identificirati tko je za to odgovoran. Na sličan način je potrebno postupati i s bazama podataka.

U svakom slučaju je potrebno znati kada su određeni korisnici bili prijavljeni za rad s bazom podataka. Pri tom je obvezno znati korisničko ime i vrijeme prijave i odjave, a mnoge regulacije zahtijevaju da se snima i lokacija (IP adresa) s koje se klijent spaja na bazu podataka. Osim toga, korisno je znati i koji računalni program je inicirao konekciju.

Treća vrsta događaja koja se također često veže uz ova dva prethodno navedena su neuspjeli pokušaji spajanja na bazu podataka. Neuspjeli pokušaji spajanja su važniji sa strane sigurnosti, i često su osnova za alarmiranje ili automatsko zaključavanje korisničkog računa. Relativno je jednostavno uspostaviti takvu kontrolu: potrebno je analizirati neuspjele pokušaje spajanja projicirane po korisničkom imenu, klijentskoj IP adresi te računalnom programu s kojeg dolaze zahtjevi za spajanje u određenom vremenskom periodu. Veliki broj neuspjelih pokušaja istog korisnika s istog mjesta u vrlo kratkom vremenu skoro sigurno indicira da se radi o pokušaju probijanja korisničke lozinke, od čega se trenutno može obraniti zaključavanjem korisničkog računa.

Snimanje traga prijava i odjava za rad s bazom podataka podržavaju svi važniji sustavi za upravljanje bazama podataka. Kako je broj ovih događaja relativno mali (pogotovo u usporedbi s brojem događaja koje se može dobiti ako se snimaju sve SQL naredbe), utjecaj na performanse SUBP-a pri obavljanju ove vrste snimanja je vrlo mali.

5.2. Rad s bazom podataka izvan uobičajenih obrazaca

Osim stvaranja traga prijava i odjava, snimanje ovih događaja može poslužiti kao osnova za detektiranje neobičnosti u korištenju baze podataka. Ideja je da se pomoću ovog traga, prateći ga kroz neko dulje vrijeme, odrede „normalni“ obrasci ponašanja u prijavama i odjavama za rad.

Na primjer, u informacijskom sustavu za podršku rada administrativnim poslovima visokoškolskih ustanova, kao što je Informacijski sustav visokih učilišta (ISVU, [Baranović2003a]) mogu se odrediti tri obrasca ponašanja prijava i odjava:

- studenti, koji se spajaju na bazu podataka kroz aplikacijski server, s točno određenim korisnikom (studomat), koriste bazu 24 sata dnevno;
- korisnici koji rade administrativne poslove na ustanovama spajaju se samo u okviru radnog vremena s različitih IP adresa izvan lokalne mreže;
- sistemski korisnici, koji predstavljaju različite servise za održavanje sustava, spajaju se samo noću s lokalnih IP adresa.

Na osnovu ovih obrazaca se mogu uspostaviti dodatne kontrole koje bi prijavljivale sva odstupanja od navedenog. Tako bi, na primjer, svako spajanje korisnika koji obavlja administrativne poslove na ustanovi izvan uobičajenog radnog vremena, ili s neke od lokalnih adresa bilo zanimljivo za daljnju analizu rada konkretnog korisnika u toj sjednici. U tom slučaju, potrebno je pratiti obavljanje svega što korisnici rade na razini pojedinih SQL naredbi, kako bi se moglo točno odrediti radi li se o zlouporabi ili ne.

Složeniji način određivanja obrazaca ponašanja, koji se ne zasniva samo na prijavama i odjavama korisnika, nego na ukupnoj djelatnosti korisnika, opisan je u [Lin1994]. Primjenom Weistrassova teorema na snimljeni trag zaključeno je kako trag za svakog korisnika mora sadržavati ponavljajuće uzorke. Otkrivanjem skrivenih neizrazitih pravila (eng. *fuzzy rules*) u tragu, te kombiniranjem s ponavljajućim uzorkom za svakog korisnika je moguće odrediti obrazac ponašanja. Svako znatnije odstupanje od obrasca ukazuje na potencijalnu zloporabu koju je potrebno temeljitije istražiti.

5.3. Podaci o klijentskim aplikacijama

Vezano uz snimanje korisničkih podataka prilikom prijave i odjave za rad sa sustavom, moguće je snimati i podatke o izvorištu rada s podacima. Iako su se ovdje radi o podacima koje bi trebalo snimati pri prijavi za rad, te je podatke korisno imati i uz trag o obavljenim operacijama nad podacima. Pri tom se prvenstveno misli na obavljanje operacija dohvata, te DML operacije izmjene podataka. Naime, korisno je znati je li se neko ažuriranje podataka dogodilo kroz npr. Microsoft Excel ili iz Java klijenta. Prema ovom podatku je moguće napraviti i analize uporabe baze podataka prema izvorišnim klijentima, bilo da se radi o analizama na razini pojedinog zapisa u relaciji ili na razini cijele baze podataka. Na osnovi ovih analiza može se lako utvrditi je li došlo do prekoračenja ovlasti. Na primjer, izvođenje operacija nad podacima iz produkcije baze podataka o ljudskim resursima u tvrtci, koje je poteklo iz nekog od razvojnih alata kojima se koriste programeri sustava, skoro sigurno indicira neovlaštene radnje.

Ovisno o informacijskom sustavu kojeg baza podataka poslužuje, korisnike se može razlikovati na različite načine. U slučaju klijent/server arhitekture, klijente se često razlikuje prema pristupnim IP adresama, te je to podatak koji se može snimati uz operacije nad podacima. Ovisno o arhitekturama računalnih mreža u kojima se klijenti nalaze, može se dogoditi da se više klijenata izvan svoje lokalne mreže prema van identificira istom IP adresom (npr, ako se koristi *network address translation* –

NAT). U tom slučaju je potrebno snimati i pripadno korisničko ime koje je iniciralo pojedinu operaciju nad podacima. Ako je, pak, arhitektura informacijskog sustava postavljena tako da se korisnici spajaju na bazu podataka kroz aplikacijski server, tada ni IP adresa, ni pristupno korisničko ime kojim se aplikacijski server identificira na bazu podataka nisu dovoljni za identifikaciju. Tada je potrebno uz operacije nad podacima snimati neki drugi identifikacijski podatak koji jedinstveno određuje pojedinog korisnika informacijskog sustava (osobu).

5.4. DDL naredbe

Praćenje izmjena sheme baze podataka, odnosno konkretno, naredbi za izmjenu podataka koje definira *Data Definition Language* (DDL) jezik, vrlo je važno kako sa sigurnosnog, tako i sa stajališta konzistentnosti i konfigurabilnosti. DDL naredbe su potencijalno najštetnije koje je moguće izvršiti nad bazom podataka, i svaki napadač ih može lako upotrijebiti kako bi kompromitirao sustav. DDL naredbe također mogu indicirati krađu informacija – npr. stvaranje dodatne tablice u koju se kopiraju podaci prije izvlačenja.

Mnogi naputci zakonskih normi i pravila se mogu protumačiti kao izričita potreba za snimanjem traga DDL naredbi. Ovdje se ne radi samo o sigurnosnom aspektu, nego i potrebi da se izbjegnu pogreške i brzo rješavaju problemi. Ukoliko postoji povijest stanja svih objekata u bazi podataka, tada bi trebalo biti relativno jednostavno vratiti neki kompromitirani objekt u starije stanje. Izmjena sheme baze podataka koju se slučajno napravi na produkcijskom umjesto razvojnom poslužitelju, pogreška je koja se može dogoditi svakom razvojnom inženjeru koji ima pristup obama poslužiteljima, a koju se može ispraviti u osjetno kraćem roku ukoliko postoji snimljeni trag tog događaja.

Snimanje traga DDL naredbi može se obavljati uz pomoć ugrađenih svojstava sustava za upravljanje bazama podataka, koristeći neki vanjski sustav za snimanje traga, ili usporedbom snimljenih shema baze podataka u različitim trenutcima. Svi veći sustavi za upravljanje bazama podataka omogućavaju snimanje tih naredbi na određeni način – konfiguracijske parametre sustava, dodatne programe koje treba pokrenuti, sistemske okidače i slično. Osoba zadužena za pregled i analizu snimljenog traga treba izraditi izvješća i izvući druge korisne informacije. Vanjski sustavi za snimanje traga pomažu u dijelu izrade izvještaja.

Periodičko snimanje shema baze podataka i naknadna usporedba je inferiornija drugim dvjema tehnikama, jer se ovdje ne radi o stalnom praćenju sustava. S druge strane, vrlo jednostavnim alatima je moguće doznati je li bilo nekih izmjena o kojima bi trebalo povesti računa. No ako jeste, ovom metodom nije moguće znati kada se to dogodilo, niti tko je izmjene učinio. Osim toga, napadač može između dva snimanja shema baze podataka napraviti izmjene, načiniti zlouporabu, te vratiti stanje sheme na staro, tako da se usporedbom dvije sheme neće vidjeti razlika. Ova je metoda dobra u slučaju upravljanja konfiguracijom sustava, no nije dostatna za sustave koji implementiraju regulatorna i sigurnosna rješenja.

5.5. Pogreške koje sustav za upravljanje bazama podataka prijavljuje korisnicima

Snimanje traga pogrešaka koje pri radu prijavljuje SUBP je, sa stajališta sigurnosti, jedna od vrlo važnih stvari koju je potrebno implementirati. Postoji velik broj situacija u kojima ovaj trag može biti koristan. Na primjer, ukoliko se radi o napadu injektiranjem SQL naredbi (eng. *SQL injection*, [Litchfield2005]), u velikom broju slučajeva napadač mora probati više puta prije nego uspije dobiti informacije koje su mu potrebne. Za svaki neuspješni pokušaj, SUBP će prijaviti jednu pogrešku o netočnom broju stupaca u upitu ili sličnu. Veliki broj takvih pogrešaka koje sustav prijavi može indicirati da se radi o ovakvom napadu. Drugi česti napad koji se lako može otkriti snimanjem pogrešaka jest pokušaj pogađanja korisničke lozinke. U ovom će slučaju SUBP prijavljivati pogreške o neuspješnom logiranju dok god napadač ne pogodi pravu lozinku. Na ovakve pogreške se može i učinkovito reagirati privremenim zaključavanjem korisničkog imena.

Pogreške mogu biti važne i sa stajališta kvalitete rada sustava. Naime, mogu biti uzrokovane pogreškama u pristupnim aplikacijama, te ih je u svakom slučaju potrebno prosljeđivati razvojnom timu. Na ovaj se način podiže kvaliteta aplikacija koje čine sustav, a u konačnici može utjecati i boljim performansama SUBP-a.

Detaljno snimanje pogrešaka omogućuju neki od komercijalnih sustava za upravljanje bazama podataka, dok se najbolji rezultati dobivaju vanjskim sustavima za snimanje traga. Oni su temeljeni na praćenju prometa između klijenta i baze podataka i obrnuto, tako da je lako moguće ustanoviti sve uzroke pogrešaka, što uz kvalitetne analize uvelike olakšava praćenje i rješavanje istih.

5.6. Izmjene programskog kôda pohranjenih procedura i okidača

Kako su pohranjene procedure i okidači objekti koji koriste potpune programske jezike, u njih je relativno lako sakriti zlonamjerni programski kôd. Zbog toga je važno pratiti promjene ovih objekata.

Snimanje traga izmjena pohranjenih procedura i okidača također se može obavljati na nekoliko načina. Najjednostavniji način jest periodičko (npr. dnevno) izvlačenje programskog kôda ovih objekata i usporedba s kôdom koji nastao u prethodnom čitanju. Ovu je metodu lako izvesti i ne zahtijeva korištenje složenih alata, ali pojavljuju se iste poteškoće koje su navedene u vezi korištenja metode pri provjeri sheme baze podataka: nedostatak informacija o pojedinim izmjenama i mogućnost napadača da vrati staro stanje objekta prije periodičkog čitanja. Druga mogućnost je snimanje unutar SUBP-a. Većina sustava za upravljanje bazama podataka podržava mogućnost snimanja traga prilikom stvaranja i izmjena pohranjenih procedura i okidača, najčešće u okviru praćenja DDL naredbi. Ipak, nije uvijek lako (ili je čak nemoguće) dobiti točan programski kôd koji je bio osnova pojedinog objekta.

Mogućnost snimanja traga izmjena pohranjenih procedura i okidača vanjskim sustavima za snimanje traga je i u ovom slučaju najbolja mogućnosti. Ovi sustavi uglavnom omogućavaju prikaz točnog programskog kôda za svaku verziju objekta,

lako mogu prikazati nastale razlike, te čak i upozoriti administratora sustava o izmjenama.

5.7. Izmjene podataka o korisnicima i njihovim dozvolama

Podaci o korisnicima i pripadajućim privilegijama nad bazama podataka su apsolutni prioritet prilikom snimanja traga izmjena. Postavljanje snimanja ovih podataka može se izjednačiti s postavljanjem sigurnosnih kamera na ulaz zgrade ili na mjesto gdje se izdaju pristupne kartice novim zaposlenicima.

Svaki napredni sustav za upravljanje bazama podataka sastoji se od složene sheme korisnika, pripadnih uloga, te privilegija koje korisnici i uloge imaju nad objektima u bazi podataka. Napadači će često pokušati podići svoje dozvole, ili će pokušati stvoriti novog korisnika koji ima veće dozvole nego što ih oni trenutno imaju. Zbog toga je važno ove podatke ne samo snimati, nego i aktivno pratiti, te imati mogućnost aktivnog obavještavanja o potencijalnim napadima. Na primjer, ako se stvara novi korisnik baze podataka tijekom noći, to je siguran znak da se radi o napadu.

U sklopu snimanja podataka o korisnicima i dozvolama, moguće je pratiti:

- dodavanje i uklanjanje korisnika i uloga,
- izmjene pridijeljenih uloga korisnicima,
- izmjene privilegija, bilo da se radi o korisnicima ili ulogama,
- izmjene korisničkih lozinki,
- izmjene sigurnosnih postavki na razini poslužitelja, baze podataka, naredbe ili pojedinog objekta.

Obzirom da izmjene ovih podataka mogu biti od visoke važnosti za funkcioniranje baze podataka, u ovom slučaju bi trebalo izbjegavati snimanje traga zasnovano na periodičkom izvlačenju iz baze podataka i usporedbi. Prigodnije je koristiti ugrađene mogućnosti snimanja traga SUBP-a, ili vanjski sustav za snimanje traga. Pri izgradnji svog nadzornog podsustava za reagiranje na potencijalne napade, također treba uzeti u obzir obrasce rada informacijskog sustava.

5.8. Korisnički sinonimi i replikacija baza podataka

Korisnički sinonim jest privatni objekt u bazi podataka koji se može shvatiti kao veza na stvarni objekt. Pod pojmom „replikacija baza podataka“ ovdje se smatra postojanje barem jedne, vezane baze podataka koja je na neki od načina za replikaciju podataka uparena s promatranom bazom podataka, te postoji barem jedna tablica čiji se podaci repliciraju u tu uparenu bazu podataka.

Iako naizgled različite, ove dvije kategorije imaju zajedničko obilježje – radi se o preusmjeravanju zanimljivih podataka prema čitaču koji ih možda ne bi smio vidjeti. Stvaranje novog privatnog sinonima ili preusmjeravanje postojećeg na zaštićenu tablicu, postavljanje replikacije te tablice, načini su za dohvat nedostupnih podataka. Dok su druge kategorije uglavnom usmjerene na trenutnu aktivnost napadača, ovdje se radi o dugotrajnijem pristupu zaštićenim podacima. Nastale izmjene ostaju do

daljnjega. Na osnovu ovoga se može zaključiti kako je dovoljan način praćenja izmjena ovih podataka upravo periodičko (dnevno) čitanje iz baze podataka i usporedba. Također, većina sustava za upravljanje bazama podataka ne podržava snimanje traga o replikacijskim pravilima.

5.9. SQL naredbe za izmjenu podataka

Snimanje traga naredbi za unos, izmjene i brisanje podataka je čest zahtjev, kako u raznim regulatornim aktima, tako i na razini vodstva projekta. Pri tom je također i česta potreba da se za svaku od ovih DML operacija čuva i stara i nova vrijednost (ukoliko se radi o izmjenama postojećih zapisa).

Osnovna briga prilikom snimanja traga ovih naredbi jest veličina samog traga koji se snima. U velikim transakcijskim sustavima, snimljeni trag može samo u jednoj godini višestruko nadmašiti veličinu same baze podataka, te je potrebno razumno koristiti ove mogućnosti. Uglavnom se snimanje traga odnosi samo na pojedine tablice, na pojedine atribute tablica, na rad određenih korisnika i slično.

Trag naredbi za izmjene podataka se može snimati na tri načina: internim sustavom SUBP-a za snimanje traga, vanjskim sustavom, te korištenjem okidača. Svi sustavi za upravljanje bazama podataka omogućavaju snimanje traga ovih DML naredbi, a razlikuju se u razini prilagodljivosti. Vanjski sustavi za snimanje traga podržavaju snimanje na osnovu raznih kriterija, korištenjem različitih filtera prema korisnicima, objektima, pristupnim aplikacijama i slično, a također su praktični prilikom prikaza snimljenog traga. Treća mogućnost, korištenje prilagođenih okidača, nudi najprilagodljiviji način snimanja traga, no uz najveći trud pri implementaciji. Zbog toga se preporuča za projekte manjeg opsega.

5.10. SELECT naredbe

Iako snimanje traga obavljanja naredbi dohvata podataka u prošlosti nije bilo tako zanimljivo, u zadnje vrijeme se sve pozornosti obraća i na taj segment rada s bazama podataka. Osnovni razlog tome je sve veća svijest o zaštiti privatnosti, pa tako mnoge tvrtke moraju imati podatke o tome tko je, kada i kako vidio određene podatke.

Kao i kod DML naredbi, i u ovom slučaju snimanje traga može potpuno zagušiti sustav. Zapravo, još i više, jer većina informacijskih sustava obavlja znatno veći broj čitanja nego ažuriranja podataka. Zbog toga je jako bitno odrediti za koje podatke će se snimati trag čitanja, a koje će korisnici moći čitati bez znanja promatrača. Ovo implicira da je u svakoj bazi podataka potrebno odrediti skupove zaštićenih podataka.

Skup zaštićenih podataka čine podaci koji zajedno otkivaju neke povjerljive informacije. Na primjer, broj zdravstvene iskaznice nije tajna informacija, kao ni nečije ime. Podaci o bolesti vezani uz broj zdravstvene iskaznice nisu tajna informacija. No, podaci o nečijoj bolesti jesu, a njih čine povezani podaci o bolesti i osobi. Ti podaci, povezani, u bazi podataka privatnog liječnika trebaju biti skup zaštićenih podataka.

Nakon što su identificirani svi skupovi zaštićenih podataka, može se realizirati snimanje traga pristupa tim podacima. Obzirom na prirodu dohvata podataka, ovaj je

trag prilično zahtjevno snimati. Najbolje rezultate postižu vanjski alati za snimanje traga, prvenstveno zbog veće prilagodljivosti od unutarnjih mehanizama SUBP-a za snimanje traga. Unutarnji mehanizmi često stvaraju previše podataka iz kojih je teže pronaći zanimljive informacije.

Okidači, koji se u posljednje vrijeme u mnogim sustavima za upravljanje bazama podataka mogu povezati sa SELECT naredbama, nisu vrlo praktični, jer također generiraju mnoštvo rezultata. No, njih se može programski prilagoditi. Tako, u slučaju da nije nužno imati ispisanu baš svaku n-torku koju je korisnik pročitao, trag se može temeljiti na samim SQL naredbama koje su korisnici izvršili. Ovakav trag je više informativnog karaktera, i ne bi poslužio ako je nužno zadovoljiti neku zakonsku regulativu, ali može biti vrlo koristan ako se želi znati je li itko čitao zaštićene podatke, a ako jeste, onda se na istom mjestu može znati tko, a uz to i kada je to napravljeno i na koji način.

5.11. Snimanje izmjena definicija snimanja traga

Potrebno je pažljivo pratiti i snimati trag svih izmjena definicije snimanja traga ili izmjena već snimljenih tragova. Ovo se može poistovjetiti s postojanjem sigurnosnih kamera na ulaznim vratima. Ukoliko provalnik može postaviti statičnu sliku ispred kamere ili naknadno promijeniti snimljenu vrpcu, onda sigurnosne kamere uopće ne vrijede. Na isti način napadač na informacijski sustav može promijeniti ono što se snima ili promijeniti snimljene tragove. Da se ovako ne bi obezvrijedilo snimanje traga u bazama podataka i postojanje snimljenog traga, potrebno je snimati i ove aktivnosti.

Snimanje traga o izmjenama snimljenog traga zapravo uvodi novi trag. Postojanje ovoga traga nije smisleno ukoliko na sustavu nije implementirana raspodjela dužnosti.

6. Snimanje traga o izmjenama podataka prilagodbom okidača

Mnoge kategorije snimanja traga odnose se na kontrolu sigurnosnih mehanizama koji su na snazi unutar baze podataka ili sustava za upravljanje bazama podataka. Snimanje traga o unosu, izmjeni i brisanju podataka, odnosno DML naredbi, osim sigurnosnog aspekta (mora se znati tko je pristupao i mijenjao zaštićene podatke) ima i forenzičnu ulogu. Detaljno praćenje povijesti podataka u promatranim relacijama baze podataka pokazuje se korisnim prilikom istraživanja povijesti aktivnosti nad određenim podacima, kao dodatna kontrola dozvola korisnika, a može poslužiti i kao osnova za kontrolu ispravnog funkcioniranja pristupnih aplikacija.

