UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Dejan I. Dunđerski

Sistem za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju

Doktorska disertacija

Beograd, 2021.

UNIVERSITY OF BELGRADE

SCHOOL OF ELECTRICAL ENGINEERING

Dejan I. Dundjerski

Automatic Intelligent Database Troubleshooting System in Cloud Environment

Doctoral Dissertation

Belgrade, 2021

Mentor:

Dr Milo Tomašević, redovni profesor

Univerzitet u Beogradu - Elektrotehnički fakultet

Članovi komisije:

Dr Miroslav Bojović, redovni profesor

Univerzitet u Beogradu - Elektrotehnički fakultet

Dr Siniša Vlajić, redovni profesor

Univerzitet u Beogradu - Fakultet organizacionih nauka

Dr Miloš Cvetanović, vanredni profesor

Univerzitet u Beogradu - Elektrotehnički fakultet

Dr Jelica Protić, redovni profesor

Univerzitet u Beogradu - Elektrotehnički fakultet

Datum odbrane:

_____ godine.

Zahvalnica

Prvenstveno želim da se zahvalim profesoru Milu Tomaševiću na izuzetnoj saradnji koju smo uspostavili tokom poslednjih deset godina. Profesor Tomašević me je uveo u svet nauke još kroz diplomski rad. I dan danas se sećam kako sam došao kod profesora sa idejom o paralelizaciji algoritama koji se upotrebljavaju u mrežnim protokolima na grafičkim procesorima. Time sam želeo da spojim dve oblasti koje su me izuzetno zanimale, oblast računarskih mreža i oblast paralelizacija procesa. Profesor je bio vrlo zainteresovan za predloženu ideju i pomogao mi je da je dalje uobličimo. Kada sam nedugo nakon pisanja i istraživanja shvatio da u isto vreme postoje i kolege u Južnoj Koreji koji su se bavili sličnom tematikom, to mi je teško palo i pomislio sam da moj diplomski rad više nema smisla. Zahvaljujući profesoru Tomaševiću, koji mi je strpljivo objasnio šta zapravo znači nauka, shvatio sam da činjenica što se još neko bavi određenom tematikom nije promašaj već da samo još više potkrepljuje izabrani pristup kao nešto što ima smisla istraživati dalje. Našu saradanju smo potom nastavili i tokom master studija u istoj oblasti. Kruna saradnje je bio prvi zajednički rad objavljen u časopisu sa faktorom. Zahvaljujući profesoru Milu Tomaševiću sam nekoliko godina nakon odbrane master rada i upisao doktorske studije u želji da nastavimo istraživački rad zajedno. Dragi profesore, veliko Vam hvala na posvećenom vremenu!

Ovom prilikom bih želeo i da se zahvalim profesoru Milanu Raspopoviću koji mi je takođe u nekoliko razgovora ukazao na pozitivne aspekte sa kojima će se čovek susresti tokom pohađanja doktroskih studija, istraživanja i daljeg upoznavanja nauke. Zahvaljujući smernicama profesora Raspopovića i profesora Koste Andrejevića sam objavio i naučnu monografiju. Tokom svih ovih godina i celog procesa rada na distertaciji, puno sam naučio tako da je evidentan uticaj spomenutih profesora i u ovoj doktorskoj disertaciji. Još jednom hvala vam puno na savetima, smernicama i prilici da upoznam nauku još više kroz vaše izuzetno iskustvo koje ste nesebično podelili samnom.

Želim da se zahvalim i kolegama iz kompanije Microsoft a posebno, Branku Kokanoviću, Igoru Iliću, Vukašinu Joviću, Dimitriju Filipoviću, Jovanu Ćukaloviću i Draganu Tomiću. Tokom razvoja proizvoda, na osnovu kog je objavljen patent, naučni rad i sada ova doktorska disertacija, oni su bili ti koji su duboko verovali i podržavali ideju. Bez njihove podrške, projekat sigurno ne bi bio završen i celokupna ideja bi ostala u domenu teorije za koju na kraju ne bi imali odgovarajuće podatke kojom bi je potkrepili i dokazali. Vukašinu Joviću, bih posebno želeo da se zahvalim na nesebičnoj pomoći i svim savetima koje mi je pružio prilikom pisanja ove disertacije. Takođe bih želeo i da se zahvalim svima koji su na bilo koji način učestvovali tokom realizacije projekta i doprineli konačnom proizvodu, a to su (u azbučnom redu) Vasić Veljko, Jovandić Isidora, Jovanović Vuk, Lazić Stefan, Lotrean Petar, Milovanović Miloš, Novaković Milan, Pantić Mladen, Petrović Đorđe, Pištinjat Neda, Šošić Miloš i Šumić Dražen.

Na kraju, želim da se zahvalim mojoj porodici koja me je podržavala u raznim idejama tokom celog života, i u onim manje dobrim, a i u onim odličnim. Svaki neuspeh koji sam doživeo je bila odlična životna lekcija, a svaki uspeh ne bi bio bitan da nemam vas sa kojima mogu da ih podelim i u sreći i veselju proslavim. Posebnu zahvalnost dugujem i supruzi Mini i sinu Lazaru, koji su imali strpljenja puno puta kada bi ih zamolio da me sačekaju da napišem još samo dve rečenice...

U Beogradu, 2021. godine Dejan Dunđerski

Rezime

Naslov: Sistem za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju

Oblast računarstva u klaudu se veoma intenzivno razvijala tokom prethodne decenije. Klaud okruženja i servisi koji takva okruženja pružaju postoje u različitim oblicima koji su i diskutovani u ovom radu. Prilikom njihove upotrebe uočeni su različiti problemi koji predstavljaju velike izazove za korisnike. Administratori i korisnici relacionih baza podataka su morali da rešavaju različite probleme i pre postojanja klaud platformi. Ažuriranje softvera, kao operativno zahtevan problem, pojavom relacionih baza podataka kao platforme u klaud okruženju postao je isključiva odgovornost pružaoca usluge. Nasuprot tome, problemi koji zahtevaju unapređivanje rada i optimizovanje relacione baze podataka postali su još izraženiji usled višeg nivoa apstrakcije koji donosi klaud okruženje i većeg broja relacionih baza koje je potrebno istovremeno održavati. S obzirom da se u klaud okruženju prikupljaju i čuvaju detaljne informacije o upotrebi servisa, u ovom radu je realizovan sistem koji na osnovu analize prikupljenih podataka olakšava korisnicima razumevanje funkcionisanja relacione baze podataka i pronalaženje uzroka problema koji se u radu sa njima mogu pojaviti. Nakon detaljnog pregleda uže i šire naučne oblasti, predložen je i definisan sistem koji uključuje dve vrste statističkih modela da bi se obezbedila i sveobuhvatnost i preciznost. Za donošenje konačnih odluka nad dobijenim podacima od strane statističkih modela o tome šta je uzrok a šta je posledica definisan je i ekspertski sistem. Opisan je i izgled infrastrukture koja je zasnovana na konceptu mikroservisa. Pored definisanog sistema, predstavljen je način organizacije tima sačinjenog od različitih aktera sa različitim odgovornostima. Konkretna implementacija sistema je izvršena u Azure platformi kompanije Microsoft. Implementirani sistem je potom podrobno testiran i evaluiran upotrebom realnog radnog opterecenja iz produkcionog okruženja Azure SQL relacione baze podataka tokom perioda od 6 meseci. Dobijeni rezultati su pokazali značajno unapređenje u pogledu performansi izvršavanja upita. Od pojedinačnih korisnika je dobijena i eksplicitna usmena i pismena potvrda o tome. Izvršena je i analiza dobijenih podataka o unapređenju korišćenja relacionih baza podataka svih korisnika platforme koji su se prijavili na ovaj sistem. Zaključak rada sadrži pravce i mogućnosti budućih istraživanja u ovoj oblasti.

Ključne reči: Klaud okruženje, Relacione baze podataka, Otkrivanje problema, Analiza problema, Ekspertski sistem, Statistički modeli, Azure SQL

Naučna oblast: Elektrotehnika i računarstvo

Uža naučna oblast: Softversko inženjerstvo

UDK broj: 621.3

Abstract

Title: Automatic Intelligent Database Troubleshooting System in Cloud Environment

Development of the cloud computing area has grown immensely in the past decade. This work evaluates various types of cloud environments and services provided to clients. Various problems have been found in the use of the cloud and these present big challenges for the users. Users and administrators of the relational databases have encountered various problems even in times before the cloud existed. Problems such as software updates of relational databases services in s a cloud platform became the responsibility of the service provider. This was a significant improvement that reduced operational costs. However, problems with service improvement and query optimizations scaled to a higher level due to the number of the relational databases and the higher level of abstraction introduced by the cloud environment. In the cloud environment very detailed information about service usage are accumulated constantly. Here is proposed a system that, based on these data, allows users to understand how the relation database works and detects the source of the problem much easier. After a detailed analysis of related work, the system is carefully designed and elaborated. It includes two types of statistical data models to provide both recall and precision, and an expert system for making final decisions. The appropriate infrastructure is based on a microservice architecture. The project team organization was composed of several actors with different skillsets. The system is implemented within the Microsoft Azure platform. Some specific details of this implementation are also presented. The system was fully tested and evaluated using real data workload from the production environment of the Azure SQL relation database in a period of 6 months. The results have shown a significant improvement in the query execution performance. A response from the customers who used this service has shown that the user experience was significantly improved. The conclusion contains an overview of the project, suggests the ideas for improvement of the system, and discusses how the similar approach can be used in scientific areas.

Keywords: Cloud computing, Relational databases, Database troubleshooting, Root cause analysis, Expert systems, Data science models, Azure SQL

Scientific field: Electrical and Computer Engineering

Scientific subfield: Software Engineering

UDC number: 621.3

Sadržaj

1. Uvod

Obast računarstva se eksponencijalno razvija već gotovo 8 decenija. Tokom tog perioda mogućnosti i upotrebe računarskih sistema su doživeli izuzetan rast. Nekada je samo jedan računar zauzimao čitave objekte dok se danas ista takva pa čak i veća procesorka snaga može naći i u prosečnom mobilnom telefonu. U početnom periodu računari su korišćeni prvenstveno u vojne svrhe dok se tokom vremena njihova upotreba proširila i na ostale naučne i privredne grane. Primena računarskih tehnologija sve više utiče na živote ljudi kao jedno od glavnih pomoćnih sredstava za obavljanje najrazličitijih zadataka.

U oblasti računarskih tehnologija posebno se ističu informacioni sistemi. Oni su se prvobitno sastojali od jednog centralnog servera koji je zbog tadašnje tehnologije bio velikih dimenzija. Usled velike cene samog hardvera i gabarita računarskih servera njihova primena je bila ograničena isključivo na velike kompanije, institucije i državne sisteme. Centralnim serverom se upravljalo preko terminala koji su sa njim bili povezani odgovarajućom računarskom mrežom. Zbog svoje arhitektire, ali i zbog porasta potreba korisnika, ovi sistemi su pokazali neke od svojih nedostataka. Prvi i osnovni nedostatak je činjenica da takav sistem sadrži jedan centralni server koji predstavlja usko grlo sistema. Ukoliko u vezi sa njim postoji neki problem ili centralni server zahteva redovno ili vanredno održavanje, celokupan sistem postaje privremeno neupotrebljiv za krajnje korisnike. Drugi nedostaci se odnose na ograničenja u pogledu performansi i propusnog opsega centralizovanog servera. Sa porastom upotrebe informacionih sistema povećavaju se i zahtevi koje od centralizovanog servera traže korisnici. Kako bi se prevazišli ovakvi i slični nedostaci, bilo je potrebno da se razviju nove paradigme u ovoj oblasti.

Početkom osamdesetih godina prošlog veka pojavila se i klasa ličnih računara (eng. ***personal computer***). Pokazalo se da su prednosti uvođenja personalnog računara u upotrebu višestruke, a ogledale su se u sve većoj pristupačnosti računara i računarskih servisa širokom krugu korisnika. Korisnik je sada mogao nesmetano i bez potrebe za serverom, na nečemu što je nekad bio terminal

za unos podataka, da samostalno obavlja sve složenije zadatke, kao što su uređenje teksta, izrada tabela, šema, dijagrama, grafičkih crteža, pa sve do zahtevnih proračuna za koje su pisani posebni programi. Informacioni sistemi su se tako u narednoj dekadi proširili ne samo u području industrijske upotrebe informacionih tehnologija, već i u svakom domu i na svakom radnom stolu.

Ipak, tek sa nastankom interneta kao globalne računarske mreže, pojavila se mogućnost povezivanja udaljenih i distribuiranih računarskih sistema. Internet je doneo velike prednosti u pogledu brzine, obima i kvaliteta komunikacije. Nekada su za razmenu informacija pisanim i štampanim materijalom bile potrebne nedelje i dani, dok danas, upotrebom interneta, poruka može da stigne sa jednog kraja sveta na drugi za svega nekoliko sekundi. Putem interneta je moguće doći i do različitih informacija putem pretraživača do kojih je nekada bilo gotovo nemoguće doći samostalno ili je potrebno vreme bilo značajno duže.

Potreban je bio čitav niz godina intenzivnog razvoja da se usavrše mrežni protokoli poveća protok informacija na internetu. Unapređeni protokoli i dostupnost različitih servisa su doveli do formiranja nove paradigme. Klaud računarstvo predstavlja novi kvalitativni korak u razvoju informacionih tehnologija. Klaud okruženje se može smatrati virtuelnim „centralizovanim serverom" koji može biti sačinjen od stotina ili hiljada fizičkih i distribuiranih servera, međusobno povezanih preko globalne ili lokalne računarske mreže. Shodno potrebama korisnika, klaud okruženje može predstavljati jedan server ili više njih.

Postoje različiti modeli klaud okruženja u kojima korisnici dobijaju različit nivo apstrakcije. Najosnovniji nivo apstrakcije predstavlja virtuelizaciju resursa na nivou fizičkog hardvera gde korisnici iznajmljuju infrastrukturu od pružaoca usluga. Naredni nivo predstavlja platformu koju korisnik iznajmljuje gde je nivo apstrakcije nešto veći i gde korisnik ne brine o infrastrukturi već je to nadležnost pružaoca usluga. Najviši nivo apstrakcije predstavlja softver kao servis, gde korisnik koristi gotov softverski paket, bez prethodne potrebe da ga instalira i detaljno konfiguriše. Ovako organizovanim serverima se pristupa putem interneta preko personalnih računara koji u toj organizaciji predstavljaju neki vid terminala za pristup klaud sistemu.

Kada se govori uopšteno o informacionim sistemima, najvažniji aspekat je postupak i metodologija koji se primenjuje prilikom čuvanja podataka. Postoje različiti pristupi čuvanju podataka, počevši od prostog upisivanja u datoteku do složenih struktura podataka u kojima postoje međusobne veze između zapisanih podataka. Pristup koji ima za cilj efikasno čuvanje podataka u struktuiranom obliku, između kojih postoje veze odnosno relacije, naziva se konceptom relacionih baza podataka. Ovaj koncept je dao odgovor na pitanje kako se podaci čuvaju, vremenom menjaju i upotrebljavaju, pogotovo kada se radi o velikim količinama podataka. U isto vreme operacije nad podacima takvog obima moraju biti efikasne i pouzdane. Klaud okruženja različitih pružalaca računarskih usluga nude različite mogućnost za rad sa relacionim bazama podataka. Osnovna ponuda uključuje mogućnost iznajmljivanja infrastrukture na kojoj se instalira i pokreće relaciona baza podataka. Sa druge strane, postoji mogućnost i iznajmljivanja jedne ili više relacionih baza podataka koje su u tom slučaju implementirane kao klaud platforma. Klaud platforma u tom slučaju osim relacione baze podataka nudi i dodatne softverske pakete koji olakšavaju rad njenim korisnicima. Jedan od osnovnih dodatnih paketa je, na primer, sistem za automatsko čuvanje i arhiviranje podataka do 30 dana.

Relacione baze podataka u klaud okruženju predstavljaju složena softverska rešenja. Pored kompleksnosti, sistemi zbog svoje nesavršenosti i mogućnosti korisnika da definiše proizvoljnu

logiku koja koristi sistem, mogu izazvati najrazličitije probleme. U lokalnom okruženju korisnika, takvi problemi su rešavani upotrebom poznate prakse i donošenjem dobrih odluka tokom projektovanja i razvoja. Danas postoji niz alata koji imaju zadatak da pomažu administratorima baza podataka prilikom obavljanja njihovih zadataka tokom testiranja relacionih baza podataka kako u razvojnim tako i u produkcionim okruženjima. Osnovni zadatak ovih alata je unapređenje dostupnosti i performansi samih relacionih baza podataka. U ekstremnim situacijama, i pored postojanja administratora baza podataka, sistemi koji poseduju složenu logiku često zahtevaju angažman posebnih eksperata koji upotrebom odgovarajućih ekspertskih alata i znanja umeju da reše i najsloženije slučajeve.

Sa druge strane, neki od problema koje korisnici u klaud okruženju prijavljuju mogu se smatrati prolaznim. Takvi problemi mogu da spadaju i u skup poznatih problema kod kojih, na primer, dolazi do otkaza hardverskih komponenti ili do softverskih problema gde se dešavaju različite greške tokom razvoja programskog koda. U kontekstu pojedinačnih manjih sistema, pronalaženje ovih problema je obično očigledno i razrešavanje je trivijalno. Ali, kada se radi o okruženju koje se sastoji od stotina hiljada servera i aplikacija, pronaći problem i rešenje za njega, imajući u vidu kompleksnost samog okruženja je za dva ili tri reda veličine teže.

Potrebne akcije za istraživanje ovakvih slučajeva zahtevaju značajno vreme. Ukoliko bi postojao sistem koji bi umeo da pravi razliku između prolaznih i trajnih problema, koji bi u isto vreme mogao da asistira korisniku i daje savete kako se problemi mogu rešiti, to bi predstavljalo značajno olakšanje i za korisnika ali i za pružaoca klaud usluga. U suprotnom, nepostojanje ovakvog sistema zahteva da korisnici ili unajmljuju eksperte ili da pružalac klaud usluga zapošljava više inženjera sistemske podrške što je gotovo nemoguće imajući u vidu veličinu klaud okruženja. Shodno tome, potrebno je automatizovati deo ili kompletan proces kako bi se olakšao i omogućio nesmetan rad relacionih baza podataka u klaud okruženju.

Imajući sve to u vidu, cilj ovog rada je projektovanje i implementacija sistema koji u vrlo kratkom vremenskom roku automatski može da utvrdi postojanje problema, prepozna njegov uzrok i ponudi moguće rešenje, ukoliko ono postoji. Poseban zahtev se ogleda u tome da predloženi sistem mora da vodi računa o tome da se uočavanje problema očekuje na nivou upita relacione baze podataka i predstavlja dodatni izazov prilikom rešavanja ovog problema.

Ovakav sistem za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju se oslanja na telemetrijske podatke o funkcionisanju servisa koji se neprestano prikupljaju i čuvaju. Oni predstavljaju ulaz za dve grupe statističkih modela koje imaju različitu i komplementarnu namenu. Prva grupa, generički modeli, ima za cilj što šire pokrivanje i obuhvatanje svih potencijalnih problema, ali se ne očekuje puna preciznost. Nakon toga, druga grupa, kategorizacioni modeli, ima za cilj da upare ono što su generički modeli detektovali sa konkretnim poznatim problemima. Za razliku od generičkih modela, kategorizacioni modeli se odlikuju velikom preciznošću ali manjim obuhvatanjem problema. Zato se rezultati dobijeni od strane modela analiziraju korišćenjem ekspertskog sistema, koji na osnovu odgovarajuće baze pravila donosi konačnu odluku o tome šta je uzrok a šta je posledica. Zbog samog pristupa, očekivani naučni doprinos se, pre svega, odnosi na pronalaženje rešenja za probleme koji zaista postoje s obzirom da su sve sprovedene analize vršene nad realnim okruženjem. Imajući u vidu složenost problema rezultati ne mogu da se dobiju za nekoliko sekundi, pa je cilj ovog rada da se realizuje sistem koji može u blisko realnom vremenu (reda 15-tak minuta) od pojave problema da ga otkrije i ponudi razumevanje problema u realnom produkcionom okruženju.

U ovom radu evaluacija nije sprovedena nad sintetizovanim podacima, koji bi ukazali na hipotetičke probleme, već je u tu svrhu upotrebljeno stvarno produkciono okruženje tokom dužeg vremenskog perioda od 6 meseci. Tom prilikom posmatrano okruženje je sadržalo oko dva miliona relacionih baza podataka od različitih korisnika zbog čega se može smatrati vrlo relevantnim radnim opterećenjem za testiranje i validaciju sistema. Takođe, imajući u vidu da je sistem sam po sebi veoma složen, doprinos se ogleda i u organizaciji rada ekspertskog tima zaduženog za implementaciju i integraciju sistema.

U nastavku rada, u drugom poglavlju, razmatrane su karakteristike klaud servisnih okruženja i različite vrste servisa. Poseban deo opisuje različite savremene pristupe analizi podataka kao što su to metode iz oblasti statistike, nauke o podacima i pretraživanja podataka. S obzirom na predmet rada, pažljivo su analizirani klaud koncepti iz ugla relacionih baza podataka. Na kraju poglavlja izvršena je postavka problema koji se rešava u ovom radu.

Postojeća rešenja, pristup istraživanja i analiziranja problema u bližoj i daljoj naučnoj oblasti su razmotreni i prikazani u trećem poglavlju. Opisana su predložena akademska rešenja, ali i sistemi iz komercijalnih sredina kompanija kao što su Microsoft, Amazon i Alibaba. Sa druge strane, posmatrajući širu naučnu oblast, akademski radovi su usmereni na rešavanje sličnih problema u srodnim oblastima kao što su to, na primer, oblast mreža i mrežnih protokola.

U četvrtom poglavlju definisani su osnovni gradivni blokovi sistema koje čine statistički generički i kategorizacioni modeli. Dat je njihov opis kao i način funkcionisanja i međusobne saradnje. Potom je predstavljen ekspertski sistem čiji je zadatak da razrešava situacije u kojima dolazi do preklapanja više rezultata od strane statistčkih modela. Na kraju poglavlja, kroz primere definisanih pravila unutar ekspertskog sistema data je ilustracija na koji način ekspertski sistem razume i procesuira podatke dobijene od strane statističkih modela.

U narednom, petom poglavlju, opisana je konkretna arhitektura na odgovarajućoj infrastrukturi koja je potrebna za realizaciju sistema. Definisan je izvor podataka i predstavljeni su zahtevi iz aspekta skaliranja samog sistema. Analizirane su različite vrste skladišta za telemetriju i definisana je upotreba za svako od njih. Aplikativni nivo sistema koji orkestrira izvršavanje statističkih modela zasnovan je na tehnologiji mikroservisa čiji je opis, takođe, dat u ovom poglavlju. Na kraju je predložen način čuvanja i objave dobijenih rezultata. Ono što je izuzetno važno, a često se u razmatranjima stavlja u drugi plan, je kontinuirano praćenje rada celog sistema ili njegovog nekog pod dela. Da bi rešenje bilo dugoročno održivo, potrebno je definisati i pratiti sve aspekte sistema i njegovog kvaliteta rada.

Šesto poglavlje daje opis organizacije tima koji je zadužen za implementaciju definisanog sistema i predstavljene arhitekture. U ovom poglavlju je izvršeno i poređenje definisanog pristupa sa pristupima koji su poznati u široj literaturi. Izuzetno je važno tačno odrediti koje zadatke koji akteri treba da obavljaju kako bi se postigla što veća efikasnosti tima, imajući u vidu da tim mogu da čine osobe sa različitim stručnostima i oblastima znanja.

Konkretna implementacija koja se zasniva na Azure klaud sistemu kompanije Microsoft opisana je u sedmom poglavlju. Predstavljen je način kako se podaci mogu uzorkovati na samom izvoru podataka, potom kako se obezbeđuje transport podataka kao i komande za instanciranje skladišta podataka. U nastavku je predstavljena implementacija modela, ekspertskog sistema, šeme relacione baze podataka u kojoj se čuvaju rezultati, kao i sam grafički interfejs koji korisnici mogu da upotrebe prilikom pretraživanja dobijenih rezultata.

Da bi formirani sistem imao praktičnu primenu, potrebno ga je i evaluirati. Osmo poglavlje sadrži evaluaciju sistema sprovedenu nad realnim okruženjem koje broji milione relacionih baza podataka. Posebna pažnja posvećena je rezultatima u saradnji sa dva korisnika platforme koji su koristili sistem i pritom potvrdili njegovu upotrebljivost. Pored tih rezultata, ilustrovani su i neki pojedinačni slučajevi koji su pomogli inženjerima sistemske podrške prilikom njihovog rada sa korisnicima koji su imali problem sa platformom. Na kraju dat je pregled rezultata dobijenih analizom ponašanja relacionih baza podataka korisnika koji koriste napravljeni sistem.

U poslednjoj, devetoj glavi dat je zaključak ovog rada, u kome je izvršen kratak sumarni pregled dobijenih rezultata i sagledani mogući pravci daljeg istraživanja.

2. Savremena servisna okruženja

Servisna okruženja predstavljaju dugotrajan trend u korišćenju računarskih sistema. Prvi računari su projektovani sa idejom da njihovim korisnicima pruže neku vrstu servisa i usluge, kako bi se određene operacije ubrzale i automatizovale što je više moguće. Razvojem računarskih sistema i složenost problema koje je moguće rešiti upotrebom računara se povećavala. U današnje vreme, servisna okruženja se više ne sastoje od jednog ili nekoliko računara, već su to izuzetno veliki centri koji se mogu smatrati globalnim računarskim sistemima za pružanje usluge korisnicima svuda na svetu.

U ovom poglavlju je izvršen uporedni pregled različitih karakteristika koje odlikuju savremena servisna okruženja. Pored toga, posebna pažnja je stavljena kako na obradu podataka u takvim okruženjima tako i na analizu iz perspektive relacionih baza podataka u klaud servisima. Na kraju će biti iznesena postavka problema, koja je bila osnova ovog istraživanja.

2.1. Karakteristike klaud okruženja

Računari su, zbog nesavršenosti tehnologije, prvobitno zauzimali čitave objekte, sale i prostorije. Korisnici su zadavali zadatke centralnom računaru putem terminala. Infrastruktura sa centralnim računarom i terminalima je kasnije zamenjena infrastrukturom koja je posedovala centralni server. Arhitektura takvog sistema se sastojala od jednog ili više centralnih računara, i više privatnih računara-terminala koji su i sami mogli da obavljaju određene zadatke za razliku od običnih terminala za pristup tradicionalnim računarima. U ovom slučaju, kompletna infrastruktura je vlasništvo kompanije koja je poseduje pa je ona odgovorna za njeno održavanje.

Pojavom Interneta kao globalne mreže, ono što je do tada bilo dostupno u lokalnoj mreži je postalo dostupno i iz bilo koje tačke na svetu. Zahvaljujući toj mogućnosti, kompanije koje su ranije

posedovale računarsku infrastrukturu, sa željom da se oslobode njenog održavanja, tu infrastrukturu iznajmljuju od pružaoca usluga u velikim hardverskim centrima. Oni se u literaturi sreću pod nazivom centri za podatke (eng. ***data center***) [1]. Takav pristup se može nazvati začetkom servisnog okruženja koje je danas poznato pod imenom klaud okruženje (eng. ***cloud***) [2].

Pružaoci usluga u klaud okruženju, takođe, nude povoljnije usluge u odnosu na cenu hardvera i cenu njegovog održavanja u slučaju posedovanja sopstvene infrastrukture [3]. To je prvenstveno moguće zahvaljujući virtuelizaciji resursa. U tom slučaju je moguće deliti iste fizičke resurse između više korisnika na siguran način. Sigurnost se postiže zahvaljujući implementaciji odgovarajućih mehanizama na nivou jezgra (eng. ***kernel***) operativnog sistema koji virtuelizuje prava pristupa. Imajući u vidu da više korisnika može koristiti isti fizički hardver, moguće je izvršiti efikasniju upotrebu hardverskih resursa u odnosu na tradicionalan pristup u kome bi fizički hardver postojao u odvojenim privatnim centrima za svakog korisnika ponaosob.

Ovakav pristup deljenih virtuelizovanih resursa koje kompanije pružaoci usluga primenjuju proizlazi iz više razloga. Upotrebom deljenih resursa, postižu se ekonomske prednosti a samim tim je moguće pružiti i jeftinije usluge. Kompanije koje pružaju usluge nižom cenom žele da motivišu korisnike da pređu iz svog privatnog okruženja u klaud okruženje. Drugi razlog za nižu cenu u klaud okruženju je posledica želje samih pružaoca usluga koji na taj način optimizuju profit na duži vremenski period. Iako kratkoročna zarada može biti veća, pružaocima usluga je isplativije da imaju više korisnika i da, i pored manje kratkoročne zarade, dugoročna zarada usled većeg broja korisnika bude značajno veća.

Kao što je već rečeno, u klaud okruženju je moguće iznajmiti jeftinije resurse u odnosu na tradicionalni pristup posedovanja sopstvene infrastrukture. Slična logika se može naći i u mnogim praktičnim primerima iz života. Na primer, u slučaju da perionica veša kao poslovni model uopšte ne postoji u određenim sredinama, svako domaćinstvo mora da poseduje svoju mašinu za veš. Te mašine za veš obično nisu iskorišćene svakog dana, već u proseku od jednom do najviše nekoliko puta nedeljno u zavisnosti od broja članova domaćinstva. Svakako se može zaključiti da se ne koriste svih 24 sata, 7 dana u nedelji. Ako jedna perionica veša sa 50 mašina može da opsluži 1000 domaćinstava, to znači da je stepen uštede iz pogleda prvobitne investicije koju svako domaćinstvo mora da odvoji u ovom slučaju u vrednosti od 950 mašina. Istina je da je kvalitet, a samim tim i materijalna vrednost svake od 50 veš mašina koje su namenjene perionicama, značajno veći od svake od 1000 veš mašina koja se prozivode za mala domaćinstva. Ipak, kada se govori o materijalnoj vrednosti 1000 mašina za veš, ona daleko prevazilazi materijalnu vrednost 50 profesionalnih mašina za veš.

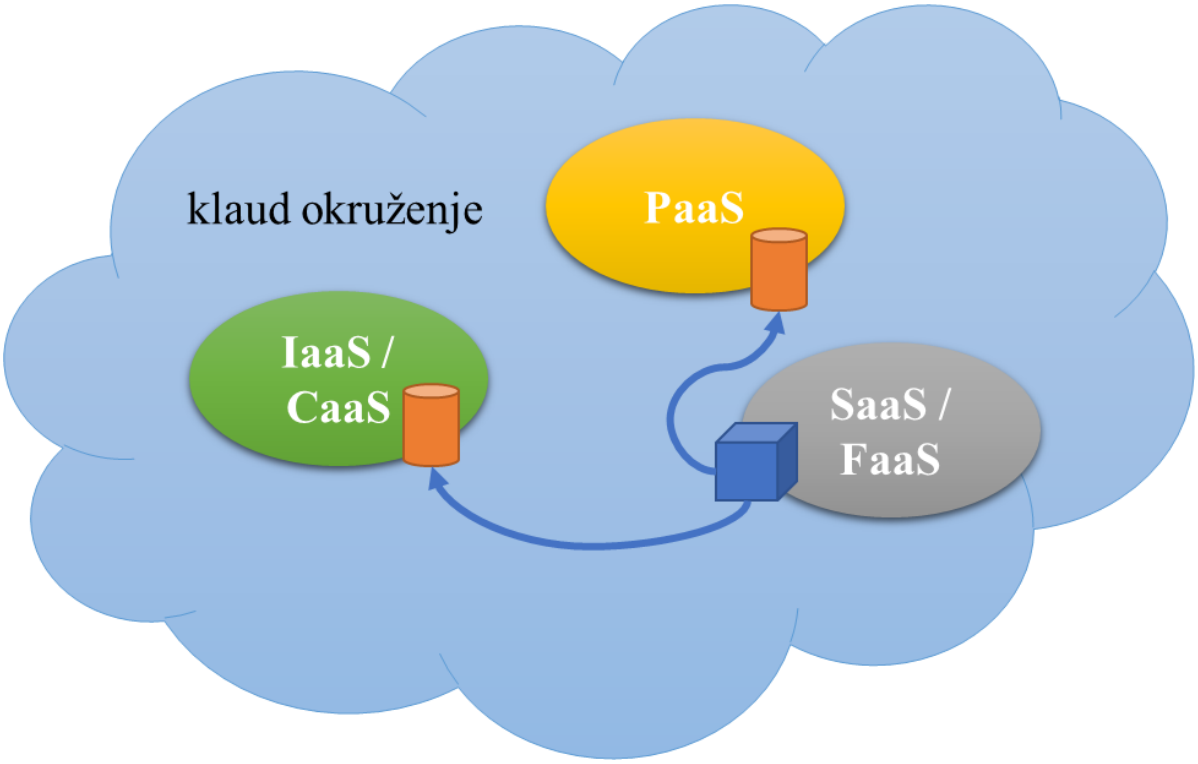
Prema tome, može se zaključiti da ušteda u klaud sistemima prvenstveno dolazi od bolje infrastrukture koja je tokom vremena više i bolje iskorišćena. Zatim, održavanje je značajno jednostavnije s obzirom da nije potrebna instalacija hardvera na lokaciji korisnika, već korisnici iznajmljuju infrastrukturu shodno svojim potrebama. Hardverski ili softverski kvar je problem pružaoca usluge, a ne samog korisnika koji koristi servis. U tom slučaju otpornost na otkaze je mnogo veća, zbog redundanse koju pružalac usluga generalno obezbeđuje. Pošto je pružaocu usluga osnovna delatnost posedovanje infrastrukture i njeno unapređenje, postoji prirodna motivacija za konstantno unapređivanje i optimizovanje procesa. Ključna prednost klaud okruženja za pružaoce usluga se ogleda u činjenici da oni mogu dodatno da optimizuju infrastrukturu u slučajevima kada više različitih korisnika u isto vreme ne koriste celokupne resurse. Sa druge strane, prednosti za same korisnike su višestruke. Kao što je to već napomenuto, ukupni troškovi su manji. Takođe,

korisnici u klaud okruženju mogu skalirati svoje aplikacije i servise i horizontalno i vertikalno s obzirom da se kapaciteti klaud okruženja smatraju neograničenim za jednog korisnika. Problemi za korisnike u ovakvom okruženju mogu doći zbog potrebe za pristupom udaljenoj lokaciji, što u lokalnim mrežama pre nije bio slučaj. Zbog toga, korisnici se mogu oslanjati na jednog ili više internet provajdera i to je van kontrole samih klaud provajdera.

Klaud sistemi danas pružaju različite nivoe apstrakcije za svoje korisnike [4]. Kao što se može videti na slici 2.1. klaud sistemi zastupljeni su u različitim oblicima. Nivoi apstrakcije u klaud sistemima su:

- Infrastruktura kao servis (***IaaS***),
- Kontejner kao servis (***CaaS***),
- Platforma kao servis (***PaaS***),
- Softvera kao servis (***SaaS***),
- Funkcija kao servis (***FaaS***).

Nivoi apstrakcije rastu po redosledu nabrajanja od servisa koji zahtevaju najveći nivo interakcije i pritom su najjeftiniji, do servisa koji ne zahtevaju održavanje, ali su ujedno i skuplji zbog pruženih naprednih usluga.



Slika 2.1. Klaud okruženje sa različitim nivoima apstrakcije

2.1.1. Infrastruktura kao servis - ***IaaS***

U okviru ***IaaS*** okruženja [4], korisnik iznajmljuje resurse u vidu vertiuelizovanog hardvera sa instaliranim operativnim sistemom. Počevši od nivoa operativnog sistema, ceo sistem je pod kontrolom korisnika. Korisniku nije omogućen pristup samo fizičkim hardverskim resursima na kojima se nalazi virtuelna mašina u datom trenutku, zbog održavanja sigurnosti između više različitih korisnika.

U okruženju u kojem je korisniku omogućen potpuni pristup operativnom sistemu virtuelne mašine, na korisniku je da u skladu sa sopstvenim potrebama izvrši konfigurisanje i podešavanje sistema. Takođe, korisnik sam instalira i održava aplikacije koje će koristiti na toj virtuelnoj mašini (serveru).

Na primer, ako se radi o serverskoj aplikaciji za razmenu elektronske pošte, na korisniku je da instalira server za razmenu elektronske pošte, da ga konfiguriše po svom nahođenju, i da ga zatim, kontinuirano održava. Pored toga, korisnik mora da vodi računa i o antivirusnim programima i ostalim paketima operativnog sistema koje je potrebno ažurirati sa vremena na vreme zbog sigurnosti samog servera. Odgovornost pružaoca usluga se u ovom modelu ograničava na održavanje hardverske infrastrukture na kojoj se pokreću virutelne mašine.

2.1.2. Kontejner kao servis - ***CaaS***

Kontejnerizacija servisa je trend koji je od skoro postao popularan [5]. U takvom okruženju korisnik kao i kod ***IaaS*** iznajmljuje virtuelizovane hardverske resurse. Korisnik kod kontejnera takođe poseduje pristup samo njegovom sadržaju i nema mogućnost pristupa fizičkim hardverskim resursima čime se postiže odgovarajući stepen sigurnosti. Kontejneri predstavljaju tehnologiju koja, u odnosu na virtuelne mašine, zauzima manju količinu resursa na nivou fizičkih mašina.

To je prvenstveno moguće zbog same implementacije, imajući u vidu da kontejner i operativni sistem na fizičkoj mašini dele deo jezgra operativnog sistema, tako da je osnovni deo kontejnera značajno manji i već postoji na operativnom sistemu samog fizičkog servera. Zbog toga postoje određena ograničenja, pa unutar kontejnera nije moguće pozivati sve vrste aplikativnih interfejsa operativnog sistema. Kontejneri su, pre svega, namenjeni mikroservisnoj arhitekturi. U odnosu na tradicionalne monolitne aplikacije koje se izvršavaju na jednom serveru ili na jednoj virtuelnoj mašini, mikroservisna arhitektura teži razbijanju aplikacija na više paketa od kojih svaki paket može predstavljati jedan kontejner.

Kao i kod virtuelnih mašina, korisnik u slučaju kontejnera iznajmljuje same kontejnere od pružaoca usluga, za koje definiše odgovarajuću sliku (eng. ***image***), na osnovu koje se inicijalizuje sadržaj i konfiguriše šta će kontejner tokom svog životnog veka izvršavati.

Na primeru serverske aplikacije za razmenu elektronske pošte, na korisniku je da izabere postojeću ili napravi novu definiciju kontejnera koja sadrži serverski softver za razmenu elektronske pošte. Definicija kontejnera može da sadrži konfiguraciju, ali je u isto vreme moguće konfigurisanje izvršiti i nakon pokretanja kontejnera. Odgovornost pružaoca usluga i u ovom modelu je ograničena na održavanje hardverske infrastrukture na kojoj se pokreću kontejneri, ali je ujedno i proširena na neke od aspekata kontejnera. U ovom slučaju korisnik ne mora da vodi računa o antivirusnim

programima, pošto se oni nalaze na operativnom sistemu fizičke mašine koji ujedno vodi računa i o ponašanju kontejnera.

2.1.3. Platforma kao servis - ***PaaS***

Kada je reč o platformskom ***PaaS*** okruženju [4], u odnosu na ***IaaS*** i ***CaaS*** okruženje, korisnik ima manji obim kontrole, ali se smanjuje odgovornost i obim posla koju korisnik mora da obavlja što predstavlja veliku prednost za korisnika platforme. Kod ***PaaS*** okruženja, korisnik iznajmljuje samo platformu. Softver kroz koji pružalac usluga nudi serverske aplikacije sa stalnim održavanjem, smatra se platformom. Korisniku je i dalje ostavljena mogućnost prilagođavanja platforme svojim potrebama putem aplikativnog interfejsa koji platforma nudi.

Na primeru servera za razmenu elektronske pošte, u ***PaaS*** smislu, korisnik bi od pružaoca usluge dobio pristup jednoj instanci serverske aplikacije koju bi mogao da konfiguriše upotrebnom ličnih parametara (***mail*** domen, numerička vrednost porta na kom ***mail*** server može da prima zahteve, itd.).

PaaS sistemi su uglavnom implementirani upotrebom ***IaaS*** ili ***CaaS*** okruženja uz dodatne aspekte kompletnog održavanja, kao i brige o kontinuiranoj dostupnosti servera. U tom slučaju, korisnik ne mora da vodi računa o tome da li su operativni sistem, antivirusni program, kao i sam server za razmenu elektronske pošte ažurirani.

2.1.4. Sofver kao servis - ***SaaS***

Softver kao servis je okruženje koje je uobičajenim korisnicima najpoznatije u odnosu na prethodna okruženja [4]. U tom smislu, softver kao servis predstavlja aplikaciju dostupnu svojim korisnicima putem mreže bez ikakvog održavanja i konfigurisanja. Konkretni oblici mogu biti različiti, počevši od aplikativnih interfejsa, web stranica pa do mobilnih aplikacija.

U kontekstu servera za elektronsku poštu, najpoznatiji ***SaaS*** je servis kompanije Google pod nazivom Gmail [6]. U ovom slučaju korisnik ima mogućnost da se registruje i da nesmetano koristi javno dostupnu klijentsku aplikaciju za elektronsku poštu. Servis koji to pruža je u potpunosti apstrahovan i zatvoren za korisnika i korisnik nema ni obavezu održavanja niti kontrolu nad ponašanjem samog servera i servisa.

2.1.5. Funkcija kao servis - ***FaaS***

Funkcija kao servis, ***FaaS***, je moderan koncept koji je prvobitno uveden od strane kompanije Amazon pod nazivom ***AWS Lambda*** [7], a potom i od strane kompanije Microsoft pod nazivom ***Azure Functions*** [8]. Ovo okruženje je poznatije inženjerima softvera u odnosu na uobičajene korisnike. Pružalac funkcija kao servisa omogućava korisniku usluga da definiše:

- aplikativni interfejs funkcije,
- telo funkcije,
- tip rezultata funkcije.

Uglavnom, ***FaaS*** koncept se implementira na arhitekturi koja može biti ili ***IaaS, CaaS*** ili ***PaaS*** tipa. Za korisnike, takva arhitektura je pogodna zato što u potpunosti eliminiše održavanje koje je i dalje obaveza korisnika u slučaju ***PaaS*** arhitekture. U ***SaaS*** slučaju, korisnici dobijaju kompletne softverske pakete, dok u ***FaaS*** konceptu dobijaju mogućnost da pokreću arbitrarnu logiku, koja uglavnom predstavlja glavnu vezu između aplikativnog korisničkog interfejsa i relacione baze podataka u pozadini celokupnog sistema. Takođe, jedna od glavnih prednosti ***FaaS*** koncepta se ogleda u mogućnosti jednostavnog horizontalnog skaliranja na nivou jednog aplikativnog interfejsa.

Kada posmatramo ***FaaS*** u kontekstu servera za elektronsku poštu, koncept bi mogao da se koristi kao nivo aplikativnog interfejsa koji je upotrebljen u ***PaaS*** rešenje putem kojeg korisnici mogu da konfigurišu server. Takođe, ***FaaS*** bi mogao da se upotrebljava i kao deo ***SaaS*** arhitekture gde je ***FaaS*** zadužen za komunikaciju sa ostalim komponentama u softveru za elektronsku poštu.

2.2. Obrada podataka u savremenim servisnim okruženjima

Pojava servisnih okruženja kako u privatnim data centrima tako i u klaudu je prvenstveno imala za cilj da se složeni zadaci pojednostave i ubrzaju. Daljim razvojem su pružaoci usluga otkrili da mogu da prikupljaju informacije o načinu upotrebe njihovih servisa kako bi mogli dalje da unapređuju i optimizuju posmatrane servise. Sa rastom takvih sistema, došlo je i do prirodnog povećanja količine prikupljenih podataka. Tako su obrada velike količine podataka i njihovo razumevanje u ograničenom vremenskom intervalu postao novi izazov.

Klaud servisi takođe donose i bogatu telemetriju (eng. ***telemetry***) na osnovu koje je moguće pratiti sve aspekte relacionih baza podataka, kao što su, na primer, broj i vremena izvršavanja upita, broj grešaka, i slično [9]. Termin telemetrija se odnosi na sačuvanu listu događaja ili logove nastale prilikom izvršavanja. Na osnovu važnosti samih podataka i stepena njihove promenljivosti podaci se uzorkuju i prikupljaju sa različitom učestanošću.

Podaci se čuvaju u struktuiranom obliku u okviru data centara u više različitih skladišta podataka. Vruće skladište (eng. ***hot storage***) se koristi u slučajevima kada je potrebna brza reakcija. Najpoznatiji primer su mehanizmi uzbunjivanja (eng. ***alerting***), gde je brz pristup podacima od izuzetne važnosti. Upiti nad takvom vrstom skladišta podataka se moraju izvršiti u odgovarajućem vremenskom periodu kako bi dobijeni rezultati bili relevantni i kako bi odgovarajuća reakcija na njih bila moguća. Sa druge strane, hladno skladište podataka (eng. ***cold storage***) sadrži velike količine podataka čiji je period čuvanja daleko duži, ali se ujedno ti podaci i koriste za različite analize čije rezultate nije kritično dobiti u kraćem vremenskom periodu.

Tehnike obrade velike količine podataka se zasnivaju na tradicionalnim pristupima kao što je statistika. Daljom upotrebom statistike u računarskim tehnologijama definisani su novi koncepti. Koncept nauke o podacima (eng. ***data science***) predstavlja primenu statistike i verovatnoće na veliku količinu podataka uz pomoć upotrebe odgovarajućih algoritama. Pretraga podataka (eng. ***data mining***) je koncept koji omogućava pronalaženje nepoznatih odgovora na pitanja zahvaljujući upotrebi statističkih metoda, računarskih tehnologija i velike količine podataka.

2.2.1. Statistika

Statistika kao oblast matematike postoji već stotinama godina. Metode istraživanja u ovoj oblasti su zasnovane na pojmovima i metodama teorije verovatnoće [10] koje se primenjuju na konkretne masovne pojave. Na osnovu teorema moguće je doneti zaključke analizom prikupljenih podataka [11].