Kao što je prikazano u prethodnom poglavlju, snimanje traga DML naredbi moguće je realizirati na nekoliko načina. Prilagodba okidača unutar baza podataka je najprilagodljiviji način, no ujedno i najsloženiji za implementaciju. Uz to, ovaj način praćenja traga ima i drugi važan nedostatak – okidači, od kojih zavisi cijeli proces, mogu biti onemogućeni od strane administratora baze podataka ili administratora sustava za upravljanje bazama podataka. Kako bi se pokrila sigurnosna komponenta ovog načina praćenja traga, potrebno ga je koristiti u kombinaciji s nekim drugim načinom snimanja traga kojim će se moći potvrditi postojanost okidača, za što može biti dobar interni sustav SUBP-a za snimanje traga. Praćenjem stanja okidača koji su nadležni za snimanje traga, te adekvatnim alarmiranjem može se osigurati vjerodostojnost traga DML naredbi snimljenih na ovaj način.

U nastavku ovog poglavlja bit će opisane ideje i algoritmi korišteni za implementaciju automatizirane izvedbe snimanja traga prilagodbom okidača. Iduće poglavlje prikazuje nekoliko primjena uporabe snimljenog traga u cilju povećanja sigurnosti sustava i forenzičnih nalaza baze podataka. Sintaksa prikazanih naredbi SQL jezika odgovara sintaksi tog jezika implementiranoj u sustavu za upravljanje bazama podataka IBM Informix, obzirom da je to SUBP na kojem je rješenje i implementirano. Uz male preinake rješenje je portabilno i na ostale sustave za upravljanje bazama podataka.

6.1. Motivacija

Iako relativno složen način za implementaciju snimanja traga DML naredbi, prilagodba okidača u cilju snimanja traga nudi nekoliko prednosti nad drugim metodama. Korištenje vanjskih sustava za snimanje traga osim financijskog, zahtijeva i dodatan angažman za administriranje i analizu snimljenog traga. Snimanje traga stanja podataka u relacije namijenjene čuvanju povijesti omogućava *ad hoc* uporabu SQL naredbi nad snimljenim tragom, što ne iziskuje stjecanje dodatnih znanja. Unutarnji sustav SUBP-a za snimanje traga DML naredbi pak često nije dovoljno prilagodljiv da bi ga se moglo koristiti u svrhu praćenja povijesti određenog podatka ili korisničke djelatnosti, što je u ovom slučaju od velike važnosti. Izradom automatiziranog načina prilagodbe baze podataka u svrhu snimanja traga DML naredbi postiže se mogućnost snimanja traga u bilo kojoj bazi podataka uz minimalnu količinu administratorskih postupaka.

6.2. Opis izvedbe

Neka baza podataka **bp** sadrži određenu količinu različitih objekata, među kojima je n relacija. Nad relacijama je moguće obaviti tri operacije za manipulaciju podacima – unos nove n-torke u relaciju (*insert*), ažuriranje ili izmjena (*update*) postojeće n-torke u relaciji te brisanje (*delete*) n-torke iz relacije. Promatrana operacija izmjene relacije (OI) je jedna tri navedene:

$$OI \in \{\text{insert, update, delete}\}.$$

Svaka relacija može imati barem jedan okidač za svaku od navedenih operacija. U tom slučaju, sustav za snimanje traga se implementira na sljedeći način:

- Rječnik baze podataka **bp** se proširuje novom relacijom r_{AUDIT} u kojoj se za svaku relaciju iz **bp** evidentiraju podaci vezani za snimanje traga za tu relaciju.
- Za svaku relaciju r za koju je u r_{AUDIT} evidentirano snimanje traga, stvara se analogna povijesna relacija, _r, koja osim svih atributa koje sadrži relacija r, sadrži i dodatne atribute koji opisuju konkretni događaj utjecaja na n-torke relacije r.
- Za svaku relaciju r za koju je u r_{AUDIT} evidentirano snimanje traga, za svaku od operacija izmjene podataka OI se modificiraju postojeći okidači nad tom relacijom kako bi uključili naredbe za snimanje traga, ili se stvaraju ukoliko ne postoje.
- Stvara se povijesna baza podataka **pbp**, koja sadrži kopije svih povijesnih relacija. Ova baza podataka služi kao spremište cijele povijesti snimljenih podataka iz originalne baze podataka **bp**, dok povijesne relacije u **bp** sadrže samo mali dio povijesti (npr. tekući dan ili tjedan). Na ovaj način je moguće održati količinu aktualnih korisnih podataka u radnoj bazi podataka blizu optimalne, a imati i cijelu povijest koja može biti na drugom računalnom sustavu i koja neće imati negativan učinak na performanse sustava za upravljanje bazama podataka.
- Pokreće se poseban proces, *backupHistory*, koji arhivira n-torke iz povijesnih relacija u radnoj bazi podatka u povijesnu bazu podataka.
- Prilagođavaju se dozvole pristupa na razini SUBP-a za sve sudionike u procesima.

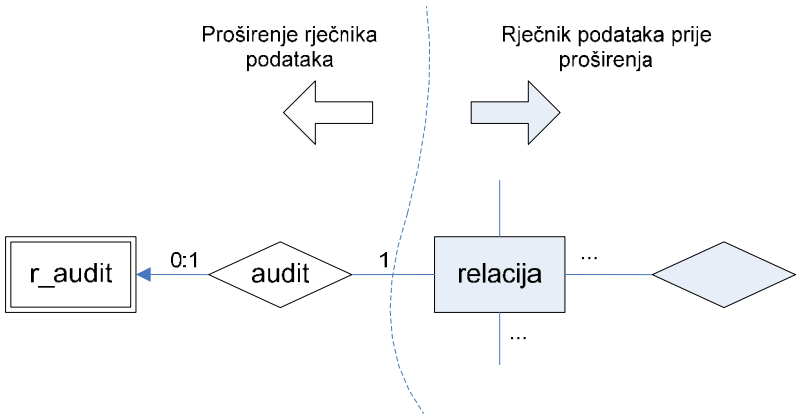
Ova je ideja detaljno razrađena u nastavku poglavlja.

6.2.1. Proširenje rječnika baze podataka

Mnogi informacijski sustavi se izgrađuju uz pomoć raznih programskih alata koji podržavaju velike spektre funkcionalnosti – od osnova rada s bazama podataka za osiguravanje perzistentnosti podataka do jednostavne izrade unificiranog korisničkog sučelja, npr. Hibernate, Spring [Hibernate2008, Spring2008]. Neke funkcionalnosti se nadovezuju na samu shemu pripadne baze podataka. Za kompletnije iskorištavanje mogućnosti i dodavanje novih mogućnosti, pokazalo se smislenim u bazu podataka

dodati i dodatne podatke o shemi baze podataka. Ovo proširenje se naziva prošireni rječnik podataka [Baranović2001]. Uporaba proširenog rječnika podataka u postupcima upravljanja repliciranim bazama podataka prikazana je u [Zakošek2004], a u [Mačkala2004] je opisana izrada sučelja prema bazi podataka s ovakvim proširenim rječnikom podataka. Podaci iz proširenog rječnika također se koriste i pri definiranju i prilagodbi ekranskih formi [Orel2001] tijekom izrade grafičkog korisničkog sučelja prema radnoj bazi podataka.

Proširenje rječnika podataka vezano uz snimanje traga odnosi se samo na jednu novu relacijsku shemu: R_{AUDIT}. N-torke relacije $r_{AUDIT}(R_{AUDIT})$ opisuju željeno stanje snimanja traga za ostale relacije u bazi podataka. Na slici 6.1. prikazan je dodatni segment ER modela rječnika podataka. Ovaj se segment veže na postojeći segment rječnika podataka preko entiteta *relacija*, koji opisuje relacije u bazi podataka i koji je dio rječnika podataka svakog SUBP-a.



Slika 6.1. Proširenje rječnika podataka podacima o snimanju traga

Relacija r_{AUDIT} definira se na shemi

R_{AUDIT} = RelSh, Snimanje, Velicina

$K_{R\text{-}AUDIT}$ = RelSh

pri čemu su atributi sheme:

- RelSh – naziv relacijske sheme relacije za koju se definira snimanje traga
- Snimanje – zastavica koja određuje da li je snimanje trenutno pokrenuto ili ne
- Velicina – inicijalna veličina prostora za pohranu odgovarajuće relacije s povijesnim podacima.

Sadržaj relacije r_{AUDIT} je početna točka implementacije snimanja traga. Taj sadržaj određuje za koje će se sve relacije snimati trag, tako da ovu relaciju treba pažljivo štititi od neželjenih promjena. Prema principu raspodjele dužnosti, sadržaj ove tablice bi trebala održavati samo uloga koja definira što se i kako snima. Prilikom svakog održavanja sustava, za svaku novu relaciju u bazi podataka potrebno je odrediti stanje snimanja traga.

Primjer 6.1: Baza podataka **bp** sadrži više relacija, među kojima su i relacije s osobnim podacima zaposlenika tvrtke, s relacijskim shemama:

OSOBA = oznOsoba, ime, prezime, JMBG

K_{OSOBA} = oznOsoba

POSAO = sifPosao, nazPosao, koefPlaca

K_{POSAO} = sifPosao

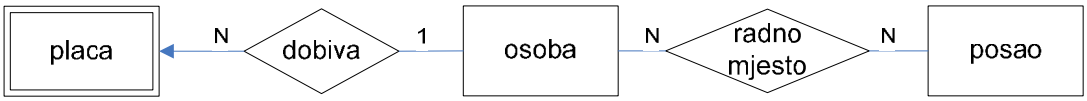
RADNO_MJESTO = oznOsoba, sifPosao, mjesecOd, mjesecDo

K_{RADNO_MJESTO} = oznOsoba, sifPosao, mjesecOd

PLACA = oznOsoba, mjesec, iznosStimulacija, iznosHonorar

K_{PLACA} = oznOsoba, mjesec

ER model ovog dijela baze podataka prikazan je na slici 6.2.:



Slika 6.2. Demonstracijski ER model

U prikazanom modelu potrebno je pratiti izmjene sadržaja triju relacija: osoba, posao i placa. Stvaranjem nove relacije u bazi podataka, r_{AUDIT}, stvaraju se preduvjeti za definiranje snimanja traga nad ovim relacijama. Stoga će sadržaj relacije r_{AUDIT} biti:

r_{AUDIT} (RelSh, Snimanje, Velicina)

OSOBA	Da	10000
POSAO	Da	10000
RADNO_MJESTO	Ne	0
PLACA	Da	40000

Veličina povijesne relacije određuje se prema očekivanoj količini podataka u originalnoj relaciji, pomnoženoj s koeficijentom koji predstavlja očekivanu učestalost izmjena sadržaja originalne relacije, koji je za svaku relaciju drugačiji, a može se odrediti na osnovu iskustva u projektiranju informacijskih sustava i poznavanjem posebnosti konkretnog sustava.

Prikazana situacija bit će korištena i u sljedećim primjerima ovog poglavlja. ❐

U relaciji r_{AUDIT} se zapisuju imena relacijskih shema onih relacija za koje se određuje želi se snimati trag ili ne. Obzirom da se u algoritmima i implementaciji zapravo radi s konkretnim relacijama, najprije je potrebno definirati pristupnu metodu za

relacijsku shemu preko njezina imena. Ova će metoda, opisana algoritmom *dohvatiRelacijskuShemu*, biti korištena u drugim algoritmima koji opisuju postupak snimanja traga. Algoritam se izvodi na radnoj bazi podataka **bp** sa shemom SBP, obzirom da ona sadrži relaciju r_{AUDIT} i relacije za snimanje traga.

Algoritam 6.1: *dohvatiRelacijskuShemu*

funkcija **dohvatiRelacijskuShemu** (ulazni argument **ImeRelSh**: niz znakova /*ime relacijske sheme */): relacijska shema

varijabla

R: relacijska shema

i: niz znakova

za svaku **R** ∈ **SBP**

i ← ime relacijske sheme **R**

ako je **i** = **ImeRelSh** tada

vrati vrijednost **R** □

6.2.2. Stvaranje povijesnih relacija

Na temelju sadržaja relacije r_{AUDIT} shema baze podataka se proširuje dodatnim relacijskim shemama. Točnije, za svaku relacijsku shemu $R_i = A_1, A_2, \ldots, A_n$ za koju je određeno snimanje traga stvara se nova relacijska shema $\_R_i = A_1, A_2, \ldots, A_n, X$ koja sadrži identične atribute kao i shema R_i, te dodatni skup atributa X koji opisuju operaciju koja se dogodila nad pripadajućom n-torkom. Atributi koji opisuju operaciju su:

- OpUser – korisnik koji je obavio operaciju,
- OpTimestamp – vremenska oznaka operacije,
- TxTimestamp – vremenska oznaka transakcije unutar koje je operacija obavljena,
- Operation – oznaka operacije,
- SessionID – oznaka korisničke sjednice u kojoj je korisnik obavio operaciju.

Dakle,

X = {OpUser, OpTimestamp, TxTimestamp, Operation, SessionID}.

Pri tom su domene ovih atributa kako slijedi:

DOM(OpUser) = sva korisnička imena pridijeljena bazi podataka,

DOM(OpTimestamp) = sve vremenske oznake,

DOM(TxTimestamp) = sve vremenske oznake,

DOM(Operation) = {'I', 'U', 'D'},

DOM(SessionID) = N, skup prirodnih brojeva.

Svaka operacija unosa, izmjene ili brisanja (OI) n-torke t_i relacije r_j, bit će zapisana kao jedna n-torka u relaciji $\_r_i(\_R_i)$. Ukoliko se radi o operaciji unosa ili izmjene, atributi A_1, A_2, …, A_n n-torke relacije $\_r_i$ će imati nove vrijednosti istih atributa promatrane n-torke relacije r_i, a ukoliko se radi o operaciji brisanja, ti će atributi imati stare vrijednosti atributa obrisane n-torke relacije r_i. Ostali atributi relacije $\_r_i$ imati će vrijednosti koje opisuju relaciju.

Stvaranje povijesnih relacija definira algoritam *SPR*.

Algoritam 6.2: *SPR*

procedura **spr** ()
    konstanta
        X: {OpUser, OpTimestamp, TxTimestamp, Operation, SessionID}
    varijabla
        R, **_R**: relacijska shema
        t: n-torka
        v: inicijalna veličina prostora za pohranu relacije
    za svaku n-torku **t** iz r_{AUDIT}
        ako je **t**[Snimanje] = 'Da' tada
            R ← dohvatiRelacijskuShemu(**t**[RelSh])
            _R ← **R** ∪ **X**
            v ← **t**[Velicina]
            stvori novu relaciju _r**(_R)** s veličinom prostora za pohranu **v** □

Primjer 6.2: Na temelju sadržaja relacije r_{AUDIT} iz primjera 6.1, procedura iz algoritma *SPR* stvorit će tri nove relacije u bazi podataka: _osoba, _posao i _placa. Shema ovih relacija je sljedeća:

_OSOBA = oznOsoba, ime, prezime, JMBG, OpUser, OpTimestamp, TxTimestamp, Operation, SessionID

_POSAO = sifPosao, nazPosao, koefPlaca, OpUser, OpTimestamp, TxTimestamp, Operation, SessionID

_PLACA = oznOsoba, mjesec, iznosStimulacija, iznosHonorar, OpUser, OpTimestamp, TxTimestamp, Operation, SessionID □

6.2.3. Nadopuna okidača naredbama za snimanje traga

Sljedeći korak implementacijskog procesa se odnosi na osiguravanje da za one relacije za koje se želi snimati trag postoje okidači za koji osiguravaju samo snimanje traga. Neki sustavi za upravljanje bazama podataka podržavaju postojanje više okidača za jednu relaciju koji se okidaju za istu operaciju, dok drugi dopuštaju postojanje samo jednog okidača za svaku operaciju nad relacijom. Algoritmi i

primjeri u daljnjem tekstu će biti ograničeni na složeniji slučaj – postojanje samo jednog okidača po operaciji nad relacijom.

Postupak nadopune okidača definira se algoritmom *PNO*. Iz rječnika baze podataka pronalazi tekstove postojećih okidača za relacije obuhvaćene snimanjem traga. Ukoliko okidač za neku operaciju nad relacijom postoji, algoritam provjerava sadrži li taj naredbu za snimanje traga. Ako takva naredba nije dio okidača, algoritam će rekreirati taj okidač s novim tekstom u kojem će biti uključena i naredba za snimanje traga. Ako pak okidač za operaciju nad relacijom ne postoji, bit će stvoren novi, koji će samo sadržavati naredbu za snimanje traga.

Naredba za snimanje traga treba biti takva da sačuva podatke o n-torki za koju se poziva. Ukoliko se radi o operaciji unosa ili izmjene, tada će se sačuvati podaci o novoj, odnosno izmijenjenoj n-torci, a u slučaju operacije brisanja će biti sačuvani podaci o izbrisanoj n-torci. Opseg naredbe za snimanje traga u okidaču treba biti takav da se izvršava za svaku unesenu, izmijenjenu ili obrisanu n-torku.

Algoritam 6.3: *PNO*

```
procedura pno ()
    varijabla
       t: n-torka
       trig: okidač
       R: relacijska shema
    za svaku n-torku t iz r_AUDIT
        ako je t[Snimanje] = 'Da' tada
            za svaku OI ∈ {insert, update, delete}
                R ← dohvatiRelacijskuShemu(t[RelSh])
                ako postoji okidač za OI nad r(R) tada
                    trig ← učitaj okidač iz rječnika podataka
                    parsiraj okidač trig
                    ako trig ne sadrži naredbu za snimanje traga tada
                        dodaj naredbu za snimanje traga u okidač trig
                        ponovo stvori okidač trig
                inače
                    trig ← novi okidač za OI nad r(R) s naredbom za snimanje traga
                    stvori okidač trig □
```

Primjer 6.3: Neka nad relacijama osoba i posao iz primjera 6.1 i 6.2 već postoje implementirani okidači kojima se osiguravaju određena pravila integriteta, i to samo za operacije unosa i brisanja, dok ne postoje okidači za operaciju izmjene podataka. Okidač koji se aktivira pri operaciji unosa podataka u relaciju osoba nazvan je *osobaIns*, a onaj za operaciju brisanja *osobaDel*. Analogno tome, okidači nad relacijom posao nazvani su *posaoIns* i *posaoDel*. Nad relacijom placa ne postoji

nijedan okidač. Primjena algoritma *PNO* nad ovom bazom podataka rezultirat će izmjenama nad postojeća četiri okidača, te stvaranjem pet novih okidača. Okidačima *osobaIns*, *osobaDel*, *posaoIns* i *posaoDel* dodane su naredbe za snimanje traga, a stvoreni su novi okidači koji sadrže jedino ove naredbe: *osobaUpd*, *posaoUpd*, *placaIns*, *placaUpd* i *placaDel*. U tablici 6.1 su prikazane SQL naredbe za stvaranje svih okidača nad relacijom osoba prije i nakon izvršenja procedure **pno**.

Okidači prije izvršenja procedure **pno**

```
CREATE TRIGGER osobaIns INSERT ON osoba
  REFERENCING NEW AS osobaNew
  FOR EACH ROW
    (EXECUTE PROCEDURE provjeriJMBG(osobaNew.JMBG));


CREATE TRIGGER osobaDel DELETE ON osoba
  REFERENCING OLD AS osobaOld
  FOR EACH ROW
    (EXECUTE PROCEDURE provjeriBrisanje(osobaOld.oznOsoba));
```

Okidači nakon izvršenja procedure **pno**

```
CREATE TRIGGER osobaIns INSERT ON osoba
  REFERENCING NEW AS osobaNew
  FOR EACH ROW
    (EXECUTE PROCEDURE provjeriJMBG(osobaNew.JMBG),
     INSERT INTO _osoba VALUES( osobaNew.oznOsoba,
                                osobaNew.ime,
                                osobaNew.prezime,
                                osobaNew.JMBG,
                                getUser(),
                                getOpTimestamp(),
                                getTxTimestamp(),
                                'I',
                                getSessionID()) ));


CREATE TRIGGER osobaUpd UPDATE ON osoba
  REFERENCING NEW AS newrow
  FOR EACH ROW
    (INSERT INTO _osoba VALUES( osobaNew.oznOsoba,
                                osobaNew.ime,
                                osobaNew.prezime,
                                osobaNew.JMBG,
                                getUser(),
                                getOpTimestamp(),
                                getTxTimestamp(),
                                'U',
                                getSessionID()) ));
```

```
CREATE TRIGGER osobaDel DELETE ON osoba
  REFERENCING OLD AS osobaOld
  FOR EACH ROW
  (EXECUTE PROCEDURE provjeriBrisanje(osobaOld.oznOsoba),
   INSERT INTO _osoba VALUES( osobaOld.oznOsoba,
                              osobaOld.ime,
                              osobaOld.prezime,
                              osobaOld.JMBG,
                              getUser(),
                              getOpTimestamp(),
                              getTxTimestamp(),
                              'D',
                              getSessionID()) ));
```

Tablica 6.1. SQL naredbe za stvaranje okidača prije i nakon izvršavanja algoritma za prilagodbu okidača

U ovom primjeru okidači nad relacijom osoba koriste pohranjene procedure provjeriJMBG i provjeriBrisanje u kojima su definirana pretpostavljena poslovna pravila. Svi izmijenjeni okidači koriste dodatne pohranjene procedure za dohvat podataka koji opisuju konkretnu operaciju koja je okinula okidač:

- getUser – vraća korisničko ime korisnika koji je obavio operaciju
- getOpTimeStamp – vraća vremensku oznaku operacije
- getTxTimestamp – vraća vremensku oznaku transkacije unutar koje je operacija obavljena
- getSessionID – vraća oznaku korisničke sjednice u kojoj je korisnik obavio operaciju.

Okidači nad relacijama posao i placa su izmijenjeni ili stvoreni analogno prikazanima. □

6.2.4. Stvaranje povijesne baze podataka

Kako je naglašeno u uvodu ovog poglavlja, povijesna baza podataka sadrži identične povijesne relacije kao i radna baza podataka, pri čemu su povijesne relacije u povijesnoj bazi podataka **pbp** označene s _pr(_PR). Relacijske sheme sadrže iste atribute:

$$\_R_i = X_i \Rightarrow \_PR_i = X_i \qquad \forall\ \_R_i \in \mathbf{BP},\ \_PR_i \in \mathbf{PBP}$$

Sadržaji povijesne relacije $\_r_i$ u radnoj bazi podataka **bp** i povijesne relacije $\_pr_i(\_PR_i)$ u povijesnoj bazi podataka **pbp** su komplementarni. Unija te dvije relacije predstavlja ukupnu snimljenu povijest relacije R:

$$\text{povijest relacije r} = \_r \cup \_pr$$

Pri tom relacija _pr sadrži povijesne podatke od prvog relevantnog snimljenog traga do nekog relativno bliskog trenutka u prošlosti (npr. u posljednja 24 sata), a relacija _r sadrži povijesne podatke koji su mlađi od tog trenutka. U određenim vremenskim

trenucima poseban proces, *backupHistory*, premješta povijesne podatke iz relacije _r u relaciju _pr.

Zbog praktičnih razloga, povijesna baza podataka se može naći na izdvojenom sustavu za upravljanje bazama podataka, koji može biti i na izdvojenom poslužitelju. Na taj način veličina relacija unutar te baze ne opterećuje radni poslužitelj baza podataka. Povijesna baza podataka također ne mora biti uvijek dostupna. Dovoljno je da bude dostupna samo u vremenu obavljanja procesa *backupHistory*, te kad je u tijeku analiziranje povijesnih podataka.

Prema naputku o raspodjeli dužnosti pristup povijesnoj bazi podataka **pbp** bi trebala imati samo osoba zadužena za pregled i analizu snimljenog traga. Pristup relacijama $\_pr_i$ koji omogućuje samo unos novih zapisa mora imati proces *backupHistory*.

Stvaranje povijesne baze podataka definirano je algoritmom *SPBP* koji se izvodi u radnoj bazi podataka **bp** sa shemom **BP**.

Algoritam 6.4: *SPBP*

procedura **spbp** (ulazni argument **PPBP**: poslužitelj baze podataka) /* povijesne */

varijabla

R, **_PR**: relacijska shema

SBP: shema baze podataka

pbp: baza podataka

SBP ← ∅

za svaku **R** ∈ **BP**

_PR ← **R**

SBP ← **SBP** ∪ **_PR**

stvori bazu podataka **pbp**(**SBP**) na poslužitelju **PPBP** ❐

Primjer 6.4: Izvođenjem procedure **spbp** u bazi podataka prikazanoj u prethodnim primjerima ovog poglavlja, stvara se nova baza podataka sa sljedećom shemom:

PBP = { _posoba, _pposao, _pplaca }

dok su relacijske sheme ovih relacija identične kao onih u radnoj bazi podataka:

_pOSOBA = oznOsoba, ime, prezime, JMBG, user, opTimestamp, txTimestamp, operation, sessionID

_pPOSAO = sifPosao, nazPosao, koefPlaca, user, opTimestamp, txTimestamp, operation, sessionID

_pPLACA = oznOsoba, mjesec, iznosStimulacija, iznosHonorar, user, opTimestamp, txTimestamp, operation, sessionID ❐

6.2.5. Proces backupHistory

Proces *backupHistory* je proces čiji je jedini zadatak vrlo jednostavan – premještanje n-torki iz povijesnih relacija u radnoj bazi podataka u analogne povijesne relacije u povijesnoj bazi podataka. U ravnomjernim vremenskim periodima ovaj proces se spaja na obje baze podataka, pročita n-torke iz relacija u radnoj bazi podataka, upiše ih u relaciju u povijesnoj bazi, te obriše iz radne baze podataka. Ostatak vremena proces nije aktivan. Zbog uštede vremena, proces odjednom prebacuje blokove n-torki umjesto jednu po jednu. Rad procesa opisuje algoritam *BH*.

Algoritam 6.5: *BH*

```
procedura backupHistory (ulazni argument bp: baza podataka, /* radna */
                         ulazni argument pbp: baza podataka, /* povijesna */
                         ulazni argument period: vremenski period obavljanja)
    varijabla
        t: n-torka
        ts: skup n-torki
    ponavljaj
        za svaku _ri ∈ bp
            ts ← ∅
            za svaku t ∈ _ri
                ts ← ts ∪ t
            upisi n-torke ts u relaciju _pri
            obrisi n-torke ts iz relacije _ri
        cekaj period vremena
    zauvijek □
```

Primjer 6.5: Neka je opisani način snimanja traga nad relacijama osoba, posao i placa funkcionalan duži period vremena, od početnog trenutka t_0. Neposredno nakon završetka obavljanja procesa *backupHistory*, u trenutku t_x, dotadašnji sadržaj relacija _osoba, _posao i _placa je prebačen u relacije _pposoba, _pposao i _pplaca, i te su relacije prazne. Nakon određenog vremena Δt, ove se relacije pune povijesnim podacima koji upravo nastaju tijekom rada sustava. Tako će n-torke u relaciji _posoba biti one starije od t_x, a n-torke u relaciji _osoba one koje su nastale od trenutka t_x do vremena $t_x+\Delta t$. Cjelokupna povijest relacije osoba, od trenutka t_0 do vremena $t_x+\Delta t$, može se u svakom trenutku dobiti kroz uniju relacija _osoba i _posoba. Isto vrijedi i za sve druge relacije čija se povijest snima. □

6.2.6. Ubrzavanje čitanja snimljenog traga

Dok je snimljeni trag u radnoj bazi podataka smješten u relacijama s relativno malim brojem n-torki, trag u povijesnoj bazi će biti u relacijama s potencijalno vrlo velikim brojem n-torki, ovisno o količini izmjena nad pojedinom relacijom. Iako čitanje iz

povijesnih relacija u povijesnoj bazi podataka nije česta operacija, kad se izvodi bit će vrlo sporo ukoliko se te relacije budu pretraživale sekvencijalnim čitanjem svih n-torki. Da bi se to izbjeglo, potrebno je napraviti indekse nad tim relacijama. Uobičajene cijene postojanja indeksa nad relacijama – potrošnja prostora za pohranu indeksnih informacija, koji raste s porastom količine zapisa u relaciji, te sporije operacije pisanja zbog ažuriranja indeksnih grana – su u ovim slučajevima niske. Naime, radi se o relacijama koje već same po sebi zauzimaju dosta prostora, i smještene su u bazi podataka koja je za to namijenjena, te zauzeće dodatnog prostora za indekse ne čini znatno povećanje. Jedina operacija pisanja koja se izvodi nad ovim relacijama jest unos novih zapisa koje obavlja proces *backupHistory*. Kako je konfiguracija rada procesa prilagodljiva, tako je moguće prebacivanje podataka obavljati jednom svake noći, te se malo dulje trajanje izvođenja procesa neće osjetiti u radu korisnika, ili je moguće izvršavati proces u kraćim vremenskim razmacima, te na taj način smanjiti trajanje izvođenja procesa.

Indeksi se mogu stvoriti nad jednim ili više atributa relacije. Oni koje će se stvoriti automatizmom nad povijesnom relacijom _pr stvaraju se nad onim atributima te relacije koji čine primarni ključ originalne relacije r. Pretpostavka je da će se potrebiti dohvati uglavnom realizirati nad tim atributima. Kako je u pretraživanju povijesti važan i trenutak obavljanja pojedine operacije, tako se za svaku povijesnu relaciju _pr stvara još jedan indeks, nad atributom koji označava vremenski indeks obavljanja operacije (OpTimestamp).

Algoritam 6.6: *GENINDEX*

procedura **genIndex** (ulazni argument **bp**: baza podataka, /* radna */
ulazni argument **pbp**: baza podataka /* povijesna */)

varijabla

r, **_pr**: relacija

PK: skup atributa

za svaku **r** ∈ **bp**

PK ← atributi koji čine primarni ključ relacije **r**

stvori indeks nad atributima **PK** relacije **_pr** u bazi **pbp**

stvori indeks nad atributom OpTimestamp relacije **_pr** u bazi **pbp** ❐

Primjer 6.6: Izvođenjem gornjeg algoritma nad bazama podataka iz prethodnih primjera, u povijesnoj bazi podataka će se stvoriti šest novih indeksa, i to:

- nad relacijom _posoba indeks nad atributom oznOsoba, te indeks nad atributom OpTimestamp
- nad relacijom _pposao indeks nad atributom sifPosao, te indeks nad atributom OpTimestamp
- nad relacijom _pplaca indeks nad atributima oznOsoba i mjesec, te indeks nad atributom OpTimestamp ❐

6.2.7. Dozvole korištenja

Proces implementacije snimanja traga na navedeni način obavlja administrator baze podataka (DBA) ili administrator sustava za upravljanje bazama podataka (DBSA). Na kraju implementacijskog procesa potrebno je odrediti ispravne dozvole svih sudionika sustava za pristup i izmjenu povijesnih podataka.

Definiranje događaja koji će se snimati obavlja se evidencijom podataka u relaciju r_{AUDIT}. Sadržaj ove relacije mogu čitati svi korisnici. Prema principu raspodjele dužnosti, dozvole za unos, izmjenu i brisanje podataka iz ove relacije treba imati samo osoba zadužena za definiranje snimanja traga.

Snimanje povijesnih podataka se obavlja pozivom naredbi iz okidača. Na izvođenje okidača ne utječu nikakve dozvole sustava za upravljanje bazama podataka, tako da se ovdje bilježi sve što se u relacijama događa.

Za unos, izmjenu i brisanje podataka iz povijesnih relacija nijedan korisnik ne treba imati dozvolu. Čitanje snimljenog traga iz povijesnih relacija treba biti dopušteno samo onim korisnicima koji su za to zaduženi. Prema principu raspodjele dužnosti, to je samo osoba zadužena za pregled i analizu traga.

Za unos, izmjenu i brisanje podataka iz povijesnih relacija u povijesnoj bazi podataka također nijedan korisnik ne treba imati dozvolu. Za dodatno osiguranje ovih relacija (npr. kako bi se i administratora obeshrabrilo da radi bilo kakve izmjene) mogu se realizirati i okidači nad svim tim relacijama. Ovi okidači će u svim slučajevima izmjene i brisanja podići iznimku, a također i u slučajevima unosa, ukoliko aktivni korisnik nije proces *backupHistory*. U navedenom primjeru bi administrator mogao onemogućiti okidač, no to ostaje zapisano u snimljenom tragu samog sustava za upravljanje bazama podataka, na koji administrator ne bi trebao imati utjecaj.

Čitanje snimljenog traga iz povijesnih relacija u povijesnoj bazi podataka treba omogućiti onim korisnicima koji su za to zaduženi, a prema principu raspodjele dužnosti, to je samo osoba zadužena za pregled i analizu traga.

Proces *backupHistory* treba imati dozvole za pristup radnoj i povijesnoj bazi podataka. U radnoj bazi podataka ovaj proces može čitati i brisati podatke iz povijesnih relacija. U povijesnoj bazi podataka može samo unositi nove zapise u povijesne relacije.

Primjer 6.6: Potrebne dozvole nad relacijama iz prethodnih primjera, s obzirom na korisnike sustava, raspisane su u tablici 6.2.

relacija	potrebne dozvole			
	korisnik zadužen za definiranje snimanja traga	osoba zadužena za pregled i analizu traga	„obični“ korisnik baze podataka	*backupHistory*
r_{AUDIT}	S, I, U, D	-	S	-
_osoba, _posao, _placa	-	S	-	S, D
_posoba, _pposao, _pplaca	-	S	-	S, I

Tablica 6.2. Dozvole korisnicima nad relacijama iz primjera

Oznake dozvola u tablici odnose se na SQL naredbe: SELECT (S), INSERT (I), UPDATE (U) i DELETE (D). Oznaka - znači da korisnik nema nikakvu dozvolu nad relacijom. ❐

6.3. Implementacija

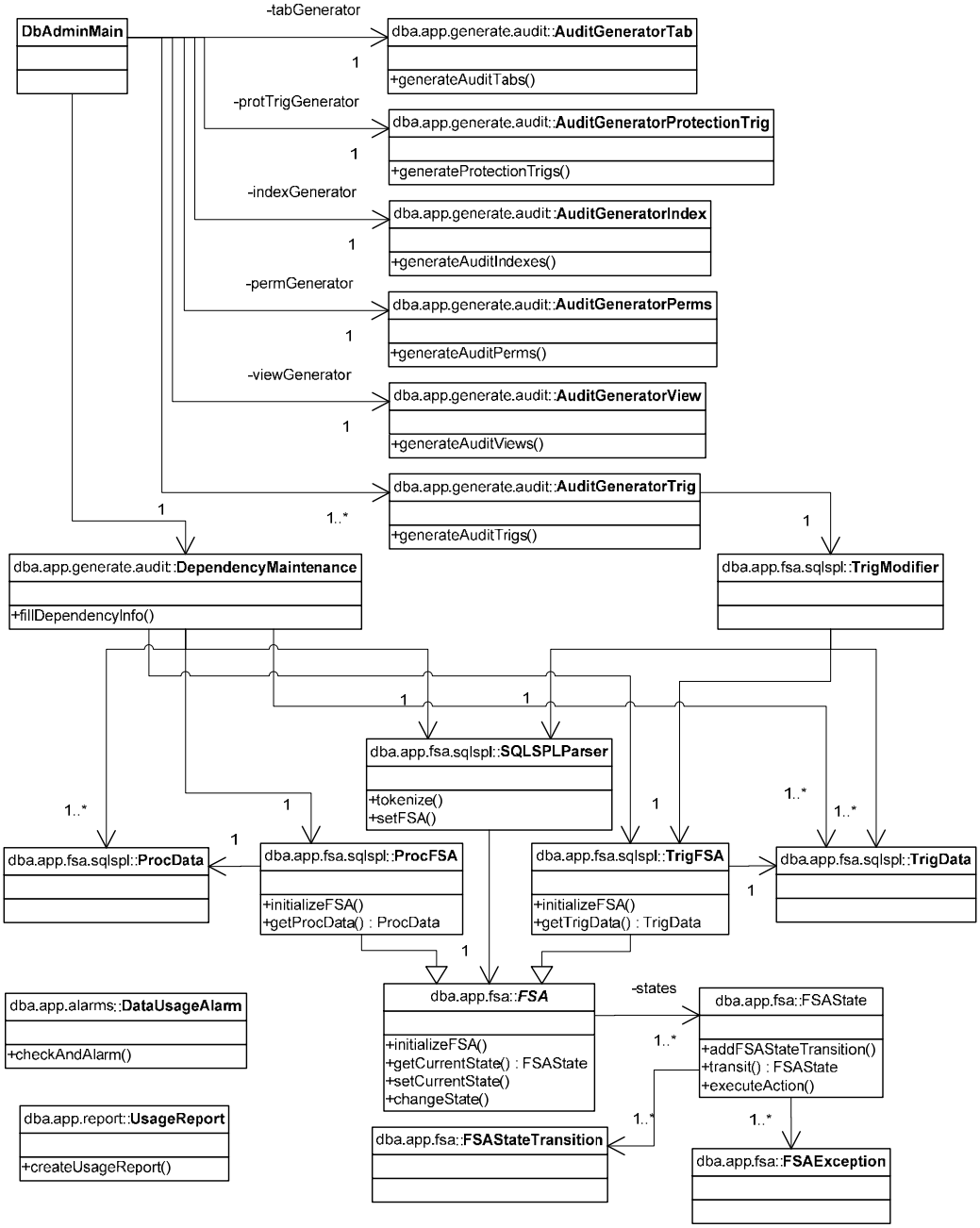
Programsko rješenje navedene ideje realizirano je u programskom jeziku Java koristeći objektno orijentirani dizajn. Proširenje rječnika baze podataka relacijom r_{AUDIT} jednokratni je postupak i obavlja se izvan programskog koda. Ostali algoritmi izvode se po potrebi, kako se mijenja sadržaj relacije r_{AUDIT}, i oni su realizirani u metodama objekata.

Većina razreda u kojima su implementirani navedeni algoritmi, kao i neki od onih razreda čija će uporaba tek biti opisana, dio su pomoćnog programa DbAdmin, opisanog i u [Mačkala2004]. Na slici 6.3 je prikazan pojednostavljeni UML dijagram dijela ovog programa koji je zanimljiv za ovaj rad.

Algoritam *SPR* (stvaranje povijesnih relacija) implementiran je u razredu AuditGeneratorTab. Ovaj razred također implementira i algoritam *SPBP* (stvaranje povijesne baze podataka).

Za izvršavanje algoritam za nadopunu okidača *PNO* zadužen je razred AuditGeneratorTrig. Pri tom se provjera sadržavanja naredbi za snimanje traga u tekstovima okidača obavlja tako što se naredbe za stvaranje okidača parsiranjem prikažu kao skup objektnih varijabli. U tu svrhu izgrađen je jednostavan parser koji obavlja razdvajanje pojedinih sintaksnih elemenata naredbe za stvaranje okidača. Ti sintaksni elementi predstavljaju ulaz u konačni automat (eng. *Finite State Automata – FSA*) pomoću kojeg se može zaključiti struktura programskog kôda okidača. Automat je na gornjoj slici predstavljeni razredom FSA, dok je FSAState razred koji definira jedno stanje automata. FSAStateTransition je opis prijelaza automata iz jednog stanja u drugo, dok je FSAException standardna iznimka koja se podiže u

slučaju pogreške rada automata. Parser koji raspoznaje tokene SQL i SPL (*Stored Procedures Language*) jezika, te na osnovu njih pokreće izmjene stanja automata implementiran je u razredu SQLSPLParser. Konkretna implementacija automata za određivanje forme okidača definirana je u razredu TrigFSA, dok se sama struktura programskog kôda okidača nakon parsiranja i prijelaza automata kroz stanja čuva u objektu iz razreda TrigData.



Slika 6.3. Razredi pomoćnog programa DbAdmin vezani uz snimanje traga

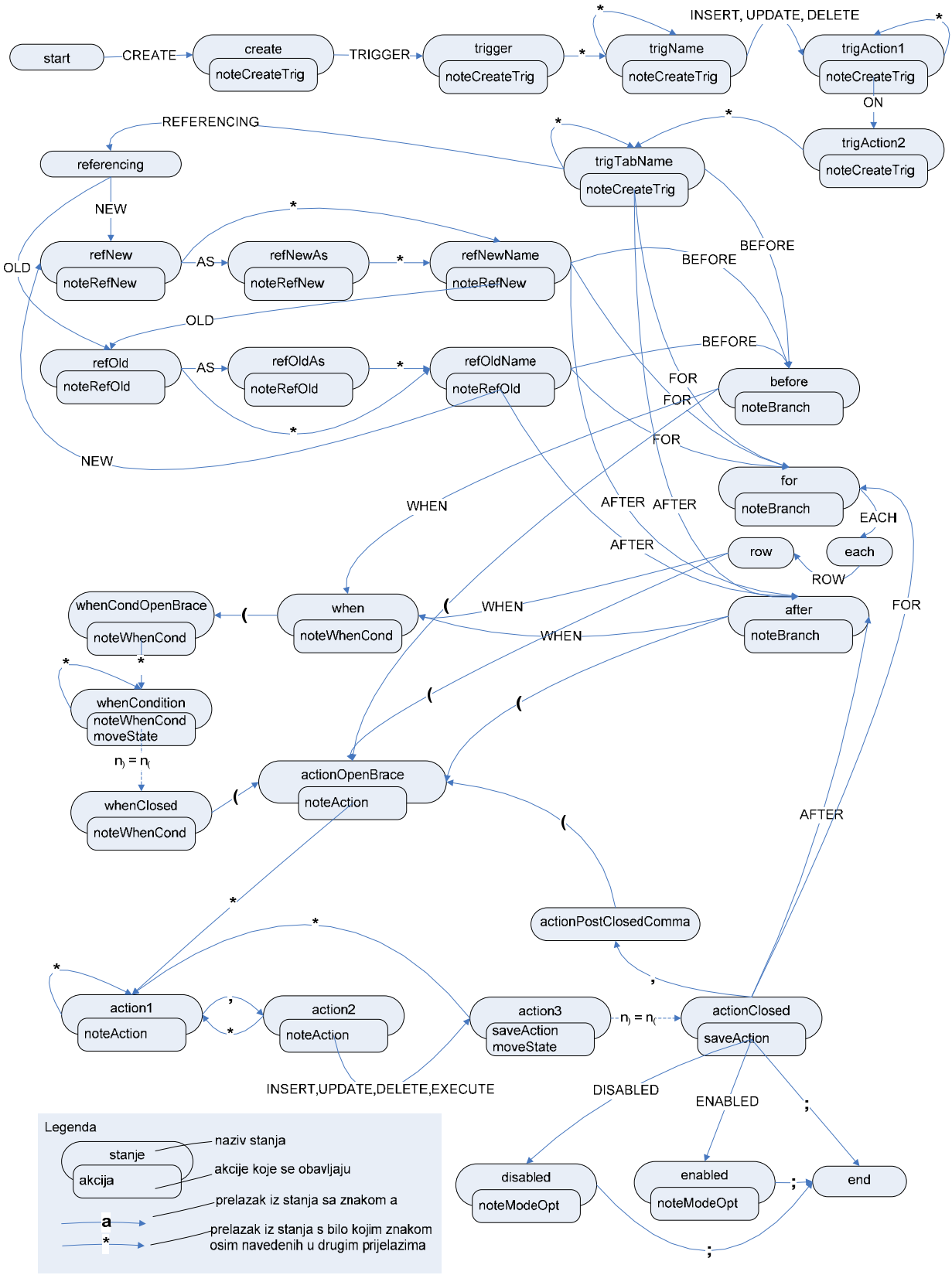
Automat koji određuje formu okidača zapravo je implementacija automata s akcijama [Srbljić2003]. Mealyjev automat predstavlja proširenje konačnog automata koje se odnosi na mogućnost obavljanja posebnih akcija ovisno o trenutnom stanju u kojem se automat nalazi i trenutnom ulazu u automat. Mooreov automat jest proširenje konačnog automata koje se odnosi na mogućnost obavljanja akcija koje ovise samo u ulasku u pojedino stanje. Iako je moguće postići isti učinak korištenjem bilo kojeg od ovih modela [Wagner2005], Mealyjev automat se pokazao prikladnijim ovom zadatku.