Tradicionalno, podaci su prikupljani od strane ljudi upotrebom anketa i upitnika. Ankete su uglavnom sačinjene od dva tipa pitanja. Prvi tip pitanja su opšteg tipa i njihova namena je da različite anketirane izvore svrsta u određene kategorije i grupe. Drugi tip pitanja predstavljaju cilj ankete i nakon njihove analize dobijaju se konkretna saznanja pa je moguće doneti određene zaključke. Ankete se sprovode nad relevantnim podskupom pod kojim se podrazumeva, na primer, raznolika grupa ljudi po različitim kriterijumima poput uzrasta, pola, bračnog stanja i slično.

Statistika se danas upotrebljava u mnogim oblastima poput medicine, fizike, meteorologije, hemije, biologije, finansijama, itd. Primeri upotrebe statistike u ovim oblastima su: saznanja o delotvornosti određenih lekova na različite grupe pacijenata, praćenje i analiza događaja ili konkretno prosečne temperature za određeno doba godine, analiza vrednosti akcija na berzama, itd. Glavni cilj statistike u ovim oblastima je da kroz analizu prikupljenih podataka potvrdi ili opovrgne određene pretpostavke koje pomažu prilikom donošenja različitih odluka.

Sa druge strane, serveri koji se nalaze u klaud servisnom okruženju predstavljaju neku vrstu emitera masovnih pojava. Informacije o njihovom ponašanju su izuzetno važne s obzirom da se daljom statističkom analizom i drugim analizama prikupljenih podataka mogu doneti različiti zaključci o načinu njihove upotrebe a nakon toga i o mogućim poboljšanjima. Statistički modeli predstavljaju efikasan način za analizu podataka i donošenje osnovnih zaključaka o posmatranim pojavama koje se mogu desiti u klaud servisnom okruženju.

2.2.2. Nauka o podacima

U odnosu na statistiku koja je prisutna već duže vreme, nauka o podacima [12] predstavlja interdisciplinarnu oblast koja se pojavila kao posledica činjenice da računarske tehnologije omogućavaju prikupljanje i čuvanje izuzetno velike količine podataka. Sa porastom hardverskih mogućnosti, kompanije su počele da upotrebljavaju računare za prikupljanje istorijskih informacija koje se čuvaju u skladištima podataka. Zbog svoje veličine, statističku analizu sadržaja ovakvih skladišta ne može uraditi jedna ili više osoba, već se u tom slučaju, takođe, upotrebljavaju računari.

Za razliku od tradicionalnih primena statistike gde se rezultati prvenstveno zasnivaju na teorijskim saznanjima nad ograničenim skupom podataka, nauka o podacima zasniva se na istim ili sličnim saznanjima koje dobijamo na osnovu automatizovane obrade velike količine sačuvanih podataka. Upotrebom sačuvanih podataka, omogućeno je odlučivanje na većoj skali koju čovek kao pojedinac u izuzetnom kratkom vremenskom intervalu ne može samostalno čuvati ili analizirati.

Analizom podataka koji su prikupljeni tokom perioda koji se desio u prošlosti moguće je predvideti i buduće događaje. Što je količina prikupljenih istorijskih podataka veća, to je i veća verovatnoća da će buduća procena biti tačna. Kompanije su, na primer, prepoznale tu korist i počele su da upotrebljavaju prikupljene podatke sve više i da nad njima sprovode analize kako bi odluke ljudi

koje vode te kompanije bile proverene i utemeljene na osnovu relevantnih podataka. Glavni cilj je svakako optimizacija rada i povećanje profita, a to se postiže donošenjem odgovarajućih odluka i eliminacijom pogrešnih odluka.

U domenu računarske tehnologije, prva mašina koja je uspela da pobedi čoveka, prvenstveno zbog količine informacija koje je posedovala i brzine njihove obrade, je specijalizovani računar IBM Watson [13]. Ovaj računar je specijalno napravljen za učešće u šahovskoj partiji. Prilikom odigravanja šahovske partije, računar je odlučivanje zasnivao na dva izvora. Prvi je baza podataka koja je sadržala veliki broj različitih šahovskih partija, dok je drugi predstavljao algoritam koji je generisao stablo poteza i određivao verovatnoću povoljnog ishoda za svaki od njih. Na osnovu ova dva izvora, analizom podataka računar je uspevao da donese odluku o tome koji je to najbolji sledeći potez koji može da odigra kako bi postigao najveću moguću verovatnoću da na kraju partije pobedi. Sličan pristup je primenljiv kao rešenje i za na mnoge druge probleme gde postoji velika količina podataka a gde je vreme prepoznavanja i razumevanja kritično. Samim tim, isključiva upotreba ljudstva za rešavanje ovakvih događaja je moguća u određenim pojedinačnim slučajevima, ali kada se govori o velikoj brzini prepoznavanja i razumevanja koje se meri u sekundama ili kada se govori o milionima problema koje treba razumeti i rešiti u kratkom vremenskom intervalu, analiza velike količine podataka od strane računara je jedini način za rešavanja takvih problema.

2.2.3. Pretraga podataka

Da bi se sačuvani podaci iskoristili na najbolji mogući način, potrebno je prvenstveno izvršiti analizu podataka. Tom prilikom teži se eliminaciji „šuma”, koji može postojati kao posledica grešaka u prikupljanju podataka. Takav pristup analize podataka se u literaturi susreće pod terminom pronalaženje skrivenog znanja.

Pretraga podataka [14] predstavlja proces otkrivanja korisnih informacija, načina ponašanja i odnosa između različitih podataka. Kada se govori o ovom procesu, podrazumevaju se velike količine podataka. U suprotnom bi to mogao da radi i čovek upotrebom anketa i drugih tradicionalnih mehanizama. Proces pronalaženja informacija i znanja iz velike količine podataka je prvenstveno upotrebljavan od strane banaka i velikih trgovinskih lanaca koji su na sistematski i automatizovan način pokušavali da reše neke od problema koje su imali u svom poslovanju.

Jednostavni primeri u tom domenu se odnose na otkrivanje zloupotreba kod elektronskog plaćanja putem kreditnih kartica ili optimizaciju prodaje proizvoda. Iako je moguće prepoznati odgovarajuće jedinstvene obrasce ponašanja upotrebom ovog procesa, njegova namena je ipak usmerena ka pronalaženju odgovarajuće grupe koja može pomoći većem broju aktera.

Tehnologija pretrage podataka u kontekstu klaud servisa posebno dolazi do izražaja imajući u vidu da prikupljeni podaci u različitim oblicima nisu sami po sebi dovoljni. Ukoliko podaci nisu upotrebljeni na pravi način, oni u tom slučaju nemaju vrednost. Potrebna je odgovarajuća analiza različitih grupa podataka, što nije jednostavan zadatak imajući u vidu veličinu samih podataka.

U današnje vreme, tehnike mašinskog učenja i pronalaženja skrivenog znanja u podacima [15] se upotrebljavaju u različite svrhe. Sa pojavom interneta, klaud servisa i mogućnošću prikupljanja velike količine podataka, pomenute tehnologije su postale izuzetno aktuelne. Glavna ideja ovih

disciplina se ogleda u prepoznavanju obrazaca ponašanja i njihovom korelisanju sa akcijama koje su ili usledile pre ili posle nekog trenutka za koji se smatra da je od interesa za korisnika.

Sa rastom samih sistema, povećava se i potreba za razumevanjem, istraživanjem i analiziranjem kompleksnih sistema u kratkom vremenskom roku. U takvim sistemima, odgovarajuća analiza podataka od strane inženjera i naučnika u implementaciji efektnih modela je ključna. Radovi [16] i [17] i sami ukazuju na ovaj trend i pristup rešavanju problema.

2.3. Relacione baze podataka kao klaud servis

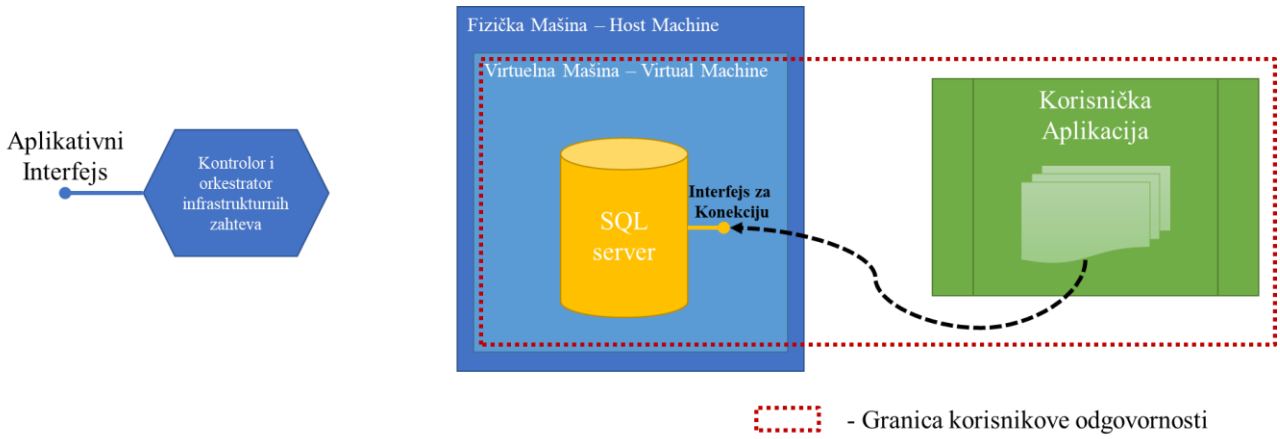
Relacione baze podataka pre svega predstavljaju tehnologiju koja na efikasan način omogućava čuvanje, pretraživanje i dohvatanje podataka. Većina aplikacija koja ima potrebu da obrađuje i čuva podatke koristi relacione baze podataka. U tim slučajevima, sadržaj podataka je takav da se nad njima mogu definisati odgovarajuće relacije. Tradicionalno je to rađeno čuvanjem informacija u datotekama, ali se ovaj pristup pokazao kao izuzetno neefikasan, posebno u situacijama kada je bilo potrebno pretraživati konkretnu informaciju u okviru velike količine podataka. Zahvaljujući postojanju relacija, pretraživanje podataka u relacionim bazama podataka je izuzetno efikasno. Interakcija aplikacije sa relacionim bazama podataka je standardizovana upotrebom SQL jezika (eng. ***structured query language***) [18] koji predstavlja još jedan razlog zašto je ova tehnologija popularna već desetinama godina.

Ukoliko se definisani klaud koncepti analiziraju iz perspektive relacionih baza podataka, njihova implementacija je dostupna u ***IaaS, CaaS*** i ***PaaS*** oblicima. Sa druge strane, ***SaaS*** i ***FaaS*** oblici vrlo često koriste relacione baze podataka za čuvanje parametara podešavanja servisa, kao i za čuvanje informacija o svojim korisnicima.

2.3.1. Relaciona baza podataka kao ***IaaS*** ili ***CaaS***

Kod ***IaaS*** i ***CaaS*** oblika relacionih baza podataka, korisnik iznajmljuje virtuelne mašine ili kontejnere i samostalno bira koji će se SQL server instalirati na njih. Nivo apstrakcije takođe postoji u hardverskom smislu, ali korisnik i dalje ima kontrolu nad aspektima operativnog sistema na virtuelnom iznajmljenom resursu gde se mogu instalirati i dodatne propratne aplikacije za praćenje rada samog iznajmljenog resursa ili za praćenje rada samog SQL servera.

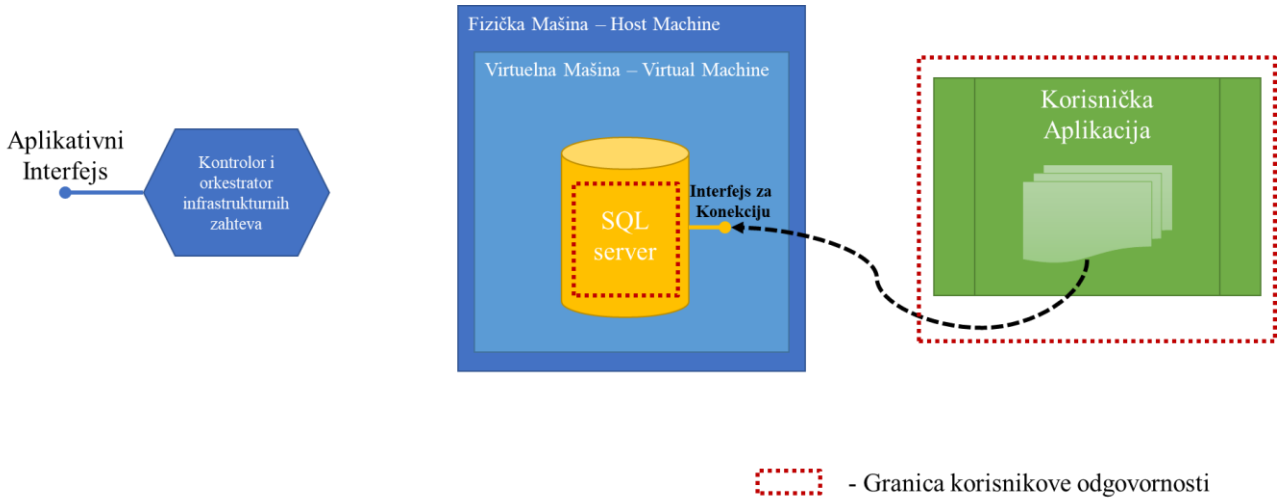
Podela odgovornosti u tom slučaju prikazana je na slici 2.2. Isprekidanom crvenom linijom označen je domen koji predstavlja odgovornost korisnika. Oblast van ovog domena se smatra odgovornošću pružaoca usluga. Kao što se može videti, korisnik nije odgovoran za fizički hardver i odgovornost pružaoca usluge je da virtuelnu mašinu ili kontejner pripremi za korisnika. Odgovornost pružaoca usluge je i da u slučaju otkaza fizičke mašine, za korisnika obavi migraciju virtuelne mašine ili kontejnera sa kompletnim sačuvanim podešavanjima i podacima na drugu fizičku mašinu.



Slika 2.2. Granica odgovornosti u ***IaaS*** i ***CaaS*** okruženju
na primeru relacione baze podataka

2.3.2. Relaciona baza podataka kao ***PaaS***

U ***PaaS*** obliku, nivo apstrakcije za krajnjeg korisnika je nešto viši i u takvom okruženju korisnik ne može da instalira aplikacije ili druge softverske module na sam operativni sistem virtuelne mašine. S obzirom na podelu odgovornosti, predstavljenu na slici 2.3. crvenom isprekidanom linijom, od pružaoca usluga se zahteva daleko više odgovornosti, a samim tim i veći broj aktivnosti koje treba da obavlja umesto korisnika.



Slika 2.3. Granica odgovornosti u ***PaaS*** okruženju na primeru relacione baze podataka

U ovoj situaciji može doći do poteškoća ukoliko korisnik treba da istražuje i analizira ponašanje same virtuelne mašine ili fizičkog servera relacione baze podataka. Ovakve operacije su tradicionalno poveravane administratoru baza podataka. U klaud okruženju administratori baza podataka više nemaju pristup kako fizičkim i virtuelnim mašinama, tako ni samoj instalaciji SQL

servera. U tom slučaju potrebno je napraviti sistem koji će omogućiti administratorima da prate ponašanje samog SQL servera i ponašanje pojedinačnih relacionih baza podataka.

Dodatna dimenzija koja usložnjava problem se odnosi i na broj relacionih baza podataka koje administrator mora da prati i podešava. Nekada su to u korisničkom privatnom okruženju (eng. ***on premise***) bile desetine baza podataka, ali se danas u klaud okruženju taj broj meri u hiljadama i stotinama hiljada.

Kako bi se smanjili operativni troškovi, u velikom broju slučajeva relacione baze podataka se više ne instaliraju na privatne servere. Danas se teži njihovoj migraciji u klaud okruženje. Najnoviji pristup u tom domenu potiče od više kompanija koje pružaju klaud usluge. Najpoznatije relacione baze podataka u klaud okruženju dolaze iz kompanije Microsoft [19], Amazon [20] i AliBaba [21]. Ove kompanije su prve ponudile svojim korisnicima relacione baze podataka u ***PaaS*** obliku koji se još sreće i pod nazivom „***Database as a service***" (skr. ***DBaaS***) model [22]. ***DBaaS*** model je zasnovan na klaud tehnologiji koja može da prati i održava milione relacionih baza podataka. Kao što je prethodno napomenuto, u ovakvom okruženju, korisnik ne treba da brine niti da vodi računa o održavanju hardvera i softvera. Takođe, svi softverski paketi koji su potrebni za nesmetno funkcionisanje servisa su odgovornost klaud provajdera [22]. U kompanijama Microsoft i AliBaba je primećeno da sve više korisnika od klaud servisa očekuju da ti servisi budu inteligentni i da dodatno olakšavaju upotrebu i dalje održavanje servisa. Kompanija AliBaba to naziva i „***Intelligence driven/self-driving***" tehnologijom [21].

2.3.3. Inteligencija u relacionim bazama podataka u klaud okruženju

Osim instalacije i ažuriranja softvera, proizvođači softvera i pružaoci softverskih usluga se trude da ponude svojim korisnicima još više olakšica. Zbog posedovanja kontrole nad fizičkim i virtuelnim mašinama pružaoci usluga nude i automatsko poboljšavanje performansi kao i automatsko prepoznavanje problema.

Prikupljanje podataka sa velikog broja relacionih baza podataka donosi izuzetne mogućnosti za analizu. Iako je u ovako velikim sistemima velika i verovatnoća da različiti korisnici koriste relacione baze podataka na različite načine, upotrebom i analizom prikupljenih podataka moguće je otkriti karakteristične oblike i šablone sličnih načina upotrebe relacionih baza podataka.

Pošto je moguće otkriti specifičan oblik ponašanja, na više različitih relacionih baza podataka koje pripadaju različitim korisnicima, postoji mogućnost za dalje analiziranje i razumevanje određenog specifičnog slučaja ukoliko do njega dođe. Na ovaj način je moguće izvršiti brže i bolje istraživanje jednog specijalnog ili više sličnih problematičnih slučajeva koji bi se mogli desiti u budućnosti. Razumevanje različitih slučajeva omogućava klaud okruženju da pruži dodatnu uslugu korisniku koja će sadržati informacije o načinu upotrebe same relacione baze podataka. Pored toga, moguće je podeliti informacije o konkretnim problemima kao i o akcijama koje je moguće primeniti kako bi se ti problemi rešili. Takođe, moguće je obavestiti korisnike o tome kako mogu bolje da konfigurišu relacionu bazu podataka ili da u potpunosti to konfigurisanje automatizuju i prepuste ga samom pružaocu klaud servisa.

2.4. Postavka problema

Kao što je tokom ovog poglavlja konstatovano, poslednja tehnološka dostignuća u industriji informacionih tehnologija promenila su sam proces i način kako se problemi istražuju u servisima. Iznajmljivanje i zakup hardverske opreme omogućavaju brži i jednostavniji razvoj aplikacija i servisa u klaud svetu. U isto vreme, ovakav pristup donosi i dodatni nivo apstrakcije.

Danas servisi, a pogotovo relacione baza podataka, nisu više dostupne korisnicima i administratorima na nivou fizičkih resursa, već im je moguće pristupiti samo putem aplikativnog interfejsa ili putem uspostavljanja direktne konekcije ka samoj relacionoj bazi podataka. U ovakvom okruženju nije više moguće podešavati ni sam server ni operativni sistem. Neke od tih operacija su dostupne isključivo putem aplikativnog interfejsa, dok su neke operacije u potpunosti blokirane i nisu omogućene za upotrebu.

Relacione baze podataka su evoluirale u veoma složena softverska rešenja. Ovakvi sistemi zbog svoje nesavršenosti ipak mogu da dovedu do toga da korisnik, zbog činjenice da može da napravi bilo kakvu proizvoljno složenu logiku, može dovesti do degradacije performansi a što može imati kaskadni uticaj na aplikaciju. Prethodno, takvi problemi su rešavani upotrebom poznate prakse i donošenjem dobrih odluka tokom projektovanja i razvoja. Takođe, administratori baza podataka su bili odgovorni da tokom testiranja, kako u razvojnim tako i u produkcionim okruženjima, kroz sitne korekcije parametara unaprede performanse samih baza podataka.

Sa druge strane, sistemi sa složenom logikom često zahtevaju posebne eksperte koji umeju da ih poprave i unaprede. Imajući u vidu da su aplikacije različite, da bi takve sisteme bilo moguće efektivno održavati, postoji potreba za ručnim podešavanjem konfiguracionih parametara. U radu [23] iznesena je činjenica da operativni troškovi kod velikih relacionih baza podataka iznose čak i do 50% ukupne investicije u softverski projekat, dok administratori baza podataka provedu oko 25% vremena u kasnijem radu na ručnom doterivanju i poboljšavanju performansi relacionih baza podataka.

U okruženju kakvo je relaciona baza podataka u klaudu, korisnik ili administrator baze podataka vrlo često može biti nesvestan postojanja problema. Takođe, može i da naiđe na neki konkretan problem sa relacionom bazom podataka. Uglavnom se u tom slučaju otvara zahtev za pomoć sistemske podrške. Imajući u vidu vrlo veliki broj relacionih baza u klaud okruženju, potrebno je pojednostaviti rad kako samim korisnicima i administratorima baza podataka tako i inženjerima sistemske podrške. Potrebno je omogućiti lakšu i bržu spoznaju o tome da li u jednoj ili više relacionih baza podataka postoji problem ili ne. U slučajevima kada je to moguće, teži se i potpunoj automatizaciji celog procesa gde se za određeni poznati problem može sprovesti i korektivna akcija koja otklanja negativne efekte zapaženog problema.

U slučaju da inženjeri sistemske podrške pružaoca klaud servisa koji istražuju problem nemaju odgovarajuću pomoć sistema, vrlo je verovatno da korisnici neće biti na vreme obavešteni na koji se način problem može rešiti, čime će zadovoljstvo korisnika klaud servisom biti ugroženo. Sledeći primer pruža bolji uvid u složenost problema koji se mogu sresti u praksi. Samo tokom 2017. godine, na osnovu internih izvora koji su navedeni u radu [24], kompanija Microsoft je po svakom zahtevu za pomoć investirala prosečno 7 radnih sati inženjera sistemske podrške. Tokom tih 7 sati korisnikovi administratori relacionih baza podataka nisu bili u mogućnosti da samostalno reše problem bez pomoći tehničke podrške pružaoca klaud servisa.

Takođe, veoma važan aspekt je mogućnost skaliranja iz aspekta praćenja većeg broja baza podataka u ***PaaS*** odnosno ***DBaaS*** okruženju. Nekada su se administratori podataka bavili pojedinačnim i specifičnim istraživanjem problema u relacionim bazama podataka. Danas je situacija nešto drugačija. Zbog količine relacionih baza podataka, pojavila se potreba i za sistemom koji bi omogućio jednostavno praćenje rada velikog broja baza podataka. Pored toga, ukoliko do problema dođe, takav sistem bi trebalo da omogući i brzo detektovanje i razumevanje problema.

Postojeća komercijalna rešenja, o kojima će biti više reči u narednom poglavlju, imaju za cilj da omoguće korisnicima naprednu podršku tokom praćenja rada SQL servera. Takva rešenja omogućavaju administratorima relacionih baza podataka da efikasno obavljaju svoje zadatke prilikom održavanja jednog ili više servera. Iako ovakva rešenja postoje, određenu grupu problema prosečni administratori relacionih baza podataka, zbog nedostatka znanja i prakse, nisu u mogućnosti da reše bez dodatne pomoći.

Pored toga, kada su gornje granice hardverskog ili virtuelnog servera dostignute, aplikacija koja koristi relacionu bazu podataka može imati problema. Problemi u relacionoj bazi podataka koji se ogledaju u aplikativnom sloju posledično rezultuju u finansijskim gubicima i u narušavanju reputacije kompanije. Čak i kada iskusni administrator relacione baze podataka dobije ovakvu vrstu problema da reši, do pronalaženja pravog rešenja mogu da prođu sati, a u nekim slučajevima i dani. Kada se ovakav slučaj posmatra unutar klaud okruženja, koje sadrži milione relacionih baza podataka, postojeće tehnike svakako ne mogu da se izbore sa tolikim obimom, što je navedeno kao motivacija za istraživanje i u radu [25].

Nasuprot tome, dosta problema koji se prijavljuju od strane korisnika relacionih baza u klaud okruženju se mogu smatrati prolaznim. U isto vreme, uzrok za detektovan problem može proisteći ili kao posledica neadekvatno napisanog koda na strani aplikacija ili kao posledica neoptimalne upotrebe relacione baze podataka. Ovakav slučaj je primećen i u radovima [25], [26] i [27] koji će detaljno biti obrađeni u narednom poglavlju.

U takvim slučajevima, dalje analize pokazuju ili da ne postoji problem sa sistemske strane ili da je došlo do povećanja upotrebe relacione baze podataka kao posledice povećanja upotrebe aplikacije koja koristi relacionu bazu podataka. U ovom kontekstu najbolji je primer internet prodavnice, čija internet stranica ili aplikacija pred novogodišnje i božićne praznike ima veću posećenost pa je samim tim kaskadno i veća upotrebljenost relacione baze podataka.

Mogućnost da se napravi razlika između pravih problema i problema koji su nastali od strane korisnika, kao posledica neke od korisničkih ili klijentskih akcija, vrlo često zahteva detaljnu analizu inženjera sistemske podrške. Ovako nešto u klaud okruženju zahteva značajno vreme.

Kao što je već napomenuto, kada bi postojao sistem koji bi umeo da razlikuje situacije i da savetuje inženjere podrške, direktna korist od postojanja takvog sistema bi se ogledala u smanjenju utrošenog vremena prilikom analiziranja problema. Zatim, upotreba ovakvog sistema omogućavala bi rano otkrivanje problema, a samim tim i formiranje mehanizama uzbunjivanja. Korisnicima klaud servisa postojanje takvih mehanizama bi pomoglo da se fokusiraju na popravljanje i unapređivanje problema iz njihovog domena i oni na taj način ne bi morali više da istražuju probleme za koje nisu stručni.

U suprotnom, ukoliko ovakav jedan sistem ne postoji, svaki od klijenata mora da napravi poseban sistem za praćenje i upozoravanje. Na kraju korisnicima je zaista važno da se bave uslugama koje oni pružaju, a da klaud servis za njih radi one zadatke koje su mu oni poverili.

Imajući to u vidu, ovaj rad ima za cilj da istraži naučnu oblast i predloži rešenje za predstavljene probleme. Rešenje u vidu sistema za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju (eng. ***Intelligent Insights***, skr. AID-TS sistem) treba da obavlja sledeće aktivnosti. Prvenstveno AID-TS sistem na osnovu dostupne statistike treba da prepozna da problem postoji u ograničenom vremenskom roku. Zatim, nad prepoznatim problemom, potrebno je izvršiti analizu koja treba da ponudi mogućnosti koje objašnjavaju zbog čega je problem nastao. Na kraju, dodatnom analizom mogućnosti treba da se dođe do tačnog uzroka detektovanog problema koji je potrebno potom proslediti korisniku AID-TS sistema.

3. Pregled pristupa istraživanju i analiziranju problema u servisima

Postojeća komercijalna rešenja orijentisana na servisno okruženje koje kontrolišu i održavaju korisnici, pre svega, imaju za cilj da omoguće korisnicima lakšu podršku tokom praćenja rada različith servisa. Takva rešenja, kao što je već diskutovano u prethodnom poglavlju, omogućavaju administratorima sistema da efikasno obavljaju svoje zadatke prilikom održavanja jednog ili više servera. U slučaju manjih okruženja koja uglavnom kontrolišu korisnici, u akademskoj zajednici pravac istraživanja je uglavnom usmeren na relacione baze podataka čiji je programski kod javno dostupan.

Većina rešenja u klaud okruženju potiče iz komercijalne sredine gde velike kompanije, pružaoci servisnih usluga, rešavaju neke od prethodno pomenutih problema. Takođe, neka od komercijalnih rešenja su opisana u naučnim radovima na konferencijama i u stručnim časopisima. Kod komercijalnih rešenja postoji nekoliko dobro poznatih proizvoda, koji pokrivaju neke aspekte istraživanja i razumevanja problema u relacionim bazama podataka. Isto tako, u široj naučnoj oblasti (na primer, oblast mreža i mrežnih protola ili oblast reklamiranja putem globalne mreže), postoje akademski radovi koji su posvećeni rešavanju sličnih problema.

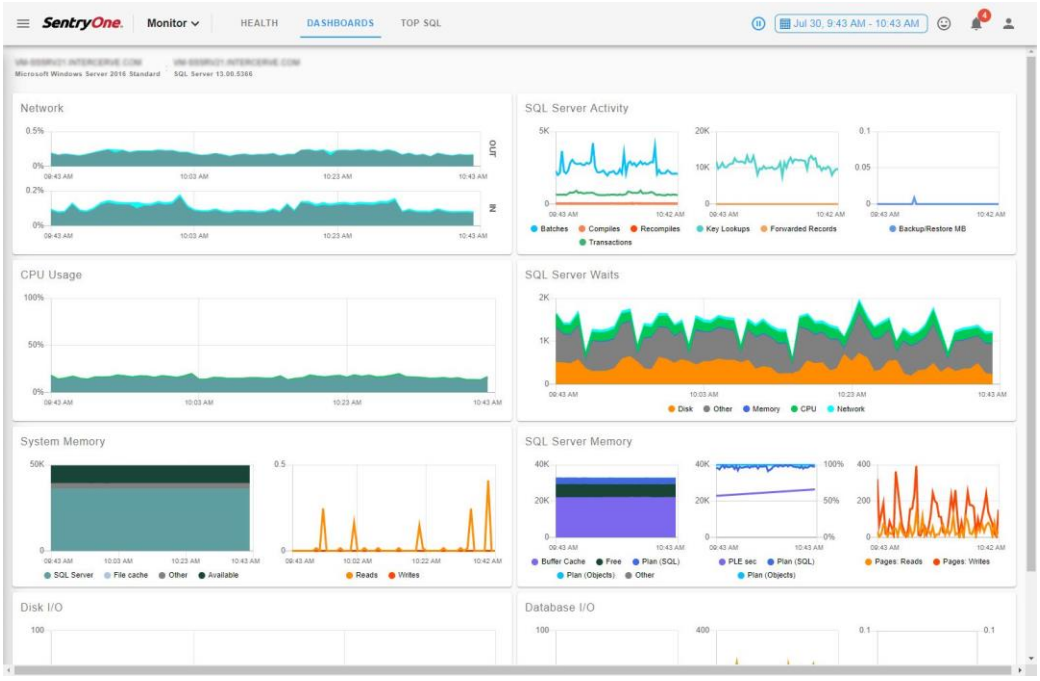
U nastavku poglavlja, prvo se izlažu analizirana komercijalna rešenja koja se koriste u okruženjima koje kontrolišu korisnici a zatim rešenja pronađena u otvorenoj literaturi. Potom su predstavljena relevantna istraživanja koja se tiču kako uopštenog servisnog okruženja iz bliskih naučnih oblasti tako i klaud servisnog okruženja.

3.1. Servisi u okruženju koje kontrolišu korisnici

U okruženju koje kontrolišu korisnici, među komercijalnim rešenjima koja služe za praćenje funkcionisanja relacionih baza podataka samo neka od njih uključuju i aspekte daljeg istraživanja problema. Sa druge strane, u akademskim krugovima se najviše istražuju upotrebe tehnika mašinskog učenja i različitih statističkih modela za detekciju i razumevanje problema. Takođe, akademski radovi analiziraju relacione baze podataka čiji je programski kod javno dostupan iz objektivnih razloga, ali to svakako ne umanjuje njihov značaj, pa su za ovaj rad izuzetno interesantni.

3.1.1. SQL Sentry Performance Advisor - alat za praćenje rada Microsoft SQL servera

SQL Sentry Performance Advisor [28] predstavlja jedno od komercijalnih rešenja namenjeno za praćenje funkcionisanja i održavanje Microsoft SQL servera. On je projektovan za tradicionalno okruženje gde se SQL server koristi na infrastrukturi koja pripada korisniku i poseduje bogat grafički interfejs predstavljen na slici 3.1. Na levoj strani komandne table aplikacije moguće je pratiti informacije poput iskorišćenja mrežne kartice, centralnog procesora i radne memorije koji su vezani za sam sistem na kom je instaliran Microsoft SQL server. Desna strana komandne table sadrži informacije vezane za sam SQL server. U prvom redu se nalaze podaci o aktivnosti samog SQL servera kao što su ukupan broj zahteva, transakcija i pretraživanja ključeva kao i količine arhiviranih podataka u megabajtima. Zatim, u narednim redovima se nalaze statistike i vremena čekanja grupisane po tipu resursa na koje zahtevi unutar SQL servera čekaju. Zauzetost memorije grupisanu po tipu upotrebe je moguće pratiti u trećem redu.



Slika 3.1. Grafičko okruženje aplikacije SQL Sentry One (slika preuzeta iz [28]).

Pogodnosti koje nudi ovaj alat za praćenje rada SQL servera se mogu iskoristiti i u klaud okruženju. Korisnik može da prati aktivnost SQL servera, vremena čekanja i upotrebljenost memorije. Na posebnoj stranici aplikacije moguće je pratiti aspekte koji su od značaja za same upite koji se najviše izvršavaju ili najduže čekaju.

Pored stranica za praćenje rada, proizvod poseduje i alate koji omogućuju istraživanje dešavanja u neposrednoj istoriji i konfigurisanje obaveštenja u slučaju određenih problema koje se vrši unošenjem granica za određenu metriku.

3.1.2. Dell Spotlight - alat za praćenje rada SQL servera

Sličan alat za praćenje rada Microsoft SQL servera napravila je i kompanija Dell pod nazivom Spotlight [29]. Njegova osnovna prednost se ogleda u intuitivnom grafičkom interfejsu za kontinuirano nadgledanje funkcionisanja jednog ili više sistema prikazanom na slici 3.2. Sa leve strane korisnik može da izabere odgovarajući SQL server, nakon čega sa desne strane dobija sve osnovne informacije o njegovom funkcionisanju.

Interfejs je hijerarhijski podeljen tako da se praćenje rada prvo može posmatrati iz perspektive aplikacije, odnosno praćenjem broja aktivnih konekcija, sesija i upita koji su uspostavljeni. Potom je predstavljena veza između ulaznih zahteva i opterećenja procesora, radne memorije i na kraju, jednog ili više diskova na kome se podaci nalaze. Pored funkcionalnosti za praćenje rada sistema, putem predstavljenog interfejsa moguće je konfigurisati granice za određenu metriku nakon kojih se pozivaju odgovarajući mehanizmi obaveštavanja ili uzbunjivanja.

Slika 3.2. Grafički interfejs aplikacije „DELL Spotlight” (slika preuzeta iz [29]).

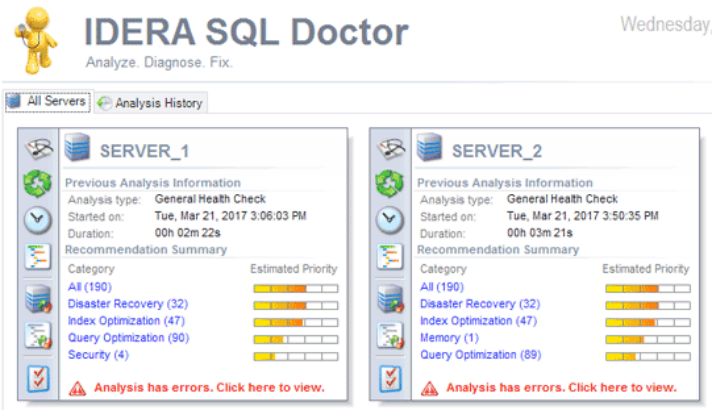
Za razliku od alata SQL Sentry, Dell Spotlight se može upotrebiti i za servere relacionih baza podataka koji su zasnovani i na drugim proizvođačima, a ne samo na rešenju kompanije Microsoft, što predstavlja značajnu prednost za korisnike koji poseduju relacione baze podataka od različitih proizvođača.

3.1.3. Idera SQL Diagnostic Manager - alat za praćenje rada Microsoft SQL servera

Nešto drugačije rešenje, u odnosu na prethodna dva je alat Idera SQL Diagnostic Manager [30]. On omogućava osnovne tehnike za praćenje rada i dijagnostifikovanje potencijalnih problema tokom rada Microsoft SQL servera. Ovaj alat takođe nudi i mogućnost praćenja rada infrastrukture na kojoj se Microsoft SQL server nalazi. Praćenje rada infrastrukture se ipak samo odnosi na tradicionalnu infrastrukturu korisnika i nije moguće koristiti ovaj alat za praćenje klaud infrastrukture.

Sa druge strane, upotrebom ovog alata moguće je pratiti i stanje na aplikativnom nivou. Pod tim se podrazumeva kompletno praćenje svih pojedinačnih upita i grupe upita koji pripadaju određenim procedurama. Upotrebom funkcionalnosti događaja (eng. ***events***) koju Microsoft SQL server nudi moguće je pratiti konkretne događaje i doći do informacija kao što su „koji upit blokira izvršavanje drugog upita" i slično.

Alat poseduje i funkcionalnost ***SQL Doctor*** koja služi za otkrivanje problema sa performansama. Upotrebom ove funkcionalnosti, kao što se može videti na slici 3.3., korisnik može da identifikuje neoptimalne planove koji se koriste za izvršavanje upita, indekse koje je moguće optimiziovati i različite probleme sa memorijom i probleme iz domena bezbednosti.



Slika 3.3. Grafički interfejs ***SQL Doctor*** aplikacije Idera SQL Diagnostic Manager (slika preuzeta iz [31]).

Deo rešenja ***SQL Doctor*** koji indetifikuje neoptimalne planove je vrlo sličan rešenju koji je kompanija Microsoft implementirala u tradicionalnom i klaud okruženju pod nazivom ***Automated plan regression correction*** [32].

Ograničenje ovog alata je činjenica da korisnik mora sam da specificira šta želi da prati, koji je potencijalni vremenski period u kome korisnik smatra da je došlo do problema, i slično. Može se zaključiti da je alat namenjen naprednim korisnicima i administratorima relacionih baza podataka, koji imaju duboko razumevanje o tome kako relacione baze podataka funkcionišu i koji se signali mogu pratiti. Za korisnike koji se tek upoznaju sa relacionim bazama podataka i koji nemaju duboka tehnička znanja upotreba ovog alata može biti izazov.

3.1.4. Alat Brenta Ozara za praćenje rada Microsoft SQL servera

Najpoznatiji alat koji je ujedno i osnovni ekspertski sistem za istraživanje i analizu Microsoft SQL servera je zasnovan na dugogodišnjem iskustvu administratora relacionih baza podataka Brenta Ozara [33]. Nakon integracije i instalacije sa određenim Microsoft SQL serverom ovaj alat prati najvažnije statistike i informacije. Na osnovu praćenih podataka moguće je identifikovati skup poznatih problema koje alat ume da prepozna. Takođe, alat daje osnovne informacije o radu samog Microsoft SQL servera kao celine ili pojedinačne relacione baze podataka.

Sa druge strane, nedostatak ovog alata je činjenica da, ako alat nije instaliran u periodu kada se problem pojavio, nije moguće retroaktivno saznati šta je dovelo do samog problema koji se istražuje. Nedostatak informacije o promenama koje su dovele do problema značajno može otežati ceo proces istraživanja.

3.1.5. Podešavanje optimalne konfiguracije servera relacione baze podataka putem mašinskog učenja

Upotreba tehnika mašinskog učenja u relacionim bazama podataka je istraživana u radu [23]. Glavni cilj ovog rada je pronalaženje optimalnih konfiguracionih parametara u relacionim bazama podataka MySQL i PostgreSQL. Sistem je napravljen tako da koriguje vrednosti konfiguracija od 4 do 16 parametara kako bi postigao optimalno podešavanje za odgovarajući tip upita. Pod paramaterima se podrazumevaju neki od sledećih:

- veličina prostora za privremeno skladištenje podataka (eng. ***buffer pool***),
- veličina prostora na disku za transakcioni log (eng. ***log file size***),
- parametar za izbor planova za izvršavanje upita,
- broj aktivnih niti koje mogu raditi u paraleli.

Analizirajući postignute rezultate u okviru ovog istraživanja prijavljeno je vreme od 45 minuta do čak 400 minuta koje je potrebno za pronalaženje optimalne konfiguracije kod relacionih baza podataka MySQL i PostgreeSQL.

U kontekstu Microsoft SQL servera u tradicionalnom okruženju, neki od spomenutih parametara su već dinamički konfigurisani od strane samog servera na osnovu grupe korisničkih upita koji se izvršavaju. Sa druge strane, u klaud okruženju neke od dinamičkih konfiguracija se i dalje koriste, ali generalno odgovornost optimalne konfiguracije u odnosu na hardver na kom je instaliran Azure SQL server je zadatak softverskih inženjera kompanije Microsoft.

3.1.6. Sistem za detekciju i razumevanje problema u MySQL serveru

Trend rešavanja problema koji se javljaju prilikom istraživanja i optimizacije performansi u relacionim bazama podataka je predstavljen i u radovima koji opisuju sisteme DBSeer [34] i DBSherlock [35]. Pristup koji je prezentovan u radu o sistemu DBSeer predstavlja rešenje za prikupljanje podataka, automatsko grupisanje sličnih upotreba relacionih baza podataka i detekciju problema sa performansama.

Nastavak istraživanja je prezentovan pod nazivom DBSherlock, gde je glavna tema usmerena ka otkrivanju i razumevanju problema koji su detektovani. U ovom radu je objašnjeno deset različitih tipova grupa problema koje sistem može da detektuje. Ti problemi su neki od onih koji se mogu pojaviti u relacionim bazama podataka zasnovanim nad MySQL serverom.

Vrlo je važno napomenuti da svi detektovani problemi objašnjeni u ovom radu proizilaze iz sintetičkog načina upotrebe. Primećeni problemi koje ovi radovi rešavaju su još jedna indikacija da je istraživanje unapređenja performansi u relacionim bazama podataka jedan izuzetno zahtevan zadatak. Kada se posmatra produkciono okruženje koje opslužuje milione relacionih baza podataka u odnosu na sintetičko testno okruženje, složenost analiziranih problema je za red veličine veća, što je takođe zaključeno i u radu [25].

3.2. Servisi u klaud okruženju

Sa pojavom klaud servisne paradigme zasnovane na distribuiranim infrastrukturama, dobro poznati problemi slični održavanju mreža i razumevanju problema koji se u takvom okruženju dešavaju pojavili su se ponovo. Problemi nisu vezani samo za računarske mreže već i za druga okruženja koja su sačinjena od uređaja koji u fizičkom smislu ne moraju biti na jednom mestu a koji na višim nivoima apstrakcije interaguju međusobno na neki način. Tokom istraživanja postojećih servisa u klaudu, pažnja je prvenstveno bila usmerena ka velikim klaud servisima. Napravljena je i analogija sa ekspertskim sistemima koji su se koristili u servisima koji su srodni klaud okruženju po svojoj složenosti. Tom prilikom posebna pažnja je usmerena na istraživanje u domenu održavanja i kontinuiranog unapređivanja takvih sistema.

3.2.1. Klaud relaciona baza podataka Azure SQL kompanije Microsoft

Kompanija Microsoft nudi nekoliko različitih rešenja za relacione baze podataka. Oni su dostupni i u ***IaaS*** i u ***PaaS*** obliku. Zbog bitnog aspekta pojednostavljenja upotrebe krajnjim korisnicima, kompanija Microsoft više investira u svoje ***PaaS*** rešenje u odnosu na ***IaaS***.

Azure SQL relaciona baza podataka u klaud okruženju se smatra prvim rešenjem ***PaaS*** tipa [19]. Projektovana je kao dostupan i pouzdan sistem koji eliminiše potrebu za održavanjem i same infrastrukture i samog SQL servera. U poređenju sa tradicionalnom verzijom SQL servera koju je moguće instalirati na korisničkoj infrastrukturi, postojanje SQL servera i relacionih baza podataka u klaud okruženju omogućava korisnicima da dobiju najažurniju verziju SQL servera sa svim novim funkcionalnostima uz svako novo ažuriranje koje se automatski sprovodi od strane pružaoca klaud usluga.

SQL server instaliran na infrastrukturi korisnika obično pokreće stotine relacionih baza podataka, dok je u okruženju kakvo je Azure SQL taj broj mnogo veći i meri se u stotinama hiljada i milionima relacionih baza podataka. Slično crnoj kutiji koja se upotrebljava u avionima, u ovakvom okruženju se čuvaju i prate svi važni parametri i statistike koje se tiču relacionih baza podataka [9].

Iako je osnova Azure SQL klaud okruženja sam SQL server, u klaud verziji servisa, pored SQL servera, instalirani su i dodatni servisi koji nisu deo tradicionalne verzije SQL servera koju korisnici instaliraju na sopstvenoj infrastrukturi. Dodatni klaud servisi se odnose na aplikacije koje omogućavaju automatsko čuvanje i arhiviranje podataka do 30 dana. Pored arhiviranja, sačuvane podatke u arhivi je moguće upotrebiti i za vraćanje stanja relacione baze podataka kroz vreme u neku izabranu tačku u istoriji (eng. ***point-in-time restore***). Osim toga, postoje i mogućnosti repliciranja podataka između više geografski udaljenih regiona. Ta usluga se prvenstveno koristi u slučaju otkaza celog regiona do kojeg može doći usled različitih faktora.

Telemetrija koju Azure SQL okruženje prikuplja je sačinjena od različitih podataka. Najvažniji podaci se prikupljaju sa periodom od nekoliko sekundi, ali zbog velike količine podataka nije ih moguće čuvati tokom dužeg vremenskog perioda. Najpoznatiji primer podataka koji se čuvaju na nivou minuta su statistički podaci o izvršavanju upita [36]. Informacije kao što su nazivi kolona, tabela i ostalih objekata u relacionim bazama podataka se uzimaju i čuvaju do dva puta na dan. Svi tako prikupljeni podaci se čuvaju u skladu sa evropskom i drugim svetskim regulativama (***General Data Protection Regulation***, skr. GDPR) [37]. Namena ovih regulativa je prvenstveno usmerena ka očuvanju privatnosti i zaštiti korisnikovih podataka kako bi se izbegle zloupotrebe.

Relaciona baza podataka Azure SQL već nudi neka rešenja za optimizaciju i razumevanje problema iz oblasti performansi, kao što su ***SQL database advisor*** [38] i ***Schema Issue advisor*** [39]. Glavni cilj pomagača (eng. ***advisor***) je da smanje pritisak na administratore baza podataka i inženjere podrške, tako što će ponuditi korisniku informacije i korisne savete kako da reše situacije koje oni zapaze u toku rada. U isto vreme, uz pomoć takvih pomagača, koji su zasnovani na različitim statističkim modelima, iskustvo korisnika sa Azure SQL platformom kao celinom je značajno poboljšano, a neželjeni efekti su smanjeni.

Kao što je već napomenuto, rešenje kompanije Microsoft nudi automatsku detekciju i optimizaciju izbora idealnog plana za izvršavanje upita i u tradicionalnoj i u klaud verziji SQL servera. Pod tim se podrazumevaju relacione baze podataka koje koriste verziju SQL servera počev od 2019. godine. Zbog svega pomenutog, kompanija Microsoft tvrdi da je Azure SQL relaciona baza podataka prva inteligentna klaud relaciona baza podataka [40].

3.2.2. Klaud relaciona baza podataka RDS kompanije Amazon

Kompanija Amazon takođe poseduje svoju verziju klaud relacione baze podataka [41]. I u ovom slučaju, slično kompaniji Microsoft, kompanija Amazon u svojoj ponudi poseduje i ***IaaS*** i ***PaaS*** rešenje. Kompanija Amazon je takođe više fokusirana na ***PaaS*** rešenje imajući u vidu da se na tržištu to smatra trendom budućnosti.

U slučaju Amazonovog rešenja, implementacija je više fokusirana na omogućavanje rada izuzetno velikih relacionih baza podataka gde jedna fizička mašina nije dovoljna za opsluživanje, već je to potrebno uraditi na desetinama drugih. Kada se razmatra aspekt praćenja i održavanja relacionih

baza podataka, kompanija Amazon je usredsređena na aspekte koji se tiču jednostavnog predstavljanja podataka i prilagođavanja platforme postojećim alatima za praćenje baza podataka. Takav pristup omogućava administratorima baza podataka da prate šta se dešava sa relacionim bazama podataka na tradicionalni način.

3.2.3. Klaud relaciona baza podataka kompanije Alibaba

Rad koji opisuje Alibaba Cloud native relacione baze podataka [21] takođe govori o aspektima relacionih baza prilagođenih klaud okruženju. U ovom radu detaljno je opisana arhitektura kao i drugi aspekti njihovog sistema. Posebno je interesantno objašnjenje upotrebe inteligencije.

Autor napominje kako relacione baze podataka u klaud okruženju imaju dominantnu ulogu u svetu ***IaaS*** i ***PaaS*** servisa, prvenstveno sa aspekta zahteva od strane korisnika, a samim tim i iz perspektive izazova sa kojim se softverski inženjeri koji rade na razvoju platforme susreću. Slično kao što se očekuje da uskoro automobili uopšte neće imati vozače, korisnici od takve platforme očekuju da će u narednoj dekadi platforma postati autonomna do te mere da korisnici ne moraju posedovati napredna znanja o relacionim bazama podataka.

Rad ukazuje i na poteškoće prilikom primene konvencionalnih metoda koje obavljaju administratori baza podataka usled velikog broja relacionih baza podataka u klaud okruženju. Glavni izazovi prepoznati u ovom radu se odnose na automatsku detekciju i analizu problema i kako da se u što kraćem roku oni otklone.

Od ostalih radova istraživača iz kompanije Alibaba najinteresantniji iz pogleda ovog istraživanja je rad [25]. On razmatra uzroke problema koji se dešavaju u relacionim bazama podataka kada se određeni upiti izuzetno sporo izvršavaju. U radu je istaknuta važnost pronalaženja usporenih upita za dugotrajnu stabilnost samog rada relacione baze podataka. Takođe je napomenuta činjenica da je zadatak dijagnostifikovanja i pronalaženja uzroka problema izuzetno težak, ali ujedno i sredstvo bez koga se ne može u složenom okruženju kakva je relaciona baza podataka u klaudu okruženju.

3.2.4. Softverski paket Enterprise Manager 13c Cloud Control kompanije Oracle

Enterprise Manager 13c Cloud Control [42] je softverski paket kompanije Oracle čiji je glavni cilj da olakša procese oko instaliranja, konfigurisanja i održavanja klaud rešenja. Klijenti kompanije Oracle su uglavnom srednja ili velika preduzeća.

Dva najznačajnija aspekta ovog proizvoda su ***Automatic database diagnostic monitor*** (skr. ***ADDM***) [43] i ***Automatic SQL tuning advisor*** [44], čija je glavna svrha da detektuju i optimizuju rad samog servera relacione baze podataka. Komponenta ***ADDM*** analizira prikupljene podatke o iskorišćenosti procesora, radne memorije, upotrebe diska, konfiguracije relacione baze podataka i konkurentnosti upita i na osnovu njih donosi zaključke koje potom čuva u odgovarajućoj sistemskoj tabeli nad kojom korisnik može da izvršava upite. ***Automatic SQL tunning advisor*** je deo Oracle SQL servera čija je glavna uloga da osigura izbor najbolje plana za izvršavane upite.

Iako kompanija Oracle pokušava da pokrije jedan vrlo specifičan tip aplikacija i način njihove upotrebe, cilj rada prezentovanog u [24] i ovog rada teži rešavanju generičkog slučaja upotrebe koji se može susresti u klaud okruženju.

3.2.5. Ekspertski sistem STARKEEPER Network troubleshooter

Jedan od prvih pristupa rešavanju problema upotrebom ekspertskih sistema [45] datira još iz 70-tih i 80-tih godina prošlog veka. U to vreme istraživanje i analiziranje zagušenja i različitih defekata na mreži bio je jedan od značajnih izazova. Rešenje koristi ekspertski sistem koji ima za cilj da mrežnim administratorima koji nemaju dovoljno iskustva prilikom analize problema u domenu mreža i mrežnih protokola pomogne i prenese domensko znanje.

Opis problema i složenost koji proističu iz analiziranja sistema, praćeni opštim principima koji su iskorišćeni u ovom rešenju mogu se primeniti za bilo koju vrstu istraživanja velikih sistema i ne moraju biti direktno vezani za domen mreža i mrežnih protokola. Sličan ekspertski sistem je razvijen i od strane telekomunikacione kompanije AT&T [26]. On je zasnovan na ekspertskim sistemima koji upotrebljavaju pravila odlučivanja [46].

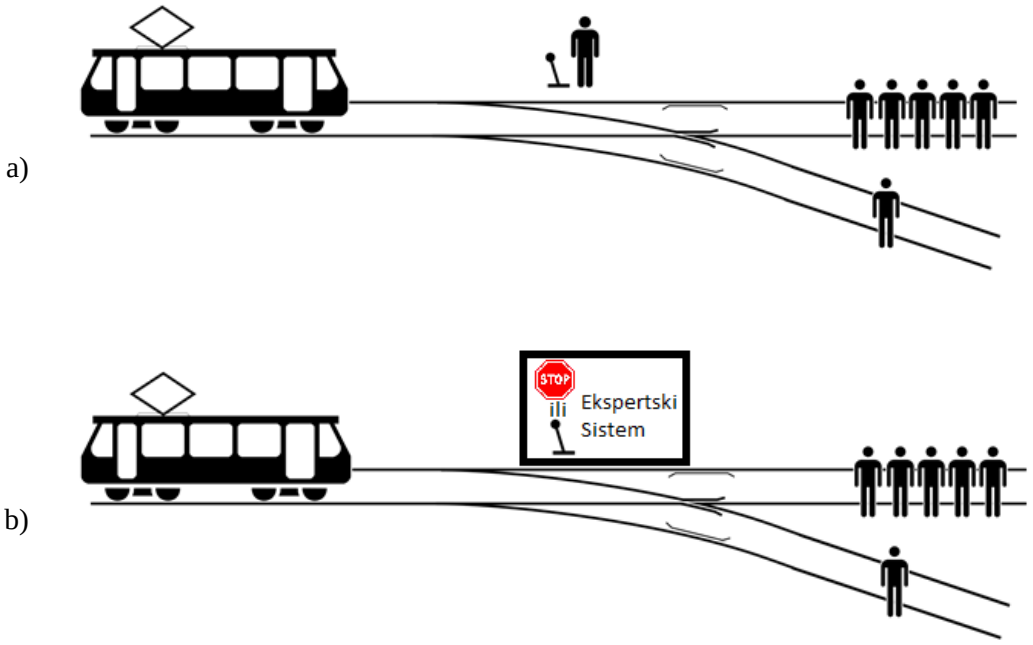
Ekspertski sistem zasnovan na pravilima implementira konkretna znanja iz posmatranog domena. Svako pravilo predstavlja jedan ili više uslova. Kada je za ulazne parametre rezultat logički tačan, smatra se da je pravilo zadovoljeno i ekspertski sistem tada donosi odluku ili izvršava konkretnu akciju koja je definisana za dato pravilo.

3.2.6. Ekspertski sistem Aware

Iako se ekspertski sistemi prvi put pominju pre više od 40 godina, njihova upotreba je i danas prisutna i smatraju se efektnom tehnikom za rešavanje određenih problema. U jednom od skorijih radova koji opisuje sistem ***Aware*** [47], ekspertski sistem je korišćen kao glavna komponenta za donošenje odluka.

U ovom radu, opisani sistem je definisan kao skup različitih senzora i detektora. Njihova uloga je slična ulozi čula koje poseduju životinje i ljudi. Kod živih bića, njihova čula im omogućavaju da osete, razviju svest o trenutnim dešavanjima oko njih i predvide buduća dešavanja. Pošto neki od nivoa svesti zapravo implementiraju ekspertski sistemi kao sistem za donošenje odluka, spomenuti koncept se može smatrati i veštačkom inteligencijom [48].

Za razliku od modela mašinskog učenja, čiji je rezultat uvek u nekoj formi, ekspertski sistemi mogu da implementiraju i pravila koja sprečavaju donošenje određenih odluka ili objavu rezultata. Kao što je prikazano na slici 3.4., koja sugeriše na etički problem diskutovan u [49], veoma je važno da u određenim situacijama umesto donošenja odluke o izboru između dve opcije, sistem ne izabere ni jednu od ponuđenih opcija.



Slika 3.4. a) Osnovni „problem tramvaja“ i skretnice (slika preuzeta iz [50]).
b) „Problem tramvaja” u slučaju kada odluku donosi ekspertski sistem.

Na taj način sistem se orijentiše na treću opciju u kojoj se ne donosi ni jedna od dve podjednako loše odluke. Razmatrajući takav pristup, moguće je upotrebiti više različitih modela mašinskog učenja gde se rezultati prosleđuju ekspertskom sistemu koji donosi odluku koji će se rezultat od ponuđenih razmatrati, ili pak u potpunosti eliminisati od daljeg procesiranja unutar šireg sistema.

3.2.7. Istraživanje problema u klaud okruženjima

Danas, kada se aplikacije i servisi instaliraju u klaud okruženju koje se nalazi na udaljenim lokacijama, problem održavanja i istraživanja problema ukoliko do njih dođe u samim aplikacijama zahteva značajno vreme, a samim tim predstavlja i veoma skupu aktivnost. U slučaju kada servis ne radi delimično ili potpuno, njegovim korisnicima se ne isporučuje određena usluga, što predstavlja ozbiljan problem koji je opisan u [51]. Citat iz spomenutog rada verodostojno opisuje stepen složenosti, „sa velikom količinom podataka koji se prikupljaju i prate, istraživanje i pronalaženje problema se svodi na potragu za iglom u plastu sena”. Metodologija koja je predstavljena u ovom radu podrazumeva:

- detekciju i praćenje ponašanja sistema,
- dijagnostifikovanje problema ukoliko do problema dođe,
- otklanjanje problema ukoliko je njegova automatizacija moguća.

Detekcija problema je obrađena upotrebom dva pristupa, reaktivnim i proaktivnim, gde je referisano i izloženo nekoliko radova iz spomenutih oblasti. Kod diagnostifikovanja problema, autori su prepoznali tri različita skupa pristupa. Prvi se odnosi na pronalaženje veza između hardvera i softvera kako bi se uspostavilo uzročno posledično stablo uticaja, čime se uticaj hardverskih

problema na softver može lako detektovati. Zatim, analizirane su tehnike korelacije između različitih signala kao potencijalno rešenje i na kraju su izloženi radovi koji u homogenim sistemima upotrebljavaju tehnike pronalaženja sličnosti i razlika za dijagnostifikovanje problema.

Izuzetno je važno u ovom radu što je prepoznato da se detekcija i dijagnoza problema uglavnom odvijaju u odvojenim koracima. Rad daje i pregled oblasti automatskog otklanjanja problema i različitim tehnikama koje su u otvorenoj literaturi dostupne.

3.2.8. Automatizacija rešavanja problema u klaudu korišćenjem mašinskog učenja

Automatizacija procesa rešavanja problema, koje su prijavili korisnici, sa željom da se smanji pritisak na inženjere i administratore klaud servisa je istraživana i od strane kolega iz kompanije Microsoft u radu [27]. Tom prilikom upotrebljene su tehnike mašinskog učenja. Na osnovu iskustva koje je izneseno u ovom radu, oslanjanje isključivo na tehnike mašinskog učenja može biti izuzetno izazovno i samim tim realizacija konačnog rešenja može biti neizvodljiva. To prvenstveno važi za situacije kada je početni skup obeleženih podataka izuzetno mali.

U slučaju kada postoji veliki broj signala koji odražavaju ponašanje sistema, upotreba tehnika mašinskog učenja predstavlja još teži zadatak. Relaciona baza podataka, sama po sebi emituje različite signale i u tradicionalnom okruženju, dok u klaud okruženju kao što je Azure SQL relaciona baza podataka, stepen složenosti je za nekoliko redova veličine veći. Samim tim i definicija i treniranje modela zasnovanih na mašinskom učenju je daleko zahtevnija i izazovnija, što je takođe zaključeno i u [25].

3.2.9. Sistem za analiziranje i detekciju problema Explain it!

Istraživači iz kompanije CISCO Tetration Analytics takođe su istraživali oblast klaud tehnologija. Za potrebe klaud servisa, oni su projektovali deklarativni sistem za analizu i detekciju problema nad podacima u obliku vremenske serije pod nazivom Explain It! [52].

Klaud servis koji je analiziran u ovom radu prikuplja milione događaja, zapisa, statistika od različitih servisa i komponenti. Ovakav pristup postoji i u Azure SQL klaud servisu gde SQL server emituje podatke, a klaud servis prikuplja razne događaje i statistike o ponašanju servera. Suočavanje sa sličnim i različitim vremenskim serijama kao i pronalaženje korelacija između njih je i u ovom radu prepoznato kao vrlo izazovan problem.

Ovaj rad takođe daje pregled primera iz realnog okruženja jednog sistema kao i studije slučaja koje su primenjene tokom evaluacije pomenutog sistema u produkciji. U zaključku se navodi kako automatizacija analize problema može biti izuzetno zahtevna, a posebno u slučaju kada je celokupan proces primenjen nad velikom količinom informacija u kojoj nema prostora za naknadnu ljudsku intervenciju tokom rada sistema.

3.2.10. Manuelna intervencija u masivnim sistemima predikcije

Potreba za intervencijama od strane ljudi prilikom planiranja, realizacije i daljeg rada sistema predikcije primećena je u radu [53]. U opširnom obrazloženju skrenuta je posebna pažnja na potrebu da čovek proveri i potvrdi odluke koje donose tehnike veštačke inteligencije. Sistemi koje donose odluke upotrebom veštačke inteligencije moraju da poseduju robustne mehanizme predikcije. Oslanjanje na jedan model u tom slučaju nije preporučljivo i teži se implementaciji više različitih modela. Njihovim sinergističkim pristupom se potom dolazi do krajnjeg rešenja sačinjenog od višestruko proverene predikcije za koju se, sa visokim procentom statističke sigurnosti, može tvrditi da je ispravno detektovana.

Takođe, veoma važan segment, prilikom osmišljavanja i uspostavljanja arhitekture takvih sistema je buduće održavanje i održanje kvaliteta predikcije tokom daljeg razvoja i upotrebe sistema. Prikazani zaključci su veoma važni za usmerenje istraživanja i izbor konačnog rešenja, imajući u vidu analiziranu oblast i njene sličnosti sa klaud okruženjima koji predstavljaju jednu vrstu masivnog sistema.

3.3. Rezime iskustava iz otvorene literature

Kao što se vidi iz analiziranih radova i dostupnih proizvoda, upotreba ekspertskih sistema, statističkih modela, nauke o podacima i upotreba tehnika mašinskog učenja najčešće predstavljaju odgovarajuća rešenja i odgovor za neke od izazova i problema koji su predmet rešavanja u okviru ovog rada.

Imajući u vidu prethodna istraživanja koja su bila usmerena na jedan mali segment korisničkih problema koji su predstavljeni u [39], a zatim sve obuhvatno prikazani u [24], tema ovog rada predstavlja nastavak istraživanja u istoj oblasti. Sličan pristup opisan je i u radu [51].

Složenost koja dolazi od veličine klaud servisa, gubici tokom perioda kada servisi nisu dostupni kao i veliki broj različitih vrsta upotrebe servisa, posebno iz perspektive istraživanja problema performansi u odnosu na tradicionalno rešenje su prepoznati kao ozbiljan problem i zahtevan zadatak i u radovima [25], [54] i [55].

Pokušaj upotrebe tehnike mašinskog učenja za rešavanje problema velikog broja zahteva za pomoć inženjerima podrške u klaud servisu predstavljen je u radu [27]. Na osnovu izazova prilikom realizacije konačnog rešenja usled malog skupa obeleženih podataka koji bi se upotrebio za modele mašinskog učenja zaključuje se da ovaj pristup nije pogodan za rešavanje problema koji se sreću u relacionim bazama podataka u klaud okruženju.

Upotreba više različitih modela u jednom robusnom sistemu prezentovana je u radu [53]. Glavna prednost ovakvog pristupa ogleda se u povećanju preciznosti usled postojanja više različitih modela koji mogu da primete probleme različitog tipa. Pristupi DBSeer [34] i DBSherlock [35] se takođe u prikazanoj arhitekturi zasnivaju na više statističkih modela za detekciju problema iz domena performansi relacionih baza podataka. Ovi radovi potvrđuju da je pristup upotrebe više statističkih modela ispravan, ali radi preciznije detekcije problema i bolje diskriminacije potrebno je definisati i implementirati dodatne statističke modele.

Sa druge strane, ekspertski sistemi se upotrebljavaju kao glavna komponenta za detekciju i razumevanje različitih situacija i problema u radovima [33], [45] i [47]. Ekspertski sistem na bazi pravila se pokazao izuzetno korisnim i može se upotrebiti kao komplementaran mehanizam za donošenje konačne odluke da li je nešto zaista problem i koji je problem tačno detektovan.

Sinergistički pristup kroz upotrebu različitih modela u više koraka i korišćenje ekspertskog sistema na bazi pravila upotrebljava tehnike korelacije signala i rezultata slično korelacijama koje se primenjuju za razumevanje vremenskih serija i problema koje su prikazane u radovima [51] i [52].

Konačni cilj ovog rada je formiranje kompletnog rešenja za efektivno detektovanje problema i njegovu analizu unutar klaud okruženja koje servisira stotine hiljada relacionih baza podataka. Informacije dobijene od strane sistema u tom slučaju treba da budu dostupne kako krajnjim korisnicima, tako i inženjerima sistemske podrške koji održavaju klaud servis.

Evaluacija predloženog rešenja treba da obuhvati korisničku upotrebu u stvarnom okruženju sa nizom različitih autentičnih tipova upotrebe baze podataka tokom dužeg vremenskog perioda. Cilj ovakvog pristupa evaluaciji je želja da se dokaže upotrebljivost i efikasnost ovakvog rešenja i u praksi.

4. Definisanje sistema za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju

Svaka relaciona baza podataka u klaud okruženju emituje različite signale u vidu statističkih podataka, informacija o greškama, vremena čekanja na različitim resursima, itd. Svi ti podaci se mogu koristiti za detekciju određenog tipa i određenih karakteristika ponašanja kako bi se razumela priroda upita koji se izvršavaju na određenoj konkretnoj relacionoj bazi podataka.

Data science modeli se sastoje od različitih statističkih modela čija je osnovna namena da detektuju određeno ponašanje na osnovu definisanog skupa signala. Konceptualno, izabran je tradicionalni pristup u odnosu na moderne tehnike mašinskog učenja. Takav pristup je primenjen prvenstveno iz više razloga. Pre svega, u okruženju kakvo je klaud servis, gde se nalazi i prati više stotina različitih metrika i signala koje dolaze od više od milion relacionih baza podataka, dok je u isto vreme obeleženi skup podataka izuzetno mali sa manje od 500 detaljno analiziranih slučajeva, tehnike mašinskog učenja su gotovo neupotrebljive. Sličan pristup gde je upotrebom više različitih klasifikacionih modela povećana preciznost predložen je i analiziran u [53]. Modeli za detekciju anomalija i njihovo klasifikovanje su takođe korišćeni i u radovima [21] i [25].

Nešto drugačiji pristup, koji se nije oslonio na automatsko detektovanje problema, je predstavljen u radu [52]. U ovom slučaju, period u kojem postoje problemi nije detektovan od strane sistema, već korisnik definiše period za koji smatra da je problematičan. Odgovornost sistema je zatim usmerena ka pronalaženju korelacija sa drugim signalima sa idejom da se na taj način olakša korisniku istraživanje samog problema. Ovaj sistem kroz nekoliko koraka usmerava korisnika na potencijalne putanje daljeg istraživanja.

Tokom istraživanja i rada na [24], pokušana je implementacija jednog modela koji bi mogao da detektuje i razume sve slučajeve, ali to nije bilo moguće zbog velike različitosti u načinima korišćenja relacionih baza podataka u klaud okruženju. Prvobitni model je iskorišćen za pronalaženje problematičnih relacionih baza podataka nad kojim je vršena dodatna analiza kako bi se problemi u potpunosti razumeli. Činjenica i da je početni skup obeleženih analiziranih slučajeva bio dosta mali je dodatno usmerila istraživanje u smeru tradicionalnog pristupa sa više statističkih modela.

Kao i u drugim velikim sistemima koji koriste veštačku inteligenciju, postoji potreba za analiziranjem rezultata samog sistema. U slučaju ovog rada, pojavila se potreba za dodatnim analiziranjem rezultata koji su dobijeni od više različitih statističkih modela. Krajnji cilj, zbog kojeg je izabran takav pristup je diktiran željom za dobijanjem što preciznijih i tačnijih analiza problema koji su primećeni. Zato je projektovan ekspertski sistem uz uključenje svih neophodnih pravila. U celokupnom sistemu za detekciju, ekspertski sistem donosi konačnu odluku šta je bitno i potrebno prikazati i prijaviti, kao i šta je uzrok, a šta je posledica.

Na osnovu prethodnog razmatranja, odlučeno je da se AID-TS sistem, koji je projektovan, implementiran, evaluiran i predstavljen u ovom radu, sastoji od dve glavne komponente:

- Statistički ***data science*** modeli
- Ekspertski sistem

4.1. Statistički modeli za inteligentno otkrivanje uzroka problema

Za detekciju i razumevanje ponašanja upotrebe relacionih baza podataka, dve grupe statističkih modela su definisane i predviđene u ovom radu:

- generički modeli,
- kategorizacioni modeli.

Generički modeli su napravljeni sa ciljem da detektuju bilo koju relacionu bazu podataka koja u odnosu na istoriju njenog funkcionisanja trenutno emituje signale koji pokazuju da se sa njom dešava nešto neočekivano. Stepen neočekivanog dešavanja se meri parametrom ozbiljnosti (eng. ***severity***).

Generički modeli, prilikom detekcije problema, pružaju sledeće informacije:

- naziv generičkog modela koji je detektovao problem,
- vremenski interval u kojem je problem detektovan,
- osnovne podatke o relacionoj bazi podataka,
- vrednost parametra ozbiljnosti.

Izuzetno je bitno napomenuti da su ovi modeli napravljeni prvenstveno sa ciljem da budu sveoubuhvatni, iako to može da se odrazi na preciznost. Konkretno, to znači da će modeli uhvatiti sve vrste anomalija, i one koje su maligne ali i one koje su benigne. Ovakav pristup može da detektuje jednostavno povećanje upotrebe i opterećenosti relacione baze podataka što je očekivano ako se sa korisničke strane poveća dolazni saobraćaj. Sa druge strane, ovim pristupom se mogu

detektovati razne vrste ozbiljnih usporenja i regresija u pogledu performansi jedne relacione baze podataka. Do ovakvih slučajeva dovode različiti faktori i o njima će biti više reči kasnije.

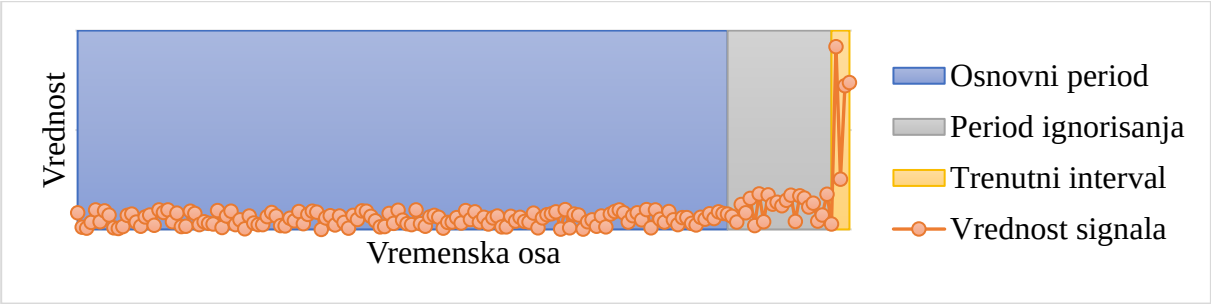
Parametar ozbiljnosti je jedan od ključnih parametara za dalje procesiranje. Bitno je da dve situacije, od kojih je jedna maligna a druga benigna, sistem može da razlikuje automatski. AID-TS sistem to čini tako što jedan deo anomalija isključuje ukoliko parametar ozbiljnosti ne prelazi određenu granicu. Smatra se da slučajevi koji su ispod navedene granice spadaju u grupu benignih anomalija i to su upozorenja koja se mogu ignorisati. Tačna vrednost te granice za svaki od generičkih modela se određuje detaljnom analizom na osnovu izlaza koji su dobijeni iz generičkih modela. Sa druge strane, ukoliko vrednost parametra ozbiljnosti prelazi granicu, problem u određenom vremenskog intervalu se dalje procesira od strane kategorizacionih modela.

Kategorizacioni modeli se koriste da uhvate konkretno ponašanje, odnosno tip neadekvatne upotrebe relacione baze podataka koji bi dao potencijalno objašnjenje za prvobitnu detekciju od strane generičkih modela. Iz perspektive nauke o podacima, za razliku od generičkih modela, oni se odlikuju se velikom preciznošću, ali sa malim stepenom obuhvatanja slučajeva koji imaju problem. To znači da za svaki interval u kome model detektuje određenu relacionu bazu podataka, sa velikom sigurnošću se ponašanje koje model treba da detektuje desilo ili se i dalje dešava. Sa druge strane, ovi modeli ne garantuju da će svaki slučaj, koji bi u idealnom sistemu trebalo da bude detektovan, zaista biti detektovan, pa se u tom aspektu oslanjaju na generičke modele.

Kategorizacioni modeli, takođe, kao rezultat obrade daju i parametar ozbiljnosti, ali u ovom slučaju njegova vrednost i interpretacija su drugačije u odnosu na generičke modele. Stepen ozbiljnosti kod kategorizacionih modela se opisuje kao mera korelacije između onog što je primetio generički model i signala koje posmatra i analizira konkretan kategorizacioni model. Što je korelacija između posmatranih signala veća, stepen parametra ozbiljnosti kategorizacionog modela je veći.

4.1.1. Generički modeli

Tokom istraživanja u produkcionom okruženju Azure SQL relacionih baza podataka, definisano je pet generičkih modela. Svi modeli funkcionišu po principu primećivanja značajnih promena, tj. anomalija. Vrednosti signala u trenutnom intervalu se porede sa vrednostima istog tog signala u prethodnom osnovnom periodu (eng. ***baseline period***). Osnovni period je vremenski interval u bliskoj istoriji tokom kojeg se smatra da relaciona baza podataka nije imala nikakvih problema. Slično pristupu u radu [52] na slici 4.1. je predstavljen posmatrani signal kroz duži vremenski period. Razlikuju se tri vremenska intervala u posmatranom periodu. Prethodni osnovni period se odnosi na skoriju istoriju. Kao što je već napomenuto, to je polazna tačka u odnosu na koju se vrši dalje poređenje. Vrednost signala u trenutnom intervalu (eng. ***current period***) poredi se sa osnovnim periodom i tim poređenjem se dolazi do saznanja da li je došlo do značajnije promene ili ne. Period ignorisanja (eng. ***ignore period***) služi za vremensko distanciranje poređena dva perioda kako bi se izbegla situacija gde ukoliko do anomalije dođe, nove vrednosti signala koje se ispoljavaju u slučaju problema utiču na osnovni period. Na taj način omogućava se da AID-TS sistem i modeli detektuju anomalije tokom dužeg vremenskog perioda. Ukoliko postoji uticaj na korisnika smatra se da je taj period ignorisanja dovoljno dugačak da ako je primećeni problem relevantan, korisnik će reagovati u tom istom periodu.



Slika 4.1. Vremenski intervali relevantni za poređenje prilikom izvršavanja modela

4.1.1.1. **QueryDuration** model

Prvi od pet modela je **QueryDuration** koji ima za cilj da prepozna regresije na nivou vremena izvršavanja različitih upita u relacionim bazama podataka. Model detektuje upit u odgovarajućem intervalu na odgovarajućoj relacionoj bazi podataka kada je zadovoljen sledeći kriterijum.

Prvenstveno se izračunava ukupno prosečno vreme izvršavanja, varijansa vremena izvršavanja i ukupan broj izvršavanja upita za trenutni period. Nad tim podacima se izračunava vrednost Welch T-testa [56]. Ako je vrednost ***t*** dobijena kao rezultat ovog testa manja od ***alfa***, detektovani upit se smatra regresovanim za posmatrani interval. Vrednost ***alfa*** od 0.01 je uzeta nakon detaljne analize sa ciljem smanjenja broja pogrešno detektovanih slučajeva.

Kod ovog modela vrednost parametra ozbiljnosti je izračunata kao razlika u vremenu trenutnog intervala u poređenju sa osnovnim periodom, pomnožena sa brojem izvršavanja upita, čime se dobija utrošeno vreme usled postojanja primećene regresije. Tačne formule za izračunavanje parametra ozbiljnosti regresije za sve generičke modele definisane su patentom [57] i date su u tabeli 4.1.

Tabela 4.1. Formule za izračunavanje parametra ozbiljnosti za pomenute generičke modele

Model	Formula za izračunavanje parametra ozbiljnosti
Query Duration	$\sum_{regressed\ queries} execution\_count_{current} \cdot (\overline{elapsed\_time}_{current} - \overline{elapsed\_time}_{baseline})$
Timeouts QueryLevel	$\sum_{regressed\ queries} execution\_count_{current} \cdot \overline{elapsed\_time}_{current}$
Timeouts DatabaseLevel	$min\left(\frac{1}{10} \cdot \frac{(\overline{aborted\_worker\_per\_request}_{current} - \overline{aborted\_worker\_per\_request}_{baseline})}{max(\sigma(aborted\_worker\_per\_request_{baseline}), 0.001)}, 99\right)$
Exceptions	$min\left(\frac{1}{10} \cdot \frac{(\overline{exceptions\_per\_batch}_{current} - \overline{exceptions\_per\_batch}_{baseline})}{max(\sigma(exceptions\_per\_batch_{baseline}), 0.001)}, 99\right)$
Compile Time	$\sum_{regressed\ queries} execution\_time_{baseline} \cdot (\overline{compile\_time}_{current} - \overline{compile\_time}_{baseline})$

4.1.1.2. **TimeoutsQueryLevel** model

TimeoutsQueryLevel model uzima u razmatranje samo one upite koji nisu uspeli da se izvrše tokom očekivanog perioda pa je njihovo vreme izvršavanja isteklo (eng. ***timed-out***). Stepen regresije se izračunava kao odnos ukupnog broja upita čije izvršavanje nije završeno tokom predviđenog vremenskog intervala u odnosu na ukupan broj svih izvršenih upita. Ako je taj odnos u trenutnom intervalu, u odnosu na osnovni interval, veći od 3 standardne devijacije, taj interval se smatra regresovanim, odnosno problematičnim.

Vrednost od 3 standardne devijacije je izabrana za sve modele imajući u vidu da je pretpostavljena normalna distribucija podataka, a zatim analizom podataka validirana i proverena. Može se zaključiti da se ovim pristupom obuhvata 0.3% svih slučajeva za koje se smatra da su najekstremniji. Glavni cilj ovakvog pristupa je da se izbegnu detekcije slučajeva u kojima ne postoji problem (lažni alarm).

4.1.1.3. **TimeoutsDatabaseLevel** model

Pošto postoje upiti u SQL serveru koji nikad ne dostignu fazu izvršavanja, a samim tim se njihove statistike i ne čuvaju u SQL serveru, model **TimeoutsDatabaseLevel** je zadužen za detekciju anomalija na nivou cele relacione baze podataka. U ovom slučaju glavni kriterijum je broj prekinutih radnih jedinica (eng. ***aborted workers***) čije je izvršavanje prekinuto pre vraćanja rezultata. Broj prekinutih radnih jedinica je specifičan signal koji SQL Server emituje na nivou jedne relacione baze podataka. U slučaju da je ukupan odnos u trenutnom intervalu udaljen više od 3 standardne devijacije u odnosu na osnovni period, relaciona baza podataka koja je primećena u posmatranom intervalu se smatra problematičnom i model je označava za dalje procesiranje.

4.1.1.4. **Exception** model

Exception model prati broj grešaka (eng. ***exceptions***). Konceptualno, model je identičan **TimeoutsDatabaseLevel** modelu. Jedina razlika je u signalu koji se posmatra. U ovom slučaju to je ukupan broj grešaka sumiran po samom numeričkom kodu greške koja se desila. SQL Server može da emituje greške na različitim nivoima počevši od serverskog nivoa pa do nivoa jedne relacione baze podataka. **Exception** model prati greške koje se dešavaju na nivou relacione baze podataka pošto je to skup grešaka koje mogu da se dese kada postoji korisnička interakcija. Ukoliko je ukupan odnos grešaka i uspešno obrađenih upita u trenutnom intervalu udaljen više od 3 standardne devijacije u odnosu na osnovni period, smatra se da je došlo do značajnog pogoršanja koje zahteva dalje procesiranje.

4.1.1.5. **CompileTime** model

CompileTime model je sličan **QueryDuration** modelu. I u ovom slučaju, jedina razlika je u samom signalu koji se prati. Ovaj model prati signal koji je merilo jedne od faza prilikom izvršavanja upita. Kada se detektuje da je vreme izvršavanja upita u toj fazi nešto duže, interval se obeležava i šalje na dalju analizu kako bi se tačno utvrdio uzrok za produženo izrvšavanje.

Različiti stepeni problema u zavisnosti od vrednosti parametra ozbiljnosti u ovom i drugim modelima su tačno definisani u patentu [57] i prikazani su u tabeli 4.2. Nakon sveobuhvatne ali grube detekcije intervala i problema pomoću generičkih modela u relacionim bazama podataka, čija

vrednost parametra ozbiljnosti prelazi definisanu granicu, precizna analiza se dalje prepušta kategorizacionim modelima.

Tabela 4.2. Stepen problema u odnosu na vrednost parametra ozbiljnosti $\boldsymbol{Sev}$

Raspon vrednosti	Objašnjenje
$\mathbf{0 < Sev < 1}$	Ne smatra se problemom
$\mathbf{1 < Sev < 10}$	Nizak nivo problema, uglavnom problemi koji se ne primećuju od strane korisnika i koji mogu biti privremenog karaktera.
$\mathbf{10 < Sev < 100}$	Srednji nivo problema, uglavnom problemi koji se mogu primetiti od strane korisnika iako su privremenog karaktera, što se najčešće dešava u slučaju kada je relaciona baza podataka kritična tačka korisnikovog sistema.
$\mathbf{100 < Sev < 1000}$	Viši nivo problema, uglavnom problemi koji su privremenog karaktera i koje korisnik sigurno primećuje.
$\mathbf{Sev > 1000}$	Najviši nivo problema, uglavnom problemi koji nisu privremenog karaktera i koji utiču na ponašanje relacione baze podataka vrlo negativno.

4.1.2. Kategorizacioni modeli

Kategorizacioni modeli su napravljeni nakon detaljne analize više različitih slučajeva koje su detektovali generički modeli. Nakon tog postupka, problemi su grupisani u 11 kategorija. Tom prilikom posmatrane su različite relacione baze podataka, koje su imale različite upite, pripadale različitim korisnicima i slično. Grupisanje je rađeno na osnovu zaključka konkretnog problema koji je uočen prilikom analiziranja. U svim ovim slučajevima to je radio posebno organizovan tim inženjera. Rezultati analize su zatim proveravani sa određenim korisnicima platforme kako bi se potvrdile pretpostavke koje su inženjeri tokom rada imali.

4.1.2.1. Kategorije **HittingResourceLimits** i **HittingWorkerLimits**

Zadatak modela **HittingResourceLimits** je da detektuje korelaciju između upita koji imaju problem usled ograničenja korišćenja procesora (skr. ***CPU***) ili ulazno-izlaznog sistema (skr. ***IO***). Osim praćenja korišćenja resursa kao glavnog signala, ova situacija se dalje koreliše i posmatranjem drugih signala kao što su ukupna vremena čekanja na sledećim tipovima čekanja:

- **sos_scheduler_yield**,
- **io_completion**.

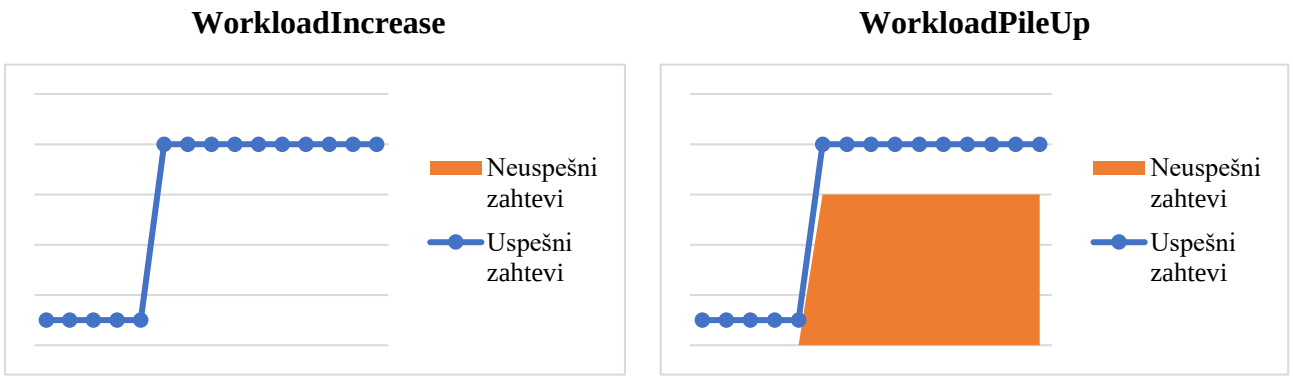
Pod terminom „tip čekanja" u kontekstu SQL servera [58], podrazumeva se specifičan tip resursa na koji proces ili nit unutar SQL servera čeka.

Model **HittingWorkerLimits** je sličan modelu **HittingResourceLimits** a razlika je samo u signalu koji se prati. U ovom slučaju je to broj aktivnih radnih niti. U SQL Serveru, radne niti su zadužene za procesiranje zahteva koji dolaze od klijenta. Ako postoji korelacija između regresije i činjenice

da je broj aktivnih radnih niti dostigao gornju granicu, interval se svrstava u kategoriju **HittingWorkerLimits**.

4.1.2.2. Kategorije **WorkloadIncrease** i **WorkloadPileup**

Modeli **WorkloadIncrease** i **WorkloadPileup** prate način na koji se relaciona baza podataka koristi. Ukoliko je moguće uvideti korelaciju između regresije koja je detektovana od strane generičkih modela i ukoliko se može primetiti da je došlo do promene načina korišćenja relacione baze podataka, slučaj se smatra porastom upotrebe (eng. ***workload increase***). To je slučaj u kome je došlo samo do povećanja broja zahteva/upita. Ukoliko je slučaj nešto kompleksniji i, pored povećanog broja zahteva/upita, neki od zahteva se ne mogu izvršiti u predviđenom vremenskom intervalu, ovo se kategorizuje kao slučaj u kome se zahtevi nagomilavaju i ne mogu da se opsluže u odgovarajućem vremenskom periodu (eng. ***workload pileup***). Slučajevi koji se detektuju na ovaj način grafički su prikazani na slici 4.2.



Slika 4.2. Razlika između kategorija **WorkloadIncrease** i **WorkloadPileup**

4.1.2.3. Kategorije **MemoryPressure** i **MemoryPileup**

Modeli **MemoryPressure** i **MemoryPileup** su slični modelima **WorkloadIncrease** i **WorkloadPileup**, ali detektuje povećanu upotrebu memorije. Tom prilikom se identifikuju upiti koji imaju velika vremena čekanja na tipu čekanja **resource_semaphore** koji odgovara resursima vezanim za operativnu memoriju. Ako postoji korelacija između velikog čekanja na operativnu memoriju i njenog veće korišćenja, uključujući i ukupno vreme čekanja, interval se obeležava kategorijama **MemoryPressure** ili **MemoryPileup**. Pritisak na memoriju (eng. ***memory pressure***) se može smatrati situacijom gde je, samo zbog povećanog broja upita ili zbog prirode samih upita upotreba radne memorije porasla. Ukoliko tom prilikom postoji situacija u kojoj neki od upita ne mogu da se izvrše u predviđenom vremenskom intervalu, taj slučaj se smatra nagomilavanjem upita koji čekaju na operativnu memoriju (eng. ***memory pileup***).

4.1.2.4. Kategorija **PlanRegression**

PlanRegression model prati promenu plana upita koji se može upariti i korelisati sa regresijom uočenom od strane generičkih modela. Ukoliko neki od upita koji je uhvaćen kao upit sa znakovima

regresije u trenutnom posmatranom intervalu u poređenju sa osnovnim intervalom, a da je pritom došlo i do promene plana za taj upit u trenutnom interval, takav slučaj se smatra regresijom za koju je odgovorna promena plana. To je vrlo često situacija koja se dešava ili kao posledica neažuriranih statistika unutar SQL servera ili kao posledica pogrešno predviđene upotrebe SQL servera, gde dolazi do izbora neoptimalnog plana koji izaziva duže izvršavanje upita.

4.1.2.5. Kategorija **Failover**

Failover model je jedan od najednostavnijih modela. Ako dođe do restartovanja relacione baze podataka ili ako dođe do situacije u kojoj se ona pomeri sa jedne virtuelne ili fizičke mašine na drugu zbog prazne operativne memorije na novoj mašini, relaciona baza podataka može imati privremeni problem zbog toga. Model detektuje ovu kategoriju tako što prati naziv mašine na kojoj se proces SQL servera izvršava uključujući i identifikacioni broj procesa.

4.1.2.6. Kategorija **Locking**

Locking model analizira grupu tipova čekanja koji se koriste za zaključavanje određenog resursa. U suštini, model detektuje slučajeve u kojima jedan ili više različitih upita koji se izvršavaju simultano zahtevaju ekskluzivni ili deljeni pristup (eng. ***lock***) nad konkretnim resursom i samim tim, takav resurs nije dostupan drugima. To se može desiti na bilo kom resursu kao što su memorija, procesor ili disk.

Stavljanje kritičnih sekcija pod operaciju ekskluzivnog pristupa se često primenjuje prilikom pravljenja aplikacija i pojava ovakvih signala ne mora da znači da postoji problem. Sa druge strane, može doći i do neželjenih efekata ukoliko definisani upit nije napravljen tako da ima paralelizaciju i kritične sekcije u vidu. Takođe može se desiti da postoji i greška u programskom kodu aplikacije koja koristi relacionu bazu podataka, zbog koje se pojavljuju ovakve situacije.

4.1.2.7. Kategorija **IncreasedMAXDOP**

IncreasedMAXDOP (**Increased Maximum Degree of Parallelism**) model odgovara slučaju kada se izvršavanje paralelizovanog upita koji se duže izvršava može korelisati sa tipom čekanja **cxpacket**. Pod oznakom **cxpacket** se prati vreme čekanja na sinhronizaciju različitih niti koje obrađuju upit u paraleli. U tom slučaju prevelika paralelizacija može dovesti do dodatne sinhronizacije koja nije optimalna i koja, u odnosu na manje paralelizovan upit ili sekvencijalan upit, iziskuje više procesorskog vremena.

4.1.2.8. Kategorije **MemoryContention** i **TempDBContention**

MemoryContention model je karakterističan za tip upotrebe relacione baze podataka gde upiti pristupaju istim memorijskim stranicama koje su locirane na fizičkom disku. Ukoliko se ovo desi u kontekstu privremene (eng. ***temp***) relacione baze podataka, koja predstavlja mesto koje korisnik privremeno može da iskoristi za neke svoje operacije, interval se dodatno svrstava u kategoriju **TempDBContention**.