Korištenjem konačnog automata s akcijama postiže se bolja sposobnost automata za donošenje potrebnih zaključaka. Implementacija ovog automata dodatno je proširena uvjetnom promjenom stanja, tj. akcija će sama promijeniti stanje automata ovisno o ulazu ukoliko je ispunjen određeni uvjet. Na slici 6.4 je prikazan Mealyjev automat korišten za određivanje forme okidača, pri čemu su uvjetne promjene stanja prikazane isprekidanim crtama, a uz njih je naveden uvjet koji mora biti ispunjen. Uvjet $n_{)} = n_{(}$ implicira da nakon parsiranja ulaza u automat broj parsiranih znakova „)“ mora biti jednak broju znakova „(“ – ovim je određeno da se završava naredba koju pojedina akcija okidača izvršava.

Akcije sa slike 6.4 koje obavlja automat za određivanje strukture okidača definirane su u tablici 6.3.

oznaka akcije	opis akcije
noteCreateTrig	zabilježi početni dio naredbe za stvaranje okidača
noteRefNew	zabilježi dio naredbe za stvaranje okidača koji se odnosi na novu n-torku relacije
noteRefOld	zabilježi dio naredbe za stvaranje okidača koji se odnosi na novu n-torku relacije
noteBranch	zabilježi podatke o trenutku izvršavanja akcija koje će biti definirane sljedećim dijelom okidača
noteWhenCond	zabilježi podatke o uvjetima izvršavanja akcija koje će biti definirane sljedećim dijelom okidača
noteAction	zabilježi podatke o akciji
saveAction	sačuvaj podatke o akciji definiranoj u prethodnom dijelu okidača, zajedno s trenutkom i uvjetima izvršavanja
changeState	uvjetno promijeni stanje automata
noteModeOpt	zabilježi podatke o načinu stvaranja okidača

Tablica 6.3. Akcije automata za određivanje strukture okidača.

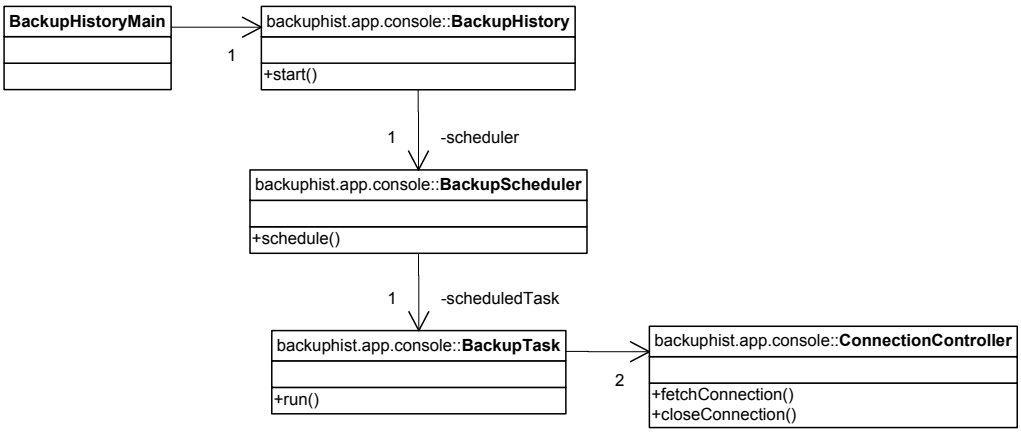


Slika 6.4. Automat za određivanje strukture okidača

Dodatne okidače nad povijesnim relacijama u povijesnog bazi podataka obavlja objekt iz razreda AuditGeneratorProtectionTrig. Indekse za ubrzavanje dohvata podataka iz tih relacija generira programski kod u razredu AuditGeneratorIndex.

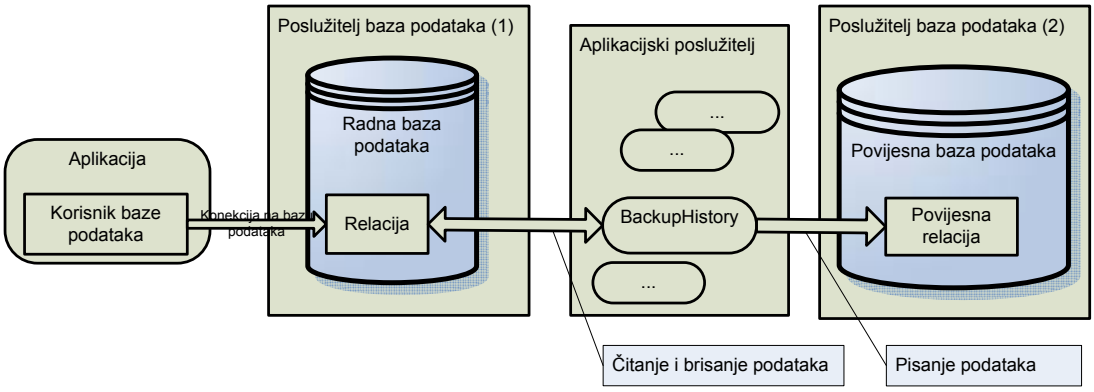
Prilikom implementacije procesa *backupHistory* važna karakteristika koju je potrebno podržati je atomarnost prebacivanja podataka iz jedne relacije u drugu, kako se podaci ne bi gubili čišćenjem povijesnih relacija u radnoj bazi podataka. Na slici 6.5 je prikazan UML dijagram s osnovnim razredima procesa.

Razred BackupHistory zadužen je za inicijalizaciju procesa, uspostavljanje početnih postavki, te pokretanje instance razreda BackupScheduler. U ovom razredu implementirano je pokretanje nakon isteka perioda čekanja. Svaki put nakon isteka perioda u izdvojenoj niti pokreće se izvršna metoda razreda BackupTask. U ovom je razredu implementirana bit procesa *backupHistory* i atomarnost prenošenja podataka. Razred koristi dvije instance razreda ConnectionController – po jednu za radnu i povijesnu bazu podataka.

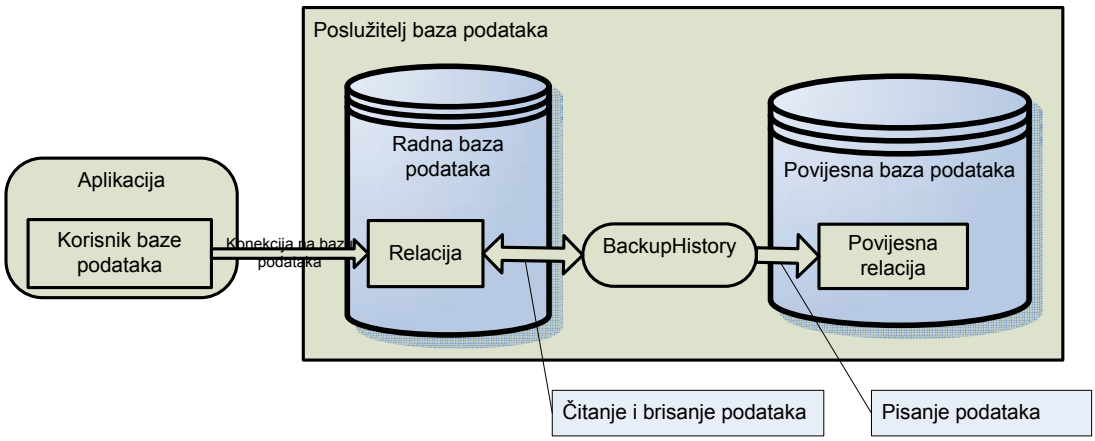


Slika 6.5. Osnovni razredi implementacije procesa *backupHistory*

Zahtjevnost procesa *backupHistory* u odnosu na sustave za upravljanje bazama podataka među kojima prenosi podatke ovisi samo o količini podataka koje je potrebno prenijeti. Praćenjem same uporabe procesa može se dovoljno dobro odrediti koliki je vremenski period u kojem je potrebno ponovno prenijeti povijesne podatke. Kako proces ne zahtijeva veliku računalnu procesorsku snagu, nije potrebna preporuka za smještanjem istoga na izdvojeni poslužitelj, iako je to moguće. Najsloženija računalna konfiguracija predstavlja dva poslužitelja baza podataka i jedan aplikacijski poslužitelj na kojem bi se između ostalih izvodio i *backupHistory* proces (slika 6.6), dok se u najjednostavnijoj konfiguraciji sve može obavljati na jednom fizičkom poslužitelju (slika 6.7).



Slika 6.6. Izvođenje procesa *backupHistory* na aplikacijskom poslužitelju, pri čemu su radna i povijesna baza podataka smještene na izdvojenim poslužiteljima



Slika 6.7. Izvođenje procesa *backupHistory* na poslužitelju baza podataka, pri čemu su i radna i povijesna baza podataka smještene na istom poslužitelju

Dodjela dozvola za sistemskog korisnika baze podataka koji obavlja proces *backupHistory* definirana je u razredu AuditGeneratorPerms na slici 6.3.

6.4. Mogućnosti proširenja

Opisani model snimanja traga moguće je proširiti na više načina. Neki od njih su opisani u ovom poglavlju.

Većina mogućih proširenja odnosi se na samu definiciju snimanja traga. Prema opisanom modelu, za pojedinu relaciju je moguće definirati veličinu odgovarajuće povijesne relacije te da li će se snimanje obavljati ili ne. Pri tom se snimanje obavlja bezuvjetno, tj. za sve korisnike baze podataka, u svim situacijama, te za sve operacije – unosa, izmjene i brisanja podataka.

6.4.1. Snimanje traga o pristupanju podacima

Opisani način snimanja traga ograničen je na operacije unosa, izmjene i brisanja podataka. Pristupanje nekim osjetljivim podacima također bi trebalo biti uzeto u obzir prilikom snimanja traga. Kako mnogi aktualni sustavi za upravljanje bazama podataka sada omogućavaju okidače na relacijama koji se okidaju na akciju dohvata podataka (SELECT), tako je ovaj način snimanja traga moguće proširiti i s tom mogućnosti. Pri tom bi algoritam za stvaranje i nadopunu okidača *PNO* radio s proširenim skupom operacija OI, koji bi sada sadržavao i operaciju čitanja.

Obzirom da je u većini informacijskih sustava omjer čitanja i pisanja podataka izrazito na strani čitanja, te da se u nekim operacijama čitanja dohvaćaju velike količine n-torki, jasno je da bi ovakav način snimanja traga vrlo brzo zagušio bazu podataka. Zbog toga bi snimanje traga o čitanju podataka obavezno trebalo koristiti u kombinaciji sa sljedećim opisanim proširenjima – definiranjem operacija za koje se snima trag za pojedinu relaciju, korisnika za kojeg se snima trag, te općenito postavljanjem uvjeta za snimanje traga.

6.4.2. Definiranje operacija za koje se snima trag

Ovo proširenje odnosi se na mogućnost definiranja za koje će se operacije snimati izmjene, umjesto da se uvijek snimaju za sve operacije nad podacima. Za to bi bilo potrebno proširiti relacijsku shemu R_{AUDIT} novim atributom, Operacije, pri čemu je

DOM(Operacije) = {∅, 'U', 'I', 'D', 'UI', 'UD', 'ID', 'UID'}.

Ukoliko vrijednost atributa Operacije u n-torki relacije r_{AUDIT} koja opisuje snimanje traga za relaciju r sadrži oznaku operacije koja se obavlja nad podacima, tada se u tom slučaju snima trag, a inače ne. Na primjer, operacija unosa n-torke u relaciju r će biti evidentirana ukoliko je vrijednost atributa Operacije bila jedna od ovih: 'I', 'UI', 'ID', 'UID'. Za slučaj da je vrijednost atributa prazna vrijednost (∅), ne obavlja se snimanje traga ni u kojem slučaju. Ovo proširenje dovodi do redundancije koja se očituje u postojanju atributa Snimanje – atribut Operacije sada određuje hoće li se, a i u kojem slučaju, snimati trag za pojedinu relaciju, tako da atribut Snimanje više ne treba postojati u relaciji r_{AUDIT}. Nova shema te relacije bila bi:

R_{AUDIT} = RelSh, Velicina, Operacije.

Prilikom implementacije ovog proširenja nužna je prilagodba izvedbe algoritma *PNO* za nadopunu okidača u koju je potrebno definirati eventualne izmjene tijela okidača prema vrijednosti atributa Operacija, te prema toj vrijednosti dodati i uvjete za snimanje.

6.4.3. Definiranje korisnika za koje se snima trag

Treće potencijalno proširenje odnosi se na mogućnost odabira korisnika za koje će se snimanje obavljati. Potrebno je osigurati neko mjesto na kojem će biti moguće definirati sve korisnike za koje se želi snimati trag za određenu relaciju. Kako se radi o podacima koji su ovisni o samoj definiciji snimanja traga, očito je da trebaju biti smješteni u dodatnu relaciju proširenog rječnika podataka, r_{AUDIT_USER}. Nakon

normalizacije na treću normalnu formu, proširenje rječnika podataka sadrži relacijske sheme:

R_{AUDIT} = RelSh, Velicina, Operacije

$K_{R\text{-}AUDIT}$ = RelSh

R_{AUDIT_USER} = RelSh, Username

$K_{R\text{-}AUDIT\_USER}$ = RelSh, Username

pri čemu je novi atribut:

- Username – korisničko ime korisnika za kojeg se želi snimati trag.

Za učinkovitost ovog proširenja najprije je potrebno odrediti očekivano ponašanje sustava pri snimanju traga. Ukoliko je za promatranu relaciju r u relaciji r_{AUDIT} definirano da se obavlja snimanje traga, te je u relaciji r_{AUDIT_USER} definirana jedna ili više n-torki sa zapisima o korisnicima za koje se obavlja snimanje traga, tada će sustav u povijesnu relaciju _r upisivati samo n-torke originalne relacije r koje su na neki način mijenjali samo taj/ti korisnici. Ukoliko u relaciji r_{AUDIT_USER} nije za relaciju r definiran nijedan korisnik, a snimanje traga se obavlja, tada se ono treba obavljati za sve korisnike baze podataka. Druga prilagodba koju je potrebno napraviti za ovo proširenje odnosi se na izvedbu algoritma *PNO* za nadopunu okidača u koju je potrebno uključiti navedeno ponašanje sustava. Najjednostavniji način prilagodbe bio bi dodavanje uvjeta u tijelo okidača na mjesto upisa zapisa u povijesnu relaciju, ako se radi o snimanju traga za samo određeni skup korisnika.

6.4.4. Postavljanje uvjeta snimanja traga

Slijedom prethodnih proširenja nameće se proširenje snimanja traga dodavanjem uvjeta. Ovo proširenje je važno zbog diskovnog prostora kojeg zauzima snimljeni trag, koji nije zanemariv, a u nekim situacijama je i neopravdano velik. Naime, snimanje traga možda nije nužno u svim slučajevima izmjene podataka određene relacije. Moguće je da se potrebe za snimanjem traga mogu svesti na nekoliko slučajeva. Na primjer, izmjena naziva radnog mjesta u relaciji posao može biti razmjerno nevažna, ali je izmjena koeficijenta plaće koji prati to radno mjesto vrlo važna. U tom slučaju dovoljan uvjet za definiranje snimanja traga bio bi: obavi snimanje samo ako se radi o izmjeni koeficijenta plaće.

Kako je za svaku relaciju moguće definirati više uvjeta pod kojima će se trag snimati, a korisno svojstvo bi bilo i da je uvjete moguće povezati uz korisnike za koje se trag snima, potrebno je novo proširenje rječnika podataka. Nakon normalizacije na treću normalnu formu, proširenje rječnika podataka sadržavalo bi sljedeće sheme:

R_{AUDIT} = RelSh, Velicina, Operacije $K_{R\text{-}AUDIT}$ = RelSh

R_{AUDIT_USER} = RelSh, Username $K_{R\text{-}AUDIT\_USER}$ = RelSh, Username

R_{AUDIT_COND} = RelSh, Condition $K_{R\text{-}AUDIT\_COND}$ = RelSh, Condition

$R_{AUDIT\_COND\_USER}$ = RelSh, Condition, Username

$K_{R\text{-}AUDIT\_COND\_USER}$ = RelSh, Condition, Username

gdje je novi atribut:

- Condition – opis uvjeta koji se primjenjuje na snimanje traga.

Očekivano ponašanje sustava pri snimanju traga u ovom slučaju naglašava sadržaj relacija r_{AUDIT_COND} i $r_{AUDIT\_COND\_USER}$. Ukoliko za relaciju r ne postoji nijedan zapis u relaciji r_{AUDIT_COND}, a za tu relaciju je u r_{AUDIT} definirano da se obavlja snimanje, tada će ono biti bezuvjetno. Ako postoji zapis, tada se promatra i sadržaj relacije $r_{AUDIT\_COND\_USER}$. Ukoliko u njoj postoji odgovarajuća n-torka, onda se navedeni uvjet primjenjuje samo na tog korisnika, a inače se primjenjuje na sve korisnike. Koristeći ovaj model moguće je definirati i bezuvjetno snimanje za pojedine korisnike, tako da i uvjetno snimanje za sve korisnike, ali i kombinaciju ova dva pristupa.

I za ovo proširenje je potrebno prilagoditi izvedbu algoritma *PNO* za nadopunu okidača u koju je potrebno uključiti opisano ponašanje sustava. Kao i u drugim slučajevima, i ovdje se radi o dodavanju kombiniranih uvjeta u tijelo okidača na mjesto upisa zapisa u povijesnu relaciju.

6.4.5. Uklanjanje zastarjele povijesti

Tijekom dugotrajnog rada ovog načina snimanja traga, količina snimljenog traga može postati vrlo velika. Uz to, dobar dio tih podataka može biti zastario i zbog toga nezanimljiv, ili je za njih istekla obveza čuvanja povijesnih podataka. Zbog ovoga je potrebno proširiti sustav i s procesom uklanjanja zastarjelih podataka iz povijesnih relacija.

Vremenski period nakon kojeg se podaci smatraju zastarjelima moguće je definirati na jednom mjestu, tako da vrijedi za cijeli sustav ili, preferabilno, za svaku pojedinu relaciju čiji se trag snima. U tom slučaju, potrebno bi bilo proširiti relacijsku shemu

R_{AUDIT} = RelSh, Velicina, Operacije, TrajanjePovijesti

gdje je novi atribut:

- TrajanjePovijesti – vremenski period koji treba isteći da bi se povijesni podaci izbrisali iz povijesnih relacija.

U cilju realizacije ovog proširenja, bilo bi potrebno uvesti u sustav novi proces, *expireHistory*. Ovaj bi se proces također trajno izvodio, te u određenim vremenskim periodima za sve povijesne relacije provjeravao da li postoje zapisi koji su prema vremenu snimanja stariji od vremena određenog vrijednošću atributa TrajanjePovijesti odgovarajuće n-torke u relaciji r_{AUDIT}. Ukoliko takvi zapisi postoje, ovaj proces bi ih izbrisao iz relacije.

Slično kao i proces *backupHistory*, ni ovaj proces ne bi imao velike zahtjeve na procesorsku snagu računala na kojem se izvodi, no kako se radi o uklanjanju n-torki iz povijesnih relacija u povijesnoj bazi podataka, logično mjesto za smještanje ovog procesa je na poslužitelju baza podataka na kojem je smještena i povijesna baza podataka.

6.4.6. Arhiviranje zastarjele povijesti

Svaki informacijski sustav može imati i vlastita pravila o dugotrajnosti čuvanja povijesnih podataka, bilo da se radi o zakonskoj potrebi ili o predostrožnosti odgovornih za vođenje zbirke podataka.