4.1.2.9. Kategorija **MissingIndex**

Model **MissingIndex** se pojavljuje za upite koji su detektovani od strane generičkih modela kod kojih postoji i korelacija sa signalima koji opisuju ulazno/izlazne operacije. To konkretno znači da bi se detektovani upiti izvršili, potrebno je pročitati informacije sa diska. U tom slučaju, postojanje indeksiranih podataka bi pomoglo i izvršavanje upita bi bilo brže.

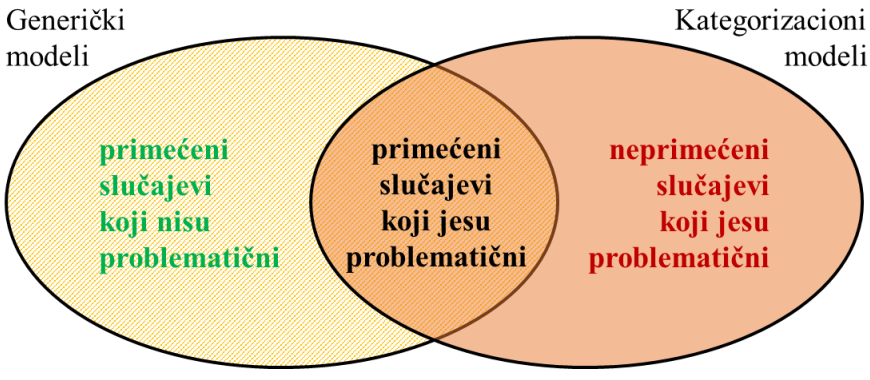
4.1.2.10. Kategorija **SlowClient**

SlowClient kategorija se detektuje kao i u drugim slučajevima kada je odgovarajući tip čekanja povećan. U ovom slučaju je to **async_network_io** tip čekanja. Ovaj tip čekanja predstavlja vreme koje je potrebno da se izvrši prenos podataka od SQL servera do klijentske aplikacije. Anomalija u ovom slučaju ukazuje ili na zagušenje na mreži ili može ukazati i na opterećenost procesora na jednoj ili drugoj strani uspostavljene konekcije gde procesor pristigle mrežne pakete na mrežnoj kartici ne stiže da obradi, ili ih obrađuje nedovoljno brzo.

4.1.3. Saradnja generičkih i kategorizacionih modela

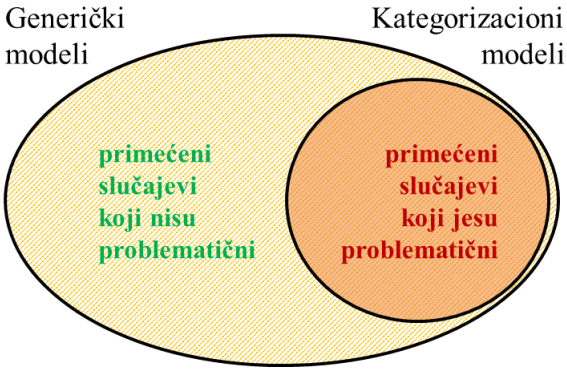
U opštem slučaju dva skupa modela, koji su namenjeni drugačijoj vrsti detekcije, posmatraju i otkrivaju različite skupove događaja. Presek ta dva skupa treba da predstavlja sigurne probleme. Razlika skupova u tom slučaju predstavlja ili skup slučajeva u kojima problem ne postoji, ili skup slučajeva koje postojeći generički modeli ne primećuju iz različitih razloga.

Kao što se može videti na slici 4.3. ovi skupovi su predstavljeni u opštem slučaju. Problemi koji su primećeni kategorizacionim modelima, a nisu primećeni od strane generičkih modela, predstavljaju oblast čiji slučajevi mogu da ukažu na mogućnost unapređenje AID-TS sistema. Takav pristup se primenjuje tokom razvijanja novog kategorizacionog modela i može se koristiti za dalja poboljšanja postojećih generičkih modela. U trenutno definisanom rešenju AID-TS sistem ne može sam sebe da unapređuje, već je za to zadužen tim inženjera koji rade na njegovom razvoju.



Slika 4.3. Međusobni odnos između izlaza dve grupe modela u opštem slučaju

U ovom radu, kategorizacioni modeli analiziraju izlaz generičkih modela. Na taj način krajnji rezultat predstavlja podskup prvobitnog skupa, što se može videti na slici 4.4. Ovakav pristup, kroz obradu u dve faze, povećava verovatnoću da će se stvarni problem obuhvatiti i klasifikovati na pravi način.



Slika 4.4. Međusobni odnos između izlaza dve grupe modela u ovom radu

Definisane modele (5 generičkih i 11 kategorizacionih modela) je moguće uporediti sa modelima koji se mogu naći u otvorenoj literaturi, gde su rešenja DBSeer [34] i DBSherlock [35] najsličnija. Njihov sistem se sastoji od 10 različitih modela koji detektuju anomalije i regresije. Definisani modeli su dobijeni nakon upotrebe sintetičkog testnog okruženja i upotrebe upita koji bi trebali da liče na upite koji u stvarnom okruženju korisnici izvršavaju. Imajući u vidu da je AID-TS sistem uključio upotrebu stvarnog klaud okruženja koji je za više redova veličine složeniji i veći u odnosu na testno okruženje, a da u isto vreme ima i više modela, očekuje se da donosi i kompletnije konačno rešenje i bolju detekciju.

Rešenje koje je realizovano u stvarnom produkcionom okruženju je prikazano u radu [25]. Ono je najbliže tehnologiji i rešenju koje je predstavljeno u ovom radu. Prikazan sistem može da detektuje 9 različitih problema. Imajući u vidu da je rešenje razvijano nad relacionom bazom podataka MySQL i da je koristilo osam signala od kojih su pet jedinstveni (CPU, IO, broj transakcije po sekundi, mreža, memorija), može se zaključiti da je ograničen po pitanju različitosti signala koji mogu detaljno da ukažu na određene probleme. Kod SQL servera koji nudi kompanija Microsoft to nije slučaj jer je broj različitih signala koje je moguće upotrebiti za analizu značajno veći. Samim tim i broj različitih slučajeva koje je moguće uhvatiti AID-TS sistemom je veći. Rad se takođe ne bavi analizom problema koji dolaze kao posledica neoptimalnog rada samog jezgra MySQL servera, što je u AID-TS sistemu pokriveno s obzirom da takav slučaj može predstavljati značajan izvor problema.

Tokom svih analiza, glavni cilj je bio da se krajnjem korisniku ponudi precizna i jasna informacija. Na početku razvoja AID-TS sistema, pokušana je primena pristupa predstavljenog u radu [27], gde se teži implementaciji jednog modela koji bi uhvatio sve slučajeve. Pokazalo se da je takav pristup u ovakvom sistemu neizvodljiv zbog malog uzorka obeleženih slučajeva, a velike raznolikosti signala koi su emitovani. Imajući to u vidu, pristupilo se definisanju više različitih modela, vrlo slično pristupu koji je sproveden u radu [53].

Važno je napomenuti da u ovakvom sistemu može doći i do sledećeg slučaja. Kada se intervali detektuju od strane generičkih modela, a zatim dodatno obogate rezultatima dobijenih iz kategorizacionih modela, može se desiti da za različite ulazne parametre primećenih slučajeva rezultati budu slični, ali da su u isto vreme uzrok i posledica različiti. Ovakav slučaj se dešava kada isti modeli detektuju jednu ili više situacija gde je ključna razlika u vrednosti parametra ozbiljnosti problema. Kao što je već napomenuto, vrednost parametra ozbiljnosti problema u kontekstu kategorizacionih modela se odnosi na stepen korelisanosti sa onim što je generički model detektovao. Iako rešenje predstavljeno u [25] primenjuje drugačiji pristup u odnosu na pristup koji je definisan kod kategorizacionih modela, primećena je sličnost u dobijenim rezultatima. Ustanovljeno je da za različite upite, u našem slučaju intervale, oni mogu imati slične potpise ali različite uzroke problema, što zahteva dalju analizu i bolje razlikovanje problema.

Ovo se može ilustrovati na jednom primeru. Model **QueryDuration** je detektovao dva različita intervala koji su kasnije procesirani od strane kategorizacionih modela i dobijene su sledeće vrednosti parametra ozbiljnosti:

- Interval I1:
    - Kategorija **Locking** - 1,
    - Kategorija **HittingResourceLimit** - 2,
- Interval I2:
    - Kategorija **Locking** - 3,
    - Kategorija **HittingResourceLimit** - 1.

Iz ovog primera se može videti da su za dva različita intervala I1 i I2 isti kategorizacioni modeli uhvatili, naizgled, jednake probleme. Bitno je primetiti da su vrednosti parametra ozbiljnosti različite. Iako izgleda da detektovane vrednosti nisu mnogo različite, konačno objašnjenje koje se isporučuje korisnicima u posmatranom intervalu se razlikuje i u direktnoj je zavisnosti od vrednosti parametra ozbiljnosti. Na primer, u intervalu I1, relaciona baza podataka je iskoristila sve raspoložive resurse što je zatim izazivalo zaključavanje resursa kao neželjenu posledicu, dok u slučaju intervala I2, problem koji je uhvaćen modelom **Locking** kao neželjenu pojavu izaziva da relaciona baza podataka koristi više resursa.

Za sistem namenjen inteligentnom otkrivanju uzroka problema u relacionim bazama podataka u klaud okruženju veoma je važno razlikovati gore spomenute slučajeve, kako bi bilo moguće dati potpuno i tačno razumevanje problema. Nakon detaljne analize podataka, zaključeno je da, zbog ovakvih i sličnih situacija, neophodno je izlazne podatke procesirati u dodatnom koraku. Za dodatni korak bi bio zadužen ekspertski sistem sa ciljem da pravi razlike između konkretnih slučajeva i da preko parametra ozbiljnosti razlikuje pojedine slučajeve. Imajući to u vidu, može se reći da je ekspertski sistem zadužen za konačnu odluku koja sadrži informaciju o tome:

- šta je problem,
- šta je uzrok problema,
- šta su posledice problema.

4.2. Ekspertski sistem

Ekspertski sistem donosi odluke na osnovu ulaznih podataka u skladu sa definisanim pravilima. On može da prepozna i razlikuje različite rezultate prethodnih statističkih modela. Detaljno istraživanje ekspertskih sistema [47], [59], [60] i [61], urađeno je prilikom definisanja arhitekture novog ekspertskog sistema a za potrebe AID-TS sistema.

Kod ekspertskog sistema koji se zasniva na pravilima, svako pravilo se sastoji od uslovnog dela i posledičnog dela. Tradicionalno, pravila se realizuju preko prostih programskih naredbi selekcije **IF – THEN**. Uslovni deo se može sastojati od jednog ili više uslova koji se povezuju logičkim operatorima. U ovom ekspertskom sistemu, spomenuti uslovi su formirani upotrebom odgovarajućih činjenica (eng. ***fact***) i operatora poređenja. Na primer, detektovane kategorije u intervalima od strane kategorizacionih modela se smatraju činjenicama. Posledični deo predstavlja rezultat pravila koji se sastoji od objašnjenja krajnjeg rezultata.

Pravila su predstavljena putem XML formata sa jasno definisanom šemom koja je data na slikama 4.5. i 4.6. Skup pravila u ovom sistemu, kao što se može videte na predstavljenoj šemi, sadrži jedno ili više pravila gde se svako od njih sastoji od izraza uslova i rezultata. Izraz uslova je predstavljen ili jednim uslovom ili kombinacijom više uslova upotrebom logičkih operanada ***AND***, ***OR*** i ***NOT***. Rezultat pravila sadrži detaljan opis problema i njegovog uzroka.

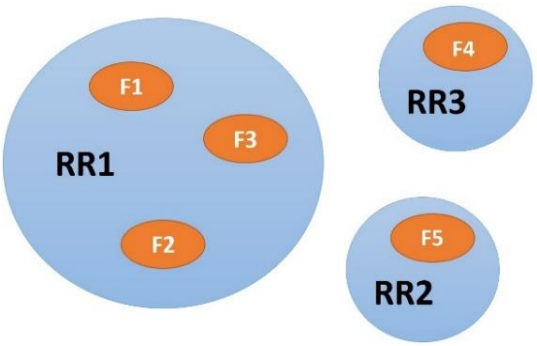
```
1  <xs:schema id="ExpertSystemRules">
2   <xs:complexType name="RuleType">
3    <xs:all>
4     <xs:element type="xs:string" name="Fact" minOccurs="1" maxOccurs="1"/>
5     <xs:element type="xs:string" name="Value" minOccurs="1" maxOccurs="1"/>
6     <xs:element type="xs:string" name="Operator" minOccurs="1" maxOccurs="1"/>
7     <xs:element type="xs:string" name="Type" minOccurs="1" maxOccurs="1"/>
8    </xs:all>
9   </xs:complexType>
10  <xs:complexType name="OrType">
11   <xs:choice minOccurs="2">
12    <xs:element name="And" type="AndType"/>
13    <xs:element name="Rule" type="RuleType"/>
14    <xs:element name="Not" type="NotType"/>
15   </xs:choice>
16  </xs:complexType>
17  <xs:complexType name="AndType">
18   <xs:choice minOccurs="2">
19    <xs:element name="Or" type="OrType"/>
20    <xs:element name="Rule" type="RuleType"/>
21    <xs:element name="Not" type="NotType"/>
22   </xs:choice>
23  </xs:complexType>
24  <xs:complexType name="NotType">
25   <xs:choice minOccurs="1" maxOccurs="1">
26    <xs:element name="And" type="AndType"/>
27    <xs:element name="Or" type="OrType"/>
28    <xs:element name="Rule" type="RuleType"/>
29   </xs:choice>
30  </xs:complexType>
31  <!--  nastavak na sledećoj strani -->
```

Slika 4.5. Definicija osnovnih tipova XML šeme za pravila ekspertskog sistema

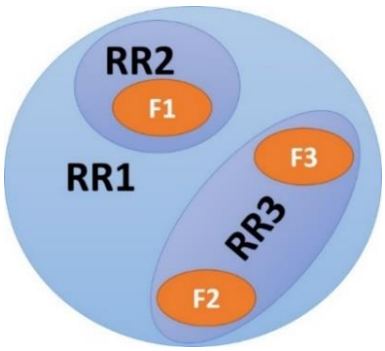
```
31  <!--  nastavak sa prethodne strane -->
32  <xs:complexType name="ExpressionType">
33   <xs:choice minOccurs="1" maxOccurs="1">
34    <xs:element name="Not" type="NotType"/>
35    <xs:element name="And" type="AndType"/>
36    <xs:element name="Or" type="OrType"/>
37    <xs:element name="Rule" type="RuleType"/>
38   </xs:choice>
39  </xs:complexType>
40  <xs:complexType name="ClassificationResultType" >
41   <xs:all>
42    <xs:element type="xs:string" name="Class" minOccurs="1" maxOccurs="1"/>
43    <xs:element type="xs:string" name="Description" minOccurs="1" maxOccurs="1"/>
44   </xs:all>
45  </xs:complexType>
46  <xs:complexType name="ClassificationRuleSetType">
47   <xs:sequence>
48    <xs:element name="Expression" type="ExpressionType" minOccurs="1"
   maxOccurs="1" />
49    <xs:element name="Result" type="ClassificationResultType" minOccurs="1"
   maxOccurs="1" />
50   </xs:sequence>
51  </xs:complexType>
52  <xs:element name="ExpertSystemRules">
53   <xs:complexType>
54    <xs:sequence>
55     <xs:element name="ClassificationRuleSet" type="ClassificationRuleSetType" />
56    </xs:sequence>
57   </xs:complexType>
58  </xs:element>
59 </xs:schema>
```

Slika 4.6. Definicija složenih tipova XML šeme za pravila ekspertskog sistema

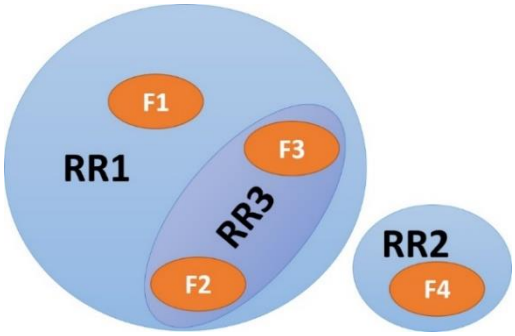
U ovako definisanom sistemu se može desiti da se pravila preklapaju. To praktično znači da je jedno pravilo podskup drugog. Ekspertski sistem je projektovan tako da se rezultati uvek pridružuju u nadskup ukoliko je to moguće zbog pojednostavljivanja logike i lakšeg primećivanja slučajeva za koje pravila nisu definisana korektno. Mogući slučajevi prihvatljivih rezultata u ekspertskom sistemu su prikazani na slikama 4.7., 4.8. i 4.9.



Slika 4.7. Prihvatljiv slučaj u kome se vrši mapiranje pet različitih činjenica na tri pravila



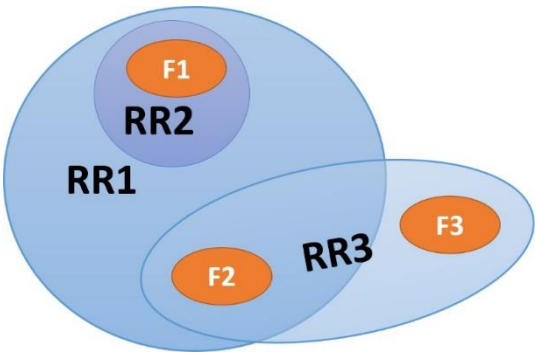
Slika 4.8. Prihvatljiv slučaj u kome se vrši mapiranje tri različite činjenice na tri pravila od kojih su dva podskup pa je dobijeni rezultat jedno pravilo



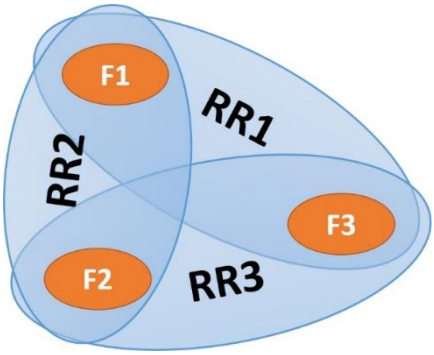
Slika 4.9. Prihvatljiv slučaj u kome se vrši mapiranje četiri različite činjenice na tri pravila od kojih su dva nezavisna a jedno je podskup

Na ovim slikama, **RR** predstavlja oznaku za rezultujuće pravilo, dok oznake **F1**, **F2** … **F5** predstavljaju činjenice koje zadati interval sadrži. Može se videti da je skup različitih činjenica moguće mapirati na različita pravila i da se u tom slučaju uvek bira nadskup kao konačan rezultat. U slučaju prikazanom na slici 4.9. konačan rezultat će sadržati rezultujuća pravila **RR1** i **RR2**, dok je rezultujuće pravilo **RR3** podskup od pravila **RR1** i samim tim pravilo **RR3** neće biti deo konačnog rezultata.

Nasuprot prihvatljivim rezultatima, na slikama 4.10. i 4.11. prikazani su slučajevi koji nisu dozvoljeni u ovom ekspertskom sistemu. U ovom slučaju 3 različite činjenice moguće je mapirati na 3 različita pravila. U drugom ekstremno ulančanom slučaju, pravilo **RR1** obuhvata **F1** i **F3**, pravilo **RR2** obuhvata **F1** i **F2**, a pravilo **RR3** obuhvata **F1** i **F3**, čime se dolazi do lančane reakcije. U toj situaciji ako ne postoji pravilo koje agregira sve tri činjenice u nad skup, sistem ne ume da odluči koje od tri pravila je potrebno istaknuti, te se u tom slučaju smatra da je došlo do greške, pa se zahteva dalje unapređenje ekspertskog sistema.



Slika 4.10. Mapiranje činjenica i pravila u slučaju preseka
koji se smatraju neprihvatljivim



Slika 4.11. Mapiranje činjenica i pravila u slučaju ulančanog preseka
koji se smatraju neprihvatljivim

Ukoliko do neželjenih slučajeva na neki način dođe, postoji mehanizam odbrane u vidu testova koji proveravaju da li na osnovu baze pravila može doći do takvih slučajeva. Generalno, ako postoji više pravila koje sadrže deo činjenica iz drugih pravila, pa samim tim formiraju lanac preklapajućih rezultata, to je neprihvatljiva situacija zato što ekspertski sistem u toj situaciji ne ume da odluči koje pravilo ima prioritet i ne može se sa sigurnošću reći šta je uzrok a šta posledica. U takvim slučajevima se inženjerskom timu šalje upozorenje da ekspertski sistem sadrži nekonzistentna pravila koje potom inženjerski tim mora da popravi.

4.2.1. Saradnja ekspertskog sistema i statističkih modela

Slučajevi koji su primećeni od strane generičkih modela, potom analizirani od strane kategorizacionih modela se, zatim, šalju na obradu ekspertskom sistemu. Tom prilikom se vrši poređenje izlaza modela sa definisanom bazom pravila ekspertskog sistema. Trenutno, baza ekspertskog sistema sadrži 175 pravila. Broj definisanih pravila je za red veličine veći u odnosu na broj različiith slučajeva koje je moguće razumeti i detektovati u radovima [25], [34] i [35].

Dva primera definisanih pravila će biti objašnjena u više detalja. Na slici 4.12. predstavljeno je pravilo koje se odnosi na slučaj kada je usporenje detektovano od strane modela **QueryDuration**. Ovaj interval bi, zatim, bio kategorizovan i od strane kategorizaconih modela **Locking**, **PlanRegression** i **NewQuery**. U tom slučaju, uzrok problema je pojavljivanje novog upita, koji nad određenim objektom upotrebljenom u samom upitu, kao posledicu izaziva njegovo zaključavanje, a samim tim taj objekat postaje nedostupan za druge upite. Time je izazvan dalji problem, gde upit koji ne može da pristupi objektu zbog ekskluzivnog zaključavanja sada ima produženo vreme čekanja, a samim tim i ukupne statistike izvršavanja su duže. Zbog regresije u statistikama izvršavanja, relaciona baza podataka odabira drugi plan misleći da je on bolji što dovodi do daljeg usporenja i regresije u odnosu na optimalno izvršavanje upita.

```
1  <ClassificationRuleSet>
2   <Expression>
3    <And>
4     <Rule>
5      <Fact>GenericModel</Fact>
6      <Operator>Equal</Operator>
7      <Value>QueryDuration</Value>
8      <Type>String</Type>
9     </Rule>
10    <Rule>
11     <Fact>Locking</Fact>
12     <Operator>GreaterThan</Operator>
13     <Value>3.0</Value>
14     <Type>Double</Type>
15    </Rule>
16    <Rule>
17     <Fact>PlanRegression</Fact>
18     <Operator>LessThan</Operator>
19     <Value>2.0</Value>
20     <Type>Double</Type>
21    </Rule>
22    <Rule>
23     <Fact>NewQuery</Fact>
24     <Operator>GreaterOrEqual</Operator>
25     <Value>4.0</Value>
26     <Type>Double</Type>
27    </Rule>
28   </And>
29  </Expression>
30  <Result><Class>Customer</Class>
31   <Description> New query '{query_hash_1}' has been detected which
   introduced locking over resource '{obj_1}' and as a consequence there was a
   plan regression on an existing query '{query_hash_2}'.
32   </Description>
33  </Result>
34 </ClassificationRuleSet>
```

Slika 4.12. Primer definisanog pravila za model **QueryDuration** u ekspertskom sistemu

Drugi primer koji je predstavljen na slici 4.13. se odnosi na slučaj kada model **Exception** detektuje povećanje broja pojavljivanja grešaka pod numeričkim brojevima **207** i **208.** Ove greške se javljaju kada se u upitima koristi nedostajući objekat ili kolona u šemi relacione baze podataka. Takođe, ovakav slučaj može da se desi i kao posledica greške u upitu koji je poslat od strane aplikacije koja ima problem u programskom kodu.

```
<ClassificationRuleSet>
 <Expression>
  <And>
   <Rule>
    <Fact>GenericModel</Fact>
    <Operator>Equal</Operator>
    <Value>Exception</Value>
    <Type>String</Type>
   </Rule>
   <Or>
    <Rule>
     <Fact>207</Fact>
     <Operator>GreaterThan</Operator>
     <Value>0.0</Value>
     <Type>Double</Type>
    </Rule>
    <Rule>
     <Fact>208</Fact>
     <Operator>GreaterThan</Operator>
     <Value>0.0</Value>
     <Type>Double</Type>
    </Rule>
   </Or>
  </And>
 </Expression>
 <Result><Class>Customer</Class>
  <Description>Query ‘{query_hash_1}’ is hitting errors which point to
missing column or object. Please double check the query or the schema of a
database {database_name}.
  </Description>
 </Result>
</ClassificationRuleSet>
```

Slika 4.13. Primer definisanog pravila za model **Exception** u ekspertskom sistemu

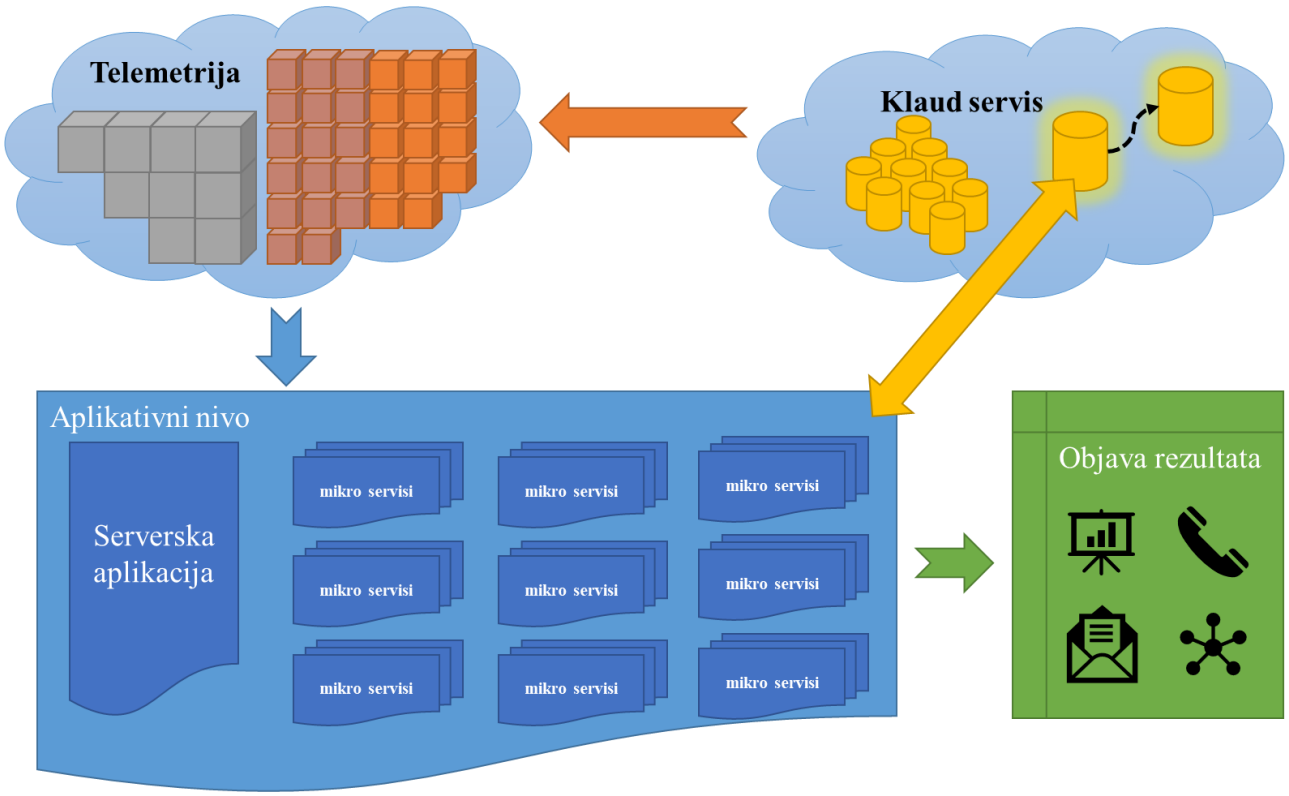
U slučaju da ni jedno od pravila ne odgovara činjenicama koje su detektovali modeli, takav redak slučaj se obeležava nepostojećim pravilom (eng. ***unknown***) i odgovarajuća vrsta upozorenja se šalje inženjerskom timu kako bi se ova nekonzistentnost otklonila. Upozorenja se šalju sa ciljem daljeg istraživanja i poboljšanja samog ekspertskog sistema kako bi mu se dodalo novo ili modifikovalo postojeće pravilo.

5. Definicija infrastrukture za datu arhitekturu

Da bi definisani AID-TS sistem uspešno radio, potrebno je definisati infrastrukturu koja odgovara njegovoj arhitekturi. Neophodno je da sistem može nesmetano da skalira sa povećanjem broja relacionih baza podataka koje je potrebno pratiti. Tom prilikom potrebno je odrediti šta je izvor podataka i kako se ti podaci čuvaju i prenose do samih modela. Potom, potrebno je definisati okruženje u kojem se izvršavaju modeli i ekspertski sistem. Podjednako je važno i kako se koristi infrastruktura za obaveštavanje korisnika o rezultatima koje sistem generiše. Definisana arhitektura je grafički predstavljena na slici 5.1.

5.1. Relacione baze podataka kao izvor podataka

Kao što je već napomenuto, i u privatnoj infrastrukturi, a i u klaud okruženju signale koje relaciona baza podataka emituje tokom korišćenja, moguće je pratiti, uzorkovati, čuvati i prosleđivati na dalju obradu. Tom prilikom, kako bi se smanjilo kašnjenje podataka do modela, potrebno je imati odgovarajući mehanizam koji te podatke efikasno prosleđuje. U tu svrhu se najčešće koriste postojeći sistemi koji omogućavaju prenos podataka i događaja. Relaciona baza podataka, u tom slučaju, efikasno emituje signale koji se šalju do sistema kolektora podataka, koji potom te podatke čuvaju u odgovarajućem obliku u jednom ili više skladišta podataka.



Slika 5.1. Definisana infrastruktura za datu arhitekturu sistema

5.2. Telemetrija

Kada se govori o telemetriji kao sveobuhvatnom skladištu podataka, u praksi se podaci mogu čuvati u struktuiranom obliku u tri vrste skladišta podataka:

- hladno skladište (eng. ***cold storage***),
- mlako skladište (eng. ***warm storage***),
- vruće skladište (eng. ***hot storage***).

Kao što se može videti na slici 5.2., na putu od izvora podataka do njihovog skladišta, podaci su prvo dostupni u vrućem, potom u mlakom, a tek na kraju u hladnom skladištu podataka.

Slika 5.2. Dostupnost prikupljenih podataka u različitim skladištima podataka

U vrućem skladištu prikupljeni podaci su dostupni vrlo brzo nakon njihovog emitovanja, a njihova dostupnost se meri u sekundama. Ono se koristi u slučajevima gde je potrebna izuzetno brza reakcija. Najpoznatiji primer upotrebe su mehanizmi uzbunjivanja, gde je brz pristup podacima od izuzetne važnosti. Izvršavanje upita nad takvom vrstom skladišta podataka mora se završiti u odgovarajućem vremenskom periodu, a rezultati moraju biti spremni u realnom vremenu. Sa druge strane, da bi se odgovarajuće performanse koje omogućavaju dobijanje rezultata u realnom vremenu očuvale, potrebno je ograničiti količinu podataka u ovakvom tipu skladišta, pa se podaci u ovakvom tipu skladišta čuvaju najviše jedan ili dva dana.

U mlakom skladištu prikupljeni podaci dostupni su u roku od nekoliko minuta. Ono se koristi u slučajevima gde je potrebna brza ali ne i momentalna reakcija. Najpoznatiji primer upotrebe se odnosi na kratkoročno analiziranje podataka prilikom istraživanja rada nekog sistema. Izvršavanje upita nad takvom vrstom skladišta podataka i dobijanje rezultata moraju biti u odgovarajućem vremenskom periodu, ali nije potrebno da se to desi momentalno. Samim tim, količina podataka koju je moguće čuvati u ovakvom tipu skladišta je veća, pa se u ovakvim skladištima podaci čuvaju između 5 i 14 dana.

Hladno skladište podataka sadrži ogromne količine podataka čiji je period čuvanja značajno duži u poređenju sa periodom čuvanja podataka u slučaju mlakog i vrućeg skladišta. Ograničavajući faktor ovog tipa skladišta proizilazi iz vremena koje je potrebno da se podaci prebace iz vrućeg ili mlakog skladišta u hladno skladište koje se meri satima. Podaci u hladnom skladištu se koriste za različite analize čije rezultate nije neophodno dobiti u kratkom vremenskom periodu. Period čuvanja podataka u ovakvom skladištu se meri mesecima pa čak i godinama u zavisnosti od veličine samog sistema koji se prati.

Imajuću u vidu definisane modele u četvrtom poglavlju, kod kojih se podaci koriste u skorijoj istoriji, arhitektura AID-TS sistema će upotrebiti mlako i hladno skladište. Mlako skladište je upotrebljeno za sam AID-TS sistem kao skladište podataka nad kojim modeli izvršavaju upite, dok se hladno skladište upotrebljava od strane inženjerskog i naučnog tima za potrebe dugoročnih analiza podataka.

5.3. Aplikativni nivo za obradu podataka

Nad skladištem podataka, u kojem se nalaze prikupljeni telemetrijski signali, potrebno je izvršavati modele. Za tu svrhu moguće je slediti različite pristupe. Jedna mogućnost je upotreba serverske aplikacije koja ima zadatak da izvrši sve modele i objavi njihove rezultate.

S druge strane, alternativu predstavljaju mikroservisi kao vrlo male aplikacije čiji je životni ciklus značajno kraći u odnosu na serverske aplikacije. Serverska aplikacija postoji stalno, dok mikroservisi postoje samo tokom vremena izvršavanja zadatka za koji su zaduženi. Primarni cilj pristupa mikroservisa je omogućavanje daljeg lakšeg skaliranja.

U slučaju ovog sistema upotrebljena su oba pristupa. Serverska aplikacija je zadužena za pravljenje mikroservisa i zakazivanje izvršavanja zadataka. Mikroservisi izvršavaju modele na osnovu zadatog zadatka sa odgovarajućim parametrima. Na taj način se postiže najveća moguća fleksibilnost, s obzirom da je particionisanje moguće učiniti i na nivou modela i na nivou podataka.

5.4. Čuvanje rezultata

Za potrebe čuvanja rezultata koje različiti modeli i ekspertski sistem proizvedu, potrebno je odgovarajuće skladište podataka. U tu svrhu upotrebljena je relaciona baza podataka sa odgovarajućom šemom. Imajući u vidu da je za izvršavanje modela izabran kombinovani pristup, gde se koriste i serverska aplikacija i mikroservisi, potrebno je čuvati i informacije u kom trenutku se neki mikroservis pušta u rad i za koji je model i skup intervala zadužen. Za praćenje stanja svakog od intervala koristi se koncept mašine stanja.

U isto vreme, potrebno je osigurati da u slučaju otkaza ili neke prirodne nepogode, relaciona baza podataka nastavi da bude dostupna. Relaciona baza podataka koja se koristi za čuvanje rezultata u ovom sistemu mora biti geografski replicirana kako bi se povećala dostupnost u spomenutim slučajevima. Ovo je izuzetno važno s obzirom da i u slučajevima otkaza AID-TS sistem treba da u svakom trenutku pruži uslugu za koju je namenjen. Korisnici sistema očekivaće da ovaj sistem radi bez prekida i da nudi razumevanje problema u približno realnom vremenu, a posebno u slučajevima kada se dešavaju problemi na samoj servisnoj strani pružaoca klaud usluga.

5.5. Objava rezultata

Na kraju procesiranja podataka, svi prikupljeni rezultati se dostavljaju korisniku u odgovarajućem obliku. Potrebno je da korisnik ima mogućnost izbora načina dostavljanja podataka. Arhitektura ovog sistema podrazumeva da je moguće dostaviti rezultate putem:

- grafičkog interfejsa,
- metodom uzbunjivanja:
    - kreiranjem telefonskog poziva,
    - slanjem telefonske poruke,
    - slanjem elektronske poruke,
    - pozivanjem odgovarajućeg jednog ili više aplikativnih interfejsa koje korisnik definiše.

5.6. Kontinuirana analiza i ažuriranje bez posledica

Arhitektura AID-TS sistema omogućava da se ovakav sistem može pratiti na jednostavan način. To podrazumeva da je potrebno pratiti sve aspekte izvršavanja ovakvog sistema putem odgovarajućih logova. Inženjeri i istraživači, čije je zaduženje stalno unapređenje ovakvog sistema, poseduju mehanizam putem kojeg mogu da analiziraju svoje pretpostavke nad podacima koji mogu biti sintetički generisani ili iz stvarnog produkcionog okruženje.

Pristup primenjen upotrebom mikroservisa nudi i mogućnost da se mikroservisima zada okruženje za koje su odgovorni i da se specificira skladište podataka u kome se čuvaju dobijene informacije tokom njihovog izvršavanja. Takav pristup omogućava da, u paraleli sa produkcionim AID-TS sistemom, postoji i deo sistema koji služi za različita testiranja, a da se tom prilikom ne utiče na krajnji rezultat koji se isporučuje korisnicima.

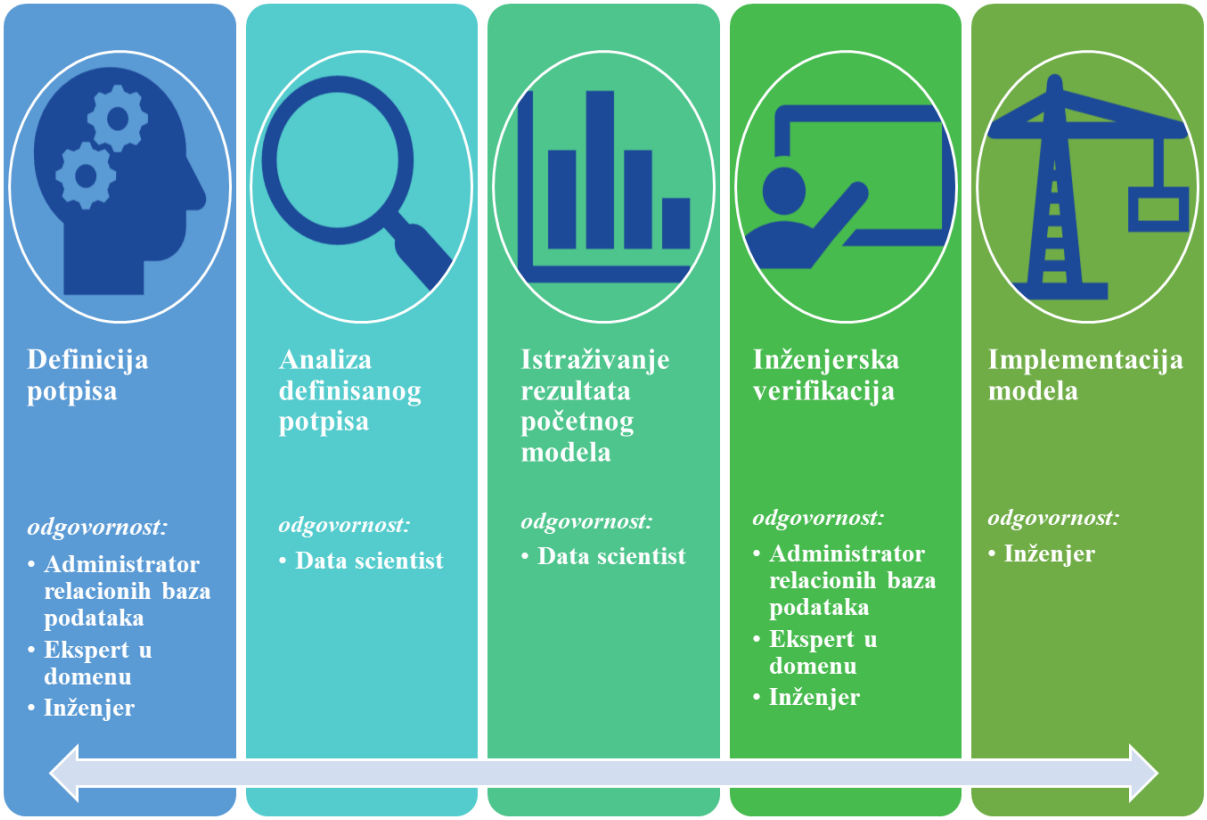
6. Organizacija tima

Organizacija tima je izuzetno važan aspekt prilikom definisanja i razvoja svakog sistema pa i kod AID-TS sistema. Imajući u vidu da za ovakav projekat tim može biti sačinjen od aktera koji poseduju različita znanja i veštine, potrebno je pažljivo pristupiti organizaciji. Tom prilikom, kako bi se tačno znala odogovornost svih aktera u timu, osmišljen je pristup rada i proces razvoja statističkih modela i pravila unutar ekspertskog sistema.

Drugi razlog zbog kojeg je organizacija rada samog tima veoma bitna je činjenica da ovako složen sistem koji ima za cilj praćenje rada jednog velikog produkcionog okruženja nije moguće napraviti u jednom koraku, već je za njegovo definisanje i razvoj potrebno više iteracija. Kada je potrebno kontinuirano održavanje sistema i postepeno doterivanje modela, neophodno je definisati proces razvoja ili neki njegov deo kako bi se osiguralo da napravljena promena ne izazove neke neželjene ili neočekivane efekte koji bi se ispoljili ka krajnjim korisnicima.

Kao što se može videti sa slike 6.1., proces definicije generičkih i kategorizacionih modela se sastoji od 5 faza [24]. Svaka od sledećih faza predstavlja odgovornost jednog ili više članova tima koji imaju odgovarajuće kvalifikacije:

- definisanje signala ili potpisa koji odražavaju konkretan način upotrebe ili tip problema koji se javlja (eng. ***thumbprint or pattern definition***),
- analiza definisanog potpisa,
- istraživanje potpisa i prototipska implementacija modela,
- verifikacija tačnosti definisanog modela u odnosu na potpis,
- implementacija modela sa optimizacijama.



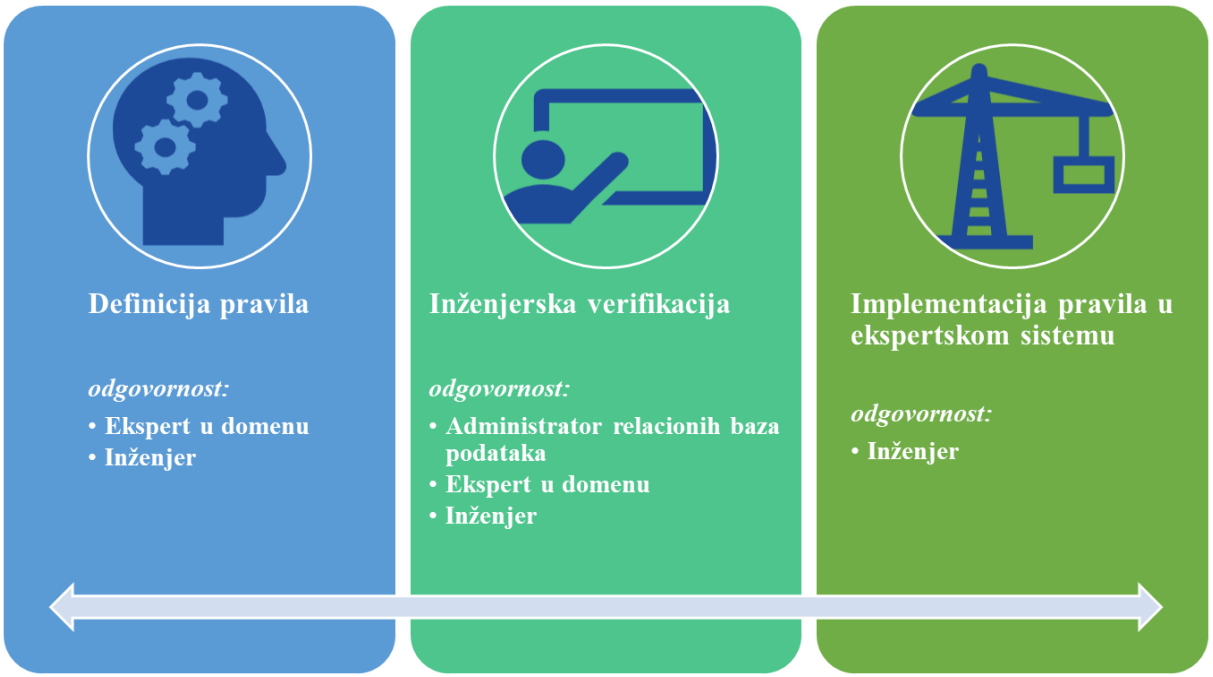
Slika 6.1. Proces implementacije modela sa podelom odgovornosti između aktera

U prvom koraku se vrši definisanje signala koji se mogu koristiti za prepoznavanje odgovarajućeg potpisa i koji će biti praćeni od strane modela. Ovaj korak uglavnom mogu da definišu akteri koji imaju znanja i veštine iz oblasti administracije ili poznavanja implemementacije relacionih baza podataka. Nakon definisanog potpisa problema, pristupa se njegovoj analizi. U ovom koraku akter koji ima znanja i veštine iz oblasti nauke o podacima analizira ponašanje signala, njegovu raspodelu, periodičnost, ponašanje na manjem ili većem uzorku relacionih baza podataka i slično. Tom prilikom se definiše i prva verzija modela. Nakon toga, pristupa se istraživanju rezultata koje je prototipski model proizveo nakon puštanja u rad u ograničenim uslovima testnog okruženja.

Ukoliko se u koraku istraživanja rezultata prototipske implementacije modela pokaže da analizirani potpis nije odgovarajući, definisanje i implementacija prototipskog modela se ponavlja vraćanjem u korak analize definisanog potpisa. U suprotnom, nastavlja se sa obradom rezultata od strane inženjera u koraku inženjerske verifikacije. Na kraju se pristupa implementaciji modela od strane softverskih inženjera koji pored same implementacije brinu i o performansama izvršavanja samih modela.