Kako uglavnom neće biti potrebno trajno čuvati povijest svih relacija, tako je potrebno omogućiti definiranje za koje relacije će se ta povijest trajno čuvati. Ovo je moguće ostvariti još jednim proširenjem relacijske sheme

R_{AUDIT} = RelSh, Velicina, Operacije, TrajanjePovijesti, TrajnoArhiviranje

gdje je novi atribut:

- TrajnoArhiviranje – oznaka da li se za opisanu relaciju trajno čuva povijest,

s domenom

DOM(TrajnoArhiviranje) = {'Da', 'Ne'}.

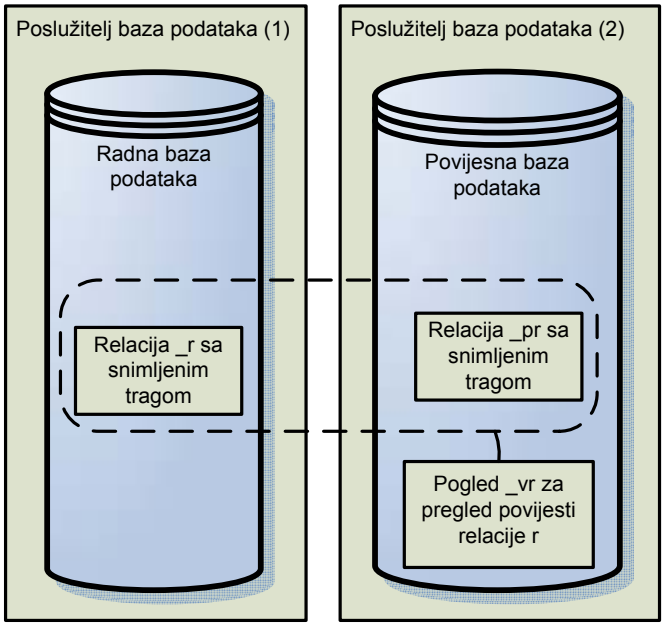
Proširenje se može izvesti izmjenom procesa *expireHistory*, na način da se za pojedinu relaciju, pri provjeri postojanja n-torki koje bi zbog njihove starosti trebalo ukloniti, ujedno vrši i provjera da li se te n-torke trebaju negdje trajno pohraniti, prema vrijednosti atributa TrajnoArhiviranje relacije r_{AUDIT}. Ukoliko se radi o slučaju da je n-torke potrebno trajno pohraniti, njih se izvozi (*export*) iz relacije i pripaja odgovarajućoj datoteci operacijskog sustava u kojoj se čuva trajna povijest promatrane relacije. Drugi proces, *archiveHistory*, koji se trajno izvodi na poslužitelju s povijesnom bazom podataka, s dugim periodom čekanja, zadužen je za dodavanje datoteke na neki *WORM* medij na kojem se čuva trajna povijest.

7. Analiziranje snimljenog traga o izmjenama podataka

Trag koji nastaje snimanjem opisanim u prethodnom poglavlju moguće je koristiti za različite analize. U ovom poglavlju prikazane su tri uporabe snimljenog traga, forenzičke i sigurnosne analize.

7.1. Pregled cjelokupne povijesti relacije

Obavljanje forenzičkih analiza treba odgovoriti na pitanje tko je, kada i što napravio nad kojim podacima. Kako bi se za relaciju za koju se snima trag moglo obavljati ovakve analize, potrebno je prvenstveno omogućiti korisniku zaduženom za analiziranje da na jednom mjestu može vidjeti cjelokupnu povijest relacije. Obzirom da je snimljeni trag relacije r podijeljen u dvije relacije, _r i _pr, te je ukupna povijest relacije predstavljena unijom n-torki tih relacija, potrebno je implementirati tu uniju. To se najjednostavnije može napraviti stvaranjem pogleda u bazi podataka. Pogled _vr treba biti definiran tako da vraća sadržaj relacija _r i _pr, pri čemu treba uzeti u obzir da se te relacije nalaze u različitim bazama podataka, a moguće i na različitim poslužiteljima (slika 7.1). Osim ispravno napisane SQL naredbe za stvaranje pogleda koji će dohvaćati podatke s dva poslužitelja, potrebno je dakle i osigurati da poslužitelji i sami sustavi za upravljanje bazama podataka na njima mogu dopustiti jedan drugom čitanje podataka.



Slika 7.1. Pogled _vr za prikaz povijesti r kreiran u povijesnoj bazi podataka

Stvaranje opisanih pogleda potrebno je učiniti za svaku relaciju za koju se obavlja snimanje traga. Postojanje ovih pogleda je prvi korak prema kvalitetnoj analizi povijesti, bilo da će ona ostati na razini SQL naredbi pomoću kojih će osoba

zadužena za pregled traga dohvaćati podatke, ili će biti ti pogledi biti iskorišteni kao dio složenijeg analitičkog alata.

Povijesni podaci se nalaze u dvije baze podataka, tako da je moguće poglede stvoriti ili u radnoj ili u povijesnoj bazi podataka. Obzirom da je povijesna baza podataka namijenjena samo povijesnim podacima, i može se nalaziti na izdvojenom poslužitelju tako da upiti koji se izvode na njoj neće opterećavati radni sustav, logično je postaviti poglede upravo u tu bazu podataka. Još važniji argument za tu odluku proizlazi iz razumne pretpostavke da relacije u povijesnoj bazi podataka sadrže nesrazmjerno više podataka od odgovarajućih relacija u radnoj bazi podataka. Ukoliko se ove dvije baze podataka nalaze na dva SUBP-a, pri izvođenju distribuiranog upita kojeg definira određeni pogled, može se dogoditi da se podaci najprije dohvaćaju iz jednog SUBP-a u drugi, te se odatle zajedno dohvaćaju na klijentsko računalo, gdje se prikazuju korisniku. Sam način rada u opisanom slučaju ovisi o načinu rada samog sustava za upravljanje bazama podataka, klijentskog programa u kojem se izvodi sam upit nad pogledom, a problem eskalira ovisno o količini podataka koja se dohvaća, odnosno postavljenim uvjetima dohvata. Ukoliko je pogled $\_vr_i$ za pregled povijesti stvoren na radnoj bazi podataka, upitom kojim bi se dohvaćali svi povijesni podaci za relaciju r_i koje taj pogled definira, moglo bi se dogoditi da se cjelokupni sadržaj povijesne relacije $\_pr_i$ najprije dohvati iz SUBP-a koji sadrži povijesnu bazu podataka u privremenu memoriju radnog SUBP-a, zatim im se pridruže podaci iz relacije $\_r_i$, te se zajedno prezentiraju korisniku. Očito je da je ovo primjer neracionalne uporabe radnog SUBP-a, te da je nužno da pogledi za pregled povijesti budu smješteni u povijesnu bazu podataka.

Stvaranje pogleda za pregled povijesti relacija za koje se obavlja snimanje traga može se opisati algoritmom *SPP*. Kako je navedeno u diskusiji, ovaj se algoritam izvodi u povijesnoj bazi podataka **pbp**.

Algoritam 7.1: *SPP*

procedura **spp** (ulazni argument **bp**: baza podataka, /* radna */
ulazni argument **PRBP**: poslužitelj baze podataka) /* radne */
varijabla
_r, **_pr**: relacija
_v: definicija pogleda
za svaku **_pr** ∈ **pbp**
_r ← odgovarajuća povijesna relacija za **_pr** iz baze **bp** s poslužitelja **PRBP**
_v ← definicija pogleda za pregled **_r** ∪ **_pr**
stvori pogled **_v** u bazi podataka **pbp** □

Primjer 7.1: Izvršavanjem gornjeg algoritma u povijesnoj bazi podataka stvorenoj u primjeru 6.4 stvorit će se tri nova pogleda: _vosoba, _vposao i _vplaca. Neka je bp radna baza podataka, a PRBP sustav za upravljanje bazama podataka na kojem se ta baza podataka nalazi. Pogled _vosoba objedinit će povijesne podatke iz ove i radne baze podataka:

```
CREATE VIEW _vosoba ( oznOsoba,
                      ime,
                      prezime,
                      JMBG,
                      user,
                      opTimestamp,
                      txTimestamp,
                      operation,
                      sessionID) AS
  SELECT *
    FROM _posoba
  UNION
  SELECT *
    FROM bp@PRBP:_osoba;
```

Na sličan način bit će generirani i pogledi _vposao i _vplaca. □

Algoritam za stvaranje povijesnih pogleda implementiran je u programskom kodu razreda AuditGeneratorView prikazanog na slici 6.3 s ostalim razredima za generiranje objekata u bazama podataka.

7.2. Pregled djelatnosti korisnika u vremenskom periodu

Jedno od najčešćih pitanja na koje je pri provođenju forenzičkih analiza potrebno dati odgovor je: „što je ovaj korisnik radio u tom vremenu?“ Kako je u svjetlu izvođenja konkretnih operacija nad bazom podataka točno određeni trenutak mnogo specifičniji nego onaj kojeg se lako može odrediti, tako se ovaj zahtjev može proširiti na prikaz djelatnosti određenog korisnika u nekom vremenskom periodu.

U ovoj analizi postoji nekoliko ulaznih parametara. Potrebno je znati korisničko ime korisnika, k, čiju se aktivnost nad bazom podataka želi vidjeti, trenutak t_0 koji, zajedno s tolerancijom Δt, predstavlja vremenski okvir prema kojem će se filtrirati traženi podaci.

Rezultat analize je prikaz djelatnosti korisnika k u vremenu od $t_0 - \Delta t$ do $t_0 + \Delta t$. U prikazu je potrebno pokazati samo one n-torke svih relacija koje su obuhvaćene snimanjem traga na koje je utjecao korisnik k u traženom vremenskom periodu, te jasno naznačiti o kojoj se operaciji radi. Ako se radi o operaciji izmjene podataka, tada je potrebno prikazati i istu n-torku prije izmjene, kako bi korisnik analize mogao vidjeti što je napravljeno. Sve rezultate potrebno je u prikazu poredati točno prema redoslijedu obavljanja, te naznačiti one koje operacije koje su obavljene u istoj transakciji.

Zbog potrebe dohvata podataka iz potencijalno velikog broja relacija, te redanja pojedinih n-torki prema vremenu obavljanja, bez obzira na relaciju iz koje potiču, ovu je analizu jednostavnije implementirati izdvojeno od sustava za upravljanje bazama podataka nego unutar njega, kroz niz pohranjenih procedura. Dohvat povijesnih n-torki se obavlja korištenjem pogleda $\_vr_i$ za pregled povijesti cjelokupnih relacija, što znači da se i upiti obavljaju u sustavu za upravljanje bazama podataka u kojem je smještena povijesna baza podataka.

Opis rada analize djelatnosti korisnika u vremenskom periodu može se opisati algoritmom *PREGLED_DJELATNOSTI.*

Algoritam 7.2: *PREGLED_DJELATNOSTI*

procedura **pregled_djelatnosti** (
ulazni argument **pbp**: baza podataka, /* povijesna */
ulazni argument **k**: korisničko ime,
ulazni argument $\mathbf{t_0}$: vremenska oznaka,
ulazni argument $\mathbf{\Delta t}$: vrijeme)

varijabla
n, **n'**, $\mathbf{n_i}$: n-torka
ns: uređeni skup n-torki
_vr: pogled za pregled povijesne relacije r
PK: skup atributa

ns $\leftarrow \varnothing$

za svaki **_vr** $\in$ **pbp**
ns $\leftarrow$ **ns** $\cup$ {**n** $\in$ **_vr** | **n**[OpUser] = **k** $\wedge$ $(\mathbf{t_0} - \mathbf{\Delta t}) \leq$ **n**[OpTimestamp] $\leq (\mathbf{t_0} + \mathbf{\Delta t})$}

ns $\leftarrow$ poredaj skup **ns** prema vrijednosti atributa OpTimestamp

ispis(„početak transakcije“)

za svaku $\mathbf{n_i} \in$ **ns**, $0 < \mathbf{i} \leq$ card(**ns**)
ako je $\mathbf{i} > 1 \wedge \mathbf{n_i}$[TxTimestamp] $\neq \mathbf{n_{i-1}}$[TxTimestamp] tada
ispis(„kraj transakcije“)
ispis(„početak transakcije“)
ako je $\mathbf{n_i}$[Operation] = 'U' tada
PK $\leftarrow$ atributi koji čine primarni ključ n-torke $\mathbf{n_i}$
_vr $\leftarrow$ pogled iz kojeg je n-torka $\mathbf{n_i}$
n' $\leftarrow$ dohvati n-torku **n** $\in$ **_vr** s istim vrijednostima atributa **PK** kao i $\mathbf{n_i}$
i najvećim **n**[OpTimestamp] koji je još uvijek manji od $\mathbf{n_i}$[OpTimestamp]
ispis(**n'**)
ispis($\mathbf{n_i}$) ◻

Primjer 7.2: Neka su baze podataka iz prethodnih primjera dio produkcijskog informacijskog sustava već dulje vrijeme. Izvršavanje gornje procedure s parametrima koji određuju ispis djelatnosti korisnika marko u trenutku 10:53:50, 25. ožujka 2008. godine, s tolerancijom 300 sekundi, može rezultirati s ispisom kako je prikazano na slici 7.2.

Na slici su prikazane operacije grupirane unutar tri transakcije. Unutar transakcije operacije se poredane prema vremenu obavljanja, i uz svaku operaciju su prikazane n-torke relacija koje su obuhvaćene operacijom. Uz one atribute koji čine primarni ključ pojedine relacije, uvijek stoji oznaka (*PK*).

Korisnik: marko
Vrijeme: 25.03.2008 10:53:50.000
Tolerancija: 300 sekundi

TxTimeStamp	Session
25.03.2008 10:49:58.827	26599

osoba

OpTimeStamp	Operation	oznOsoba(PK)	ime	prezime	jmbg
25.03.2008 10:51:12.195	izmjena	0001028750	Ivan	Ivić	3011900010166
		0001028750	Ivan	Ivanić	3011900010166

TxTimeStamp	Session
25.03.2008 10:53:50.559	26599

posao

OpTimeStamp	Operation	sifPosao(PK)	nazPosao	koefPlaca
25.03.2008 10:53:57.796	unos	39656301	dizajner informacijskih sustava	2,45

osoba

OpTimeStamp	Operation	oznOsoba(PK)	ime	prezime	jmbg
25.03.2008 10:53:57.796	unos	0001039987	Ivana	Markić	2002900025555

placa

OpTimeStamp	Operation	oznOsoba(PK)	mjesec(PK)	iznosStimulacija	iznosHonorar
25.03.2008 10:53:57.796	unos	0001039987	01-2008	0,0	0,0

TxTimeStamp	Session
25.03.2008 10:54:06.339	26599

placa

OpTimeStamp	Operation	oznOsoba(PK)	mjesec(PK)	iznosStimulacija	iznosHonorar
25.03.2008 10:54:09.188	unos	0001039980	01-2008	1200,00	0,00
25.03.2008 10:55:09.188	unos	0001020017	01-2008	1000,00	300,00
25.03.2008 10:56:09.188	unos	0001028750	01-2008	1000,00	400,00
25.03.2008 10:56:09.188	unos	0001023343	01-2008	1200,00	0,00

Slika 7.2. HTML prikaz djelatnosti korisnika marko u vremenu 300 sekundi prije i nakon 10:53:50, 25.3.2008.

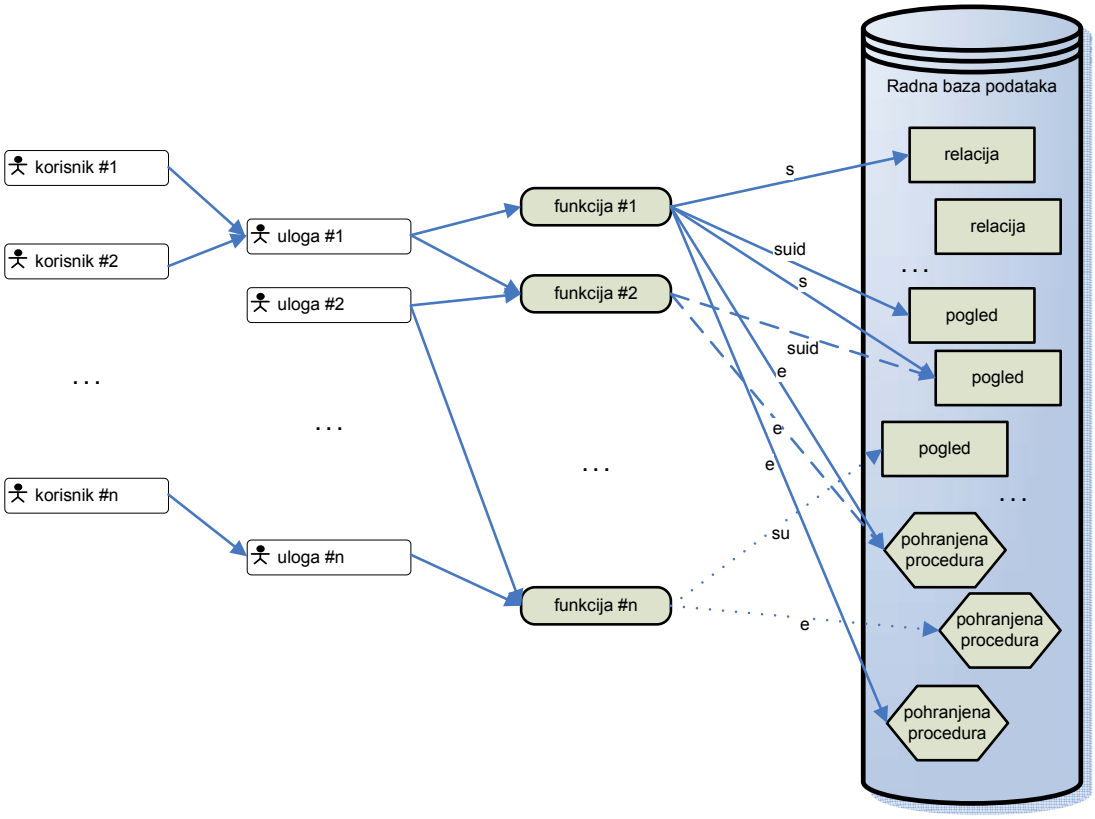
Prva se sastoji od samo jedne operacije, radi se o izmjeni zapisa u relaciji osoba. Prikazana su dva retka – stara i nova vrijednost n-torke, pa je moguće vidjeti da se radilo na izmjeni prezimena osobe. Druga transakcija pokazuje da se radilo o unosu novog zaposlenika – evidentirana je nova osoba, te i novo radno mjesto. Kako se ne obavlja snimanje traga za relaciju radno_mjesto, tako nije vidljivo stvarno zaposlenje ove osobe na to radno mjesto. U istoj transakciji je obavljen unos podataka o plaći za tekući mjesec, s pretpostavljenim vrijednostima, te se može zaključiti da je ovaj unos iniciran okidačem. Treća transakcija sadrži operacije unosa podataka o plaćama za nekoliko djelatnika za tekući mjesec. □

Algoritam *PREGLED_DJELATNOSTI* implementiran je u metodama razreda UsageReport prikazanog na slici 6.3.

7.3. Kontrola stvarnih i deklariranih dozvola korisnika

Jedno od mogućih proširenja rječnika podataka je vezano uz opis pravila pristupa podacima, detaljno opisano u [Anzil2005]. U raznim informacijskim sustavima implementiranima uz prošireni rječnik podataka kako je spomenuto u prethodnom poglavlju, ovo je proširenje korišteno za definiranje i realizaciju pravila pristupa podacima koja se temelje na ulogama (eng. *Role-Based Access Control – RBAC* [Ferraiolo2003]). Pojednostavljeno prikazano, stvarnim korisnicima informacijskog sustava pridjeljuju se uloge u bazi podataka (slika 7.3). Svakoj ulozi pripada jedna ili više funkcija, dok svaka funkcija predstavlja skup dozvola za provođenje DML

operacija nad objektima u bazi podataka (relacije, pogledi, pohranjene procedure, …).



Slika 7.3. Pojednostavljeni prikaz dodjele dozvola korisnicima sustava temeljem uloga i funkcija

Kako različite uloge u sustavu mogu značiti pristup različitom skupu podataka, npr. različitim n-torkama iste relacije, tako su za pristup i manipuliranje podacima stvoreni redukcijski pogledi. Svaki od redukcijskih pogleda nad pojedinom relacijom osigurava da korisnik ima pristup samo onom skupu n-torki koji njegova trenutna uloga implicira.

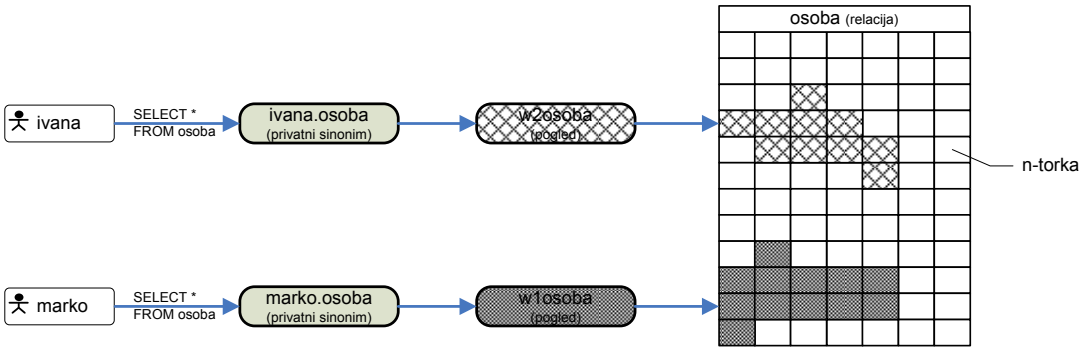
U konačnici, kako bi se olakšala izgradnja aplikativne programske podrške, za korisnike baze podataka se stvaraju privatni sinonimi za odgovarajuće redukcijske poglede. Privatni sinonim je objekt u bazi podataka koji predstavlja alternativno ime za drugi objekt, relaciju ili pogled, a stvara se za pojedinog korisnika. Ukoliko se privatni sinonimi za poglede nad istom relacijom nazovu jednako za sve korisnike kojima su stvoreni, tada se u aplikacijskom kodu može koristiti to isto ime za pristup podacima relacije, a sam sustav za upravljanje bazama podataka će, već prema tome o kojem se korisniku radi, prilikom izvođenja upita, zamijeniti sinonim odgovarajućim pogledom.