Posebno je bitno napomenuti da se iz svakog koraka proces vraća u prethodne korake, ukoliko akteri u okviru radnji koje obavljaju tokom ovih koraka iskažu zabrinutost za kvalitet rezultata, prvobitnu definiciju signala ili potpisa. Na taj način se osigurava da jedan definisani model na bazi konkretnog potpisa razmotri i pregleda više različitih aktera kako bi se postigao što bolji krajnji rezultat.

Vrlo sličan inženjerski pristup se primenjuje i tokom implementacije pravila u okviru ekspertskog sistema. Tokom definicije i implementacije pravila za ekspertski sistem prethodni proces je redukovan u podskup operacija koje su prikazane na slici 6.2. Prvo se definiše pravilo na osnovu ulaznih podataka od strane eksperta u domenu ili od strane softverskog inženjera koji se razume u oblast koje dato pravilo prepoznaje.



Slika 6.2. Proces implementacije pravila u ekspertskom sistemu sa podelom odgovornosti

Zatim se u fazi inženjerske verifikacije uzima slučajan uzorak intervala sa rezultatima modela i razmatraju se rezultati koji se dobijaju primenom definisanog pravila ekspertskog sistema od strane eksperata koji imaju znanja i veštine u oblasti administracije relacionih baza podataka ili u oblasti implementacije relacionih baza podataka. Verifikacija se vrši sa ciljem potpunog razumevanja konačanog uzroka problema u tom konkretnom intervalu. Nakon potpunog razumevanja sledi faza implementacije pravila u ekspertskom sistemu. U toj fazi se vrši konkretno dodavanje novog pravila u bazu znanja tj. bazu pravila ekspertskog sistema.

Potreba za ovakvim pristupom se pojavila nakon nekoliko neuspešnih iteracija prilikom definicije početnih modela. Podela na uloge nije postojala, već su svi akteri procesa radili sve i samim tim njihove oblasti znanja i interesovanja nisu upotrebljavane na pravi način. Jednostavan primer neefikasnosti se mogao uočiti prilikom razvoja jednog od modela, od strane *data scientist-a*, koji nije mogao da se izvrše nad skladištem podataka jer se nije obratila pažnja na potrebne optimizacione korake. Sa druge strane, inženjer koji je razvijao jedan drugi model nije obraćao dovoljno pažnje na njegovu preciznost već na efikasnost izvršavanja.

U daljem istraživanju utvrđeno je da se pomenuti procesi mogu prikazati i uz pomoć standardizovanih metoda. Pregledom literature, pronađeni su metodi organizacije vrlo slični modelima prikazanim na slikama 6.1. i 6.2., a opisuju se putem RACI [62] ili RACIS [63] matrica

[64]. Njihova glavna primena je u procesima kod kojih postoje različite uloge aktera koji utiču na sam proizvodni proces koji se sastoji od različitih faza. Smatra se da upotreba matrica, zahvaljujući jednostavnom vizuelnom prikazu, pospešuje šanse za uspešnost samog projekta koji se prati.

Akronim RACI(S) predstavlja skraćenicu za:

- R – (eng. ***responsibility***) – odgovornost za izvršavanje aktivnosti ili zadatka,
- A – (eng. ***accountability***) – odgovornost za planiranje aktivnosti ili celokupnog projekta,
- C – (eng. ***consulting***) – akteri koji mogu biti konsultovani u toku date aktivnosti,
- I – (eng. ***information***) – informisanje aktera tokom date aktivnosti,
- S – (eng. ***supportability***) – akteri koji su zaduženi za održavanje.

Prilikom popunjavanja datih aktivnosti, za svaku od aktivnosti se analiziraju sledeći aspekti:

- Da li za posmatranu aktivnost postoji bar jedan akter koji je odgovoran (R) za izvržavanje date aktivnosti?
- Da li za posmatranu aktivnost postoji previše odgovornih (R) aktera za izvršavanje?
    - Smatra se da previše odgovornih aktera može dovesti do usporavanja izvršenja aktivnosti usled međusobne korespodencije između aktera.
- Da li postoji bar jedan akter koji je odgovoran (A) za planiranje aktivnosti i celokupnog projekta?
    - Smatra se da je najefikasnije ukoliko postoji tačno jedan akter koji je odgovoran za planiranje i donošenje odluka tokom razvoja projekta.
- Da li je previše aktera konsultovano u toku date aktivnosti?
    - Ne postoji tačan broj koji se može povezati sa rečju previše u ovom slučaju ali cilj je da postoji što manji broj aktera koji se konsultuju kako bi se postigla što veća autonomija i kako bi akteri mogli samostalno da izvršavaju datu aktivnost.

Konkretna definicija RACIS matrice za potrebe razvoja AID-TS sistema prikazana je u tabeli 6.1. Kao što se može videti u datoj tabeli, za razliku od prethodno definisanih aktera na slikama 6.1. i 6.2., u slučaju RACIS matrice pojavljuju se i dva nova tipa aktera: projektant / arhitekta sistema i inženjer sistemske podrške. Oni prethodno nisu pominjani zbog toga što se njihova uloga podrazumevala u samom timu.

Uloga inženjera sistemske podrške je da pomogne ostalim akterima tokom istraživanja i verifikacije u produkcionom okruženju. Imajući u vidu da inženjeri sistemske podrške imaju dodatna prava pristupa samom sistemu koja, ukoliko ne postoji kompletna telemetrija, mogu olakšati razumevanje problema direktnim pristupom.

Kada se govori o ulozi projektanta ili arhitekte sistema, pored odgovornosti za donošenje odluka, taj akter je zadužen za:

- definisanje načina organizacije tima,
- definisanje arhitekture sistema,
- izbor tehnologija koje će biti upotrebljene u implementaciji.

Sa druge strane, primećuje se i nešto drugačija raspodela odgovornosti koja je dobijena nakon analize datih aktivnosti shodno prethodno definisanim pitanjima, a u cilju bolje organizacije samog

tima. Tako, na primer, softverski inženjer više nije odgovoran za aktivnost definicije potpisa i pravila, već se to očekuje od administratora relacionih baza podataka ili eksperta u domenu.

Tabela 6.1. RACIS matrica kod organizacije tima za razvoj inteligentnih sistema

Uloge / Aktivnosti	Administrator relacione baze podataka	Ekspert u domenu	Softverski inženjer	Data Scientist	Projektant/Arhitekta Sistema	Inženjer sistemske podrške
Definicija potpisa	**R**	**R**		**C**	**A**	
Definicija pravila	**R**	**R**			**A**	
Analiza definisanog potpisa		**C**	**I**	**R**	**A**	
Istraživanje rezultata početnog modela		**C**	**I**	**R**	**A**	**S**
Inženjerska verifikacija	**I**	**I**	**R**		**A**	**S**
Implementacija modela		**C**	**R**	**C**	**A**	
Implementacija pravila u ekspertskom sistemu		**C**	**R**	**C**	**A**	

R – odgovornost za izvršavanje
A – odgovornost za planiranje
C – konsultantska pomoć
I – zahtev za informacijom
S – pomoć u vidu podrške prilikom obavljanja operacije

Zatim, analizom definisanog potpisa i izvršavanja početnog modela i istraživanja njegovih rezultata, se isključivo bave *data scientist*-i dok se eksperti u domenu mogu konsultovati, a softverski inženjeri informisati o samim rezultatima. Time se softverski inženjeri mogu unapred pripremiti za fazu verifikacije, a potom i za fazu implementacije.

U tim kasnijim fazama, konsultante predstavljaju eksperti u domenu i *data scientist*-i sa kojima se softverski inženjeri uglavnom konsultuju oko konverzije modela iz prototipa u produkcioni model. U tom postupku se lako može desiti da dođe do smanjenja preciznosti, tako da se može pojaviti potreba za dodatnim konsultacijama.

7. Implementacija sistema na definisanoj arhitekturi

Arhitektura definisana u prethodnim poglavljima implementirana je na konkretnoj klaud platformi. U tu svrhu izabrana je platforma kompanije Microsoft pod nazivom Azure zbog nekoliko konkretnih prednosti u odnosu na druge platforme. Azure platforma nudi desetine različitih infrastrukturnih servisa koje je moguće iskoristiti za implementaciju samog AID-TS sistema. Sa druge strane, Azure SQL relaciona baza podataka kao servis koji je izvor podataka predstavlja jednu od najvećih platformi sa najraznovrsnijim relacionim bazama podataka iz perspektive njihove upotrebe koje pripadaju stotinama hiljada korisnika. Samim tim, veća je i potreba za automatizacijom detekcije i rešavanja problema od strane korisnika ali i od strane inženjera sistemske podrške.

U okruženju Azure SQL, telemetrija se već čuva u postojećem mlakom skladištu podataka, koje se naziva Azure Data Explorer [65]. Ovo skladište podataka je zasnovano na tehnologiji čuvanja podataka u operativnoj memoriji što veoma skraćuje izvršavanje upita. S obzirom da se ovaj izvor pokazao izuzetno pouzdanim za druge potrebe poput usputnih analiza, sistem je upotrebljen i za svrhu kontinuiranog praćenja. Podatke sa SQL servera kao izvora podataka do Azure Data Explorera kao skladišta treba preneti na odgovarajući način. Za prenos podataka koristi se Azure Event Hub koji podržava direktnu integraciju [66] sa Azure Data Explorer-om.

Odgovarajuće upite nad skladištem podataka Azure Data Explorer izvršavaju mikroservisi. Svaka instanca mikroservisa asocirana je sa jednim ili više regiona koji sadrže relacione baze podataka. Ti servisi izvršavaju modele u paraleli i čuvaju rezultate u relacionoj bazi podataka, koja je deo platforme Azure SQL. U zavisnosti od opterećenja regiona, broj instanci mikroservisa je moguće povećati čime se smanjuje pritisak na pojedinačne instance servisa. Particionisanje se može vršiti po grupi regiona, po jednom regionu ili po podskupu relacionih baza podataka unutar jednog regiona.

Relaciona baza podataka koja se koristi za čuvanje rezultata ovog sistema je geo-replicirana upotrebom tehnologije objašnjene u [67], kako bi se povećala dostupnost u slučaju otkaza celog regiona iz bilo kojeg razloga. Ovo je vrlo važno s obzirom da ostatak Azure SQL servisa, u kom relacione baze podataka u svakom trenutku mogu imati probleme različitog tipa, očekuje da sistem za detekciju problema kontinuirano radi i pruža analizu problema u približno realnom vremenu. Na kraju, svi ovi rezultati se dostavljaju korisniku u odgovarajućem obliku kao što je to specificirano dokumentacijom u [68]. Korisnik može da konzumira rezultate preko različitih informacionih kanala u zavisnosti od podešavanja.

Ovakva vrsta infrastrukture je upotrebljena dva puta. Identična infrastruktura se može izvršavati u paraleli. U tom repliciranom okruženju se testiraju novi modeli, nove verzije postojećih modela i nove verzije infrastrukture. Na taj način se testiranje vrši pre puštanja u produkciono okruženje. Veoma je važno voditi računa o ažuriranju sistema na adekvatan način i potrebno je da se greške što ranije primete i eliminišu kako bi se na taj način izbegao uticaj na krajnje korisnike sistema.

U nastavku poglavlja prethodno definisani AID-TS sistem je predstavljen kroz opis nekih konkretnih detalja implementacije.

7.1. Telemetrija

Kao što je već rečeno, izvor podataka su same relacione baze podataka. Odgovarajući agenti koji su deo Azure SQL servera, čitaju statistike koje relaciona baza emituje. Nakon čitanja podaci se šalju komponenti Azure Event Hub koji te podatke prosleđuje do komponente Azure Data Explorer. Podaci koji su potrebni za detekciju problema se skupljaju periodično na svakih 5 minuta.

7.1.1. Signali dobijeni iz relacionih baza podataka

Postoji veliki broj različitih signala koji se dobija iz relacionih baza podataka i njihove okoline. Pod okolnim signalima se podrazumevaju sve informacije o ponašanju virtuelne i fizičke mašine na kojima se relaciona baza podataka nalazi. U uskom kontekstu relacionih baza podataka, najvažniji podaci koji se koriste kao signali su:

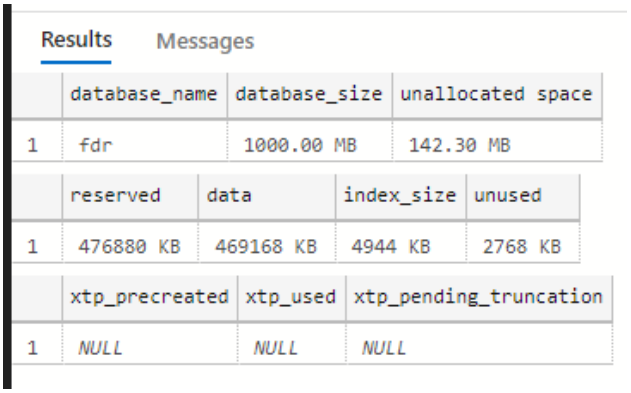
1. Veličina baze podataka pod kojom se podrazumeva isključivo veličina korisničkih podataka uključujući i veličinu transakcionog loga.
2. Broj ukupnih zahteva po sekundi, gde jedan zahtev predstavlja jedinicu izvršavanja iz pogleda SQL servera. To može biti jedan upit, grupa od više upita kao i poziv procedure ili funkcije.
3. Ukupno vreme čekanja u jedinici vremena na određenim tipovima čekanja. Svaki tip čekanja predstavlja jedan brojač koji bliže prikazuje zašto su zahtevi čekali tokom svog izvršavanja, ukoliko je čekanja bilo. S obzirom da vrednosti koje se dobijaju prosleđivanjem upita predstavljaju ukupno vreme čekanja od početka rada SQL servera potrebno je izvršiti obračun za poslednji interval, tj. poslednjih pet minuta. To se postiže tako što se u privremenoj tabeli unutar SQL servera čuva ukupno vreme čekanja od pokretanja SQL servera u trenutku kada je zapamćen prethodni interval. Zatim, ta vrednost se oduzima od

ukupnog vremena čekanja u trenutnom intervalu. Tako dobijene vrednosti se nazivaju ***delta*** vrednostima i one označavaju dužinu čekanja u periodu trajanja jednog intervala.

4. Broj aktivnih zahteva i aktivnih sesija tokom intervala gde broj aktivnih zahteva predstavlja aktivne zahteve koji se mapiraju na aktivne niti u trenutku uzimanja podataka, a broj aktivnih sesija predstavlja broj otvorenih konekcija između klijenta i SQL servera.
5. Broj grešaka kao i same greške koje se javljaju usled sistemske ili korisničke neodgovarajuće upotrebe SQL servera.

7.1.1.1. Upit za dohvatanje veličine baze podataka

Dohvatanje veličine baze podataka moguće je izvršiti pozivanjem sistemske SQL procedure pod nazivom ***sp_spaceused***. Rezultati koji se dobijaju izvršavanjem komande su prikazani su slici 7.1. Prvi red čine informacije o nazivu, ukupnoj veličini i prostoru koji trenutno nije rezervisan unutar relacione baze podataka. Sledeći niz čine tačne informacije o ukupnoj memoriji koja je zauzeta od strane svih objekata u relacionoj bazi podataka, zatim tačna veličina koju čine sami podaci, veličina struktura i sadržaj podataka koji sadrže indeksirane informacije i veličina trenutno alociranih slobodnih objekata koji nisu još uvek upotrebljeni. U poslednjem redu, ukoliko relaciona baza podataka poseduje tabele čiji se sadržaj čuva u operativnoj memoriji računara, informacije o veličini pre-kreiranih memorijskih struktura, njihovo iskorišćenje i ukupna veličina strukutura koje čekaju na brisanje nakon završetka transakcije su predstavljene respektivno.



Results Messages

	database_name	database_size	unallocated space
1	fdr	1000.00 MB	142.30 MB

	reserved	data	index_size	unused
1	476880 KB	469168 KB	4944 KB	2768 KB

	xtp_precreated	xtp_used	xtp_pending_truncation
1	NULL	NULL	NULL

Slika 7.1. Dobijeni rezultat prilikom izvršavanja SQL procedure ***sp_spaceused***

7.1.1.2. Upit za dohvatanje ukupnog broja zahteva

Dohvatanje ukupnog broja zahteva se prati uz pomoć sistemske tabele ***sys.dm_resource_governor_workload_groups***. Izvršavanjem upita koji je prikazan na slici 7.2. se prvenstveno ubacuje informacija o trenutnom stanju brojača o ukupnom broju zahteva od početka rada SQL server procesa u privremenu tabelu. Privremena tabela postoji od prvog izvršavanja upita do restartovanja SQL servera ili njegovog brisanja. Potom se nad privremenom tabelom izvršava upit koji izračunava razlike između svih susednih intervala i na taj način se dobija tačan broj zahteva u svakom od intervala. Uzorkovanjem poslednjeg intervala dobija se informacija o ukupnom broju upita koju agent telemetrije prosleđuje dalje.

```
INSERT INTO #temp_total_requests
   SELECT GETUTCDATE() AS Timestamp, total_request_count
   FROM sys.dm_resource_governor_workload_groups

SELECT TOP 1 * FROM
(
   SELECT Timestamp,
         (total_request_count -
               LAG(total_request_count, 1)
               OVER
               (ORDER BY Timestamp DESC)
         ) AS total_request_count
   FROM
         #temp_total_requests
) Deltas
WHERE
   total_request_count IS NOT NULL
```

Slika 7.2. Upit za izračunavanje ukupnog broja zahteva

7.1.1.3. Upit za dohvatanje vremena čekanja po raznim tipovima čekanja

Dohvatanje ukupnog vremena čekanja u jedinici vremena je moguće dobiti izvršavanjem upita koji je prikazan na slici 7.3. SQL server prati ukupna vremena čekanja u sistemskoj tabeli ***sys.dm_db_wait_stats*** od početka rada SQL servera. Periodičnim uzorkovanjem podataka iz ove tabele dobijaju se ***delta*** vrednosti za odgovarajući posmatrani interval slično kao i kod ukupnog broja zahteva.

```
INSERT INTO #temp_waits
   SELECT GETUTCDATE() AS Timestamp, *
   FROM sys.dm_db_wait_stats

   SELECT TOP 1 * FROM
   (
         SELECT Timestamp,
               (waiting_task_count -
                     LAG(waiting_task_count, 1)
                     OVER
                     (ORDER BY Timestamp DESC)
               ) AS waiting_task_count,
               (wait_time_ms -
                     LAG(wait_time_ms, 1)
                     OVER
                     (ORDER BY Timestamp DESC)
               ) AS wait_time_ms
         FROM
               #temp_waits
   ) Deltas
   WHERE
         waiting_task_count IS NOT NULL OR wait_time_ms IS NOT NULL
```

Slika 7.3. Upit za izračunavanje vremena čekanja po tipu čekanja

7.1.1.4. Upit za dohvatanje broja aktivnih sesija i zahteva

Broj aktivnih sesija i zahteva je moguće dobiti izvršavanjem upita nad sistemskom tabelom koja prati i ukupan broja zahteva pod nazivom ***sys.dm_resource_governor_workload_groups***. Upit za dobijanje vrednosti je predstavljen na slici 7.4. I u ovom slučaju se koriste privremene tabele za čuvanje informacija. Daljom obradom privremene tabele dobijaju se tačne vrednosti o broju aktivnih zahteva i uspostavljenih sesija u uzorkovanom vremenskom intervalu.

```
INSERT INTO
   #temp_active
   SELECT
         GETUTCDATE() AS Timestamp,
         active_request_count,
         active_session_count
   FROM
         sys.dm_resource_governor_workload_groups

SELECT TOP 1 * FROM
(
   SELECT
         Timestamp,

         (
               active_request_count -
                     LAG(active_request_count, 1)
                     OVER
                     (ORDER BY Timestamp DESC)
         ) AS active_request_count,

         (
               active_session_count -
                     LAG(active_session_count, 1)
                     OVER
                     (ORDER BY Timestamp DESC)
         ) AS active_session_count
   FROM
         #temp_active
)
Deltas
WHERE
   active_session_count IS NOT NULL
   OR
   active_request_count IS NOT NULL
```

Slika 7.4. Upit za izračunavanje broja aktivnih sesija i zahteva

7.1.1.5. **XEvent** sesija za praćenje grešaka

Praćenje informacija o greškama postiže se definisanjem **XEvent** sesije koja prati informacije o greškama i beleži ih u privremeni kružni bafer. To je moguće učiniti izvršavanjem skripte prikazane na slici 7.5. Skripta se sastoji od dva dela. Prvi deo briše sesiju ukoliko ona već postoji, dok se u drugom delu skripte pravi nova sesija za događaje pod nazivom ***sqlserver.error_reported***.

```
IF EXISTS(
   SELECT * FROM sys.database_event_sessions
   WHERE name='err_session'
)
   DROP EVENT session err_session ON DATABASE;

CREATE EVENT SESSION err_session
ON DATABASE
   ADD EVENT
         sqlserver.error_reported
   ADD TARGET
         package0.ring_buffer
               (SET
                     max_memory = 1024
               );

ALTER EVENT SESSION err_session
   ON DATABASE
   STATE = START;
```

Slika 7.5. Komanda za pravljenje **XEvent** sesije

Zatim, definisanoj sesiji je moguće pristupati upitom prikazanom na slici 7.6. Ekstrakcija podataka daje informacije o kodu greške i odgovarajućoj poruci, kao i o tačnom vremenskom trenutku kada se greška dogodila.

```
SELECT
   Timestamp,
   ErrorInfo.value('(/event/data/value)[1]', 'BIGINT') AS ErrorNumber,
   ErrorInfo.value('(/event/data/value)[8]', 'NVARCHAR(MAX)') AS ErrorMessage
FROM (
   SELECT
         ed.value('@timestamp', 'datetime') AS Timestamp,
         ed.query('.') AS ErrorInfo
   FROM
   (     SELECT CAST(xet.target_data AS xml) as target_data
         FROM
               sys.dm_xe_database_session_targets AS xet
         JOIN
               sys.dm_xe_database_sessions AS xe
               ON (xe.address = xet.event_session_address)
         WHERE xe.name = 'err_session'
   )
   AS T1
   CROSS APPLY
   target_data.nodes('//RingBufferTarget/event') t(ed)
) T2
```

Slika 7.6. Upit za dohvatanje vrednosti sačuvanih tokom trajanja **XEvent** sesije

7.1.1.6. **QueryStore** signali

Drugi **QueryStore** signali se prikupljaju i vezani su za statistike na nivou pojedinačnih upita koji se izvršavaju u relacionim bazama podataka. To su informacije koje se prate putem skladišta statistika upita (eng. ***query store***) [36]. Najznačajnije su informacije koje se dobijaju iz sledećih sistemskih tabela:

- ***sys.query_store_runtime_stats_interval*** – sadrži informacije o intervalu u kom su praćene statistike izvršavanja upita,
- ***sys.query_store_runtime_stats*** – sadrži informacije o statistikama konkretnog upita koji se prati,
- ***sys.query_store_wait_stats*** – sadrži informacije o tipovima čekanja i vremenu čekanja specifičnih upita.

7.1.1.7. Ostali dostupni signali

Pored statistika upita, iz perpsektive performansi važne informacije se prikupljaju i na nivou operativnog sistema virtuelnih mašina. Primer predstavljaju različiti brojači koji prate ponašanje virtuelnog procesora, opterećenost memorije, protok na virtuelnim mrežnim karticama i slično (eng. ***performance counters***). Operativni sistem Windows pruža aplikativni interfejs putem kojeg je moguće definisati upite za uzorkovanje odgovarajućih brojača.

Takođe, isti pristup se primenjuje i na nivou fizičkih mašina, gde se izdvajaju i prate podaci koji opisuju stanje hardvera. Primer takvih podataka čine informacije o brzini čitanja podataka sa diska, greškama koje se javljaju prilikom čitanja radne memorije i tome slično.

7.1.2. Prenos podataka preko komponente Azure Event Hub

Nakon prikupljanja podatka koje emituju relacione baze podataka, potrebno je iste preneti na adekvatan i što brži način do skladišta podataka. U konkretnoj implementaciji u tu svrhu se koristi Azure Event Hub [69]. To je komponenta koja može da primi i procesira milione podataka u svakom trenutku i koristi se pre svega zbog svoje skalabilnosti i činjenice da njegovo postojanje u celokupnom sistemu prenosa podataka dodaje najmanje moguće vreme.

Na slici 7.7. je predstavljen primer programskog dela koda koji pročitane podatke dalje emituje i šalje ka komponenti Azure Event Hub. Slični delovi koda se koriste i za emitovanje i slanje ostalih podataka sa jedinom razlikom u samom sadržaju i u destinaciji na koju se šalju.

S obzriom da Azure Event Hub podržava integraciju sa ostatkom Azure eko sistema, jednostavnom konfiguracijom se može povezati konkretan Azure Event Hub sa više različitih destinacija od kojih je Azure Data Explorer jedna od njih u ovom konkretnom slučaju [66].

```
1  using (var errorClient = new EventHubProducerClient(endpoint,
   errorEventHubName))
2  {
3    foreach (var err in errors)
4    {
5      var record = string.Format("{{\"timeStamp\": \"{0}\", \"errorNumber\":
   \"{1}\", \"errorMessage\": \"{2}\" , \"source\": \"{3}\"}}\n",
6        DateTime.UtcNow,
7        err.Number,
8        Err.Message,
9        err.SourceResourceId);
10
11       EventData eventData = new EventData(Encoding.UTF8.GetBytes(record));
12       eventData.Properties.Add("Table", "SqlErrors");
13       eventData.Properties.Add("Format", "json");
14
15       using EventDataBatch eventBatch = producerClient.CreateBatch();
16       eventBatch.TryAdd(eventData);
17
18       producerClient.SendAsync(eventBatch);
19   }
20 }
```

Slika 7.7. Primer programskog koda za slanje podataka o greškama ka Azure Event Hub

7.2. Skladište podataka Azure Data Explorer

Azure Data explorer [65] predstavlja skladište podataka koje svojim korisnicima omogućava brzo i jednostavno skaliranje. Zbog prirode arhitekture i same implementacije skladišta podataka, najviše korišćeni podaci su dostupni u radnoj memoriji. Činjenica da su podaci stalno dostupni u radnoj memoriji omogućava izuzetno brzo izvršavanje upita nad celokupnim skladištem podataka.

Skaliranje skladišta podataka za ceo Azure SQL servis je izvršeno tako da svaki region poseduje jednu svoju instancu skladišta podataka. Podaci se od izvora u skladište, kao što je to već napomenuto, prebacuju upotrebom komponente Azure Event Hub.

Bitno je napomenuti da je pre samog povezivanja potrebno napraviti skladište podataka što je moguće učiniti putem Azure internet portala. Sintaksa koja se upotrebljava za pisanje upita se naziva ***Kusto Query Language*** (skr. ***KQL***). Formiranje tabela unutar skladišta podataka se postiže jednostavnom komandom čija je sintaksa predstavljena na slici 7.8.

```
1  .create table SqlErrors (
2     Timestamp: datetime,
3     ErrorNumber: int,
4     ErrorMessage: string,
5     Source: string
6  )
```

Slika 7.8. Primer komande za pravljenje tabele za čuvanje informacije o greškama unutar skladišta podataka Azure Data Explorer

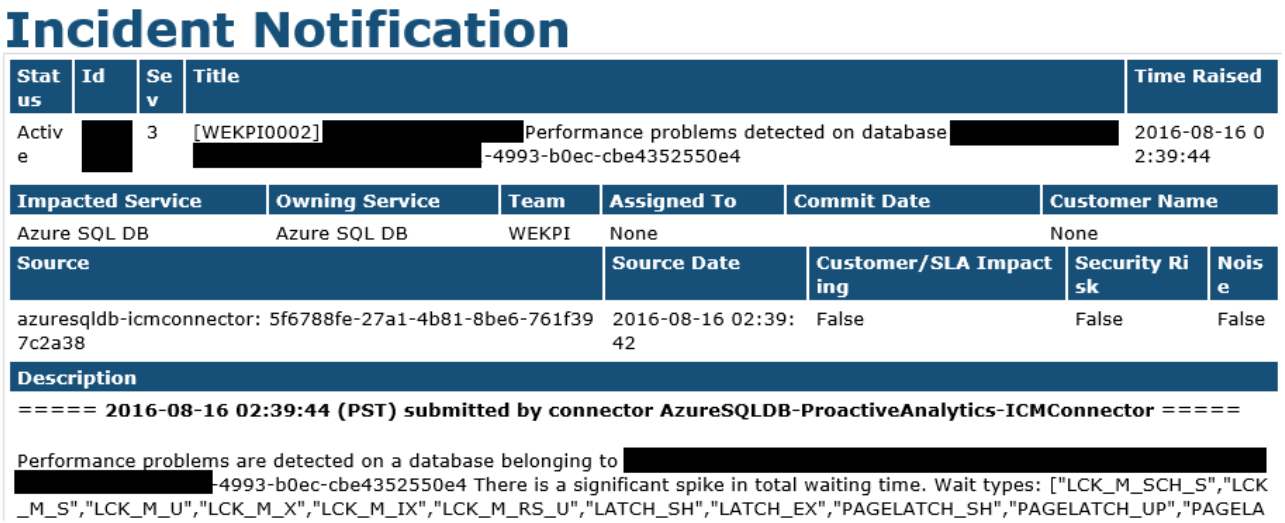
Komanda treba da sadrži naziv tabele koja se pravi, a potom u zagradama se definišu kolone sa pridruženim tipom podataka za svaku od kolona. U datom primeru, prva kolona pod nazivom *Timestamp* se odnosi na datum i vreme kada se greška desila. Zatim slede kolone koje sadrže informaciju o numeričkoj vrednosti greške i samoj poruci greške koja se desila. Na kraju kolona pod nazivom *Source* sadrži informaciju o serveru i relacionoj bazi podataka na kojoj se greška desila.

Nad podacima unutar skladišta podataka pokreću se odgovarajući upiti koji predstavljaju definisane modele. Sam KQL jezik je izuzetno jednostavan za pisanje modela. Takođe, omogućava i korišćenje naprednih funkcija kao što je, na primer Welch-T test koji se upotrebljava u **QueryDuration** generičkom modelu za određivanje vrednosti ***t*** sa ciljem smanjenja broja pogrešno detektovanih slučajeva.

7.3. Modeli

Implementacija generičkih i kategorizacionih modela je sprovedena na osnovu rezultata dve vrste prethodnih analiza. Na sličan način, kao što je to urađeno u radovima [25] i [27], prva analiza je sprovedena nad postojećim prijavljenim slučajevima (108 slučajeva) koje su korisnci relacionih baza podataka Azure SQL prijavili inženjerima sistemske podrške kao problematične. Svi prijavljeni slučajevi su bili validni i predstavljali su konkretne probleme.

Druga vrsta analize je sprovedena nad posebno definisanim skupom relacionih baza podataka. Analizirani skup je sadržao 397 različitih slučajeva, koji su smatrani problematičnim na osnovu prvog definisanog generičkog modela. Na slici 7.9. predstavljen je primer jednog takvog slučaja gde je dobijena poruka putem mehanizma za uzbunjivanje, nakon čega je sprovedeno dalje istraživanje o tom konkretnom slučaju.



Incident Notification

Status	Id	Sev	Title	Time Raised
Active	[redacted]	3	[WEKPI0002] [redacted] Performance problems detected on database [redacted] [redacted]-4993-b0ec-cbe4352550e4	2016-08-16 02:39:44

Impacted Service	Owning Service	Team	Assigned To	Commit Date	Customer Name
Azure SQL DB	Azure SQL DB	WEKPI	None		None

Source	Source Date	Customer/SLA Impacting	Security Risk	Noise
azuresqldb-icmconnector: 5f6788fe-27a1-4b81-8be6-761f397c2a38	2016-08-16 02:39:42	False	False	False

Description

===== **2016-08-16 02:39:44 (PST) submitted by connector AzureSQLDB-ProactiveAnalytics-ICMConnector** =====

Performance problems are detected on a database belonging to [redacted] [redacted]-4993-b0ec-cbe4352550e4 There is a significant spike in total waiting time. Wait types: ["LCK_M_SCH_S","LCK_M_S","LCK_M_U","LCK_M_X","LCK_M_IX","LCK_M_RS_U","LATCH_SH","LATCH_EX","PAGELATCH_SH","PAGELATCH_UP","PAGELA

Slika 7.9. Poruka dobijena putem mehanizma uzbune
o relacionoj bazi podataka u kojoj je primećen problem

Kao što je predstavljeno u poglavlju posvećenom organizaciji tima, tokom definicije modela sprovedena je podela unutar tima između različitih aktera u sistemu na osnovu odgovornosti koje su samoj ulozi dodeljene. Prilikom definicije kategorizacionih modela primenjen je proces u široj literaturi poznat pod nazivom ***knowledge engineering*** [45]. Taj proces podrazumeva dugotrajan rad i konstantno unapređenje kroz više iteracija. Na samom početku, pokušana je implementacija jednog modela koji bi uhvatio sve različite problematične slučajeve, ali s obzirom da Azure SQL okruženje sa milionima relacionih baza podataka predstavlja bogato i raznovrsno okruženje iz aspekta upotrebe SQL servera, to nije bilo dovoljno. Dobijeni rezultati su jasno ukazali na potrebu za dodatnim modelima kako bi se došlo do tačnog razumevanja različitih problema koje sistem treba da detektuje. Prvobitni model je doživeo postepene promene i trenutno se skup modela sastoji od 5 generičkih i 11 kategorizacionih modela. Novodefinisani modeli su omogućili preciznu detekciju i razumevanje svih slučajeva iz analize.

Pošto je definisanje modela bio iterativan proces, AID-TS sistem je tako postavljen da se može jednostavno proširiti novim modelima. U slučaju otkrivanja postojećeg, do sada nepoznatog problema od strane sistema, ili u slučaju pojave nove vrste problema, vrlo je jednostavno definisati nove modele i implementirati ih u postojećem sistemu.

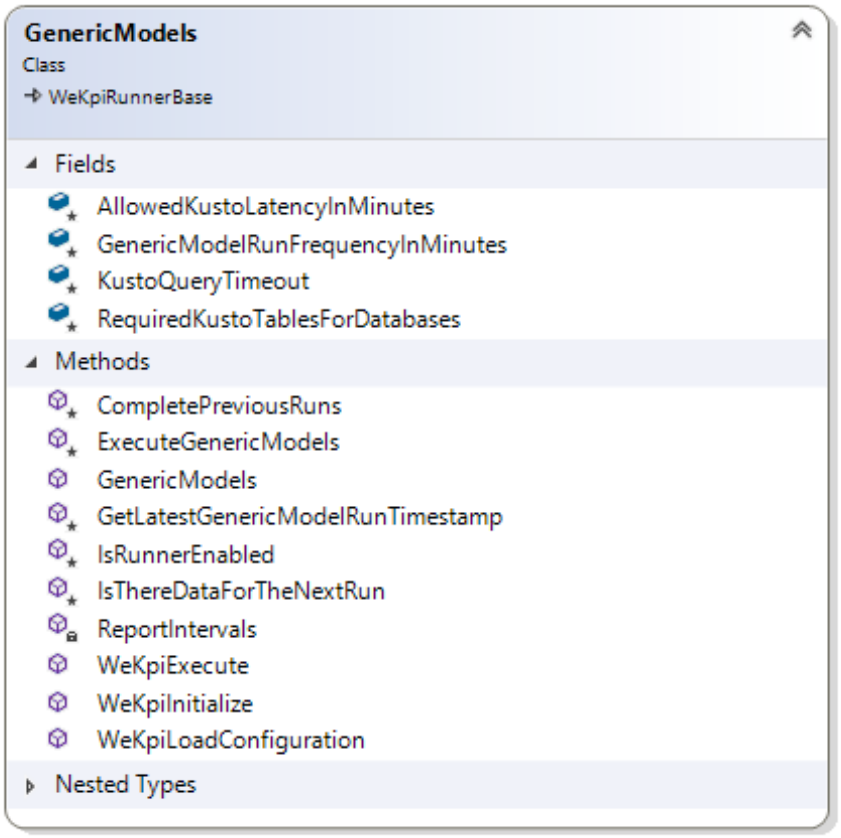
7.3.1. Generički modeli

U konkretnoj implementaciji, svi generički modeli su izvedeni iz osnovne klase koja je prikazana na slici 7.10. Klasa sadrži sledeće atribute:

- ***AllowedKustoLatencyInMinutes*** – sadrži dozvoljeno vreme kašnjenja podataka u skladištu podataka izraženo u minutima. Upotrebljava se pre izvršavanja modela za proveru sadržaja tabela unutar skladišta podataka.
- ***GenericModelRunFrequnecyInMinutes*** – predstavlja periodu izvršavanja modela.
- ***KustoQueryTimeout*** – predstavlja vreme za koje izvršavanje modela nad skladištem podataka mora da se završi. U suprotnom se smatra da je skladište podataka preopterećeno i izvršavanje modela se prekida.
- ***RequiredKustoTablesForDatabases*** – predstavlja listu tabela unutar skladišta podataka koji se upotrebljavaju u modelu.

Pored atributa, klasa sadrži i sledeće najvažnije metode:

- ***GenericModels*** – metoda koja vraća model koji instanca klase izvršava.
- ***ExecuteGenericModel*** – metoda koja izvršava konkretan model.
- ***IsThereDataForTheNextRun*** – metoda koja vrši proveru da li u skladištu podataka postoji dovoljno podataka kako bi model mogao da se izvrši za tekući vremenski interval.
- ***WeKpiExecute*** – metoda koja izvršava potrebne pripremne operacije i izabrane generičke modele.
- ***WeKpiInitialize*** – metoda koja inicijalizuje objekat date klase.



Slika 7.10. UML dijagram osnovne klase generičkih modela

Generički modeli se izvršavaju sa periodom od 15 minuta. Ovaj interval je izabran nakon detaljne analize postojećih podataka. Tokom te analize sledeća dva aspekta su uzimana u obzir:

- preciznost i pokrivenost generičkih modela i
- veličina upotrebljenih podataka, uključujući i sva ograničenja koja skladište podataka nameće.

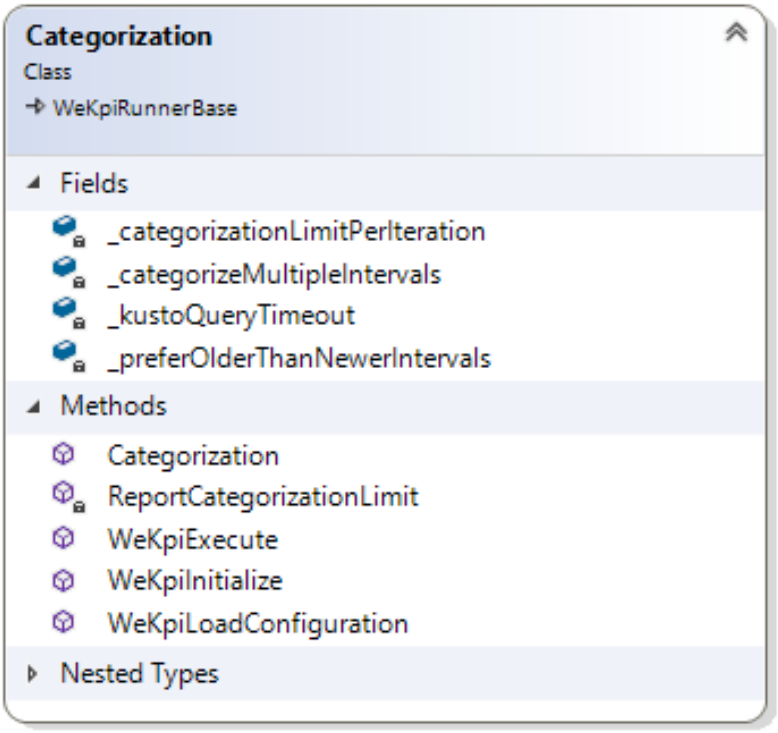
Osnovni period u svim modelima, kao što je to već rečeno, obuhvata period od 7 dana u bliskoj istoriji, gde se period od poslednjih 24 sata ignoriše i koristi se kao vremenski prozor. Ovaj prozor treba da pomogne da se dugotrajne manje anomalije lakše detektuju. Na taj način se daje mogućnost modelima da takve slučajeve uhvate u prvih 24 sata.

Slično kao u radu [52], vrednosti dobijene u osnovnom periodu se smatraju referentnim vrednostima koje karakterišu sistem u kojem ne postoji aktuelan problem. Vremenski period koji se posmatra proističe iz činjenice da nije moguće procesirati količinu podataka koja premašuje 7 dana, a da se pritom performanse skladišta podataka značajno ne naruše.

7.3.2. Kategorizacioni modeli

Kategorizacioni modeli se u odnosu na generičke modele razlikuju prvenstveno po ulaznim podacima. Kao što se može videti sa slike 7.11., osnovna klasa koja predstavlja kategorizacione

modele nasleđuje osnovnu klasu za modele. Ključna razlika je u tome što osnovna klasa „**Categorization**“ sadrži atribut sa svim intervalima koji su prosleđeni od strane generičkih modela na dalju obradu. Na osnovu tih intervala se određuje skup relacionih baza podataka nad kojima se vrši provera i oni se koriste kao ulazni parametri za kategorizacione modele.



Slika 7.11. Osnovna klasa kategorizacionih modela

7.3.3. Saradnja modela

Saradnja komponenti sistema opisana je jednostavnom procedurom koja je prikazana na slici 7.12. Ovaj algoritam prihvata kao ulazne parametre ime regiona i vremena početka i kraja intervala za koji se vrši analiza. Ime regiona se koristi za pristupanje odgovarajućoj banci podataka u kojoj se nalaze svi relevantni podaci za odgovarajući region. Na osnovu datog početka i kraja intervala vrši se izbor odgovarajućeg osnovnog perioda (eng. ***baseline period***). Bitno je napomenuti da iako je procedura prezentovana kao sekvencijalna, kritične sekcije je moguće paralelizovati kako bi se smanjilo vreme detekcije (skr. **TTD**).

Prvo, sistem iterira kroz sve generičke modele i dodaje detektovane intervale u listu svih intervala. Ukoliko je parametar ozbiljnosti problema ispod odgovarajuće granice interval se smatra korektnim (obeležava se kao ***non-issue***) i procesiranje se za taj interval završava. U suprotnom, interval se obeležava kao spreman za dalju obradu.

```
1  Inputs:
2  R: region name for which algorithm is executed
3  T: timestamp of end of the processed interval
4  Output:
5  I: set of detected intervals for particular databases including information
   about detected models, query hashes and provided root cause analysis
6
7  foreach model in generic_models do
8     I.addRange(model(R, T));
9  end
10
11 foreach Ij in I do
12    if (Ij<=minimum_severity) then
13          Ij.RCA.Class = NonIssue
14          Ij.State = Completed
15    else
16          Ij.State = ReadyForCategorization
17    end
18 end
19
20 foreach model in categorization_models do
21     model(R, I);
22 end
23
24 foreach Ij in I do
25     Ij.State = ReadyForExpertSystem
26 end
27
28 foreach Ij in i do
29    matchedRules = @()
30    foreach rulej in expert_system_rules do
31          if (rulej.match(Ij)) then
32                matchedRules.add(rulej)
33          end
34    end
35
36    Ij.RCA =AggregateMatchedRules(matchedRules)
37    Ij.State = Completed
38 end
```

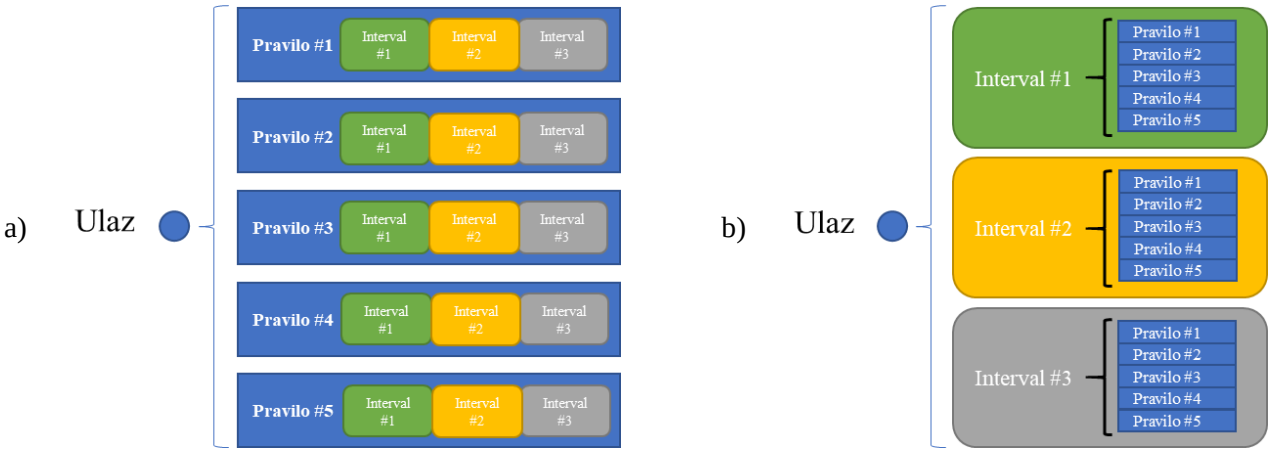
Slika 7.12. Algoritam detekcije i procesiranja problematičnih intervala za relacione baze podataka

Svim kategorizacionim modelima se potom prosleđuju intervali koje treba dalje ispitivati. Nakon izvršavanja modela i obrade intervala informacije o prepoznatim kategorijama se dodaju samom intervalu ukoliko je model prepoznao problem za specifičnu relacionu bazu podataka i odgovarajući interval.