Primjer 7.3: Neka nad relacijom osoba postoje dva redukcijska pogleda, w1osoba i w2osoba, koji su definirani tako da, ukoliko ih izvodi isti korisnik, vraćaju različite

rezultate. Tada je moguće za korisnike marko i ivana stvoriti privatne sinonime na sljedeći način:

```
CREATE PRIVATE SYNONYM marko.osoba FOR w1osoba;

CREATE PRIVATE SYNONYM ivana.osoba FOR w2osoba;
```



Slika 7.4. Pristup različitim n-torkama relacije preko istoimenih sinonima

Ukoliko je u aplikacijskom kodu potrebno dohvatiti sve osobe, bit će dovoljno napisati upit:

```
SELECT *
  FROM osoba;
```

Ako za rad s aplikacijom bude prijavljena korisnica ivana, SUBP će vratiti podatke koje vraća pogled w2osoba, a ako za rad s aplikacijom bude prijavljen korisnik marko, SUBP vraća podatke koje bi vratio pogled w1osoba (slika 7.4). ❐

Na osnovu opisanog može se zaključiti nad kojim sve objektima u bazi podataka pojedini korisnik *treba imati* dozvole rada. Snimljeni trag, pak, omogućava dobivanje informacija o tome što je sve pojedini korisnik radio u zadanom vremenu. Usporedbom dozvola koje bi korisnik trebao imati s onim što je zbilja radio, može se zaključiti ukoliko je u nekom trenutku došlo do prekoračenja dozvola, te alarmirati administratore za sigurnost sustava.

7.3.1. Prikupljanje podataka o tranzitivnim dozvolama

Opis dozvola koje bi korisnik trebao imati nije adekvatan za ovu usporedbu. Naime, korisnik ne mora imati dozvolu za izmjenu zapisa u određenoj relaciji, no dovoljno je da ima dozvolu za izvođenje pohranjene procedure koja vrši tu izmjenu, pa će se izmjena i dogoditi, a bit će i snimljena u trag, kako je obavljena od strane tog korisnika. Ukoliko se pak, ta pohranjena procedura izvodi kao akcija okinuta okidačem nad nekom drugom relacijom, tada korisnik ne mora imati niti dozvolu za izvođenje ove procedure, nego samo za akciju nad drugom relacijom koja pokreće okidač. U cilju identificiranja svih relacija koje korisnik zbilja može promijeniti, potrebno je odrediti na koje sve indirektne načine može doći do promjene pojedine relacije, te na taj način zapisati dozvole koje se dobivaju tranzitivno, preko neke

druge dozvole. Tranzitivne dozvole evidentiraju se u posebnoj relaciji, $r_{TRANZIT}$, s relacijskom shemom:

$R_{TRANZIT}$ = ImeObjekta, ImeRelacije $K_{R\text{-}TRANZIT}$ = ImeObjekta, ImeRelacije

pri čemu je

DOM(ImeObjekta) = imena svih objekata iz radne baze podataka

DOM(ImeRelacije) = imena svih relacija iz radne baze podataka

Postojanje zapisa u relaciji $r_{TRANZIT}$ označava da postoji tranzitivna ovisnost objekta ImeObjekta o relaciji ImeRelacije u kontekstu dozvola. Ukoliko korisnik ima dozvolu nad objektom ImeObjekta, tada ima dozvolu i nad relacijom ImeRelacije.

Sadržaj relacije $r_{TRANZIT}$ ne mijenja se tijekom uobičajenog rada baze podataka, jer ovisi samo o objektima pohranjenima u bazu podataka, i to javnim i privatnim sinonimima, pohranjenim procedurama, okidačima i pogledima. Ukoliko se ovi objekti ne mijenjaju tijekom uobičajenog rada informacijskog sustava, tada bi sadržaj relacije $r_{TRANZIT}$ trebao biti jednak u tom vremenu. Pretpostavka je da je tako, te da se objekti baze podataka mijenjaju samo tijekom administrativnih zahvata pri kojima se cjelokupna baza podataka usklađuje s novijim verzijama pristupnih aplikacija. Da bi se sadržaj relacije $r_{TRANZIT}$ osvježio, nakon uskladbe baze podataka se izvršavaju programi koji implementiraju algoritme za popunjavanje te relacije, a koji su opisani u nastavku ovog poglavlja. Pretpostavlja se da je relacija prije izvođenja algoritama prazna.

Ovisnost pogleda, javnih i privatnih sinonima najlakše se utvrđuje pretraživanjem rječnika baze podataka, obzirom da svaki SUBP zbog interne funkcionalnosti mora čuvati ove ovisnosti jasno zapisane. Utvrđivanje ovisnosti pogleda o relacijama obavlja se izvođenjem algoritma *VIEWDEPEND* u radnoj bazi podataka. U sljedećim algoritmima se koristi i procedura dohvatiRelacijeOKojimaOvisiObjekt koja za bilo koji ulazni objekt iz baze podataka vraća relacije o kojima taj objekt ovisi, a te podatke dobiva iz postojećeg sadržaja relacije $r_{TRANZIT}$.

Algoritam 7.3: *VIEWDEPEND*

procedura **viewdepend** (ulazni argument **SBP**: shema baze podataka /*radne*/)

varijabla

v: pogled

n: n-torka

r: relacija

rs: skup relacija

o: objekt

os: skup objekata

za svaki pogled **v** ∈ **SBP**

os ← objekti o kojima ovisi pogled **v**

za svaki **o** ∈ **os**

rs ← relacije o kojima ovisi objekt **o**

za svaku **r** ∈ **rs**

n[ImeObjekta] ← ime pogleda **v**

n[ImeRelacije] ← ime relacije **r**

ako **n** ∉ $r_{TRANZIT}$ tada

upisi n-torku **n** u relaciju $r_{TRANZIT}$ ❐

Zbog spomenute osobine SUBP-ova da u rječniku podataka čuvaju podatke o ovisnosti pogleda o relacijama, ovaj algoritam je najlogičnije implementirati kao pohranjenu proceduru u radnoj bazi podataka.

Obzirom da se pogledi mogu graditi nad drugim pogledima, prikazani algoritam ima jedan nedostatak. Naime, u složenijim situacijama se može dogoditi da pogled koji se koristi za definiranje drugog pogleda (dakle, onaj o kojem drugi ovisi) još uvijek nije obrađen, te stoga u tom trenutku nije moguće odrediti tranzitivnu ovisnost relacija preko tog pogleda. Da bi se ove situacije pokrile, najjednostavnije je rješenje ponavljati izvođenje procedure dok god se sadržaj relacije $r_{TRANZIT}$ mijenja:

Algoritam 7.4: *VIEWDEPENDTOTAL*

procedura **viewdependtotal** (ulazni argument **SBP**: shema baze podataka /*radne*/)

varijabla

$\mathbf{ts_1}$, $\mathbf{ts_2}$: relacija

ponavljaj

$\mathbf{ts_1}$ ← $r_{TRANZIT}$

viewdepend (**SBP**)

$\mathbf{ts_2}$ ← $r_{TRANZIT}$

dok je $\mathbf{ts_1} \neq \mathbf{ts_2}$ ❐

Primjer 7.4: Neka je u radnoj bazi podataka nad relacijama osoba i placa stvoren pogled sa sljedećom definicijom:

```
CREATE VIEW v_osobeBezHonorara AS
   SELECT *
     FROM osoba
     WHERE EXISTS (
       SELECT *
       FROM placa
       WHERE placa.oznOsoba = osoba.oznOsoba
       AND iznosHonorar = 0);
```

U ovom slučaju SUBP će u rječniku podataka imati zapisano da je ovaj pogled ovisan o navedenim relacijama, a izvođenjem algoritama *VIEWDEPEND* će se to i upisati u relaciju $r_{TRANZIT}$:

$r_{TRANZIT}$	(ImeObjekta,	ImeRelacije)
	v_osobaBezHonorara	osoba
	v_osobaBezHonorara	placa

❐

Utvrđivanje ovisnosti privatnih i javnih sinonima prema relacijama obavlja se izvođenjem algoritma *SYNDEPEND*. Redoslijed izvođenja algoritama je važan jer se sinonimi mogu odnositi kako na relacije tako i na poglede. Zbog toga je bitno da se prije ispitivanja ovisnosti sinonima već poznaju ovisnosti pogleda o relacijama, tako da se tranzitivno preko pogleda može doći do ovisnosti sinonima o relacijama.

Algoritam 7.5: *SYNDEPEND*

procedura **syndepend** (ulazni argument **SBP**: shema baze podataka /*radne*/)
    varijabla
        s: sinonim
        n: n-torka
        r: relacija
        rs: skup relacija
        o: objekt
        os: skup objekata
    za svaki sinonim **s** ∈ **SBP**
        os ← objekti o kojima ovisi sinonim **s**
        za svaki **o** ∈ **os**
            rs ← relacije o kojima ovisi objekt **o**
            za svaku **r** ∈ **rs**
                n[ImeObjekta] ← ime sinonima **s**
                n[ImeRelacije] ← ime relacije **r**
                ako **n** ∉ $r_{TRANZIT}$ tada
                    upisi n-torku **n** u relaciju $r_{TRANZIT}$ ❐

Kao i algoritam za utvrđivanje ovisnosti dozvola pogleda nad relacijama, tako je i ovaj logično implementirati kao pohranjenu proceduru u radnoj bazi podataka.

Primjer 7.5: Neka je u radnoj bazi nad pogledom iz prethodnog primjera stvoren javni sinonim na sljedeći način:

```
CREATE PUBLIC SYNONYM osobaBezHonorara FOR v_osobaBezHonorara;
```

Nakon izvođenja algoritma *SYNDEPEND*, sadržaj relacije $r_{TRANZIT}$ bit će proširen dodatnim zapisima:

$r_{TRANZIT}$	(ImeObjekta,	ImeRelacije)
	osobaBezHonorara	osoba
	osobaBezHonorara	placa

❐

Nakon što su poznate ovisnosti pogleda i sinonima o relacijama, moguće je odrediti i ovisnosti pohranjenih procedura o njima. Da bi se moglo zaključiti koja pohranjena procedura koristi koje objekte u bazi podataka, potrebno je poznavati njezin programski kod. Parsiranjem programskog koda pohranjene procedure dobije se popis svih objekata koje ona koristi – relacija, sinonima, pogleda i drugih pohranjenih procedura, te je korištenjem do sada poznatih tranzitivnih ovisnosti moguće doći do konačnog popisa relacija kojih se dotiče korištenjem te procedure.

Algoritam *PROCDEPEND* opisuje određivanje ovisnosti pohranjenih procedura o relacijama:

Algoritam 7.6: *PROCDEPEND*

procedura **procdepend** (ulazni argument **SBP**: shema baze podataka /*radne*/)
    varijabla
        p: pohranjena procedura
        sp: podaci o proceduri dobiveni parsiranjem
        n: n-torka
        r: relacija
        rs: skup relacija
        o: objekt
        os: skup objekata
    za svaku proceduru **p** ∈ **SBP**
        sp ← parsiraj proceduru **p**
        os ← objekti o kojima ovisi procedura **sp**
        za svaki **o** ∈ **os**
            rs ← relacije o kojima ovisi objekt **o**
            za svaku **r** ∈ **rs**
                n[ImeObjekta] ← ime procedure **p**
                n[ImeRelacije] ← ime relacije **r**
                ako **n** ∉ $r_{TRANZIT}$ tada
                    upisi n-torku **n** u relaciju $r_{TRANZIT}$ ❐

Još jedan tip objekata u bazi podataka koji utječu na međusobnu ovisnost relacija su okidači. Okidač kojeg pokreće akcija nad relacijom r_1 može uzrokovati obavljanje druge akcije nad relacijom r_2. Korisnik koji izvodi operaciju nad prvom relacijom

uopće ne mora imati nikakve dozvole nad drugom relacijom da bi se postupak izvršio, te su okidači također vrlo važni u ovom kontekstu. Okidači zapravo određuju ovisnost relacija jednih o drugima. Kako bi se uspješno moglo zaključiti na koji način su relacije ovisne jedne o drugima također je potrebno parsirati njihov programski kod. Algoritam koji opisuje ovaj postupak vrlo je sličan prethodnima:

Algoritam 7.7: *TABDEPEND*

procedura **tabdepend** (ulazni argument **SBP**: shema baze podataka /*radne*/)
    varijabla
        t: okidač
        st: podaci o okidaču dobiveni parsiranjem
        n: n-torka
        $\mathbf{r_1}$, $\mathbf{r_2}$: relacija
        rs: skup relacija
        o: objekt
        os: skup objekata
    za svaku relaciju $\mathbf{r_1} \in$ **SBP**
        za svaki okidač **t** nad $\mathbf{r_1}$
            st ← parsiraj okidač **t**
            os ← objekte o kojima ovisi okidač **st**
            za svaki **o** ∈ **os**
                rs ← relacije o kojima ovisi objekt **o**
                za svaku $\mathbf{r_2} \in$ **rs**
                    n[ImeObjekta] ← ime relacije $\mathbf{r_1}$
                    n[ImeRelacije] ← ime relacije $\mathbf{r_2}$
                    ako **n** $\notin r_{TRANZIT}$ tada
                        upisi n-torku **n** u relaciju $r_{TRANZIT}$ ❐

Slično kao što je to situacija s pogledima, tako i pohranjene procedure mogu koristiti druge pohranjene procedure, te se može dogoditi situacija da procedura o kojoj tranzitivno ovisi druga još uvijek nije u potpunosti obrađena tako da se ne može dobiti podatke o relacijama o kojima ona ovisi. Identično je i s okidačima, kroz koje se zaključuje ovisnost relacija o drugim relacijama. U ovim je slučajevima također potrebno ponavljati opisane algoritme na sličan način kao što je to slučaj s pogledima, dok se sadržaj relacije $r_{TRANZIT}$ mijenja. Ipak, algoritmi *PROCDEPEND* i *TABDEPEND* su povezani. Ukoliko okidač nad relacijom r_1 ovisi o nekoj pohranjenoj proceduri koja mijenja relaciju r_2, tada su bitne i ovisnosti relacije r_2, koje možda u tom trenutku nisu poznate. Zbog toga je potrebno zajedno ponavljati ova dva postupka dok god se mijenja sadržaj relacije $r_{TRANZIT}$:

Algoritam 7.8: *PROCTABDEPENDTOTAL*

procedura **proctabdependtotal** (ulazni argument **SBP**: shema baze podataka)

varijabla

ts₁, ts₂: relacija

ponavljaj

ts₁ ← $r_{TRANZIT}$

procdepend (**SBP**)

tabdepend (**SBP**)

ts₂ ← $r_{TRANZIT}$

dok je **ts₁** ≠ **ts₂** □

Nakon što su poznate ovisnosti relacija jednih o drugima, što se može postići tek uz obradu okidača nad relacijama, lako se može shvatiti kako u relaciji ovisnosti opet može do nedostatka zapisa. Naime, nakon što su na definirane ovisnosti pogleda i sinonima o relacijama, dolazi do definiranja ovisnosti tih relacija o drugima, te bi relaciju trebalo proširiti s novim zapisima – tranzitivnim ovisnostima pogleda i sinonima o relacijama. Ova proširenja tada i dalje ne bi bila konačna jer se opet širi baza ovisnih objekata, te bi opet trebalo proširiti i tranzitivne ovisnosti pohranjenih procedura i okidača o relacijama, i tako dalje, u krug. Zbog toga bi umjesto algoritma *PROCTABDEPENDTOTAL* koji je neadekvatan, bilo potrebno obaviti algoritam koji će ponavljati sve postupke dok god se sadržaj relacije $r_{TRANZIT}$ mijenja. Kako bi se smanjio broj iteracija algoritma, prije samog ponavljanja se obavi definiranje svih poznatih ovisnosti pogleda i sinonima:

Algoritam 7.9: *DEPENDTOTAL*

procedura **dependtotal** (ulazni argument **SBP**: shema baze podataka /*radne*/)

varijabla

ts₁, ts₂: relacija

viewdependtotal (**SBP**)

syndepend (**SBP**)

ponavljaj

ts₁ ← $r_{TRANZIT}$

procdepend (**SBP**)

tabdepend (**SBP**)

viewdepend (**SBP**)

syndepend (**SBP**)

ts₂ ← $r_{TRANZIT}$

dok je **ts₁** ≠ **ts₂** □

U sklopu implementacije algoritama *PROCDEPEND* i *TABDEPEND* najzanimljiviji su postupci parsiranja programskog koda pohranjenih procedura i okidača. Za određivanje forme okidača koristi se već prikazani automat (slika 6.4 i tablica 6.3). Akcija ovog automata označena sa *saveAction* zadužena je spremanje podataka o pojedinoj akciji okidača. Sama akcija okidača može biti jedna od tri SQL naredbe za izmjenu podataka ili izvršavanje pohranjene procedure, te se lako može doznati koji objekti sudjeluju u ovom okidaču. Proširenje se odnosi samo na podatke o okidaču, i to s popisom objekata koje pojedini okidač koristi.

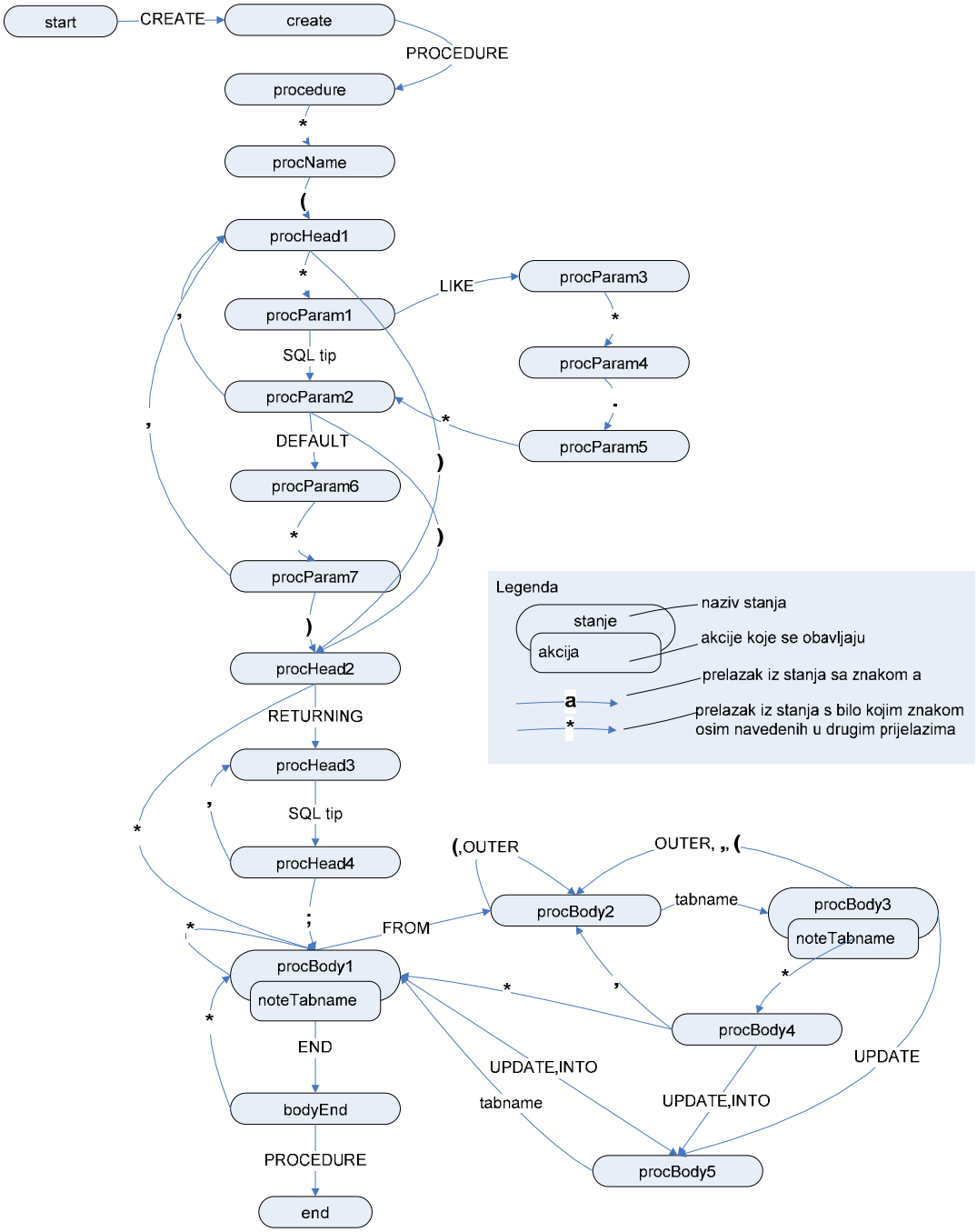
Određivanje ovisnosti pohranjenih procedura obavlja se uz pomoć parsera njihova programskog koda i automata za određivanje strukture pohranjene procedure. I ovaj automat je implementiran kao Mealyjev automat, i prikazan je na slici 7.5. Radi jednostavnosti prikaza, nisu uzeta u obzir stanja potrebna za uklanjanje programskih komentara iz koda pohranjene procedure.

Samo je jedna akcija koju obavlja automat za pohranjene procedure (sa slike 7.5), koja se potencijalno poziva iz dva stanja:

oznaka akcije	opis akcije
noteTabname	zabilježi ime relacije koju ova procedura koristi

Tablica 7.1. Akcije automata za pohranjene procedure

Automat i parser za pohranjene procedure su implementirani u razredima ProcFSA i SQLSPLParser, prikazanima na slici 6.3, dok je obavljanje ostalih algoritama iz ovog poglavlja pridijeljeno metodama razreda DependencyMaintenance.



Slika 7.5. Automat za određivanje ovisnosti pohranjene procedure o relacijama

7.3.2. Usporedba obavljenih operacija s deklariranim dozvolama

Poznavanje svih tranzitivnih dozvola nad relacijama koje korisnici implicitno dobivaju preko dozvola na druge objekte preduvjet je za mogućnost provjere je li korisnik svojom uporabom baze podataka izišao izvan tih deklariranih granica.

Provjera se obavlja periodički, a interval između provjera ne bi trebao biti dulji od jednog dana, kako bi administrator mogao pravovremeno reagirati. Prije usporedbe potrebno je iz povijesnih relacija dohvatiti sve korisnike koji su u periodu između dvije provjere nešto radili nad podacima. Nakon toga se za svakog od korisnika dohvate dozvole koje deklarativno treba imati, te se od njih, koristeći popis tranzitivnih dozvola iz relacije $r_{TRANZIT}$, dobije popis relacija koje bi smio mijenjati. U konačnici se za svakog korisnika usporedi popis relacija koje smije mijenjati s relacijama koje je mijenjao u promatranom periodu, te se alarmira administrator sustava ukoliko postoje oni koji su mijenjali relacije koje prema deklariranim dozvolama ne bi smjeli.

Pri provjeri se obavljene operacije uspoređuju s deklariranim dozvolama korisnika, umjesto sa stvarnima, koje korisnik trenutno posjeduje, a koje se također može dobiti upitima u rječnik podataka svake baze podataka. Ukoliko dođe do kompromitiranja baze podataka od strane uljeza, jedna od najčešćih posljedica je pridjeljivanje većih dozvola korisnicima nego što bi trebali imati, što se postiže relativno jednostavnim SQL naredbama. Izmjene u proširenom rječniku podataka koje bi utjecale na deklarativne dozvole korisnika zahtijevaju od uljeza puno bolje poznavanje semantike samih podataka u bazi i rada sustava. Ukoliko uljez dolazi unutar organizacije i možda poznaje semantiku proširenja rječnika podataka vezano uz deklarativne dozvole, može učiniti promjenu i nad tim podacima, no u ovom slučaju treba imati na umu da su deklarativne dozvole transparentne i vidljive raznim operaterima baze podataka za razliku od stvarnih dozvola u bazi koje su vidljive samo administratoru sustava.

Postupak provjere postojanja dozvola za operacije koje su obavljene opisan je algoritmom *CHECKUSAGE*:

Algoritam 7.10: *CHECKUSAGE*

procedura **checkusage** (ulazni argument **time**: vrijeme posljednje provjere)

varijabla

n, **n'**: n-torka

ns, **ns'**: skup n-torki

_vr: pogled za pregled povijesne relacije r

pbp: povijesna baza podataka

user: korisničko ime

users: skup korisničkih imena

rs: skup imena relacija

users ← ∅

za svaki **_vr** ∈ **pbp**

users ← **users** ∪

{**n**[OpUser] ∈ **_vr** | **n**[OpTimestamp] ≥ **time** ∧ **n**[OpUser] ∉ **users**}

za svaki **user** ∈ **users**

ns ← deklarativne dozvole za korisnika **user**

ns' ←∅

za svaki **n** ∈ **ns**

ns' ← **ns'** ∪ {**n'**[ImeRelacije] ∈ $r_{TRANZIT}$ | **n'**[ImeObjekta] = **n**[ImeObjekta]}

/* **ns'** sada sadrzi popis svih relacija koje **user** smije mijenjati */

rs ←∅

za svaki **_vr** ∈ **pbp**

ako postoji {**n** ∈ **_vr** | **n**[OpTimestamp] ≥ **time** ∧ **n**[OpUser] = **user**} tada

rs ← **rs** ∪ (ime pripadne radne relacije r za pogled **_vr**)

/* **rs** sada sadrzi popis svih relacija koje je **user** mijenjao nakon **time** */

ako postoji {**n'** ∈ **ns'** | **n'**[ImeRelacije] ∉ **rs**} tada

upozori administratora □

Provjera opisana gornjim algoritmom implementirana je u razredu DataUsageAlarm, prikazanom na slici 6.3.

7.3.3. Mogućnosti proširenja

Premda je provjera korisničke djelatnosti relativno složena, prikladna implementacija vrlo je važna sa stajališta sigurnosti informacijskog sustava, te su ovdje navedena neka moguća proširenja koja bi pridonijela funkcionalnosti.

Vrlo važno proširenje provjere odnosi se na poznavanje operacije koju je korisnik izvršio, odnosno za koju ima dozvolu. Uključivanje ovog podatka u provjeru dodatno bi postrožilo istu i ne bi se moglo dogoditi da korisnik koji ima dopuštenje samo za unos podataka u relaciju neprimijećeno napravi brisanje iz iste. U ovom slučaju bilo bi potrebno proširiti relacijsku shemu $R_{TRANZIT}$ atributima za definiranje skupa operacija nad relacijom koji podrazumijeva izvođenje neke operacije nad promatranim objektom, na sljedeći način:

$R_{TRANZIT}$ = ImeObjekta, OperacijaObjekt, ImeRelacije, OperacijeRelacija

gdje su novi atributi:

- OperacijaObjekt – operacija koja se izvršava nad objektom ImeObjekta, a koja u konačnici implicira dozvole za operacije OperacijeRelacija nad relacijom ImeRelacije
- OperacijeRelacija – popis operacija nad relacijom ImeRelacija koje implicira dozvola koju n-torka opisuje.

Pri tom domene novih atributa trebaju biti:

DOM(OperacijaObjekt) = {'S', 'U', 'I', 'D', 'E'},

DOM(OperacijeRelacija) = {'S', 'U', 'I', 'D', 'SU', 'SI', 'SD', 'UI', 'UD', 'ID', 'SUI', 'SUD', 'SDI', 'UID', 'SUID'}[1].

Iako je operacija koja je obavljena nad n-torkom relacije zapisana u pripadnoj povijesnoj relaciji, te postoje točni opisi dopuštenih operacija u definiciji dozvola u proširenom rječniku podataka, ovo proširenje je relativno složeno za implementaciju. Proces bi zahtijevao točno zaključivanje što svaka pohranjena procedura odnosno okidač izvršava nad svakom pojedinom relacijom (kojoj, između ostaloga, može pristupiti i kroz pogled i/ili sinonim), što bi znatno usložilo opisane postupke za popunjavanje sadržaja relacije $r_{TRANZIT}$. Pri tom treba uzeti u obzir i način korištenja objekta kroz kojeg korisnik tranzitivno može pristupiti relaciji, te za svaku operaciju koju je moguće izvršiti nad objektom istražiti moguće uporabe relacija.

Druga mogućnost proširenja odnosi se na postupke popunjavanja sadržaja relacije $r_{TRANZIT}$. Umjesto popunjavanja cjelokupnog sadržaja relacije, učinkovitije bi bilo održavati njezin sadržaj pri izmjeni objekata u bazi podataka. Ovo je proširenje također implementacijski vrlo zahtjevno, jer treba uzeti u obzir utjecaj svakog izmijenjenog objekta na sadržaj relacije. Na primjer, uklanjanje postojećeg okidača nad relacijom treba utjecati na sve objekte koji su preko te relacije mogli tranzitivno utjecati na druge relacije. U ovom bi slučaju za svaku tranzitivnu vezu među dozvolama bilo potrebno bilježiti i sve objekte preko kojih ta tranzitivna veza postoji. Kako bi se sadržaj relacije mogao efikasno održavati, također bilo bi potrebno uspoređivati trenutno i novo stanje objekata u bazi podataka koji se mijenjaju pri izmjeni verzije baze podataka.

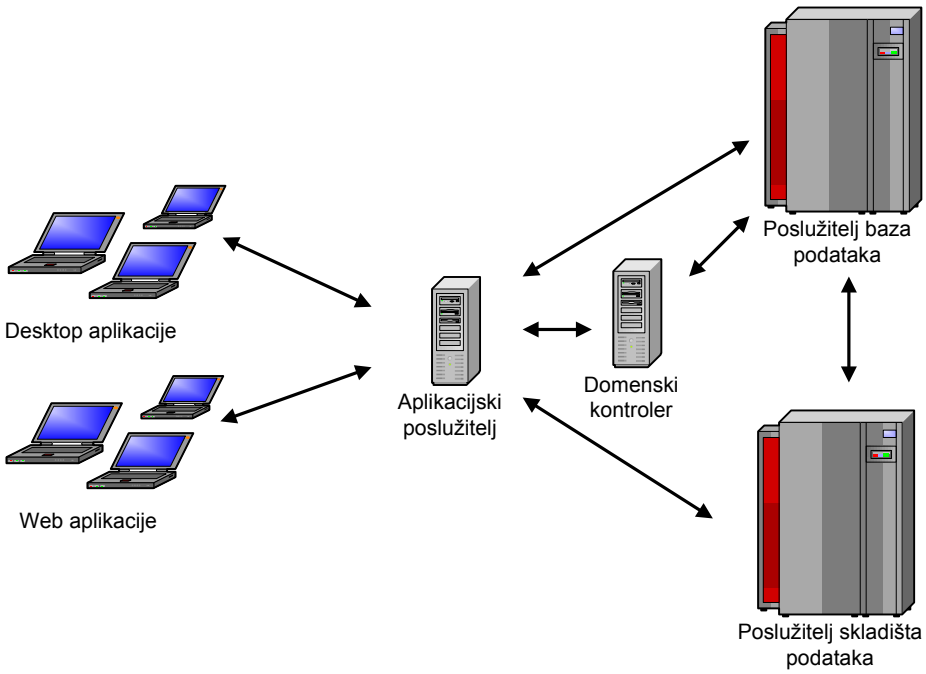
Ovaj postupak bi bilo moguće donekle automatizirati ukoliko bi sustav za upravljanje bazama podataka omogućivao postojanje sistemskih okidača. Sistemski okidači su objekti baze podataka slični običnim okidačima nad relacijama, no okidaju se na događaje izmjene objekata u bazama podataka. Adekvatnim programiranjem sistemskih okidača, uz poznavanje pojedinosti o svim tranzitivnim vezama bilo bi moguće programski zaključiti kako izmjena pojedinog objekta utječe na dozvole te na taj način održavati relaciju $r_{TRANZIT}$. Trenutno samo neki od dostupnih komercijalnih sustava za upravljanje bazama podataka omogućuju postojanje sistemskih okidača.

[1] I u ovom slučaju oznake S, U, I i D označavaju osnovne operacije nad podacima (dohvat, izmjena, unos, brisanje), nastale prema akronimima pripadnih SQL naredbi. Oznaka E obilježava dozvolu za izvršavanje (EXECUTE) pohranjene procedure.

8. Prikaz primjene snimanja traga o izmjenama podataka

Opisani sustav za snimanje traga unutar baze podataka razvijen je paralelno s razvojem Informacijskog sustava visokih učilišta (ISVU, [Baranović2003a]) na Zavodu za primijenjenu matematiku (danas Zavod za primijenjeno računarstvo), Fakulteta elektrotehnike i računarstva, te je tako automatski postao njegovim dijelom. ISVU je informacijski sustav namijenjen podršci administrativnim poslovima i procesima koji se odvijaju na visokoškolskim ustanovama. Korisničku i administratorsku programsku potporu čini više od deset programskih modula, kako samostojećih (eng. *desktop*), tako i internetskih (eng. *web*). Korisnici sustava su nekoliko tisuća zaposlenika visokih učilišta – nastavnog i administrativnog osoblja, te više od 100 tisuća aktivnih studenata.

Središte informacijskog sustava je radna baza podataka. Sustav za upravljanje bazama podataka koji upravlja ovom bazom je IBM Informix, i smješten je na za to određenom poslužitelju. Korisnici internetskih modula pristupaju bazi podataka preko aplikacijskog poslužitelja na kojem se izvodi za to predviđena poslužiteljska aplikacija. Korisnici samostojećih modula također pristupaju bazi podataka preko aplikacijskog poslužitelja, ali se komunikacija odvija preko tunelirajućeg modula za enkripciju/dekripciju podataka, s obzirom da su podaci koji putuju preko mreže kriptirani. Aplikacijski poslužitelj je također pristupna točka za ISVU skladište podataka [Baranović2003b], koje je smješteno unutar izdvojenog SUBP-a na poslužitelju skladišta podataka. Korisnici se za pristup sustavu autenticiraju uporabom imenika smještenog na izdvojenom poslužitelju – domenskom kontroleru. Opisani poslužitelji i komunikacijski kanali prikazani su na slici 8.1.



Slika 8.1. Poslužitelji i komunikacije u ISVU

Radna baza podataka ISVU sustava sastoji se od 450 relacija s podacima, te oko 350 pripadnih povijesnih relacija. Povijesna baza podataka smještena je unutar istog SUBP-a na istom poslužitelju. U povijesnoj bazi podataka je također 350 povijesnih relacija. Razna poslovna pravila su u radnoj bazi podataka implementirana pomoću 1700 pohranjenih procedura i 360 okidača. Nakon primjene algoritma za proširenje okidača, broj okidača u radnoj bazi podataka je 1080. Primjenom algoritma za stvaranje okidača za zaštitu povijesnih relacija u povijesnoj bazi se stvara još tri puta onoliko okidača koliko ima povijesnih relacija.

Povijest se u radnoj bazi podataka snima od svibnja 2001. Od tada do kolovoza 2008. godine, snimljeno je ukupno više od 100 milijuna n-torki koje predstavljaju isto toliko operacija nad podacima. Snimljene operacije izvedene su unutar 20 milijuna transakcija. Povijesne relacije s najvećim brojem operacija sadrže po više milijuna zapisa.

Proces *backupHistory* izvršava se na aplikacijskom poslužitelju, dok su radna i povijesna baza podataka smještene na poslužitelju baza podataka. Proces se pokreće jednom svake noći te obavlja prebacivanje podataka iz radne u povijesnu bazu podataka, koje u prosjeku obuhvaća 100 tisuća n-torki. Prosječno trajanje izvođenja procesa je 3 minute.

Stvoreno je 350 pogleda za pregled cjelokupne povijesti relacija. U radnoj bazi podataka smješteno je 670 redukcijskih pogleda kojima se ograničava pristup korisnicima, te su za svakog od korisnika prema opisanim dozvolama, stvoreni privatni sinonimi koji pokazuju na te poglede. Ukupan broj privatnih sinonima u radnoj bazi podataka je veći od 730 tisuća.

Primjenom algoritma za izračun tranzitivnih ovisnosti dozvola, relacija $r_{TRANZIT}$ napuni se sa oko 4 milijuna zapisa. Analiza uporabe podataka i pripadnih opisanih dozvola se pokreće svake noći, uz prosječno trajanje izvođenja od oko 20 minuta.

Osim snimanja traga unutar baze podataka, u informacijskom sustavu je implementirano i snimanje traga na razini SUBP-a. Među događajima koji se prate su oni koji su zanimljivi za kombiniranje sa snimanjem traga unutar baze podataka: spajanje na baze podataka, stvaranje i uništavanje relacija, pogleda i okidača, te izmjena stanja aktivnosti okidača.

Tijekom proteklih godina funkcioniranja sustava, djelatnici zaposleni na administriranju sustava i podršci korisnicima, više su stotina puta koristili povijesne relacije u cilju forenzičkih ispitivanja povijesti podatka, bilo da se radilo o zahtjevu koji potiče od korisnika sustava, ili o provjeri funkcioniranja složenih poslovnih pravila implementiranih u sustavu. Bez snimljene povijesti podataka ovi bi zadaci bili nemogući. Dodatno snimanje traga na razini SUBP-a ne pokazuje zlouporabe modela snimanja traga unutar baze podataka.

Opisani način snimanja traga u bazama podataka također je implementiran unutar Informacijskog sustava održavanja u Hrvatskoj elektroprivredi (ISOHEP, [Hebe11998]). U ovom slučaju su objekti za snimanje traga naknadno uvedeni u sustav, te se snimanje obavlja od svibnja 2006. godine. Sustav se sastoji od četiri radne baze podataka, sve s istim shemama, dok su podaci distribuirani. Ne postoje

povijesne baze podataka, već se cjelokupna povijest čuva unutar povijesnih relacija u radnim bazama podataka.

Svaka od radnih baza podataka sadrži preko 450 relacija, od kojih se povijest snima za njih 408. Iz snimanja povijesti su izuzete relacije s velikim brojem zapisa i velikim brojem izmjena. Nad ostalim relacijama je do travnja 2008. obavljeno preko 450 tisuća operacija koje su zabilježene u povijesnim relacijama radnih baza podataka.

Osim navedenih, opisani način snimanja traga implementiran je i unutar sustava Mozvag, informacijskog sustava za podršku postupku vrednovanja studijskih programa [Matijašević2007]. Korisnici sustava su zaposlenici svih visokih učilišta u Republici Hrvatskoj, a koja predlažu izvođenje studijskih programa. U postupku vrednovanja istih aktivni su domaći i inozemni recenzenti, Agencija za znanost i visoko obrazovanje (AZVO), Nacionalno vijeće za visoko obrazovanje (NVVO), te Ministarstvo znanosti, obrazovanja i športa (MZOŠ). Osim jednostavnih tipova podataka, sustav služi i za razmjenu i pohranu velikog broja dokumenata koji su dio postupka.

Snimanje traga u ovom sustavu započelo je odmah pri početku rada, od ožujka 2005. godine. Od 140 relacija u radnoj bazi podataka, povijest se snima za njih 63. Povijesne relacije smještene su osim u radnoj i u izdvojenoj povijesnoj bazi podataka. U njima je do kolovoza 2008. godine evidentirano više od 860 tisuća operacija. Sami dokumenti koji su dio postupka su pohranjeni kao atributi relacija, kako radnih tako i povijesnih. Aktualni dokumenti, oni u radnoj bazi podataka, zauzimaju oko 12 GB prostora za pohranu, dok svi oni u povijesnoj bazi podataka zauzimaju oko 20 GB prostora za pohranu. Premda se može dobiti dojam da je ova količina podataka nepotrebno prisutna, dokumenti su vrlo važni za aktere procesa, te je mogućnost dohvata starijih verzija dokumenata znatno vrjednija.

Trajnim snimanjem traga i uporabom tog traga u potrebitim analizama u tri navedena informacijska sustava pokazuje se da je ono zasnovano na ispravnim pretpostavkama i implementirano na prikladan način.

9. Zaključak

Zadatak rada bio je istražiti načine snimanja traga u relacijskim bazama podataka, odrediti komponente sustava za upravljanje relacijskim bazama podataka čiji je rad potrebno snimati, istražiti mogućnosti postojećih rješenja za snimanje traga, te predložiti prikladno rješenje snimanja traga korisničke djelatnosti na razini baze podataka u cilju forenzičnih i sigurnosnih rješenja. Predloženo rješenje bilo je potrebno isprobati u radu informacijskog sustava, te prezentirati podatke o funkcioniranju rješenja.

U okviru zadatka identificirane su i potrebe za snimanjem traga. Naime, iako se postojanje snimljenog traga može opravdati već time što se njegovom analizom može potvrditi ili opovrgnuti ispravnost sigurnosne politike informacijskog sustava, postoje i zakonske norme koje obvezuju tvrtke za provođenjem snimanja traga. Norme pokušavaju nametnuti ova pravila kako bi spriječile razne, u prvom redu financijske, pronevjere, a dobar dio njih iniciran je i iz sve veće svjesnosti o potrebi zaštite osobnih podataka, koji se skupljanju u velikom broju informacijskih sustava.

Način na koji se dolazi do informacije koja će postati snimljeni trag identificiran je kao glavni kriterij za razlučivanje metoda snimanja traga u radu sustava za upravljanje bazama podataka. U skladu s time, određene su četiri metode: snimanje traga unutar klijentske aplikacije, unutar baze podataka, unutar sustava za upravljanje bazama podataka, ili uopće izvan sustava za upravljanje bazama podataka. Iako svaki od prezentiranih načina snimanja traga ima svoje prednosti i nedostatke, općenito je snimanje traga koje započinje unutar klijentske aplikacije najmanje prikladno, dok je snimanje traga izvan SUBP-a, vanjskim sustavom za snimanje traga najprikladnije i najsigurnije rješenje, no ujedno i financijski najzahtjevnije. Snimanje traga na razini SUBP-a uvelike ovisi o mogućnostima konkretnog SUBP-a, i često nije dovoljno prilagodljivo. Ukoliko se, pak, obavlja snimanje traga na razini baze podataka, potrebno je prilagođavati okidače unutar te baze podataka. Uz to, ti okidači moraju stalno biti aktivni, kako bi se izbjegle kompromitirajuće situacije. Na primjer, administrator sustava može onemogućiti okidače dok izvršava određene izmjene zapisa u relaciji, te će se na taj način dogoditi izmjene koje neće biti zapisane u snimljeni trag.