7.4. Ekspertski sistem

Nakon izvršavanja svih modela, kao što se vidi sa slike 7.12., intervali se označavaju kao spremni za fazu u kojoj ekspertski sistem vrši procesiranje na osnovu svoje baze pravila. Procesiranje unutar ekspertskog sistema je u potpunosti paralelizovano. Sličan pristup sa paralelnim procesiranjem od

strane sistema je pomenut kao adekvatan pristup i u radu [47] gde se paralelizacija vrši na nivou pravila. U radu [24], kao i u ovom radu, postignut je maksimalan stepen paralelizacije na nivou jednog intervala i jednog pravila. Sličnosti i razlike ova dva pristupa se mogu videti na slici 7.13. U slučaju paralelizacije na nivou pravila (slika označena sa 7.13.a), svaka nit je odgovorna za određeno pravilo i vrši obradu više intervala. Sa druge strane, maksimalan nivo paralelizacije se postiže kada je jedna nit zadužena za jedan interval i koja se potom dalje deli i paralelizuje za svako pravilo koje postoji u bazi pravila (slika 7.13.b).



Slika 7.13. Uporedni prikaz paralelizacije procesiranja unutar ekspertskog sistema u a) radu [47] i b) ovom radu

Definicija pravila u ekspertskom sistemu i par primera iz baze pravila ekspertskog sistema su već predstavljeni u poglavlju 4.2. Ukoliko postoji pravilo koje zadovoljava ulazne parametre koji su dobijeni iz intervala, takvo pravilo se čuva u listi pravila. Na kraju, lista primećenih pravila se agregira na način opisan u 4.2. i konačan rezultat se isporučuje korisniku. Programski kod u programskom jeziku C# koji vrši agregaciju pravila prikazan je na slici 7.14.

Agregiranje rezultata se vrši u dva prolaza. Prvo se lista primećenih pravila sortira u opadajućem poretku na osnovu broja činjenica koja pravila sadrže. Zatim se uzima po jedno pravilo iz liste i poredi sa svim ostalim pravilima koji imaju manji ili jednak broj činjenica. Ako se nađe pravilo čije činjenice predstavljaju podskup nekog drugog skupa činjenica koji definišu drugo pravilo, takvo pravilo se izbacuje iz liste i procesiranje se nastavlja dalje.

Nakon završetka prvog prolaza, gde su eliminisana sva prepoznata pravila čija je definicija zasnovana na činjenicama koje se mogu svrstati u podskup činjenica nekog drugog prepoznatog pravila, u drugom prolazu se vrši provera da li su preostala pravila sačinjena od skupova činjenica disjunktna. Ukoliko to nije slučaj, odgovarajući parametar pod nazivom ***chainDetected*** se postavlja na vrednost ***true*** i prekida se dalja obrada intervala. Ovakva situacija se smatra ne prihvatljivom po definiciji sistema i ako do takvog slučaja dođe, mehanizmom uzbunjivanja se inženjerski tim obaveštava kako bi se izvršila analiza i sprovele korektivne mere u vidu redefinicije postojećih i definicije novih pravila.

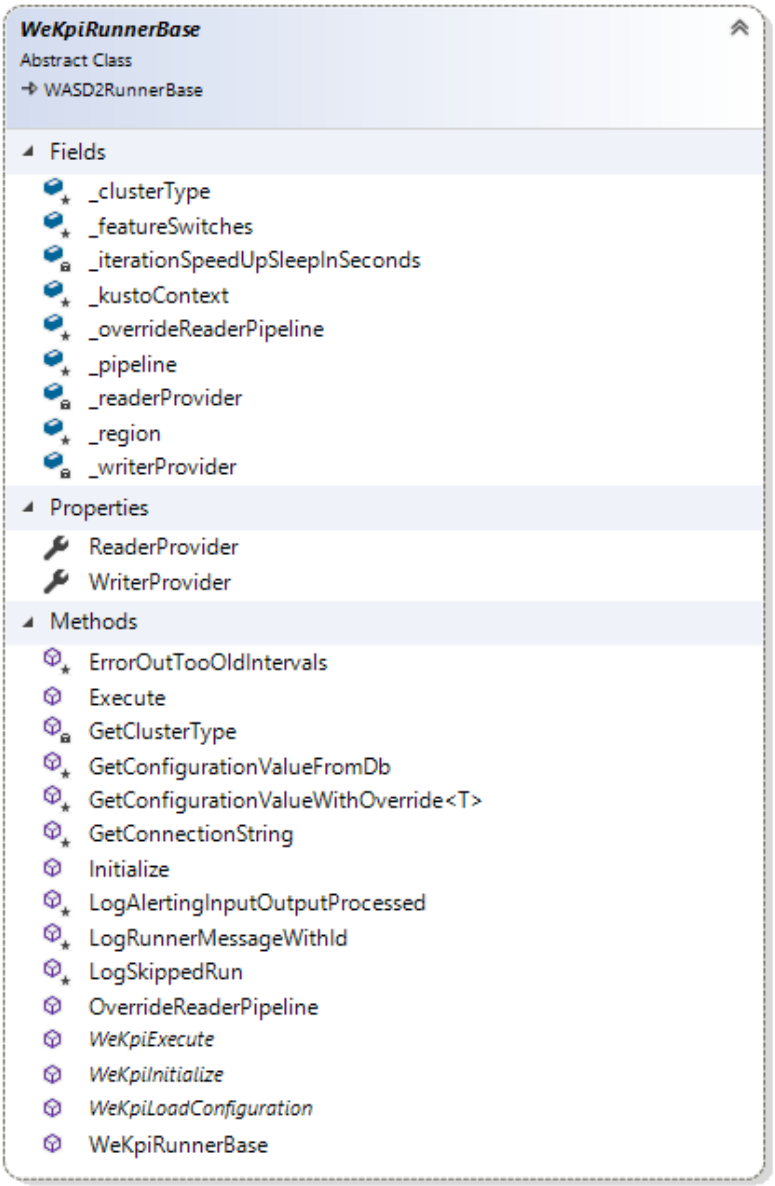
```
1  bool MergeSubResultsIntoSuperResults(ProcessingResult input)
2  {
3      bool chainDetected = false;
4
5      // Exclude subsets that are covered by supersets,
6      // leave sets that do not fall into superset.
7      //
8      input.ClassificationResult = input.ClassificationResult
9          .OrderBy(r => r.RuleSet.Facts.Count).ToList();
10
11     for (int j = 0; j < input.ClassificationResult.Count(); j++)
12     {
13         var result = input.ClassificationResult[j];
14
15         for (int i = j + 1; i < input.ClassificationResult.Count(); i++)
16         {
17             int matchedCnt = result.RuleSet.Facts
18                 .Intersect(input.ClassificationResult[i].RuleSet.Facts)
19                 .Count();
20
21             if (matchedCnt == result.RuleSet.Facts.Count)
22             {
23                 input.ClassificationResult.RemoveAt(j);
24                 j--;
25                 break;
26             }
27         }
28     }
29
30     // Check if there is still remaining chain reaction.
31     //
32     for (int j = 0; j < input.ClassificationResult.Count(); j++)
33     {
34         var result = input.ClassificationResult[j];
35
36         for (int i = j + 1; i < input.ClassificationResult.Count(); i++)
37         {
38             int matchedCnt = result.RuleSet.Facts
39                 .Intersect(input.ClassificationResult[i].RuleSet.Facts)
40                 .Count();
41
42             if (matchedCnt > 0 && matchedCnt != result.RuleSet.Facts.Count)
43             {
44                 chainDetected = true;
45                 break;
46             }
47         }
48
49         if (chainDetected)
50         {
51             break;
52         }
53     }
54
55     return chainDetected;
56 }
```

Slika 7.14. Programski kod za agregaciju pravila

7.5. Server i mikroservisi

Server je implementiran kao beskonačna petlja koja se aktivira u predefinisanim intervalima. Nakon svakog aktiviranja, na osnovu podataka iz relacione baze podataka, server zaključuje koje modele treba da pokrene i da zakaže njihovo izvršavanje na odgovarajućoj instanci mikroservisa.

Pokretanje mikroservisa instancira objekat klase **WeKpiRunnerBase** koja je predstavljena na slici 7.15. Tom prilikom se specificira nad kojim regionom ili nad kojom grupom intervala se izvršava određena analiza. Ova osnovna klasa poseduje i implementaciju polja za čitanje i pristup konfiguraciji sistema. Na osnovu konfiguracije ***featureSwitches*** određeni delovi programskog koda se aktiviraju ili preskaču prilikom izvršavanja sa glavnim cilljem bolje kontrole tokom uvođenja novih izmena. Pored konfiguracije, bazna klasa je zadužena i za instanciranje klijentskog objekta ***_kustoContext*** koji se koristi za komunikaciju sa skladištem podataka.



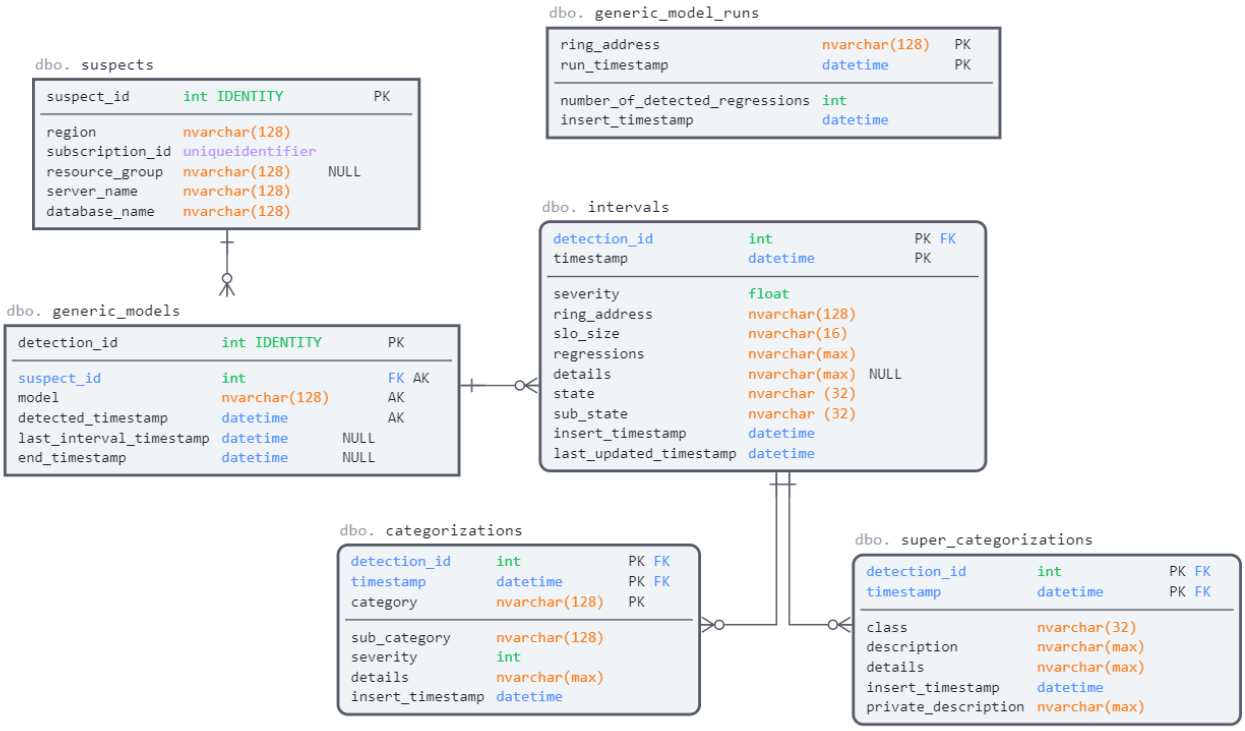
Slika 7.15. Osnovna klasa mikro servisa

Osnovna metoda ***Initialize***, pored učitavanja osnovne konfiguracije, sadrži pozive apstraktnih metoda ***WeKpiInitialize*** i ***WeKpiLoadConfiguration*** koje se implementiraju u konkretnim izvedenim klasama. Ove dve metode se koriste za dodatna podešavanja i učitavanje dodatnih konfiguracija koje specifični modeli mogu zahtevati. Na taj način se postiže granularniji pristup gde se može učitati specifičan upit za odgovarajući model ili učitati i inicijalizirati dodatne objekte klasa koje dalje apstrahuju izvršavanje samog konkretnog modela. Pored toga, nakon svih potrebnih učitavanja, u okviru metode ***Execute*** se nalazi i poziv ka apstraktnoj metodi ***WeKpiExecute*** koja u konkretnim modelima sadrži implementaciju izvršavanja samog modela.

U slučaju generičkih i kategorizacionih modela teži se visokoj paralelizaciji pa se za svaki od modela instancira odgovarajući objekat klase **WeKpiRunnerBase**. U slučaju ekspertskog sistema, zbog manje složenosti i vremena trajanja obrade podataka, instancira se isključivo jedan objekat u vremenskom intervalu, dok se paralelizacija vrši na nivou više niti koje se pokreću unutar jedne klase.

7.6. Relaciona baza podataka za čuvanje rezultata sistema

Kao što je već objašnjeno, za potrebe čuvanja podataka koje celokupni AID-TS sistem proizvodi i na osnovu kojih se prati stanje intervala upotrebljena je relaciona baza podataka Azurel SQL. Šema relacione baze podataka je predstavljena na slici 7.16.

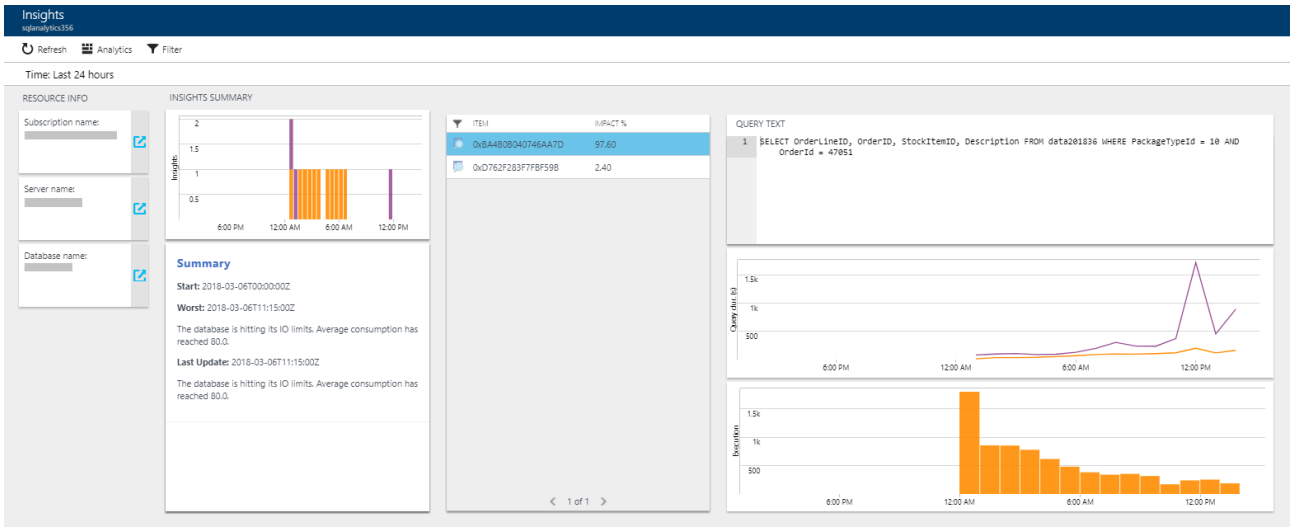


Slika 7.16. Šema relacione baze podataka

Osnovna tabela u kojoj se čuvaju podaci o relacionoj bazi podataka koja je detektovana od strane generičkih modela je na slici predstavljena pod nazivom ***suspects***. U njoj se čuvaju osnovne informacije poput naziva servera i naziva relacione baze podataka. Informacije o modelu koji je detektovao probleme u određenoj bazi podataka, kao i svi intervali u kojima je ta baza podataka primećena, čuvaju se u tabelama ***generic_models*** i ***intervals***, respektivno. Određeni interval može imati jednu ili više detektovanih kategorija koje se čuvaju u tabeli ***categorizations***. Na kraju, informacije dobijene obradom intervala od strane ekspertskog sistema se za svaki interval čuvaju u tabeli ***super_categorizations***. Tabela ***generic_model_runs*** služi za praćenje izvršavanja modela na osnovu koje se vrši zakazivanje obrade sledećih intervala i zakazivanje rada mikroservisa.

7.7. Objava rezultata

Na kraju procesiranja svi spomenuti podaci se objavljuju korisnicima kroz grafički interfejs koji se zove Azure SQL analytics. Primer izgleda grafičkog interfejsa je dat na slici 7.17. Na levoj strani predstavljen je prozor u kome se daje objašnjenje određenog problema, dok se na desnoj strani mogu videti svi upiti koji imaju neku vezu sa primećenom regresijom koja se desila u datom intervalu. Izborom upita iz liste svih upita može se dobiti i tekst samog upita kao i detaljnije informacije o njegovom izvršavanju tokom vremena.



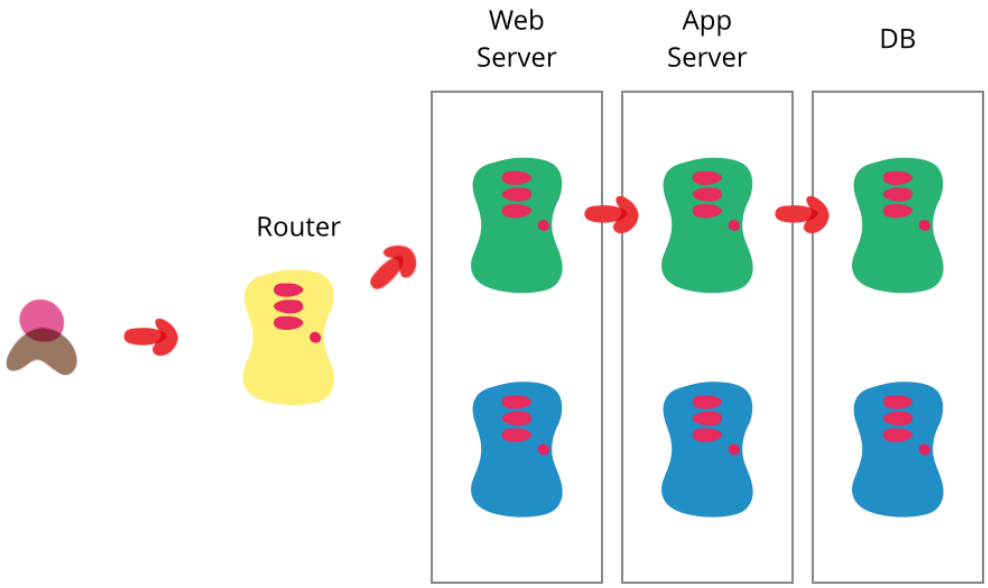
Slika 7.17. Azure SQL grafički interfejs za prikaz rezultata AID-TS sistema (slika preuzeta iz [70])

Pored grafičkog prikaza rezultata, informacije je moguće dobiti i putem drugih komunikacionih kanala kao što su SMS poruke ili elektronska pošta. Korisnik ima mogućnost da putem Azure portala izabere komunikacioni kanal kojim želi da prima ovakve informacije.

7.8. Kontinuirano testiranje

Pored aktivnog produkcionog okruženje A, postoji i testno okruženje B koje služi za pripreme promena i njihovo testiranje. Krajnji cilj se odnosi na pitanje da li promena koja je napravljena u sistemu donosi poboljšanje. Sličan pristup upotrebe još jednog okruženja za testiranje je diskutovan u radu [71]. Rad je fokusiran na veb servise i njihovo testiranje. Ključna razlika u odnosu na implementaciju u ovom radu se odnosi na činjenicu da se predlaže podela u različite grupe tako da i krajnji korisnici imaju različito iskustvo sa servisom koji upotrebljavaju. U AID-TS sistemu, okruženje B je isključivo upotrebljeno za analizu rezultata od strane inženjera i naučnika, a podaci iz ovog sistema se ne šalju krajnjim korisnicima. Pristup gde se B okruženje isključivo koristi za testiranje sistema se koristi u centrima kontrola letenja širom sveta kao što je to slučaj u rešenju ***TopSky – ATC*** kompanije Thales [72].

Ovaj pristup je sličan pristupu ***Blue/Green deployment*** prikazanom na slici 7.18., koji je prvi put definisan u [73]. U slučaju AID-TS sistema, A okruženje (***Blue***) je uvek zaduženo za isporučivanje rezultata korisnicima, dok je B okruženje (***Green***) uvek ono koje se koristi za testiranje. Pristup ***Blue/Green*** podrazumeva kontinuirano prebacivanje odgovornosti putem odgovarajućeg rutera (eng. ***router***), sa jednog na drugi sistem, kako bi se eliminisala potreba za dodatnim ažuriranjem sistema, što ovde nije slučaj.



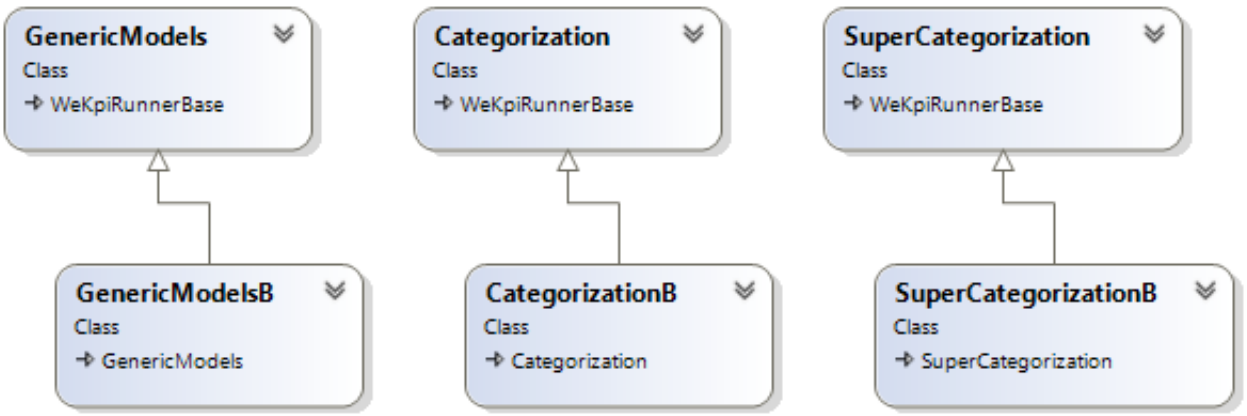
Slika 7.18. ***Blue/Green deployment*** pristup sa upotrebom rutera (slika preuzeta iz [73])

Naučnici i inženjeri imaju različite metrike i različite ciljeve kada vrše testiranje promena. Primarni fokus svakog naučnika je kvalitet modela. Oni koriste okruženje B da testiraju nove modele i da uporede postojeće modele sa njihovim novim verzijama koje sadrže odgovarajuće promene. Ovakav pristup im pomaže da dođu do zaključka da li promena u sistemu donosi pozitivne ili negativne efekte na krajnji rezultat. Sličan pristup se primenjuje i u toku inženjerskog procesa, ali sa jednom

bitnom razlikom. Primarni fokus inženjera je da pronađu greške i defekte u samom programskom kodu, da provere kakve su performanse prilikom izvršavanja modela i da li nova verzija sistema donosi sveobuhvatno poboljšanje ukupnog vremena potrebnog za detekciju i dostavljanje podataka krajnjem korisniku.

Nakon nekoliko dana izvršavanja promena u B okruženju, ukoliko se upoređivanjem rezultata i vremena izvršavanja iz A i B okruženja dobijaju rezultati koji pokazuju poboljšanje ili bar jednako dobro ponašanje sistema, nova verzija se propagira i u A okruženje. Na taj način se obezbeđuje odgovarajući tranzicioni period, a da se pritom vrši agilno testiranje koje nema negativan uticaj na izbacivanje novih verzija sistema.

Prilikom implementacije, okruženje B nasleđuje okruženje A. Informacije se čuvaju u dodatnoj paralelnoj relacionoj bazi podataka. UML dijagram koji opisuje primenjeno nasleđivanje predstavljen je na slici 7.19. Ovakav pristup je primenjen kako bi se olakšao dalji razvoj modela i sistema. Ako postoji potreba za testiranjem konkretne komponente u sistemu, ovakav koncept omogućava preusmeravanje i prihvatanje izlaza iz određene faze A okruženja. Tako replicirani ulazni podaci se mogu zatim koristiti u B okruženju za testiranje odgovarajuće sledeće faze.



Slika 7.19. UML dijagram okruženja A i B

8. Evaluacija

Proces evaluacije je sproveden tokom dužeg vremenskog perioda nad svim relacionim bazama podataka unutar Azure SQL okruženja kompanije Microsoft. Azure SQL platforma za relacione baze podataka se sastoji od više miliona instanci relacionih baza podataka čije su statistike i informacije o problemima prikupljeni za period od 6 meseci (period od 8 kalendarskih meseci sa pauzom od dva meseca). Posmatrane relacione baze podataka pripadaju različitim korisnicima pa je, samim tim, i njihova upotreba u nekim slučajevima slična, ali u nekim i potpuno drugačija čime je dobijen raznovrstan i primeren uzorak za analizu.

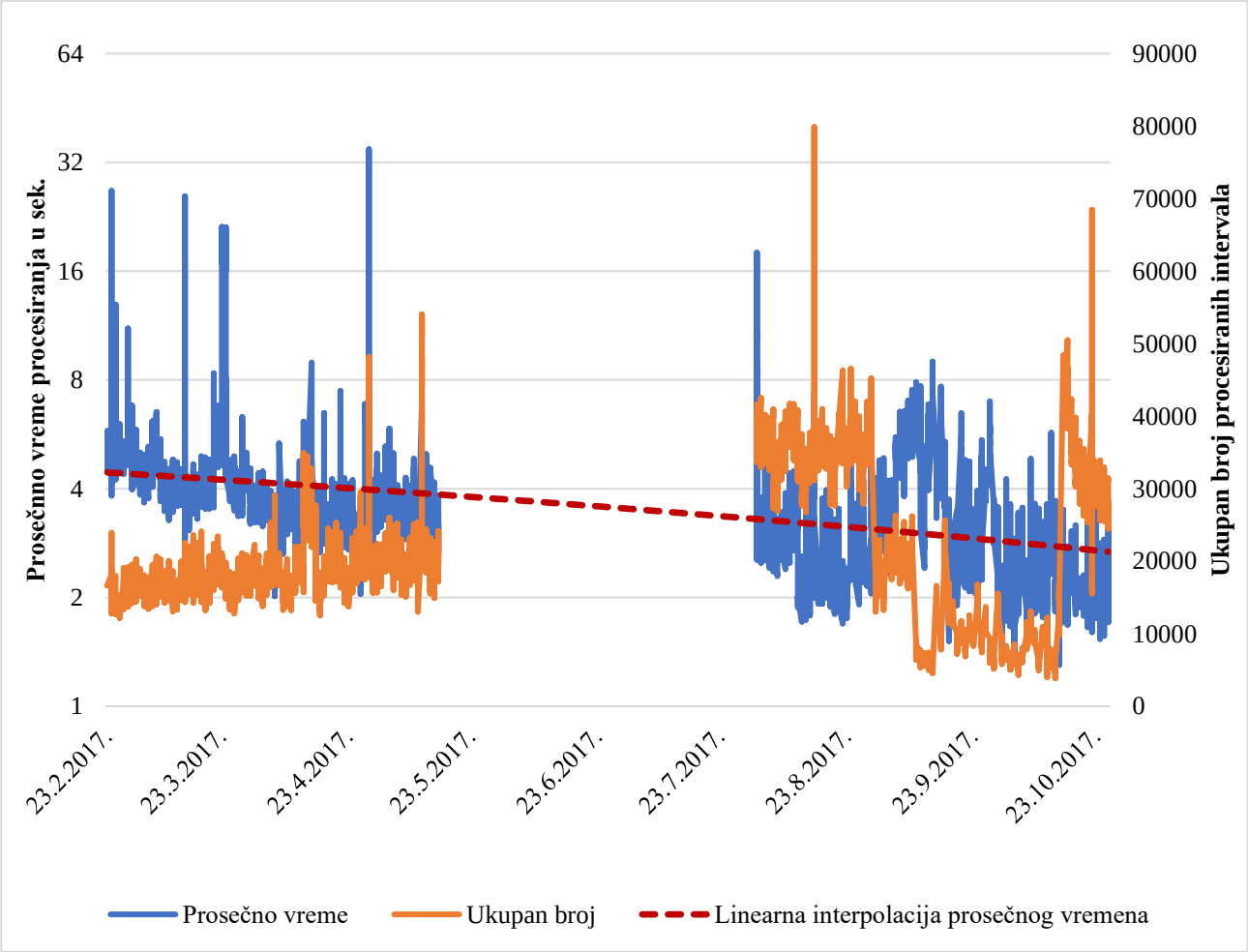
Implementirano rešenje je validirano iz različitih aspekata. Glavni indikator unapređenja iz korisničke perspektive je efikasnija upotreba relacione baze podataka, kada se upiti izvršavaju što je moguće brže i bez grešaka. Drugi važan indikator performanse koji je praćen se odnosi na vremena za detekciju, procesiranje i izveštavanje o problemu. Glavni parametar koji utiče na složenost sistema i potrebe za njegovim skaliranjem je broj intervala koji zahtevaju detekciju, procesiranje i detaljnu analizu. Dodatni bitan indikator je i primenljivost formiranog sistema na posao koji obavljaju inženjeri sistemske podrške i u kojoj meri im sistem u tome pomaže.

8.1. Evaluacija infrastrukture

U prvoj iteraciji iskorišćena je postojeća infrastruktura zasnovana na mikroservisima koja je već imala konfigurisano mesto za čuvanje rezultata unutar relacione baze podataka Azure SQL. Šema ove relacione baze podataka je bila jednostavna i imala je jednu tabelu u kojoj su se vrednosti čuvale u formatu 'ključ-vrednost'. Upotreba tog formata za potrebe realizovanog sistema nije optimalna i iskazala se kao jedno od uskih grla sistema.

Na slici 8.1., prikazano je poboljšanje prosečnog vremena za procesiranje detektovanih intervala. U prvom periodu je korišćen prethodni neoptimalni sistem dok je u kasnijem periodu korišćen novi, razvijeni sistem. Iako je broj intervala rastao, u isto vreme je prosečno vreme procesiranja zahteva linearno opalo (crvena isprekidana linija trenda).

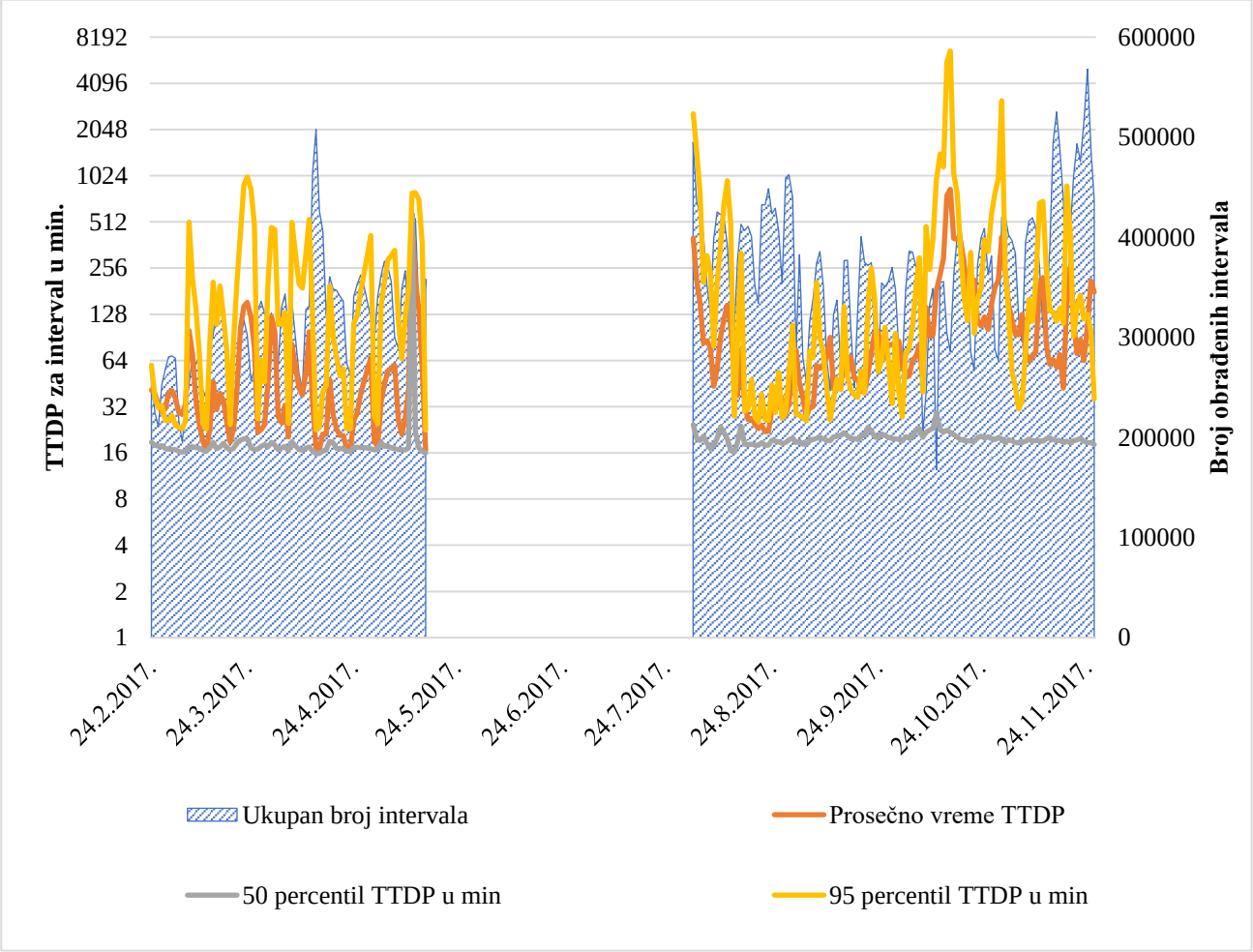
U početku, vreme potrebno za izvršavanja intervala je iznosilo u proseku 5 sekundi, dok je kasnije taj prosek spušten na skoro 2.5 sekunde. Treba imati u vidu da je broj obrađenih intervala u međuvremenu skočio 2 puta što znači da je postignuto ukupno ubrzanje od skoro 4 puta. Primećenom ubrzanju nije doprinela isključivo nova optimizovana šema relacione baze podataka u kojoj su čuvani podaci, već i dodatne optimizacije kategorizacionih i generičkih modela koje su implementirane.



Slika 8.1. Broj intervala i vreme procesiranja u periodu evaluacije

Tokom rada infrastrukture, veoma je bitno pratiti vreme za detekciju, procesiranje i izveštavanje o problemima (skr. **TTDP**). Ove metrike su korišćene kao glavni indikatori stabilnosti infrastrukture. Na slici 8.2. je predstavljen ukupan broj primećenih intervala i vremena potrebna za obradu tih intervala tokom posmatranog perioda. Vreme izvršavanja je predstavljeno putem tri različita indikatora i to su:

- prosečno vreme TTDP predstavlja prosečno vreme obrade svih posmatranih intervala u tom trenutku,
- 50ti percentil, predstavlja vrednost 50te grupe od ukupno 100 podeljenih grupa posmatranih intervala sortiranih po vremenu izvršavanja u rastućem poretku, što zapravo znači da se pola posmatranih intervala obradi brže ili za vreme 50te grupe,
- 95ti percentil, predstavlja vrednost 95te grupu od ukupno 100 podeljenih grupa posmatranih intervala sortiranih po vremenu izvršavanja u rastućem poretku, što zapravo znači da se većina posmatranih intervala obradi brže ili za vreme 95te grupe.



Slika 8.2. Vreme detekcije, procesiranja i objavljivanja intervala tokom perioda evaluacije

Vreme detekcije i obrade intervala u 50tom percentilu je konstantno i iznosi 16 minuta dok prosečno vreme varira i do 60 minuta. Ovakvo ponašanje sistema se objašnjava činjenicom da su modeli detektovali određene relacione baze podataka koje su konstantno imale neke od problema. Veći broj takvih relacionih baza podataka utiče na izvršavanje i nagomilavanje intervala koje je potrebno procesirati.

Pošto je procesiranje dva intervala za jednu bazu trenutno implementirano na sekvencijalan način, to zahteva dve ili više iteracija pa je samim tim vreme veće od 16 minuta, imajući u vidu da se

iteracije izvršavaju na svakih 15 minuta. U isto vreme, može se videti da je ceo proces za većinu relacionih baza podataka koje imaju neki problem završen u periodu od 15 do 20 minuta.

Na slici se može primetiti skok koji se desio 11.10.2017. Daljom analizom utvrđeno je da je taj skok korelisan sa instalacijom nove verzije SQL servera koja je u sebi imala skrivenu grešku. Ova greška je izazvala problem na više različitih relacionih baza podataka. Važno je istaći da je taj problem detektovan praćenjem rada AID-TS sistema od strane inženjerskog tima. Zahvaljujući sistemu opisanom u ovom radu, ta greška je brzo uočena i ispravljena.

Generalno gledano, kako bi se dalje skratila vremena izvršavanja, različiti pristupi se mogu primeniti. Neki od njih su paralelno procesiranje intervala za istu bazu, dalja optimizacija modela ili upotreba druge vrste skladišta podataka za čuvanje telemetrije.

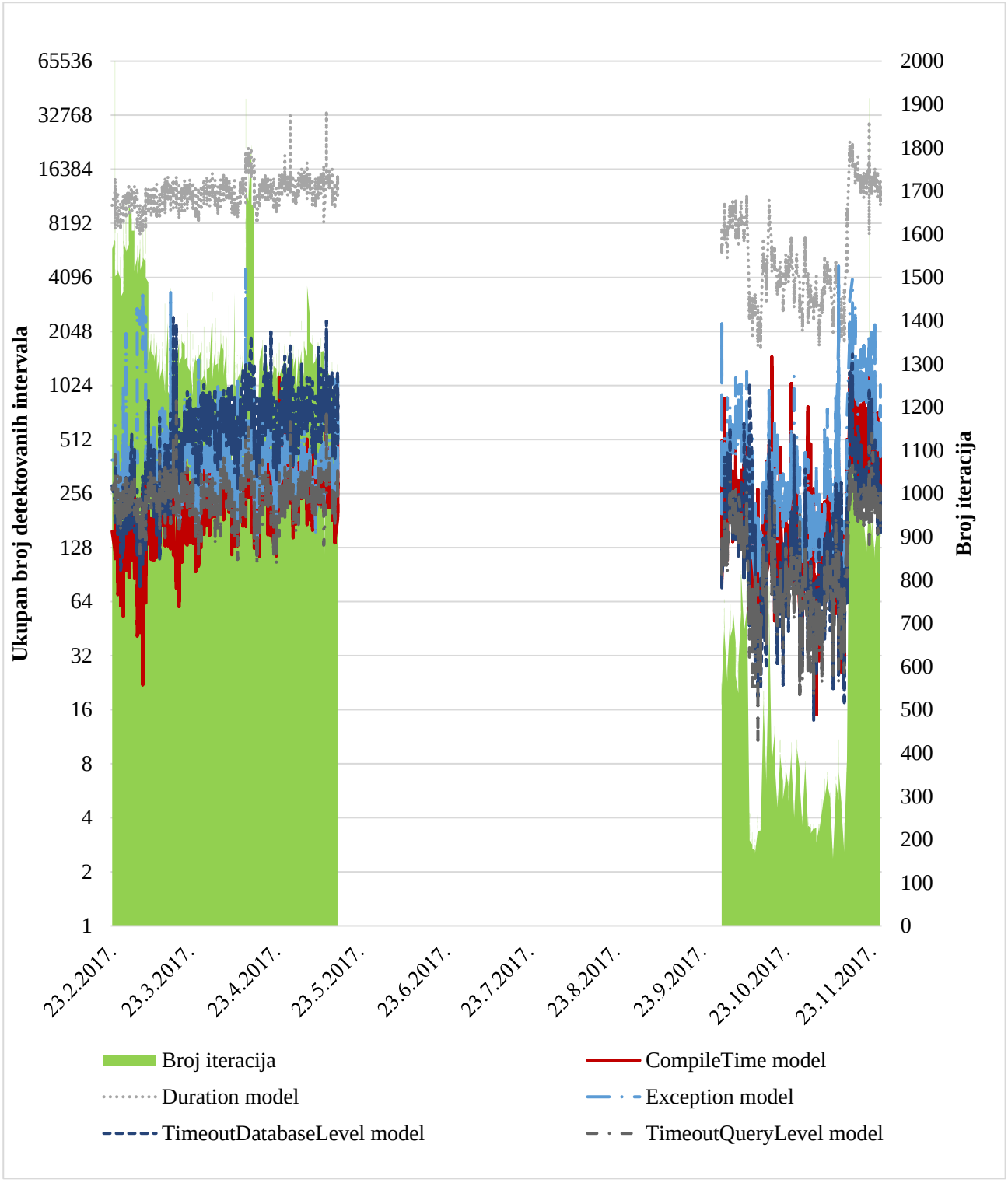
8.2. Evaluacija statističkih modela i ekspertskog sistema

Funkcionisanje novog AID-TS sistema zasnovanog na statističkim *data science* modelima i ekspertskom sistemu neprekidno su praćeni uz pomoć indikatora performansi. Broj iteracija po satu je korišćen kao jedan od indikatora i on predstavlja jasan pokazatelj koliko je koji model izvršavan na celom okruženju. Drugi praćeni indikator je broj detektovanih relacionih baza podataka od strane svakog od modela.

Vrednosti indikatora za generičke modele su predstavljeni na slici 8.3. Jasno se vidi da je u kasnijem periodu, počev od 30.09.2017. godine, broj iteracija značajno manji nego u ranijem periodu. Takav rezultat se može korelisati sa činjenicom da je broj različitih unapređenja modela učinjen u kasnijoj verziji sistema, što je povećalo pouzdanost sistema i smanjilo vreme izvršavanja modela. Pre smanjenja vremena izvršavanja često se dešavalo da se, usled greške ili isteklog očekivanog vremena za obradu od strane skladišta podataka, isti model za isti interval izvršavao više puta, pa je samim tim i ukupan broj iteracija po satu bio veći.

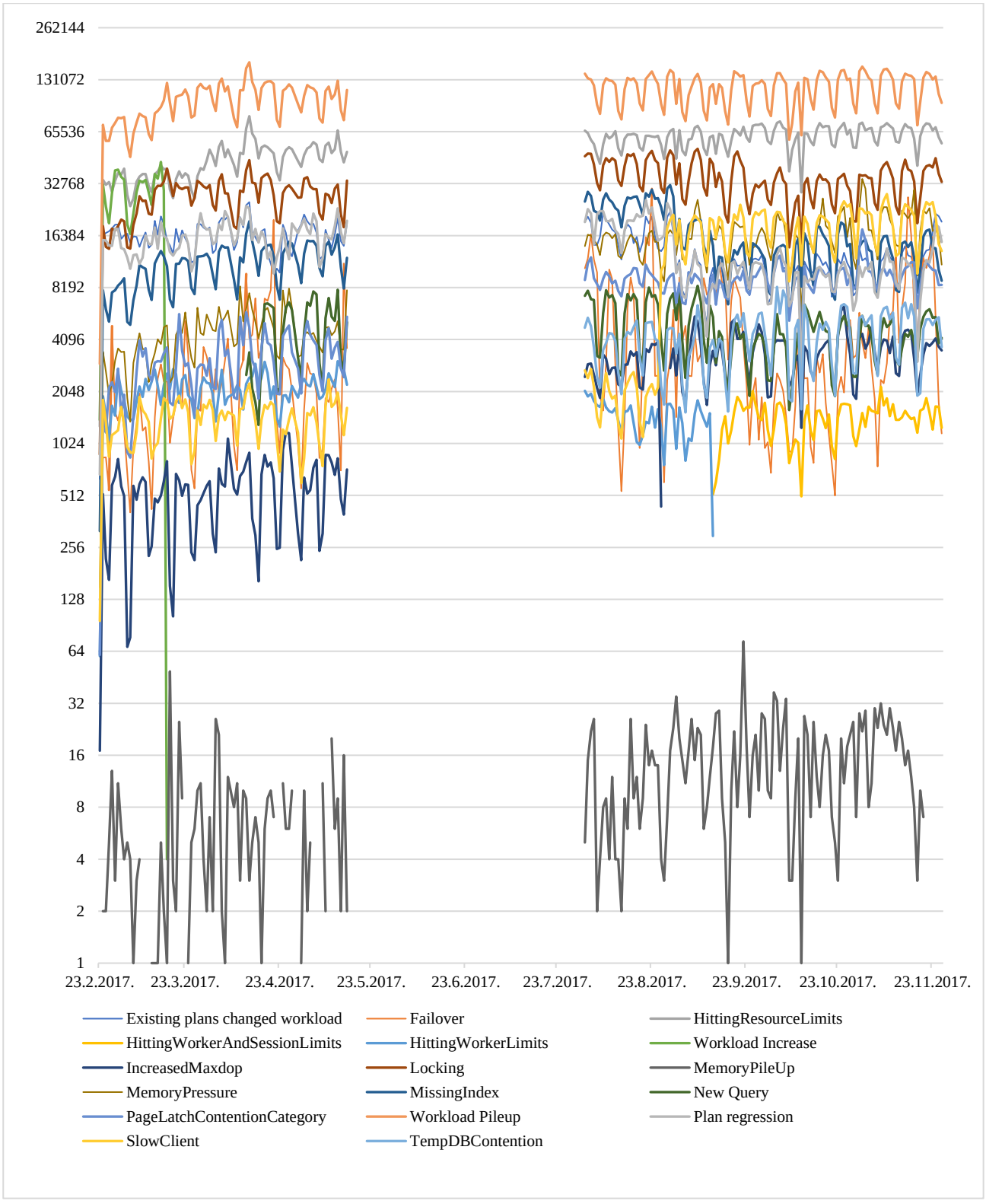
Imajući u vidu da se broj regiona i baza kontinuirano povećavao tokom vremena evaluacije, broj iteracija se posle nekog vremena ponovo ustalio na 1000 po satu. Kada se posmatra broj detekcija za svaki model po jednom satu, može se videti da model **QueryDuration** ima 8000 detektovanih intervala po satu. Takav trend se objašnjava činjenicom da se najveći broj problema kod korisnika javlja usled produženog vremena izvršavanja upita, što je prepoznato i u radu [25]. Drugi najzastupljeniji problemi u relacionim bazama podataka, sa oko 1000 detektovanih intervala u jedinici vremena, su detektovani od strane **Exception** modela. Podskup tih problema je prepoznat kao veoma važan i problematičan i u radu [39]. Ostali modeli su u proseku detektovali oko 200 intervala.

Korelacija između broja iteracija i detektovanih intervala ja takođe vidljiva u periodu od 9.5.2017. do 12.5.2017. Nakon detaljne analize, zaključeno je da je jedan od korisnika koji je imao veći broj relacionih baza podataka, zbog promene u dizajnu korisničke aplikacije, izazvao neoptimalno korišćenje ovog skupa relacionih baza podataka. Taj slučaj je, takođe, primećen od strane AID-TS sistema i korisnik je dobio odgovarajuće obaveštenje.



Slika 8.3. Broj problema detektovanih pojedinim generičkim modelima tokom perioda evaluacije

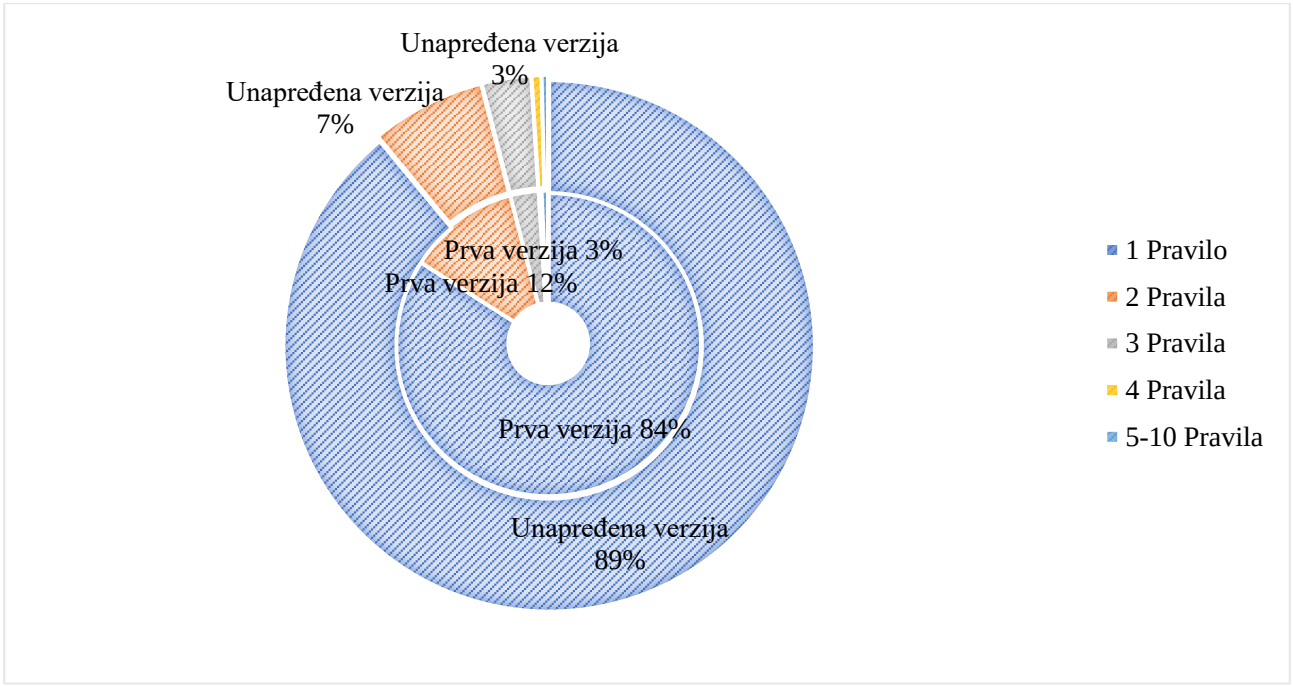
Nakon procesiranja intervala od strane generičkih modela, dalja obrada se vrši pomoću kategorizacionih modela. Na slici 8.4., je prikazan broj intervala detektovan tokom posmatranog perioda od strane kategorizacionih modela.



Slika 8.4. Broj problema detektovanih pojedinim kategorizacionim modelima tokom perioda evaluacije

Najviše detektovanih problema spada u grupu problema koji se često dešavaju u relacionim bazama podataka. To su kategorije **HittingResourceLimits**, **HittingWorkerLimits**, **WorkloadPileUp** i **Locking**. Iako su to generalni problemi koji se često dešavaju, oni mogu predstavljati značajan problem za prosečnog administratora relacionih baza podataka koji, uglavnom, nisu vični njihovom razumevanju i rešavanju. U ovom slučaju potrebno je dublje znanje o funkcionisanju SQL servera kako bi se tačno identifikovao problem. Mogućnost da se ti slučajevi automatski detektuju, a potom i automatski razumeju velika je prednost ovog sistema.

Poslednji mehanizam u sistemu predstavlja procesiranje intervala od strane ekspertskog sistema, sa svim dobijenim informacijama od strane generičkih i kategorizacionih modela. Na slici 8.5. prikazan je procenat intervala koji su detektovani odgovarajućim brojem pravila u periodu od 8 kalendarskih meseca. Taj period obuhvata približno 45 miliona procesiranih intervala.



Slika 8.5. Procenat intervala koji su detektovanih sa odgovarajućim brojem pravila

U ekspertskom sistemu, velika većina intervala u ranijem periodu evaluacije (84%) se mapira u tačno jedno pravilo. Nakon unapređenja taj broj je povećan za 5% i iznosi 89%. To znači da za 89% detektovanih slučajeva AID-TS sistem daje tačno objašnjenje uzroka. Narednih 10% slučajeva su prepoznati od strane dva (7%) i tri (3%) različita pravila. U tim slučajevima je ostavljeno korisniku da izabere koji će pravac izabrati prilikom daljeg istraživanja problema s obzirom da sistem to nije mogao sam da zaključi. U 1% slučajeva kao rezultat je dobijeno više od tri različita pravila.

Na osnovu dosadašnjeg iskustva detekcija sa više od tri pravila obično odgovara slučajevima koji traju duži vremenski period od 24 sata kada postoji ozbiljan problem sa relacionom bazom podataka koji izaziva i druge problematične manifestacije. Ove manifestacije obično nisu uzrok problema ali ih sistem validno otkriva pošto zaista postoje. Uzrok problema u ovom slučaju se može pronaći analizom geneze problema za konkretnu relacionu bazu podataka. Pod tim se podrazumeva

pronalaženje prvog intervala u kom je problem primećen od strane AID-TS sistema. U svim takvim slučajevima prvi interval koji je primećen od strane Sistema ne poseduje više od 3 različita pravila. Uglavnom, jedno ili najviše dva pravila opisuju konkretan problem, dok su ostali detektovani problemi i pravila posledične manifestacije konkretnog uzroka.

Može se zaključiti da je kvalitet pravila adekvatan, imajući u vidu da je 99% primećenih problema uhvaćeno sa tri ili manje od tri pravila. Takav rezultat ukazuje na visoku preciznost informacija koje AID-TS sistem isporučuje za krajnjeg korisnika Azure SQL klaud servisa.

8.3. Evaluacija korisničkog iskustva sa predstavljenim sistemom

Kroz komunikaciju sa korisnicima Azure SQL klaud servisa njihovo pozitivno iskustvo sa platformom i sistemom za detekciju je potvrđeno više puta. Ovde će biti prikazana dva konkretna primera putem kojih je opisana delotvornost sistema. Konkretni primeri se odnose na dva korisnika koji su bili u programu ranih korisnika kojima je omogućena upotreba ovog sistema, s obzirom da su imali veliki broj relacionih baza podataka u svom portfelju. Takođe, predstavljen je i opisan uticaj AID-TS sistema na sve korisnike koji su se nakon javne dostupnosti sistema registrovali za njegovo korišćenje.

8.3.1. Primer korisnika C#1

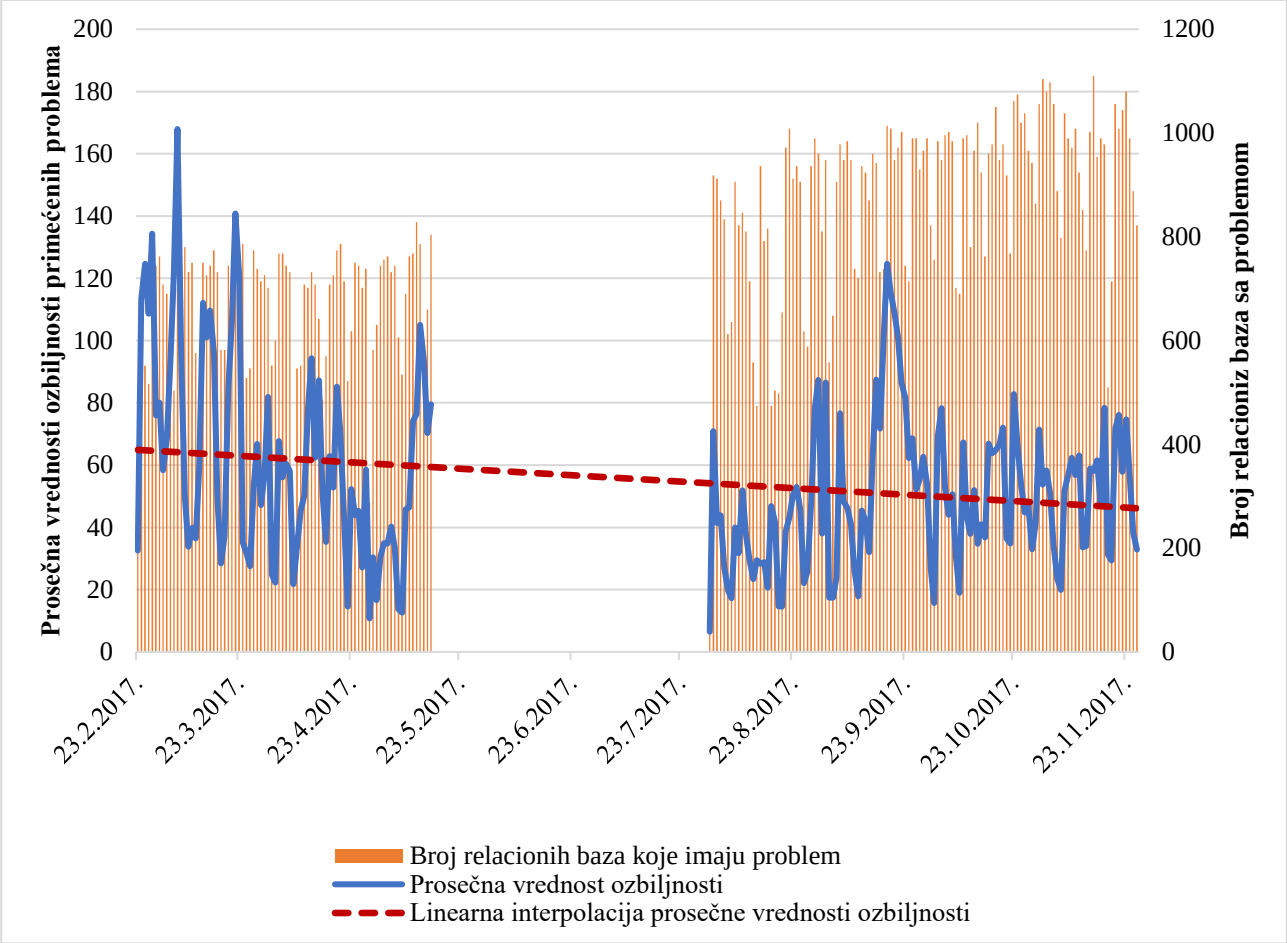
Tip upotrebe prvog korisnika (C#1) se može opisati na sledeći način. Aplikacija koristi više relacionih baza podataka, gde svaka relaciona baza podataka predstavlja jedinicu mere (eng. ***shard***). Sve relacione baze podataka imaju identičnu šemu, ali podaci u okviru jedne grupe relacionih baza podataka ne moraju pripadati istom krajnjem korisniku aplikacije. To implicira da ponašanje samih instanci relacionih baza podataka mnogo zavisi od različitih korisnika same aplikacije.

Na osnovu iskustva inženjera koji su razvili aplikaciju i celokupan servis oko nje, istraživanje i unapređenje takvog sistema može biti izuzetno kompleksno. U takvom okruženju AID-TS sistem otkrio je nekoliko problema i u samoj šemi relacione baza podataka ali i u samoj korisničkoj aplikaciji.

Sistem je o svim detektovanim problemima obavestio korisnika koji je, potom, preuzeo odgovarajuće korake da unapredi i aplikaciju i šemu relacione baze podataka. Nakon toga, uočene su značajno bolje performanse i poboljšano iskustvo krajnjih korisnika. Ovo nije primećeno samo od strane sistema već je dobijena i eksplicitna potvrda od strane korisnika da je AID-TS sistem pomogao u detektovanju i razumevanju problema.

Na slici 8.6. se može videti unapređenje tokom vremena. Takođe, može se primetiti da je broj relacionih baza podataka povećan, dok je prosečna vrednost parametra ozbiljnosti problema smanjena tokom vremena. To znači da, nakon što je korisnik počeo da koristi AID-TS sistem, u daljem periodu nisu se više pojavili značajniji problemi na većem broju relacionih baza podataka. Zahvaljujući ovom sistemu problemi koji su se javljali na manjem broju relacionih baza podataka su popravljeni relativno brzo, a ovaj sistem je omogućio i da se ti problemi ne raširе na ostatak

korisničkih relacionih baza podataka, što je, na kraju, doprinelo smanjenju uticaja na kompletnu korisničku platformu.



Slika 8.6. Unapređenje kod korisnika C#1 tokom perioda evaluacije

8.3.2. Primer korisnika C#2

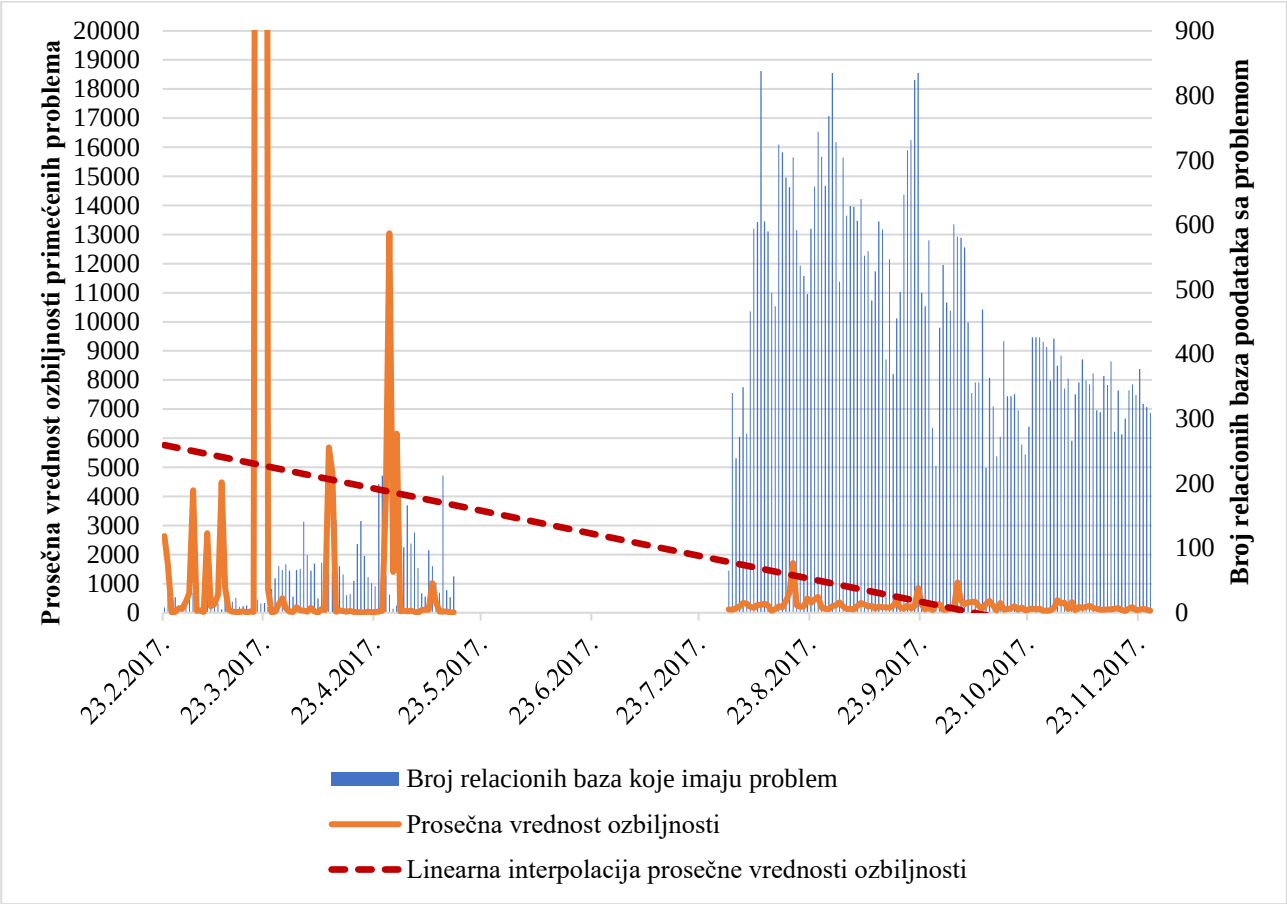
Slično korisniku C#1, drugi korisnik (C#2) je takođe imao arhitekturu sa više relacinoih baza podataka. Bitna razlika u odnosu na korisnika C#1 je u tome što korisnik C#2 nije imao više različitih korisnika svog servisa. Svi upiti koji su izvršavani su slični i pripadaju istom aplikativnom interfejsu.

AID-TS sistem je ukazao na to da je korisnik imao problem koji je detektovan kategorijom **TempDB Contention**. U ovom slučaju sistem je dao predlog, koji je između ostalog poznat i u praksi administratora baza podataka, o upotrebi funkcionalnosti SQL servera pod nazivom **in-memory temp database** [74].

Upotreba ove funkcionalnosti eliminiše potrebu za privremenom bazom podataka i omogućava upotrebu radne memorije čime se postiže veći protok informacija. Ukupno postignuto ubrzanje za korisnika je prikazano na slici 8.7. Takođe, na ovoj slici se može primetiti i značajan skok koji se

desio 23.3.2017. o kojem je korisnik obavešten automatski. Daljom analizom rezultata samog sistema zaključeno je da je nova verzija aplikativne logike imala grešku, koja je uticala na neoptimalnu upotrebu relacionih baza podataka.

U nastavku, nakon jula 2017. godine, broj relacionih baza podataka se značajno povećao. Ovakav rezultat je posledica korisnikovog proširenja kapaciteta. Može se takođe zaključiti da tokom proširenja nisu svi problemi rešeni pre ekspanzije korisničke platforme, pa je sistem i dalje nastavio da primećuje određene, manje ozbiljne probleme. U nastavku je korisnik dalje unapređivao upite i način upotrebe relacione baze podataka, nakon čega je dobijena potvrda da je to urađeno uz veliku pomoć AID-TS sistema.

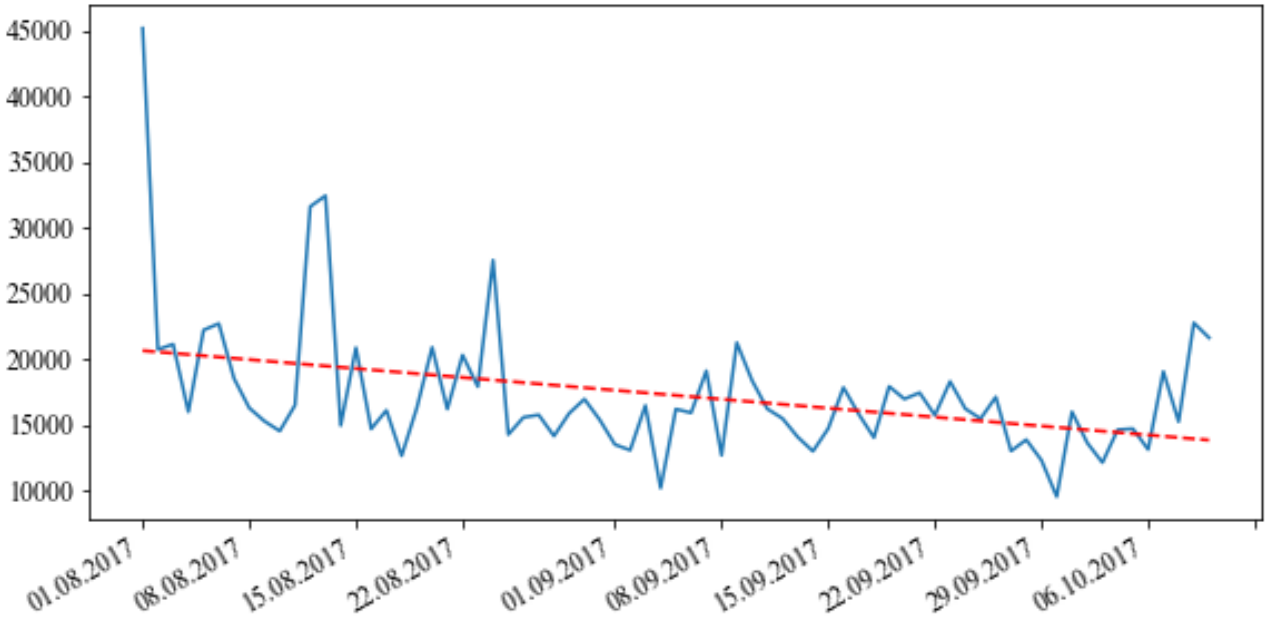


Slika 8.7. Unapređenje kod korisnika C#2 tokom perioda evaluacije

8.3.3. Ostali korisnici platforme Azure SQL koji koriste AID-TS sistem

Nakon što je AID-TS sistem predstavljen u javnosti, 2463 korisnika su se registrovala za njegovo korišćenje i izabrali da dobijaju informacije koje sistem generiše. Tom prilikom, sistem je pratio rad njihovih 740471 relacionih baza podataka. U skupu ovih korisnika, skoro polovina njih (1103) je imala problem na nekim od svojih relacionih baza podataka. Od strane sistema od svih posmatranih relacionih baza podataka, 48117 relacionih baza podataka je imalo neki problem koji je detektovan od strane sistema.

Na slici 8.8. možemo videti ukupnu prosečnu vrednost parametra ozbiljnosti sa linijom trenda (crvena isprekidana linija) koja pokazuje kontinuirano poboljšanje za korisnike koji su se registrovali za upotrebu ovog sistema.



Slika 8.8. Ukupno poboljšanje kod korisnika koji koriste AID-TS sistem

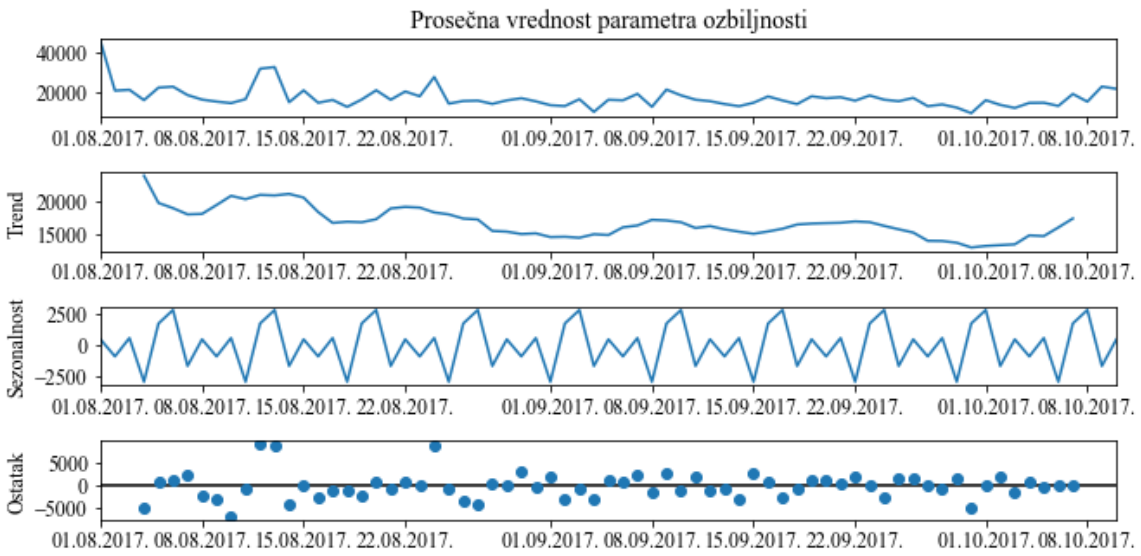
Na osnovu dostupnih podataka, vidljiv je pad prosečne vrednosti parametra ozbiljnosti do 11.10.2017., kada se desila ranije opisana regresija unutar SQL servera koja je, takođe, bila uhvaćena zahvaljujući praćenju rezultata ovog sistema. Zbog ove regresije, trendovi su analizirani do te tačke pošto ne postoji precizan način da se kvantifikuje efekat regresije na strukturu vremenske serije nakon tog trenutka.

Nad vremenskom serijom prosečnih vrednosti parametra ozbiljnosti problema izvršena je dekompozicija na tri komponente primenom aditivnog modela koji podrazumeva da za svaki trenutak u posmatranoj vremenskoj seriji vrednost parametra ozbiljnosti se može predstaviti kao suma tri različite komponente predstavljene formulom:

$$VrednostSignala(t) = Trend(t) + Periodičnost(t) + Ostatak(t)$$

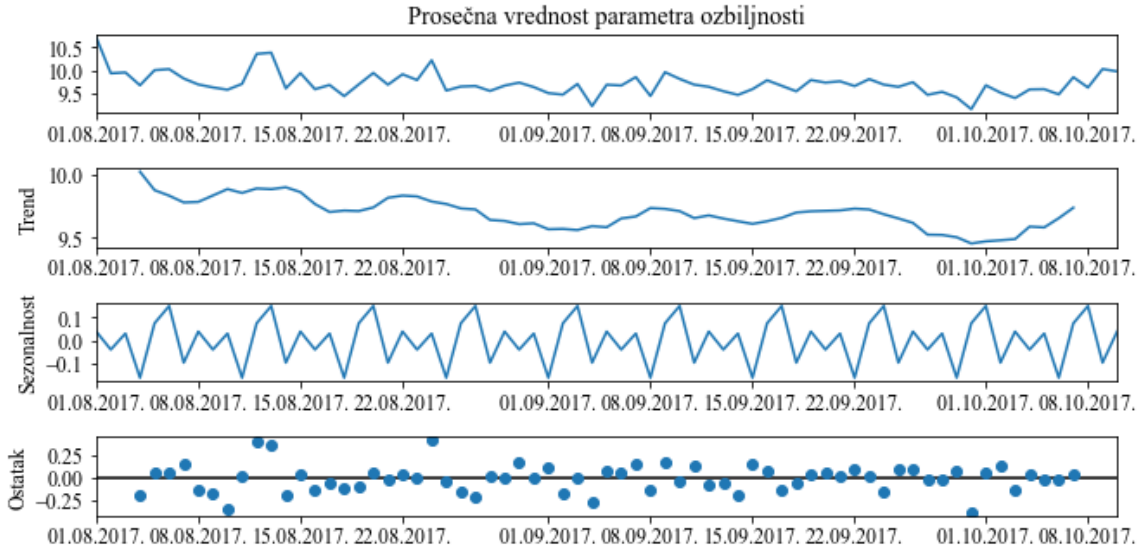
Na slici 8.9. predstavljene su uporedo dobijene komponente vremenske serije:

1. prosečna vrednost parametra ozbiljnosti
2. komponenta trenda (eng. ***trend***)
3. komponenta koja predstavlja periodičnost (eng. ***seasonal***)
4. komponenta koja predstavlja ostatak (eng. ***residual***)



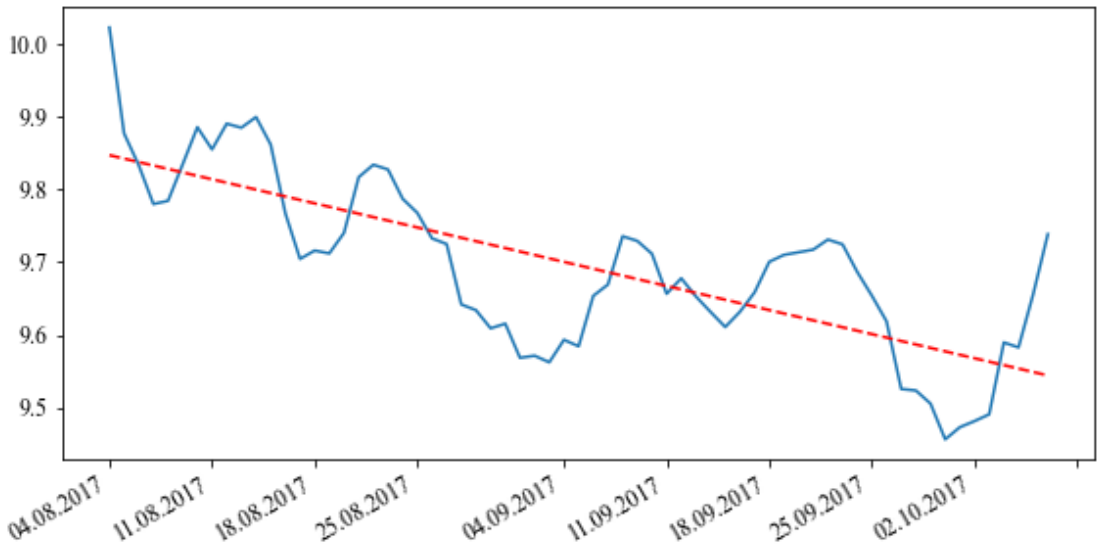
Slika 8.9. Vremenska serija prosečnih vrednosti parametra ozbiljnosti
i njena podela na tri komponente

Metodologija vremenske dekompozicije se primenjuje u situacijama kada je potrebno dokazati određene karakteristike posmatrane vremenske serije. U slučaju analize prosečne vrednosti parametra ozbiljnosti upotrebom ove metodologije treba dokazati da je ova vrednost opadajućeg karaktera. Kako bi se smanjila varijabilnost kroz vreme na vremensku seriju primenjena je logaritamska funkcija što je predloženo kao adekvatna metoda u ovakvim analizama u [75]. Na slici 8.10. je prikazana ponovna dekompozicija vremenske serije vrednosti logaritma za svaku prosečnu vrednost parametra ozbiljnosti.



Slika 8.10. Komponente vremenske serije prosečnih vrednosti parametra ozbiljnosti
obrađenih ***log*** funkcijom

Primenom linearne regresije nad komponentom trenda vremenske serije dobija se vrednost ***p*** koja iznosi $9{,}11199 \cdot 10^{-13}$. Time se odbacuje nulta hipoteza da je koeficijent nagiba linearne regresije jednak nuli. Na slici 8.11. crvenom isprekidanom linijom predstavljena je dobijena funkcija, čime se i vidi jasan opadajući trend.



Slika 8.11. Linearna regresija primenjena na komponentu trenda

Pošto je periodičnost na nivou sedam dana, a komponenta trenda je dokazano opadajuća, potrebno je utvrditi i da su vrednosti komponente ostatka zaista slučajno raspoređene i da nemaju uticaja na opadajuću prosečnu vrednost parametra ozbiljnosti. Ukoliko bi komponenta ostatka imala uticaj, to bi značilo da je zaključak o tome da je trend opadajući netačan i da se ne može sa statističkom sigurnošću reći da je došlo do opadanja vrednosti parametra ozbiljnosti.

Da bi se dokazalo da komponenta ostatka nema uticaj i da je u pitanju stohastički proces, primeniće se ARIMA model na vremensku seriju komponente ostatka čime se ona razdvaja na dve komponente i to komponentu statističkog procesa i komponentu belog šuma (eng. ***white noise***). Komponenta statističkog procesa se modeluje uz pomoć ARIMA modela, koji potiče od kombinacije dva modela, ***Auto Regressive*** (skr. AR) i ***Moving Average*** (skr. MA) modela [76].

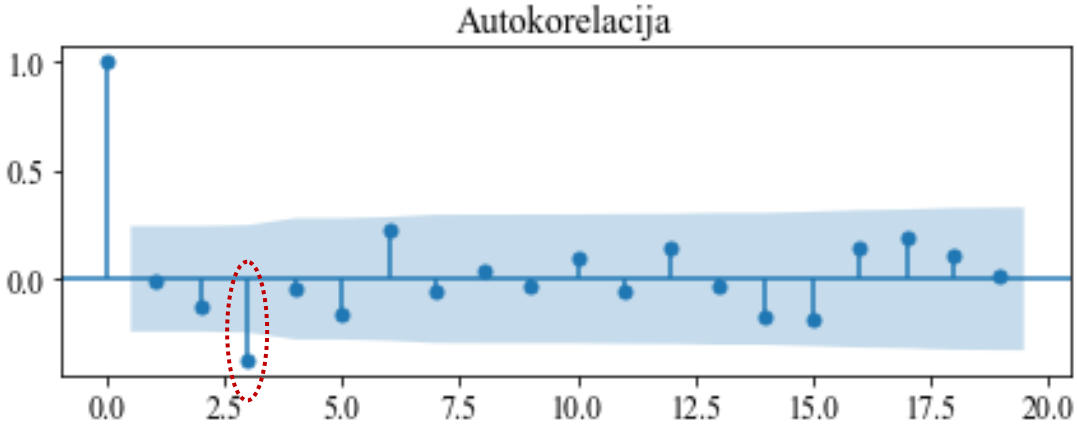
Kod AR modela jedna vrednost vremenske serije u nekom trenutku t_n zavisi od odgovarajuće vrednosti vremenske serije u nekom drugom trenutku koji je pomeren u odnosu na trenutak t_n za odgovarajući vremenski period (eng. ***lag***), dok kod MA modela, vrednost vremenske serije u trenutku t_n zavisi od svih prethodnih vrednosti vremenske serije.

ARIMA model se najčešće upotrebljava u situacijama kada je na osnovu postojeće vremenske serije potrebno predvideti buduće događaje i vrednosti signala. Takođe, ARIMA model je moguće primeniti i u obrnutim slučajevima kada je potrebno dokazati da je vremenska serija sačinjena od stohastičke komponente i komponente belog šuma.

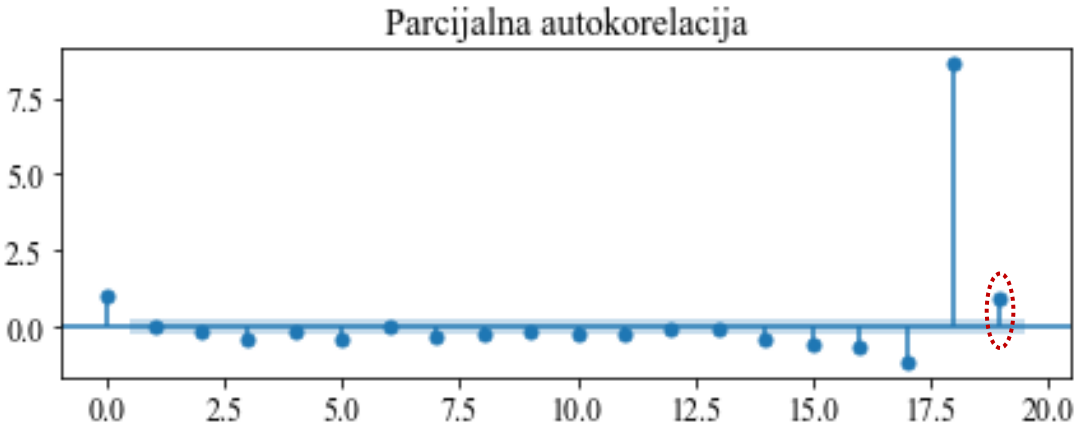
ARIMA model se konfiguriše uz pomoć ***p***, ***d*** i ***q*** parametra. Parametar ***p*** se odnosi na AR deo modela, ***q*** na MA deo modela, dok parametar ***d*** predstavlja broj operacija diferenciranja potreban za dobijanje stabilne vremenske serije. Imajući u vidu da je komponenta trenda već izbačena iz vremenske serije, smatra se da je vrednost parametra ***d*** jednaka 0 pošto preostale komponente predstavljaju stabilnu vremensku seriju. Da bi se dobile vrednosti parametra ***p*** i ***q***, potrebno je nad komponentom ostatka izračunati stepen autokorelacije i parcijalne autokorelacije, respektivno.

Autokorelacija je metoda koja su primenjuju na signale kako bi se utvrdilo postojanje repetativnih obrazaca, pa se njenom primenom dobija vrednost parametra ***p***. Parcijalna autokorelacija sa druge strane daje stepen korelacije između vrednosti signala i svih vrednosti tog signala u prethodnim trenucima, pa se njenom primenom dobija vrednost parametra ***q***.

Dobijeni grafici su prikazani na slici 8.12. i slici 8.13. Zaključuje se da su parametri ARIMA modela *q = 3* (maksimalna vrednost *X* ose za koju je vrednost auto korelacije statistički značajana) i *p = 19* (maksimalna vrednost *X* ose za koju je vrednost parcijalne auto korelacije statistički značajna). Ovi parametri se koriste za treniranje ARIMA modela nad komponentom ostatka.

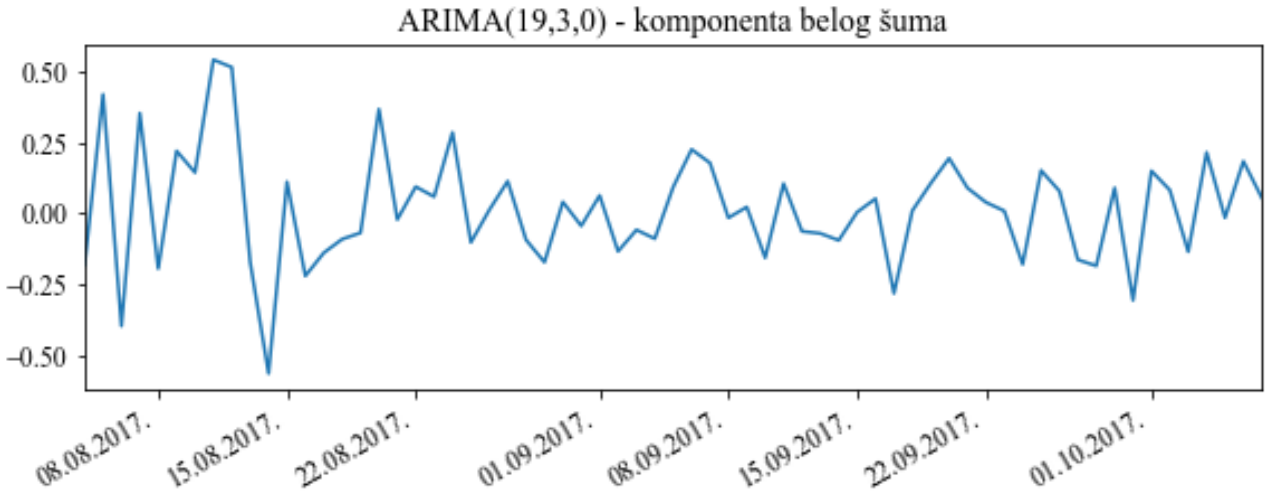


Slika 8.12. Autokorelacija posmatrane vremenske serije parametra ozbiljnosti



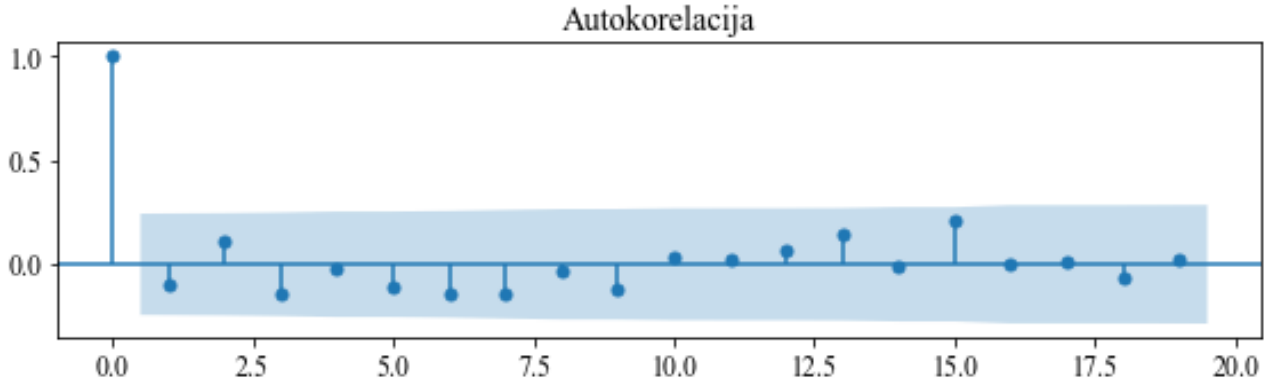
Slika 8.13. Parcijalna autokorelacija posmatrane vremenske serije parametra ozbiljnosti

Treniranjem ARIMA modela nad komponentom ostatka vremenske serije, dobija se vremenska serija koja sadrži komponente ARIME i belog šuma. Komponenta belog šuma prikazana je na slici 8.14., za koju se ponovo računa autokorelacija i parcijalna autokorelacija da bi se ustanovilo da je zaista reč o šumu i da se ni jedna vrednost vremenske serije belog šuma ni na koji način ne može korelisati sa vrednošću istog signala u nekom drugom trenutku. Ukoliko se to pokaže, takav rezultat potvrđuje validnost ARIMA modela koji je upotrebljen u analizi.

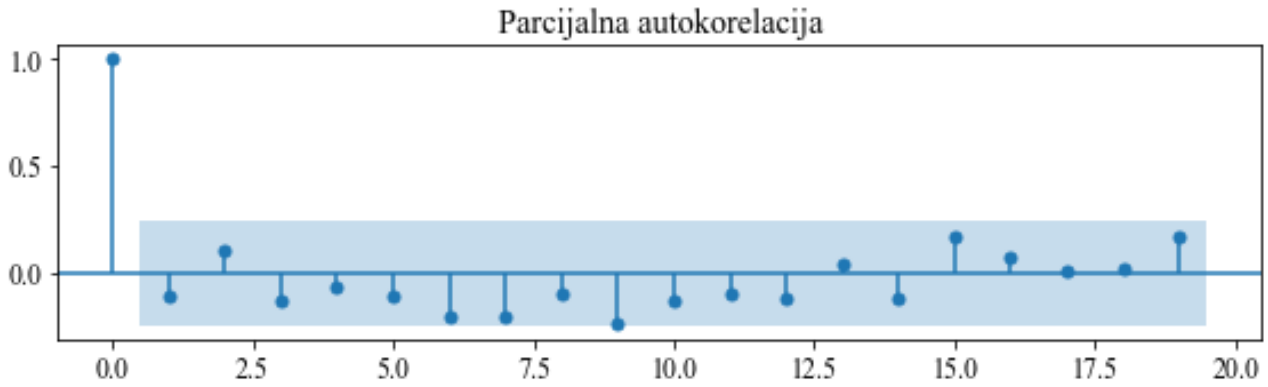


Slika 8.14. Komponenta šuma ARIMA(19,3,0) modela

Na osnovu dobijenih grafikona na slikama 8.15. i 8.16., zaključuje se da parcijalna autokorelacija i autokorelacija ne postoje sa statističkom sigurnošću od 95%, što znači da su sve vrednosti počevši od prvog posmatranog vremenskog perioda unutar intervala pouzdanosti (eng. ***confidence intervals***), čime se i dokazuje da komponenta šuma zaista predstavlja šum i da je ARIMA model validno upotrebljen za dokazivanje da je komponenta ostatka vremenske serije zaista sačinjena od stohastičke komponente i komponente belog šuma.



Slika 8.15. Autokorelacija komponente šuma ARIMA modela

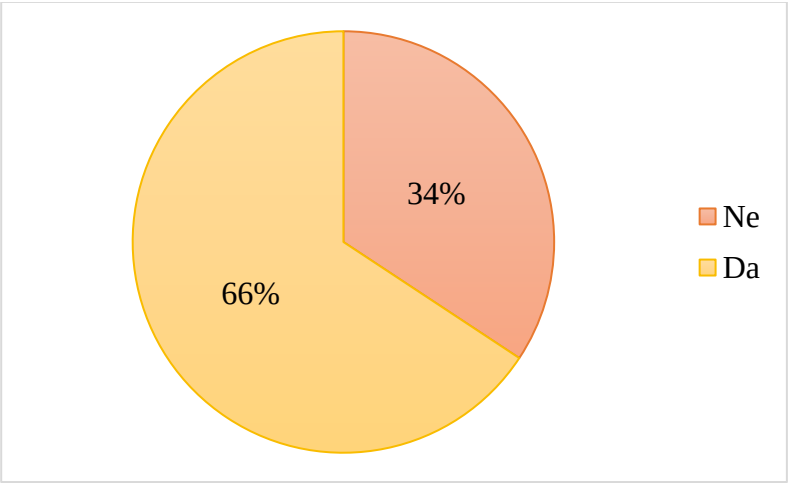


Slika 8.16. Parcijalna autokorelacija komponente šuma ARIMA modela

8.4. Evaluacija sistema kroz specifične slučajeve od strane inženjera sistemske podrške

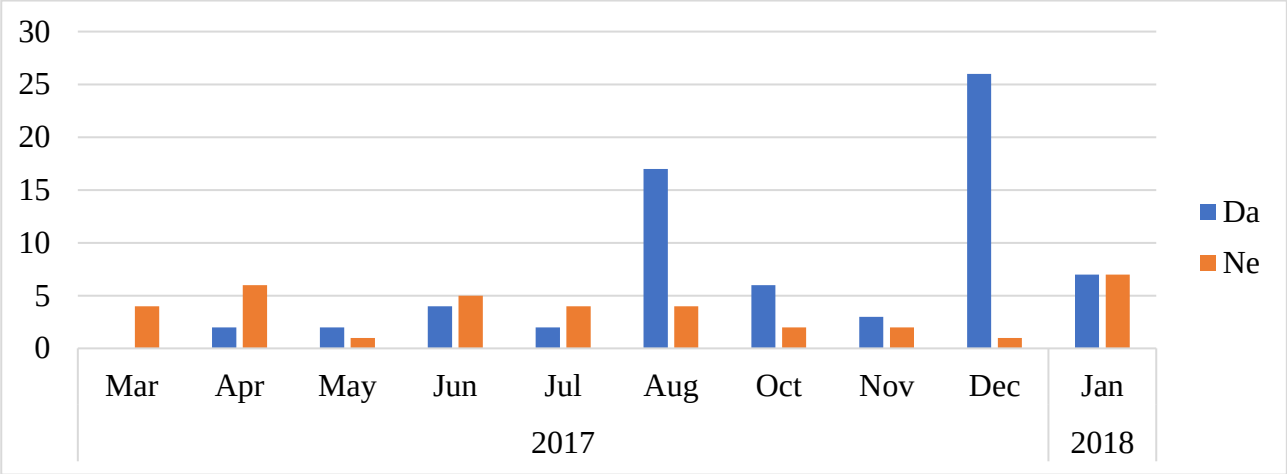
Tokom razvoja celokupnog AID-TS sistema, on je korišćen i za interne potrebe. Tom prilikom je vršeno i anketiranje inženjera koji pripadaju sistemskoj podršci kompanije Microsoft na temu primenljivosti samog sistema za potrebe otkrivanja i razumevanja problema koji su prijavljivani sistemskoj podršci.