Identificirane su sljedeće aktivnosti korisnika sustava za upravljanje bazama podataka koje je potrebno nadzirati: prijave i odjave korisnika za rad s bazom podataka, rad korisnika u odnosu na vremenske obrasce rada, postojanje podataka o klijentskim aplikacijama kojima se korisnici spajaju na bazu podataka, izvršavanje DDL naredbi za manipulaciju objektima u bazi podataka, prijave pogrešaka u radu SUBP-a, izmjene programskog kôda onih objekata baze podataka koji sadrže programski kôd, izmjene podataka o korisnicima sustava i njihovim dozvolama, postojanje i rad s korisničkim sinonimima i repliciranim bazama podataka, izvršavanje DML naredbi za manipulaciju podacima, izvršavanje čitanja nekih specifičnih podataka, te izmjene nad samim definicijama snimanja traga. Za svaku od navedenih aktivnosti raspravljeni su i najprikladniji od prikazanih načina snimanja traga.

U cilju snimanja korisničke djelatnosti na razini baze podataka, predloženo je rješenje snimanja traga koje nastaje unutar baze podataka, a koje se temelji na okidačima nad relacijama. Trag se snima za svaku relaciju posebno i sprema se unutar komplementarne relacije koja osim identičnih atributa kao i radna, sadrži i dodatne, koji opisuju pojedinu snimljenu operaciju. Cjelokupna povijest se periodički prebacuje u drugu, povijesnu bazu podataka koja sadrži povijesne relacija istih shema. Prikazani su algoritmi stvaranja povijesne baze podataka, povijesnih relacija u radnoj i povijesnoj bazi podataka, prilagodbe okidača u cilju snimanja povijesti, prebacivanja povijesnih podataka iz radne u povijesnu bazu, te je opisana i programska implementacija. Definiran je način ubrzavanja pristupa snimljenom tragu i problematika dozvola korisnika sustava s aspekta snimanja traga.

Korištenje snimljenog traga prikazano je stvaranjem pogleda za pregled cjelokupne povijesti relacije, stvaranjem izvještaja o djelatnosti korisnika u zadanom vremenskom periodu, te izvođenjem kontrole stvarnih i deklariranih dozvola korisnika sustava, a koja se temelji na drugim proširenjima radne baze podataka opisanim u referenciranim radovima.

Opisano rješenje ugrađeno je u tri nezavisna informacijska sustava. Prikazani su neki pokazatelji funkcioniranja ovih sustava važni za djelovanje navedenih algoritama za uspostavu snimanja i za analizu snimljenog traga u tim sustavima. Iskustva administrativnog osoblja zaduženog za održavanje sustava u odnosu na mehanizme snimanja i na mogućnosti iskorištavanja snimljenog traga su pozitivna.

Slijedom prikazanog uspješnog djelovanja u više informacijskih sustava može se zaključiti kako je predloženo rješenje, temeljeno na prethodnom istraživanju, adekvatno za snimanje traga unutar baza podataka.

Literatura

[Alawi2006] A. I. Al-Alawi, E. A. Hafedh. Auditing of Information Privacy. Information Technology Journal 5(1). 2006.

[Ambeo2006] Ambeo. Ambeo Data Privacy Brochure. (http://www.ambeo.com/acrobat/AmbeoBrochureDataPrivacyBrochureFINAL061705.pdf [1.8.2008]). Colorado Springs, Colorado. 2006.

[Anzil2005] J. Anzil. Upravljanje pravilima pristupa podacima temeljenim na ulogama u složenim informacijskim sustavima. Magistarski rad. Zagreb. 2005.

[Apexsql2007] ApexSQL LLC. ApexSQL Audit Datasheet. (http://www.apexsql.com/datasheets/ApexSQL_audit_datasheet.pdf [1.8.2008]). Chapel Hill, North Carolina. 2007.

[Baranović2001] M. Baranović, S. Zakošek, L. Mačkala. Creating database-aware java classes based on extended data dictionary and abstract classes. Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics, Vol. 7. Orlando, Florida. 2001.

[Baranović2003a] M. Baranović, M. Borčić, D. Hunjet i ostali. Informacijski sustav visokih učilišta. Ministarstvo znanosti i tehnologije. Zagreb. 2003.

[Baranović2003b] M. Baranović, M. Madunić, I. Mekterović. Data Warehouse For High Education Information Systems. Proceedings of the 25th International Conference on Information Technology Interfaces. Zagreb. 2003.

[Berg1975] J. L. Berg. Data Base Directions, The Next Steps. Proceedings of the Workshop of the National Buerau of Standards and Association for Computing Machinery. Fort Lauderdale, Florida. 1975.

[Beauchemin2007] B.Beauchemin. SQL Server 2005 Security Best Practices – Operational and Administrative Tasks, Microsoft White paper. 2007.

[CE1981] Council of Europe. Convention for the Protection of the Individuals with regard to Automatic Processing of Personal Data. http://conventions.coe.int/Treaty/EN/Treaties/Html/108.htm [1.11.2007]. 1981.

[Codd1970] E. Codd: *A Relational Model for Large Shared Data Banks*, Communication of the ACM. 1970.

[Congress1996] United States Congress. Health Insurance Portability and Accountability Act.

http://www.cms.hhs.gov/HIPAAGenInfo/Downloads/HIPAALaw.pdf [1.8.2008]. 1996.

[Congress1999] United States Congress. Gramm-Leach-Bliley Act. http://www.ftc.gov/privacy/glbact [1.8.2008]. 1999.

[Congress2002] United States Congress. Sarbanes-Oxley Act. http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_bills&docid=f:h3763enr.tst.pdf [1.8.2008]. 2002.

[CS2002] California Senate. Bill 1386. http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html [1.8.2008]. 2002.

[Date2000] C. Date. An introduction to database systems. Addison-Wesley Publishing Company. Reading, Massachusetts. 2000.

[Dayal1995] U. Dayal, E. Hanson, J. Widom. Active database system. In Modern database systems, the object model, interoperability, and beyond. Won Kim, ed. Addison-Wesley Publishing Company. Reading, Massachusetts. 1995.

[EP1995] The European Parliament and the Council of the European Union. Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data. http://europa.eu.int/eur-lex/lex/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML [1.8.2008]. 1995.

[EP2002a] The European Parliament and the Council of the European Union. Directive 2002/22/EC on universal service and users' rights relating to electronic communications networks and services (Unversal Service Directive). http://eur-lex.europa.eu/pri/en/oj/dat/2002/l_108/l_10820020424en00510077.pdf [1.8.2008]. 2002.

[EP2002b] The European Parliament and the Council of the European Union. Directive 2002/58/EC concerning the processing personal data and the protection of privacy in the electronic communications sector. http://eur-lex.europa.eu/pri/en/oj/dat/2002/l_201/l_20120020731en00370047.pdf [1.8.2008]. 2002.

[Ferraiolo2003] D. F. Ferraiolo, D. R. Kuhn, R. Chandramouli. Role-Based Access Control. Artech House. Boston. 2003.

[Hebel1998] Z. Hebel, M. Baranović, S. Zakošek. ISOHEP - Computerised maintenance management system for Croatian National Electricity (HEP). Proceedings of the Euromaintenance '98. HDO. Zagreb. 1998.

[Hibernate2008] Red Hat Inc. Hibernate. http://www.hibernate.org/ [1.8.2008]. 2008.

[IBM2006] International Business Machines Corporation. IBM DB2 Version 9, Administration Guide: Implementation. 2006.

[IBM2007] International Business Machines Corporation. IBM Informix Security Guide, Version 11.10. 2007.

[Jajodia1995] S. Jajodia, S. Gadia, G. Bhargava. Logical design of audit information in relational databases. Information Security: An Integrated Collection of Essays. IEEE Computer Society Press. Los Alamitos, California. 1995.

[Lin1994] T. Y. Lin. Anomaly Detection, A Soft Computing Approach. Proceedings of the 1994 workshop on New security paradigms. Little Compton, Rhode Island. 1994.

[Litchfield2005] D. Litchfield, C. Anley, J. Heasman, B. Grindlay. The Database Hacker's Handbook: Defending Database Servers. Wiley Publishing Inc. Indianapolis, Indiana. 2005.

[Lumigent2007] Lumigent Technologies Inc. Lumigent Audit DB 6 Technical Datasheet. (http://www.lumigent.com/resources/AuditDB_6_technical_DS.pdf [1.8.2008]). Acton, Massachusetts. 2007.

[Lunt1993] T. F. Lunt. A survey of intrusion detection techniques. Computers and Security 12 (4). Elsevier Advanced Technology Publications. Oxford, UK. 1993.

[Mačkala2004] L. Mačkala. Objektno orijentirani pristup izradi sučelja s bazom podataka. Magistarski rad. Zagreb. 2004.

[Matijašević2007] B. Matijašević, H. Rončević, O. Orel. Agile Software Development Supporting Higher Education Reform. Proceedings of the 29th International Conference on Information Technology Interfaces. Zagreb. 2007.

[Mazer2005] M. Mazer. Database Auditing – Essential Business Practice for Today's Risk Management. http://www.theiia.org/ITAudit/index.cfm?act=itaudit.archive&fid=5617 [8.8.2007]. Institute of Internal Auditors. 2005.

[Mercuri2003] R. T. Mercuri. On Auditing Audit Trails. Communications of the ACM Vol. 46, No 1. ACM. New York, NewYork. 2003.

[Natan2005] R. B. Natan. Implementing Database Security and Auditing. Elsevier Inc. Burlington, Massachusetts. 2005.

[NCSC1987] National Computer Security Center. A Guide to Undestanding Audit in Trusted Systems. Fort George G. Meade, Maryland. 1987.

[Oracle2007a] Oracle Inc. Oracle Audit Vault Administrator's Guide, 10g Release. Redwood City, California. 2007.

[Oracle2007b] Oracle Inc. Oracle Database Security Guide, 11g Release. Redwood City, California. 2007.

[Orel2001] O. Orel. Programski sustav za definiranje ekranskih formi u programskom jeziku Java. Diplomski rad. Zagreb. 2001.

[Richardson2004] K. Richardson. Keeping Data Private (and knowing it), Ambeo White Paper. Ambeo. Colorado Springs, Colorado. 2004.

[Ruebush2005] M. Ruebush. SQL Server 2005 and Oracle 10g Security Comparison, Microsoft White Paper. Microsoft. 2005.

[Sabor2001] Hrvatski sabor. Ustav Republike Hrvatske – pročišćeni tekst. Narodne novine 41/2001. Zagreb. 2001.

[Sabor2003] Hrvatski sabor. Zakon o zaštiti osobnih podataka. Narodne novine 103/2003. Zagreb. 2003.

[Sabor2005] Hrvatski sabor. Zakon o potvrđivanju konvencije za zaštitu osoba glede automatizirane obrade osobnih podataka i dodatnog protokola uz konvenciju za zaštitu osoba glede automatizirane obrade osobnih podataka u vezi nadzornih tijela i međunarodne razmjene podataka. Narodne novine, Međunarodni ugovori 4/2005. Zagreb. 2005.

[Schneier1999] B. Schneier, J.Kelsey. Secure Audit Logs to Support Computer Forensics. ACM Transactions on Information and System Security. New York, New York. 1999.

[Snodgrass2004] R. T. Snodgrass, S. S. Yao, C. Collberg. Tamper Detection in Audit Logs. Proceedings of the 30Th VLDB Conference. Toronto, Canada. 2004.

[SoftTree2005a] SoftTree Technologies Inc. DB Audit White Paper. (http://www.softtreetech.com/dbaudit/db_audit_wp.pdf [1.8.2008]). 2005.

[SoftTree2005b] SoftTree Technologies Inc. DB Audit User Guide. (http://www.softtreetech.com/dbaudit/db_audit.pdf [1.8.2008]). 2005.

[Spring2008] SpringSource Inc. Spring Framework. http://www.springframework.org/ [1.8.2008]. 2008.

[Srbljić2003] S. Srbljić. Jezični procesori 1. Element. Zagreb. 2003.

[Stonebraker1987] M. Stonebraker, J. Anton, E. Hanson. Extending a database system with procedures. ACM Transactions on Database Systems (TODS), Vol. 12, No. 3. 1987.

[Tizor2007] Tizor Systems. Mantra Overview. http://www.tizor.com/uploadDocs/Mantra_Overview_Sept2007.pdf [1.8.2008]. Tizor. Maynard, Massachusetts. 2007.

[Upscene2006] Upscene Productions. MSSQL LogManager Factsheet. (http://www.iblogmanager.com/download/MSSQLLM_Factsheet.pdf [1.8.2008]). Oss, The Netherlands. 2006.

[Vlada2004] Vlada Republike Hrvatske. Uredba o načinu pohranjivanja i posebnim mjerama tehničke zaštite posebnih kategorija osobnih podataka. Narodne novine 139/2004. Zagreb. 2004.

[Wagner2005] F. Wagner. Moore or Mealy model? http://www.stateworks.com/active/download/TN10-Moore-Or-Mealy-Model.pdf [1.8.2008]. 2005.

[Weber1998] R. Weber. Information Systems Control and Audit. Prentice Hall Inc. Upper Saddle River, New Jersey. 1998.

[Zakošek2004] S. Zakošek. Postupci upravljanja konzistentnošću i raspoloživošću u repliciranim bazama podataka. Doktorska disertacija. Zagreb. 2004.

Sažetak

NADZOR NAD RADOM KORISNIKA U RELACIJSKIM BAZAMA PODATAKA

Tijekom više desetljeća postojanja raznih informacijskih sustava, postalo je jasno da podaci koje oni sadrže mogu biti vrlo vrijedna imovina. Svaku vrijednost je potrebno zaštititi na prikladan način, pa se tako radi i s podacima. U ovom kontekstu govori se o zaštiti informacija kako od vanjskih upada, tako i od neovlaštenog korištenja informacija od strane zaposlenika tvrtke ili administratora sustava. Da bi to bilo moguće, potrebno je nadzirati rad korisnika sustava. Jedan od mehanizama potvrde ispravnosti sigurnosnih mjera jest snimanje traga (eng. *auditing*). Trag u kojem je snimljeno što je tko, kada i kako radio na sustavu nudi mogućnost provedbe analiza koje mogu potvrditi ili opovrgnuti sigurnosne protokole.

Sve veća važnost zaštite informacija dolazi od povećane svjesnosti svih korisnika informacijskih sustava o važnosti zaštite osobnih podataka. Upravo je to bila motivacija mnogih država da uspostave razne zakonske norme koje propisuju snimanje traga kao mehanizam provjere zaštićenosti podataka. Pregled ovih normi dan je u drugom poglavlju.

Obzirom da se većina informacijskih sustava bazira na bazama podataka, sustav za upravljanje bazama podataka i same baze podataka su ono mjesto na koje je potrebno usmjeriti pozornost pri nadzoru korisničke aktivnosti. Uvodne definicije pojmova vezanih u teoriju baza podataka, te razmatranja principa raspodjele dužnosti pri snimanju traga i zaštiti snimljenog traga čine treće poglavlje.

Prema načinu na koji se dolazi do informacije koja čini trag koji se može snimiti, moguće je odrediti nekoliko metoda za snimanje traga: snimanje unutar klijentske aplikacije, snimanje unutar baze podataka, snimanje unutar sustava za upravljanje bazama podataka, te snimanje izvan sustava za upravljanje bazama podataka. Načini funkcioniranja, prednosti i nedostatci ovih metoda prikazani su u četvrtom poglavlju. Također su prikazane i mogućnosti snimanja traga nekih komercijalnih sustava za upravljanje bazama podataka.

Pojedini događaji u radu sustava za upravljanje bazama podataka koji su zanimljivi za snimanje traga su obrađeni u petom poglavlju: prijava i odjava korisnika za rad, rad izvan uobičajenih obrazaca, podaci o klijentskim aplikacijama, DDL naredbe, SQL pogreške koje sustav prijavljuje korisnicima, izmjene programskog kôda objekata u bazama, izmjene podataka o korisnicima i dozvolama, korisnički sinonimi i podaci o replikaciji baza podataka, SQL naredbe za izmjenu podataka, SELECT naredbe, te izmjene definicija snimanja traga.

U šestom pogavlju detaljno je razrađena i prikazana ideja i implementacija snimanja traga izmjena podataka unutar baza podataka, metodom prilagodbe okidača. Radi se o automatiziranoj izmjeni i održavanju postojećih i dodavanju novih okidača unutar baze podataka za koju se obavlja snimanje traga. Definicija snimanja traga nalazi se unutar proširenja rječnika baze podataka. Snimljeni trag privremeno se smješta u

povijesne relacije unutar radne baze podataka, a periodički se preseljava u za to namijenjene relacije u posebnoj bazi podataka, koja može biti smještena i na izvodjenom poslužitelju. Diskutirana su moguća proširenja.

Primjena snimljenog traga koji nastaje na opisani način u forenzičkim i sigurnosnim analizama opisana je u sedmom poglavlju rada. Opisane su ideje i implementacije triju takvih analiza. Diskutirana su moguća proširenja.

Osmo poglavlje prikazuje funkcioniranje opisanog načina snimanja traga i prikazanih analiza u tri živa informacijska sustava – ISVU, ISOHEP i Mozvag, dok posljednje poglavlje donosi zaključak rada.

Summary

MONITORING OF USERS' ACTIONS IN RELATIONAL DATABASES

During the several decades of existence of various information systems, it has become obvious that the data can be a very valuable asset. Each property should be taken care of, and the data should not be different. Protecting data from the external attacker is necessary, but also from the company's employees or system administrators. To confirm that security measures are set correctly, auditing is often used. The audit trail that contains information of who, when and how worked on the system can be used to verify or decline security protocols.

Increasing importance of data protection is also influenced by increased users' awareness of how valuable personal data could be. That was the motivation for many countries to impose legislative regulations for protection of personal data. Some of these regulations are shown in the second chapter.

As most of the information systems rely on databases, the database management system and the database itself is the main focus spot to attend to while monitoring the users' activity. Introductory definitions, as well as some considerations of segregation of duties principle and audit trail protection are part of the third chapter.

Considering the origin of the audit trail, there are several methods of database auditing: auditing inside of client application, auditing inside the database, auditing inside of database management system and external auditing. Functioning, advantages and disadvantages of these methods are shown in the fourth chapter. Auditing capabilities of a several commercial database management systems are also displayed.

There are several events that should be considered while deciding what should be audited in the database management system. These events are discussed in the fifth chapter: logging in and out of the system, logging in after the working hours, client application information, DDL operations, SQL errors generated by users, changes in stored procedures and triggers, changes of user data and permissions, private synonyms, data replication, data changes, SELECT operations and audit rules definitions.

The idea and the implementation of auditing of data changes by adjusting the database triggers is shown in detail in the sixth chapter. The auditing is managed by automated changes and maintenance of existing triggers inside the audited database, as well as creation of new ones. Definition of auditing is stored in additional table in the database's extended system catalog. The audit trail is temporarily stored in history tables inside a working database, and is periodically transferred to dedicated history tables in a history database that could be set up on a different server. Possible extensions of this audit model are also discussed.

Usage of the audit trail, that is created by this audit method in a forensic and safety analysis, is described in the chapter seven. Ideas and implementations of three analysis methods are described. Possible extensions are also discussed.

Chapter eight shows the application of proposed audit method in three different production information systems – ISVU, ISOHEP and Mozvag. Chapter nine concludes the paper.

Ključne riječi

Sigurnost sustava za upravljanje bazama podataka

Snimanje traga u bazama podataka

Nadzor korisničke aktivnosti

Automatizirana izmjena objekata u bazama podataka

Analiza snimljenog traga

Keywords

Database Management System Security

Database Auditing

User Activity Monitoring

Automated changing of database objects

Audit Trail Analisys

Životopis

Ognjen Orel rođen je u 13. studenog 1976. u Sarajevu. Maturirao je 1995. godine u IX gimnaziji u Zagrebu. Na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu (FER) diplomirao je u rujnu 2001. na smjeru Računarstvo. Diplomski rad iz područja baza podataka, s temom „Programski sustav za definiranje ekranskih formi u programskom jeziku Java“ izradio je kod mentorice prof. dr. sc. Mirte Baranović.

Od rujna 2001. godine do danas zaposlen je u Sveučilišnom računskom centru Sveučilišta u Zagrebu (Srce) u Sektoru za posebne programe na radnim mjestima informatičara, informatičara specijalista, te konačno voditelja projekata. Redoviti poslovi uključuju modeliranje, izradu i održavanje informacijskih sustava temeljenih na bazama podataka. Do svibnja 2004. djeluje na razvoju Informacijskog sustava visokih učilišta (ISVU) na Zavodu za primijenjenu matematiku Fakulteta elektrotehnike i računarstva.

Od 2001. do 2004. u sklopu nastavnih aktivnosti na Fakultetu elektrotehnike i računarstva sudjelovao je u nastavi iz predmeta Uvod u baze podataka, Baze podataka i Programiranje, a tijekom 2001. i 2002. u nastavi iz predmeta Programiranje na Tehničkom veleučilištu u Zagrebu.

Područja istraživanja su mu sigurnost sustava za upravljanje bazama podataka, modeliranje informacijskih sustava, aplikacijska programska sučelja za rad s podacima te grafička korisnička sučelja. Koautor je tri znanstvena rada iz ovih područja.