U periodu od aprila 2017. do januara 2018. godine, ukupno je bilo 105 prijavljenih slučajeva koji su imali neki problem vezan za jednu ili više Azure SQL relacionih baza podataka. Kao što se može videti sa slike 8.17., u dve trećine slučajeva AID-TS sistem je dao odgovor koji je pomogao u istraživanju problema koji su se tada dešavali sa relacionom bazom podataka.



Slika 8.17. Procenat uspešne detekcije i razumevanja u slučajevima prijavljenim sistemskoj podršci

U preostalih 34% slučajeva, sistem nije detektovao ni jedan problem. Od tih 36 slučajeva, problem koji je postojao nije uspešno detektovan u 21 slučaju. Kao što se vidi sa slike 8.18., do avgusta meseca 2017. godine, većina slučajeva nisu uspešno detektovani, zato što je u to vreme korišćena starija verzija rešenja koja nije imala sva unapređenja. Pomenuti slučajevi, su između ostalog, bili korišćeni za dalje unapređenje sistema, gde se počevši od avgusta meseca, kada je unapređeni sistem pušten u rad, na istoj slici može videti skok u broju slučajeva u kojima je sistem uspešno primećivao probleme i davao tačno objašnjenje za njih.



Slika 8.18. Raspodela uspešnosti detekcije problema kroz posmatrani period

U preostalih 15 slučajeva, u kojima sistem nije detektovao probleme, ispostavilo se da problemi koje je korisnik prijavio nisu bili vezani za relacionu bazu podataka koju su koristili, već se uzrok problema nalazio u njihovim sistemima i aplikacijama. U tim slučajevima, iako AID-TS sistem nije dao direktan odgovor šta je korisnikov problem, ipak je mnogo pomogao inženjerima sistemske podrške. Odsustvo dobijenog rezultata od strane AID-TS sistema u tim slučajevima ukazuje da ne postoji problem i da relaciona baza podataka normalno funkcioniše. Takva informacija daje dodatnu pomoć inženjerima sistemske podrške koji u komunikaciji sa korisnicima mogu da potvrde da prijavljen problem nije vezan za Azure SQL platformu.

Tokom posmatranog perioda od interesantnih pojedinačnih slučajeva mogu se izdvojiti četiri slučaja koji predstavljaju reprezentativane primere i opisuju pomoć koju korisnici mogu dobiti zahvaljujući AID-TS sistemu.

8.4.1. Problem u softverskoj biblioteci

Prvi scenario je izuzetno sličan scenariju koji je opisan u radu [39], a tiče se problema koji nastaju zbog nekoordinisanih izmena šeme relacione baze podataka. U navedenom radu, problem je nastao kod IoT uređaja gde je promena u šemi relacione baze podataka dovela do toga da uređaji postanu neoperativni. U sličnom slučaju koji je naknadano prijavljen, problem se manifestovao na sličan

način, detektovan je generičkim modelom **Exception**, a potom klasifikovan od strane ekspertskog sistema.

Otkriveno je da se desila greška sa kodom „**201**“ koja označava da specifični parametar nije podešen na odgovarajući način. U ovoj situaciji je uzrok bio nešto drugačiji. Problem je bio vezan za biblioteku koju je korisnik relacione baze podataka Azure SQL nudio korisnicima svog servisa. Naime, biblioteka omogućava da korisnici pošalju prazan string kao parametar, što je u suštini bio nedostatak provere ulaznih parametara i u samoj biblioteci, ali i u relacionoj bazi podataka.

Iako AID-TS sistem nije bio u mogućnosti da detektuje ceo splet okolnosti, ipak je usmerio inženjere podrške koji su zajedno sa korisnikom vrlo brzo došli do potpunog razumevanja problema. Korisnik je napravio novu verziju biblioteke i ažurirao je šemu relacione baze podataka, koja sada u SQL proceduri proverava i validnost pomenutog parametra.

8.4.2. Problem sa neusklađenom šemom relacione baze podataka

U drugom scenariju je, takođe, došlo do problema sa greškama koje se odnose na neusklađenost šeme relacione baze podatka ili neusklađenost upita koju aplikacija koristi. U ovom slučaju nije bilo moguće pronaći odgovarajuću tabelu i kolone u tabeli, unutar relacione baze podataka. U tabeli 8.1. su prikazane greške koje su se tom prilikom desile.

Tabela 8.1. Prijavljene greške

Poruka greške	Broj grešaka
Invalid object name '#####_######.##_####_####view'.	162
Invalid object name '#####_######.whitelisted_subscriptions'.	12
Invalid column name 'cab_id'.	8

Interesantna je činjenica da je relaciona baza podataka koja je imala problem vlasništvo tima u okviru kompanije Microsoft. Nakon primanja informacije o greškama tim je analizom pređašnih dešavanja došao do zaključka da je izvršena promena šeme relacione baze podataka. Promena je dovela do neusklađenosti aplikacije i relacione baze podataka.

Zahvaljujući ovom sistemu, taj propust je otkriven 15 minuta nakon što je promena relacione baze podataka izvršena, a pronalaženje i rešavanje problema je rešeno u roku od 70 minuta. Uticaj ove situacije na krajnje korisnike je bio minimalan, opet zahvaljujući AID-TS sistemu koji je uspeo da detektuje problem vrlo brzo nakon njegovog nastajanja.

8.4.3. Problem sa performansama usled povećane upotrebe relacione baze podataka

Inženjerima sistemske podrške je prijavljen slučaj povećane upotrebe i otežanog rada jedne od relacionih baza podataka. Sistemska podrška je, nakon dobijanja svih potrebnih informacija za dalje istraživanje slučaja, prvenstveno konsultovala rezultate dobijene od strane AID-TS sistema. Treba

imati na umu da u ovom konkretnom slučaju ekspertski sistem još uvek nije bio u potpunosti razvijen i da nije imao mnogo pravila. Zbog toga, dobijeni rezultati su zahtevali dodatno zaključivanje o tome šta je uzrok, a šta posledica.

Analiza ovog slučaja kasnije je pomogla u razvijanju samog ekspertskog sistema. Sistem je tom prilikom detektovao usporenje izvršavanja upita koji su povezani sa dva uzroka problema. Povezani uzroci su bili identifikovani kategorijama **WorkloadPileUp** i **PlanRegression**.

Problem nagomilavanja zahteva se dešavao najčešće. Konkretan upit je detektovan kao upit koji pri izvršavanju dovodi do usporavanja celokupne relacione baze podataka.

Problem izbora neoptimalnog plana za izvršavanje upita je primećen za tri različita upita koji su bili izvršavani sa lošijim planovima. Sistem je detektovao sledeće:

- *„QueryID(19227028) has good Plan ID(9349321) but it was executed using bad Plan ID(9359824)"*
- *„QueryID(18566711) has good Plan ID(9044963) but it was executed using bad Plan ID(9046080)"*
- *„QueryID(19129478) has good Plan ID(9287436) but it was executed using bad Plan ID(9290179)"*

Potom je sistemska podrška predložila korisniku relacione baze podataka sledeća rešenja, uređena po prioritetu:

- analiza problematičnog upita sa izmenom strukture ili logike kako bi se postigle bolje performanse,
- forsiranje odgovarajućeg dobrog plana za tri upita koji su imali problem.

Krajnji zaključak je ukazao da je najveći problem koji je imao najveći uticaj na korisnika proizilazio od nagomilavanja problematičnog upita. Nakon unapređenja upita drugi problemi nisu bili relevantni za performanse relacione baze podataka. U ovom slučaju ekspertski sistem, koji je kasnije razvijen, automatski pomaže u razlikovanju i odredjivanju prioriteta različitih problema, što je u ovom slučaju urađeno ručno od strane inženjera sistemske podrške.

Korisniku koji je prijavio problem je, zatim, data mogućnost da se uključi i u program ranih korisnika koji mogu da isprobavaju razvijani sistem pre nego što je javno dostupan svim ostalim korisnicima. Nakon više mesečne upotrebe, zaključili su da upotreba AID-TS sistema detektuje probleme i pomaže im u njihovom rešavanju. Korisnik je potvrdio da je inženjerima u okviru korisnikove kompanije prethodno trebalo od jednog do 6 meseci da otkriju šta je problem i kako da ga poprave. Takođe, pre upotrebe sistema, korisnikov inženjerski tim uglavnom nije znao odakle da krene prilikom istraživanja problema.

Nakon upotrebe sistema, korisnik je potvrdio da se period koji im je potreban da reše problem smanjio sa 6 meseci na 24 sata. Takođe, korisnik je nastavio da upotrebljava sistem ne samo za rešavanje najvećih problema, već i za dalju optimizaciju relacione baze podataka. S obzirom da je korisnik posedovao više relacionih baza podataka, AID-TS sistem je koristio i kao pomoć za praćenje razvoja relacione baze podataka i aplikacije koja koristi relacionu bazu podataka.

Korisnik je ažurirao svoje relacione baze podataka određenim ritmom, uglavnom sa jednim danom razmaka između dve relacione baze podataka. Zatim, u sklopu validacije promena, oslanjao se na AID-TS sistem, kako bi utvrdio da li postoji problem u relacionoj baza podataka sa ažuriranom verzijom aplikacije i šeme podataka. Korisnik je takođe zaključio da je ukupna cena korišćenja sistema za detekciju bila značajno jeftinija u odnosu na troškove koje je korisnik imao usled prethodne neoptimalne upotrebe relacione baze podataka.

8.4.4. Problem sa novokreiranim indeksom

Korisnik je prijavio nemogućnost procesiranja aplikativnih upita od strane relacione baze podataka Azure SQL. Problem se pojavljivao nasumično. Jedini način kako je korisnik mogao da privremeno reši problem sa svoje strane je bilo restartovanje aplikacije koja je slala upite. Nakon trećeg ponavlja korisnik je zatražio pomoć od inženjera sistemske podrške.

Inženjeri sistemske podrške su analizirali izveštaj AID-TS sistema koji je detektovao sledeće:

- *„QueryID(1*688793*) has good Plan ID(9*41278*) but it was executed using bad Plan ID(9*46167*)”*
- *Memory Pressure*

Ispostavilo se da je došlo do promene plana koji je doveo do produženog izvršavanja upita i povećane upotrebe memorije. Kao posledica nedostatka slobodne memorije, drugi upiti koji su izvršavani od strane aplikacije nisu mogli da dobiju deo memorije koji su dobijali ranije, tako da je došlo do celokupnog preopterećenja relacione baze podataka. S obzirom da je izvršavanje upita trajalo ekstremno dugo, restartovanje aplikacije je pomagalo zato što bi se tom prilikom prekinula konekcija sa klijentske strane pa bi se i samo izvršavanje upita na serverskoj strani prekinulo. Prekidanjem izvršavanja upita i zauzeta memorija je bila oslobođena pa su ostali upiti mogli nesmetano da je koriste.

S obzirom da je promena plana uzrok, a povećana upotreba memorije posledica, dalji fokus istrage je usmeren na razumevanje zašto je došlo do promene plana. Ispostavilo se da je kreiranje novog indeksa od strane korisnika nad jednom od korišćenih tabela nekoliko dana pre nego što se problem desio u problematičnom upitu izazvao izbor pogrešnog lošijeg plana za izvršavanje upita. SQL server sadrži mehanizme odbrane [32] u tom slučaju, ali ga je korisnik eksplicitno isključio. Nakon brisanja problematičnog indeksa i ponovnog testiranja upita, SQL server je ponovo izabrao adekvatan plan.

Nakon ovog slučaja razmatrano je uvođenje novog modela kategorizacije koji bi umeo da detektuje kreiranje novog indeksa i potencijalnih negativnih efekata na ostale operacije unutar relacione baze podataka.

9. Zaključak

Konstanta evolucija klaud okruženja i zahtevi globalnog tržišta značajno utiču na tok razvoja infrastrukture i korisničkog iskustva sa klaud platformama. Moderne aplikacije koje su već orijentisane ka klaud okruženju i svetu servisa se u samom projektovanju oslanjaju na neke od prednosti klaud okruženja. Sa druge strane, korisnici iz tradicionalnih privatnih okruženja su motivisani da migriraju u klaud okruženje zbog jeftinije infrastrukture, a i zbog mnogo lakšeg održavanja. Sa druge strane pružaoci usluga imaju za cilj da obezbede bolje performanse, optimalniju upotrebu hardverskih resursa i omogućavanje tehnologija kao što je, na primer, geografska replikacija podataka.

Relacione baze podataka predstavljaju tehnologiju koja na efikasan način omogućava čuvanje, pretraživanje i dohvatanje podataka. U klaud servisnom okruženju relacione baze podataka su apstrahovane na nivo servisa koje korisnik može da zakupi. Administratori relacionih baza podataka nemaju pristup virtuelnim mašinama i instalaciji relacione baze podataka u takvom okruženju. Dodatna dimenzija koja povećava složenost odnosi se na broj relacionih baza podataka koje administrator mora da prati i podešava, a taj broj se u klaud izražava u hiljadama i stotinama hiljada. Zbog same količine relacionih baza podataka javlja se potreba za sistemom koji omogućava jednostavno praćenje rada velikog broja relacionih baza podataka. Pored toga, ukoliko dođe do problema, takav sistem treba da omogući brzo detektovanje i razumevanje problema. Prateći moderne trendove, ovaj rad ima za cilj da iskoristi obilje podataka o ponašanju klaud servisa za razumevanje različitih načina upotrebe relacione baze podataka Azure SQL. Unapređenje samog posmatranog servisa i poboljšanje korisničkog zadovoljstvo samom platformom je primarni razlog istraživanja u ovoj oblasti.

Sistem za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju je projektovan da automatski detektuje probleme na relacionoj bazi podataka u klaud okruženju i da dostavi korisniku jasnu informaciju o tome koja relaciona baza podataka ima problem i da odredi

njegov uzrok. Sve to ima za cilj da se smanji vreme tokom kojeg relaciona baza podataka ima problem, a time se postiže i smanjenje troškova korisnika.

Sistem je implementiran upotrebom postojećih servisa u okviru Azure ekosistema kompanije Microsoft. Nakon detaljne analize podataka prikupljenih u klaud servisu definisane su dve vrste statističkih modela: generički i kategorizacioni modeli. Generički modeli imaju za cilj da prepoznaju što više potencijalnih problema pa je naglasak na što većoj obuhvatnosti. Njihova preciznost nije velika, ali spektar različitih problema koje umeju da prepoznaju je značajan. Sa druge strane, kategorizacioni modeli se odlikuju izuzetnom preciznošću i njihov je zadatak da klasifikuju i prepoznaju ono što su generički modeli primetili kao problem.

Razvojem sistema pojavila se i potreba za dodatnim analiziranjem rezultata koji su dobijeni od statističkih modela. Zbog što preciznijih i tačnijih analiza problema koji su primećeni, projektovan je ekspertski sistem baziran na pravilima. U celokupnom sistemu, ekspertski sistem je na kraju zadužen da predstavi rezultat u formi razumevanja problema za svaku relacionu bazu podataka.

Rezultati evaluacije nad vrlo velikim brojem baza podataka u skoro polugodišnjem periodu pokazuju da je vreme procesiranja ponuđenog rešenja tokom razvoja smanjeno oko dva puta, dok je broj intervala koje je potrebno procesirati porastao isto toliko, što kumulativno daje ubrzanje od oko 4 puta. U otvorenoj literaturi, kada se posmatra veličina servisa koji broji više miliona relacionih baza podataka, nije pronađeno ovako brzo vreme procesiranja u poređenju sa postojećim rešenjima u industriji i u akademiji.

Glavni postignuti doprinosi ogledaju se u sofisticiranim statističkim modelima koji u kombinaciji sa bogatim ekspertskim sistemom definisanim na bazi pravila pruža razumevanje problema. Ključna prednost ovakvog jednog sistema proizilazi iz činjenice da je vreme potrebno za istraživanje i razumevanje problema sada značajno manje. U isto vreme, ovaj sistem jednostavno može da skalira i da tom prilikom vodi računa o radu miliona različitih relacionih baza podataka.

Kroz evaluaciju sistema prikazano je više konkretnih primera u kojima je definisani sistem pomogao samim korisnicima i inženjerima sistemske podrške. Pritom je više stotina relacionih baza podataka nakon upotrebe sistema unapređeno, a zatim i kontinuirano optimizovano. Sistem je, takođe, korišćen od strane inženjera sistemske podrške. Analizom zadovoljstva inženjera sistemske podrške ponuđenim rezultatima, implementirani su dodatni modeli i pravila u ekspertskom sistemu. U 20% slučajeva otkriveni su nedostajući modeli i pravila. Sistem je pravilno prepoznao ostalih 80% slučajeva od kojih se ispostavilo da 14% slučajeva nije imalo ni jedan problem sa relacionom bazom podataka već sa samom korisničkom aplikacijom. Informacija i saznanje da je sa platformom sve u redu je od značajne vrednosti inženjerima sistemske podrške s obzirom da im omogućuje da u komunikaciji sa korisnicima argumentovano dokažu korisnicima da problem prijavljen sa njihove strane nije vezan za Azure SQL platformu, već da se uzrok problema nalazi na strani korisničkog informacionog sistema.

Nakon isporučivanja sistema svim korisnicima platforme, za 33.6% relacionih baza podataka u celom Azure SQL okruženju eksplicitno je zatraženo uključivanje sistema za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju. Ako se posmatraju samo relacione baze podataka za koje je sistem bio konfigurisan da isporučuje rezultate, za 6.49% relacionih baza podataka je sistem prepoznao neki od problema, dok ostale relacione baze podataka nisu imale ni jedan problem.

Nakon realizacije sistema otvaraju se i sagledavaju mogućnosti daljeg unapređivanja sistema i pravci budućeg istraživanja u ovoj oblasti. Iako su neki delovi rešenja paralelizovani, određeni segmenti kao što je to, na primer, izvršavanje kategorizacionih modela za različite intervale koji se tiču iste relacione baze podataka se izvršavaju sekvencijalno. Paralelizacija obrade različitih intervala je moguća i smanjila bi vreme detekcije i obrade intervala.

Kada se govori o telemetriji i transferu podataka od izvora do skladišta podataka, prenos podataka i naknadno procesiranje informacija povećava vreme detekcije. U idealnim okolnostima, vreme detekcije problema bi trebalo meriti u sekundama, a ne u minutima. Potencijalni pristup daljeg unapređivanja se može usmeriti u pravcu reorganizacije arhitekture sistema. U tom slučaju, generički modeli bi mogli da se izvršavaju unutar samih relacionih baza podataka. Time bi se skratilo vreme koje se čeka na propagiranje osnovnih signala do mlakog skladišta podataka.

Ortogonalan pristup koji se, takođe, smatra pogodnim za istraživanje u budućnosti je definisanje brzih modela koji bi ukazali na činjenicu da problem postoji, bez detaljnog objašnjenja problema. U tom slučaju, i dalje bi se upotrebljavali postojeći modeli za dobijanje detaljnog uzroka problema, ali vreme do podizanja mehanizma uzbune bi bilo značajno kraće.

Takođe, potencijalno dalje istraživanje se može usmeriti na situacije u kojima ekspertski sistem ne sadrži odgovarajuće pravilo i obeleži interval kao „***Unknown***“. Istina, takvi slučajevi se ne dešavaju često, ali svakako predstavljaju prostor za poboljšanje. Slučajevi za koje razumevanje problema nedostaje, a koji su prijavljeni od strane korisnika, zahtevaju novu definiciju potpisa upotrebe i projektovanje i implementaciju odgovarajućeg kategorizacionog modela i pravila u ekspertskom sistemu. Jedan od takvih primera je opisan u radu gde bi posedovanje modela koji detektuje novokreirani indeks pojednostavilo istragu koja je tada obavljena od strane inženjera sistemske podrške. Oblast koja isto tako može biti interesantna za dalju analizu je upotreba tehnika klasterisanja nad izlazom ekspertskog sistema, slično metodologiji predstavljenoj u radu [77].

Iz korisničkog, a i sistemskog ugla, korisno bi bilo osmisliti proširenje sistema koje bi omogućilo korisnicima da daju povratnu informaciju o kvalitetu dobijenih rezultata od strane sistema. Posedovanje informacija o kvalitetu pruženih informacija iz perspektive korisnika je vrlo važan aspekt. Daljim istraživanjem u tom pravcu i upotrebom ovih informacija bio bi definisan izvor koji bi mogao da ponudi odgovor na pitanje koliko je ceo sistem kvalitetan i koje je njegove oblasti potrebno dalje unapređivati.

Pristup iz ovog rada se takođe može primeniti i na druge naučne oblasti, kao što su, na primer, oblasti računarskih mreža, vazduhoplovstva ili medicine. U slučajevima kada je potrebno analizirati veliki broj različitih signala, a da je pritom skup poznatih slučajeva i podataka izuzetno mali, tehnike mašinskog učenja su se pokazale kao ograničavajuće. Slični pristupi prikazani u otvorenoj literaturi kao i pristup definisan u ovom radu se smatraju odgovarajućim rešenjem za ove složene slučajeve. Takođe, u domenu tehnika mašinskog učenja, sinergistički pristup upotrebe ekspertskih sistema zajedno sa statističkim modelima može da pruži kompletno rešenje koje vodi računa kako o međusobnoj povezanosti između različitih rezultata tako i o etičkim pitanjima.

10. Reference

[1] M. Arregoces, Data Center Fundamentals, Indianapolis: Cisco Press, 2004.

[2] N. Antonopoulos i L. Gillam, Cloud Computing; Principles, Systems and Applications, London: Springer, 2010.

[3] G. Shroff, „Cloud computing economics,“ u *Enterprise Cloud Computing: Technology, Architecture, Applications*, Cambridge, Cambridge University Press, 2010, pp. 64-67.

[4] K. Jamsa, Cloud computing - SaaS, PaaS, IaaS, Virtualization, Business Models, Mobile, Security, and More, Burlington: Jones & Bartlett Learning, 2013.

[5] IBM, „What is Containers as a service (CaaS)?,“ IBM, 1 5 2017. [Na mreži]. Available: https://www.ibm.com/services/cloud/containers-as-a-service. [Poslednji pristup 1 4 2021].

[6] Alphabet inc, „Gmail,“ Google, 1 4 2004. [Na mreži]. Available: https://mail.google.com/. [Poslednji pristup 1 4 2021].

[7] Amazon Web Service, „Release: AWS Lambda on 2014-11-13,“ Amazon, 13 11 2014. [Na mreži]. Available: https://aws.amazon.com/releasenotes/release-aws-lambda-on-2014-11-13/. [Poslednji pristup 1 4 2021].

[8] Microsoft, „Azure Functions,“ Microsoft, 1 1 2020. [Na mreži]. Available: https://azure.microsoft.com/en-us/services/functions/. [Poslednji pristup 10 4 2020].

[9] W. Lang, F. Bertsch, D. DeWitt i N. Ellis, „Microsoft Azure SQL database telemetry,“ u *Proceedings of the Sixth ACM Symposium on Cloud Computing*, Kohala Coast, Hawaii, USA, 2015.

[10] J. Mališić, „Osnovni pojmovi matematičke statistike,“ u *Verovatnoća i matematička statistika*, Beograd, Krug, 1999, pp. 139-140.

[11] I. Bronštejn, K. Semendjajev, G. Musiol i H. Milig, „Matematička statistika,“ u *Matematički priručnik*, Beograd, SOHO graph, 2004, pp. 791-808.

[12] V. Dhar, „Data Science and Prediction,“ *Communications of the ACM*, t. 56, br. 12, pp. 64-73, 2016.

[13] E. Siegel, Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die, Wiley, 2013.

[14] C. Clifton, „Data mining,“ Britannica, 12 5 2009. [Na mreži]. Available: https://www.britannica.com/technology/data-mining. [Poslednji pristup 1 2 2021].

[15] L. Cao, P. Yu, C. Zhang i H. Zhang, Data Mining for Business Applications, New Delhi, India: Springer US, 2009.

[16] L. Cao, „Data Science: Chanllenges and Directions,“ *Communications of the ACM*, t. 60, br. 8, pp. 59-68, 2017..

[17] F. Lo, „What is data science?,“ DataJobs.com, 2017. [Na mreži]. Available: https://datajobs.com/what-is-data-science. [Poslednji pristup 10 1 2018].

[18] C. Hursch i J. Hursch, SQL, the structured query language, New York: TAB books, 1988.

[19] P. Bernstein, I. Cseri, N. Dani, N. Ellis, A. Kalhan, G. Kakivaya, D. Lomet, R. Manne, L. Novik i T. Talius, „Adapting Microsoft SQL server for cloud computing,“ u *Proceeding of the 27th International Conference on Data*, Hannover, Germany, 2011.

[20] S. Sivasubramanian, „Amazon dynamoDB: a seamlessly scalable non-relational database service,“ u *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, Scottsdale, Arizona, USA, 2012.

[21] F. Li, „Cloud-native database systems at Alibaba: opportunities and challenges,“ *Proceedings of the VLDB Endowment*, t. 12, br. 12, p. 2263–2272, 2019.

[22] C. Curino, E. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan i N. Zeldovich, „Relational Cloud: A Database-as-a-Service for the Cloud,“ u *5th Biennial Conference on Innovative Data Systems Research*, Asilomar, California, USA, 2011.

[23] D. V. Aken, A. Pavlo, G. J. Gordon i B. Zhang, „Automatic Database Management System Tuning Through Large-scale Machine Learning,“ u *Proceedings of the 2017 ACM International Conference on Management of Data*, Chicago, Illinois, USA, 2017.

[24] D. Dundjerski i M. Tomasevic, „Automatic database troubleshooting of Azure SQL Databases,“ *IEEE Transactions on Cloud Computing*, pp. 1-16, 2020.

[25] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu, F. Li, C. Chen i D. Pei, „Diagnosing root causes of intermittent slow queries in cloud databases,“ *Proceedings of the VLDB Endowment*, t. 13, br. 10, pp. 1176-1189, 2020.

[26] P. H. Callahan, „Expert systems for AT&T switched network maintenance,“ *AT&T Technical Journal,* t. 67, br. 1, pp. 93-103, 1988.

[27] V. Nair, A. Raul, S. Khanduja, V. Bahirwani, Q. Shao, S. Sellaminckam, S. Keerthi, S. Herbert i S. Dhulipalla, „Learning a Hierarchical Monitoring System for Detecting and Diagnosing Service Issues,“ u *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Sydney, Australia, 2015.

[28] SQL Sentry, „Sql Sentry preformance advisor,“ 2016. [Na mreži]. Available: http://www.sqlsentry.com/products/performance-advisor/sql-server-performance. [Poslednji pristup 2017].

[29] DELL software, „Spotlight on SQL Server Enterprise Edition,“ 2016. [Na mreži]. Available: https://software.dell.com/documents/spotlight-on-sql-server-enterprise-datasheet-67894.pdf. [Poslednji pristup 2016].

[30] Idera, „Idera SQL diagnostic manager,“ 2016. [Na mreži]. Available: https://www.idera.com/productssolutions/sqlserver/sqldiagnosticmanager/resources. [Poslednji pristup 2016].

[31] Idera, „SQL Doctor,“ IDERA, 21 3 2017. [Na mreži]. Available: https://www.idera.com.au/learn-more/sql-doctor/. [Poslednji pristup 13 5 2021].

[32] J. Popovic, „Automatic plan correction in SQL Server 2017 | SQL Server Database Engine Blog,“ Microsoft, 17 5 2017. [Na mreži]. Available: https://blogs.msdn.microsoft.com/sqlserverstorageengine/2017/05/17/automatic-plan-correction-in-sql-server-2017/. [Poslednji pristup 10 1 2018].

[33] O. Brent, „sp_AskBrent,“ 2016. [Na mreži]. Available: https://www.brentozar.com/askbrent/. [Poslednji pristup 2016].

[34] D. Y. Yoon i B. Mozafari, „DBSeer: pain-free database administration through workload intelligence,“ u *Proceedings of the 41st International Conference on Very Large Data Bases*, Kohala Coast, Hawaii, USA, 2015.

[35] D. Yoon, N. Niu i B. Mozafari, „DBSherlock: A Performance Diagnostic Tool for Transactional Databases,“ u *SIGMOD'16 International Conference on Management of Data*, New York, USA, 2016.

[36] B. Nevarez, „The Query Store,“ u *High Performance SQL Server: The Go Faster Book*, Santa Clarita, California, USA, APRESS, 2016, pp. 45-70.

[37] European Union, „General Data Protection Regulation,“ 25 5 2018. [Na mreži]. Available: https://gdpr-info.eu/. [Poslednji pristup 16 2 2021].

[38] Microsoft, „Database Advisor performance recommendations for Azure SQL Database,“ 3 10 2020. [Na mreži]. Available: https://docs.microsoft.com/en-us/azure/azure-sql/database/database-advisor-implement-performance-recommendations. [Poslednji pristup 16 2 2021].

[39] D. Dundjerski, S. Lazic, M. Tomasevic i D. Bojic, „Improving schema issue advisor in the

Azure SQL database,“ u *25th Telecommunications Forum TELFOR*, Belgrade, Serbia, 2017.

[40] Microsoft, „Azure SQL Database: The world's first intelligent cloud database service,“ 22 8 2017. [Na mreži]. Available: https://techcommunity.microsoft.com/t5/Microsoft-Ignite-Content-2017/Azure-SQL-Database-The-world-s-first-intelligent-cloud-database/td-p/98549. [Poslednji pristup 18 5 2018].

[41] Amazon RDS, „Monitoring Amazon RDS - Amazon Relational Database Service,“ 9 3 2018. [Na mreži]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-ug.pdf#CHAP_Monitoring. [Poslednji pristup 11 3 2018].

[42] Oracle, „Overview of Oracle Enterprise Manager Cloud Control 13c,“ 2016. [Na mreži]. Available: https://docs.oracle.com/cd/E63000_01/EMCON/overview.htm#EMCON109. [Poslednji pristup 2018 10 10].

[43] Oracle, „Automatic Performance Diagnostics,“ 2017. [Na mreži]. Available: https://docs.oracle.com/database/121/TGDBA/pfgrf_diag.htm#TGDBA026. [Poslednji pristup 2018 1 10].

[44] Oracle, „Automatic SQL tuning,“ 2018. [Na mreži]. Available: https://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm#CHDJDFGE. [Poslednji pristup 10 1 2018].

[45] T. E. Marquess, „STARKEEPER Network troubleshooter: an expert system product,“ *AT&T Technical Journal*, pp. 137-154, 1988.

[46] B. G. Buchanan i E. H. Shortliffe, Rule-Based Expert systems, Massachusetts, USA: Addison-Wesley, 1984.

[47] Q. Zhao, „Reasoning with Awareness and for Awareness,“ *IEEE Systems, Man, & Cybernetics magazine*, pp. 35-38, 2017.

[48] J. Moor, „The Dartmouth College Artificial Intelligence Conference: The Next Fifty years,“ *AI Magazine*, t. 27, br. 4, pp. 87-89, 2006.

[49] J. Thomson, „Killing, Letting Die, and the trolley problem,“ *Monist: An International Quarterly Journal of General Philosophical Inquiriy*, t. 59, pp. 204-217, 1976.

[50] McGeddon, „The trolley problem: should you pull the lever to divert the runaway trolley onto the side track?,“ 6 3 2018. [Na mreži]. Available: https://en.wikipedia.org/wiki/Trolley_problem#/media/File:Trolley_Problem.svg. [Poslednji pristup 20 3 2021].

[51] K. Soila, J. Tan, L. Hu, M. Kutare, M. Kasick, K. Schwan, P. Narasimhan i R. Gandhi, „Performance troubleshooting in data centers: an annotated bibliography,“ *ACM SIGOPS Operating Systems Review*, t. 47, br. 3, pp. 50-62, 2013.

[52] V. Jeyakumar, O. Madani, A. Parandehgheibi, A. Kulshreshtha, W. Zeng i N. Yadav, „ExplainIt! -- A declarative root-cause analysis engine for time series data,“ u *SIGMOD'19 International Conference on Management of Data*, Amsterdam, Holland, 2019.

[53] T. Raeder, B. Dalessandro i F. Provost, „Design principles of massive, robust prediction systems,“ u *18th ACM SIGKDD international conference on Knowledge discovery and data mining*, Beijing, China, 2012.

[54] X. Zhu, „Application performance management using learning, optimization, and control,“ u *5th ACM/SPEC international conference on Performance engineering*, Dublin, Ireland, 2014.

[55] P. Kanuparthy, D. Lee, W. Matthews, C. Dovrolis i S. Zarifzadeh, „Pythia: Detection, Localization, and Diagnosis of Performance Problems,“ *IEEE Communcations Magazine: Monitoring and Troubleshooting Multi-Domain Networks using Measurement Federations,* pp. 55-62, 2013.

[56] B. L. Welch, „The Generalization of ‘Students’ problem when several different population varlances are involved,“ *Biometrika,* t. 34, br. 1-2, pp. 28-35, 1947.

[57] D. Dundjerski, M. Pantic, I. Ilic, P. Lotrean, S. Lazic, C. J. Cunningham, V. Vasic, J. Cukalovic, V. Jovic i D. Petrovic, „AUTOMATIC DATABASE TROUBLESHOOTING“. USA Patent 15/817,181, 18 11 2017.

[58] Microsoft, „sys.dm_os_wait_stats (Transact-SQL),“ 10 2 2021. [Na mreži]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-wait-stats-transact-sql?view=sql-server-ver15#WaitTypes. [Poslednji pristup 20 3 2021].

[59] J. Rothenberg, J. Paul, I. Kameny, J. R. Kipps i M. Swenson, „Evaluating Expert System Tools: A Framework and Methodology,“ RAND Corporation, Santa Monica, California, 1987.

[60] P. Ganesh Kumar, C. Rani, D. Devaraj i T. Victoire, „Hybrid Ant Bee Algorithm for Fuzzy Expert System Based Sample Classification,“ *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* t. 11, br. 2, pp. 347-358, 2014.

[61] M. S. Hossain, S. Rahaman, A. Kor, K. Andersson i C. Pattinson, „A Belief Rule Based Expert System for Datacenter PUE Prediction under Uncertainty,“ *IEEE Transactions on sustainable computing*, t. 2, br. 2, pp. 140-151, 2017.

[62] M. Jacka i P. Keller, „RACI Matrices,“ u *Business process mapping : improving customer satisfaction*, Hoboken, USA, Wiley, 2009, pp. 257-258.

[63] R. Hightower, „RACIS matrices,“ u *Internal Controls Policies and Procedures*, Hoboken, USA, Wiley, 2008, p. 83.

[64] D. Baker, u *Multi-Company Project Management: Maximizing Business Results Through Strategic Collaboration*, Fort Lauderdale, Florida, USA, J. Ross Publishing, 2009, p. 58.

[65] Microsoft, „What is Azure Data Explorer?,“ 24 9 2018. [Na mreži]. Available: https://docs.microsoft.com/en-us/azure/data-explorer/data-explorer-overview. [Poslednji pristup 10 10 2018].

[66] Microsoft, „Ingest data from Event Hub into Azure Data Explorer,“ 13 8 2020. [Na mreži]. Available: https://docs.microsoft.com/en-us/azure/data-explorer/ingest-data-event-hub. [Poslednji pristup 20 8 2020].

[67] P. Mazumdar, S. Agarwal i A. Banerjee, „Business Continuity and Security with Azure SQL Database,“ u *Pro SQL Server on Microsoft Azure*, Berkeley, CA, USA, Apress, 2016, pp. 157-188.

[68] J. Kemnetz i C. Watson, „Overview of Azure Diagnostic Logs,“ Microsoft, 21 8 2017. [Na mreži]. Available: https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-overview-of-diagnostic-logs. [Poslednji pristup 10 1 2018].

[69] Microsoft, „Azure Event Hubs — A big data streaming platform and event ingestion service,“ 13 1 2021. [Na mreži]. Available: https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-about. [Poslednji pristup 17 2 2021].

[70] Microsoft, „Monitor database performance with Intelligent Insights - Azure SQL Database | Microsoft Docs,“ 6 12 2020. [Na mreži]. Available: https://docs.microsoft.com/en-us/azure/azure-sql/database/intelligent-insights-overview#set-up-with-azure-sql-analytics. [Poslednji pristup 18 2 2021].

[71] R. Kohavi i R. Longbotham, „Online Controlled Experiments and A/B Testing,“ u *Encyclopedia of Machine Learning and Data Mining*, Boston, MA, USA, Springer, 2017, pp. 922-928.

[72] THALES, „TopSky - ATC,“ 2021. [Na mreži]. Available: https://www.thalesgroup.com/en/topsky-atc. [Poslednji pristup 20 3 2021].

[73] M. Fowler, „BlueGreenDeployment,“ 1 3 2010. [Na mreži]. Available: https://martinfowler.com/bliki/BlueGreenDeployment.html. [Poslednji pristup 20 6 2020].

[74] Microsoft, „Memory-optimyed tempdb metadata,“ 30 10 2019. [Na mreži]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-database?view=sql-server-ver15#memory-optimized-tempdb-metadata. [Poslednji pristup 20 3 2021].

[75] L. Metcalf i W. Casey, „Introduction to data analysis - Log Transformation,“ u *Cybersecurity and Applied Mathematics*, Szngress, 2016, pp. 43-65.

[76] R. Shumway i D. Stoffer, „Time Series Regression and ARIMA Models,“ u *Time Series Analysis and Its Applications. Springer Texts in Statistics*, New York, Springer, 2000, pp. 89-212.

[77] D. Ferreira, M. Zacarias, M. Malheiros i P. Ferreira, „Approaching Process Mining with Sequence Clustering: Experiments and Findings,“ *Business Process Management - Lecture Notes in Computer Science*, t. 4714, pp. 360-374, 2007.

Spisak slika

Spisak tabela

Biografija autora

Dejan Dunđerski je rođen 10. avgusta 1987. godine u Beogradu. Osnovnu školu „Vladimir Ilič Lenjin“ završio je 2002. godine kao nosilac Vukove diplome. Nakon toga, upisao je Matematičku gimnaziju koju je završio 2006. godine.

Elektrotehnički fakultet u Beogradu upisao je 2006. godine. Diplomirao je 13.7.2010. godine na Odseku za Softversko inženjerstvo, sa prosečnom ocenom 9.71. Diplomski rad pod naslovom „Paralelizacija BGP protokola rutiranja korišćenjem grafičkih procesora“ izradio je pod mentorstvom prof. dr Mila Tomaševića i odbranio ga sa ocenom 10.

Master akademske studije upisao je 2010. godine na Elektrotehničkom fakultetu u Beogradu, na modulu Računarska tehnika i informatika. Studije je završio 12.3.2012. godine sa prosečnom ocenom 10.00, odbranom master rada „Paralelizacija protokola rutiranja u okviru softverskih rutera korišćenjem grafičkog procesora“ kod mentora prof. dr Mila Tomaševića.

Doktorske akademske studije upisao je u novembru 2014. godine na Elektrotehničkom fakultetu u Beogradu. Sve ispite predviđene studijskim programom položio je sa ocenom 10. U toku doktorskih studija objavio je dva rada u međunarodnim časopisima sa impak faktorom, jedan rad u domaćem časopisu i više radova na konferencijama. Tokom istraživačkog rada iskazao je interesovanje za oblasti paralelizacije, optimizacije procesa i inteligentne sisteme u klaud okruženju. Bio je recenzent više naučnih radova na domaćim konferencijama.

Prvi komercijalni ugovor, za aplikaciju za praćenje naleta aviona i pilota, Dejan je potpisao u drugom razredu srednje škole. Potom, tokom osnovnih studija je pomagao u porodičnoj firmi, te je tom prilikom bio zadužen za projektovanje i puštanje u rad informacionog sistema za praćenje i upravljanje proizvodnjom u fabrici aviona. Sa sestrama, ocem i prijateljem osnovao je *startup* kompaniju pod nazivom Sterna koja se bavi projektovanjem vrlo lake letelice sa idejom da se letenje demokratizuje i postane dostupno široj populaciji.

U isto vreme, od oktobra 2011. godine zaposlio se kao inženjer za automatske sisteme kontrole letenja u Agenciji za kontrolu letenja Srbije i Crne Gore. U julu 2012. godine, odlučio je da svoju karijeru nastavi u razvojnom centru kompanije Microsoft u Beogradu gde i danas radi kao softverski inženjer na Azure SQL platformi. Tokom rada u kompaniji Microsoft učestvovao je i vodio više velikih i značajnih projekata od kojih su najpoznatiji proizvod ***Inteligent insight*** na osnovu kog je napisana i ova doktorska disertacija i klaud platforma za velika preduzeća i javne ustanove koja obezbeđuje najviši nivo izolacije i sigurnosti pod nazivom ***Azure SQL Managed Instance***.

obrazac izjave o autorstvu

Izjava o autorstvu

Ime i prezime autora: Dejan Dunđerski

Broj indeksa: 5019/2014

Izjavljujem

da je doktorska disertacija pod naslovom

Sistem za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju

- rezultat sopstvenog istraživačkog rada;
- da disertacija u celini ni u delovima nije bila predložena za sticanje druge diplome prema studijskom programima drugih visokoškolskih ustanova;
- da su rezultati korektno navedeni i
- da nisam kršio autorska prava i koristio intelektualnu svojinu drugih lica.

Potpis autora

U Beogradu, 26.8.2021.

obrazac izjave o istovetnosti štampane i elektronske verzije doktorskog rada

Izjava o istovetnosti štampane i elektronske verzije doktorskog rada

Ime i prezime autora: Dejan Dunđerski

Broj indeksa: 5019/2014

Studijski program: Softverski inženjering

Naslova rada: Sistem za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju

Mentor: prof. dr Milo Tomašević

Izjavljujem da je štampana verzija mog doktorskog rada istovetna elektronskoj verziji koju sam predao radi pohranjivanja u Digitalnom repozitorijumu Univerziteta u Beogradu.

Dozvoljavam da se objave moji lični podaci vezani za dobijanje akademskog naziva doktora nauka, kao što su ime i prezime, godina i mesto rođenja i datum odbrane rada.

Ovi lični podaci mogu se objaviti na mrežnim stranicama digitalne biblioteke, u elektronskom katalogu i u publikacijama Univerziteta u Beogradu.

Potpis autora

U Beogradu, 26.8.2021.

obrazac izjave o korišćenju

Izjava o korišćenju

Ovlašćujem Univerzitetsku biblioteku „Svetozar Marković“ da u Digitalni repozitorijum Univerziteta u Beogradu unese moju doktorsku disertaciju pod naslovom:

Sistem za inteligentno otkrivanje uzroka problema u relacionim bazama podataka u klaud okruženju

koja je moje autorsko delo.

Disertaciju sa svim prilozima predao sam u elektronskom formatu pogodnom za trajno arhiviranje.

Moju doktorsku disertaciju pohranjenu u Digitalnom repozitorijumu Univerziteta u Beograd i dostupnu u otvorenom pristupu mogu da koriste svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice (Creative commons) za koju sam se odlučio.

1. Autorstvo (CC BY)
2. Autorstvo – nekomercijalno (CC BY-NC)
3. Autorstvo – nekomercijalno – bez prerada (CC BY-NC-ND)
4. Autorstvo – nekomercijalno -deliti pod istim uslovima (CC BY-NC-SA)
5. Autorstvo – bez prerada (CC BY-ND)
6. Autorstvo – deliti pod istim uslovima (CC BY)

Potpis autora

U Beogradu, 26.8.2021.

1. **Autorstvo.** Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence, čak i u komercijalne svrhe. Ovo je najslobodnija od svih licenci.
2. **Autorstvo – nekomercijalno.** Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca ne dozvoljava komercijalnu upotrebu dela.
3. **Autorstvo – nekomercijalno – bez prerada.** Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, bez promena, preoblikovanja ili upotrebe dela u svom delu, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca ne dozvoljava komercijalnu upotrebu dela. U odnosu na sve ostale licence, ovom licencom se ograničava najveći obim prava korišćenja dela.
4. **Autorstvo – nekomercijalno – deliti pod istim uslovima.** Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence i ako se prerada distribuira pod istom ili sličnom licencom. Ova licenca ne dozvoljava komercijalnu upotrebu dela i prerada.
5. **Autorstvo – bez prerada.** Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, bez promena, preoblikovanja ili upotrebe dela u svom delu, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca dozvoljava komercijalnu upotrebu dela.
6. **Autorstvo – deliti pod istim uslovima.** Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence i ako se prerada distribuira pod istom ili sličnom licencom. Ova licenca dozvoljava komercijalnu upotrebu dela i prerada. Slična je softverskim licencama, odnosno licencama otvorenog koda